CAPTAIN COSMO'S WHIZBANG

By Jeff Duntemann

For Me and You and the 1802!
WHAT IS THIS? It's a book, by cracky, about the 1802; hopefully the oddest and most entertaining book on any microprocessor ever written. The 1802 is, after all, an odd and entertaining chip. This view is not shared by all. Physicist Mike Brandl said he could swallow a mouthful of sand and barf up a better microprocessor than the 1802, and another colleague claims its instruction set demands that he program with his left hand. Bitch, bitch, bitch. I kinda like it. Much of this material I doped out while recovering from hernia surgery not long ago and couldn't lift anything heavier than a 40-pin DIP. I had a lot of fun and thought you might like to be copied in on it.

Like everything else I do, this book is an experiment. If I don't take a serious loss on production and mailing costs, I may do up another one. I've got a little gimcrack on the bench that'll make you people drool: an easy-to-build thermal printer for the 1802 that you can make for seventy bucks flat with all new parts. I'm working on an automatic phone dialer board and a few other things. Selectric interface. Robotics. Ham radio stuff. All kindsa things. Are you interested? Would you lay out another five beans for a Volume II? Let me know; drop me a note with any and all comments and spare not the spleen; I'm a hard man to offend and I love crackpot letters.

I might also consider buying small programs for publication if I find a market for this thing. Cartoons, too; I love cartoons; no book or magazine worth a bran muffin lacks cartoons. Whatever you do, send me copies if you're sending me things; never send originals. I can't promise to send them back. I will promise not to print anything of yours without making arrangements with you. I may be weird, but I'm honest.

ABOUT THE AUTHOR: Jeff Duntemann still has his appendix, tonsils, and all of his teeth except one which never grew in. He wears a size 8 medium bowling shoe and hates the living hell out of Disco. He messes with anything electronic, is nice to women, and goes berserk over a good pepperoni pizza. If you know what FIAWOL means you'll know where to find him every Labor Day and at other times during the year, and no, if you don't know he's not going to tell you.

(There are at least 2 1802's on Mars! Eat mundane death, Z80!!)

CAPTAIN COSMO'S WHIZBANG COPYRIGHT 1980 by Jeff Duntemann. All rights reserved. No, it is not all right for you to run off extra copies for your buddies. Be a sport. If it was easy you would have written it yourself, right? Right. Thanks. I do appreciate it.
Here a flip; There a flip; Everywhere a no-op...

It was a couple of years ago that I was riding the Clark St. bus with Gus, a friend of mine. He was reading the new Popular Electronics over my shoulder. We were looking at a picture of a piece of perboard with a bus on it, held in the man’s hands. It was supposed to be a computer. Gus saw my interest and shook his head. "No good," he said. "It multiplexes the address bus."

"Oh," I said, and turned the page. I knew of computers only as complicated digital devices that moved bits around in memory, added a lot, and generally ran on ideas I never understood. I had read snippets in various magazines about vectored interrupts, real-time clocks, dual display computer business, and so on. Nobody said much about what a computer actually did. And I'll never forget the incandescent blaze of understanding that roared out of my ears when I understood what a computer did with a series of instructions stored in memory.

**EPIPHANY!! A COMPUTER IS A BOX WHICH FOLLOWS A PLAN! (to steal a phrase from Ted Welch)"** I knew then that there was no limit to what one could do with a computer. I knew then that, given time, I could do everything.

"...uhhh...dollar on all the digit readouts." I shrugged and handed him a ten-spot. "How 'bout things that readouts that said HEXADECIMAL over them. No, never forget the incandescent blaze of understanding that roared out of my ears when I understood what a computer did with a series of instructions stored in memory.

"And if it didn't work, I could always make an ashtray out of it."

**HAMFESTS--Now, you may already have your ELF built. But if you haven't built it yet, or are still gathering parts, give a listen. My ELF cost me fifty bucks total. I saved a bundle just by not going to those unscrupulous dealers. I bought six LM380's for a dollar once--and got six burned out LM380's. All I knew about the dealer is that he was an unscrupulous asshole. May the [object redacted] fill his bed with stockerded IC's, pin side up. Beware. But by and large, hamfest material is all right.

"IT MAY OR MAY NOT COME IN THE MAIL--I'm constantly in touch with a large number of basement tinkerers who buy large quantities of supers electronics. What I'll do here is share some of their and my experiences with you. If a dealer isn't mentioned here, that only means No Data Available. Give 'em a try. It's a competitive field, and jerky dealers don't last long.

The finest of the surplus dealers is unquestionably Digi-Key. Low prices and superb service, quick and free, and if I've never gotten a faulty item from them in over 25 orders. Their handling fee is only 75 cents, and they back-order only rarely. I buy all my CMOS (except COSMAC IC's) from them. James is also good, and very fast shipping. Their prices are higher than they should be, however. I suggest using COSMAC IC's from Quest or Anacorna. Quest has a good record, and although their shipping is a little slow, their prices are good and I have never received a bad item. Advanced Computer Products has a tremendous selection of IC's and computer paraphernalia. Their catalog is a must-have. One of my friends got screwed over by them; but they have always gotten me good prices on the stuff he ordered. Ampl'Anny, is excellent but has a rather limited selection. Godbout has superior service but high prices. Jade is also good, but sometimes friends have gotten faulty IC's, but others have been satisfied. Integrated Circuits Unlimited has little computer stock, but their prices are low, scrawny, and quite unsuited to the hobby. All manufacturers, they have an excellent service. Poly Faks is a real screwball outfit. Reading their catalog takes some experience. When they say something has '100's of uses' it's probably only a few. "100 or more" means 100 or 250. LED products are dismal. They frequently describe various parts as "great for projects;" I suppose a lot of other people sprinkle them on their corn flakes. They are almost all overpriced. Resistors and caps, OK. The catalog is worth having because they do sell a lot of slim, all stuff unavailable elsewhere, but if you send you crap--send it back, and demand a refund. You'll get it.

UNDER NO CIRCUMSTANCES buy from ace, Edile's, TK Graphics, MiniMicroArt, Delta, ETCO, Formula International, or Trinico. These companies have screwed over people I know, and I'm trying to drive them into bankruptcy. Do your part. Send their books elsewhere.

CMOS RAM BARGAIN--New Time Electronics sells prime NEC 5101-3 CMOS RAM for $4.50. This is the RAM buy of the decade. Real CMOS that you can keep alive with a couple of hearing-aid batteries and a few capacitors for forever. They are plug-in with 2101's--you can use them interchangeably. Unless you're starving on welfare, spend the extra couple bucks and have an all-CMOS system. You won't regret it. On the TIL-311's, you're looking at 1.579 MHz and drew only 25 milliamps. Try and beat that!

**WIRING WOES--There is only one way for a basement tinkerer to build a computer. By wire-wrapping. Period. (Unless you buy a kit or readymade PC, that is.)** The idea is to design your own PC board is silly. You can't re-fry an egg, and you can't re-etch a PC board. Make one little mistake, and...you have to throw away your whole system. Wire-wrap is the most expeditious, quickest way to build a computer. Wire-wrap, then, is the only way to build a computer. If you can't re-fry an egg, you can't re-wrap a PC board. Start small, and get used to the idea. There are a few good guides out. I like the one by Steve Jobs. The guides are all worth the money, and the guides are all worth the money. But I think the best guide is your local librarian. The best guide is your local librarian.

So check out the hamfests. How to find them? Read the ham magazines QST, QEX, ARRL. There are often sold at computer stores and electronics stores. Each has a listing of hamfests for the coming months. Make up a shopping list, and go to it. You may occasionally get burned by unscrupulous dealers. I bought six LM380's for a dollar once--and got six burned out LM380's. All I knew about the dealer is that he was an unscrupulous asshole. May the [object redacted] fill his bed with stockerded IC's, pin side up. Beware. But by and large, hamfest material is all right.

Silver tarnish conducts electricity! (Failed Chem 101, huh? It shows.) Maybe it looks flimsy, and someday the chaos of a densely wrapped
board offends your prurient orderly sensibilities. Tough luck. You're a damned fool if you do it any other way.

WATCH OUT--A common mistake to new computer builders lies in toggle-switches. When I built my ELF, I wired my toggles upside-down, thinking that when the toggle handle is up, the center terminal is connected to the top terminal. Wrong-o. Think of it like this: The toggle handle points to the two terminals which are shorted together. Like in the diagram below. This problem was compounded for me by the fact that the I drove my TTL-111 hex display chips with inverting CMOS drivers, CD4097's. (Sheer stupidity on my part.) The toggles were inverting the binary bits they were feeding the computer, and the 4097's were re-inverting those bits for my eyes, so that while everything looked normal, nothing ran right. Hence I contracted my first case of The-Chip-Is-Bad Fever, when nothing can possibly be wrong except for a bad CPU. Don't you believe it. More on TCIE Fever later on.

GIVE YOURSELF ROOM--If you haven't already started your ELF, for chrissake get a bigger piece of perfboard. There's nothing sacred about the parts layout shown in the EE series, except possibly to keep the crystal close to pins 1 and 30. The other stuff you can put anywhere it fits, especially if you wire-wrap. 5 1/2 X 8 is a good size. Don't let all the empty space bother you. There's plenty of things you can add or later to fill things up.

RAM CHOICES--Whatever you do, don't use the RAM expansion circuit given in EE March '77. It's hopelessly awkward and needlessly time-consuming. Remember, that was 3 years ago. Today we have the 2114 1024 X 4 static RAM for eight bucks. Two of them will give you 1K RAM if you include a CD4042 latch to save address lines 8 & 9. If you haven't started your ELF, use the circuit in Fig. 2 instead of 2101 RAM. If yours is built, I think it would be worth ripping the 2101 sockets out and replacing them with a pair of 2114's. (Which are smaller, physically, than 2101's!) Wow you see the merits of wire-wrapping!

1 K - The easy way!

GROUND ALL UNUSED CMOS INPUTS ON 4042, 4081

Fig 1

POWER SUPPLY POINTERS--It's silly to scrap on power supply capacity. An L630K-5 costs you a buck at most, and will supply an amp and a half if properly heat-sunk. Use the one in the big power-transistor TO-3 case. It will permit your ELF to support a lot of accessories and peripherals, like an ASCII keyboard and paper tape reader, as well as lots more RAM. Another idea is to use a 6 volt motorcycle battery instead of an AC supply, and just recharge the battery every few weeks. I have done this to no ill effect on my ELF, even with TTL chips on the board. Also, your computer will then be portable. Just don't get battery acid in the wiring. Four AA ni-cads make a dandy ELF supply. If you plan to add a lot of accessories, use four C ni-cads. This will give you a computer you can throw in your briefcase to while away the hours on long bus, train, and car rides. (Using it on a plane might be frowned upon. Especially if the captain hears your switching transients on his radio/locution unit.)

COSMAC I/O PORTS--I suspect very few people have used the CDP1852 COSMAC I/O port because of its steep $9.50 price. It's a good unit, and now available from Anacrona for $1.85 in the new 'consumer plastic' version. (More on that further down.) It performs the function of either of the CD4508 ports shown on page 35 of EE 9/76, with the advantage that all those NAND gates are included on the chip. Also, there is a CLEAR input which will set latch contents to zero. When CLEAR goes low, the data registers are cleared.

The diagram should tell the whole story. You can use either the input port or output port separately; they don't depend on one another for operation.

8-Bit Input/Output Port

Fig 2

2114

Fig 3
If you are very sure you will not be needing more than 3000 ports and 3 output ports, you can do away with the 4028 and simply connect one of the N lines to the two CS2 pins. Since a 50 cent IC more than doubles your I/O capacity to 7 of each kind of port, you're silly not to use the 4028. The asterisked connection is up to you: you choose which pair of I/O opcodes you want to control the ports, look up the binary code presented by the N lines for that pair of opcodes, and then connect the CS2's to the 4028 output line which decodes that binary code. That same 4028 can handle all your other I/O decoding as well; it is not limited to the pair of ports shown here. Remember to ground pin 11 of the 4028!

OP-80 Bootstrap Loader

MA
00 90 FB 10 AF Set RF to 9101
05 3F 10 Jump to pgm unless INP
07 E0 07 Wait for RDX
07 EF Point X to RQ
0A 6B Input firm reader to H (RF)
0B 64 Display byte & inc. RF
0C 7B 7A Generate reader reset pulse
0E 30 07 Wait for next RDX

Interface for the OP-80A Paper Tape Reader

<table>
<thead>
<tr>
<th>Pin 9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>D9</td>
<td>D10</td>
</tr>
<tr>
<td>D8</td>
<td>D9</td>
</tr>
<tr>
<td>D7</td>
<td>D8</td>
</tr>
<tr>
<td>D6</td>
<td>D7</td>
</tr>
<tr>
<td>D5</td>
<td>D6</td>
</tr>
<tr>
<td>D4</td>
<td>D5</td>
</tr>
<tr>
<td>D3</td>
<td>D4</td>
</tr>
<tr>
<td>D2</td>
<td>D3</td>
</tr>
<tr>
<td>D1</td>
<td>D2</td>
</tr>
<tr>
<td>D0</td>
<td>D1</td>
</tr>
</tbody>
</table>

* PINS 9,10,11,12,13 ARE TIED TOGETHER FROM BOTH CHIPS

VERSION, VOLTAGE, & SPEED—There is no difference whatsoever in the way RCA manufactures the CDP1802CD and the CDP1802D. It's the same chip. Individual differences in overall circuit leakage determine how high a voltage each chip will run at without drawing excessive current and overheating. You have to remember that voltage and voltage alone determines how fast a CMOS chip will run. To make an 1802 run at five megas, you have to crank the voltage up to ten volts. If you crank a 'CD' version chip up, even assuming a 5V supply, it will probably fry in its own leakage dissipation. Ergo, it takes a 'D' version to run at five megas, only because its leakage is low enough to handle a ten volt supply. RCA tests each 1802 as it comes off the line with an ammeter in series with the supply. As the supply goes up, the current draw will go up, and somewhere along the way each chip "redlines." The chips that redline around seven volts are stamped 'CD'; those that make it all the way to fifteen volts become 'D's.' Like Gus Flassig used to say, everybody makes 5% resistors; those that don't make the grade get silver bands or no bands at all.

I say this because I've heard of people paying for the premium 'D' chips and thinking they can put a six meg crystal in their ELF and run super speeds at 5 volts. Won't work, as a matter of fact, even assuming a 5V supply, when your clock speed goes much above 2 megas, two things begin to happen: First, the internal crystal oscillator may not start, and once started, may run erratically. You can check this yourself with a good scope; up past 2 meg your square wave is anything but square. Secondly (and more seriously), at higher speeds your TPA pulse starts running late. Eventually it will occur during the time the address lines switch from high address to low address, and your address decoding system will start decoding indeterminate addresses. In other words, your computer goes berserk. This will happen no matter how good a chip you have. To run fast, you have to crank up the supply voltage. There's just no way around it. So if you do up the supply, make damned sure your other chips can handle it! (TIL311's ain't cheap!!!)
MA OPCDE

10 FB 00 B2 B3 B5 B6 Initialize high order registers
16 FB FF AC Initialize stack pointer
19 FB 23 A3 Initialize MAIN PC
1C FB 4E A5 Initialize BSUB PC
1F FC 5E A6 Initialize HSUB PC
22 D3 P=0
23 D5 B1 D5 A1 Enter starting address from keyboard to R1
27 6C Input toggles to stack:
29 B3 33 If toggles are not 00, go to M(33)
2A FB 2E A0 Repaint RO
2D D0 P=0
2E 61 B3 B1 A3 Transfer R1 into R3
33 D0 P=0 (jump to execute program at M(R(3)))
39 F6 Shift D one bit right
34 3B 38 If DF=0, go to M(3B)
38 D5 Call BSUB
37 E1 X=1
3B 64 Display M(R(1)) and increment R1
39 30 36 Loop again and display next byte
3B F6 Shift D one bit right
3C 3B 44 If DF=0, go to M(44)
3E D5 Call BSUB
3F E1 X=1
40 B1 Store D at M(R(1))
41 64 Display M(R(1)) and increment R1
42 30 3E Loop again and store next byte
44 3E 44 Wait for 4DA
46 E1 X=1
47 6B Input byte from tape reader to M(R(1))
4B 64 Display inputted byte and increment R1
4C 4A 4A Generate reader reset pulse
4B 30 44 Loop again and input next tape byte

4D D3 BSUB Return from BSUB
4E DA Call BSUB
5F FE FE FE FE Shift D four bits left
53 A0 Store D in R0.0
54 D6 Call BSUB
55 B0 Fetch contents of RO.0
56 F1 M(R(2)) OR D, put in D
57 E1 Store D at M(R(2))
58 64 22 Display stack and decrement stack pointer
5A 30 4D Go to Return

5C FC 05 Return from HSUB
5E E2 X=2
5F FC 01 Add D+1, put in D
61 FA 0F D AND OF, put in D
63 E2 Store D at M(R(2))
64 62 22 Output byte to CD4515 scanner chip: dec. R2
6B 6D B1 Output low byte to CD4515 scanner chip: dec. R2
69 7B Turn G on
69 FB 09 B4 Load debounce timing constant
6C 24 94 3A 6C Decrement constant and test for done
70 7A Turn G off
71 3F 71 Wait for key released
73 30 5C Go to return

HOW TO USE THIS PROGRAM:

This op system is actually an adaptation of the EHOPS-256 op system given in the original ELF series in PE. It requires the scanning hex keyboard described in the second article, without any modification. It also requires an input port strobed by the 68 INPUT instruction.

Operation is very similar to that of EHOPS-256, with the important exception that a two byte starting address must be entered from the hex keyboard. The toggle commands for write, inspect, and run are identical to EHOPS-256. 02 for write, 01 for inspect, and 00 for run.

04 is an additional command which will cause the program to input a string of data from an OP-80 paper tape reader to any location in memory with the first byte input to the starting address. To input from tape, position the tape on the reader so that the first byte you wish to input is immediately to the left of the phototransistor strip. Make sure the sprocket hole sensor is covered by tape and that the SP lamp is off. Set 04 into the toggles and flip RUN up Enter the two-byte starting address with the lower-order byte first. Then pull the tape through the reader, taking care to keep the tape flat over the phototransistor strip. As the tape moves through the reader, the data being input will appear on the hex displays.

Note that in order to run through this op system will use R3 as its main program counter. Also, R5 and R6 cannot be used in your programs if at any point in the program you intend to call BSUB. However, BSUB may be called any time hex input is desired if those registers are respected.
HOW TO USE THIS PROGRAM:

This is an operating system which will handle up to 65K of RAM. It requires an OP-80 paper tape reader interfaced through an input port strobed on the 6B INP instruction. To write in memory, set 02 in the toggle switches before running the program. Flip RUN up. Enter the higher-order byte of the address at which you wish to begin writing. Push INPUT; G will come on. Enter the lower-order byte of the address, and push INPUT again. Your address is set. Begin writing at that location by entering a byte in the toggles and pushing INPUT. Each byte will be stored at the next highest memory location.

To sequentially inspect memory, set 01 in the toggle switches. Flip RUN up. Enter the starting address as above. Once your starting address has been set, push INPUT. The contents of memory at your starting address will be displayed. Pushing INPUT will display successive memory locations. The byte set into the toggles while displaying is not important.

To input a program or data string from paper tape through the OP-80 reader, position the paper tape on the reader so that the first punched byte you wish to input is immediately to the left of the reader's phototransistor strip. Be sure that the sensor spot which reads the sprocket holes is covered by paper between holes, and that the SP lamp is off. (Be sure, though, that the lamp lights when the sensor spot is exposed to light.) When the tape is in position, set 04 into the toggles, and flip RUN up. Enter the starting address of the program or data string as above. Then pull the tape through the reader, taking care to keep the tape absolutely flat over the phototransistor strip. When you see that the punched portion of the tape has been pulled completely past the phototransistor strip, stop pulling and halt the computer. Pulling past the end of the significant data will load 00's in memory, possibly over other programs or data.

To run a program which has already been stored in memory, set 00 into the toggles and flip RUN up. Input the starting address as above. The program will begin running immediately after entry of the lower-order address byte. R3 will be the program counter, regardless of how the program was written. Be careful if you use subroutines that your routines return to MAIN program counter R3. Returning to other registers as MAIN program counter may lead to halting, endless loops, tapeworms, and other bizarre computer behaviour.

If you load a program from paper tape and it will not run properly, load it again, perhaps a little more slowly, before assuming something is wrong. Then, if you have no success, check to see (by inspecting memory) whether the first tape byte was actually loaded. If you positioned the tape so that the SP lamp was on when you hit RUN, the program will skip the first tape byte. Move the tape a hair to the left and load again.

THE 1802 MASS STORAGE PROBLEM—I don't really recommend paper tape, but if you have a Teletype with a punch, what the hell. Unless you stomp on it, the stuff is pretty well indestructible, and skilled hands reads as reliably as cassette. I have had difficulties with cassette machines that make me distrust them as well, but they're pretty much what we have. You people who also have a "big" machine with floppy disk capability ought to consider some sort of program for storing 1802 programs on disk, and reading them into the 1802 through a parallel port. I am currently working out the means to feed an eighty-dollar ELF from a $4000 $100 280 mainframe system. Is this silly? Well, that depends. Both are computers, regardless of cost. And while I can't expect my ELF to compute Fourier transforms, I also can't expect my mainframe to hide inside Cosmo the robot. Viva la difference!
HOW MUCH RAM?--A lot of hardware oriented people go ape-shit building their first computer, particularly a computer like the ELF which has no assembler available for it. They go to great lengths adding 4K RAM and then spend all their time diddling with short programs that never make it out of the first memory page. Be realistic. If this is your first computer, you damned well won't need more than 1K for quite awhile. With 2114's as cheap now as 2101's were in 1976, you may well start and build your ELF using the circuit below. It requires no complex memory-decoding systems, and will give you MORE than enough room to get hopelessly lost in.

HOW FAST RAM?--Another good question. An 1802 newcomer generally notices that COSMAC's clock runs fast. Up to 6.4 Mhz, in fact, using the D version at 10V. The machine should really rip, huh? Well, there's a joke in it. Each COSMAC machine cycle takes eight clock cycles. And each instruction takes at least two machine cycles. The 250, on the other hand, needs no more than 5 cycles to execute an instruction, with a couple of exceptions. Lay that against the 1802's minimum 16, and you'll see that the 1802 is way behind in the up speed sweepstakes. Maximum allowable 1802 memory access time is 4.5 clock cycles. The table below translates this figure into a maximum memory access time in microseconds for various clock speeds.

<table>
<thead>
<tr>
<th>Clock Speed</th>
<th>1 Clock</th>
<th>4.5 Clock</th>
<th>Use At Least</th>
</tr>
</thead>
<tbody>
<tr>
<td>875 Khz</td>
<td>1.14 us</td>
<td>5.13 us</td>
<td>2.5 us</td>
</tr>
<tr>
<td>1 Mhz</td>
<td>1.0 us</td>
<td>4.5 us</td>
<td>2.0 us</td>
</tr>
<tr>
<td>1.5 Mhz</td>
<td>.67 us</td>
<td>3.0 us</td>
<td>1.5 us</td>
</tr>
<tr>
<td>1.7604 Mhz</td>
<td>.57 us</td>
<td>2.56 us</td>
<td>1.5 us</td>
</tr>
<tr>
<td>2.0 Mhz</td>
<td>.5 us</td>
<td>2.25 us</td>
<td>1.0 us</td>
</tr>
<tr>
<td>2.5 Mhz</td>
<td>.4 us</td>
<td>1.8 us</td>
<td>500 ns</td>
</tr>
<tr>
<td>3.579 Mhz</td>
<td>.28 us</td>
<td>1.26 us</td>
<td>500 ns</td>
</tr>
<tr>
<td>4.0 Mhz</td>
<td>.25 us</td>
<td>1.125 us</td>
<td>650 ns</td>
</tr>
<tr>
<td>5.0 Mhz</td>
<td>.20 us</td>
<td>.9 us</td>
<td>450 ns</td>
</tr>
<tr>
<td>6.0 Mhz</td>
<td>.17 us</td>
<td>.75 us</td>
<td>350 ns</td>
</tr>
</tbody>
</table>

One thing you do have to remember in figuring memory access time is to add in the propagation delays through any latches or gates which lie between the address bus and the memory address inputs. Time through a 4042 latch is about 150 ns max, with the newer latches quite a bit faster than those manufactured more than five years ago. Propagation delay through simple NOR/NAND/gates is almost always less than 40 ns and can usually be disregarded. Access time on most static RAM these days is less than 450 ns; darned few take longer than 860 ns. This gives you a lot of elbow room, so unless you're going to a complex, large RAM system with a lot of decoding, buy the slowest, cheapest RAM you can find and don't worry about it.

WHEREZAT PROGRAM AT7?--Let's say you've carefully toggled in that 256 byte wonder you stayed up all last night writing, flip the infamous RUN switch up, and then---What? Nothing happens. What is that beast doing? That's pretty simple. It either stopped dead in its tracks somewhere (usually by hitting a 00 byte) or it's chasing its tail in a blind loop with no exit. Swell. But how do you know which, and where? Easy. Build a memory address display. The circuit is given below.

This circuit displays the instruction the computer is fetching at any particular time. If you run a program and the displays read 00 02 with no ambiguity about any of the numbers, then your program has survived a halt having executed the instruction at 00 02 last. What's more likely is that the numbers on the display will look funny. This means that the computer is looping, and it is displaying a series of addresses (addresses of all instructions within the loop) so quickly that your eyes are laying the numbers on top of one another. The way out of this one is to flip run and down quickly, and note down the resultant address display. When you flip RUN down you stop a program in its tracks, with the display giving you the instruction it was fetching when you hit the switch. If your loop is long enough, the laws of chance say you will eventually see every address within the errant loop. If one or more addresses seen to come up a great deal more frequently than the others, you may well have a loop within a loop. Keeping track of these stopping addresses will eventually tell you exactly where the computer is wandering. How you
fix it is your problem. But an address display is one of the simplest and most powerful debugging aids you can add to an ELF. The hex displays are expensive as hell, sure (unless you went to the same hamfest I did) but you will get your money's worth.

Note that in a minimal 256 byte ELF you only need the first two displays, DIS0 and DIS1. Three displays will take you up to 4K, and four will give you all 65K of memory space. So take heart; 4 TIL's are not necessary to make the display work. But why not wire all four sockets, even if you won't pop for the displays right away? Leaving an empty socket won't hurt anything, and you never know when you'll run across a guy selling the displays for a buck apiece.

This circuit also illustrates one use of the two state code lines, SC0 and SC1. These lines are used to tell the hardware what sort of machine cycle the computer is currently going through. They are also useful in interrupt and DMA processing. The Pixie graphics chip uses them to help generate the video sync pulses, for example. A careful look at the SC0/SC1 timing diagram can help a lot if you're trying to use interrupts or DMA I/O. They're the only way to unambiguously tell what sort of machine cycle is going on at any particular time.

COSMAC PRICE BREAKTHROUGH--In the year since I started putting this booklet together, RCA and Hughes have released plug-in packages containing versions of several chips in the COSMAC family. At this time I have only seen them available from Anaconda, but they should show up elsewhere at any time. The 1802 in a thirteen buck, and the 1852 I/O port is only $1.85! Now there is no longer any excuse not to use the 1802 family with the 1802 CPU. RCA's application notes on designing with the family as easy as piling building blocks atop one another. Good show.

STUDIO II CONVERSION--Another weird recent happening was Radio Shack's buyup of all RCA's remaining Studio II TV games. Just before Christmas 1978 every store was full of them, for $50. After Christmas they were all gone. I'll bet a box of TIL-311's that if you haunt the local garage sales, you'll eventually find a perfectly good Studio II for ten or fifteen bucks, once the family got bored with its rather slow, rather tame B&W TV games. An outfit called ARESCO (address in the back) puts out a conversion package for $5, which enables you to build a board which plugs into the game cartridge slot, enabling the Studio II to function as an ELF-type computer. I've seen it; I warrant you it's worth the money if you have one of these items. The TIL's and keypad chips are super, and the RAM is CMOS CD2P1822, which is a bloody expensive chip, plastic or otherwise. A full schematic and lots of other goodies are available from ARESCO. First class.

COSMAC'S FATAL FLAW--Those of you who haven't built an ELF yet ought to consider this; those who have probably know it already: COSMAC video graphics are alllilililowwwww. This is a real head-scratcher, considering that the head-muscle RCA has at its command: Why tie the clock of an already slow (at 6 meg) machine down to a paltry 1.76 Mhz just for the sake of a video display generator? The 1861 uses TPA and TP8 to generate sync pulses. Certainly, by adding a few flip-flops, they could have doubled (at least) the permissible clock speed to color burst. (Those nifty cheap 3.579 crystals, TV surplus) They lost their shirts on the Studio II because the games were, to be charitable, ponderous. One possible solution presents itself, now that COSMAC chips are so cheap: Build an ELF with two CPU's; one for processing at 4 megs or higher, and the other, which can be plastic, only handling display output to the TV screen. The hooker in this is setting up a screen buffer which both CPU's can access without fighting over the data bus or control lines. Otherwise it's no big deal for somebody with a little time and some RCA data sheets. Who's gonna do it?? (Added note: I recently learned that RCA had done exactly that; designed a computer to compete with the TRS-80, using two 1802's in such a configuration. Price was to be $500. It was called RECOMP-I, and it was pulled from the market at the last minute for mysterious reasons. Sunuvibitch. Those guys just can't seem to get their heads together about a real home computer using the 1802!!)

A CRITICAL LOOK AT THE COSMAC VIP--In my continuing effort to spend myself bankrupt, I bought a VIP, assembled, from Advanced Computer Products. I gotta say this for the place: seven days after the check was dropped in the mail, the computer landed on my doorstep. It's a purty machine, and the built-in keypad sure beats the bluejeans off of toggle switches. It's not a hell of a lot like the ELF, really; there's no operating system on ROM located at hex 8000, and run with some hocus-pocus using control lines which I don't really like. Still, it's a very good op system, particularly for the cassette interface. VIP means Video Interface Processor, and that is just what it is: everything the VIP does is tied to the 1861 display system. This eliminates any need for LED displays, with a couple of exceptions, but it ties you down to that silly-ass 1.764 Mhz clock.

The VIP uses 2114 RAM, which enables you to get up to 4K pretty easily and cheaply. It's shipped with only 256, but that's plenty for awhile. One gripe I have is that the Q line is tied internally to a little electromechanical honker. This honker is irritatingly squeaky and draws me nuts until I installed a little toggle switch on the board-cover to turn it on or off as desired. There are two 22/44 pin edge connectors on top of the VIP, enabling you to plug in either RCA's own accessory boards, or else boards you wrap yourself on Vector cards. RCA's boards are rather expensive, though they are a pretty royal blue. The only thing now is the Super Sound board, which has its own little language, PN-8. That means Play It Now. I'd say Play It Eventually was more like it; coding music into the tables takes a little practice and quite a bit of time. For all that, it's worth it. Sound is good, and after only two weeks of suffering I got it to play Bach's "little" fugue in G minor, which drew applause from my wife. Super Sound uses the interrupt line, so you can't run the display simultaneously. Shame. I had the worst urge to animate an image of Bach turning over in his grave.
For all its flaws, the VIP is probably worth the money. They have an EPROM programmer for it which I intend to get eventually; intelligent EPROM burners cost a lot more than VIPs. The worst thing about the VIP is something that can be said of the ELF-II from Netronics or Quest's Super ELF. If you don't wire wrap it yourself, you won't learn as much. What are you doing this for? If you want to learn microcomputer hardware and software without going broke, the Popular Electronics ELF has no equal. If you just want to get into software, one of the others is just as good, and may be better. At this point the worst thing about the ELF-II is that their PROMs are an 802, and have lots of hardware add-ons. RCA seems to ignore the fact that their VIP could, in fact, be a lot more than just a game-hacker. I look upon it as a great potential COSMAC development system, and that is what I will continue to use it for.

AN EPROM CARD FOR THE VIP--The 2716 EPROM is a nasty exception to our beloved law that big-money ICs grow cheaper as time passes. The damned things still cost sixty bucks--when they're in stock. You can find the triple-supply version for forty bucks here and there, but the 802 is a single-supply machine, and I gag on building two whole power supplies for the benefit of a single IC. One solution is to take a look at the 2758 EPROM. This is a single-supply version of the 2716, arranged as 1024 x 8 bits, which is to say plenty of room. How many 1K programs have you written recently?!

The circuit below will plug directly into the expansion socket of the VIP. It will work just as well with an ELF, except that you have to give the ELF a way to disable RAM memory. I'll explain that shortly.

This is a good example of a relocatable ROM block, and about as simple a one as you'll find, so this might be a good time to give you folks who haven't met such a beast yet a little tutorial on how they work. The 4042 latch saves memory address bits A8 through A11 at TPA time. A8 & A9 are applied to the 2758's A8 & A9 inputs, and are actually used to select bytes on the chip. (Those two taken with A0 through A7 make ten address lines, and 2 to the 10th power is 1024.) A10 & A11, on the other hand, are used to indicate what

**ELF / VIP EPROM Card**

![Diagram of ELF / VIP EPROM Card](image)

**2758**

8K (1K x 8) UV ERASABLE PROM

**PIN CONFIGURATION**

- **PIN NAMES**
  - A0: Address Line 0
  - A1: Address Line 1
  - A2: Address Line 2
  - A3: Address Line 3
  - A4: Address Line 4
  - A5: Address Line 5
  - A6: Address Line 6
  - A7: Address Line 7
  - A8: Address Line 8
  - A9: Address Line 9
  - A10: Address Line 10
  - A11: Address Line 11

**MODE SELECTION**

- **INPUTS**
  - Vcc
  - GND
  - RD
  - WR
  - CE

- **PROGRAM SELECT**
  - Program
  - Program Erase

- **OUTPUTS**
  - O0
  - O1
  - O2

- **OPERATION**
  - Write
  - Read

**ACTUATE ONLY 1 SET OF SW-1 CONTACTS AT A TIME!**

VIP 2758 EPROM CARD 220/79
single 1K block of memory the 2758 represents. In this circuit, you can set up SW-1 to "locate" the 2758 in any of the 1K blocks in the first 4K of 1802 memory space.

When the 2758's CS line is low, the chip is selected and can be addressed to fetch a byte from inside. If CS is high, the chip will not respond to any other control signals. The D0-D7 outputs represent an input impedance tri-state, which is good as having the chip "cut out" of the memory system. The 2758 must remain in this state until the 1802 sends out the right 2-bit pattern on A10-A11. The 4556 is a 2-line to 4-line decoder. At any given time one of the 4 output lines is low. The others remain high. SW-1 allows one of the output lines to address that necessary output signal to the CS input. One of those output lines corresponds to one of the four possible combinations of high-low states on A10-A11. For example, when A10 & A11 are both low (0), output 0 (pin 4) of the 4556 goes low. If SW-1 contacts A are closed, the 2758 will be selected, and will respond to memory addresses 0000-03FF. Each of the four SW-1 contacts places the 2758 in a different range of memory, indicated by the table beside the circuit.

So much for selecting the 2758. In order to avoid a "fight" between the 2758's output lines and the output lines of the rest of the system, you must simultaneously disable the rest of the memory system. The VIP conveniently gives you a line called MINH. If this line is brought high, VIP internal memory is deselected. Some people like to do this by selecting and deselecting with the same signal, but the polarities required are not the same. Therefore, you have to use a signal that selects the 2758 before it will deselect internal memory. I was tricky and avoided putting another chip on the board by using the second half of the 4556 as an inverter. If you don't try to explain how this is done, but if you have a full 4556 data sheet you can follow it along from the truth table. It would be good practice, so do it anyway.

The way this decoding scheme works, the delays in the 4556 allow both the 2758 and internal memory to be selected at once. A fight? No, because the 2758 has another pin, OE (output enable) which must be brought low before the 2758's outputs come down from their high-impedance limbo. The 1802's signal MRD does this at the proper time, and everyone is happy.

If you have an ELF wired as in the PE article, hooking this card to it is even simpler. Connect both the 2101's pin 17a together, but separate them from the 5V supply. Now connect the line directly to the 2758's and to both the 2101's pin 17's, and you're home free. You don't even need to use the other half of the 4556 as an inverter, because the 2101's have 2 select inputs, neither of which are used in the PE circuit. If pins 17 go low, the 2101's are deselected. Or, if the pins are brought high, the same thing happens. Use pin 17 allows us to use the same signal which selects the 2758 to deselect the 2101's.

I've gone on at length here so beginners can get a feel for memory system juggling. The idea is to have only one set of outputs selected at any one time. You start with the required select polarity on a memory chip's chip select pin, and work back toward the CPU and whatever logic it takes to present the chip select pin with the state it needs to turn that chip on when its address comes up on the address bus.

If you design memory systems like this. If two sets of outputs get selected at once, you have a "fight" which results in overheating chips and possibly ruin them. They may take the 1802 with them to CMOS heaven, so it could be an expensive mistake. If ever you run a memory system for the first time and it seems to "lock up" (another good reason to use address display) cut power immediately and take another long, hard look at your circuit diagram.

PROGRAMMING EPROMS--No, I don't have a magic circuit built and debugged to program your 2758's for you. I get them programmed at work, and thus in my lazy way never pursued the issue. You have to admit, a $12,000 industrial EPROM burner sure beats the pants off a perfboard kluge! A $100 add-on board is available from the VIP which will do the job. I've not seen the circuit, so I can't say if it will work with an ELF. If anyone knows, by all means tell me; I've been waffling as to whether to order that board. The algorithm for programming 2758's is simple, however. I won't repeat it here; get the Intel data book and read up on it. My guess is that you would use a 555 as a 50 millisecond one shot to "stretch" a machine cycle to 50 milliseconds by bringing the WAIT line low with the desired byte on the data bus and selected on the address bus. If you hold data on the pins for 50 ms while applying a pulse to a program pin and 25 volts to another pin, the byte will be stored. There's more to it than that, but if you ever get it working I'd love to hear about it. Nuff said for now.

COSMO'S FACE--I take that back; there is something that the VIP is very good at: Giving my robot a face. For awhile I've been tinkering with a clanking heap of surplus submarine parts and wheelchair motors named Cosmo Klein. The Klein is an obscure mathematical allusion to the Klein Bottle, whose insides are identical to its outsides. Cosmo is a little like that, especially when he tips over and sends his insides spilling out onto the floor. Well, I got the notion that a CMOS-generated face would be a marvelously humanizing touch. And so it is. If you want to see a good color picture of Cosmo and my VIP (with my own idiotically grinning mug in the background) check out Look Magazine dated April 30, 1979; it's the one with Jane Fonda on the cover. Maybe your library has it. The program which generates the face is included in this book, so I won't describe it here. Though you can't see it, my ELF is also inside, vainly trying to keep the monster from falling on his face. A CMOS robot is an old dream of mine, and I'm working on it, but for now I must pronounce his control circuitry (save for his face) a failure. Now you know who Captain Cosmo is. Yes, indeed, that cute cartoon on the cover has a real model. Maybe in a future issue I'll have a little more to say about his inner workings. Once they work.

Cosmo the Robot with his access plates removed, so that you can see the mess inside. An ELF is hiding above and behind the VIP. On the right is Cosmo's arm. Not long after Bill Colsher took this photo, Cosmo tried to squeeze a Schlitz and stripped out his finger drive motor, proving that it takes a robot to crush his bare hand with a beer can.
ZOUNDS!—Jim Lisowski gave me a marvelous little sound-effects routine which he wrote for his ELF-II; however, it will run as given on a basic ELF as well, as long as either machine has a single-page configuration. Jim did not initialize the high-order register halves, so although the program might run on an expanded-memory system, it might not. Initialization of high-order registers was no problem; the only hassle is pushing all the rest of the program down a few bytes with all the breathing addresses correct. Really, you ELF hackers ought to be able to do that in your sleep, so I leave it to you.

Using it is easy. When entering the program into your ELF, decide which delay routine you want to use, and key it in. The delay routine must end with a branch to 00, or the program will get hopelessly lost. When you run the program toggle (or key on the ELF-II) bytes to change the nature of the sound effects.

This requires that the Q line be audio amplified to provide the sound. You can plug it into your stereo system (use the high-impedance phone input) or, better still, wire up a little audio amp right on the ELF board. What I have is a little vector plugboard with an amp and speaker right on it. I anticipated VIP Simple Sound was going to happen after RCA announced it, but then again, the concept is pretty obvious. However you build it, it's nice to have available, and I think all ELF hackers should have a VIP-complete plugboard so they can make use of VIP add-on hardware. More on that little item in a later subheading.

AN ELF ADAPTER BOARD FOR THE VIP—Just because the VIP has more bells & whistles than the ELF doesn't necessarily mean that the VIP can run ELF software. In fact, the VIP is sadly lacking in virtually all basic ELF 1/0. About all they have in common is the Q lamp. I found it necessary and convenient to design a board for the VIP's X-BUS which would supply toggles, LED display, and INPUT switch to the VIP. This essentially gives you a 2K ELF with cassette interface, Pixie mod. and other good stuff. Especially that cassette interface. Lack of which is the biggest single reason ELF programs remain small and rather trivial.

By design or by coincidence, the VIP and ELF do not use any of the same I/O codes, so adding ELF 1/0 to the VIP remained as easy as plugging in a board without further modifications to the VIP. I wire-wrapped the whole thing on a Vector 3662-5 plugboard. There was enough room after I was through to add the memory address display given earlier in this book, something else the VIP badly needs.

The devices on the board are accessed the same way and by the same codes and instructions ELF 1/0 is accessed. The only difference, really, is that the toggles can no longer be used to load programs via DMA as with the ELF's LOAD switch. (You won't really miss that too much now, will you?) The toggles can still be used to load programs from within the ETOPS operating system, which will, in fact, run on a VIP with this board.

A little more good news lies in wait if you have added the hex keypad to your ELF. The VIP hex keypad is driven in precisely the same way as the ELF keypad, by scanning the keyboard through a 4515 4-line to 16-line latching decoder. If you add this circuit to your VIP, the VIP will also run ELF software which uses the ELF hex keypad. Even the I/O lines are the same (62 OUTPUT instruction and EF3). ELF software incorporating the Pixie mod. for video will also run on the VIP, as the 1861 is turned on and off via the same I/O lines.

A word of warning here involving all add-on boards to any computer. Don't plug them in or yank them out with the power still on. You may not hurt anything—l've got a whole wall of machines and again—but you may also kill one or more of the IC's on the add-on board. A little rider on most CMOS IC data sheets tells you not to connect circuit to a board under power—and they mean it.
### I/O DATA STROBES

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>61</td>
<td>Turns 861 off</td>
</tr>
<tr>
<td>62</td>
<td>Latches scan byte to hex keyboard</td>
</tr>
<tr>
<td>63</td>
<td>Strobes a byte to the 508 output port</td>
</tr>
<tr>
<td>64</td>
<td>Used by op system to transfer control to code at H(00 00) but after that it's OK for user tasks</td>
</tr>
<tr>
<td>65</td>
<td>Not used by VIP</td>
</tr>
<tr>
<td>66</td>
<td>Not used by VIP</td>
</tr>
<tr>
<td>67</td>
<td>Turns 861 on</td>
</tr>
<tr>
<td>68</td>
<td>Strobes a byte in from the 508 output port</td>
</tr>
<tr>
<td>69</td>
<td>Not used by VIP</td>
</tr>
</tbody>
</table>

### SEARCH OF COSMAC STANDARDS—Trying to run somebody else’s software on your machine is usually a pain in the butt. Their I/O strobes stir the wrong ports, and their cabling routines go looking for ROM in places where you have empty air. Etc. I can think, Furthermore, of half a dozen small outfits and one large one (RCA), selling COSMAC hardware, and none of them are the least bit compatible to any other. Bad scene. The S100 New Altair bus was helped along by the fact that the Altair, at its inception, was all there was. This makes de facto standards a lot easier to create.

Below I've included diagrams of the pinouts on the 2 VIP expansion slots, X-BUS & IO-BUS. IO-BUS is a much more limited pinout, but it has the advantage of using only one side of a 22/44 pin edge card. This makes it easier for home PC makers to make their own boards; I have yet to make a 2-sided PC which I have been pleased with. X-BUS, while double-sided, lacks nothing I can discover, and if you're smart you'll home-hack your hardware on Vector wire-wrap cards, which are double-sided by nature.

#### CS (Pin X) is an output. Whenever the operating system ROM is selected, this pin will be at +5 V. Otherwise, this pin is at ground. Since your ELF has no op system ROM (the using RCA's ROM isn't a bad idea, if you can get one; see below) leave this pin unconnected. It is of minimal use any way.

#### XTAL (Pin 4) in the VIP is the raw crystal oscillator output of 3,521,280 Hz. In the VIP this frequency is halved, and the 1602 runs at 1.76364 MHz. The VIP, you see, does not use the 1602's on-chip oscillator. The reasoning (for the VIP) is that with one crystal they get a clock frequency for a CD-type chip (must be less than 2 MHz) and the color-burst frequency (3.52 MHz) for generating color video on the VIP Color Add-On Board. To use the VIP color board in an ELF, you will have to duplicate these conditions. You might as well use the same circuit as in the VIP, see below. I dislike using TTL in an all-CMOS system, but not all 7400's will work in this mode. You may have to mess with different values for the resistors to make any particular 7400 work. Some may not work at all.

I should mention that this circuit should look familiar to people who have added the Pixie graphics mod to their ELFS; it's the same one mentioned on page 43 of the July '77 RE. It's a good idea to use this mod if for no other reason than it allows you to use an el-cheapo (I've seen them for 49 cents) color burst crystal.

#### Most of the inputs and outputs on X-BUS are self-explanatory; the bulk of them are taken right from the 1602 chip socket. I'll briefly discuss the unfamiliar signals so you know what to do with them.
The RUN comes right from pin 3 of the 1802. It's high when the VIP is running and low when not running, and it's controlled by the RUN switch, not by any 1802 internal logic. It's useful for clearing I/O ports and such. (Check out my use of this pin on the 1952 I/O circuit on Page 3.)

INDIS is an input; it is tied through a resistor to ground. If it remains until some outside device pulls it to +5V, when that happens, all VIP internal memory will be disabled, meaning that all outputs will "float" harmlessly as high and if you add an external memory board, you must make provision to disable internal memory, no matter if you have an ELF, VIP, or whatever. To make this work on an ELF, tie the pin 19's of all 2101 memory chips together and connect them to contact 19 (pure coincidence) of the X-BUS. Then (and don't skip this item) connect a 10K 1/4W resistor between X-BUS contact 19 and ground. If you omit the resistor, the chip enables of your 2101's will drift from ground to +5V, turning your memory chips on and off and raising havoc. This also leaves your memory chips open to damage by static electricity. Details count. If you're using the 2114 memory circuit shown on Page 2 of this book, things are a little trickier. You have to insert an OR gate between the 4081 and the chip select pins of the 2114's. The diagram below gives the necessary data. The end result is that regardless of what the output of the 4081 is doing, a high on the INDIS input to the OR gate will bring the 2114 chip selects high and disable the chips.

CDEF (X-BUS contact 20) is a weird one. Whenever VIP memory page C,D,E, or F are selected, this pin gets to ground. The hooker is that only a tiny percentage of VIP's have those last four pages of internal memory. VIP's are sold with only 2K, which translates to pages 0 through 7. My hunch is that you had better leave this one disconnected. None of the VIP add-on hardware uses it, as far as I can tell.

So much for the X-BUS. Setting up an IO-BUS is easier. The only non-1802 contact is INST, contact K. INST is a latch strobe for the parallel input port in the VIP. When you bring this pin to ground, a byte is latched into the parallel input port from some external device. I personally don't approach that approach, that approach puts the burden of generating the latch strobe on the external device. My 1852 I/O port arrangement automatically strobes the input port with TPB every machine cycle. This means, in essence, that any data which hangs around for at least one machine cycle will get latched into the port. If you prefer it the other way, it's no big thing to separate the input port from TPB and strobe it externally. On the 1852, however, a high level pulse sets the latch. (The VIP uses 4080's again for its parallel ports, with a different strobe polarity.) Tie this 1852 pin 11 to ground through a 10K resistor and bring it to +5V to latch a byte in.

If you want to make your 1852 I/O system emulate the VIP's I/O ports (and that's a good idea) you have to strobe the input port with a 6B INPUT instruction. This translates to using pin 3 of the 4028 chip (all this refers to the 1852 I/O system, Page 3). Fortunately, the VIP strobes its output on a 6B OUTPUT instruction, which also uses pin 3 of the 4028. On my diagram, replace the asterisk with a 3, and your 1852 I/O system will emulate the VIP's.

How far you carry VIP/ELF compatibility is a good question. Carry it far enough, and you no longer have an ELF, but a VIP. For that matter, it's possible entirely to lay hands on a VIP manual and wire-wrap your own VIP from scratch. The only "exotic" part is the op system ROM, and I have heard of people ordering it through RCA distributors; the part number is CDP566. Everything else can be scrounged through normal channels. If any reader has made a VIP from scratch, I would like to hear about it.

I made the choice between the Netronics ELF-II and the VIP and I'll stick with it, even though Netronics has a whole lot more software available at this time. If you would prefer to configure your expansion slots to follow the ELF-II bus, I'm also including a description of its contacts. I don't like it because the contacts are expensive, hard to find, and somewhat fragile. But—take your choice. Since I've not played very much with the ELF-II, detailed description of the contacts is beyond me. You might get in touch with Jim Lisowski, my partner in Milwaukee, and if he reports enough interest we will cover the subject in a future volume of Captain Coopa.
SUPER SOUND REDUX--The last time I mentioned VIP Super Sound, I had had it less than a month and hadn't figured it out to my own satisfaction. Well, time makes many things clear. With a few months to ponder the device, I find I really like it. This is, of course, a little late to sell a tape of Super Sound Christmas music, but I have several recorded, and by Christmas '80 I may well have a good tape to hawk. Check with me.

GOOD KING HENRY VIII

A NOTE TABLE

| 04 01 |
| 06 6F 6F 6F 71 6F 6F AA |
| 06 6C 6A 6C 6E AF AF 76 74 |
| 7E 7A 73 71 71 6A 6A 6C |
| 6E AF AF 6A 5A 6C 6E 6F |
| 6F B1 76 74 73 71 AF B4 |

A MEASURE TABLE

| 03 00 |
| 01 05 08 0C 01 05 08 0C |
| 0E 1B 15 19 1B 1F 22 26 |
| 28 29 00 |

B NOTE TABLE

| 03 01 |
| 6F 71 73 74 76 74 B3 |
| 94 33 71 36 34 73 71 AF |
| 71 74 B6 6F 71 B3 74 73 |
| 7E 73 71 73 74 76 73 74 |
| 71 73 7B BA 73 71 6E 88 |
| 84 F3 E0 |

B MEASURE TABLE

| 03 02 |
| 01 05 08 0D 01 05 08 0D |
| 10 11 16 19 1B 21 24 27 |
| 29 2A 00 |

BREAK TABLE

| 02 70 |
| 13 01 C0 16 01 E0 00 |

The VIP Super Sound Board is an interrupt-driven music generator. It's centered around the new CDP1863 chip, which is a CMOS programmable frequency divider. As best I can tell, the 1863 works like this: You strobe a hex byte into its latch and apply a relatively high frequency square wave to an input pin. The chip divides the input frequency by the hex byte and applies the result to an output pin. In Super Sound the input frequency is the 3.579 crystal frequency from the 1863 clock. The output frequency can pretty obviously fall anywhere you want to put it, from subaudible to the low RF. This is not a sure description as I have not yet been able to lay mitts on an 1863 data sheet, but it's close.

Briefly, and in very general terms, the Super Sound hardware system works like this: (follow the bouncing diagram, if you will.) The twin CDP1863's are programmable frequency dividers. You latch a hex byte into each one via the OUT instruction. Each 1863 will divide an input frequency by the hex byte latched into its registers. Each time you latch a different hex byte into one of the 1863's, a different tone will be produced by that channel.

You can also change the input frequency. One of the nifty things about Nature is the way musical octaves are related: Each octave is twice the frequency of the one beneath it, and half the frequency of the one above it. Sounds like binary to me, and by using a series of simple divide-by-two flipflops, Super Sound produces separate frequencies that are precisely one half, one fourth, and one eighth the input frequency. Those three taken with the unaltered input frequency make four input frequencies which can be applied to the 1863's—all harmonically related! That means that, for a given hex divider byte inside the 1863, the four input frequencies will produce the same note across four octaves.

One of the four input frequencies is selected by something called a crossbar switch; there is one such switch for each 1863. This device is the electrical equivalent of a 4-position rotary wafer switch. Instead of turning a knob, a binary code is strobed into the crossbar switch, which routes the input frequencies electronically.

Super Sound also has a multivibrator putting out pulses at a 50-250 Hz rate, adjustable by a little trimpot. The pulses strobe the interrupt line very much like the 1861 strobes the interrupt line to produce video refresh. (This is what we mean when we say a peripheral is "interrupt-driven.") Each time the pulser triggers an interrupt, PIN-8 checks to see if it's time to begin playing the next note or the next measure. That's all done in software; the board merely supplies the signals and timing. Spinning the trimpot changes the tempo, from ponderous Adagio to fizzy Allegro. Neat, huh?
There's somewhat more to it than that, but the rest is enhancement. PIN-8 has the capability to change the output envelope shape, giving you a full application of flanging or other instruments. Also, the amplitude can be varied across sixteen levels. The effects, when used together, are subtle and surprisingly good.

The hardware is excellent. Interface is simpler than it sounds. There seems to be no hardware reason to keep you from plugging Super Sound into an ELF, assuming you had set up a 22/44 pin connector to look like the VIP X-BUS. (A little mod of it in the future, elsewhere.) Super Sound requires a 3.579 clock frequency and a memory inhibit input (no big deal) but otherwise needs nothing that an ELF cannot easily supply.

...there's a helluva hook in it. Super Sound is helpless without the PIN-8 'language', and PIN-8 requiresulp.172 bytes or RAM. Unless I've read it wrong, nobody else in twenty has that much RAM. Most ELF-II and Super ELF machines can't handle it either, I think, though expansion is easier on them. Yes, PIN-8. Play It Now. Arrgh. Super Sound comes with a scrappypatch little booklet virtually handwritten by the Great White ELF-Father, Joe Weisbecker. It looks very much . . . and then you come to the source. What's there is good, but it isn't very much . . . and then you come to PIN-8. What you get is a hex dump of 1792 bytes, printed on what must be a pin copy. It looks very much . . . and then you come to the source. What's there is good, but it isn't very much . . . and then you come to PIN-8.

You probably already know that a string of opcodes is tough to dope out if you didn't just write them ten minutes ago. That's why you put comments on the right side of your paper, to make them easier to understand. Initially, that's what your "source" is; it's the part of your program that makes the opcodes easier to understand.

The majority of programs you buy (excepting those with the English language statements called "source", you only get the object code, on a disk, tape, or on paper. No effort is made to explain how the thing works. In fact, some programs burn a lot of mental calories trying to devise ways of keeping people from even displaying or copying the object code, all in the name of desperate paranoid secrecy.

Howcum? Not sure. Programming houses claim they don't want other companies to learn their techniques. That may be wise, as Sturgeon's law applies here; 90% of programming techniques are likely to be inefficient, sloppy, and full of kludgeywork. If the customer knew what he was getting, he wouldn't pay $150 for a goddam mess. Busnel baskets can cover sins as well as magic.

Individual hobbyist programmers have even less excuse. Damn few ever even try to make money off their programming skill, so passing out their techniques hurts not even themselves. They don't need to "source" their programs, even for a profit.

Ultimately, it must be a power thing. If only you have the source, you in effect control the program even after you sell it. Without the source, it's hard to change a program to custom-tailor it to specific needs. The man who has the source has the power. The majority of programmers, myself very definitely included, are ex-wimps and high school nerds, social misfits who were stepped on and laughed at during their development. It's human not to want to let go of power when you finally get it. Human. But paranoid. And needless. I double-dog dare all of you to find anything in this book that will tell you the source has the power. The major source's power is the power of source. That's why I paid $150. I don't want to give you the source and let you have the power. I dare you to match my promise; I'll sell you a tape with Super ELF-8. Play It Now. It makes music.

In conclusion of this subject, I'll say that PIN-8 is probably larger than it has to be. A COSMATIC software wizard could probably write a PIN-8 in two or three RAM pages. You'd probably lose the steel guitar effects and maybe one channel, but it would still make music. I'd like to see Super Sound for an M-80. As Mr. Tolkien knew, Elves and music are more than one. I'd pay for such a program. Not much, maybe, but more than you'd get by giving it to an 1802 newsletter for free.

BINARY ARITHMETIC SUBROUTINE BOOK--RCA has a good-sized book devoted to a set of subroutines used to do the important stuff like binary and BCD arithmetic. Arithmetic is tough on the 1802, and a lot of junior ELF hackers lay out five beans for this book, only to find they can't use it. The book is Manual WP-M-206, "Binary Arithmetic Subroutines for RCA C0S1MATIC Microprocessors." The subroutines give you the
The subroutines use quite a few of the 1802's registers. You will have to use some arithmetic subroutines and arithmetic subroutines and stack and whatnot into a 256 byte ELF, there's no room left for a program to use the subroutines. The subroutines use quite a few of the 1802's registers, so if you want to use both the subroutines and the registers, you have to custom-tailor SCRT to save any registers your program wants to use, and I'm not even sure I can do that right now! For this book to be worth it, you must have: 1. An 1802 machine with at least 1K RAM; 2. An application that demands considerable number-crunching; and 3. Thorough understanding of SCRT, enough to reconfigure it to save any registers your application program intends to use. This is not trivial. (I tried it.) Save a pin. Grow a little before you buy it.

FAST/SLOW CLOCK GENERATOR--Jim Lisowski has come up with a terrific little geegaw that will work on any VEP or ELF computer. The 1802 family is totally static; that is, you can slow down the basic clock speed to one clock pulse every ten minutes, or even 60 and stop things dead in their tracks. While it might seem dumb to have a ten second machine cycle, think again: You can actually watch the various control lines change state at such a clock speed. It's a helluvalot of education into the 1802 and up theory generally. If you have one of those glopper clips with LED's for each pin, you can see TP4 and TP6 and the SC lines go on and off. Even a VOM will show things switching around. First rate.

What Jim put together basically an adjustable 555 oscillator glued piggyback atop a 74L00 soldered to a 10 or 18 pin DIP header. A little slide switch switches the clock output from the 74L00 (full 3.579 MHz) to the 555, which is adjusted by a little thumbwheel trimout. How the thing goes together depends on what parts you decide to use, but mine turned into a very compact little item which you could probably use for a hockey puck without damage.

I built mine on an 18-pin DIP header to have a little extra room. I recommend doing this. You start by testing all the parts. I mean it! Once you glue something with Eastman 910, it takes God the Father with a jackhammer with a needle tip also helps. Use a small soldering iron with a needle tip also helps. Use a little heat as you can to avoid softening the plastic of the DIP header. Installing the switch comes next. I found a teeny-tiny one that actually fit between opposite contacts on the end of the DIP header. I soldered the outside switch contacts to the header and then glued the body of the switch to the end of the 74L00. Wire up what you can from the switch to the IC's. Now add the thumbwheel pot. I used the very common little one with a blue plastic wheel; Radio Shack sells them. It goes atop the 555, thumbwheel up (1) and contacts pointing back toward the switch. Eastman-910 it to the 555, and finish the wiring. The resistor fit well between the pot contacts and the switch; the cap had to be slung along the side. Put the everything where it fits, and glue it all together. The reason for the glue is subtle: You're going to be pulling and pushing this thing into and out of an IC socket, and you'd prefer not to yank it apart if it gets stuck.

Drive a 2-speed clock home today!

At this point, take some wire-wrap wire and wire what you can without regluing the switch, resistor, pot, or cap. It takes small fingers, time, and skill to do this well. A small soldering iron with a needle tip also helps. Use a little heat as you can to avoid softening the plastic of the DIP header. Installing the switch comes next. I found a teeny-tiny one that actually fit between opposite contacts on the end of the DIP header. I soldered the inside switch contacts to the header and then glued the body of the switch to the end of the 74L00. Wire up what you can from the switch to the IC's. Now add the thumbwheel pot. I used the very common little one with a blue plastic wheel; Radio Shack sells them. It goes atop the 555, thumbwheel up (1) and contacts pointing back toward the switch. Eastman-910 it to the 555, and finish the wiring. The resistor fit well between the pot contacts and the switch; the cap had to be slung along the side. Put the everything where it fits, and glue it all together. The reason for the glue is subtle: You're going to be pulling and pushing this thing into and out of an IC socket, and you'd prefer not to yank it apart if it gets stuck.
Operation is dirt-simple. The module replaces the clock generator IC. With the slide switch in one position, the output (pin 8) of the 74L00 will feed the clock line, and there will be a 3.579-MHz clock. Flip the switch, and the clock speed comes from the 555, depending on the setting of the pot.

On the ELF-II you can leave the module in place all the time. On the VIP the blue plastic cover won’t close over it, so I had to make it easily removable. Sadly, it won’t work on the original ELF configuration unless you’ve added the Pixie graphics mod, which is still another another reason to buy an 861.

Obviously, the 555 will not make the VIP’s video function properly, but it’s fun to watch the contortions on the screen as you rev the old video function properly, but it’s fun to watch the thumbwheel. Cheers!

INTERRUPTS MADE SIMPLE—Except for the Pixie video mod (which most people built without understanding a lick of it) damned few ELF hackers have made any use of the 1802’s interrupt facility. Certainly the hardware is simple enough to use: a low pulse on the interrupt pin will give you an interrupt. Bang! But who knows exactly what an interrupt is, much less how to use one in a program?

At the bottom line, an interrupt is a tap on the CPU’s shoulder, instructing it to drop what it is doing and take care of the other thing right away. My best example I can supply is a hypothetical program (hypothetical because I haven’t got the bugs out of it yet) for a buffered Morse Code keyboard. A Morse Code keyboard is a device on which you type ASCII characters and output Morse Code. Hams use them frequently, merrily typing away at a chirpy 120-wpm rate, chomping the code reply in their heads. A buffered keyboard is one in which typed characters are stored in a buffer while previous characters are being sent in Morse Code. An unbuffered keyboard must complete each character before a new one is typed; such a program exists somewhere in this book. It would seem, on the surface, that the CPU must be able to do two things at once: To output dots and dashes in proper Morse format, and simultaneously listen to the keyboard for new ASCII keypresses. Nay, nay. Enter the interrupt.

The program constantly monitors a 256-byte segment of memory, which we call a buffer. When the buffer is empty, the CPU keeps looping and checking. Let us say (without discussing the mechanism yet) that forty bytes are written into the buffer at a rapid succession. The CPU immediately begins translating each ASCII byte into its Morse equivalent, and sending out the dots and dashes on the Q line. It has work to do for several seconds.

A second later, three more ASCII bytes are typed in. The processor is busy making more dashes, but the ASCII keyboard brings the interrupt line low for a moment. WHAM! The CPU jumps to a routine which accepts the first ASCII byte and stores it on the top of the buffer. The entire process takes twenty microseconds or so. Then the CPU returns to complete a dot which is now twenty microseconds longer than dots usually are. That dot is not even complete when the second of the three bytes comes in and the interrupt line goes low. Another twenty microseconds store the byte on the buffer, and when the CPU finishes its dot, the third byte comes through during the space between the dot and the following dash. The space grows by twenty microseconds. I doubt anybody will notice.

As long as nothing is typed on the keyboard, the CPU is continuing transmitting Morse characters until the buffer is once again empty. But as soon as a key is pressed, the interrupt is initiated and the key byte is stored in the buffer.

So much for the concept. How does the 1802 implement it? The interrupt line is checked by the CPU along with the TPB pulse of every "execute" machine cycle. If during that TPB pulse the interrupt pin is low, the interrupt begins with the following machine cycle.

There is a flip-flop within the 1802 called the Interrupt Enable flip-flop. (IE, we will call it.) When the 1802 starts running, this flip-flop is set high from factory. It permits interrupts; when reset, the 1802 ignores the interrupt line. The first thing the 1802 does after it begins an interrupt is to reset IE. This prevents any more interrupts until it is again reset by the program. Next, the 1802 stores the current value of X and P side by side as a single byte in an 8-bit subbyte called T. (Note that it is the number of the registers serving as X and P that are saved, not the 16-bit contents of those registers. That comes later, under program control.) Finally, the 1802 puts 1 in P and 2 in X. You had better have a workable I/O routine pointed to by register 1, because the 1802 is going to start executing whatever code begins at M(R(1)). At this point the dedicated interrupt hardware inside the 1802 chip leaves off, and the work is taken up by the interrupt software, running with R1 as the program counter.

Implicit in the interrupt concept must be some way of going back to exactly what the "main" 1802 program was doing at the time the interrupt began. The interrupt hardware saved the "main" P and X in the T register. The interrupt program must put that P and X somewhere. "Somewhere" means M(R(X)) by way of a SAVE instruction, and unless you change X first off, it will be M(R(2)) as set by the interrupt hardware.

In a program that uses interrupts, R(2) should point to the top of a free area of memory at least five or six bytes in size. A convenient place is the very top few bytes of some memory page, that is, XX FF - XX FF, with R(2) pointing to XX FF to start. This sixteen-byte area is called the "stack." It's a stack because we can stack bytes up for storage in that area, and then flip them off the stack for re-use, much like a stack of dinner plates. (Yes, I realize that we stack the bytes downward from XX FF, but if it bothers you, flip your damned computer over.)

There's no reason this stack can't be used for other purposes in a program. Therefore, don't assume that M(R(2)) points to a free memory location. It may, in fact, point to some data needed by the "main" program. So, let's assume (and program accordingly) that any stack location below the "stack pointer" (R(2) here) is free, and any location above the stack pointer (between the stack pointer and XX FF) contains usable data. Any time you want to store a byte on the stack, DECREMENT R(2) one location first. Then, in this case, you would SAVE your T register onto the stack in that free location.

OK. Now we have X and P from the "main" program safely stored on the stack. We also will want to save the D register, certainly. That can go on the stack by decrementing R(2) again and then performing a STORE VIA R(2) instruction.

What else needs to be stored? That depends. It is possible to save the DP flag by performing a RING SHIFT RIGHT instruction to get DP back into
the D register, and then SAVE the D register as described above. Also, if your interrupt routine wants to use any of the same registers the "main" program is using, you have to save those as well. This is done using GET HIGH R(X) and GET LOW R(X) instructions to move the register halves into the D register, and then storing the D register as above. Remember to DECREMENT R(2) before you store something!!

As illustration, the following snippet will take care of the initial housekeeping for an interrupt routine that shares D, DF, and R(9) with the "main" program:

```
XX 20  22   DEC R(2)  
21  70   SAVE T to M(R(2))
22  22   DEC R(2)
23  73   STORE VIA X & DECREMENT
24  75   RING SHIFT RIGHT
25  70   STORE VIA X & DECREMENT
26  90   GET LOW R(9)
27  70   STORE VIA X & DECREMENT
28  99   GET HIGH R(9)
29  52   STORE VIA R(2)
```

(Onward with the interrupt routine's real work.)

Notice that your stack has grown five bytes by this little operation, so you had better have at least that much (and preferably a good deal more) space available for the stack to grow.

This process can be extended to include any or all GP registers, but as you can see, saving any large number of registers would make for a very long interrupt routine that takes lots of memory space and execution time. If at all possible, assign certain registers to the "main" program and leave them alone in your interrupt routine. If your interrupt routine does not affect the DF flag, don't save it. Often you can get by simply by saving X & P from the register, and then D, the video refresh routine for the Pixie mod does this.

Once the interrupt routine has done all the saving it has to do, it runs just like any other 1802 program, except that interrupts are inhibited by the reset state of the IE flag. Eventually you will want to get back to the serious business of the "main" program, so all that data that you saved on the stack has to be popped off again and plugged back into the CPU's inner workings.

Writing the restore section of the interrupt routine is a matter of doing everything again in reverse. Usually, a STORE VIA X & DECREMENT instruction stores the contents of the D register at M(R(X)) and then decrements R(X); the LOAD VIA X & ADVANCE instruction loads the contents of M(R(X)) into D and then increments R(X). For example, start by restoring R(9). DF may be restored by loading its byte from the stack and then using a SHIFT LEFT instruction to move the most significant bit back into DF. D is restored just by loading its byte from the stack into D.

The last step uses the RETURN instruction to take the stored X & P from their side-by-side byte and insert them back in the registers from whence they came. This will, in effect, "turn on" the "main" program and then increment the routine's start, again as though no interrupt had occurred. RETURN also sets the IE flag to 1, and enables further interrupts. However, the program counter for the interrupt routine, R(1), now points to the end of the interrupt routine rather than the beginning, so the next time an interrupt occurs, the 1802 begins executing instructions on garbage that lie just beyond the end of the interrupt routine, and if experience is any guide the computer will go berserk. The trick is to perform a short routine immediately preceding the true beginning of the interrupt routine, where your RETURN instruction should lie. Once the RETURN is executed, R(1) will be pointing right at the start, ready to go for the next interrupt that comes down the line.

To recap, here is a model of a typical interrupt routine. In this case, it saves X & P and D (essential for all interrupt routines) as well as DF and R(9). (optional)

```
MA CODE
1F 70   RETURN
XX 20  22   DEC R(2)
21  70   SAVE T to M(R(2))
22  22   DEC R(2)
23  73   STORE VIA X & DECREMENT
24  75   RING SHIFT RIGHT
25  70   STORE VIA X & DECREMENT
26  90   GET LOW R(9)
27  73   STORE VIA X & DECREMENT
28  99   GET HIGH R(9)
29  52   STORE VIA R(2)
```

(Code for real interrupt business here)

```
XX 59  72   LOAD VIA X & ADVANCE
5A  B9   PUT HIGH R(9)
5B  72   LOAD VIA X & ADVANCE
5C  A9   PUT LOW R(9)
5D  72   LOAD VIA X & ADVANCE
5E  F2   SHIF T LEFT
5F  72   LOAD VIA X & ADVANCE
60  30   SHORT BRANCH
61  1F   (to XX 1F)
```

People who start playing with interrupts soon start to worry about a disturbing possibility: What happens if the interrupt line goes low soon after the 1802 begins running, before the 1802 has a chance to point R(1) to the interrupt routine? Only R(0) is cleared to 00 00 upon CPU reset (Read that line again); R(1) may contain any old garbage until you fill it with the interrupt routine address. If an interrupt occurs in the few dozen microseconds between RUN time and the time you point the interrupt routine, the machine will probably go berserk. There's a trick to get around it. Consider the following 3-byte instruction sequence:

```
E3   SET X = R(3)
71   DISABLE
93   (hex byte 53; not intended as a STORE VIA R(3) instruction!)
```

This sequence should be used in a program where P=3 and X=5, if it were used in a program where P=2 and X=2, the hex byte would be 23 and not 53. What we've done here is "fooled" the 1802 into reading its XP byte from the regular program sequence instead of the stack, by setting X=1 in the first instruction. (Review the action of the DISABLE instruction here; it reads the byte from M(R(X)) and interprets it as XP; then loads the X & P registers with the XP it read. Since the XP you stored immediately after the DISABLE instruction is the long integer at that point, you've changed nothing--except the state of the interrupt enable flipflop, IE. Interrupts will be disabled until a similar sequence with a RETURN in place of the 3-DOA is encountered. In this way, you can turn interrupts on and off at any time within the program, as required.

That special case of disabling interrupts immediately can be handled even more simply. At RUN time, P and X are both 0, so the SET X instruction is unnecessary. Loading the sequence:

```
71   00 as the first two bytes in memory will disable interrupts at RUN time immediately. Interrupts will be disabled until a three-byte RETURN sequence as described above is encountered.
```

Another caution is worth considering: If your interrupt line is held low for any long period of time (longer than a millisecond is a verrrrrrrrr long time) your interrupt routine may run and return before the interrupt line goes high again. If that happens, you will immediately get another interrupt, and whatever job the interrupt routine is supposed to do will be done twice, or three times, or more. Even worse, you may make sure that your interrupt pulse to the interrupt line is shorter than the execution time of the interrupt routine. It's relatively easy to configure a 555 timer as a one-shot, which means that you feed it a long pulse and get a short pulse out.
An Animated Video Face System

This program puts an animated face on a TV screen. Next Halloween, mount your CRT in the wall instead of a pumpkin. (CRTs don't rot, and candles are hardly state-of-the-art...) It will goggle, glow, and grin at passing children until one of them puts a brick through your window and takes CRT. Or, if you're not into Halloween, mount a CRT and a VIP atop your robot, and it will look a good deal dumber than it may in fact be. But it will be a lot more popular at parties.

Whatever you decide to do with it, this is how it works: When the program is run, debris from previous programs is erased from the screen. $SMLCPY$ copies a smiling face pattern to the display page from page 03. The smile as stored has no eyes. Eyes are written in immediately after the smile is copied. This done, subroutine $EF4TST$ is called. This samples flag line $EF4$. If $EF4$ is low, 01 will be stored in the memory location where CHIP-8 stores variable D. If $EF4$ is high (normal condition) 00 will be stored in variable D. Immediately after returning to the MAIN program, $VD$ is tested. If it is found to be 01, CHIP-8 subroutine $CHPLCK$ is called. $CHPLCK$ scans a tongue-shaped pattern across the smiling face's upper lip about three-fourths of the way, reverts and goes back to its starting point, then vanishes. During this time the tongue is scanned, closed eyes are displayed on the smiling face.

If $CHPLCK$ was called, the program returns to the beginning and starts again. If not, a timer is begun, and nothing further happens for about three fourths of a second. When the timer decays down to 00, a random number from 00 to 0F is pulled. If this number was 00, $FRNPCY$ is called, which copies a whole new face (also minus eyes) into the display page. This is a frowning face, and looks rather as though the creature was just forced to sample a mouse-entails-on-eye sandwich. Eyes are immediately copied in, and the eyes begin again. Each time the timer is begun, 01 is incremented by four and the eyes are recopied. This means the eyes change every three fourths of a second, and follow a definite sequence of rolling, blinking and looking around. The frowning face remains on the screen for four countdowns of the timer, and then for four eyes-changes. After that time the smile is copied back into the display page, and the whole sequence is begun again.

There is a CHIP-8 subroutine called $PGPEEK$ which will permit you to peek at any page in memory. This is strictly a bonus and is not used in normal execution of the program. To use $PGPEEK$ replace the CHIP-8 instruction $0200$ with $1550$. This will jump immediately to $PGPEEK$. $PGPEEK$ waits for a keypad key pressed, then displays the memory page corresponding to the number pressed. If you press a number for which no memory page is socketed (say, page 0C in a 2K VIP) the display will be all white. $PGPEEK$ does not depend on any other subroutine in the system and can be used in other VIP programs.

The only extra hardware required by this system is some means for pulling EF4 low to signal the chop-licking subroutine. This does not need to be bounceless at all; in fact, touching a wire to the ground pins with a short make just fine. I used this method to be able to trigger the tongue by remote control at appropriate times, say, when a suitably lecherous woman entered the room. Most women feel silly slapping a TV set, and some might even be amused, since TV sets have no legs and are considered "safe." For Halloween applications you might want to replace Chip's subroutine $EF4TST$ with a CHIP-8 subroutine for pulling and testing a random number to determine whether or not to trigger $CHPLCK$. This particular subroutine, stored at the location where $EF4TST$ is now, will do the job well:

```
06 20 000E 00 00
4D05 Skip if VO  05
6D01 VO  01
00EE Return to MAIN
```
If you decide to use this subroutine, the call instruction at 02 18 will have to be changed to 2620, because the new subroutine is a CHIP-8 subroutine, and call processes are a little different. When in place, the shop-licking process will be initiated automatically, and no hardware need be added to the VIP.

Note that this program requires a 3K VIP. Remember, that highest RAM page is the display buffer, and the next to highest RAM page is a CHIP-8 playground and best to stay out of. If you have a 4K system, make these program changes:

05 FF 0E 0E 0E 0E 06 04 0F 06 21 0E 06 53 OF

These changes all involve display and variable-storage page pointers. If your VIP is some jacked-up super system with offcard RAM, I will leave the modifications to you.

And that's about it. Presumably, you could make up a whole range of face expressions and store each as a separate face in a page of RAM. Certainly there's room enough in the VIP for three or four more faces, and that's a lot. Animating the eyebrows would be a cinch if you wanted to take the time. Why not do it?

ANIMATED FACE CONTROL SYSTEM PAGE 05

CHIP-8 SUBRoutines

MA OPCODE SUBROUTINE: CHPLCK
05 00 D124 ERASE LEFT EYE
02 D324 ERASE RIGHT EYE
04 A2BC POINT I TO CLOSED EYES
06 D124 SHOW LEFT EYE CLOSED
08 D324 SHOW RIGHT EYE CLOSED
0A 6311 V5 = 30
0C 6630 V6 = 30
0E 6709 V7 = 09
10 6709 V7 = 09
12 6709 V7 = 09
14 6A01 VA = 01
16 A400 POINT I TO TONGUE
18 D659 SHOW TONGUE
1A 86A5 V6 = V6-VA
1C F71E I = I+V7
1E 3610 SKIP IF V6 = 1D
20 151B GO TO 0518
22 86A4 V6 = V6-VA
24 0640 DO SUB: I=DEC9
26 D659 SHOW TONGUE
28 3630 SKIP IF V6 = 30
2A 1522 GO TO 0522
2C A290 POINT I TO FIRST EYES
2E 6400 VA = 00
30 D124 SHOW LEFT EYE
32 D324 SHOW RIGHT EYE
34 00EE RETURN TO MAIN

ANIMATED FACE CONTROL SYSTEM PAGE 06

MACHINE LANGUAGE SUBRoutines

MA OPCODE SUBROUTINE: SMLCPY
06 00 F8 03 BD POINT RD.1 TO PAGE 03
03 F8 0B BE POINT RE.1 TO PAGE 0B
06 F8 FF AD EA POINT TO TOPS OF PAGES 03 & OB
0A BE X = E
0B OD LOAD M(R(D)) INTO D
0C 2D DEC RD
0D 73 STORE D AT M(R(E)) & DEC RE
0E 8E GET LOW BYTE RE.O
0F 3A OB IF D = 00, GO TO (06 OB)
11 OD LOAD BOTTOM BYTE OF PAGE 03
12 5E STORE D IN BOTTOM BYTE OF PAGE OB
13 D4 RETURN TO MAIN

MA OPCODE SUBROUTINE: BF4TST
06 20 F8 0A BE POINT RE TO RAM LOCATION WHERE
23 F8 FD AE CHIP-8 STORES VALUE OF VD
26 3F 2C IF EFS IS LOW, G, TO M(06 AC)
28 F8 01 5E D4 LOAD O1 INTO VD AND RETURN TO MAIN
2C F8 00 5E D4 LOAD GO INTO VD AND RETURN TO MAIN

MA OPCODE SUBROUTINE: IDEC9
06 40 2A 2A 2A DECREMENT RA (WHERE CHIP-8 STORES
43 2A 2A 2A THE VALUE OF MEMORY POINTER I)
46 2A 2A 2A NINE TIMES
49 D4 RETURN TO MAIN

MA OPCODE SUBROUTINE: PRCPY
06 50 F8 07 BD POINT RD.1 TO PAGE 07
53 F8 0B BE POINT RE.1 TO PAGE 0B (DISPLAY)
56 F8 FF AD EA POINT TO TOPS OF PAGES 07 & OB
5A BE X = E
5B OD LOAD M(R(D)) INTO D
5C 2D DEC RD
5D 73 STORE D AT M(R(E)) & DEC RE
5E 8E GET LOW BYTE RE.O
5F 3A 5B IF D = 00, GO TO M(06 5B)
61 OD LOAD BOTTOM BYTE OF PAGE 07
62 5E STORE D AT BOTTOM BYTE OF PAGE OB
63 D4 RETURN TO MAIN

ANIMATED FACE CONTROL SYSTEM PAGE 04

TONGUE PATTERN STORAGE

MA DATA
04 00 1C 22 22 22 2C 00 00 00 00
09 12 33 33 33 63 22 22 22 22
12 12 33 33 33 63 22 23 23 23
1B 12 33 33 33 63 22 23 23 23
24 12 33 33 33 63 22 23 23 23
2D 12 33 33 33 63 22 23 23 23
36 12 33 33 63 63 22 23 23 23
3F 12 33 33 63 63 22 23 23 23
48 12 33 33 63 63 22 23 23 23
51 12 33 33 63 63 22 23 23 23
5A 12 33 33 63 63 22 23 23 23
63 12 33 33 63 63 22 23 23 23
6C 12 33 33 63 63 22 23 23 23
73 12 33 33 63 63 22 23 23 23
7E 12 33 33 63 63 22 23 23 23
87 12 33 33 63 63 22 23 23 23
90 02 33 33 63 63 22 23 23 23
99 02 33 33 63 63 22 23 23 23
A2 02 33 33 63 63 22 23 23 23
AB 10 13 13 13 13 63 63 63 63
BA 00 33 33 63 63 22 23 23 23
BD 10 33 33 63 63 22 23 23 23
C6 12 33 33 63 63 22 23 23 23
CF 10 33 33 63 63 22 23 23 23
D8 02 33 33 63 63 22 23 23 23
E1 02 33 33 63 63 22 23 23 23
EA 12 33 33 63 63 22 23 23 23
F3 10 33 33 63 63 22 23 23 23
FC 00 00 00 00

ANIMATED FACE CONTROL SYSTEM PAGE 04

TONGUE PATTERN STORAGE
### MA Opcode Interpretations

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 F8 00 B3 B4 B5 B7</td>
<td>Initialize high order register bytes</td>
</tr>
<tr>
<td>06 B8 B9 BA BB BC BD</td>
<td>Initialize stack pointer</td>
</tr>
<tr>
<td>0C F8 FF A7</td>
<td>Initialize DELAY PC</td>
</tr>
<tr>
<td>0F F8 52 A8</td>
<td>Initialize ASCII lookup table</td>
</tr>
<tr>
<td>12 F8 74 A9</td>
<td>Initialize Morse element counter</td>
</tr>
<tr>
<td>15 F8 01 AC</td>
<td>X=7</td>
</tr>
<tr>
<td>18 E7</td>
<td>Branch to KEYER if INPUT pressed</td>
</tr>
<tr>
<td>19 37 62</td>
<td>Call DELAY twice</td>
</tr>
<tr>
<td>1B D8 D8</td>
<td>Loop until key pressed</td>
</tr>
<tr>
<td>1D 3E 1D</td>
<td>Input ASCII byte from keyboard</td>
</tr>
<tr>
<td>1F 68</td>
<td>Output ASCII byte to display; dec. R7</td>
</tr>
<tr>
<td>20 64 27</td>
<td>Load first value from ASCII table into D</td>
</tr>
<tr>
<td>22 09</td>
<td>Go to M(2B) if table byte = 0</td>
</tr>
<tr>
<td>23 32 2B</td>
<td>Load keyboard byte into D</td>
</tr>
<tr>
<td>25 07</td>
<td>X=9</td>
</tr>
<tr>
<td>27 F3</td>
<td>Test for table=keyboard via XOR</td>
</tr>
<tr>
<td>28 32 34</td>
<td>Go to M(34) if keyboard byte found</td>
</tr>
<tr>
<td>2A 38</td>
<td>Skip next instruction</td>
</tr>
<tr>
<td>2B 1C</td>
<td>Increment Morse element counter RC</td>
</tr>
<tr>
<td>2C 19</td>
<td>Increment lookup table pointer</td>
</tr>
<tr>
<td>2D 8C</td>
<td>Fetch Morse element counter to D</td>
</tr>
<tr>
<td>2E FD 08</td>
<td>Subtract D from 08</td>
</tr>
<tr>
<td>30 3B 00</td>
<td>Go to start if Morse element counter &gt;08</td>
</tr>
<tr>
<td>32 30 22</td>
<td>Go test keyboard byte against table again</td>
</tr>
<tr>
<td>34 89</td>
<td>Fetch address of found table byte to D</td>
</tr>
<tr>
<td>35 FC 2E</td>
<td>Add 2E to address to point byte to encoder</td>
</tr>
<tr>
<td>37 A9</td>
<td>Store encoder address in R9</td>
</tr>
<tr>
<td>38 09</td>
<td>Load encoder into D</td>
</tr>
<tr>
<td>39 38</td>
<td>Skip next instruction</td>
</tr>
<tr>
<td>3A 8D</td>
<td>Fetch stored encoder to D</td>
</tr>
<tr>
<td>3B FE</td>
<td>Shift encoder one bit left</td>
</tr>
<tr>
<td>3C AD</td>
<td>Store shifted encoder in RD.0</td>
</tr>
<tr>
<td>3D 3B 47</td>
<td>Go to M(47) if bit was 0 (dit)</td>
</tr>
<tr>
<td>3F 7B D8 D8 D8 7A D8</td>
<td>Form dah (Q on 3 time units, off 1 unit)</td>
</tr>
<tr>
<td>45 30 4B</td>
<td>Go to M(4B)</td>
</tr>
<tr>
<td>47 7E D8 7A D8</td>
<td>Form dit (Q on 1 time unit, off 1 unit)</td>
</tr>
<tr>
<td>4B 2C 8C</td>
<td>Decrement Morse element counter &amp; fetch to D</td>
</tr>
<tr>
<td>4D 32 00</td>
<td>Go to start if Morse character is finished</td>
</tr>
<tr>
<td>4F 30 3A</td>
<td>Go to M(3A) for next element</td>
</tr>
</tbody>
</table>

### DELAY

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>51 D0</td>
<td>Return to MAIN</td>
</tr>
<tr>
<td>52 E7</td>
<td>X=7</td>
</tr>
<tr>
<td>53 6C AB</td>
<td>Read byte from toggles and store in RB</td>
</tr>
<tr>
<td>55 F8 FF AA</td>
<td>Store inner timing constant FF in RA.0</td>
</tr>
<tr>
<td>58 2A 8A</td>
<td>Decrement RA &amp; fetch it to D</td>
</tr>
<tr>
<td>5A 3A 5B</td>
<td>Loop again if RA ≠ 00</td>
</tr>
<tr>
<td>5C 2B 8B</td>
<td>Decrement RB &amp; fetch it to D</td>
</tr>
<tr>
<td>5E 3A 55</td>
<td>Loop again if RB ≠ 00</td>
</tr>
<tr>
<td>60 30 51</td>
<td>Go to RETURN</td>
</tr>
</tbody>
</table>
**AN UNBUFFERED MORSE KEYBOARD PROGRAM—**This program is useful in keyboard practice, or as a demonstration of what the 8082 can do with minimal add-ons. You do need an ASCII keyboard, but you can find them floating around used for cheap, and even inexpensive for $50 if you're careful. The VIP ASCII keyboard is very nice once you get used to the elastomer and the squeak which subs for touch feedback. Though now that I think of it, I'm not sure how much of the feel might get in the way of audible Morse Code, so…maybe not the VIP keyboard. I used the Radio Shack TTL keyboard, but I don't recommend. It uses the individual TTL chips that suck up power and are hot…yea coot. Fortunately a friend of mine sat on it and broke the PC board, so it's behind me now.

Whatever keyboard you use, the idea is to bring EF3 low whenever a key is pressed. Most ASCII keyboards have a Data Ready line or something like that. If your Data Ready line goes high when a key is pressed you may have to change the instruction at 00 1D to 00 1B. When the program senses that EF3 has changed state, it will strobe a byte from an input port to the bus with a 6B INPUT instruction.

Then the fun begins. The program has what we call a "lookup table." Two lookup tables, in fact. One holds the ASCII character set, and the other an artificial "encoder byte" containing casts digits as 0's and dashes as 1's. The program scans down the ASCII table, matching the keyboard byte to the digits in the table. When it finds a match, it adds the address of the data byte to stores a byte which matches the keyboard byte. The address now points to the encoder byte.

But...the encoder bytes for E, I, S, & H are all the same. The Morse characters for these letters are dit, didit, diddit, and dididdit, respectively. So 00 will, in fact, be the encoder byte for any character made out of pure dits. The problem is, the encoder byte tells you nothing about which actual character is. That information is included in the table in the form of 0's and 1's between character groups. Hams will notice that the characters are grouped by the number of elements (dits or dashes) in the character. E and T both have one element, A, H, & U have two elements, and so on. A register half (RC.0) is started out with a value of one, and every time the scan software encounters a 00 in the ASCII table, it increments the element counter, RC.0. By the time the program finds the matching ASCII byte, the element counter will contain the number of elements in the character. So E, I, S, & H all come out different, as they should.

The Morse character is created by shifting bits out of the encoder byte and testing them. 0's cause a jump to a dit-former which turns Q on for the time DELAY is called, so you can change the speed of the Morse characters by changing the setting in the toggle switches. I should point out that this will not work on a VIP which you toggle switches via an HFE-adapter device like I described previously.

A bonus in this program is a little electronic keyer routine which is entered by depressing INPUT at the time you start the program running. The keyer is used by connecting paddles to EF3 and EF4 and connecting them to produce a continuous string of dits and dashes. The dits and dashes are self-completing, but the program has no memory, so you must let the current dit or dash finish before jumping ahead to the next one.

This problem extends to the keyboard program, in which the keyboard causes an interrupt to input the ASCII byte in a 256-byte FIFO (First In First Out) buffer. With such a program you could type erratically and get 256 letters ahead of the computer before you start eating your own tail and losing data. Sadly, at press time it doesn't work yet. Maybe one of these issues...

The best way to test this program is to use 0 to drive a reed relay, and then drive a CPD (Code Practice Oscillator) with the relay. That way you can leave it on and eventually use the reed relay to key your transmitter. Do take care to keep stray RF out of the computer. The first time I tried it, a raging tide of RF leakage raced through my power supply and wiped out 2K of RAM clean as a whistle! Dandididit dahandididit!
**VIDEO SCRATCHPAD PROGRAM**  
**BY JEFF DUNTEMANN**  
**REV 9**  
**2/3/78**

**MA OPCODE**

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 F8 00 B1 B2 B3 B5</td>
<td>Initialize high-order registers and byte pointers</td>
</tr>
<tr>
<td>06 B6 B8 A9 AB</td>
<td>Initialize bit pointers</td>
</tr>
<tr>
<td>0A F8 01 B9 BB BD</td>
<td>Initialize interrupt PC</td>
</tr>
<tr>
<td>0F F8 80 AC AA</td>
<td>Initialize stack pointer</td>
</tr>
<tr>
<td>13 F8 28 A1</td>
<td>Initialize MAIN PC</td>
</tr>
<tr>
<td>16 F8 FF A2</td>
<td>Initialize BNKWRIT PC</td>
</tr>
<tr>
<td>19 F8 66 A3</td>
<td>Initialize DOTWRIT PC</td>
</tr>
<tr>
<td>1C F8 5A A5</td>
<td>Initialize DELAY PC</td>
</tr>
<tr>
<td>22 F8 45 A8</td>
<td>Begin executing MAIN PC</td>
</tr>
</tbody>
</table>

**INTERRUPT**

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>26 72 70</td>
<td>Restore D, X, &amp; P</td>
</tr>
<tr>
<td>28 22 78 22 52</td>
<td>Push P, X, &amp; D onto stack</td>
</tr>
<tr>
<td>2C C4 C4 C4</td>
<td>NOP's for sync delay</td>
</tr>
<tr>
<td>2F F8 01 B0 F8 00 AO</td>
<td>Re-point R0 to display page</td>
</tr>
<tr>
<td>35 80 E2</td>
<td>Prepare for first DMA cycle</td>
</tr>
<tr>
<td>37 E2 20 A0</td>
<td>DMA reset</td>
</tr>
<tr>
<td>3A E2 20 A0</td>
<td>DMA reset</td>
</tr>
<tr>
<td>3D E2 20 A0</td>
<td>DMA reset</td>
</tr>
<tr>
<td>40 3C 35</td>
<td>Test for refresh done</td>
</tr>
<tr>
<td>42 30 26</td>
<td>Go to return</td>
</tr>
</tbody>
</table>

**DELAY**

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>44 D3</td>
<td>Return to MAIN</td>
</tr>
<tr>
<td>45 F8 07 BE</td>
<td>Load timing constant into RE</td>
</tr>
<tr>
<td>48 2E</td>
<td>Decrement RE</td>
</tr>
<tr>
<td>49 9E</td>
<td>Load RE.1 into accumulator</td>
</tr>
<tr>
<td>4A 3A 48</td>
<td>Loop again if not done</td>
</tr>
<tr>
<td>4C 30 44</td>
<td>Go to return</td>
</tr>
</tbody>
</table>

**DOTWRIT**

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4E D3</td>
<td>Return to MAIN</td>
</tr>
<tr>
<td>4F 89 AB</td>
<td>Update byte pointer</td>
</tr>
<tr>
<td>51 8A AC</td>
<td>Update bit pointer</td>
</tr>
<tr>
<td>53 EB</td>
<td>X=B</td>
</tr>
<tr>
<td>54 F1</td>
<td>Combine bit pointer &amp; screen via OR</td>
</tr>
<tr>
<td>55 5B E2</td>
<td>Write dot to screen</td>
</tr>
<tr>
<td>57 30 4E</td>
<td>Go to return</td>
</tr>
</tbody>
</table>

**BNKWRIT**

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>59 D3</td>
<td>Return to MAIN</td>
</tr>
<tr>
<td>5A 89 AB</td>
<td>Update byte pointer</td>
</tr>
<tr>
<td>5C 8A AC</td>
<td>Update bit pointer</td>
</tr>
<tr>
<td>5E FB FF</td>
<td>Inverts D via XOR IMMEDIATE</td>
</tr>
<tr>
<td>60 EB</td>
<td>X=B</td>
</tr>
<tr>
<td>61 F2</td>
<td>Combine bit pointer &amp; screen via AND</td>
</tr>
<tr>
<td>62 5B E2</td>
<td>Write blank to screen</td>
</tr>
<tr>
<td>64 30 59</td>
<td>Go to return</td>
</tr>
</tbody>
</table>
66 E2 69  Turn CDP 1861 on
68 3F 75  Skip clearing routine unless INPUT pressed
6A F8 FF AD  Point RD to top of display page
6D ED  X=D
6E F8 00 73  Store 00 on screen & decrement pointer
71 8D  Load pointer into D
72 3A 6E  Loop again if not done
74 5D  Store 00 in last byte of display page
75 E2 6C  Input toggles
77 F6 33 89  Tests "move right" bit & branches
7A F6 33 98  Tests "move left" bit & branches
7D F6 33 A7  Tests "move down" bit & branches
80 F6 33 AF  Tests "move up" bit & branches
83 F6 3B B7  Tests dot/blank bit
86 7B  Turn Q on
87 30 B7  Go to EXECUTE
89 BA  Store D in RA.1
8A 8A  Fetch temporary bit pointer
8B F6 33 92  Shift right and test for border cross
8E AA  Update bit pointer
8F 9A 30 7A  Put old D back in D & return to shift & test
92 19  Increment temporary byte pointer
93 76  Shift bit back into other end of bit pointer
94 AA  Update bit pointer
95 9A 30 7A  Put old D back in D & return to shift & test
98 BA  Store D in RA.1
99 8A  Fetch temporary bit pointer
9A FE 33 A1  Shift left and test for border cross
9D AA  Update bit pointer
9E 9A 30 7D  Put old D back in D & return to shift & test
A1 29  Decrement temporary byte pointer
A2 7E  Shift bit back into other end of bit pointer
A3 AA  Update bit pointer
A4 9A 30 7D  Put old D back in D & return to shift & test
A7 BA  Store D in RA.1
A8 89  Fetch temporary byte pointer
A9 FC 08  Add 08 to D & put sum in D
AB A9  Update byte pointer
AC 9A 30 80  Put old D back in D & return to shift & test
AF BA  Store D in RA.1
B0 89  Fetch temporary byte pointer
B1 FF 08  Subtract 08 from D & put difference in D
B3 A9  Update byte pointer
B4 9A 30 83  Put old D back in D & return to shift & test
EXECUTE

B7 D5 D8 D6 D8  Generate one "wink" of cursor
BB 31 C0      Go to M(CO) if Q is on
BD D6        Call DOTWRT & write on screen
BE 30 68 Go to test for clear
C0 D5      Call BNKWRT & write on screen
C1 7A      Turn Q off
C2 30 68 Go to test for clear

HOW TO USE THIS PROGRAM: Load it carefully through the toggle
switches. No peripherals but the Pixie video chip and monitor are
required. All control is through the toggles and the INPUT switch.
When the program is run, memory page 01 is displayed on the screen.
Normally, just after power-up, this page will contain random bits
scattered throughout, especially if it is CMOS RAM. To begin with a
clear (all black) screen, hold INPUT depressed while flipping the RUN
switch up. The routine at M(6A-70) will store 00 in all memory
locations within page 01, blanking the screen. The screen can be
cleared at any time by pushing INPUT.

When run, a winking cursor will appear in the extreme upper left
hand corner of the screen, corresponding to bit 7 of M(01 00). By
flipping the first four toggle switches up and down, the cursor will
move around the screen, leaving a "trail" of white behind it. If the
fifth toggle is flipped up, the cursor will "eat" white areas on the
screen by writing 0 bits behind it. The cursor appearance is always
the same, regardless of its "polarity." To remove the cursor from the
middle of a complicated design without disturbing the screen pattern,
reset the program by flipping RUN down and up again. The cursor will
be sent back to its home position. The screen pattern will not
change.

The clearing routine can be modified to fill the screen with a
white background to draw upon in black by changing the byte at M(6F)
to FF. Toggle 0 makes the cursor move to the right. Toggle 1 makes
it move to the left. Toggle 2 makes it move downward. Toggle 3 makes
it move upward. If the cursor is sent off any side of the screen, it
will appear immediately on the opposite side. Kaleidoscopic displays
can be created by greatly increasing the speed of the program. The
delay subroutine should be re-written thus:

D3 F8 01 AE 2E 8E 3A 48 30 44

The cursor will then move at great speed, and by randomly flipping the
direction-control toggles a crazy-quilt pattern will continually weave
on the screen.

Enjoy it!
A CRITICAL LOOK AT THE COSMAC VIP
AN ANIMATED VIDEO FACE SYSTEM
AN ELF ADAPTER BOARD FOR THE VIP
AN EPROM CARD FOR THE VIP
AN UNBUFFERED MORSE KEYBOARD PROGRAM
BINARY ARITHMETIC SUBROUTINE BOOK
CD vs. D version 1802's
CMOS RAM BARGAIN
COSMAC I/O PORTS
COSMAC PRICE BREAKTHROUGH
COSMAC'S FAULT FLAW
COSMO'S FACE
Disabling ELF RAM
Disabling interrupts
Eastman 910 warning
ELF address display circuit
ELF/VIP software compatibility
ELF-II expansion bus pinout
EPILOGUE
EVERYTHING YOU NEVER WANTED TO KNOW ABOUT PAPER TAPE
FAST/SLOW CLOCK GENERATOR
GIVE YOURSELF ROOM
“Gerdenia" Super Sound listing
H-10 paper tape reader/punch
HAMFESTS
Hex keyboard circuit
HOLY FAST RAM?
HOW MUCH RAM?
I/O strobe expansion using 4026
IN SEARCH OF COSMAC STANDARDS
Interrupt enable flipflop (IE)
INTERRUPTS MADE SIMPLE
IO-BUS
IT MAY OR MAY NOT COME IN THE MAIL
Loading large programs
Lookup tables
Mail order reviews
Morse keyboard/keyer program
OP-40A bootstrap loader program
OP-60A paper tape reader
Prog' o of Cosmo the Robot
PIN-8
Plastic “consumer” version 1800 chips
POWER SUPPLY POINTERS
Program debugging with memory address displays
Programmer's paranoia
RAM CHOICES
RAM speed vs. clock speed table
Recom-p
Reed relay driver from Q
Relocatable ROM block
Robot face displays
Robot face program
Sample interrupt routine
Source code vs. object code
Stack
State code lines (SCO & SC1)
STUDIO II CONVERSION
Super Sound block diagram
SUPER SOUND REDUX
TEHOPS (Tape Entered Hex Op System)
TETOPS (Tape Entered Toggle Op System)
THE SOURCE WILL ALWAYS BE WITH YOU
THE 1900 MASS STORAGE PROBLEM
TickTokTech ad
Toggle switch warning
UNDER NO CIRCUMSTANCES
Vancing bootstrap paper tape loader program
VERSION, VOLTAGE, & SPEED
Video scratchpad program
VIP expansion bus pinouts & description
VIP I/O line summary
WATCH OUT
VIP I/O port
WHEREZAT PROGRAM
WIRE WRAPPING
WIRING WOES
X-BUS
ZOUNDS!
Zounds program
1802 speed limitations
1804 & 1806 rumors
1852 I/O port
1863
2114 RAM circuit & pinout
2758 sound circuit & pinout
4515 decoder/latch pinout
4556 decoder pinout

EPILOGUE—Look, don't be fooled. I've tried my goddamnest to make the 1802 sound attractive, and to make messing around with computers on a fundamental level seem irresistible. I've cautiously omitted tales of torn hair involving bad solder joints on connector cables, leaky bypass caps, and program glitches that must be due to a bad CPU chip. (That hasn't happened yet. I've been waiting for three years...) I've omitted accounts of dodging Unidentified Flying Pads for spending too much time crouched over a piece of perfboard crawling with bugs. If you have Elves in your mouse, all of this should be familiar. If you do not yet have an ELF—well, the entire ELF story is available in reprint from Popular Electronics Magazine; write to them for particulars. You really ought to build one. Really.

I built my first ELF in 1976, and started writing this book in 1977. Here it is, the first month of 1980. Where do we go from here? Well, there's the tale of the CDP1804. Chrysler Corporation asked RCA to design a single-chip microcomputer to handle their new emission control system. RCA saw dollar signs, sat down, and came up with the 1804. On one chip are 2K ROM, 64 bytes RAM, and a programmable counter/timer that can measure the length of pulses, generate square waves, count events, and generate processor interrupts. Best of all, the 1804 has 33 new instructions, all two-byte opcodes beginning with 68. Now you know what RCA was saving that mysterious opcode 68 for. There is a single instruction to load both halves of a register with immediate data, or data from R(X). You can now copy an entire register to R(X) with a single instruction. You can disable or enable interrupts by setting IE directly with a single instruction that affects nothing else. All housekeeping for Standard Call & Return are now handled by two instructions. Beautiful. Now then, the only snag is that Chrysler is about to go broke, leaving RCA with ten million 1804's, all mask programmed to count smog particles coming out of exhaust manifolds. Even if Chrysler survives, RCA will have its hands so full making Chrysler's 1804's that it won't be putting the chip into general release. The only way you'll be able to get one is by taking your Dodge Aspen apart. So it goes.

A marginally reliable source says that Hughes Corporation is designing the CDP1806, which will be upwardly compatible with both the 1802 and the 1804. Since there are now 223 unused '65 opcodes, that leaves them a lot of room. No bets on what the 80's will bring. An 18167? A CMOS 2764? No bets, guys. No bets at all.

If anybody wonders what you do with your evenings, just tell them what I do: Come on over and meet my computer some day.

It's the one with the pointed ears.
CHIP 8 DISPLAY

Reprinted from the RCA Engineer
### CHIP 8 - Program Sheet

**Program Name:**

<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reprinted from the RCA Engineer
THE 1802 SOURCE!

Ne'er was there a well so deep...

WHEN your microprocessor is #5, you don't just try harder, you have to dig halfway down to the Mono for parts, hardware, and software. I've done some heavy legwork in the last 1802 years, so it might be neiborly to lay out what I've found so it can be useful. This is the best list of COSMAC suppiers I've been able to put together. It's in no particular order. Use it in good health.

NETRONICS R&D LIMITED; 333 Litchfield Rd. New Milford, CT 06776; Phone: (203) 354-9575. This is the home of the ELF-II, an excellent kit-form 1802 machine. Lots of hardware and software add-ons available, including ASCII keyboard, color, music synthesis, BASIC, light pen, editor/assembler/disassembler, and lots more coming up. Famous for using smaller print in their ads than I use in this book, but that's cool.

QUEST ELECTRONICS; 20 NETRONICS R&D PO Box 95054; Phone: (408) 988-1640. Quest is a super-fine mail-order outfit for parts in general. Their Super ELF kit is as good as or better than the ELF-II. It has a ROM monitor and richer I/O, not as cool software at this time. Quest also publishes Questdata, a monthly software newsletter for 1802 users, for $12/year. Until very recently Quest also advertised a basic ELF kit for about $50, but no longer available. The more you see of the Super ELF, the better I like it. Get their catalog, and Questdata.

ANACRON; PO Box 2206P, Culver City CA 90230; Phone: (213) 641-4064. Anacron now carries a full line of the plastic (LE) series of 1802 developers. The 1802 is $950; the 1802B 1802 is $990; the 1802S 1802 is $1,900; the 1802U 1802 is $1,980. For my money the plastic case is better than the ceramic; the legs don't fall off as easily. This is the place for individual chips.

H. C. WILL SOFTWARE; PO Box 347, Pinebrook NJ 07518. This is a new one, and I have no experience with him, but he sells VP and possibly ELF software. Send him a SASE for his program list. Can't hurt to try.

CREASEO; 6303 Golden Hook, Columbia MD 21044.CREASEO won't say what its damned acronymic name means, but it's a tremendous outfit that publishes the PER computer board oriented to work you can do. It sells several boards relating both to computer board oriented to war d everything of his I've seen has been terrific. Send for a list.

COSMAC USER'S GROUP; PO Box 7162, LA CA 90062. Watch for out this one; I've received only three issues spaced out over a year and a half, and nothing since May '79. The publisher, Patrick Kelly, sounds funnier than what's going on. He blamed the long delay between two of the issues on an amnesia attack following shock treatment for severe depression. Yeah, I sympathize, but what the hell's passable newsletter when it shows up. Caveat Elfhacker.

DIGITAL SERVICE & DESIGN; PO Box 741, Newark OH 43055. Wyndham Davies is in charge. Unless he's gone bankrupt or gotten divorced with us, DSAD makes the best of the 1802 sound system, including 16K RAM card, 32K EPROM card, I/O network card (whatever that is) and floppy disk controller card (I've never had one of these cards can we had for $125. I know nothing more about him at this time, again, it never hurts to write and ask. Tell him you saw it in Captain Cosmo's Whizbang.

CUDDLY SOFTWARE; 26 Thorndale Terrace, Rochester NY 14611; Phone: (716) 326-8259. Ne, this is not me, though he's scarcely three miles up the road. What he sells, at this time, are two mighty fine programs, an OP System and a Trace Program. The OP System is a collection of subroutines that forms a screen resolution and moves bytes around. I can't do it justice here; you should write for the specs. The Trace Program is the intriguing one, allowing single-stepping or a program with dynamic display of processor registers after each step! It requires an ASCII keyboard and 3K minimum, but it sounds jammed useful. No price info at this time. Check it out!

O. C. STAFFORD ELECTRONICS; 427 S. Benbow Road Greensboro NC 27401. Ozie Stafford is a ham and a great guy to deal with. He sells an 1802 computer board oriented toward ham radio and repeater control, plus lots of other circuit board relaying both to computers and ham radio. Everything of his I've seen has been terrific. Send for a list.

MICRO-SYSTEMS; 801 Cedar Circle, Somersport NY 14559. Lee Hart, proprietor. Lee sells what is undoubtedly the most sophisticated piece of 1802 engineering I've ever seen: The SSC100-X1 single board 1802 computer. For a tiny list of by 5-inch card he's got 2K RAM, up to 4K ROM, 2 latched parallel ports, 2 optically isolated serial ports for RS232 or 20 mil current loop, bus terminator resistors, logic for multiplexed I/O lines, and miscellaneous whatnot. It requires some onboard 1/0 to talk to, but it can be jumpered out almost any way. It has an excellent monitor, IDIOT4, is available on ROM, and Lee will soon make FORTH available. This board is not for the beginner in the same way the ELF is, but if you've made an ELF work you can handle it. Utterly, utterly first class!

RCA COSMAC VIP MARKETING; New Holland Avenue, Lancaster PA 17604. Yes, kids, you can in fact order VIP products direct from RCA. I did it, and it took 2 whole months for the stuff to get in. Not an enviable track record, but it can be done. RCA sells Super Sound, Simple Sound. ASCII Keyboard (nice, by the way, and cheap) Color, Expansion Board, and some other odds and ends. Send for the VIP list for current prices. RCA also sells a COSMAC evaluation kit for about 400 bucks, which is a big board with a TTY interface and room for 4K RAM. I don't think it's worth the money, but suit yourself. The Microtutor II is an ELF but not as nice, and I whole lot more expensive. RCA also sells big mainframes using the 1802 in the 5-10 kilobuck range. If you're that rich you shouldn't be reading trash like this anyway. The whole, except for the VIP expansion boards, you can do much better elsewhere.

OPTIMAL TECHNOLOGY; Blue Wood 127, Barlyville VA 22924; Phone: (504) 978-6842. Also sells 1802 EPROM programmers with software for the 1802. I've heard nothing more about the devices though, and have been itching to order one for some time. Write for specs and a catalog is about all I can say.
THE INSIDE STORY:

ANATOMY OF THE 1802

Jeff Duntemann
301 Susquehanna
Rochester NY 14618
Hi, yawll:

After skulking around in the underground for more than a year, Captain Cosmo's Whizbang has finally made the big time with a Real Book Review in Kilobaud courtesy satisfied reader Larry Stone. Orders are coming in once again, long after I had pretty much given up hope of selling another copy. Many of you have written asking the price, well, the price was on the review when Larry sold it, but Wayne Green (who seems not to believe in Free Plugs for kitchen table operations) corflued it out. Thank you all for writing. Let's see what I can tell you.

CCW is still available. $5.00 plus .75 postage. Handling free, as is the grape jelly on the envelopes resulting from the handling. I mean, this is a real Kitchen Table Operation. Although I don't take credit cards, checks are cool, as is cash, for that matter. My mailroom staff is honest, since my mailroom staff is me, and I long ago learned not to cheat at solitaire. In fact, since there's no such thing as a $5.75 bill, if you send a five-spot the postage is on the house.

For those of you who haven't run across a Whizbang yet, it's probably unlike any other publication in the hobbyist world. Like all the rest of us, I started out in microcomputers as a total ignoramus, with the difference that I was an ignoramus with a long memory. I remembered what it was like to be stuck up a familiar creek with no stick and nobody to run to for help, and I hoped to be able to ease a few of you over some of the hard spots.

Parts of CCW are over three years old now, and the micro world has aged some. Would you believe I paid $39.95 for my first 1802? (Still have the little bugger, too.) Today, a good scrounger with a wire-wrap gun can make an unbelievably cheap computer. 2114 RAM is now three or four bucks a shot--cheaper than a first run movie. There is still no better, cheaper way to get into computers than via the 1802. Order the Popular Electronics 'Elf' reprint series first, and CCW second, and you'll be on your way.

What am I working on these days? PASCAL, mostly. That, and the 8086 on the S100 bus, and other various and sundry. Why did I abandon the 1802? Money. I make more money on a single computer-magazine article than I did on the whole CCW project. That, and somebody else gets to worry about mailing things out and screwups at the printers'. On the other hand, nobody else would have allowed me to do CCW in their magazine as I wanted to do it--so for that alone, it was worth the effort.

What new 1802 gossip? That's the problem--there isn't any. RCA seems to be abandoning its problem child, and the 1804 seems to have sunk like the Titanic. People are flocking to National Semi's CMOS
Z80 lookalike. The future of the 1802 looks kinda grim from here.

But there is yet one bright spot in the 1802 scene: TMSI. This is the same outfit I listed in CCW as Micro Systems, but they changed their name and moved to Michigan not long after CCW went to press. Since then their single board 1802 computer has gone through several revisions and is now the BASYS/1, and is truly remarkable. The future of the 1802 is with threaded code languages, like FORTH and TMSI's similar 8TH. Hang BASIC—that's for kids. Get yourself a BASYS and try 8TH—it is truly the ONLY 1802 machine I unhesitatingly recommend.

So--see what you can do with the 1802. As always, you're pretty much on your own. The magic you create will be of your own making—but that's the very best kind of magic there is. Best to you all—it's been fun. Read the computer mags and you'll bump into me now and then, but to be honest, you'll never hear me say things quite the same way I said them in CCW.

Cheers, and keep on hackin'--

Jeff Duntemann KB2JN

---

Carol and Jeff Duntemann
301 Susquehanna Rd.
Rochester, NY 14618