VIPER

VOLUME 2, ISSUE 7 AN ARESCO PUBLICATION FEBRUARY 1980, $2.50

TABLE OF CONTENTS

Please note that the January 15 pre-publication price for the third volume of PIPS for VIPS was well received! We were able to print 200 copies altogether, so there are plenty more (at $19.95 with the program tape). Thanks to all of you who pre-paid and waited so long for your order.

ATTENTION: We have a new editor! Be sure to read this month's Editorial (and, of course, the publisher's notes) for a real pleasure!

SUBSCRIPTION RATES, ADVERTISING RATES, & OTHER INFORMATION

The VIPER is published ten times each year by ARESCO, at 6303 Golden Hook, Columbia MD, 21044, and mailed to subscribers on or around the 15th of each month except June and December. The single copy price is $2 per copy and the subscription price is $15/10 issues of the current volume. Subscriptions do not carry over from one volume to the next, and readers who want less than the full volume should send $2 for each issue desired. Renewals are accepted during the last two months of the current volume year, and the first issue of each volume is published in July.

Second class postage paid in Columbia MD 21045 (USPS 520-550).
Postmaster: Send all address changes to ARESCO, P O Box 1142, Columbia MD 21044.

Subscriptions: Subscription orders should be sent to P O Box 1142, Columbia, MD, 21044; not to the street address given above. USA residents: $20/10 issues mailed second class; $25/10 issues mailed first class. Non-USA residents: $20/10 issues mailed second class; $30/10 issues mailed first class. VISA/ MC/ personal checks, cash, money orders accepted as payment. All funds should be in US dollars. Checks drawn on foreign banks should include any exchange rate difference in currencies.

Advertising rates: Non-commercial ads by subscribers: $10/3 lines of 60 characters each. Non-commercial ads by non-subscribers: $15/3 lines of 60 characters each. For commercial ads, please write for advertising rates. Payment must accompany ad in any event.

Staff: Publisher: Terry L Laudereau, ARESCO, Inc.
Editor: Tom Swan
Subscriptions: Sandy Nolan

Contributions: Readers are encouraged to submit material of interest to VIP owners. However, it must be understood that the function of the VIPER is to duplicate the materials--and to mail them out to other VIPER readers. We cannot pay (sigh) for your efforts. In time, perhaps.....

EDITORIAL

by Tom Swan

One year ago almost to the day, I dug out one dusty, neglected COSMAC VIP and decided to write a group of articles and programs with the hope that some readers would pass more easily over many of the hurdles I had had to claw my way around in my never ending attempt to learn more about programming. The project grew into the PIPS FOR VIPS series, and the overwhelming response to PIPS' publication has shown me just what an insatiably curious, eager to learn, determined bunch you VIP people are! Wonderful. I feel as though we are all aboard the Santa (VIPER) Maria headed into uncharted waters set upon discovering new worlds. And I can't tell you how pleased I was when Rick and Terry, faced with the management of a growing company, asked me to become one of the officers of the ship. Would I edit the VIPER from now on? You bet!

As your editor, however, I'm more the cook in the galley than the captain on the bridge. The VIPER is <u>your</u> newsletter; a vehicle for relaying <u>your</u> experiences to other VIP programmers. Each issue of the VIPER has been prepared (and will continue to be prepared) from the stores in the hold. You tell us what you want to eat, and we'll do what we can to whip it up from the raw material you bring on board!

One thing I hope to accomplish is a thorough check of the programs printed here. Now I can't purchase a new video board just to check out a driver routine, but I will do what I can not to <u>introduce</u> errors into an author's code. Though the initials of my name suggest a relation to a famous microcomputer (my middle initial is "R"), I am only human, darn it, and errors are bound to occur. I will walk the psychological gangplank for each one.

Carmelo's "Killer Robots" in this issue is a good example. I hand loaded the program directly from the typed code you see printed here and it ran perfectly. If it doesn't work for you, then something was either dropped at the printer (unlikely) or you have made a loading error somewhere. In the past, Terry and Rick understandably didn't have the time to hand load each program. We do, and in most cases, we <u>will</u>. "We" includes my wife Anne who does all my dirty work such as typing this editorial, etc. (Nice game, by the way. Thank <u>you</u> Carmelo!)

As Terry has always said, your comments and suggestions are welcome. The VIPER is building an accumulation of original VIP programming knowledge, and we want to add your contributions. To those of you who have written, we look forward to hearing from you again. To you who intend to write, go ahead, pick up a pencil and send us your ideas. Don't be left on shore. We sail with the tide. Anchors Aweigh!

Best regards and good luck with your programming

TOM

Tom Swan, editor

2.07.02

PUBLISHER'S NOTE

We are delighted to have Tom with us! At last we have an editor who is truly qualified to answer technical questions about the VIP! Welcome aboard, Tom!

About the **light pen**...We've had enough positive response from readers that we bought up a carload of the pens, printed up a bunch of manuals, and can offer them to you at $19.95 per package. No cassette tapes. The manual gives listings of the demonstration programs (which are in BASIC), so they can be "translated" into CHIP-8 or machine language. So now we can take orders, folks....anybody interested?

RCA is doing some advertising with us these days...let's encourage them to do so by ordering products from them instead of through ARESCO. We don't make a "profit" through sales of RCA products, and they can ship directly to you instead of through us (which should help the turn-around time a lot).

Rumored: VIP II is being retooled to include a conventional keyboard; completion date tentatively late February. Seems too good to be true, in light of past rumors that the VIP II was being dropped altogether! We'll keep you posted as the plot thickens. **PIPS IV**, now only a twinkle in its author's eye, is beginning to take form rapidly. No orders yet, please, since we haven't even been able to get Tom to hint about the contents.

Jeff Duntemann has published an 1802-oriented booklet: **Captain Cosmo's Whizbang** ($5.00). It's the most entertaining reading we've come across in a long time - and jampacked full of good, hard information for 1802 aficionados. A must for your 1802 library, and available from Jeff (he prefers cash rather than checks, and orders must be prepaid) by writing to him at 301 Susquehanna, Rochester, NY, 14618.

The **Studio II RAM Card** is ready at last - $45 without the two 2114 RAM chips, completely assembled, otherwise. With all the necessary paperwork, of course. Our thanks to Paul Everitt and David Hall for their contribution (they designed it) to the effort.

As many of you have guessed (and phoned or written to tell us) there were a few errors in the FOOTBALL program by Frank Awtrey (see issue 2.05.12). John C Hanner was the first to call and let us know something was wrong with our listing (damn! I worked overtime on that one, and had two other people check it!). In any event, here are the corrections, with our thanks to all of you who have helped us spot the errors:

```
Change location 05D7 from 20 to 30
                06CA from 0D to 00
                0D30 from FB to F8
```

I guess three errors in all that code isn't all that bad, but it is such a nuisance to anyone who tries to punch in the code! Again....I apologize. See you next month!

Terry

Dear VIPER - Don King (one of your subscribers) and I have developed several programs for VIP Tiny BASIC including a crude version of STAR TREK. We are also working on doubling the tape data rate, and sending the tape data over the telephone lines, acoustically coupled to eliminate problems with direct connection to the telephone lines.

Thank you for providing a newsletter for and about the most powerful (as we all know) eight bit processor around today, the 1802. Please keep some non-VIP articles, but as your title indicates the majority of each issue should pertain to the VIP. - Marvin Kraska

Editor's Note - Marvin also mentioned he will be sending details on a "high speed A/D, D/A (analog/digital) converter with a sample rate of 10,000 samples per second capable of digitizing, storing, and reconstructing voice!!" The mouth waterith over! - Tom

Dear VIPER - I have enjoyed the Little Loops columns, and I find I agree with Tom in his review of the VIP700 BASIC in that it's tiny & slow. When I first connected a keyboard (an AMkey) to it, the automatic repeat on the keyboard (8/sec, after a small delay) was faster than the BASIC response, and led to multiple repeats (unless I was fast with the fingers). Luckily, I was able to get one of the RCA keyboards, and my previous problem was not repeated.

I had intended to use the keyboard/BASIC combination to learn new vocabulary words, but I was disappointed to find that this BASIC only inputs numeric data, and not alphabetic info. Aha, I thought, I'll add a machine language subroutine, but this BASIC turned out to be closed to that way also. I think there might be some hope if I try working with the CHIP-8 II and some additional routines for display, etc.

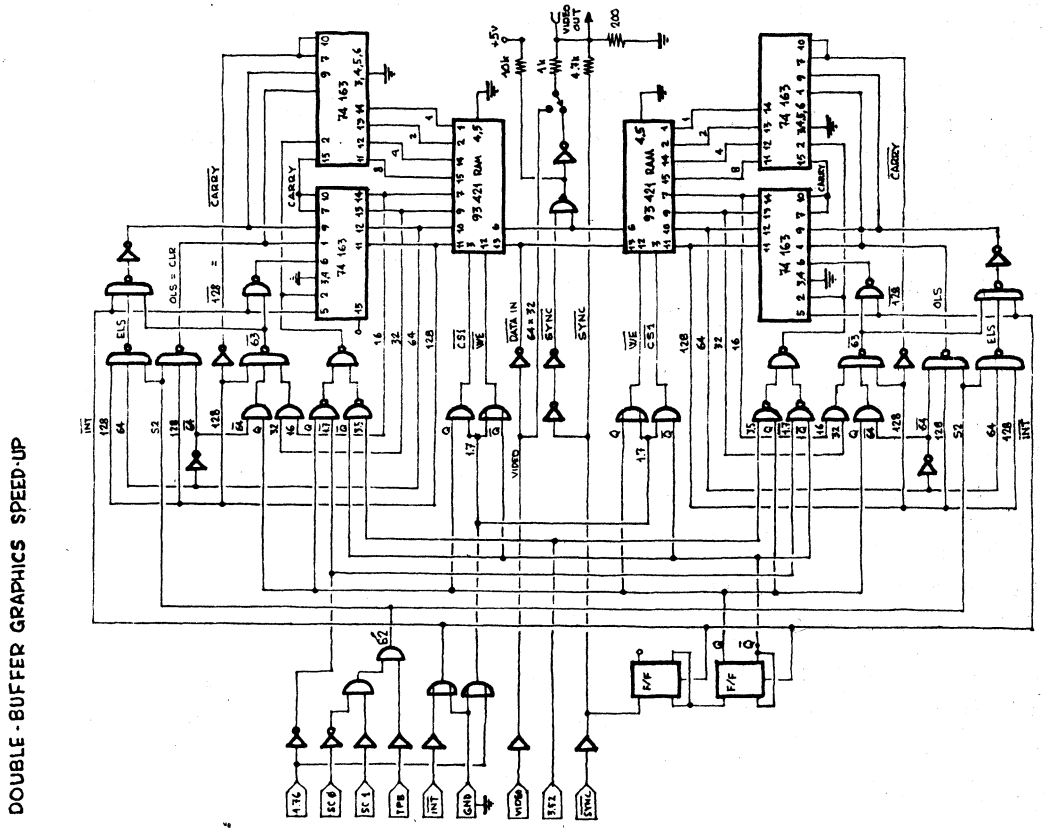
At least this BASIC does provide for some nice graphics-oriented commands, which leads me to start thinking about the light pen recently mentioned in da VIPER. - Eric Nabel

Dear VIPER - In the August '79 VIPER Al McCann requests a one-page version of Ben Hutchinson's circuit diagram and also asks if the RAMs are "wire-ORed' together. Feeling the need for a clearer diagram myself, I prepared the sketch shown here. I followed B.H.'s diagram in the February '79 VIPER as faithfully as I could. I hope I have not introduced any errors. It is evident that the tri-state RAMs are directly connected. Since the same diagram also appeared in the April '79 IPSO FACTO and was redrawn with errors, I am submitting a copy of the sketch to ACE as well.

In PIPS FOR VIPS, Volume 1, page 31, Tom Swan presents a cursor on/off subroutine for Text Editor-21, located at 03C0-03D2. The function can be performed more elegantly in 10 bytes instead of 19 by substituting an SDI #7F for the XRI #20 at location 03C7 and eliminating the code from 03C8 to 03D0. Subtracting 32_{10} from 127_{10} yields 95_{10} and vice versa, which effectively compliments the cursor each time through. - F. Rodgers, Switzerland

```
03C6  0F  LDN  RF   ;get current value
03C7  FD  SDI #7F16;subtract it from (itself + its
      7F            ;  alternate value)
03C9  5F  STR  RF   ;put back new (alt.) value
03CA  D5  SEP  R5   ;return
(9 bytes now free)  (RF must be initiated to 2016)
```

Editor's Note - From the sun baked Sierra Madres of Mexico to the snowy peaks of your Alps, thanks. You are quite correct. - Tom



DOUBLE-BUFFER GRAPHICS SPEED-UP

CHIP-8III

by John Chmielewski

VIPER has published various modifications of CHIP-8 to implement input/output instructions. Using the VIPER page code scheme, the articles are as follows:

1.03.04 CHIP-8I by Rick Simpson (replaces BMMM with BOKK, B1X0 & B1X1),
1.07.27 Corrected CHIP-8I code (there is still one error in the function B1X1, but it works!),
2.02.08 I/O PORT DRIVER ROUTINE by James Barnes (adds FXF2 & FXF5, but has no provisions for testing EF4).
2.04.05 KEYBOARD KONTROL by Tom Swan (adds FX00 to CHIP-8I).

CHIP-8III is a modification of the functions and code contained in the above referenced articles. The purpose is to try to provide an upgraded CHIP-8 completely compatible with the original. Hopefully, the input/output instructions provided will be suitable for most applications dealing with I/O. If future articles standardize on CHIP-8III readers will benefit by not having a different version of CHIP-8 for each article using the input/output ports.

The following instructions are added to CHIP-8 by CHIP-8III:

```
FX00 - VX = input port if key is pressed
       VX = 0 if no key depressed
       (EF4 is used as the input flag)
FXF2 - VX = input port
       Waits for a key to be pressed & released.
       (EF4 is used as the input flag)
FXF9 - Output port = VX
```

Make the following code additions to CHIP-8 as given under each function name:

FX00

```
0100  3F D4   BN4      ;if no key pressed, set VX=00, exit
0102  E6      SEX  R6  ;set I/O pointer to VX
0103  6B      INP   3  ;set input, put in VX
0104  D4      SEP  R4  ;exit
```

The above code is similar to that in VIPER 2.04.05. By changing location 0100 from B4 to BN4, the routine fits into 5 locations. By changing location 0101 data D4 to 04, the above routine is exactly equivalent to Tom Swan's. But it is one byte shorter, making his modification to the interpreter unnecessary. The above routine will function

as is in the sample program Tom gave, but it's additional feature of setting VX to zero at no keypress eliminates the need for the instruction in the program that sets VX to zero for the keyboard test.

FXF2

```
01F2  E6     SEX  R6  ;set I/O pointer to VX
01F3  3F F3  BN4     ;wait for keypress
01F5  6B     INP  3  ;set input, put in VX
01F6  37 F6  B4      ;wait for key release
01F8  D4     SEP  R4 ;exit
```

The above code is similar to that in VIPER 1.07.27 starting at location 01F2. Note that the code in that issue still contains an error. The corrected code should read: 00F2 3F F2 6B 37 F5 D4.

FXF9

```
01F9  E6     SEX  R6  ;set I/O pointer to VX
01FA  63     OUT  3   ;output VX.  VX = VX +1
01FB  D4     SEP  R4  ;exit
```

The above code is similar to that in VIPER 2.02.08. By eliminating the resetting of R6, the routine by James Barnes fits in the above remaining three memory locations available in CHIP-8.

CHIP-8I did contain one additional instruction not included in CHIP-8III. That is the B0KK instruction. I feel this instruction is of limited use, and as such, I'd rather preserve the complete CHIP-8 interpreter.

* * *

One word of caution. This implementation of the instruction FX00 assumes DF to be zero as the branch to 01D4 is to a section that rotates DF into VX. It seems that this is the case following the decoding of any CHIP-8 instruction, but you should be aware that if DF was equal to one, the FX00 would not result in VX being set to zero. Also, do not output VF with the FXF9. CHIP-8 will crash!
Thank you, John, for consolidating input/output into CHIP-8III. It should have been done long ago. - Tom

LITTLE LOOPS by Tom Swan

IT'S GETTING BETTER ALL THE TIME

The little Mexican children, when they come to visit, love my VIP. Their favorite game is the Figure Shooting at Moving Target. "El Senior chicito con la Pistola" (The little man with the pistol) is what they call him and they all burst out laughing whenever someone reminds them of him. They also cannot understand why my CRT monitor does not pick up TV stations. "So, it's bad isn't it?" they ask me.

Big kids aren't always so impressed. The one comment I hear more often is "Well, it's ok, but it's so slow!"

Then a few months ago, VIPER ran an article on a Fast DXYN instruction which is limited to displaying one bit at a time. It was a nicely done program, and is quite fast. But when I attempted to convert a game such as Wipe Off or "El Senior," there was no easy way around the one bit limitation. I certainly didn't want to construct "El Hombre" out of single dot DXYN's!

For a long time this whole question of speed worried me. As I thought about the Chip-8 display instruction, I came to realize that Chip-8's slowness just didn't make sense. The computer is certainly capable of top speed graphics, but Chip-8 sure doesn't seem to be. Why not?

Then, on the upteenth journey through Chip-8's mysteries I discovered a very interesting fact about the display sub with the help of VIPER's Vol. 1 issue 2 breakdown of the interpreter. With a <u>single byte change</u> I have achieved the same speed of the Fast DXYN instruction! And it works for <u>all</u> Chip-8 programs. Next time someone says Wipe Off is too easy, simply change the byte at location 00AC to EC (normally = 00) and run the program. That's all you have to do and I bet you'll have trouble following the ball let alone playing the game! Same with kaleidoscope. You only need to change the one byte to achieve super fast speed.

As with many modifications, this change has one drawback. <u>Sometimes</u>, but not always, a figure will appear to move with less grace. Location 00AC in the display sub is normally set to 00, an IDL instruction that causes the processor to wait at that location until receiving an interrupt request. <u>Only</u> after the video display chip, the 1861, requests data will the program continue past 00AC. Therefore, the use of a DXYN instruction may actually delay your program up to 1/60 of a second while waiting for the go-ahead signal. That's a lot of time for a computer.

There was a good reason for the IDL instruction at 00AC, however, and you must make a careful decision whether to leave it in or

take it out. For very smooth graphics, it may be needed. . for super fast wall ball games -- definitely not. The purpose of the IDL is to insure that all changes to the display are never interrupted in midpoint causing whatever figure you are displaying to be cut in half for a moment until the interrupt is completed. It was not a design oversight -- in fact I find it to be quite a clever device -- but for many programs you will never know the difference. Except you may need timing loops to slow the balls down!

If you want to see some real speed, try this change along with the HI-RES CHIP-8 modifications and the sample random line drawing program presented elsewhere. Holy Warp Drive Batman!

OPTIMIZATIONS

Speed isn't the only factor of a good computer program. In fact some applications may neglect speed as a negligible advantage allowing for less expensive design. One thing about speed in electronics; it usually comes at a higher price.

Most algorithms, however, are evaluated in terms of speed vs. application. The ability to prove the speed of a program may require a mathematics degree, but you do not need any special training to discover ways to improve your programs. Take Chip-8 again, for example, and let's look for a way to optimize one of its instructions, 00E0 ERASE DISPLAY.

The first things to look for in optimizing routines are loops. Even though a loop may only occupy 10 bytes of computer space, it may be executed 100 times. If each of those bytes is an instruction, the execution time is the same as a routine having 1000 instructions. One-half the size of a 2K VIP! Removing only one instruction from the 10 byte loop will result in 100 instruction executions less each time the loop is used. This times the number of times the loop itself will be needed gives the relative amount of time saved. If the 10-byte, 100 pass loop is used 100 times for example, one less instruction results in a program that is effectively 100 x 100 or 10,000 instructions shorter! Things do mount up and they can mount up fast.

This principal may be used to vastly improve the erase display instruction. Let's look at them side by side. (I've drawn arrows to help in comparing the loops.)

Old 00E0 ERASE DISPLAY #1

```
00E0  9B      GHI  RB ;Display page address is kept in RB.1
  E1  BF      PHI  RF ;RF.1=RB.1/RF is erase pointer
  E2  F8 FF   LDI $FF ;
  E4  AF      PLO  RF ;RF.0=FF (Last byte)
  E5 ┌93      GHI  R3 ;(R3.1=00)
  E6 │5F      STR  RF ;Store zero to erase byte @ M(R(F))
  E7 │8F      GLO  RF ;Test pointer
  E8 │32 DF   BZ      ;If done, exit to 00DF
  EA │2F      DEC  RF ;Pointer -1
  EB └30 E5   BR      ;Loop until 256 bytes erased
```

New 00E0 ERASE DISPLAY #2

```
00E0  9B      GHI  RB ;See above comments -
  E1  BF      PHI  RF ;     set RF = last byte of display
  E2  F8 FF   LDI $FF ;          "          "       "
  E4  AF      PLO  RF ;          "          "       "
  E5  EF      SEX  F  ;X=F eliminating need for STR RF/DEC RF pair
  E6 ┌93      GHI  R3 ;Get 00 byte
  E7 │73      STXD    ;Store @ M(R(F)), pointer automatically
  E8 │8F      GLO  RF  decremented
  E9 └3A E6   BNZ     ;Test low byte of pointer
  EB  5F      STR  RF ;If ≠ 00, then loop to continue erasing
  EC  D4      SEP  R4 ;Erase last display byte (@ 0X00)
                      ;Return
```

Both routines occupy the same space 00E0-00EC. The new subroutine contains 11 instructions, the old has 10 instructions. (The apparent discrepency is from an additional 2 byte branch in routine #1. Timing 1802 programs is simplified by the fact that all instructions except for 3-cycle types not used here, take the same execution time.)

However, the inner loop of the second erase sub is two instructions shorter than sub #1. The first loop will execute 256 times, the second 255 loops with byte #256 erased by the single instruction at 00EB (#2). On the last loop of sub #1, two less instructions will be executed due to the early exit condition at 00E8 (#1).

Let's see how this affects the loops' relative speeds.

	00E0 #1		00E0 #2
Number times loop is executed	256		255
x number instructions in loop	x 6		x 4
	1536		1020
Less special exit on last loop	- 2		- 0
Total executions in loop	1534		1020
Plus number instructions outside loop	+ 4		+ 7
Total executions each call	1538		1027
Difference (1538 - 1027)		511	

Sub #2 results in 511 less instruction executions on each use of the 00E0 command. That's like eliminating two pages of machine language code, and all we did was shorten the loop by two instructions.

The rule is simply: keep everything that does not have to be in the loop out of the loop. Do this even if it results in more code in most cases.

Good luck with your programming. May all your little loops be

as short as possible.

(To all of you who have written: Thank you, I have enjoyed reading your comments. If you'd like a reply, I'd appreciate receiving a stamped, self-addressed envelope due to the number of letters I receive. Eventually, I make it down to the Burro Express office with all of my return correspondence.)

PROJECTS:

1) Design an all purpose Chip-8 subroutine that will allow the action in a game to go progressively faster the longer a person plays the game. Incorporate the change into the Wipe Off program.

2) Design a machine language subroutine that will automatically toggle the DXYN instruction from fast to slow or from slow to fast. It should be completely relocatable, that is, use no branching instructions.

LAST MONTH'S ANSWERS:

1) Equal values are always brought together during any sort. For either a Bubble or an Insertion Sort, like values retain their relative position in memory. For this reason, the two sorting techniques are said to be "stable."

2)

	Bubble Sort	Insertion Sort *
Pass 0	9;1;3;7;2;6;4;8;5	9;1;3;7;2;6;4;8;5
" 1	1;3;7;2;6;4;8;5;9	1;9;3;7;2;6;4;8;5
" 2	1;3;2;6;4;7;5;8;9	1;3;9;7;2;6;4;8;5
" 3	1;2;3;4;6;5;7;8;9	1;3;7;9;2;6;4;8;5
" 4	1;2;3;4;5;6;7;8;9	1;2;3;7;9;6;4;8;5
		1;2;3;6;7;9;4;8;5
*Do not be deceived by the apparent lengths of these results. Insertion is still faster!		1;2;3;4;6;7;9;8;5
		1;2;3;4;6;7;8;9;5
		1;2;3;4;5;6;7;8;9

3) Use V6 V7 for the display X & Y (or other variables not used during sorting). Keep last value entered in V8 and sort after each value is entered. Test if V8=0. If so, branch to display the sorted results. Here's an undocumented Chip-8 Sort Test #3 that will do this which you may use to test either sort sub as described last month.

```
0200  66 00 67 00 6D 00 A4 00
0208  FD 1E F0 0A F0 55 88 00
0210  7D 01 F0 29 D6 75 76 05
0218  23 00 38 00 12 06 60 04
0220  F0 18 66 00 67 08 6C 00
0228  A4 00 FC 1E F0 65 F0 29
0230  D6 75 76 05 7C 01 5C D0
0238  12 28 12 3A 00 00 00 00
```

The insertion sort is "made" for this technique of sorting on-the-fly because of the assumption that all records are already in order before an insertion is made!

NEXT MONTH: PRIME TIME

2.07.11

MUSICODE

by Gilbert Detillieux

The advantage of the music program described in the January, 1979 VIPER is that both the hardware and the software required are very simple. The disadvantage is that encoding music can be long and tedious, especial when a lot of 3/4, 3/8, and 3/16 notes are to be played. These durations must be calculated, since they are not given in the table.

Since the computer is better equipped for handling boring calculations that we are, why not make the VIP do the dirty work? This is what I've done with MusiCode.

The user can enter simple codes for pitch and duration, and the computer will calculate the required frequency and number of cycles to play. MusiCode uses the same hardware as was shown in the January, 1979 VIPER.

The new codes for pitch and duration are shown in Figure 1A and 1B. For each note, enter the pitch code first, then enter the duration code. End the list with ED to stop the tune or with EE to replay the tune from the beginning.

The program ignores memory locations containing 00, so this code can be used to delete a note (or to reserve space for entry of future notes). The tempo is controlled by the byte at location 003D. The rest time between notes is stored in location 0064.

Enter the program beginning at location 0000 (through 00C6). Then begin entering the music list starting at 0100. To play the tune, push key C after having entered RUN mode. If you have entered ED at the end of the list, the music will stop at the end of the tune. You can start it again by pressing key C. The tune will play continuously if you ended the list with EE.

The program shown is for the VIP. You can easily modify it to run on an ELF by changing location 0000 to 3F and location 001A to 64. Then, to hear the tune after it has been entered, push the IN button.

Figure 2 contains a coding sheet for your music codes, as well as reminders for the appropriate keys to press. Hope you all enjoy the "Mystery Tune"!

Figure 1A

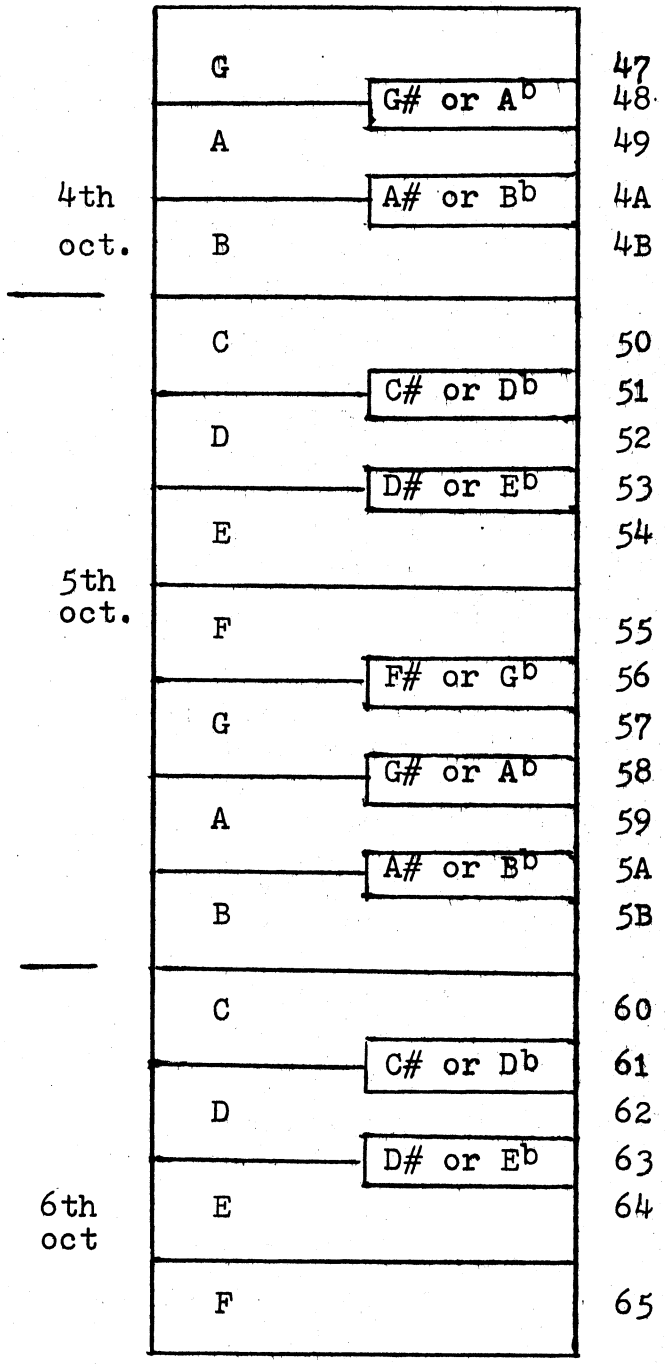


Figure 1B

DURATION	DURATION CODE
Full note	01
Half note	02
1/4 note	04
1/8 note	08
1/16 note	10
1/32 note	20
1 1/2 note	81
3/4 note	82
3/8 note	84
3/16 note	88
3/32 note	90
3/64 note	A0

* Enter ED or EE at end of music list.

Listing I MusiCode

```
0000  3E 00 90 B2 B9 FC 01 B5 F8 00 A5 F8 FF A2 E2 05
0010  FF ED 32 00 33 02 45 32 0F F6 AD 63 22 FC 40 A9 09
0020  A7 89 FC 20 A9 09 A8 05 F6 AD 33 32 83 F8 A8 8J
0030  30 28 45 FE 3B 3C 88 F6 52 88 F4 A8 F8 10 A9 87
0040  FB 43 32 4B 31 49 7B 30 4B 7A E2 87 87 01 3A 4C
0050  89 FF 01 A9 32 5C C4 C4 E2 E2 30 3F 88 FF 01 A8
0060  3A 3C 7A F8 06 B4 24 94 3A 66 30 0F 00 00 00 00
0070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080  00 00 00 00 00 00 00 00 3F 3B 38 34 31 43 43 43
0090  2D 2A 28 25 22 20 1E 1C 1A 18 16 14 43 43 43 43
00A0  13 11 10 0E 0D 0C 00 31 34 37 3A 3E 2D 2D 2D 2D
00B0  41 45 49 4E 52 57 5D 62 68 6E 75 7B 2D 2D 2D 2D
00C0  83 8B 93 9E A5 AE 00 00 00 00 00 00 00 00 00 00
```

Listing II Mystery Tune

```
0100  57 88 59 10 57 08 54 84 4C 08 57 88 59 10 57 08
0110  54 84 4C 08 62 04 62 08 5B 84 4C 08 60 04 60 08
0120  57 84 4C 08 59 04 59 08 60 88 5B 10 59 08 57 88
0130  59 10 57 08 54 84 4C 08 59 04 59 08 60 88 5B 10
0140  59 08 57 88 59 10 57 08 54 04 4C 04 62 04 62 08
0150  65 88 62 10 5B 08 60 84 64 04 4C 04 60 88 57 10
0160  54 08 57 88 55 10 52 08 50 02 4C 04 EE 00 00 00
```

2.07.13

Figure 2

LOC	PITCH CODE	DUR CODE
00		
02		
04		
06		
08		
0A		
0C		
0E		
10		
12		
14		
16		
18		
1A		
1C		
1E		
20		
22		
24		

LOC	PITCH CODE	DUR CODE
26		
28		
2A		
2C		
2E		
30		
32		
34		
36		
38		
3A		
3C		
3E		
40		
42		
44		
46		
48		
4A		

LOC	PITCH CODE	DUR CODE
4C		
4E		
50		
52		
54		
56		
58		
5A		
5C		
5E		
60		
62		
64		
66		
68		
6A		
6C		
6E		
70		

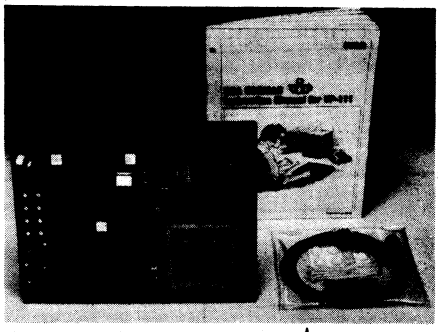
LOC	PITCH CODE	DUR CODE
72		
74		
76		
78		
80		
82		
84		
86		
88		
8A		
8B		
8C		
8E		

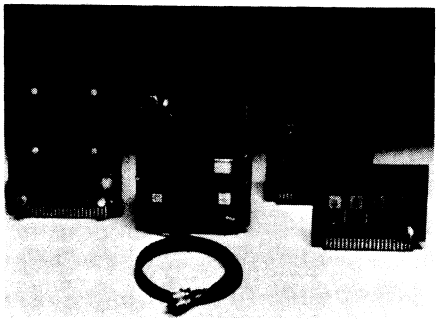
Be sure to enter ED or EE at end of list!

ED = Stop playing the tune
EE = Repeat the tune continuously

CODING SHEET FOR MUSIC - MAY BE COPIED

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.

RCA

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.

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.

2.07.15 VIPER

KILLER ROBOTS
By Carmelo Cortes

First before I describe the game, I'd like to thank Mr. Tom Swan for his books Pips for Vips I and II. For without his utility programs, Text Editor-21, Assembler-3, Chip-8 Program Editor, and Messager, I would have never written this program.

You see I am the lazy sort when it comes to programing, I hate the drudgery of figuring out all those goto and subroutine addresses. And with Assembler-3 I don't have to worry about all that, I can put my full concentration on writing the program.

Also the Chip-8 Program Editor allowed me to easily work on and debug my program after assembly. And the Messager Program lets me display messages so easily, I wonder how I did without it before.

So, I'd like to say, Thank You Tom for putting the "FUN" back in Chip-8 programing for me.

Now the game.

I got the program from an artical in Personal Computing*, for the TRS-8Ø Level 1 basic

Mine is a bit different as dictated by the differences in language. In this game, you are trapped in a room with two manhunting (personhunting?) robots, that have heat sensors for eyes, and so can detect you by your body heat. These will track you down and kill you if they can. But, in the room, unseen and unknown by the robots, are force beams, which give off no heat, and will destroy anything that moves into them (including humans). The human can see, and so can avoid these force beams.

The robots will run into the force beams and/or into each other, in which case the stationary robot will destroy the mobile one. When a robot is destroyed, it will be replaced with another, but in a random location.

The object of the game is to manuvoer yourself in such a way, that the robots move into the force beams or each other. You win when all the robots are destroyed. (There are 2Ø in all).

The game starts on a playing field of 64 squares, (like a chess board) on which 4 walls are placed in random locations, with the two robots being placed next, also randomly. Lastly the human is randomly placed. You move by pressing keys 1 to 9, 1 is upperleft, 2 is up, 3 is upper right, and so on. Pressing 5 will leave you where you are. (So will the rest of the keys but 5 is easier to remember).

```
Ø3ØØ   239A   DO TITLE

  START
Ø3Ø2   2346   DO BORDR
  Ø4   237E   DO WALLS
  Ø6   2334   DO ROBOT
  Ø8   83ØØ
  ØA   847Ø
  ØC   2334   DO ROBOT
  ØE   85ØØ
Ø31Ø   867Ø
  12   236A   DO HUMAN
  14   6DØØ
  16   6A12   # OF ROBOTS
  18   68ØØ
  1A   69ØØ

  MLOOP
Ø31C   23F8   DO KEYP
  1E   38FF
Ø32Ø   24FE   DO RMOV1
  22   39FF
  24   2532   DO RMOV2
  26   48FF
  28   132E   YES GO NEXT
  2A   26ØØ   DO MESS
  2C   131C   GO MLOOP
  2E   49FF
Ø33Ø   15AA   YES GOTO WIN
  32   132A   NO GO BACK

  ROBOT
Ø334   23BC   DO PICK
  36   87ØØ
  38   23BC   DO PICK
  3A   A3D6   RBITS
  3C   DØ78
  3E   3FØ1
Ø34Ø   ØØEE   YES RETURN
  42   DØ78   NO ERASE
  44   1334   GOTO ROBOT

  BORDR
Ø346   A368   BBIT
  48   6ØØØ
  4A   61ØØ
  4C   DØ11
  4E   613F
Ø35Ø   DØ11
  52   7ØØ1
  54   3Ø4Ø
  56   134A LOOP TILL DONE
  58   6Ø3F
  5A   71FF
  5C   DØ11
  5E   6ØØØ
Ø36Ø   DØ11

Ø362   31Ø1
  64   1358   LOOP TILL DONE
  66   ØØEE

  BBIT
Ø368   8ØØØ

  HUMAN
Ø36A   23BC   DO PICK
  6C   81ØØ
  6E   23BC   DO PICK
Ø37Ø   82ØØ
  72   A3DE   HBITS
  74   D128
  76   3FØ1
  78   ØØEE   RETURN
  7A   D128
  7C   136A   DO HUMAN

  WALLS
Ø37E   62Ø4   # OF WALLS
Ø38Ø   23E6   DO PICKY
  82   81ØØ
  84   23E6   DO PICKY
  86   A3CE   WBITS
  88   DØ18
  8A   3FØ1
  8C   1392   NO, GO ANOTHER
  8E   DØ18
Ø39Ø   138Ø   GO TRY AGAIN
  92   72FF
  94   32ØØ
  96   138Ø   LOOP TILL DONE
  98   ØØEE   RETURN

  TITLE
Ø39A   6ØØØ
  9C   6118
  9E   A3AC   MESSC
Ø3AØ   Ø244   CALL MESSAGER
  A2   DØ15
  A4   6C9Ø
  A6   24AC   TIMER
  A8   Ø23Ø   ERASE
  AA   ØØEE

  MESSC
Ø3AC   2Ø4B
  AE   494C
Ø3BØ   4C45
  B2   522Ø
  B4   524F
  B6   424F
  B8   5453
  BA   ØØØØ
```

```
  PICK
Ø3BC   CØØ7
  BE   A3C6   BOXES
Ø3CØ   FØ1E
  C2   FØ65
  C4   ØØEE

  BOXES
Ø3C6   ØØØ8
  C8   1Ø18
  CA   2Ø28
  CC   3Ø38

  WBITS
Ø3CE   5AA5
Ø3DØ   5ABD
  D2   BD5A
  D4   A55A

  RBITS
Ø3D6   ØØ3C
  D8   2424
  DA   1824
  DC   42ØØ

  HBITS
Ø3DE   ØØ18
Ø3EØ   1866
  E2   1824
  E4   24ØØ

  PICKY
Ø3E6   CØØ7
  E8   A3FØ   SBOX
  EA   FØ1E
  EC   FØ65
  EE   ØØEE

  SBOX
Ø3FØ   Ø82Ø
  F2   1Ø18
  F4   2Ø28
  F6   3Ø18

  KEYP
Ø3F8   FEØA
  FA   A3DE   HBITS
  FC   D128
  FE   4EØ1
Ø4ØØ   1426   GO UPLF
  Ø2   4EØ2
  Ø4   143Ø   GO UP
  Ø6   4EØ3
  Ø8   1436   GO UPRT
  ØA   4EØ4
  ØC   144Ø   GO LF
Ø4ØE   4EØ6
Ø41Ø   1446   GOTO RT
  12   4EØ7
  14   144C   GOTO DNLF
  16   4EØ8
  18   1456   GOTO DN
  1A   4EØ9
  1C   145C   GOTO DNRT
  1E   D128
Ø42Ø   4FØ1
  22   1566   GOTO DEAD
  24   ØØEE

  UPLF
Ø426   31ØØ
  28   71F8
  2A   32ØØ
  2C   72F8
  2E   141E   GO SHOW

  UP
Ø43Ø   32ØØ
  32   72F8
  34   141E   GO SHOW

  UPRT
Ø436   3138
  38   71Ø8
  3A   32ØØ
  3C   72F8
  3E   141E   GO SHOW

  LF
Ø44Ø   31ØØ
  42   71F8
  44   141E   GO SHOW

  RT
Ø446   3138
  48   71Ø8
  4A   141E   GO SHOW

  DNLF
Ø44C   31ØØ
  4E   71F8
Ø45Ø   3238
  52   72Ø8
  54   141E   GO SHOW

  DN
Ø456   3238
  58   72Ø8
  5A   141E
```

```
  DNRT
Ø45C   3138
  5E   71Ø8
Ø46Ø   3238
  62   72Ø8
  64   141E    GO SHOW

  DD
Ø466   D348
  68   6CØ8
  6A   24AC    DO TIMER
  6C   D348
  6E   ØØEE

  DD1
Ø47Ø   D568
  72   6CØ8
  74   24AC    DO TIMER
  76   D568
  78   ØØEE

  RDES1
Ø47A   D348
  7C   6EØ1
  7E   FE18
Ø48Ø   A3D6    RBITS
  82   6EØF
  84   D348
  86   D348
  88   7EFF
  8A   3EØØ
  8C   1484    LOOP TILL DONE
  8E   A4B6    DBIT1
Ø49Ø   2466    DO DD
  92   A4BE    DBIT2
  94   2466    DO DD
  96   A4C6    DBIT3
  98   2466    DO DD
  9A   4AØØ
  9C   14A8    YES GO SKIP
  9E   7AFF
Ø4AØ   2334    DO ROBOT
  A2   83ØØ
  A4   847Ø
  A6   ØØEE
  A8   68FF
  AA   ØØEE

  TIMER
Ø4AC   FC15
  AE   FCØ7
Ø4BØ   3CØØ
  B2   14AE    LOOP TILL DONE
  B4   ØØEE

  DBIT1
Ø4B6   AA55
  B8   AA55
  BA   AA55
  BC   AA55

  DBIT2
Ø4BE   55AA
Ø4CØ   55AA
  C2   55AA
  C4   55AA

  DBIT3
Ø4C6   AA55
  C8   AA55
  CA   AA55
  CC   AA55

  RDES2
Ø4CE   D568
Ø4DØ   6EØ1
  D2   FE18
  D4   A3D6    RBITS
  D6   6EØF
  D8   D568
  DA   D568
  DC   7EFF
  DE   3EØØ
Ø4EØ   14D8    LOOP TILL DONE
  E2   A4B6    DBIT1
  E4   247Ø    DO DD1
  E6   A4BE    DBIT2
  E8   247Ø    DO DD1
  EA   14EC    NOP
  EC   4AØØ
  EE   14FA    YES GO SKIP
Ø4FØ   7AFF
  F2   2334    DO ROBOT
  F4   85ØØ
  F6   867Ø
  F8   ØØEE
  FA   69FF
  FC   ØØEE

  RMOV1
Ø4FE   A3D6    RBITS
Ø5ØØ   D348
  Ø2   8B1Ø
  Ø4   8C2Ø
  Ø6   9B3Ø
  Ø8   1518    YES GO NEXT
  ØA   8B35
  ØC   4FØØ
  ØE   73F8
```

2.07.19

```
Ø51Ø   4FØ1
  12   73Ø8
  14   9C4Ø
  16   1522   GO SHOW
  18   8C45
  1A   4FØØ
  1C   74F8
  1E   4FØ1
Ø52Ø   74Ø8
  22   D348
  24   513Ø
  26   152C   NO, GO OVER
  28   924Ø
  2A   1566   GO DEAD
  2C   4FØ1
  2E   247A   DO RDES1
Ø53Ø   ØØEE

  RMOV2
Ø532   A3D6   RBITS
  34   D568
  36   8B1Ø
  38   8C2Ø
  3A   9B5Ø
  3C   154C   YES GO NEXT
  3E   8B55
Ø54Ø   4FØØ
  42   75F8
  44   4FØ1
  46   75Ø8
  48   9C6Ø
  4A   1556   GO SHOW
  4C   8C65
  4E   4FØØ
Ø55Ø   76F8
  52   4FØ1
  54   76Ø8
  56   D568
  58   515Ø
  5A   156Ø   GO OVER
  5C   926Ø
  5E   1566   GO DEAD
Ø56Ø   4FØ1
  62   24CE   DO RDES2
  64   ØØEE

  DEAD
Ø566   A3DE   HBITS
  68   D128
  6A   26D2   DO HDEST
  6C   Ø5E4   CALL MLS1
  6E   Ø23Ø
Ø57Ø   6E1Ø
  72   A67Ø   MESS5
  74   Ø244   CALL MESSAGER
  76   DDE5

Ø578   6C8Ø
  7A   24AC   DO TIMER
  7C   Ø23Ø
  7E   A67C   MESS6
Ø58Ø   Ø244   CALL MESSAGER
  82   DDE5
  84   6E18
  86   A688   MESS7
  88   Ø244   CALL MESSAGER
  8A   DDE5
  8C   6C8Ø
  8E   24AC   DO TIMER
Ø59Ø   Ø23Ø
  92   A6B4   MESSA
  94   6E1Ø
  96   Ø244   CALL MESSAGER
  98   DDE5
  9A   A6C4   MESSB
  9C   6E18
  9E   Ø244   CALL MESSAGER
Ø5AØ   DDE5
  A2   Ø5EØ   CALL MLS3
  A4   15D4   GOTO BACK
  A6   15D4   NOP
  A8   15D4   NOP

  WIN
Ø5AA   Ø5E4   CALL MLS1
  AC   Ø23Ø
  AE   A696   MESS8
Ø5BØ   6E1Ø
  B2   Ø244   CALL MESSAGER
  B4   DDE5
  B6   6E18
  B8   A6A4   MESS9
  BA   Ø244   CALL MESSAGER
  BC   DDE5
  BE   6C9Ø
Ø5CØ   24AC   DO TIMER
  C2   Ø23Ø
  C4   A6B4   MESSA
  C6   6E1Ø
  C8   Ø244   CALL MESSAGER
  CA   DDE5
  CC   A6C4   MESSB
  CE   6E18
Ø5DØ   Ø244   CALL MESSAGER
  D2   DDE5

  BACK
Ø5D4   FFØA
  D6   3FØF
  D8   15D4   NO, GOTO BACK
  DA   Ø5EA   CALL MLS2
  DC   Ø23Ø
  DE   13Ø2   GOTO START
```

```
  MLS3
Ø5EØ   Ø312
  E2   12D4

  MLS1
Ø5E4   Ø19B
  E6   FFØ3
  E8   BBD4

  MLS2
Ø5EA   Ø29B
  EC   FCØ3
  EE   BBD4

Ø5FØ   ØØØØ   NOT USED
  F2   ØØØØ    "    "
  F4   ØØØØ    "    "
  F6   ØØØØ    "    "
  F8   ØØØØ    "    "
  FA   ØØØØ    "    "
  FC   ØØØØ    "    "
  FE   ØØØØ    "    "

  MESS
Ø6ØØ   CØØ7
  Ø2   4ØØØ
  Ø4   ØØEE
  Ø6   4ØØ1
  Ø8   A638   MESS1
  ØA   4ØØ2
  ØC   ØØEE
  ØE   4ØØ3
Ø61Ø   A648   MESS2
  12   4ØØ4
  14   ØØEE
  16   4ØØ5
  18   A656   MESS3
  1A   4ØØ6
  1C   ØØEE
  1E   4ØØ7
Ø62Ø   A664   MESS4
  22   6C3Ø
  24   24AC   DO TIMER
  26   Ø5E4   CALL MLS1
  28   Ø23Ø
  2A   6E18
  2C   Ø244   CALL MESSAGER
  2E   DDE5
Ø63Ø   6C3Ø
  32   26E2   DO ERASE
  34   Ø5EA   CALL MLS2
  36   ØØEE

  MESS1
Ø638   2Ø52
  3A   554E
Ø63C   2Ø4C
  3E   494B
Ø64Ø   452Ø
  42   4845
  44   4C4C
  46   21ØØ

  MESS2
Ø648   2Ø2Ø
  4A   2Ø2Ø
  4C   4C4F
  4E   4F4B
Ø65Ø   2Ø4F
  52   5554
  54   21ØØ

  MESS3
Ø656   2Ø4E
  58   4F54
  5A   2Ø54
  5C   4841
  5E   542Ø
Ø66Ø   5741
  62   59ØØ

  MESS4
Ø664   2Ø2Ø
  66   2Ø2Ø
  68   2Ø59
  6A   494B
  6C   4553
  6E   21ØØ

  MESS5
Ø67Ø   2Ø2Ø
  72   2Ø2Ø
  74   4152
  76   5252
  78   4721
  7A   21ØØ

  MESS6
Ø67C   2Ø2Ø
  7E   2Ø2Ø
Ø68Ø   534F
  82   5252
  84   5921
  86   21ØØ

  MESS7
Ø688   2Ø2Ø
  8A   594F
  8C   5527
  8E   5645
Ø69Ø   2Ø44
  92   4945
  94   44ØØ
```

```
  MESS8
Ø696   2Ø2Ø
  98   2Ø2Ø
  9A   414C
  9C   5249
  9E   4748
Ø6AØ   5421
  A2   ØØØØ

  MESS9
Ø6A4   2Ø2Ø
  A6   594F
  A8   552Ø
  AA   4D41
  AC   4445
  AE   2Ø49
Ø6BØ   5421
  B2   ØØØØ

  MESSA
Ø6B4   5Ø52
  B6   4553
  B8   532Ø
  BA   4B45
  BC   592Ø
  BE   462Ø
Ø6CØ   544F
  C2   ØØØØ

  MESSB
Ø6C4   2Ø2Ø
  C6   5354
  C8   4152
  CA   542Ø
  CC   4147
  CE   4149
Ø6DØ   4EØØ

  HDEST
Ø6D2   FE18
  D4   6E15
  D6   D128
  D8   D128
  DA   7EFF
  DC   3EØØ
  DE   16D6
Ø6EØ   ØØEE

  ERASE
Ø6E2   24AC
  E4   Ø23Ø
  E6   ØØEE
```

LOADING INSTRUCTIONS

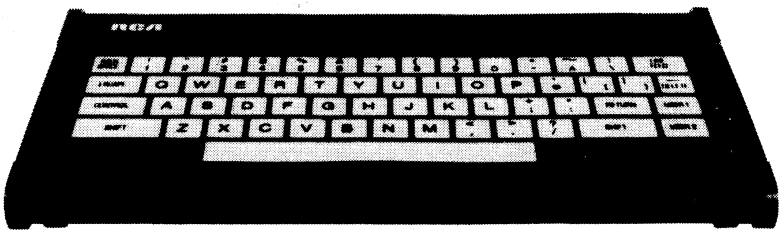
This program uses the 2-page display described in the september, 1978 issue of the VIPER, with the changes made by Tom Swan as described in his book, Pips for Vips I. These can be found in the Chapter intitled, Character Designer, on pages 82 & 83. next you must load his Messager program at Ø244, making the changes desribed on page 1Ø2. Then load 2 pages of the character set, at Ø7ØØ, again from his Character Designer program. Last you must change the byte at Ø25E to Ø7, and the bytes at Ø24D and Ø28B to FD.

Then load 4 pages of the program starting at Ø3ØØ. Flip up the reset switch and play.

HAVE FUN!

2.07.22

ASCII encoded keyboards as low as $65.*



The RCA VP-601 keyboard has a 58 key typewriter format for alphanumeric entry. The VP-611 ($15 additional*) offers the same typewriter format plus an additional 16 key calculator type keypad.

Both keyboards feature modern flexible membrane key switches with contact life rated at greater than 5 million operations, plus two key rollover circuitry.

A finger positioning overlay combined with light positive activation key pressure gives good operator "feel", and an on-board tone generator gives aural key press feedback.

The unitized keyboard surface is spillproof and dustproof. This plus the high noise immunity of CMOS circuitry makes the VP-601 and VP-611 particularly suited for use in hostile environments.

The keyboards operate from a single 5-volt, DC power supply, and the buffered output is TTL compatible. For more information contact RCA Customer Service, New Holland Avenue, Lancaster, PA 17604.

Or call our toll-free number: 800-233-0094.

RCA

*Optional user price. Dealer and OEM prices available.

2.07.23

SUPER DUPER 3-WAY MEMORY DIAGNOSTIC

by Tom Swan

If you are experiencing unusual difficulties getting your programs to run, you may have a bad memory chip. Though your VIP manual contains a simple memory test, it is not designed to catch what could be subtle defects in RAM. Plus, it is difficult to test all of memory as that program was written to do only a small section at a time.

This 3-way memory diagnostic will put your VIP through the paces using byte manipulations that duplicate most any condition RAM is likely to experience during normal program runs -- and then some. Even pattern sensitivities are unlikely to escape scrutiny. Should your VIP pass this test, you can be sure your memory is as good as Dumbo's. If not, you'll know exactly where the error occurred.

HOW TO OPERATE

Load the following machine language program starting at 0000, then save one page (also at 0000) on tape for future uses. Flip to run and the test will begin. All available on-card RAM except for 0000-00FF will be tested. If you suspect an error in the first 256 bytes of RAM, you should switch that chip first with a known good one, then run the test.

There are three parts to the test and each page is displayed on screen while it is being tested.

1) TEST A -- Full Page: Each page of RAM is filled with bytes from 00 to FF then the <u>full</u> page is checked for accuracy.

2) TEST B -- Byte by Byte: Each byte is filled with values from 00-FF and each <u>value</u> is checked for accuracy.

3) TEST C -- Sequencer: An entire page is filled with a sequence from 00-FF. The sequence is swapped front to back twice then checked for accuracy.

The test advances through each page and will end automatically when the last page has been tested. This will hold true for 2K, 3K or 4K systems. At the end of the test, provided there were no errors, all memory from 0100-0XFF (0X=highest on-card RAM page) is cleared to 00 bytes and an automatic return to the VIP operating system is performed so you do not have to reset the computer to continue programming.

If there is an error, the test will halt and a beeper will sound until you shut it off. The bad byte's address is then contained in R5 and is also deposited at 00E0-00E1 which you may examine using the VIP operating system. To hear what this sounds like, change: 00A0; F8 01 A4 30 06 but reload the original program before an actual test.

The program could be sped up by removing the timing loops, but these were included to give a bad byte the time to do whatever it is doing wrong. A slow leak (indicating a power supply problem most likely) could be missed by too fast of a test. You may want to <u>increase</u> the timing loop values for an even slower test. These are at locations 002A and 0069 for Test A and C. Test B's speed can not be as easily changed.

TESTING OFF-CARD RAM

When you purchase a new block of memory, you should run this memory test to be sure your RAM operates properly. The memory test needs to be adjusted, however, for testing off-card RAM.

Make the following changes:

```
00A0  A4  PLO   R4   --R4.0=00 (00 was in D)
  A1  F8  LDI
  A2  XX           --Insert start page minus one
  A3  B4  PHI   R4
  A4  F8  LDI
  A5  XX           --Insert end page address
  A6  BE  PHI   RE
  A7  30  BR       --Return from patch
  A8  06
```

At 00A2 you must insert the starting address of the RAM block to be tested <u>minus</u> one. If your off-card RAM starts at address 1000, you would place the value 0F in 00A2. At 00A5, place the address of the last page of memory to be tested. If your memory extends to 1FFF, insert the value 1F in 00A5.

Also make the following change:

```
0092  F8  LDI      --
  93  XX           --Insert high on-card address
```

Place the value of your highest on-card RAM page at 0093 to insure a proper return to the VIP operating system when testing off-card RAM. (For a 4K system this would be the hex value "0F"; 3K="0B" and 2K="07".)

FURTHER NOTES

The program is straightforward and should be understandable with the help of the comments to those of you with a little 1802 experience. You do not need to understand it to use it of course. Nothing wrong with that.

If you like developing your own system software, you may be interested in the section that causes an automatic return to the VIP operating system without having to reset, flipping to run with Key C depressed. It offers a unique way to end a program such as this memory test.

First all memory is cleared (except for page "0") with the routine at 0089-008F though this step is not necessary.

At 0091 the video is turned off. It will be turned on again momentarily but we need to use R1 which serves as the interrupt program counter while the video chip is on. After turning the video off, R1 is set to the address of the last byte of on-card RAM. The operating system needs this address to set up the display properly.

Next, R2 is set to 8028 where the operating system normally begins to run when you flip the run switch up while holding Key C down. P is set equal to 2, and the operating system begins running in R2.

That's it -- not too complicated, but a neat trick you may want to steal for your own stuff.

Best of luck!

REGISTER ASSIGNMENT

R0 -- DMA pointer--set in interrupt
R1 -- Interrupt program counter (at 8146)
R2 -- Stack pointer (at 00FF and down)
R3 -- Program counter starting at 0013
R4 -- Holds test page address
R5 -- Work register for byte manipulations
R6 -- " " " " "
R7 -- Not used
R8 -- R8.1=timer/R8.0=tone/both in interrupt
R9 -- Not used (but changed by interrupt routine)
RA -- Not used
RB -- RB.1 hold display page
RC -- Not used
RD -- Not used
RE -- RE.1 highest page to be tested
RF -- RF.0 used to hold test values

3-WAY MEMORY DIAGNOSTIC

```
0000  90  GHI   R0 ;R0.1=00 on start
  01  B2  PHI   R2 ;R2.1=00 - initialize stack pointer
  02  B3  PHI   R3 ;R3.1=00 -     "       PC
  03  B4  PHI   R4 ;R4.1=00 -     '       test pointer
  04  30  BR       ;Branch to patch @ 00A0
  05  A0
  06  F8  LDI
  07  81           ;Initialize R1=8146
  08  B1  PHI   R1
  09  F8  LDI      ;For use as interrupt PC
  0A  46
  0B  A1  PLO   R1
  0C  F8  LDI   FF ;R2.0=FF to complete
  0D  FF
  0E  A2  PLO   R2 ;Initialization of stack pointer (00FF)
  0F  F8  LDI

0010  13
  11  A3  PLO   R3 ;R3 is ready to become PC
  12  D3  SEP   R3 ;R3 is program counter
  13  E2  SEX   2  ;Stack pointer is R2
  14  22  DEC   R2 ;Prepare to turn on video
  15  69  INP   9  ;Video on
  16  12  INC   R2 ;Reset stack pointer
  17  94  GHI   R4 ;Start main loop
  18  FC  ADI      ;Increase R4 to next memory page
  19  01           ;For test (0100 first time through)
  1A  B4  PHI   R4
  1B  BB  PHI   RB ;Also set display to view page tested
  1C  F8  LDI      ;Set RF.0=00 (test byte holder)
  1D  00           ;    "    "       "       "
  1E  AF  PLO   RF ;    "    "       "       "
```

TEST A -- FULL PAGE

```
001F  94  GHI   R4 ;Reset/set R5 to test page
  20  B5  PHI   R5 ;    "   "  "      "
  21  84  GLO   R4 ;R5 will be work register
  22  A5  PLO   R5
  23  8F  GLO   RF ;Get test byte from RF.0
  24  55  STR   R5 ;Store at test location
  25  15  INC   R5 ;Next location
  26  85  GLO   R5 ;Test if R5.0=00
  27  3A  BNZ      ;If ≠ 00, loop to fill page with
  28  23           ;        value in RF.0
  29  F8  LDI      ;Load R8.1 with maximum timer value
  2A  04           ;    "                "      "      "
  2B  B8  PHI   R8 ;    "                "      "      "
  2C  98  GHI   R8 ;Test R8.1 (decremented each interrupt)
```

```
002D  3A  BNZ      ;If ≠ 0, loop to waste time here
  2E  2C           ;    "          "          "
  2F  94  GHI  R4  ;Reset R5 to top of test page

0030  B5  PHI  R5  ;  "          "          "
  31  8F  GLO  RF  ;RF.0 holds current test byte
  32  52  STR  R2  ;Push test value onto stack
  33  05  LDN  R5  ;Get byte from location being tested
  34  F3  XOR      ;Compare with byte on stack
  35  3A  BNZ      ;If ≠, branch to error (byte in R5)
  36  B0
  37  15  INC  R5  ;R5 + 1 for next byte
  38  85  GLO  R5  ;Test if R5.0=00
  39  3A  BNZ      ;If not = 00, loop to test full page
  3A  33
  3B  1F  INC  RF  ;RF + 1 for next test value
  3C  8F  GLO  RF  ;Test RF.0
  3D  3A  BNZ      ;If ≠ 0, loop to do full range of test
                    values
  3E  1F           ;(Test A ends when RF.0=00 again)


                   TEST B -- BYTE BY BYTE
003F  94  GHI  R4  ;Reset R5 to top of test page
  40  B5  PHI  R5  ;(R5.0 already = 00)
  41  8F  GLO  RF  ;Get test value--RF.0=00 to start
  42  52  STR  R2  ;Push test value onto stack
  43  55  STR  R5  ;And also at current test location
  44  E2  SEX  R2  ;NOP's -- kill time
  45  E2  SEX  R2  ;  "        "     "
  46  E2  SEX  R2  ;  "        "     "
  47  E2  SEX  R2  ;  "        "     "
  48  05  LDN  R5  ;Get back value from test location
  49  F3  XOR      ;Compare with value on stack
  4A  3A  BNZ      ;If ≠, branch to error (byte in R5)
  4B  B0
  4C  1F  INC  RF  ;Next test value
  4D  8F  GLO  RF  ;Test RF.0
  4E  3A  BNZ      ;If RF.0 ≠ 00, loop to test next value
  4F  41

0050  15  INC  R5  ;Next test location
  51  85  GLO  R5  ;Test R5.0
  52  3A  BNZ      ;If R5.0 ≠ 00, loop to test next location
  53  41


                    TEST C -- SEQUENCER
0054  94  GHI  R4  ;Reset R5 to top test page
  55  B5  PHI  R5
```

```
0056  8F  GLO   RF ;Get initial test value (00)
  57  55  STR   R5 ;Store value at test location
  58  15  INC   R5 ;Next test location
  59  1F  INC   RF ;Next value
  5A  8F  GLO   RF ;Test RF.0
  5B  3A  BNZ      ;If ≠ 00, loop to full page
  5C  57               with sequential values 00-FF
  5D  25  DEC   R5 ;Reset R5 to last byte on test page
  5E  94  GHI   R4 ;Set R6 to first byte on test page
  5F  B6  PHI   R6 ;  "                "          "

0060  84  GLO   R4 ;  "                "          "
  61  A6  PLO   R6 ;  "                "          "
  62  06  LDN   R6 ;Begin exchange
  63  52  STR   R2 ;Push M(R(6))
  64  05  LDN   R5 ;Get M(R(5))
  65  56  STR   R6 ;Store @ M(R(6))
  66  02  LDN   R2 ;Pop old M(R(6))
  67  55  STR   R5 ;Store @ M(R(5))
  68  F8  LDI      ;Timer value
  69  02
  6A  B8  PHI   R8 ;Put in R8.1 (auto decrement)
  6B  98  GHI   R8 ;Test R8.1
  6C  3A  BNZ      ;If ≠ 00, loop to kill time here
  6D  6B
  6E  16  INC   R6 ;Next location down
  6F  25  DEC   R5 ;Next location up

0070  86  GLO   R6 ;Test R6.0
  71  3A  BNZ      ;If ≠ 00, loop to continue exchange
  72  62
```

TEST SEQUENCE

```
0073  15  INC   R5 ;Reset R5 to top of test page
  74  F8  LDI      ;Make sure RF.0 = 00 -- first value
  75  00           ;  "        "          "     "
  76  AF  PLO   RF ;  "        "          "     "
  77  05  LDN   R5 ;Get value at test location
  78  52  STR   R2 ;Push onto stack
  79  8F  GLO   RF ;Get current test value
  7A  F3  XOR      ;Compare with byte on stack
  7B  3A  BNZ      ;If ≠, branch to error (R5 contains
  7C  B0               bad byte
  7D  15  INC   R5 ;Next test location
  7E  1F  INC   RF ;Next test value
  7F  85  GLO   R5 ;Test if R5.0=00

0080  3A  BNZ      ;If ≠ 00, loop until page is tested
  81  77               (R5 set to next highest page)
  82  25  DEC   R5 ;R5 points to last byte current page
```

```
0083  95  GHI   R5 ;Test R5.1
  84  52  STR   R2 ;Push R5.1 onto stack
  85  9E  GHI   RE ;Get saved value of top memory page
  86  F3  XOR      ;Compare with byte on stack
  87  3A  BNZ      ;If ≠, loop to do another page
  88  17           ;Test ends when R5.1=RE.1
```

NO ERRORS -- EXIT ROUTINE

```
0089  F8  LDI      ;Begin erase of memory
  8A  00
  8B  55  STR   R5 ;Store 00 byte @ M(R(5))
  8C  25  DEC   R5 ;Next byte
  8D  95  GHI   R5 ;Test R5.1
  8E  3A  BNZ      ;If ≠ 00, loop to erase all but last
  8F  89            memory page
```

RETURN TO OPERATING SYSTEM

```
0090  22  DEC   R2 ;Prepare for turning off video
  91  61  OUT   1  ;Turn video off
  92  E2  SEX   2  ;NOP (see notes on testing off-card RAM)
  93  9E  GHI   RE ;RE.1 contains high on-card RAM page
  94  B1  PHI   R1 ;Put in R1 to duplicate operating
  95  F8  LDI      ;System start-up conditions
  96  FF
  97  A1  PLO   R1 ;R1 = 0XFF
  98  F8  LDI      ;Set R2 (no longer stack pointer
  99  80            now) to 8028 where the
  9A  B2  PHI   R2  operating system begins
  9B  F8  LDI       running (normally on Key C/run)
  9C  28
  9D  A2  PLO   R2
  9E  D2  SEP   R2 ;Call operating system- exit memory test
```

PATCH

```
00A0  A4  PLO   R4 ;R4.0=00
  A1  91  GHI   R1 ;Get high memory page
  A2  BE  PHI   RE ;Store in RE.1
  A3  30  BR       ;Branch back from patch
  A4  06
```

ERROR DETECTED

```
00B0  F8  LDI      ;Set RF = 00E0
  B1  00           ; "  "     "
  B2  BF  PHI  RF  ; "  "     "
  B3  F8  LDI      ; "  "     "
  B4  E0           ; "  "     "
  B5  AF  PLO  RF  ; "  "     "
  B6  95  GHI  R5  ;Store error byte in R5
  E7  5F  STR  RF  ;At 00E0 via RF
  B8  1F  INC  RF  ; "  "     "
  B9  85  GLO  R5  ; "  "     "
  BA  5F  STR  RF  ; "  "     "
  BB  F8  LDI      ;03 = tone value
  BC  0F
  BD  A8  PLO  R8  ;Sound tone
  BE  F8  LDI      ;02 = timer value
  BF  1F

00C0  B8  PHI  R8  ;Set timer = 2X tone
  C1  98  GHI  R8  ;Test timer
  C2  3A  BNZ      ;If ≠ 0, loop to kill time
  C3  C1
  C4  30  BN       ;Then repeat to beep
  C5  BB
```

ORDERING INFORMATION

On a blank sheet of paper, write your name, address, city, state, and zip code. And, if you plan to use a credit card for payment, write the card number and expiration date. If you're using Master charge, there are four other digits above your name; the "Interbank number" - and we need that number, too. If you want us to ship your order via UPS COD, give us a street address (not a postoffice box number) and write "COD" instead of the credit card number. Then write down the name of the product you'd like to receive and put it all in an envelope (with your check, if you're not using a credit card or COD) and mail it to us!

PIPS FOR VIPS VOLUME I...$19.95
PIPS FOR VIPS VOLUME II...$19.95
PIPS FOR VIPS VOLUME III.$19.95
PIPS FOR VIPS VOL._____$14.95*
*without the program cassette!
VIPER VOL. I **or** VOL II...$15.00
Studio II Stuff:
PROM Card (assembled)..$55.00
MicroStudio News...$10/6 issues
RAM Card (assembled, but without the two 2114 RAM ICs..$45.00
Backplane Card (4 connectors mounted, not soldered)...$36.00

Please note that you have to bring two signals out from the S II in order to use the RAM card! All appropriate paperwork is included with the card you order. We also have the RCA service manual & Studio II schematic ($3.50) and the 1861 data sheet ($1.50) You can get the original information package, which will tell you how to do the conversion on your own, without buying our cards, for $5.00. Can't hardly lose, can you?