VIPER

VOLUME, ISSUE 8 & 9 AN ARESCO PUBLICATION MARCH/APRIL 1980, $4.00

TABLE OF CONTENTS*

*Due to the huge number of articles in this issue, the table of contents is continued on the back of this page.

TABLE OF CONTENTS (continued)

THE VIPER is published ten times per year by ARESCO, Inc. 6303 Golden Hook, Columbia MD 21044 and mailed to subscribers on the last day of each month except June and December.

Subscription price is $20 for all 10 issues (all past and future issues) of the current volume. Non-USA subscribers should add $10 for air-mail delivery.

Second Class Postage paid in Columbia MD 21045. USPS:502-550 ISSN: 0199-1566. POSTMASTER: Please send all address changes to ARESCO, P O Box 1142, Columbia MD 21044.

Readers should NOT correspond with RCA concerning VIPER material.

EDITORIAL "THINK DIGITAL"

by Tom Swan

I've been sitting here staring out the window, occasionally adding my entry in a paper airplane contest with a little boy downstairs who was hurt playing soccer. If only the big ciruela tree were a few feet more to the right and if only I could get a little more lift, I bet I could keep a tiny paper plane up for over a minute considering how high we are from the ground.

Then I asked myself a question I ask at least 20 times a day. "Could a computer help?" Well sure. You would have to calculate paper size, weight, lift, design, things like that and maybe a computer could help lead to a superior paper airplane. On second thought, I have more important things to do so I suppose I'll stick with the design my brother David showed me. Still a VIP could graphically plot wing shapes, etc. It's an intriguing idea.

I am sure there are plenty of VIPers -- and other computerniks -- sitting in front of their video screens mentally scratching the insides of their craniums for something to do with the things. Just as I am positive there are thousands of writers staring out their windows wondering what to say next. (Can you hear me clearing my throat just now?)

What's a good way to come up with programming ideas? The best way, I think, is to ask that question "Could a computer help?" Ask it even if it seems silly. Look sideways and upside down at things. But always "think digital" and keep THE QUESTION in the back of your mind. Next time a paper airplane flies past you'll find yourself running for the run switch, a light bulb (LED?) burning brightly above you.

Naturally I think the VIPER is an excellent source of programming inspiration too. That's why we're doing what we do, but to really keep the light bulb lit, we need to hear from you.

Starting today, I'll keep a list of ideas sent in by readers to be published in full a couple of months from now. If everyone sends in one idea for a program that they don't have the time or experience to write for themselves, we'd end up with a digital pool of ideas just waiting to be keyed hexadecimally into memory. I think this would be preferable and for sure more creative and productive than for me or anyone to write the whole list. Then, if you find an idea you'd like to tackle, we can publish the results here.

Remember, think digital!

I read Phil Sumner's letter this month (see Reader I/O) with great concern. Phil wrote to say he noticed a shift from general interest articles in the VIPER toward PIPS related material. Since I am the author of the series and am now editing the VIPER as well, I fear that Phil and other readers may get the erroneous impression that Tom Swan is somehow "taking over."

In no way is this true. I certainly didn't accept the position of editor with any intention of furthering my own special interests. The reason I got involved with the VIP in the first place was because it aims at an audience generally interested in learning more about computers whatever a person's special interest may be. The VIPER will continue to focus the bulk of its articles on that audience.

There is another side to the question. I have noticed several past issues of the VIPER which were slanted toward other special interests, so much so that they were not of much use to me either! For example, a hardware packed newsletter does little for someone who occasionally forgets which end of the soldering iron to hold. I am an incapable fumble fingers with a circuit design and articles that discuss electronics in detail just aren't my cup of tea. But the articles deserve to be published.

The same holds true for other concerns. Last month's features were definitely slanted toward owners of the Tiny Basic Board. The October issue was particularly hardware oriented. There have been music issues. Others may have been weighted heavily toward 1802 machine language business and I seem to remember a time when the Elf computer held the spotlight. (And VIPER readers nipped that one in the bud properly!)

But what's the answer? Should we refuse to publish an article unless it is totally general in nature? I hope not! VIPers are a diversified lot and the newsletter must necessarily reflect the varied interests of its readership. The fact is: there will always be some material in any publication which some readers may not find interesting or useful or even comprehensible. To someone else, that very same article may contain answers to questions long since given up as lost causes. Damned if you do and damned if you don't publish it.

I do agree, however, that no special interest -- hardware, assembly language or PIPS -- should be allowed to gain the upper hand. The purpose of this editorial is to assure Phil and other non-PIPS readers that I will pay special attention to including general interest material in each and every issue of the VIPER. Each issue should have something potentially useful to everyone.

Also, I am a great believer in the adage that the facts speak for themselves. To that end, I conducted my own survey of the last four issues. Just how much material has been published which requires some special knowledge, hardware or book in order to be used? I define general interest as any article containing software

that will run on an unmodified VIP without a keyboard or other plug in peripheral. (I did not consider memory limitation since 4K VIPS are practically the standard now.) A special interest article is the opposite of that, and of course PIPS related articles are of an obvious nature. Reviews, readers letters and advertisement were not included. Though most of these are probably general in interest they are not always general in subject and would only confuse the results. Here are the results of the survey for VIPER, Vol 2, issues 4,5,6, and 7.

PIPS related articles	- 3	per cent of total	- 15.8%
General interest	- 10	per cent of total	- 52.6%
Other Special interest	- 6	per cent of total	- 31.6%
TOTAL	- 19	TOTAL	- 100. %

There seems to be approximately 50% of general interest material and 50% of special interest material in the last four issues. (52.6% vs. 47.4% to be exact.) In the February issue alone, the results are:

PIPS related articles	- 1	per cent of total	- 20%
General interest	- 3	per cent of total	- 60%
Other Special interest	- 1	per cent of total	- 20%
TOTAL	- 5	TOTAL	- 100%

The percentages are less significant for a single issue because it is a case of 20 per cent or nothing for a single article out of five.

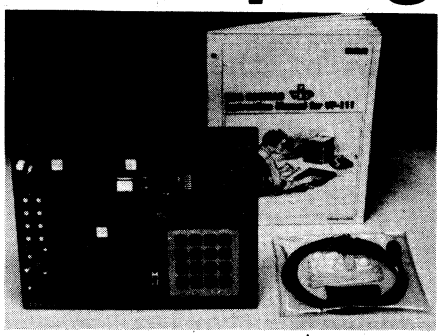
I'd appreciate some feedback from other readers on this. What do you think of the balance presented in VIPER and how should the balance be kept? How much on one subject is too much? Or do you want to cut out all special interests here? Let's hear your comments and try to arrive at a workable solution.

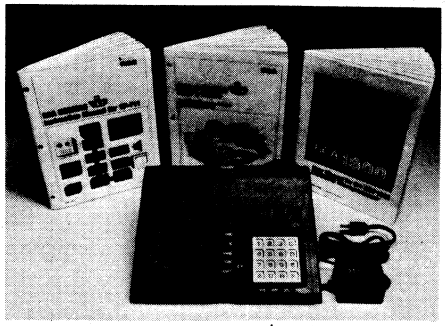
To Phil Sumner, thanks are in order for pointing out a weakness in our publishing policies. Though we are not engaged in any active shift toward any special interest group, it is true that the balances are left swinging in the wind. If the wind doesn't happen to blow your way then you're out in the cold and that's wrong.

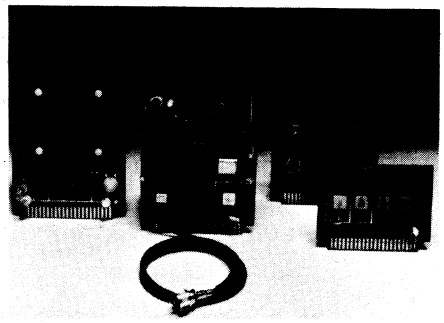
One final comment is to remind everyone that I don't want to write most of the material myself. We are more than happy to publish all the general interest articles we receive -- but we have to receive them to publish them.

We all look forward to hearing your comments and are confident that the outcome will be a better newsletter for all!

The VIP hobby computer: Start programming for only $99.







New! VP 111 Microcomputer.... $99. Assembled* and tested.

Features:

- RCA 1802 Microprocessor.
- 1K Bytes static RAM.

Expandable on-board to 4K.
Expandable to 32K Bytes total.

- 512 Byte ROM operating system.
- CHIP-8 interpretive language or machine language programmable.
- Hexidecimal keypad.
- Audio tone generator.
- Single 5-volt operation.
- Video output to monitor or modulator.
- Cassette interface—100 Bytes/sec.
- Instruction Manual with 5 video game listings, schematics, CHIP-8, much more!

Ideal for low-cost control applications.

Expandable to full VIP capability with VP-114 Kit.

*User need only connect cables (included), a 5-volt power supply, and speaker.

New low price! The original VIP.. $199. Completely assembled and tested.

All the features of the VP-111 plus:

- A total of 2K Bytes static RAM.
- Power supply.
- 8 Bit input port.
- 8 Bit output port.
- I/O port connector.
- System expansion connector.
- Built-in speaker.
- Plastic cover.

Three comprehensive manuals:

- VIP Instruction Manual—20 video game listings, schematics, much more.
- VIP User's Guide—operating instructions and CHIP-8 for the beginner.
- RCA 1802 User's Manual (MPM-201B) - complete 1802 reference guide.

COSMAC VIP lets you add computer power a board at a time.

With easy-to-buy options, the versatile RCA COSMAC VIP means even more excitement. More challenges in graphics, games and control functions. For everyone, from youngster to serious hobbyist.

Built around an RCA COSMAC microprocessor, the VIP is easy to program and operate. Powerful CHIP-8 interpretive language gets you into programming the first evening. Complete documentation provided.

Send the coupon now...
Complete the coupon below and mail to: RCA VIP Customer Service, New Holland Avenue, Lancaster, PA 17604.

Or call toll free (800) 233-0094
to place your Master Charge or VISA credit card order. In Pennsylvania, call (717) 397-7661, extension 3179.

RCA

Please send me the RCA COSMAC VIP items indicated.

- ☐ **VP-111** New low cost Microcomputer (See description above) **$99**
- ☐ **VP-114** Expansion Kit for VP-111—Includes 3K RAM, I/O Port and connectors... **$ 76**
- ☐ **VP-711** VIP—The original VIP Microcomputer (See description above) **$199**
- ☐ **VP-44** RAM On-Board Expansion Kit—Four 2114 RAM IC's. Expands VP 711 memory to 4K bytes ..... **$ 36**
- ☐ **VP-590** VIP Color Board—Converts VIP to color. Four background and eight foreground colors..... **$ 69**
- ☐ **VP-595** VIP Simple Sound Board—Provides 256 programmable frequencies. For simple music or sound effects. Includes speaker ..... **$ 30**
- ☐ **VP-550** VIP Super Sound Board—Turns your VIP into a music synthesizer! Two independent sound channels. On-board tempo control. Outputs to audio system..... **$ 49**
- ☐ **VP-551** 4-Channel Super Sound—Includes VP-576 expander, demo cassette and manual. Requires VP-550 and 4K RAM ..... **$ 74**
- ☐ **VP-570** VIP Memory Expansion Board—Plug-in 4K RAM memory..... **$ 95**
- ☐ **VP-580** VIP Auxiliary Keypad—Adds two-player interactive capability. 16-key keypad with cable. Connects to sockets on VP-590 or VP-585..... **$ 20**
- ☐ **VP-585** VIP Keypad Interface Board—Interfaces two VP-580 Auxiliary Keypads to VIP..... **$ 15**
- ☐ **VP-560** VIP EPROM Board—Interfaces two 2716 EPROMs to VIP..... **$ 34**
- ☐ **VP-565** VIP EPROM Programmer Board—Programs 2716 EPROMs. With software..... **$ 99**
- ☐ **VP-575** VIP Expansion Board—Provides 4 buffered and one unbuffered expansion sockets..... **$ 59**
- ☐ **VP-576** VIP Two-Board Expander—Allows use of 2 Accessory Boards in either I/O or Expansion Socket..... **$ 20**
- ☐ **VP-601** ASCII Keyboard—128-character ASCII Encoded alphanumeric keyboard..... **$ 65**
- ☐ **VP-611** ASCII/Numeric Keyboard—Same as VP-601 plus 16 key numeric keypad..... **$ 80**
- ☐ **VP-620** Cable: Connects ASCII keyboards to VIP..... **$ 20**
- ☐ **VP-700** VIP Tiny BASIC ROM Board—BASIC code stored in 4K of ROM..... **$ 39**
- ☐ **VP-710** VIP Game Manual—Listing for 16 exciting games..... **$ 10**
- ☐ **VP-720** VIP Game Manual-II—More exciting games (Available 2nd qtr. '80) ..... **$ 10**
- ☐ **MPM-201B** CDP1802 User Manual—(Included with VP-711)..... **$ 5**
- ☐ Please send more information..... —

Enclosed is $_____ for items checked plus shipping & handling charge of $3.00.
Add your state and local taxes $_____ Total enclosed $ _____
I enclose ☐ check or ☐ money order or, charge my ☐ VISA/Bank Americard
☐ Master Charge.

Credit card account No. _____
Master Charge Interbank No. _____ Expiration date: _____

Signature (required for credit card orders): _____
Name (please type or print): _____
Street address: _____ City: _____
State & Zip: _____ Telephone: ( ) _____

Make checks payable to RCA Corp. Prices and specifications are subject to change without notice.

2.08/09.06

Dear VIPER - My recent telephone conversation with you confirmed my suspicions that the parity bit from the new RCA keyboard creates problems in Tom Swan's ASSEMBLER-3. Your suggestion to cut the MSB lead will fix the problem but I have an inherent dislike (chicken-heart) of cutting foil on printed circuits.

Applying the following software changes to Tom's EDITOR-21, modification for keyboard 1 or 2 will also do the job.

For Modification #1

Address		
0031	30	Branch to
0032	C8	00C8
0033	00	- spare -

For Modification #2

Address	
002D	30
002E	C8
002F	00

For both Modifications

```
Address
00C8        FE      SHL ;Set MSB to 0
00C9        F6      SHR
00CA        52      STR ;Put corrected code in stack
00CB        B8      PHI ;Put in R8.1
00CC        30      BR  ;Branch to 00B3
00CD        B3
```

The program is taking the input byte that is in D register and shifting it left to drop the MSB. It is then shifted right to restore the original data except that a Ø is placed in the MSB.

Since the original byte was stored in the stack (R2), the corrected byte needs to replace it and this is done at address 00CA.

It is possible to make this modification by adding only 3 bytes but it requires some rearrangement of the existing program.

I assume you have the modification from RCA on adding a DPDT switch to the Tiny BASIC board so that CHIP-8 can be used without pulling the board. I did make that modification (drilled holes, cut foil, laid down with a wet rag on my head) and it works very well. - Bob DeHaven

Dear Bob - RCA pulled a switcheroo on their original design and my modifications were constructed on the prototype RCA keyboard that Rick Simpson sent me. The same problem may show up in my "Step by Step" article and elsewhere when ASCII codes are compared in software. In the interest of keeping standards standard, I will continue to use normal ASCII codes in my programs that need them. Another possible fix would be to logically AND all input characters with hexadecimal $7F to set the most significant bit to zero. Frankly, I don't see the purpose of the parity bit anyway, and if you are not squeamish about cutting a foil trace, I suggest you do it. - Tom

Dear Rick and Terry:

I had hoped to write this letter in an optimistic mood, with constructive comments, inputs of my own, or maybe even a new article (yes, I'm still thinking of that!). But over the last few months, I have read each new issue of VIPER with a growing sense of dissatisfaction; with the arrival of the February issue, those feelings turned into dismay.

Whether you realize it or not, the VIPER has largely turned into something that I cannot use, and neither can other people like me. As a relative beginner, I passed up the opportunity for the PIPS series; I wasn't interested in becoming a programmer, and in fact am just now learning machine language programming. But now I find that the VIPER is shifting so that PIPS FOR VIPS is a necessity if I'm to use the VIPER material. The perfect example of this is the Killer Robots game, which uses many of the routines from PIPS. But I can't use it, and neither can anyone else who <u>didn't</u> buy PIPS.

I think you have some hard decisions to make. Either continue as you seem to be going, turn the VIPER into a programmer's newsletter, and accept the inevitable shrinkage in circulation as people decline to renew. Or shift the VIPER back to running a respectable amount of general interest articles and information that are usable to <u>all</u> your readers, not just the programming elite. By definition, this would mean that <u>some</u> articles and programs must be revisions or extensions of previously published material (VIPER or RCA manuals), or they must be complete and self-contained. Care to put the vote to your readership??

Note that you run the same risk in printing programs for the accessory boards such as super sound. That would also bother me, but I haven't noticed enough of that to be really objectionable. There usually was something in that issue for me also. I guess my basic beef is that the something for me has gotten less and less lately.
Enough of the negative stuff -- time to think positive. I basically like the newsletter in its present form, with contents out front and regular features. I do have reservations about Tom Swan as the editor -- the newsletter may lean even more toward programmers as a result; the newsletter's also in very real danger of becoming a one man show; and I basically feel that editors should mainly editorialize and only occasionally write. But I'm more than willing to wait and see how it works out.

To get back on the positive track, let me hit you (and whoever else you care to invite to the "party") with a couple of ideas (or downright challenges, if that's what it takes to get someone interested).

1. It's about time that the VIP became useful for something other than a toy, game, or hobby. The next most logical thing to do with the VIP is to make it into an educational device, especially for math -- why not now? The software multiply and divide routines are available; so are routines for displaying any combination of digits. Conversion

between hex and decimals may be a problem when handling large numbers, but that "problem" should be solvable. Handling of negative numbers may be more of a problem. Why can't some of our elite programmers put such a package together? Or maybe put at least part of it together, such as handling negative decimal numbers?

2. There is a definite dearth of logically written explanatory material on machine language programming. Why can't someone put together a manual on the topic? The machine language equivalent of the VIP User's Guide that Terry did? I would be very willing to collaborate on such a manual -- I think I can contribute in some ways, and can act as a trial filter for most of the rest. And if you wait long enough, I might be able to do the whole thing. Some of the natural pieces for such a manual are surfacing in my work now; I'm the Astro Electronics instructor for the Microprocessor System Design laboratory course. Rick is probably familiar with C51/CL51, the fundamentals course; what I'm teaching is CL56, which goes much deeper into a lot of interesting topics, like machine language. Individual pieces of the manual could be published in VIPER as they are generated, then they could all be collected, reorganized, and edited to produce the manual. I have the first draft of such a piece now, something called "A Universal Machine Language Branching Technique", which discusses a handy software technique that every real programmer knows but every beginner has to find out the hard way. And there are other potential topics right behind it.

This letter has gotten too long already, yet! For the next one, I won't wait so long! Thanx for everything - Phil Sumner

Phil - You're absolutely right. And we hadn't even noticed the shift! It is people like you, who give us substantial constructive criticisms, offering new ideas & solutions, that keep VIPER on track! Thank you.

I see VIPER as an information exchange. People write about things they're interested in. We "pretty it up" and make a lot of copies, then send copies to anyone who is willing to share the printing & mailing costs. - Terry

Dear VIPER - I have a VIP with 4K & Color, I am also a ham radio operator. I would like to interface the VIP to a circuit to be able to receive morse code and display it with the VIP. I am not sure where the output of this A/D converter would connect to the VIP. My knowledge of machine language is not all that good to be able to convert their coding to the VIP. If you or anybody you know of could help me on this it would be greatly appreciated. I would think this could be of great value to many VIP users as well as the many hams that could be interested. Thank you. - Ted Bratcher

Dear Ted - Sorry but I'm not a ham (though some would disagree!) In your letter you also included a copy of a circuit taken from REMark magazine. Because I assume that diagram is copyrighted by the publisher of REMark, I did not include it here as you requested. Still, if any readers respond with similar interests, I'll be happy to help get you together with them. Thanks for writing! - Tom

Dear VIPER - I'm very impressed with your publication. I soon hope to publish my VIP-1802 mini-assembler in the VIPER.

The assembler is designed to work in a standard VIP using the hexpad. This means that hexadecimal op codes must still be used. This plus the lack of any macro capability means I haven't written a true assembler. My mini-assembler will, however, allow the writing of (Ahhhh) fully relocatable source code! The assembler will support the use of labels and/or absolute addressing. All branches and subroutine calls to labels will be resolved. In addition, code can be assembled to run anyplace in RAM using a simplified ORG directive. Keep up the good work. - Steven Blasnik (P.S. Tom Swan's relocatable PC change was beautiful!)

Dear Steven - We look forward to receiving your assembler. - Tom

Dear VIPER - Sam Hersh's Editor is especially nice but should be relocated to the highest page with a jump to it at 0200. Then you can run a program in CHIP-8 by loading all but the first two bytes at 0202, checking and editing then replacing the jump-to at 0200 with the first two bytes and run. (Maybe it would take a long branch and four bytes, but you get the idea).

Also, I have a VIP for sale. It includes 2K RAM, Tiny BASIC, improved RESET-RUN switch, 2 digit hex display on outport port, 8 bit binary LED display for output, Cortes audio output amp, switch for Tiny BASIC, all cables for TV & cassette, AC power supply, a stack of documentation 3 to 4 inches thick (including all VIPER issues), & original shipping cartons. All this for a $200 cashier's check.

One more thing. Tiny BASIC needs an OUT statement/command. (all BASICs are weak on this, I think). Who says you'll never want to get something out on the output port? I don't favor any schemes for POKEing around with machine language routines to accomplish this. I'm working on a hardware fix for it. Anyone interested? - Don Hartley

Dear VIPER - I am experiencing some difficulty with my VP-700 Tiny BASIC. Sometimes my machine runs through a 60-Key loop without a key being depressed. I wonder if any of your readers have had a similar problem and found a solution?

Keep up the good work and thanks for any help. - Jerry Krizek

Dear VIPER - My system is a home brew 1802 resembling the ELF, however, I have made modifications to it as well as the CHIP-8 such that my machine is indistinguishable from a VIP. I saw my first VIP a few weeks ago and was pleased with myself as to my simulation.

Your newsletter (magazine) is a constant source of enjoyment both for me and non-computerist wife. At some point I will send you my CHIP-8 version of ONE-DIMENSIONAL LIFE (BYTE, 3:12, p68 (1978)) and PAPER, SCISSORS, STONE.

I have implemented Charlie McCarthy's (VIPER, May '79) Hi-resolution graphics and was very pleased with it. I hope to see some software support in Volume 2 or to contact Mr. McCarthy directly. - William Gilbert

Dear William - At one time I had contemplated writing a CHIP-8 version of LIFE but ended with a machine version because of speed. Thanks for your letter and we look forward to seeing your programs. - Tom

Dear VIPER - In the Nov. '79 VIPER (2.05), Tom Swan reviewed Tiny BASIC and commented that there were no provisions for machine language additions. I received one of the early boards (it only cost me $29!) and had the same frustration. I called RCA, talked with Joe Rudy, and discovered that there was a way to get at least limited machine language capability. Andy Modla of RCA Princeton Labs had written a way to achieve this; I called him and got the instructions with the understanding that he intended to send it to VIPER to be published.

I have used this approach with my Tiny BASIC board, and it works! I get the feeling that these directions could be modified to provide a multiple number of machine language subroutines, but I haven't had time to look into it. Perhaps someone out there might want to dig into this further. - Randy Holt

Dear Randy - Someone already has! See the article "More Tiny BASIC Machine Language Subs" by C.D. Smith in this issue. - Tom

Dear VIPER - Having recently installed a home brew RAM board, and stomping on a fair share of hardware bugs in the process, I think the following modifications to the Memory Test Program in the VIP Manual (VIP-300, p.32) may be useful to 1802-ers in testing erratic memories.

Changing the error-trap to provide a dynamic, rather than static, display can reveal "iffy" address lines. This change flashes the error location continuously, and the tone is suppressed since having hardware problems is annoying enough!

006C 07 FB FF 30 6C (last byte @ 0071)

In my case, this revealed multiple simultaneous access within the 1K under test; it can be determined from the screen which MA lines are faulty.

Not trapping errors is also useful in observing overall behavior in the 1K under test, revealing faulty address lines, un-enabled chips, or chips not write-enabled, or which fail to respond to signals. These failures are observed as vertical solid lines corresponding to the bit-width of the chip (4 bits for 2114s, 1 for 2101s, etc.) for enable problems. Dead address lines will show as horizontal lines; a combination of enable and address problems will show as squares or cross-hatch. To disable the trap, change 0054 to '38'.

Fortunately, my board is running nicely, thank you; with the exception of 1-2114 with a bad bit, all problems were bad connections, either soldered, or the more insidious pin-to-socket contact.
- P. V. Piescik, Cuddly Software

A LETTER TO PIPS III OWNERS

from Tom Swan

Well, it's humble pie time. When I wrote the Reversi type game VIP-FLOP for PIPS FOR VIPS III, I thought I knew how to play the game. I have a good friend who used to tell me, "Tom, never assume that you know anything." (Roy -- you were right again.)

Not having the box version down here in Burro Land, I relied on a sketchy Creative Computing article and one match played at a computer show as the basis for the game rules. As many of you have rightly pointed out, I made one lovely boo-boo. My head hangs down in shame and I print here one of Terry Laudereau's famous "sighhhh's" -- a long'n.

My mistake was to assume that moves only need to be adjacent to any piece in order to be counted as legal. The correct rules require that at least one opponent piece be flipped for the move to be legal. When there are no legal moves possible, the player must forfeit the turn.

Now for the good news.

CARMELO CORTES TO THE RESCUE!

(con't)

Dear Rick and Terry,

I've enclosed my corrections to the VIP-FLOP game. Also I've included a routine that allows you or the computer to forfeit a turn when there are no legal moves left.

Unfortunately, because of lack of room you can no longer ask the computer to suggest a move for you. Now when you press Key C, you are forfeiting your turn.

I hope Tom comes up with a better change than mine, this way I can have my forfeit and computer advice too!

Thank you

Carmelo Cortes

Thanks, Carmelo. The modifications listed below work fine and the only improvement I can think of is to, as you suggest, make the forfeit automatic so that Key C will work as before. I'll give it a try, and if I have any success, publish the results next month.

They say "To err is human." Sometimes I wish I was a computer! (Guess which one.)

CHANGES TO VIP-FLOP

```
036C  4A FF
  6E  15 ØE
  70  13 8C
 --
0390  13 EA
 --
03EA  4A FF 13 F2 24 D4 13 92
  F2  26 C4 13 66
03F6  4A FF 13 C8 34 Ø1 13 BØ
  FE  13 B2
 --
03AE  13 F6
 --
03D2  13 E6
 --
0877  F6
 --
087C  BD Ø7 AD F8 FF AF F8 FF
  84  AE F8 24 A3 D3 ØA FB Ø1
  8C  32 9E 3Ø 94 F8 24 A3 D3
  94  ØA FB 8Ø 32 9Ø ØA FB Ø1
  9C  32 FB 9D 56 8D 57 1E 8E
  A4  FB Ø2 3A 85 1F 8F FB Ø2
  AC  3A 82 3Ø F6
 --
08F6  F8 E1 A3 D3 DC F8 E6 A3
  FE  D3 DC
 --
09E1  F8 FF A6 56 D4 F8 FF A6
  E9  F8 ØØ 56 D4
```

PRELIMINARY DATA

Floating Point BASIC Interpreter

for the VP-711 MicroComputer

VP-701 BASIC is a full-sized BASIC interpreter for the VP-711 or Expanded VP-111 MicroComputer. This BASIC includes features that up till now were only dreamed about by VP-711 owners: one and two dimensional arrays; string variables string functions; machine language subroutine calls; plus FLOATING POINT MATH with trigonometric and transcendental functions.

VP-701 BASIC contains over 70 statements and functions including custom tailored commands for use with the VP-590 Color Board and the VP-595 Simple Sound Board. VP-701 BASIC provides video mapping of 16 characters by 11 lines.

VP-701 BASIC is RAM-resident, requiring a minimum of 16K bytes of memory for the interpreter plus RAM for program area. VP-701 BASIC comes on cassette tape and includes a user guide.

NUMBER RANGES

Floating Point: $\pm$ 1.7014 x 10 $^{\pm 38}$, 32 bit (approximately 6 decimal digits).

Integer: $\pm$ 2,147,483,647, 32 bit (10 decimal digits).

VARIABLE TYPES

Numeric (Simple) A - Z

Numeric (Dimensioned) A(1) - Z (256,256)
DIM Statement required. i.e. DIM A(100, 10)
Size of arrays are limited by memory.

String A$ - Z$
Up to 96 characters each.

FLOATING POINT VS INTEGER

All numeric variables default to floating point. To define variables as integer use DEFINT X statement which sets all variables through variable X to integer. To change variables back to floating point reissue the DEFINT command. (DEFINT with no variable resets all variables to FLT. PT.)

During input VP-701 BASIC converts all numeric entries into an internal format to facilitate program execution.

GRAPHICS

VP-701 BASIC also contains special commands to utilize graphics and color.

The video display is arranged as a rectangle having 64 elements horizontally and 128 elements vertically. The graphic commands COLOR and SHOW have their origin (Ø, Ø) at the upper left-hand corner of the screen.

The PLOT command has its origin (Ø, Ø) at the lower left-hand corner to facilitate an easy transition from ordinary cartesian coordinate geometry to the video screen surface.

RCA MicroComputer Products|New Holland Avenue
Lancaster, PA 17604

2.08/09.14

COLOR

When used with the VP-590 Color Board, VP-701 BASIC can program the video field for color. The color statement by itself will sequence through 4 background colors. When parameters are included after the color statement, the foreground colors can be set to the colors listed below. The color field is broken up into 8 blocks horizontally and 4 mirrored 32 bit blocks vertically.

COLOR CODES

0 - Black	4 - Green
1 - Red	5 - Yellow
2 - Blue	6 - Cyan
3 - Magenta	7 - White

VOCABULARY

ABS	GET	PRINTER ON
ASC	GOKEY	PRINTER OFF
ATN	GOSUB	PSAVE
CALL	GOTO	PT
CHR$	HIT	READ
CLC	IF	REM
CLD	INPUT	RESTORE
CLS	INT	RETURN
COLOR	INUM	RND
COS	KEY	RUN
DATA	LEN	SGN
DEFINT	LET	SHOW
DEFUS	LIST	SIN
DIM	LOG	SQR
DLOAD	MEM	STEP
DSAVE	MID$	STORE
END	NEW	TAB
EOD	NEXT	TIMER
EOP	PEEK	TIME
EXIT	PI	TONE
EXP	PLOAD	TVOFF
FIXED	PLOT	TVON
FNUM	POKE	USR
FOR	PRINT	WAIT
FREQ	PRINTAT	

CHIP-8E

by Gilles Detillieux

It seems that everyone has come up with extensions to CHIP-8. Many of these would be very useful but obviously they could not all be incorporated in a 512 byte interpreter. This gives rise to many incompatible systems.

The approach I took was to rewrite the interpreter, taking the following points into consideration:

1) It should be compatible with all previously written CHIP-8 programs. Execution should begin at 0200 and all instructions should remain unchanged (with the exception of BMMM and 004B which are rarely if ever used in any CHIP-8 programs*.)

2) It should be readily adaptable to various 1802 machines.

3) It should incorporate I/O instructions.

4) The added instructions should be useful enough to warrant their implementation in the interpreter. Less useful ones (or those used less often) can be handled as machine language subroutines.

My interpreter, which I call CHIP-8E (for extended), features the following new instructions:

00ED	STOP Replaces filler at location 00ED with instruction 23 DEC R3. Since R3 is the program counter, execution stops at this point.
0151	WAIT FOR TIME UP Examines content of timer and exits once timer equals 00
00F2	NO OPERATION Useful for deleting instructions in a program. Executes a D4 SEP R4 at 00F2, which makes it return to fetch the next instruction.
0188	SKIP Unconditionally skips next two byte instruction.
5XY1	SKIP IF VX > VY Skips next two byte instruction if VX is greater than VY.

---

*So far, the only CHIP-8 program I have seen which uses an instruction not available in CHIP-8E is "VIP Bowling" in the VIP Game Manual. This program contains a B53A instruction at 0538. In this case, the new F01B instruction can directly replace it.

5XY2 MI = VX:VY
Transfers variables X through Y to memory. Works like FX55.

5XY3 VX:VY = MI
Load variables X through Y with data from memory. Works like FX65.

BBMM BRANCH BACKWARD MM BYTES
Go to current instruction location minus MM. (See VIPER 2.01.11)

BFMM BRANCH FORWARD MM BYTES
Go to current instruction location plus MM. (See VIPER 2.01.11) (BB00 and BF00 are non-terminating loops)

FX03 OUTPUT = VX
Content of VX sent to output port 3.

FX1B SKIP VX BYTES
Skips amount of bytes indicated by content of VX. If VX is 00, the next instruction is executed. (Substitute for old BMMM)

FX4F TIME = VX; WAIT FOR TIME UP
Sets timer value of VX then waits at 0151 until timer is 00.

FXE3 VX = INPUT
Waits for strobe at $\overline{EF4}$ then reads content of input port 3.

FXE7 VX = INPUT
Reads content of input port 3 without waiting for strobe.

The listing for CHIP-8E, (listing I), is for the VIP. For use with a different 1802 system, modification will be required.

If you are using the system described by Bobby Lewis, make the changes shown in listing II.

If you have a keypad at input port 4, and an IN button,(like the ELF II and SUPER ELF) make the changes shown in listing III.

If you have a keypad at input port 4 which provides a key-pressed strobe at $\overline{EF3}$ make the changes shown in listing III, using 3E instead of 3F and 36 instead of 37.

If your keypad is different than the systems mentioned above, you will have to rewrite instructions FX0A at locations 010A to 0114, and EX9E/A1 at locations 019F to 01AD.

I hope this version of CHIP-8 will satisfy the needs of most people, and perhaps introduce a measure of standardization for extended CHIP-8 interpreters.

Listing I CHIP-8E

```
0000  C4 91 BB FF 01 B2 B6 F8 CF A2 F8 00 A5 F8 02 B5
0010  F8 81 B1 F8 46 A1 90 B4 F8 1D A4 30 E0 E2 69 96
0020  B7 E2 94 BC 05 F6 F6 F6 F6 32 42 FE FC 45 AC 45
0030  F9 F0 A6 05 F6 F6 F6 F6 F9 F0 A7 4C B3 0C A3 D3
```

(Cont.)

```
0040  30 1F 45 B3 45 30 3E 00 FA 00 F3 01 83 01 8B 01
0050  75 00 D7 00 DA 01 AE 01 91 01 CA 01 EA 01 D1 00
0060  65 01 9F 01 01 06 FA 07 BE 06 FA 3F F6 F6 F6 22
0070  52 07 FE FE FE F1 AC 9B BC 9A BF 8A AF 45 FA 0F
0080  AD A7 F8 D0 A6 93 B7 87 32 A3 27 4F BD 9E AE 8E
0090  32 9B 9D F6 BD 97 76 B7 2E 30 8F 9D 56 16 97 56
00A0  16 30 85 00 EC F8 D0 A6 93 A7 8D 32 D0 2D 06 F2
00B0  32 B5 F8 01 A7 46 F3 5C 02 FB 07 32 C9 1C 06 F2
00C0  32 C5 F8 01 A7 06 F3 5C 2C 16 8C FC 08 AC 3B AA
00D0  F8 FF A6 87 56 12 D4 45 56 D4 45 E6 F4 56 D4 00
00E0  9B BF F8 FF AF 94 5F 8F 32 F2 2F 30 E5 23 42 B5
00F0  42 A5 D4 15 85 22 73 95 52 25 45 A5 86 FA 0F B5
0100  D4 45 A3 E6 63 26 D4 98 56 D4 F8 81 BC F8 95 AC
0110  22 DC 12 56 D4 06 B8 D4 06 A8 D4 E6 15 30 EE E6
0120  8A F4 AA 9A 7C 00 BA D4 00 F8 81 BA 06 FA 0F AA
0130  0A AA D4 1A 1A EA F8 FF AE AF 06 FF 64 1F 33 3B
0140  FC 64 FF 0A 1E 33 42 FC 0A 73 8E 73 8F 5A D4 06
0150  B8 98 3A 51 D4 22 86 52 F8 F0 A7 07 5A 87 F3 17
0160  1A 3A 5B 12 D4 22 86 52 F8 F0 A7 0A 57 87 F3 17
0170  1A 3A 6B 12 D4 45 76 76 33 95 7E 07 3B 84 E6 F7
0180  3B 88 D4 45 E6 F3 3A 82 15 15 D4 45 E6 F3 3A 88
0190  D4 45 07 30 8C 7E 22 87 52 86 A7 33 6B 30 5B 45
01A0  F6 E6 62 26 33 A9 36 88 D4 3E 88 D4 00 00 45 FA
01B0  0F 3A B6 07 56 D4 AF 22 F8 D3 73 8F F9 F0 52 E6
01C0  07 D2 56 F8 FF A6 94 7E 56 D4 45 AA 86 FA 0F BA
01D0  D4 19 89 AE 93 BE 99 EE F4 56 76 E6 F4 B9 56 45
01E0  F2 56 D4 3F E3 37 E5 E6 6B D4 E5 86 FC 01 85 3B
01F0  F9 F4 A5 95 7C 00 B5 25 D4 F7 A5 95 7F 00 30 F6
```

Listing II CHIP-8E Modifications
(for system described by Bobby Lewis, VIPER, March '79)

```
0000  F8 07  (CHIP-8 patch no longer required)
0011  0E
010B  0E
012A  0E
01A2  67
01A6  37
01A9  3F
```

Listing III CHIP-8E Modifications
(for ELF II and SUPER ELF)

```
0000  F8 0X  (0X is display page)
0011  --     (High address for Interrupt routine)
0014  --     (Low address for Interrupt routine)
010A  E6 3F 0B 37 0D 6C FA 0F 56 D4 00
012A  --     (High address for Pattern table)
01A1  3F AA 22 6C 06 F3 12 FA 0F 33 8E 30 86
```

INVERT

by Robert Lindley

The VIP will select nine random digits, one through nine, and display them. These numbers are not duplicated, each appears exactly once. The object of the game is to try to get your line of digits in order before the VIP puts its line in order.

A possible starting display might be:

```
2 7 9 3 6 8 1 5 4  YOU
2 7 9 3 6 8 1 5 4  ME
I N V E R T
```

If you press key 6, the new display is:

```
6 3 9 7 2 8 1 5 4  YOU
9 7 2 3 6 8 1 5 4  ME
I N V E R T 6
```

As illustrated, everything from the left end of the row to the given key number is inverted. Also the VIP selects its own inversion. The selection made by the VIP uses an efficient algorithm, but it does not take full advantage of natural order. In the case of a tie -- you win.

The program was coded in a simple straight forward manner. The main program is first and is mostly calls to subroutines to do the work. There is one unique thing in the program, it uses a pair of subroutines (SAVE and RESTR) to save and restore the memory pointer. This neatly solves the problem of using one pointer for both data array reference and display reference.

LISTING

```
Ø2ØØ  START:   64ØØ -- Signal first pass of game
  Ø2           221E -- Do RNDM
  Ø4           226Ø -- Do SHINV
  Ø6  NEXT:    23Ø2 -- Do SHROW
  Ø8           2288 -- Do WINLS
  ØA           3CØØ -- Test for start new game
  ØC           12ØØ -- Go to START
  ØE           22DC -- Do GETIN
  1Ø           23Ø2 -- Do SHROW
  12           A3F8 -- YOUR
  14           2326 -- Do DOINV
  16           2376 -- Do MYCHS
  18           A4Ø8 -- MINE
  1A           2326 -- Do DOINV
  1C           12Ø6 -- Go to NEXT
  1E  RNDM:    A418 -- TEMP
```

```
Ø22Ø             6ØØØ -- Start loop count
  22  R1:        FØ55 -- Put Ø to 9 in array temp
  24             7ØØ1 -- Increment loop count
  26             3ØØA -- Exit on 10 count
  28             1222 -- Go to R1
  2A             6ØØØ -- Set zero to store
  2C             61ØA -- Start loop at next location
  2E  R2:        FØ55 -- Zero out array locations A to F
Ø23Ø             71Ø1 -- Increment loop
  32             311Ø -- Exit on 17 count
  34             122E -- Go to R2
  36             6BØ1 -- Start loop count
  38  R3:        CCØF -- Get random # Ø to F
  3A             A418 -- Temp - restore array origin
  3C             FC1E -- Add offset
  3E             FØ65 -- Fetch random array element
Ø24Ø             4ØØØ -- Try again if zero in array
  42             1238 -- Go to R3
  44             A4Ø8 -- MINE - Computer array
  46             FB1E -- Add offset
  48             FØ55 -- Store
  4A             A3F8 -- YOUR - Player array
  4C             FB1E -- Add offset
  4E             FØ55 -- Store
Ø25Ø             A418 -- Temp - Scratch pad array
  52             FC1E -- Add random offset
  54             6ØØØ -- Zero to mark as selected
  56             FØ55 -- Store
  58             7BØ1 -- Increment loop
  5A             3BØA -- Exit loop on 10 count
  5C             1238 -- Go to R3
  5E             ØØEE -- Return - 9 random # selected
Ø26Ø  SHINV:     6DØØ -- X screen location
  62             6E1Ø -- Y screen location
  64             A3DA -- I
  66             DDE5
  68             7DØ6
  6A             A3EØ -- N
  6C             DDE5
  6E             7DØ6
Ø27Ø             A3E6 -- V
  72             DDE5
  74             7DØ6
  76             A3C2 -- E
  78             DDE5
  7A             7DØ6
  7C             A3EC -- R
  7E             DDE5
Ø28Ø             7DØ6
  82             A3F2 -- T
  84             DDE5
  86             ØØEE -- Return
  88  WINLS:     6DØØ -- X screen location
  8A             6E18 -- Y screen location
  8C             3BØ1 -- VB=1 means player wins
  8E             12A6 -- Go to MYWIN
```

```
Ø29Ø            22C8 -- Do SHYOU
  92            A3D4 -- W
  94            DDE5
  96            7DØ6
  98            A3DA -- I
  9A            DDE5
  9C            7DØ6
  9E            A3EØ -- N
Ø2AØ            DDE5
  A2            8CBØ -- Indicate end of game
  A4            12C2 -- Go to WXIT
  A6  MYWIN:    3CØ1 -- VC=1 means computer wins
  A8            ØØEE -- Return for no winner
  AA            22C8 -- Do SHYOU
  AC            A3C8 -- L
  AE            DDE5
Ø2BØ            7DØ6
  B2            A3BØ -- O
  B4            DDE5
  B6            7DØ6
  B8            A3CE -- S
  BA            DDE5
  BC            7DØ6
  BE            A3C2 -- E
Ø2CØ            DDE5
  C2  WXIT:     FØØA -- Wait for key - start new game
  C4            ØØEØ -- Erase screen
  C6            ØØEE -- Return
  C8  SHYOU:    A3AA -- Y
  CA            DDE5
  CC            7DØ6
  CE            A3BØ -- O
Ø2DØ            DDE5
  D2            7DØ6
  D4            A3B6 -- U
  D6            DDE5
  D8            7DØC
  DA            ØØEE -- Return
  DC  GETIN:    6D28 -- X screen position
  DE            6E1Ø -- Y screen position
Ø2EØ            44ØØ -- Do not erase on first pass
  E2            12E8 -- Go to INPUT
  E4            F429
  E6            DDE5 -- Erase prior input
  E8  INPUT:    F2ØA -- Wait for input key
  EA            42ØØ -- Try again if zero
  EC            12E8 -- Go to INPUT
  EE            8320 -- V3-V2
Ø2FØ            72Ø6 -- Add 6
  F2            64FØ
  F4            8242
  F6            32ØØ -- Skip if key < 10
  F8            12E8 -- Go to INPUT
  FA            F329
  FC            DDE5 -- Show current input
  FE            8434
```

```
Ø3ØØ             ØØEE -- Return
  Ø2  SHROW:     6DØØ -- X screen position
  Ø4             6EØØ -- Y screen position
  Ø6             A3F8 -- YOUR
  Ø8             2356 -- Do DISPN
  ØA             8BCØ
  ØC             7DØ2
  ØE             22C8 -- Do SHYOU
Ø31Ø             6DØØ -- X screen position
  12             6EØ8 -- Y screen position
  14             A4Ø8 -- MINE
  16             2356 -- Do DISPN
  18             7DØ2
  1A             A3BC -- M
  1C             DDE5
  1E             7DØ6
Ø32Ø             A3C2 -- E
  22             DDE5
  24             ØØEE -- Return
  26  DOINV:     Ø392 -- Do save - save memory pointer
  28             61ØØ -- Start loop counter
  2A  DO1:       71Ø1 -- Increment counter
  2C             Ø39E -- Do RESTR - restore memory pointer
  2E             F11E -- Add array offset
Ø33Ø             FØ65 -- Fetch element
  32             A418 -- Temp - array origin
  34             F11E -- Add array offset
  36             FØ55 -- Store in scratch array
  38             5Ø3Ø -- Exit on number match
  3A             132A -- Go to DO1
  3C             62ØØ -- V2=Ø
  3E             71Ø1 -- Increment array pointer
Ø34Ø  DO2:       72Ø1 -- Increment destination pointer
  42             71FF -- Decrement array pointer
  44             A418 -- Temp array origin
  46             F21E -- Add offset
  48             FØ65 -- Fetch array element
  4A             Ø39E -- Do RESTR - restore memory pointer
  4C             F11E -- Add offset
  4E             FØ55 -- Store element
Ø35Ø             31Ø1 -- Test for end of count down
  52             134Ø -- Go to DO2
  54             ØØEE -- Return
  56  DISPN:     6CØ1 -- Set winner flag
  58             61Ø1 -- Set array pointer - loop count
  5A             F11E -- Add offset
  5C             Ø392 -- Do SAVE - save memory pointer
  5E  DS:        Ø39E -- Do RESTR - restore memory pointer
Ø36Ø             FØ65 -- Fetch array element
  62             Ø392 -- Do SAVE - save memory pointer
  64             5Ø1Ø -- Test for number match
  66             6CØØ -- Reset winner flag
  68             FØ29
  6A             DDE5 -- Show array number
  6C             7DØ5
  6E             71Ø1 -- Increment loop count
```

```
Ø37Ø             31ØA -- Test for end of array
  72             135E -- Go to DS
  74             ØØEE -- Return
  76  MYCHS:     63ØA -- Point to top of array
  78  MY:        73FF -- Decrement pointer
  7A             A4Ø8 -- MINE - array origin
  7C             F31E -- Add offset
  7E             FØ65 -- Fetch element
Ø38Ø             9Ø3Ø -- Look for sequence mismatch
  82             1378 -- Go to MY
  84             81ØØ -- V1=VØ
  86             A4Ø8 -- MINE - array origin
  88             FØ65 -- Skip over element zero
  8A             FØ65 -- Fetch element one
  8C             9Ø3Ø -- Look for mismatch
  8E             831Ø -- Invert at next sequential #
Ø39Ø             ØØEE -- Return
  92  SAVE:      E2   -- SEX 2
  93             96   -- GHI 6
  94             BF   -- PHI F
  95             F89Ø -- LDI 9Ø
  97             AF   -- PLO F
  98             9A   -- GHI A
  99             5F   -- STR F
  9A             1F   -- INC F
  9B             8A   -- GLO A
  9C             5F   -- STR F
  9D             D4   -- SEP 4
  9E  RESTR:     E2   -- SEX 2
  9F             96   -- GHI 6
Ø3AØ             BF   -- PHI F
  A1             F89Ø -- LDI 9Ø
  A3             AF   -- PLO F
  A4             4F   -- LDA F
  A5             BA   -- PHI A
  A6             4F   -- LDA F
  A7             AA   -- PLO A
  A8             D4   -- SEP 4
  A9             ØØ
  AA  Y:         885Ø
  AC             2Ø2Ø
  AE             2ØØØ
Ø3BØ  O:         7Ø88
  B2             8888
  B4             7ØØØ
  B6  U:         8888
  B8             8888
  BA             78ØØ
  BC  M:         88D8
  BE             A888
Ø3CØ             88ØØ
  C2  E:         F88Ø
  C4             FØ8Ø
  C6             F8ØØ
  C8  L:         8Ø8Ø
```

```
Ø3CA            8Ø8Ø
  CC            F8ØØ
  CE  S:        788Ø
Ø3DØ            7ØØ8
  D2            FØØØ
  D4  W:        8888
  D6            A8A8
  D8            5ØØØ
  DA  I:        7Ø2Ø
  DC            2Ø2Ø
  DE            7ØØØ
Ø3EØ  N:        88C8
  E2            A898
  E4            88ØØ
  E6  V:        8888
  E8            5Ø5Ø
  EA            2ØØØ
  EC  R:        F888
  EE            F8AØ
Ø3FØ            9ØØØ
  F2  T:        F82Ø
  F4            2Ø2Ø
  F6            2ØØØ
  F8  YOUR:
Ø4Ø8  MINE:
  18  TEMP:
```

PROGRAMMING HINTS

by H.C. Will IV

"Patching" is a term used by programmers when they wish to get something they missed into their program. This is accomplished by doing a jump (or call) to a location where the missing instructions are stuck in. This is useful when you missed one of the first instructions.

After completing a program, save it on tape and then use one of the VIP drawing programs to create a message or picture that you want to be displayed. Reload your program and save it and the display page on tape. From then on, whenever you load that program from tape, your message or picture will be displayed at the end of the load. Remember, when designing your display, that you must leave room at the bottom of the screen for the monitor to display the address and the contents of that address. Since the display page must be saved along with the program in one shot, this method will cause all recordings done this way to be the same length (i.e. 8 pages for a 2K RAM VIP).

Hint - when you don't know how many pages to load of a program on tape, try loading 15(F) pages. Even if the program is only one or two pages long, the entire length will be properly read in. Stop the tape when the tape light and Q signal come on. - Tom

SOFTWARE CHANGES AND OTHER GOOD STUFF

by George Ziniewicz

BMMM in CHIP-8

To allow any variable to be the index for the branch, change:

01A4 from F8 to E7
A5 FØ to 8D

Then using John Bennett's FXF2 code in Vol 1, issue 10 (page 7) you can control the starting variable for FX55-65 instructions and let any variable be added to the BMMM address.

The following sequence will work:

FXF2 -- RD.0=VX
BMMM -- Where X = variable to index with

EYE, EYE CAPTAIN

Give the Figure Shooting at Moving Target better chances of scoring a hit by giving him an eye.

02AC from 7C to 74

CHIN UP!

Correct the face in Dot Dash

04CC from 84 to 86

REVERSE VIDEO

Here is a MLS (Machine language subroutine) to swap all bits white to black and vice versa. Just call with the instruction ØMØØ where M is the page address of the MLS. The routine can be entered into any memory page beginning at byte ØØ.

```
ØMØØ  9B    REVVD: GHI  RB     ;Load display page(s) start
  01  BC           PHI  RC
  02  F8 01        LDI  #1     ;Load # pages (01 or 02)
  04  AD           PLO  RD
  05  F8 00        LDI  #0     ;Load display start address
  07  AC           PLO  RC
  08  0C    REVV1: LDN  RC     ;Get display byte
```

```
ØMØ9  FB FF       XRI  #$FF    ;Invert
  0B  5C          STR  RC      ;Put back
  0C  1C          INC  RC
  0D  8C          GLO  RC
  0E  3A 08       BNZ  REVV1   ;Loop till done
  10  2D          DEC  RD
  11  8D          GLO  RD
  12  3A 08       BNZ  REVV1   ;All pages done?
  14  D4          SEP  R4      ;Return
```

Some of the things I'm working on include:

1) A hardware Reverse Video for full screen instantaneous reversal. It will use three chips and four bytes of software.

2) CHIP-8 compatible music (approx. 30 bytes) with the syntax: A___; point to music string/Ø___; go to music MLS. No display off commands required and one byte includes frequency and duration ending the string with ØØ. A 60hz interrupt beat provides interesting effects. Also, there are 16 preprogrammed frequencies in the look up table.

3) CHIP-8 single-dual-etc. Text display with control commands. (Less than 2 pages including character map). Included will be 46 characters with home, line feed, carriage return, tone, delay, wait for key, and user defined controls.

4) Time Lapse Controller for electrically operated movie cameras.

5) Simple control for up to 8 electrical appliances.

6) Adventure style game where you hunt for treasure.

7) Simple CHIP-8 Character Designer

8) Color Organ, Standard CHIP-8 color control, real Kaliedoscope.

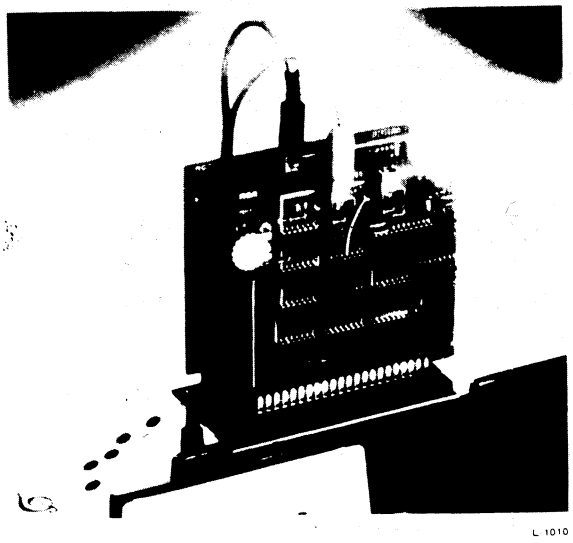
9) MLS library

EDITOR'S NOTE: George, you really have some good ideas. Thanks for the modifications and the above inspirations! - Tom

VP-551

Four-Channel Super Sound
Expansion Package

Four Channel Music Synthesis For The COSMAC MicroComputer



L 1010

Four Independent Sound Channels
Note Frequency, Duration & Envelope Control
Data Cassette With Two Music Programs
Four Octave Range
Versatile Four-Channel Software

The VP-551 Four Channel Super Sound Package includes a VP-551 circuit board, a VP-576 Two Board Expander, a data cassette with two music programs and an Instruction Manual.

You will be impressed with the music playing capability of the COSMAC MicroComputer Super Sound with this added four channel synthesis. Each channel can be programmed for frequency and note envelope independently. You can create four part harmony or a melody with chord accompaniment or the illusion of several instruments playing at once. Tempo is adjustable with an on-board potentiometer.

The all new four channel software allows you to transpose your songs up or down over a two octave range a half step at a time with just the stroke of a key for each half step! But that's not all . . . program up to eight songs at one time and select any song, select any sequence or randomly play any of the programmed songs. You can continuously repeat the songs or you can play them through once.

The carefully written, well organized manual shows you how. It includes a source code listing, music tables and addresses, and as an additional bonus, the classic, Fuga of Bach, as a programming example.

Operation of Four Channel Super Sound requires your VP-711 or expanded VP-111 MicroComputer, your VP-550 Super Sound board and any stereo system.

RCA **MicroComputer Products**

2.08/09.26

LIFE

by Tom Swan

Plenty has been written about the game of LIFE. For references see Scientific American, October 1970 where the game first appeared in Martin Gardner's column on page 120. (I was lucky to find a mint condition copy of this issue only last year tucked away in a corner of a used book store. I held my breath as I paid the too-good-to-be-true price of 50¢ or so.) If you can find a copy, check out the RCA ad on page 55. It's a public relations thing about "... a monster. The computer." entitled "Are they for us or against us?" Computers do "exactly what they are told," says the ad. Oh yeah?

John Conway invented LIFE. Computers proved that Conway's simple rules of living and dying have practically unlimited research possibilities. Each cell in the two dimensional world is surrounded by up to eight neighbors. The conditions of those neighboring cells plus the cell under consideration determine whether that cell will live, die or be born on the next generation of <u>all</u> cells.

The rules are:

1) Survival -- if a living cell has exactly two or three neighbors it will go on living
2) Deaths -- if a living cell has four or more neighbors it will die from overcrowding. If a living cell has one or no neighbors it will die of loneliness. (Can you stand it?)
3) Births -- if a non-living (empty) cell is surrounded by exactly three neighbors, a living cell will be born in its place. Else it will go on non living in the next generation.

The VIP display is the world of LIFE. In this version the world is actually spherical with each edge and corner joining the opposite. This provides complete wrap around, and some unfortunate collisions occasionally. White bits represent living cells while dark bits are the non living or empty ones.

When you flip to run, any pattern previously placed into memory page $0400-$04FF will be acted on according to the rules of LIFE. You are actually seeing two separate pages because $0500-$05FF is used for page switching to smooth out the animation. The program will run with 2K of memory or more.

You may enter patterns using the write mode of the VIP operating system to deposit cells into the world at $0400. A better way is to use one of the video drawing programs. The one in the VIP manual will do, but you must enter the following change so that the drawing is done on page 4.

0238 from 03 to 04

Even better would be to combine the two programs, but this has given us hours of fun in this form. Actually I forgot about this program and just found it on the bottom of a pile of papers. It was my

first machine language program written about the same time that the VIPER was having its birth pains, springing into LIFE.

LISTING

```
0000  90 B3 A5 AA AC AF F8 81 B1 F8 46 A1 F8 02 B2 B4
  10  B6 BD F8 04 B7 BA BB BE F8 05 BF F8 FF A2 E2 F8
  20  31 A4 F8 11 A6 F8 01 AD F8 2C A3 D3 69 38 1F F8
  30  00 5F 8F FB FF 3A 2E AF 8A FF 09 A7 DD 8A FC 07
  40  A7 DD 8A FF 01 A7 DD 8A FC 01 A7 F8 08 AD DD 8A
  50  FF 07 A7 F8 08 AD DD 8A FC 09 A7 F8 08 AD DD 8A
  60  FF 08 A7 8A FC 08 AE 95 FE FE D6 B5 F8 15 A6 07
  70  D6 F8 15 A6 0E D6 F8 35 A4 0A D4 F8 51 A4 D4 F8
  80  31 A4 07 D6 0E D6 0A D4 F8 51 A4 D4 F8 31 A4 07
  90  FE D6 0E FE D6 0A FE D4 F8 51 A4 D4 F8 31 A4 07
  A0  FE FE D6 0E FE FE D6 0A FE FE D4 F8 51 A4 D4 F8
  B0  31 A4 07 FE FE FE D6 0E FE FE FE D6 0A FE FE FE
  C0  D4 F8 51 A4 D4 F8 31 A4 07 FE FE FE FE D6 0E FE
  D0  FE FE FE D6 0A FE FE FE FE D4 F8 51 A4 D4 F8 31
  E0  A4 07 76 76 76 76 D6 0E 76 76 76 76 D6 0A 76 76
  F0  76 76 D4 F8 51 A4 D4 F8 31 A4 95 D6 F8 15 A6 07

0100  76 76 76 D6 F8 15 A6 0E 76 76 76 D6 0A F6 3B 11
  10  15 F6 3B 15 1C F8 51 A4 D4 F8 31 A4 8A FC 01 33
  20  40 1A 1F 30 35 00 00 00 00 00 F8 00 B3 00 00 00
  30  00 00 00 00 00 F8 00 B3 00 00 00 00 00 00 00 00
  40  9A 73 9F 73 60 72 BA B7 BE BB F0 BF F8 00 AA AF
  50  30 2A 00 00 00 00 00 00
0158 - 01FF -- not used

0200  D3 07 F6 95 7E B5 30 00 07 FE 30 03 00 0C 00 00
  10  D3 FE 3B 15 1C FE 3B 19 1C FE 3B 10 1C 30 10 00
  20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  30  D3 FE 3B 35 1C FE 3B 39 15 FE 3B 30 1C 30 30 00

0250  D3 85 FB 00 32 5B 8C FB 02 32 60 8C FB 03 3A 6B
  60  0F FE FC 01 5F F8 00 A5 AC 30 50 0F FE 30 64
```

PROGRAMMING HINTS

by H.C. Will IV

Record (in voice) on tape, before each program, the program name, number of pages and the place where documentation and/or directions can be found (this aids in the location and use of taped programs and the VIP Monitor overlooks peoples voices).

When debugging CHIP-8 programs put a DØØF instruction in place of a current instruction to see if it is being executed. If the instruction is being executed, junk will appear on the screen. This instruction can also be followed by a "jump to itself" instruction. This type of debugging aid is helpful when trying to find out which instructions are being executed, if they are being executed and in what order they are being executed. It also aids in finding out whether "SKIP" instructions are "SKIP"ing or not. A jump to the beginning (or end) of the program can also provide the same results. An even more interesting implementation of this debugging tool is to call an assembly language subroutine that in turn does a jump to the monitor (8ØØØ) so that you can look at where the CHIP-8 variables are stored. This "tool" is referred to as a breakpoint.

SPACE WARS SPEED UP

by Tom Swan

The following modifications to Space Wars in PIPS FOR VIPS I will cause the phasor fire to go faster. It also speeds up the target's ability to take evasive action, so watch out!

```
038E  1582          ;Go to Patch #1
 --
03B4  1588          ;Go to Patch #2
 --
0576  F8 00 BF F8   ;MLS TOGGLE subroutine switches
  7A  AC AF 0F FB   ;  CHIP-8 interpreter between
  7E  EC 5F D4 00   ;  high/low speed.
 --
0582  0576          ;CALL TOGGLE MLS - switch to high speed
  84  663F          ;Patched instruction from phasor sub
  86  1390          ;Return from Patch #1 - go fire phasors
  88  0576          ;CALL TOGGLE MLS - switch to low speed
  8A  00EE          ;Return from subroutine (instead of jumping
                       back
```

SIMPLE SIMON ADAPTATION

by Tim Longcor

There was a printing error in the original code of the Simple Simon game by Pete Kellner, Vol 2, Issue 2, at location 032C. It should be 5780 instead of 3780. After I got the program running, I decided I would add skill levels and a time limit for responding.

At the beginning of the game hit key 1 through 4 for whatever skill level you would like to try. The winning sequences are 8, 14, 20, and 26 respectively. If you take longer than 4 seconds to respond, you will lose. The speed of the game now increases after every fifth note rather than every note. The rest of the program operates the same as the original.

```
0200  1332      02B8  12C0      0350  2310      0376  6614
                                0352  6505      0378  4719
0208  120C      02D6  1382      0354  75FF      037A  6619
                                0356  2240      037C  8160
021A  228A      0332  F00A      0358  3500      037E  811E
                0334  6908      035A  1354      0380  127A
0230  234C      0336  4001      035C  F00A      0382  2310
0232  2310      0338  6908      035E  400F      0384  1200
0234  1204      033A  4002      0360  1200
                033C  690E      0362  400E
0274  1276      033E  4003      0364  2320
0276  1368      0340  6914      0366  135C
                0342  4004      0368  4705
028A  65FF      0344  691A      036A  6605
028C  F515      0346  22E0      036C  470A
                0348  6600      036E  660A
02AA  12B2      034A  1202      0370  470F
                034C  5790      0372  660F
02B2  F507      034E  00EE      0374  4714
02B4  3500
02B6  1298
```

FLIP-A-ROUND

This game shows the numbers 1 through 9 in a random sequence. The object is to arrange them in order from left to right in the least number of tries. An example of how the program works is as follows; Suppose "271485396" is the starting sequence. Pressing key 5 will give you "841725396." Note that only the first five numbers "flip-a-round." After a few tries, you will begin to catch on.

```
0200  60 00 61 00 62 00 A3 0F F2 55 A2 F2 F9 65 A2 FC
  10  F9 55 6A 00 CB 03 CC 06 8B C4 4B 00 12 14 A2 FC
  20  FB 1E F0 65 40 00 12 14 A3 06 FA 1E 7A 01 F0 55
```

```
0230  60 00 A2 FC FB 1E F0 55 3A 09 12 14 6A 00 22 BA
  40  FE 0A 4E 00 12 40 8C E0 6D 0A 8C D5 3F 00 12 40
  50  8B E0 6D 00 A3 05 FE 1E F0 65 A3 11 7D 01 FD 1E
  60  F0 55 7E FF 3E 00 12 54 A3 06 FB 1E F8 65 A3 12
  70  FB 1E F8 55 A3 12 F8 65 A3 06 F8 55 7A 01 A3 0F
  80  FA 33 30 01 12 EE 31 02 12 EE 32 03 12 EE 33 04
  90  12 EE 34 05 12 EE 35 06 12 EE 36 07 12 EE 37 08
  A0  12 EE 38 09 12 EE 22 BA 60 00 61 40 F1 18 70 01
  B0  30 07 12 AC F0 0A 00 E0 12 00 00 E0 64 00 61 09
  C0  62 0D A3 06 F0 65 F0 29 D1 25 71 05 74 01 A3 06
  D0  F4 1E 34 09 12 C4 61 28 62 1B A3 10 F0 65 F0 29
  E0  D1 25 71 05 A3 11 F0 65 F0 29 D1 25 00 EE 22 BA
  F0  12 40 00 01 02 03 04 05 06 07 08 09 xx xx xx xx
```

PRELIMINARY

VP-3301

INTERACTIVE DATA TERMINAL

L-1009

Microprocessor Controlled
Color Graphics
Low Cost
ASCII Encoded
Programmable & Resident Character Set

This professional quality terminal is suitable for a wide variety of industrial, educational, business and personal applications requiring interactive communication between computer and user. Microprocessor intelligence and LSI video control integrated circuits bring performance, features and flexibility at low cost.

The character display format, 40 characters by 24 lines or 20 characters by 12 lines, is software selectable. Each character or all characters may be displayed in one of eight colors (or gray scales on B/W display). Display background may be one of eight colors (or gray scales on B/W display). There are 125 resident displayable characters or you can define your own characters - Greek letters and other foreign alphabets, graphic symbols, large graphics building blocks, playing card suits, unique character fonts, and "little green men". Reverse video feature creates visual emphasis on single or multiple character, words or lines.

The terminal communications interface is industry standard asynchronous RS-232C or 20 mA current loop with six switch selectable baud rates. Switch selectable configuration control includes line/local, upper case only, full/half duplex, data word formatting, plus two control code options.

A built in tone generator, used for aural keypress feedback, can be programmed for end-of-line bell, error messages or even music.

The terminal utilizes modern flexible-membrane, key switches with a light positive activation pressure. Contact life is rated at greater than five million operations. A finger-positioning overlay combined with the positive keypress action gives good operator "feel". The unitized keyboard surface, impervious to liquids or dust particles, combined with high noise immunity CMOS circuitry make this unit particularly suitable for use in hostile environments.

The base band video output can be directly connected to a 525 line color or black and white video monitor or with RF modulation to a standard color or black and white TV set. A wall receptacle type 5 volt 650 mA power supply is included.

RCA **MicroComputer Products**

2.08/09.32

VP-3301 Interactive Data Terminal Specifications - [Preliminary]

KEYBOARD

Format: 58-key typewriter format - includes two user-definable keys (Switch closures).

Key Switches: Flexible membrane, polycarbonate material.

Rollover: Two key.

User Definable Keys: Two SPST switches (30 V, 0.1 A, 1 W max.).

Key Life: Greater than 5-million operations.

Audio: Onboard speaker gives aural keypress feedback. Also provides bell and error indication. Programmable frequency.

Code: ASCII (American Standard Code for Information Interchange).

Character Set:

Standard Upper/Lower Case - 128 codes generated:
- 95 printable characters
- 33 control characters

Upper Case Only (Switch Selectable) - 102 codes generated:
- 69 printable characters
- 33 control characters

Break Key: Causes continuous transmission of binary state zero.

DISPLAY

Screen Format: Software selectable, 24 lines of 40 characters or 12 lines of 20 characters, one page of data.

Character Font: 5 x 6 dot matrix in a 6 x 8 block, descenders for lower case characters.

Resident Character Set: 52-upper and lower case alphabetic characters, 10-numerals, 32-punctuation and math characters, 31-control characters (Switch selectable control character display for programming convenience).

User Definable Characters: Any bit pattern in a 6 x 8 character block may be set with a command control sequence. A total of 125 characters may be defined at one time.

Cursor: Blinking reverse video. May be set to mid-blinking or turned off with a command control sequence.

Reverse Video: Control character turns reverse video on and off permitting one or multi character, word, or line reverse video.

Character Color/Intensity: Choice of 8 colors or 8 levels of gray scale.

Background Color/Intensity: Choice of 8 colors or 8 levels of gray scale.

Serial Video: Composite color video line, NTSC compatible. 1.0 V p-p into 75 Ω termination.

Video Connector: Female, RCA phone plug.

USER CONTROL

Switch Selectable: Baud rate, current loop/RS-232C, line/local, upper case only, full/half duplex, parity bit-even, add, mark, or space, 1/2 stop bits, control execute ON/OFF, control display ON/OFF.

Control Characters: Back space (cursor ←), linefeed (cursor ↓), upline (cursor ↑), fore space (cursor →), carriage return, horizontal tab, bell, clear screen, home cursor, delete, reverse video on, reverse video off.

Command Control Sequences: Character color or gray scale, background color or gray scale, locate cursor, clear to end of line, clear to end of screen, define character bit pattern, keyboard ON/OFF, display format (40 x 24/20 x 12), cursor ON/OFF, display ON/OFF, define command control delimiter, tone generator.

INDICATORS

Clear to Send: LED on when "clear to send" true, acts as power on indicator when "clear to send" not used.

Bell: 250 m sec. audible tone.

Error: Beeping audible tone indicate input or output overrun.

DATA FORMAT - TRANSMIT

Method: Asynchronous; serial-by-bit; serial by character.

Code: ASCII; 1 start bit, 7 data bits, 1 parity bit (odd, even, mark, or space), 1 or 2 stop bits.

Mode: Switch selectable: half or full duplex, line/local.

EIA Data Rate: 110, 300, 1200, 4800, 9600, 19.2K baud; switch selectable.

CL Data Rate: 110, 300 baud; switch selectable.

Control Signals: Request to send (output signal), clear to send (input signal), EIA RS-232C compatible.

DATA FORMAT - RECEIVE

Method: Asynchronous; serial-by-bit; serial by character.

Code: ASCII; 1 start bit, 7 data bits, 1 parity bit-ignored, 1 or 2 stop bits.

Mode: Switch selectable.

EIA Data Rate: 110, 300, 1200, 4800, 9600, 19.2K baud; switch selectable (Clear to receive signal must be used for data rates > 300 baud).

CL Data Rate: 110, 300 baud, switch selectable.

Character Rate: 508 character-per-sec max.

Control Signals: Clear to receive (output signal), EIA RS-232C compatible.

COMMUNICATIONS INTERFACE

EIA: RS-232C compatible, ± 12 V nominal output signal voltage.

Current Loop: 20 mA to 60 mA max, voltage capability 30 V max.

Mode: EIA or current loop switch selectable.

Connector: 25 pin, female, subminiature "D"-type.

PHYSICAL

Size: 13.1" L x 7" D x 2" H.

Shipping Weight: 5 lbs. approx.

POWER

Input: 5 V DC @ 650 mA nom. from included 120 V, 60 Hz wall receptacle type power supply.

Switch: Power ON/OFF,

ENVIRONMENTAL

Temperature:
- Operating: 0 to 50° C
- Storage: -40 to +85° C

Humidity: 90% RH @ 30° C, non-condensing.

Vibration: 3-axes - .3" p-p, 5 to 14 Hz
3 g, 14 Hz to 2K Hz

Shock: 3-axes - 50 g, 11 m sec, 1/2 sine wave.

RCA MicroComputer Products|New Holland Avenue
Lancaster, PA 17604

Printed in U.S.A./3-80
VP-3301

2.08/09.33

VIP Keyboard by Doug Wolf

This program turns your VIP into a four octave keyboard. The program uses the Simple Sound Board VP595. It uses CHIP-8I which is CHIP-8 with the following changes: starting at 01A4: 86 FA 01 3A AC E5 63 D4 E7 45 FA 01 3A F2 63 D4, and starting at 01F2: 3F F2 6B 3F F5 D4. To cover four octaves two keys are used to either increment of decrement the octave presently used by the keyboard.

The keyboard:

```
1  2  3  C
4  5  6  D
7  8  9  E
A  0  B  F
```

is equivalent to:

```
D# X  F# G#
D  E  F  G
A  B  C  (higher octave)
A# X  C# (lower octave)
```

Variables

V0=frequency select	V4=key input 4	V8=key input 8	VC=key input C
V1=key input 1, no note	V5=key input 5	V9=key input 9	VD=key input D
V2=octave	V6=key input 6	VA=key input A	VE=key input E
V3=key input 3	V7=key input 7	VB=key input B	VF=key input F, (note duration)

When MI=00 no note is played (no keys are pushed).

Tables of notes start at 290. The four octaves are in sequence. Each octave is in this order: 00, D#, XX, F#, D, E, F, A, B, C, A#, C#, G#, G.

Initialize Values

```
0200   6101   V1=01
0202   6200   V2=00
0204   6303   V3=03
0206   6404   V4=04
0208   6505   V5=05
020A   6606   V6=06
020C   6707   V7=07
020E   6808   V8=08
0210   6909   V9=09
0212   6A0A   VA=0A
0214   6B0B   VB=0B
0216   6C0C   VC=0C
0218   6D0D   VD=0D
021A   6E0E   VE=0E
021C   6FFF   VF=FF
021E   6000   V0=00
```

Scan Keyboard for Notes

```
0220   A290   I=290
0222   E1A1   If(V1≠hex key) skip
0224   F11E   I=291
0226   E3A1   If(V3≠hex key) skip
```

```
0228   F31E   I=293
022A   E4A1   If(V4≠hex key) skip
022C   F41E   I=294
022E   E5A1   If(V5≠hex key) skip
0230   F51E   I=295
0232   E6A1   If(V6≠hex key) skip
0234   F61E   I=296
0236   E7A1   If(V7≠hex key) skip
0238   F71E   I=297
023A   E8A1   If(V9≠hex key) skip
023C   F81E   I=298
023E   E9A1   If(V9≠hex key) skip
0240   F91E   I=299
0242   EAA1   If(VA≠hex key) skip
0244   FA1E   I=29A
0246   EBA1   If(VB≠hex key) skip
0248   FB1E   I=29B
024A   ECA1   If(VC≠hex key) skip
024C   FC1E   I=29C
024E   EDA1   If(VD≠hex key) skip
0250   FD1E   I=29D

                    Scan Keyboard for Octave Change

0252   EEA1   If(VE≠hex key) skip
0254   1278   Go to 0278   Subroutine to increment octave
0256   EFA1   If(VF≠hex key) skip
0258   1284   Go to 0284
025A   F21E   I=I+V2   Select octave

                       Check for No Key Pressed

025C   F065   V0=MI   Get tone
025E   4000   If(V0≠00) skip
0260   1270   Go to 270   Subroutine for No Key Pressed

                            Sound Generated

0262   B100   Output port=V0
0264   FF18   Tone duration=VF
0266   1220   Go to 0220

                    Subroutine for No Key Pressed

0270   6100   V1=00
0272   F118   Tone duration=V1
0274   6101   V1=01   Reset V1
0276   1220   Go to 0220

                            Increment Octave

0278   422A   If(V2≠2A) skip
027A   1220   Go to 0220
027C   720E   V2=V2+0E
027E   EEA1   If(VE≠hex key) skip
0280   127E   Go to 027E   Wait for key release
0282   1220   Go to 0220
```

Decrement Octave

```
0284   4200   If(V2≠00) skip
0286   1220   Go to 0220
0288   72F2   V2=V2+F2
028A   EFA1   If(VF≠hex key) skip
028C   128A   Go to 028A   Wait for key release
028E   1220   Go to 0220
```

Note Table

0290	00	B0	00	94	BB	A6	9D	FD
0298	DE	D2	EB	C6	84	8B	00	58
02A0	88	49	5D	53	4E	7C	6F	68
02A8	75	62	41	45	00	2B	BE	24
02B0	2E	29	26	3E	37	34	3A	31
02B8	20	22	00	15	55	12	16	14
02C0	13	1E	1B	19	1D	18	10	11

The New York Amateur Computer Club has announced the creation of a regular "Computer Fair and Flea Market" to be held on the first Sunday of each month.

Computing systems exhibiting applications software, computer company vendors, representatives from local computer clubs, will be there. Meetings, lectures and tutorials on hardware, software, business systems and beginners classes will also be held.

PLACE: Irving Plaza
12 Irving Place (corner of East 15 St. NYC.)
TIME: 11am to 5pm. Lectures from 1pm.
ADMISSION: $1.50
DATES: March 2, April 13, May 4, June 1

For more information, contact Daniel at 212-263-3003 or phone the New York Amateur Computer Club at their "Hot Line" number 864-4595.

SET CARRY/CLEAR CARRY

by Tom Swan

The 1802 machine language instruction set contains no instructions to directly set or clear the one bit carry register DF. Normally, the procedure is to rotate a zero or a one bit from D into DF but of course this will change the value in D. A simpler and shorter method is given below.

```
To clear DF:  ADI  $00  ;Add 0 to anything sets DF = 0
To set DF  :  SMI  $00  ;Subtract 0 from anything sets DF = 1
```

Notice that only DF is changed by either of these commands -- the value in the D register is not disturbed.

VIP TINY BASIC MACHINE LANGUAGE SUBROUTINE

by Andrew A. Modla

Several VIP computer hobbyists have expressed a desire to call machine language programs from TINY BASIC. Although hooks to machine language programs were not designed into TINY BASIC due to space limitations, there is a way to execute machine language programs. The technique fakes a new statement type and when executed by BASIC, calls the machine language program.

Statements have the following internal format:

Binary line No.	Length of Statement	Statement type code	Source statement	ØD	End of Statement
2	1	1		1	

The machine language program resides in the source statement area.

To construct a machine language program, do the following steps:

1) Clear the statement area by typing NEW.

2) Enter the following two BASIC statements:
   1 GO TO 10
   2 REM

3) Holding down Key E enter the VIP operating system by turning on the RUN switch.
   You will be modifying memory as follows:

ADDRESS	DATA	
118E	1300	This is the new free memory address
1213	ED	Line 2 statement length
1214	C4	Machine language statement type
1215 to 1232	4E	Printable Character E (filler)
1233	0E	Causes inner interpreter to enter machine language
1234		Your machine language program starts here
12FF	0D	End of statement code

   The machine language program must end with D4 03 08 99 which cannot extend beyond location 12FE. This means you have 199 decimal bytes available for your machine language subroutine. 00 and 0D cannot be used in your ML program. To load 00 into D use 91 (GHI 1).

4) Holding down the backspace key on your ASCII keyboard, turn on the run switch. This procedure prevents the loss of your BASIC program.
   Add the following line:
   3 RETURN
   Your BASIC program begins with line 10.
   You call the machine language program with a GOSUB 2.

The following information is useful for writing machine language programs:

1) The ML subroutine program counter is register 5.
2) Use only registers R7, RA, RB, RC, RD, RE, and RF
3) R6.1 contains the page address of the variables and must not be altered. (1100)
4) The display starts at location 1000 (one page)
5) The variables A-Z start at location 11A7 (two bytes per variable)
6) The array A starts as the free address in 118E.
7) The stack occupies the last page of memory and the low order portion may be used for storage. (Page address R2.1)
8) The BASIC source program starts at 120A.

LITTLE LOOPS by Tom Swan

PRIME TIME

They just finished building a bullring right below our window -- it's painted bright red and yellow and orange, and it's the icing on the cake of our now brown Mexican countryside. Here they do not have bloody Spanish bullfights -- they are more like rodeos though there is far more drinking, taco eating, laughing and shouting in the stands than there is action in the ring. This has absolutely nothing to do with this month's column, unless you want to compare a Mexican's interest in rodeos with my interest in BASIC language programming. Every couple of weeks they have a rodeo, then everybody goes back to normal. Every once in a while, I write a program in BASIC (even on our new "baby" in the house, an APPLE II, I'm more interested in machine language programming.) Problem is, I seldom go back to normal.

I don't want to sound too "ho-hum" about BASIC. It offers a good way to learn programming and judging from the number of readers letters and articles this month, a lot of you own VIP Tiny BASIC Boards. CHIP-8 is also an excellent language for cutting your programming teeth. Though it lacks the readability of a documented BASIC program, it is more suitable for use on a VIP computer.

The following program is straight from an algorithm (that means a procedure or plan for a programming task) from Vol 1 of The Art of Computer Programming by Donald E. Knuth, page 143. It generates a list of prime numbers -- as many as you ask for up to the limits of your memory, variables size, and patience. One interesting line (#140) gives the remainder after a division providing the function X MOD Y which is not in Tiny BASIC's vocabulary. You may leave the REM's (remarks) out without affecting the program. Notice that the little b's with lines through them are spaces (b).

But what if you don't have the BASIC board? In that case you will need to program in either 1802 machine language or CHIP-8 probably. In fact, that brings me to the point I would like to make. There are tons of BASIC programs out "there" and only a few kilos of CHIP-8 programs (we're doing our best to change that). Because BASIC is fairly easy to read and understand, even if you don't have the board, you should try to pick apart a few BASIC programs.

One thing that may surprise you is how easily a BASIC listing translates into CHIP-8! At first this may seem to be awkward, especially if CHIP-8 is new to you. Here are a few suggestions to help.

BASIC - PRINT
CHIP-8- Printing numbers is not difficult but words are another thing. Those of you with PIPS FOR VIPS Vol I can use Messager to translate PRINT statements to CHIP-8. Otherwise just leave the PRINT lines out or simplify them graphically using DXYN instructions.

BASIC - Less and greater than
CHIP-8- Use 8XY5 instruction to subtract X - Y. Then test VF to see if the result was negative (X<Y) or positive (X≥Y).

BASIC - Subscripted variables (e.g. A(N))
CHIP-8- Define a memory area to hold the array of numbers A(0), A(1), A(2)...A(i). Set any CHIP-8 variable VX equal to N. Set I to Base of memory area for array. Add VX to I with FX1E. Get A(N) with F065 instruction which sets VØ = A(N)

-or-

Insert VØ = VX into A(N) with F055 instruction.

BASIC - Multiply and divide
CHIP-8- Write subroutines in CHIP-8 or machine language for these functions.

With these four suggestions and your VIP manual, you should be able to convert PRIME TIME to CHIP-8. One problem still remains, however. When you PRINT something in BASIC, everything else scrolls up to make room for the new line. That is not so easy to simulate in CHIP-8, so next month I suppose I have my work cut out for me! Good luck!

PROJECTS:

1) Convert PRIME TIME to CHIP-8.

2) Speed up the BASIC LISTING by using TVON, TVOFF instructions. After all primes have been calculated, print out a neatly formatted list from the array (A(N) instead of one at a time.

Contest: First person to write to me with an explanation of what "CHIP-8" means wins a free CHIP-8 game (undocumented but with instructions) never before published. In case of Postmark ties, the first correct answer I see wins. I figured it out -- can you? (Even Rick Simpson doesn't know -- at least that's what he told me!) No, Terry, you are not eligible.

PRIME TIME LISTING

```
5    REM REQUIRES VIP TINY BASIC
10   CLS
20   PRINT "HOW MANY PRIMES"
30   INPUT X
35   IF X<2 THEN 10
40   PRINT
45   PRINT "OK.  HERE GOES!"
49   REM FIRST PRIME IS '2'
50   A(1) = 2
55   PRINT "1 = 2"
58   REM J IS NUMBER PRIMES FOUND
59   REM N IS ODD CANDIDATES (LIKE KENNEDY AT THIS POINT)
60   N=3
70   J=1
80   J=J+1
90   A(J)=N
94   REM OUTPUT PRIME NUMBER
95   PRINT J;"⁄=⁄"; A(J)
99   REM TEST FOR DONE
100  IF J=X THEN 999
110  N=N+2
111  REM NOW DIVIDE CANDIDATES BY
112  REM PRIMES FOUND (A(K)'s) SO FAR
113  REM Q IS THE QUOTIENT
114  REM R IS THE REMAINDER
120  K=2
130  Q=N/A(K)
140  R=N-Q*A(K)
149  REM PRIME POSSIBLE? (R≠0)
150  IF R=0 THEN 110
159  REM PRIME? (Q<=PRIME DIVISOR)
160  IF Q<=A(K) THEN 80
169  REM N IS NOT PRIME
170  K=K+1
180  GOTO 130
999  END
```

LAST MONTH'S ANSWERS:

1) Wipe Off Modifications -- Auto Speed Up (plus auto restart on Key 0 at end of game.) Load CHIP-8 Wipe Off and normal CHIP-8 as listed in the VIPmanual then enter the following.

```
00AC          EC            -- Switch to Hi-speed CHIP-8
 --
021E          670A  ;V7=0A  -- Optional -- # balls = 10 for higher
 --                             challenge
0234          1300   INIT   -- Jump to initialize V8,V9 -- each ball
 --
0242          131C   TIME   -- Jump to speed control -- during play
 --
```

```
029C          1308    CHNGE -- Jump to increase speed -- each paddle
 --                             hit
02C8          1328    RSTRT -- Jump to restart on Key Ø -- end of game
 --
0300  INIT:   6804  ;V8=04 -- V8=speed--value=starting speed
  02          6904  ;V9=04 -- V9=increase--value=#paddle hits to increase
  04          FF0A  ;WAIT  -- Patched instruction from 0234
  06          1236   RET1  -- Return from INIT patch

0308  CHNGE:  4800  ;SK≠0  -- If speed ≠ 0, jump to 030C
  0A          1318   C2    -- If speed = 0, (highest speed) Jump to exit
  0C          3900  ;SK=0  -- If increase = 0, then jump to speed up
  0E          1316   C1    -- If increase ≠ 0, jump to decrement V9
                               (no speed up)
0310          6904  ;V9=04 -- Reset V9 = #paddle hits to each increase
  12          78FF  ;V8-1  -- Speed up ball by factor of 1
  14          1318   C2    -- Jump to exit
  16  C1:     79FF   V9-1  -- Decrement V9- no speed up this paddle hit
  18  C2:     A2CB   BALL  -- Patched instruction from 029C
  1A          129E   RET2  -- Return from CHNGE patch

031C  TIME:   F815  ;TI=V8 -- TIMER = value of V8 (auto decrement)
  1E  T1:     FF07  ;VF=TI -- VF= current timer value
  20          3F00  ;SK=0  -- When timer goes to 00, skip to exit
  22          131E  ;T1    -- Else loop to test timer again
  24          A2CD   PADDL -- Patched instruction from 0242
  26          1244   RET3  -- Return from time patch

0328  RSTRT:  FF0A  ;WAIT  -- Wait for any key to be pressed
  2A          3F00  ;SK=0  -- If Key Ø, skip to start new game
  2C          1328   RSTRT -- Else loop back for another keypress
  2E          00E0  ;ERASE -- Erase screen--prepare for new game
  30          1200   BEGIN -- Jump to begin new game
```

NOTE: These changes cause the Wipe Off ball to speed up once every four paddle hits until it flies at top speed. There are four speed changes overall. To vary these parameters, change the values of the following locations. Also please note that the change to location 021E is optional -- I prefer a shorter 10-ball game, but you may select any number of turns you wish.

LOCATIONS OF INTEREST

```
0301                    -- Starting speed of each ball (00=fastest)
0303; 0311              -- Number paddle hits to each speed change
                        -- (Value at 0303 must equal value at 0311)
0309                    -- Top speed of balls (00=fastest)
```

SUPER DENSE WIPE OFF VARIATION

In addition to the changes above, the following create a different action filled game. (NOTE: Addresses are not sequential. Be sure to enter the changes at the right addresses.)

```
0202          6A0A  ;VA=0A -- Change to 10 rows of dots
0206          6B0B  ;VB=0B -- Change to 11 columns
020C          7006  ;V0+6  -- Spacing between each column
0214          7102  ;V1+02 -- Spacing between each row
0284          46DC  ;SK≠DC -- Change score--DC=220 maximum
02CC          90E0         -- Data/02CC=Dots/02CD=Paddle
```

2) MLS TOGGL - completely relocatable

```
F800            TOGGL: LDI  $00    ;Set RF to $00AC--address
BF                     PHI  RF     ;  inside DXYN instruction
F8AC                   LDI  $AC    ;  in CHIP-8 interpreter
AF                     PLO  RF     ;
0F                     LDN  RF     ;Get byte @ $00AC
FBEC                   XRI  $EC    ;Flop from $EC to 00, or 00 to $EC
5F                     STR  RF     ;Put back to change speed
D4                     SEP  R4     ;Return
```

NEXT MONTH: SCROLL UP FOR THE MYSTERY TOUR

CORRECTIONS CORRECTIONS CORRECTIONS CORRECTIONS OH NUTS CORRECTIONS

VOL 1 ISSUE 4 OCT. 1978 PAGE 11 "Non Video Operating System" by Joseph Czajkowski contains an error in the schematic. The correct pin connections for the HP 5082-7340 displays are: pins # 4 and #6 to ground; pin #5 to enable. Pin #5 of the top display is incorrectly hooked to ground, in the printed schematic.

VOL 2 ISSUE 5 NOV. 1979 PAGES 2.05.16 and 2.05.19 "Football" by Frank Awtrey contains two typographical errors:

Ø6CA : ØØEE not ØDEE
ØD3Ø : F8Ø2 not FBØ2

Thanks to Tim Longcor for weeding these little buggers out.

Just a quick note that may be of interest to other Elf'ers who enjoy the VIP modification in issue #9 April 1979. In addition to the error previously brought to your attention Loc. 8104 should be 3E not 3F - I've discovered that some of the longer programs in the RCA game manual crash because they get into the CHIP-8 display page relocated at page 7. To correct this change the instruction at Loc. 0CF0 from F807 to F80B. All works well now. - Walt Pinner

APOLOGIES

We had hoped to never have to do a combined issue - but here is just exactly that. I had pneumonia for three weeks in February and early March - and no sooner did I recover from that when my daughter (who lives in California) had a real catastrophe in her life and I had to fly back to be with her for the next two weeks. So - the result was: no newsletter in March. Sure hope this big fat issue helps make up for it all. - Terry

MORE TINY BASIC MACHINE LANGUAGE SUBS

by C. D. Smith

Recently you received a procedure from Andy Modla for adding a machine language subroutine to the ROM based VIP Tiny BASIC, (Part number VP-700). I have enclosed a procedure I developed using Andy Modla's memo to allow calling several machine language subroutines. I intended it to be self explanatory however, if you have any questions about its use please contact me through the VIPER.

```
!M
0000 ;           0001 ..
0000 ;           0002 ..     VIP TINY BASIC MACHINE LANGUAGE SUBROUTINE
0000 ;           0003 ..
0000 ;           0004 ..     C.D. SMITH      80-01-08   INITIAL RELEASE
0000 ;           0005 ..
0000 ;           0006 ..     BEGIN BY INITIALIZING THE STATEMENT AREA
0000 ;           0007 ..     BY ENTERING
0000 ;           0008 ..
0000 ;           0009 ..     NEW
0000 ;           0010 ..     1 GO TO 10
0000 ;           0011 ..     2 REM A MACHINE LANGUAGE SUBROUTINE.
0000 ;           0012 ..
0000 ;           0013 ..     NOTE: THE ABOVE THREE STATEMENTS MUST BE
0000 ;           0014 ..           ENTERED EXACTLY AS SHOWN.
0000 ;           0015 ..
0000 ;           0016 ..     EXECUTE THE VIP OPERATING SYSTEM BY
0000 ;           0017 ..     RESET/KEY E/RUN
0000 ;           0018 ..
0000 ;           0019 ..     MODIFY MEMORY AS FOLLOWS
0000 ;           0020 ..
0000 ;           0021        ORG #1233
1233 0E;         0022 CODE:  ,#0E
1234 ;           0023 ..
1234 ;           0024 ..     THE USER PROGRAM CALL STARTS HERE
1234 ;           0025 ..
1234 ;           0026 ..     NOTE: USER PROGRAM CALL MUST NOT CONTAIN
1234 ;           0027 ..           EITHER A #00 OR A #0D.
1234 ;           0028 ..
1234 ;           0029 ..................................................
1234 F812BF;     0030 VIPUSR: LDI A.1(TPC);PHI F
1237 F83BAF;     0031         LDI A.0(TPC);PLO F
123A DF;         0032         SEP F
123B ;           0033 ..
123B ;           0034 VARZ=#11D9
123B ;           0035 ..
123B F811BE;     0036 TPC:    LDI A.1(VARZ);PHI E
123E F8D9AE;     0037         LDI A.0(VARZ);PLO E
1241 4EB54EA5;   0038         LDA E;PHI 5;LDA E;PLO 5
1245 D5;         0039         SEP 5
1246 ;           0040 ..................................................
1246 ;           0041 ..
1246 ;           0042 ..     USER PROGRAM CALL ENDS HERE
```

```
1246 ;          0043 ..
1246 ;          0044 ..     NOTE: YOUR MACHINE LANGUAGE PROGRAM
1246 ;          0045 ..           MUST END WITH A LONG BRANCH TO ENDING
1246 ;          0046 ..
1246 ;          0047          ORG *
1246 D40308;    0048 ENDING:  SEP 4,#0308
1249 99;        0049          GHI 9
124A 0D;        0050          LDN D
124B ;          0051 ..
124B ;          0052 ..       BASIC CONTINUES FROM HERE
124B ;          0053 ..
124B ;          0054 BASIC:   ORG *
124B ;          0055 ..
124B ;          0056 ..       MODIFY NEXT BASIC STATEMENT
124B ;          0057 ..       LOCATION AS FOLLOWS
124B ;          0058 ..
124B ;          0059          ORG #118E
118E 124B;      0060 NEXT:    ,A(BASIC)
1190 ;          0061 ..
1190 ;          0062 ..       MODIFY STATEMENT 2 LENGTH AND TYPE
1190 ;          0063 ..       AS FOLLOWS
1190 ;          0064 ..
1190 ;          0065          ORG #1213
1213 38;        0066 LENGTH:  ,A.0(BASIC-LENGTH)
1214 C4;        0067 TYPE:    ,#C4
1215 ;          0068 ..
1215 ;          0069 ..       RESTART BASIC BY RESET/BACKSPACE/RUN
1215 ;          0070 ..
1215 ;          0071 ..       ENTER A RETURN STATEMENT AS FOLLOWS
1215 ;          0072 ..
1215 ;          0073 ..       3 RETURN
1215 ;          0074 ..
1215 ;          0075 ..       FOLLOWED BY A BASIC PROGRAM
1215 ;          0076 ..
1215 ;          0077 ..........................................
1215 ;          0078 ..
1215 ;          0079 ..       EXAMPLE
1215 ;          0080 ..
1215 ;          0081 ..       5 REM LET Z=ADDRESS OF SUBROUTINE(BASE 10)
1215 ;          0082 ..       10 Z=5120
1215 ;          0083 ..       15 GOSUB 2
1215 ;          0084 ..       20 PRINT Z
1215 ;          0085 ..       25 GOTO 10
1215 ;          0086 ..       30 END
1215 ;          0087 ..
1215 ;          0088 ..       THIS IS THE TEST EF4 ROUTINE
1215 ;          0089 ..       FOR THE HOME ENVIRONMENT MONITOR.
1215 ;          0090 ..       THE STATE OF EF4 IS TESTED AND VARIABLE
1215 ;          0091 ..       Z IS SET TRUE IF EF4 IS TRUE AND SET
1215 ;          0092 ..       FALSE OF EF4 IS FALSE.
1215 ;          0093 ..
1215 ;          0094          ORG 5120
1400 F811BF;    0095 EFTEST:  LDI A.1(VARZ);PHI F
1403 F8D9AF;    0096          LDI A.0(VARZ);PLO F
1406 915F1F;    0097          GHI 1;STR F;INC F
```

```
1409 3F0E;      0098          BN4 EFEXIT
140B F801;      0099          LDI #01
140D 5F;        0100          STR F
140E C01246;    0101 EFEXIT:  LBR ENDING
1411 ;          0102          END
0000
```

Extended Display Subroutine
by C. H. Anderson

CHIP-8 calling sequence:

AMMM: Point "I" to data set describing object to be displayed
0300: Call extended display machine language routine
FxFy: Vx,Vy coordinates for the object

Data format: starting at MMM
NB: Number of bytes on each line
NL: Number of lines
Data: NB*NL bytes describing object

Register Allocation:

R6.0: CHIP-8 variable pointer; shift loop counter

R7 : Used to hold bits in shift operation

RA : Memory pointer for data block, CHIP-8 "I" pointer

RC : Display memory pointer

RD.0: Loop counter for bytes displayed on each line

RD.1: Storage for NB, number of bytes on each line

RE.0: Counter for lines, initialized with NL

RE.1: Storage for number of shifts (Vx "AND" 07)

RF.0: Storage for the starting value of RC.0 on each line

RF.1: Hit Flag; initialized to 00, changed to 01 on hit

The stack is used to store the carry over bits of each shift operation.

```
0300: 22                 : Decrement Stack
    (Initialize RC= 8*VY+VX/8 + 0C00

0301: 45 A6              : R6 points to Vx
  03: 06 FA 07 BE        : Vx@07=XL;put into RE.1
  07: 06 FA 3F           : Vx@3F;ensure object is on display
  0A: F6 F6 F6 52        : divide by 8 and store on stack

030E: 45 A6              : R6 points to Vy
  10: 06 FE FE E2 AC     : 4*Vy put into RC.0
  15: F8 00 BF 7E BC     : Clear Hit Flag;00+carry put in RC.1
  1A: 8C FE AC           : 2*(4*Vy) put into RC.0
  1D: 9C 7E BC           : Put high bits of 8*Vy into RC.1

0320: 8C F4 AC AF        : Add Vx/8 into RC.0,put it into RF.0 too
  24: 9C 7C 0C BC        : Add 0C into RC.1

0328: 4A BD              : Store NB in RD.1
  2A: 4A AE              : Store NL in RE.0
```

2.08/09.45

```
NEW LINE 032C: 9D AD          : Put NB into RD.0 (counter for bytes)
           2E: F8 00 52       : Clear stack

CONT LINE0331: F8 00 A7       : Clear R7.0
           34: 9E A6          : Put XL into R6.0 (shift counter)
           36: 4A B7          : Data byte put into R7.1
         0338: 30 41          : Goto SHIFTENT
LOOP       3A: 97 F6 B7       : Shift contents of R7.1 to right
           3D: 87 76 A7       ; Shift carry over bits into R7.0
           40: 26             : R6-1
SHIFTENT   41: 86 3A 3A       : If R6.0 is not zero goto LOOP
-------------------------------------------------------------------
         0344: 97 F4 B7       : Add carry over from last byte to R7.1
           47: EC             : Set X to C
           48: F2 32 4E       : Test for hit, if not goto WRITE
           4B: F8 01 BF       : Hit Flag =01
WRITE      4E: 97 F3 5C       : Write byte to display with "XOR"
           51: E2 87 52       : Store new carry over byte on stack
-------------------------------------------------------------------
(Increment RC to next byte on line, test for and correct X wrap around)

         0354: 8C FC 01 AC    : RC.0+01,do it this way so RC.1 doesn't/
           58: FA 07 3A 60    : @07,if not 0 goto CONT          change
           5C: 8C FF 08 AC    : Move RC.0 up one line on display
CONT       60: 2D 8D 3A 31    : RD.0-01,if not 0 goto CONT LINE
-------------------------------------------------------------------
(Write last carry over bits on line)
         0364: 87             : Get carry over byte
           65: EC F2 32 6C    : Test for hit, if not goto WRITE2
           69: F8 01 BF       : Set Hit Flag to 01
WRITE2     6C: 87 F3 5C       : Write to display with "XOR"
           6F: E2             : Reset X to 2
-------------------------------------------------------------------
(Move RC to next line, test for and correct Y wrap around)
         0370: 2E 8E 32 87    :NL-1, if 0 goto EXIT
           74: 8F FC 08 AC AF : Get first value of RC.0 on line, move
                                to next line,store new RC.0 in RF.0
           79: 9C 7C 00 BC    : RC.1+carry
           7D: FF 10 3A 2C    : -(0C+04), if not 0 goto NEW LINE
           81: 9C FF 04 BC    : Reset RC to top of display
           85: 30 2C          : Goto NEW LINE
-------------------------------------------------------------------
EXIT     0387: F8 FF A6       : R6 points to VF
           8A: 9F 56          : Store Flag in VF
           8C: 12             : Increment stack
           8D: 00 D4          : Idle;Return
-------------------------------------------------------------------
Upon returning from the subroutine I=I+(2+NB*NL)+1.With DXYN I=I.
-------------------------------------------------------------------
```

There are no limitations on the values for NB or NL. However, if NB is larger than 08 or if NL is longer than the display the object will write over itself.

The routine will work with Ben Hutchinson's 128H by 64V circuit (VIPER VII,pg 7)with the following changes, which are facilitated with a NOP (E2) I have at 0313.

0309 7F; 0313 FE; 0359 0F; 035E 10; 0376 10;

The four pages of memory assigned to the display are assumed to start at 0C00. To change to a new page address (PGA), replace

0C with PGA at 0326. Inaddition,for the Y wrap around to work it is necessary to change the value at 037E to PGA+04.

Extended Display Subroutine

Options-Comments

Calling sequence and data format options:

(1) Fx and Fy can reside in the data set,preceeding NBNL, by changing 45 to 4A at 0301 and 030E.

(2) NB and NL can follow FxFy in the program by changing 4A to 45 at 0328 and 032A.

(3) There is room to set Fx to a fixed value in the subroutine. 0301: F8 Fx A6 06 FA 07 BE 46 FA 3F F6 F6 F6 52 E2(nop). The variable VY must then be the next CHIP-8 variable after VX (V0 and V1 for example).

This routine will work with the original CHIP-8 1 page display or the 2 page display if the value at 037E is changed to PGA+01 or PGA+02 respectively, where PGA=the top page of display memory.

There are times when it is desireable to superimpose the data block on top of what is already present without erasing anything. To do this change F3,XOR, to F1,OR, at locations 034F and 036D. It is interesting to experiment with the other logical and arithmetic operations at these locations.

A more useful option is to simply write the object over what is already on the screen. This is illustrated in the accompanying examples. To implement this option put a NOP instruction such as 5C or EC at locations 034F and 036D. NB is limited to 07 with this option.

The IDLE command 00 located at 038D stops all action until an interupt is called and hence can slow down a program. It can be replaced with a return D4.

When the 4 page interrupt such as Tom Swan described in VIPER(2.06.04) is used the program will continue while the display action is occuring. It is possible that the program will change the display memory before it gets displayed, usually toward the bottom of the screen. This may cause funny things to happen such as odd blanks appearing, especially with the IDLE removed. Sometimes this can be cured by synchronizing the program to the interrupt with judiciously placed IDLE calls. This is done with a machine language routine consisting solely of 00 D4 that is called by the CHIP-8 program.

If the number of bytes in the data block is more than about 16 then motion becomes jumpy because each byte must be shifted over 7 times when the lower 3 bits of VX are 111 and not at all when VX ends in 000.

Extended Display Subroutine (Examples)

These sample programs use Tom Swan's HI-RES CHIP-8 routines (VIPER 2.06.04) and will run with 2K of memory. They use the option that V0 and V1 are the coordinates , so be sure to make that change before entering the programs. The first writes and erases the object with the standard XOR option. The second demonstrates the smoother flicker free motion obtained using the overwrite option. The object is one of the many interesting ones I have found that can be made with the bar shaped dot.

```
0244: 6010 6110 A280 0300 6E00 7E02 EEA1 125A 4E0A 124C 124E
025A: A280 0300 4E02 71FF 4E04 70FF 4E06 7001 4E08 7101 1248
```

Data describing object (2 bytes wide-14(Hex) lines high)

```
0280: 02 14 00 00 03 00 0F 00
  88: 1E 00 3C 00 7C 00 7F F0
  90: 7F F0 7F 00 7F 00 7F F8
  98: 7F F8 7F 00 7F 00 7F F0
  A0: 7F F0 7F 00 1F 00 0F E0
  A8: 00 00
```

Move the object around using keys 2-4-6-8.

To change this program to the overwrite mode first put EC at 034F and 036D, then change the jump at 0252 from 125A to 125E so it skips the erase commands that are no longer required. Note that the object has "0" bits surrounding it on all sides and it is only allowed to move by increments of 1 so that no "1" bits of the previous frame escape being erased.

Chicken Crossing the Road Routine:

Leave the extended display subroutine in the over write mode and try the following animated sequence that uses the feature that the I pointer is not reset after calling the subroutine.

```
0244: 6303 6000 6169 A280 0300 6160 60FF 6205 A28A 0300
0258: F315 F407 3400 125A 7001 72FF 3200 1256 1252
```

Data:

```
0280: 07 01 FF FF FF FF FF FF FF FF ..road
028A: 01 09 06 7D 3E 1C 08 18 24 42 63  ...Frame #1
0295: 01 09 06 7D 3E 1C 08 08 14 24 36  ..#2
02A0: 01 09 06 7D 3E 1C 08 08 08 08 0C  ..#3
02AB: 01 09 06 7D 3E 1C 08 08 18 18 1C  ..#4
02B6: 01 09 06 7D 3E 1C 08 08 14 24 36  ..#5 same as #2
```

Variable V3 set at 0244 determines the speed of the action.

SCRT JUMP TABLE

by Leo F. Hood

In the PIPS FOR VIPS, I noticed that use is made of the Standard Call and Return subroutines. Below is a short program that when used with the SCRT allows subroutines to be accessed with a table look-up. This jump routine is useful to me in implementing additions to my disassembler program. The routine fakes the return address in Register 6 and jumps to the desired routine through the normal subroutine return program. The original calling location is of course saved on the stack prior to the jump.

The pertinent code for calling the JUMP program is included, but not the actual table since that is dependent on the application.

JUMP SUBROUTINE

```
27C0  96          JUMP: GHI  R6
  C1  73                STXD
  C2  86                GLO  R6
  C3  73                STXD
  C4  48                LDA  R8
  C5  52                STR  R2
  C6  90                GHI  R0
  C7  F4                ADD
  C8  B6                PHI  R6
  C9  48                LDA  R8
  CA  A6                PLO  R6
  CB  D5                SEP  R5
```

EXAMPLE

```
2722  90                GHI  R0     ;R0 contains address
  23  FC 07             ADI  #07    ;  of relocatable code
  25  B8                PHI  R8     ;R8 used for index
  26  F8 50             LDI  #50    ;  to table
  28  A8                PLO  R8
  29  8B          NEXT: GLO  RB     ;RB used for temporary storage
  2A  52                STR  R2     ;  of command code
  2B  08                LDN  R8
  2C  FB 7F             XRI  #7F    ;7F is end of table
***2E  32 D0            BZ   #D0    ;D0 is location of Error routine
  31  F5                SD          ;Compare command code to
  32  32 3E             BZ   #3E    ;  table entry branch if equal
  34  88                GLO  R8     ;If command not found
  35  FC 02             ADI  #02    ;  jump to
  37  A8                PLO  R8     ;  next entry
  38  98                GHI  R8     ;  "          "
  39  7C 00             ADCI #00    ;  "          "
  3B  B8                PHI  R8     ;  "          "
  3C  30 29             BR   #29
  3E  D4 07 C0    JMP:  SEP4 #07C0  ;Call JUMP
2741  30 03             BR   #03    ;When routine has finished, return
                                     will be to this location which
                                     branches to beginning of the
                                     routine which is not shown.
***2730  48             LDA  R8     ;(Sorry!)
```

2.08/09.49

NON-COMMERCIAL ADVERTISING

FOR SALE: Keyboard, Radio Shack 277-117 with I/O board and connectors for VIP. F. H. Bremer, 175 W Albanus Street, Philadelphia, PA 19120 (215) 455-6576

FOR SALE: VP-550 Super Sound Board with manual and cassette. Works fine, but it isn't what I thought it was. $40.00 Jerry Krizek (213) 338-2696 after 5:00 PM California time

FOR SALE: VIP, VP-590 (Color) and VP-550 (Supersound). All the manuals, Volumes 1, 2, and 3 of PIPS FOR VIPS, and approximately 25 game tapes. $400. I got an APPLE! Bob Brock, (301) 730-0922 before 9:00 PM Eastern Time.

* NOTE: We will print small "want ads" such as these for free, but they must be strictly non-commercial. They are for your information only, and we can't vouch for their accuracy or legitimacy. The advertisers listed here are VIPER subscribers.

* A T T E N T I O N *

ARESCO is attempting to get out of the hardware retail business, so we're selling our stock of VIP products. Here's an inventory of the material we have on hand, along with the prices we're asking for each piece. They'll be sold on a first-come, first served basis, so you might want to call in your order with a Master Charge or VISA number.

		NOW!	RCA
2	VP-711 COSMAC VIP Microcomputers.....	$179	199
2	VP-111 Microcomputers. Same as VIP, but user must install I/O ports, system expansion capability, and 5v power supply.	80	99
3	VP-595 Simple Sound Board	27	30
2	VP-550 Super Sound Boards	44	49
1	VP-570 Memory Expansion Board	76	95
2	VP-585 Keyboard Interface Card for hex keyboard	13	15
1	VP-575 Expansion Board	49	59
3	VP-576 Two-Board Expander	17	20
2	VP-565 EPROM Programmer Board	80	99
5	VP-620 Cable: ASCII Keyboards to VP-711	17	20
5	VP-700 Tiny BASIC ROM Board	35	39

In addition, we have about 50 sets of Volume 1 of the VIPER, which we can sell for our absolute printing & postage cost of $12.00, and a few VIP manuals.....We have a few loose back issues from Volume 1 VIPER, so if you're missing an issue, give us a call. Our cost per issue is $1.20.

ARESCO will not be publishing a volume 3 VIPER, due to circumstances we cannot control. If there is anyone out there who would like to take over the effort, let us know about you, and we'll let you know who the past subscribers are. Our thanks to all of you for your support! See you in May! - Terry

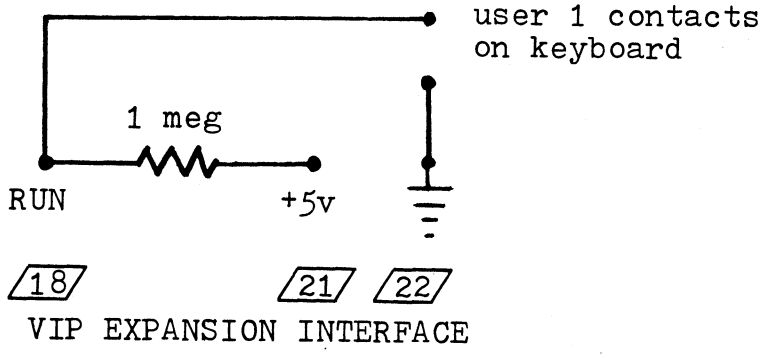
2.08/09.50

KEYBOARD RESET

by Steve Medwin

I've made a simple modification to the RCA ASCII keyboard that turns the "User #1" key into a reset key. This allows you to reset the VIP without flipping the run/reset switch.

All you need is one resistor, an edge-card board for the expansion interface and some wire. A schematic follows.



To use: The reset/run switch must be left in run. Then to reset the VIP, just press and release the User #1 key.

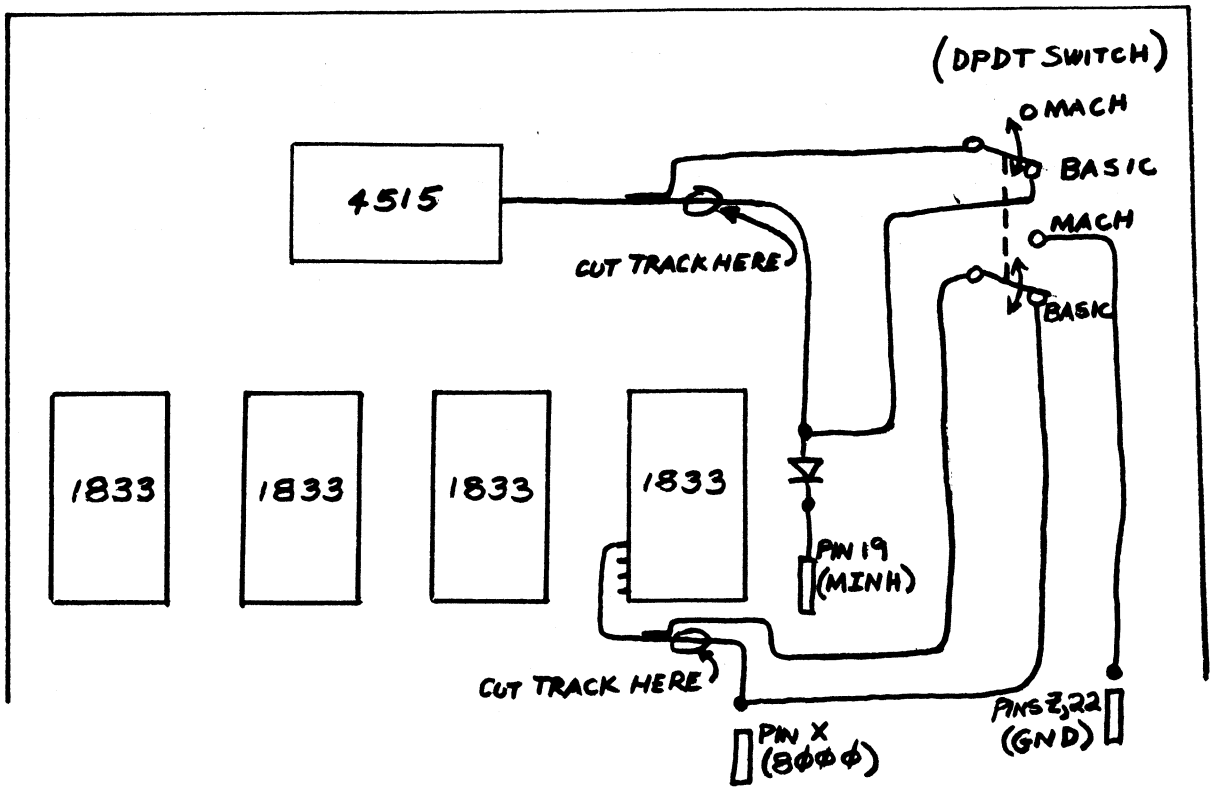
TINY BASIC DISCONNECT SWITCH

by Randy Holt

When the Tiny BASIC board is inserted into the VIP Expansion Interface connector, it doesn't permit the VIP to execute CHIP-8 or machine language programs. I disliked always taking the board out to run other programs, so I made the modifications described below. The double-throw, double-pole switch will operate in BASIC in one position and will permit CHIP-8 and machine language routines in the other position. Change the position of the switch only when the VIP is in the RESET mode.

On component side of board, cut track between CD4515 pin 10 and the diode, and between connector pin X (8000) and CDP1833 pin 21 as shown on the diagram. Solder wires as shown to the DPDT switch using plated-through holes wherever possible. I affixed the switch to the upper-right corner of the Tiny BASIC board.

TINY BASIC DISCONNECT SWITCH
(continued)



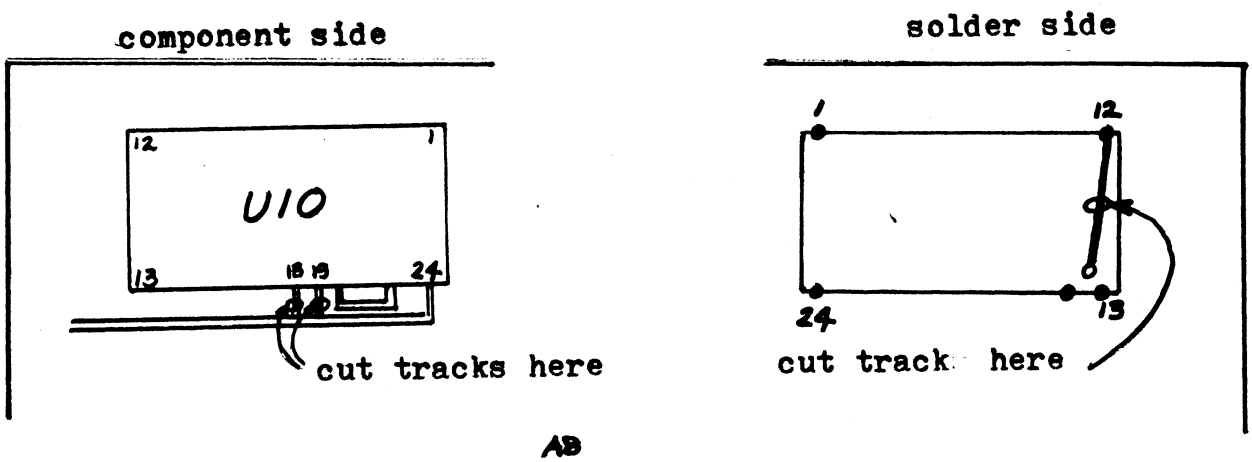
EXPANDED ROM MONITOR

by Randy Holt

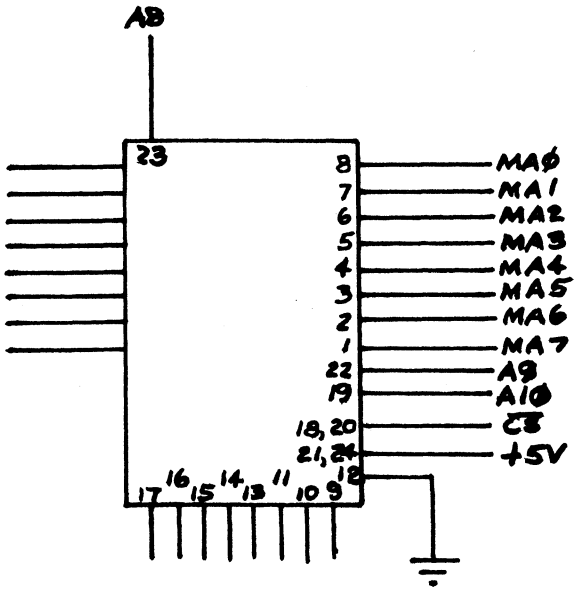
Wouldn't it be nice if you could replace the 512-byte VIP ROM Monitor chip with a 2048-byte EPROM that would then permit extra ROM space for a more sophisticated monitor and/or be used to store some of your favorite subroutines? It turns out that with just 3 cut tracks and the addition of 4 wires to the VIP board, you can have this capability!

The VIP ROM Monitor chip will still operate in the U10 socket; and when a 2716 EPROM is installed, it will exist in the stand-by (low power) mode unless it's addressed. The VIP EPROM Programmer card can be used to program the 2716.

EXPANDED ROM MONITOR



Revised Schematic:



On the component side, cut tracks from +5VDC to U10-18 and U10-19.
On the solder side, cut the track from U10-12(GND) to U10-21.
On the solder side, add wires as follows:

+5VDC to U10-21
U10-20 to U10-18
U10-22 to U16-6 (A9)
U10-19 to U8-11 (A10)

I have a few ideas that I hope to try soon. I'll put together an article if things "pan" out. How about some of you other VIP-ists? How can we best utilize this extra space?