VIPER

February - March 1982

Volume 3, Number 6 Journal of the VIP Hobby Computer Assn.
The VIPER was founded by ARESCO, Inc. in June 1978

****

Contents

The VIPER, founded by ARESCO, Inc., in July 1978, is the Official Journal of the VIP Hobby Computer Association. Acknowledgement and appreciation is extended to ARESCO for permission to use the VIPER name. The Association is composed of people interested in the VIP and computers using the 1802 microprocessor. The Association was founded by Raymond C. Sills and created by a Constitution, with By-laws to govern the operation of the Association. Mr. Sills is serving as Director of the Assocation, as well as editor and publisher of the VIPER.

The VIPER will be published six times per year and sent to all members in good standing. Issues of the VIPER will not carry over from one volume to another. Individual copies of the VIPER and past issues, where they are available, may be sent to interested people for $3 each. Annual dues to the Associatioh, which includes six issues of the VIPER, is $12 per year.

VIP and COSMAC are registered trademarks of RCA Corporation. The VIP Hobby Computer Associatin is in no way associated with RCA, and RCA is not responsible for the contents of this newsletter. Members should not contact RCA regarding material in the VIPER. Please send all inquiries to VIPHCA 32 Ainsworth Avenue, East Brunswick, NJ 08816.

Membership in the VIP Hobby Computer Association is open to all people who desire to promote and enjoy the VIP and other 1802 based systems. Send a check for $12 payable to "VIP Hobby Computer Association" c/o R. C. Sills at the above address. People outside of the United Stated, Canada or Mexico please include $6 extra for postage. All funds must be in U>S. Dollars.

Contributions by members or interested people are welcome at any time. Material submitted by you is assumed to be free of copyright restrictions and will be considered for publication in the VIPER. Articles, letters, programs, etc., in camera-ready form on 8.5 by 11 inch paper will be given preferential consideration. Please send enough information about any program so that the readers can operate the program. Fully documented programs are best, but "memory dumps" are OK if you provide enough information to run the program. Please indicate in your material any key memory locations and data areas.

ADVERTISING RATES

1. Non-commercial classified ads from members, 5 cents per word, minimum of 20 words. Your address or phone number is free.

2. Commercial ads and ads from non-members, 10 cents per work, minimum of 20 words. Your address or phone number is free.

3. Display ads from camera ready copy, $6/half page, $10/page.

Payment in full must accompany all ads. Rates are subject to change.

If you write to VIPER/VIPHCA, please indicate that it is OK to print your address in your letters to the editor, if you want that information released. Otherwise, we will not print your address in the VIPER.

DATA ENCRYPTION FOR COSMAC

by Tom Swan

Although sharing is a Viper motto, you may have a reason from time to time for encrypting or coding your files so only you can use them. Even if you don't have anything to hide, however, I think you'll find this method for data encryption fascinating and simple.

There are many algorithms for coding a computer's memory or any group of data. Banks make heavy use of data encryption to prevent theft, and people who use large time-sharing systems and networks often encode their files because of the many eyes constantly peering into the memory banks of these systems. Naturally, there are just as many people who spend lots of time breaking into everyone else's codes and reading their things, if only for the fun and satisfaction of puzzling out the latest uncrackable method.

Most of the data encryption algorithms require the use of a key, a mature term for a 'secret code.' You could use your initials, the name of a pet dog, or some other term familiar to you and not likely to be guessed by others. Of course you don't want to forget your key or you'll find yourself breaking into your own security system late into the night just to read that love letter you were composing on the old CRT the other evening.

In general, the longer the key, the better the security. Even more effective is to use two or more keys and encode everything several times. In machine language, the time to encrypt 4,000 bytes, the typical VIP memory size, is less than one second regardless of the length of the key.

THE STANDARD METHOD

The exclusive-or instruction is typically used to encrypt bytes in a computer's memory. Any byte exclusive-ored with another is sufficiently scrambled to make it unreadable. The interesting thing about the exclusive-or is its ability to recover the original information by simply re-exclusive-oring the same thing again.

In other words:

$AB XOR $CD --> $66 XOR $CD --> $AB

You can see this quality a lot easier in the following sequences:

$FF XOR $00 --> $FF XOR $00 --> $FF

$FF XOR $FF --> $00 XOR $FF --> $FF

On a bit-for-bit level the above two examples prove that the exclusive-or works as a flip flop when applied twice in a row to some value. The original value is always recovered the second time through.

Exclusive-oring all of memory with some value would be one crude way of encrypting your data. The result is fairly easy to crack, however, even if it means performing 256 tests to find the code that was used. A large file can be cracked by checking only the first several bytes, exclusive-oring these with every byte value, and looking for the one value that makes sense, that is, the key that turns the junk into words, program code, CHIP-8 instructions, or whatever.

A superior approach is to use a string as the encryption key. The first character (byte) of the string is applied to the first byte of the data, the second character to the second byte, etc. If, as is likely, the end of the string is reached before the end of the data, we start over again with the first character of the key string and continue.

Only if the same key string is used to decode the encrypted data will the original be recovered. The characters in the string must also be in the same order. Curiously, if two or more separate strings are applied at different times, the order of decoding the data is not important. If you encode a file using 'MICKEY' and then 'MOUSE', in other words, you could decode the same file by <u>first</u> using 'MOUSE' and then 'MICKEY' and still end up with the original.

The following program uses the standard scheme to encrypt any range of bytes in the VIP's memory. Any 1802 computer can use the program. If you have additional memory as I do in my VIP system, you can store the routine and flip the write protect switch on your memory card to keep the code there when you need it without having to reload from tape.

The program is written to run beginning on any memory page boundary <u>except</u> for page zero. The first few bytes of memory must be set up by you before calling the program. These are:

Address	Instruction	Description
0000	C0 xx yy	; Long branch to address xxyy where the encryption program lives.
0003	nn nn	; Where nnnn=the number of bytes+1 to encrypt.
0005	xx yy	; Where xxyy=the address of the last byte to encrypt.
0007	aa bb cc .. 00	; Where aa bb cc .. is a string of any length ending with a 00 byte.

Let's say you want to encrypt a Text Editor-21 text file. (Text Editor-21 is included and described in my Pips for Vips books, Volumes I and II. Any text file or any other kind of file may be encrypted with the following program, however.)

The text file is six pages long. We will hand load the file from tape into memory addresses 0100 through 06FF, encrypt the file, then write it back to tape in encoded form. Because the exclusive-or instruction can be used to recover the original information, the exact same procedure will be used to decode the encrypted file.

We have stored the data encrypting program at memory addresses $0E00 through $0E21. Our secret code, or key, will be the word 'COSMAC' represented in ASCII form. After entering the encrypting program and the text file, we would key in:

```
0000        C0 0E 00           ;Jump to encryption program at $0E00
0003        06 00              ;Length of file + 1 in bytes
0005        06 FF              ;Address ($06FF) of end of file
0007        43 4F 53 4D 41 43 00   ;'COSMAC' as an ASCII string
                                    ending with a 00 byte.
```

Flip to run and the text file will be encoded. Flip to run a second time and it will be transformed back to the original. The same program and key will code as well as decode the file.

I have done one thing a little differently from the standard encryption algorithm. The standard method normally codes a file from the lowest memory address to the highest. The following COSMAC version works in the reverse, starting from the last memory address and proceding to the first byte in the file. This was done to take advantage of the way a stack pointer (RD in the program) can point anywhere in memory and be decremented with a single store (STXD) instruction. The key string, however, is processed character by character in normal order.

The next time you have something to hide, the Cosmac Data Encryption Program will accomplish the task. Oh, one little caution. Don't forget your keys.

```
;                            ----------------------------------
;                            |                                |
;                            |  COSMAC DATA ENCRYPTION PROGRAM |
;                            |                                |
;                            ----------------------------------
;
;               Will run on any page boundary (nn) except nn=00

nn00  F8 00     ENCRYPT:LDI #00  ;set RF to address parameters
  02  BF                PHI RF   ;...at address $0003
  03  F8 03             LDI #03
  05  AF                PLO RF
  06  4F                LDA RF   ;pick up number of bytes + 1
  07  BC                PHI RC
  08  4F                LDA RF
  09  AC                PLO RC   ;RC = number of bytes + 1 (count)
  0A  4F                LDA RF   ;pick up ending address
  0B  BD                PHI RD
  0C  4F                LDA RF
  0D  AD                PLO RD   ;RD = ending address
  0E  ED                SEX RD   ;set X=D for stack instructions
  0F  9F        E1:     GHI RF

nn10  BE                PHI RE   ;set RE=RF -- address of string
  11  8F                GLO RF   ;...also resets RE on loops back
  12  AE                PLO RE
  13  4E        E2:     LDA RE   ;get a character (byte) of string
  14  32 0F             BZ  E1   ;if at end of string, go reset RE
  16  F3                XOR      ;exclusive or with M(R(X)) (X=D)
  17  73                STXD     ;store and decrement at M(R(X))
  18  2C                DEC RC   ;count bytes done
  19  9C                GHI RC   ;...check high byte of count
  1A  3A 13             BNZ E2   ;loop if not done
  1C  8C                GLO RC   ;...check low byte of count
  1D  3A 13             BNZ E2   ;loop if not done
  1F  7B                SEQ      ;set Q line high to signal when done

  20  30 20     E3:     BR  E3   ;stop

                        .END
```

FOUR INPUT ENERGY LOGGER PROGRAM

G. Endres, 38 Yantecaw Av, Bloomfield, NJ 07003

<u>BRIEF</u>: This program is designed to run for up to 9 days at a time and to use the VIP Microcomputer as a logging device. The program emulates a 4 channel event recorder such as might be used in making heating surveys of thermostat cycling. It has seen many other uses as well. Four K of memory is recommended.

The backbone of this system is a clock/calendar routine that is easily adjusted to run with an accuracy of a few seconds per week. It calls up subroutines that run four seperate elapsed time counters which measure events in terms of run time per hour in intervals of seconds. Also, at the end of each hour, it sequentially stores the date, time, and counter data away in memory, then resets the counters for the next hour.

A single software patch allows access to the stored data from another routine which is stored, but not used, with the main program. The ability to easily make up a cassette of a data "run" is an immensely useful part of the process, and it is best done just after the software patch is made and before readout of the stored data is done.

<u>HARDWARE</u>: There are no major hardware modifications to the VIP. A minimum 2 K memory should operate the program, but won't give much data storage. The four counters operate by pulling pins J,H,F, and E to ground. Unless dry contacts are available on the sensors used, opto-isolators (e.g. 4N28) can be used with the transistor portion hanging directly from the input pin of the VIP to ground. Pin L is pulled to ground to enable program to roll. While connections are being made to the input port, it may also be helpful to connect LED status monitors on corresponding output port pins V,U,T, and S. These pins go high, +5v., whenever the program is running and J,H,F, and E are open circuited. They provide a handy monitor during data taking when it is not possible to haul along a video monitor.

Although high accuracy timekeeping may not be necessary for all uses, it has been obtained by replacement of fixed capacitor C4 (33 pf.) in the timebase oscillator circuit with a variable 7 - 47 pf. cap. After replacement, load the program and set the time with WWV. Run for a day or so, and then check the run-out again using WWV. Measure the frequency at Expansion Port pin 1 using any reasonable electronic frequency counter. Simply calculate the percentage time your clock runs out, and move the timebase frequency by a compensating percentage. You will note that this empirical method is not badly hurt by counter inaccuracy in measurement of the exact correct frequency of timebase.

Power interruptions could be a big problem. These are easily avoided by "floating" 5 alkaline cells in series with a diode across the raw B $^{+}$ on its way into the 3 terminal regulator (should be) mounted on the VIP printed circuit card. Self-charging NiCads would really be icing on the cake, I suppose.

OPERATION: I load the 6 page program from a tape that has stored zeros in the remainder of its F pages. Next, it is necessary to load a few specific memory locations with the Hexidecimal values for year (1981 = 51 in hex), month, day, hour, and min. starting at memory location 0603 thru 0607. Listen to WWV, throwing the run switch always starts with the time you have loaded and seconds at 00.

The video display should be a clock with the correct time and date. Four counters displayed below it should be exercised one at a time to see which is which and that they are all working. Your first 'hour' of data taking will be shorter than a full hour--discard it. To access the stored data, change location 020A to hold 1510. Now any keypress on the VIP keypad will advance the display thru memory.

A very nice modification to this program would be to gin out hard copy on a printer--but short of that, you can Xerox up some forms and copy your data off the TV screen fairly well.

SOFTWARE WALK AROUND: Locations 0000 to 01FF hold Rick Simpson's version of CHIP-8I. Locations 020C thru 029A put the clock/calendar display on the screen, and 02A0 thru 02FC moves it around. At 0310 thru 039C and 0400 thru 0476 are the calls to display the counters. Data is examined and counters operated by 03A0 thru 03EB. Hourly storage is 0480 thru 0506. Locations 0510 to 0563 operate data retrieval when called, by software patch. Data storage is from 0610 onward. Additional custom subroutine calls, one per second, minute, or hour can be inserted with a familiar 2MMM instruction at 02B0, 02CC, 02E4 if desired. Be careful that the routines added will still allow the program to loop around in less than a second, as it must to do timekeeping.

APPLICATION: Like other things in this world, this program has yet to be used in its originally intended application. But several studies have been completed at WWRL radio station in two areas:

(1) To look at the long-term modulation stability of the station.
(2) To look at the level setting practices of the D.J. s in terms of overload and dead air.

A Belar AMM-3 Modulation Monitor was used in the first appliation. It has remote status lamp connections which were directly connected to the VIP input port for the tests.

Homemade hardware was used in the second studies. With the meters on the audio console fully pegged offscale, a comparitor was set on the audio output to fire just below full output clip. Dead air was defined at a tone at less than -15 Vu for more than 10 sec., before giving output to the computer.

A modification of this program has been used to successfully generate and decode time and date information on the station's audio logger system. These are 24 hour tape recordings made of both the AM and FM operations. A 2K VIP is permanently dedicated to this task.

COMPUTER LISTING: FOUR INPUT ENERGY LOGGER PROGRAM

0000 to 01FF CHIP-8I:VIPER Sept. 1978, p.4, and subsequent corrections-Nov. 1978, p.11 & Feb. 1979. p.27.

0200	220C	6D00	6E3C	2280	24B0	12AA	6A06	6B00
210	A600	F765	A29C	F433	225A	226A	1220	0000
220	A29C	F533	225A	226A	122C	0000	A29C	F333
230	225A	1236	0000	6A0C	7B06	A29C	F633	225A
240	2274	1248	0000	0000	A29C	F733	225A	00EE
250	A29C	F265	F029	DAB5	7A06	A29C	F265	F129
260	DAB5	7A06	F229	DAB5	00EE	7A06	A296	DAB5
270	7A06	00EE	7A06	A290	DAB5	7A06	00EE	2274
280	6B06	6A24	2274	A29C	FD33	225A	00EE	93AF
290	2000	0000	2000	0000	7000	0000	0000	0176
2A0	A600	F765	FE07	3E00	12A4	603C	F015	23C0
2B0	12B6	12B6	12B6	A600	F765	2280	7D01	3D3C
2C0	12FA	220C	6D00	7701	A600	F755	12D2	12D2
2D0	12D2	A600	F765	373C	12F4	6700	7601	A600
2E0	F755	2480	12E8	12E8	A600	F765	3618	12F4
2F0	6600	7501	A600	F755	220C	2280	12A0	0000
0300	0130	012F	012F	012F	003C	560A	8561	B150
310	2320	2340	2360	2380	00EE	0000	0000	0000
320	A300	F965	A29C	F033	6A00	6B0C	2250	A300
330	F965	A29C	F133	6A12	6B0C	225A	00EE	0000
340	A300	F965	A29C	F233	6A00	6B12	2250	A300
350	F965	A29C	F333	6A12	6B12	225A	00EE	0000
360	A300	F965	A29C	F433	6A24	6B0C	2250	A300
370	F965	A29C	F533	6A36	6B0C	225A	00EE	0000
380	A300	F965	A29C	F633	6A24	6B12	2250	A300
390	F965	A29C	F733	6A36	6B12	225A	00EE	0000
3A0	8BA6	8AB6	8BA6	8AB6	8BA6	8AB6	8BA6	8ABE
3B0	8BAE	8ABE	8BAE	8ABE	8BAE	8ABE	00EE	0000
3C0	B1E1	8AE0	B1A0	13CA	13CA	23A0	4A00	2400
3D0	8BE0	23A2	4A00	2420	8AE0	23A4	4A00	2440
3E0	13E2	8BE0	23A6	4A00	2460	00EE	0000	0000
3F0	0000	0000	0000	0000	0000	0000	0000	0000
0400	2320	A300	F965	7101	3164	1410	6100	7001
410	A300	F955	2320	00EE	----	----	----	----
420	2340	A300	F965	7301	3364	1430	6300	7201
430	A300	F955	2340	00EE	----	----	----	----
440	2360	A300	F965	7501	3564	1450	6500	7401
450	A300	F955	2360	00EE	----	----	----	----
460	2380	A300	F965	7701	3764	1470	6700	7601
470	A300	F955	2380	00EE	----	----	----	----
480	A600	FA65	7A01	A600	FA55	8850	8960	A300
490	F765	A610	FA1E	FA1E	FA1E	FA1E	FA1E	FA1E
4A0	FA1E	FA1E	FA1E	FA1E	F955	4AD8	14AC	2310
4B0	6000	6100	6200	6300	6400	6500	6600	6700
4C0	1500	----	----	----	----	----	----	----
0500	A300	F755	2310	00EE	----	----	----	----
510	2280	A600	FA65	151A	151A	6A00	A600	FA55
520	FF0A	220C	2310	A600	FA65	A610	FA1E	FA1E
530	FA1E	FA1E	FA1E	FA1E	FA1E	FA1E	FA1E	FA1E
540	F965	A300	FA55	A600	F765	8580	8690	A600
550	F755	A600	FA65	7A01	6700	A600	FA55	220C
560	2310	1520	----	----	----	----	----	----

CHIP 8-I COMPUTER LISTING

This interpreter modification provides input and output port operations by these instructions:

B1X0	Value in variable X on Output Port
BOKK	Value of constant KK on Output Port
B1X1	Copy value on Input Port to Variable X and wait for EF4 low.

Normal Chip 8 instruction BMMM is not available.

Full discussion of the Chip 8-I in Sept 1978 VIPER. This listing includes subsequent corrections.

0000	FFBB	FF01	B2B6	F8CF	A2F8	81B1	F846	A190
010	B4F8	1BA4	F801	B5F8	FCA5	D496	B7E2	94BC
020	45AF	F6F6	F6F6	3244	F950	AC8F	FA0F	F9F0
030	A605	F6F6	F6F6	F9F0	A74C	B38C	FC0F	AC0C
040	A3D3	301B	8FFA	0FB3	4530	4022	6912	D400
050	0001	0101	0101	0101	0101	0101	0100	0101
060	007C	7583	8B95	B4B7	BC91	EBA4	D970	9905
070	06FA	07BE	06FA	3FF6	F6F6	2252	07FA	1FFE
080	FEFE	F1AC	9BBC	45FA	0FAD	A7F8	D0A6	93AF
090	8732	F327	4ABD	9EAE	8E32	A49D	F6BD	8F76
0A0	AF2E	3098	9D56	168F	5616	308E	00EC	F8D0
0B0	A693	A78D	32D9	06F2	2D32	BEF8	01A7	46F3
0C0	5C02	FB07	32D2	1C06	F232	CEF8	01A7	06F3
0D0	5C2C	168C	FC08	AC3B	B3F8	FFA6	8756	12D4
0E0	9BBF	F8FF	AF93	5F8F	32DF	2F30	E500	42B5
0F0	42A5	D48D	A787	32AC	2A27	30F5	0000	0000
0100	0000	0000	0045	A398	56D4	F881	BCF8	95AC
110	22DC	1256	D406	B8D4	06A8	D464	0A01	E68A
120	F4AA	3B28	9AFC	01BA	D4F8	81BA	06FA	0FAA
130	0AAA	D4E6	06BF	93BE	F81B	AE2A	1AF8	005A
140	0EF5	3B4B	560A	FC01	5A30	404E	F63B	3C9F
150	562A	2AD4	0022	8652	F8F0	A707	5A87	F317
160	1A3A	5B12	D422	8652	F8F0	A70A	5787	F317
170	1A3A	6B12	D415	8522	7395	5225	45A5	86FA
180	0FB5	D445	E6F3	3A82	1515	D445	E6F3	3A88
190	D445	0730	8C45	0730	84E6	6226	45A3	3688
1A0	D43E	88D4	86FA	013A	ACE5	63D4	E745	FA01
1B0	3AF2	63D4	4556	D445	E6F4	56D4	45FA	0F3A
1C0	C407	56D4	AF22	F8D3	738F	F9F0	52E6	07D2
1D0	56F8	FFA6	F800	7E56	D419	89AE	93BE	99EE
1E0	F456	76E6	F4B9	5645	F256	D445	AA86	FA0F
1F0	BAD4	3FF2	6B3F	F5D4	0000	0000	00E0	004B

Thanks to George Gadbois, W3FEY for providing the memory dump of this program.

*** VIP OWNERS ***

HOW WOULD YOU LIKE TO BE ABLE TO SWITCH YOUR VIP TO RUN, PRESS "1" ON YOUR HEX KEYPAD, AND HAVE YOUR COMPUTER ASK YOU IF YOU WANT A "WARM" OR "COLD" START FOR YOUR VP-701 FLOATING POINT BASIC...JUST LIKE THAT?

ANNOUNCING THE MODEL 14EPM 14K EPROM CARD BY G. J. KRIZEK.

THE 14EPM GIVES YOU THE CONVENIENCE DESCRIBED ABOVE PLUS, AT THE FLIP OF A SWITCH, ALLOWS YOU TO REVERT TO A STANDARD 4K VIP FOR GAMES OR WHATEVER.

OR, THE 14EPM CAN BE USED TO PROVIDE UP TO SEVEN 2K BLOCKS OF EPROM ADDRESSABLE TO ANY 2K INCREMENT OF VIP MEMORY.

THE 14EPM USES POPULAR 2716 (5V) EPROMS.

THE 14EPM WILL BE AVAILABLE AS A BARE BOARD OR ASSEMBLED AND TESTED.

NOGOTIATIONS ARE IN PROGRESS TO ALLOW OUR SALE OF RCA'S VP-701 FLOATING POINT BASIC.

TENTATIVE PRICES:* BARE BOARD WITH DATA.......$ 49.00
A&T WITH BLANK EPROMS......$ 139.00
A&T WITH EPROMS PROGRAMED
WITH DATA PROVIDED BY YOU**$ 145.00
A&T WITH RCA'S VP-701......$ ***.**

*PRICES AND SPECIFICATIONS SUBJECT TO CHANGE.
**DATA TO BE ON VIP COMPATIBLE CASSETTE IN 2K BLOCKS.
***.**WILL BE DETERMINED BY THE OUTCOME OF NEGOTIATIONS.

AVAILABILITY: MAY, 1982

G. J. KRIZEK
722 N. MORADA AVE.
WEST COVINA, CA.,91790

Note: **Paul Piescik's MACHINE CODE column will be back with us in the next VIPER. In the meantime, here is a 'Little Loops" by Tom Swan which deals with machine language.**

LITTLE LOOPS by Tom Swan

Dream Machine

Many of you have expressed an interest in learning 1802 machine language. A lot of you know some and want to learn more. Unfortunately there just isn't enough space here to write a full 1802 primer -- but for those of you who want to "get into machine language," here are some ideas on how to get started.

I will assume that you have written a few programs, probably in CHIP-8, and understand something about binary and hexadecimal numbers sometimes written with a dollar sign in front to denote "hex number", addressing, program flow and jumping (we will call it "branching"), and the general concept of a loop. In addition you should have access to a documented list of 1802 machine instructions and their assembly "mnemonics" which are simply abbreviations of what a particular instruction does. "LDA" is the assembly mnemonic for the machine language instruction "4N." Whenever you see an "N" in a hexadecimal number, it indicates that any of the 16 hexadecimal digits may replace it. That digit refers to one of the 1802's internal registers, numbered from 0 - F and written as R0, R1, R2...RF.

The 1802 has 16 of these registers each of which is 16 bits in length. A register, like a CHIP-8 variable, may be set to equal various values required by your program. The registers are physically inside the 1802 microprocessor -- CHIP-8 variables are not. Each register is really composed of two 8-bit halves and each half may be viewed as an <u>individual</u> register if the program requires such a distinction. When we want to refer to the whole register, we will write "RN." To refer to the higher or leftmost half of RN we will write "RN.1." To refer to the lower or rightmost half of RN we will write "RN.0."

There is another important register in the 1802 called the D register. It is also called the accumulator because results of most all operations appear or are accumulated there. It is only capable of holding an 8-bit binary value and may therefore only contain one half of any register at a time.

Unlike CHIP-8 variables, registers may hold addresses as well as absolute values such as counts and constants. (The numbers are the same -- an address is only a number and may be treated as such -- but the <u>use</u> of that number gives it special meaning.) Being 16 bits long, any register may be set to the address of any location in memory from $0000 to $FFFF. You may think of a register, when it is being used to address memory, as being similar to the CHIP-8 "I" pointer. In fact, register RA <u>is</u> the "I" pointer -- the CHIP-8 interpreter sets register RA and uses that register to address memory for all of the CHIP-8 instructions that use "I".

When values are to be tested and worked upon, they almost always need to be brought into the D register first. Register halves may be transferred to and from D as well as bytes in memory addressed by registers.

Many operations require two eight-bit operands, for example ADD and XOR. One of the numbers will always be placed in D. The other is in memory somewhere and must be addressed by one of the 16 registers. Another four bit register called X is used to select this register which contains the address in memory of the other number or operand to be operated on. (Not the operand itself -- the register is being used to point to the number.)

One of the 1802 registers is always used as the program counter containing a 16-bit address from where the next 1802 instruction is to be found. Like the four bit register X, a four bit register P is used to specify which of the 16 registers is to be the program counter. When P is equal to 3, for example, then the program will run at the address contained in R3 automatically incrementing unless the program branches to another address.

There is another way to locate the second operand for an operation requiring two numbers. This is called "immediate addressing" and the second operand immediately follows the actual machine language instruction that needs it. To accomplish this trick, the program counter register, whichever has been designated by "P", is used to address the immediate operand which will be automatically skipped following the operation to be performed.

When referring to bytes in memory which are being addressed by some register, parenthesis are used to indicate the indirect reference. The letter "M" stands for memory and when we want to refer to the byte, say, addressed by register RC we will write M(RC). Often, for no real reason, the "C" will also be enclosed like this: M(R(C)). This still means "the memory byte addressed by register RC, not the value of RC itself."

Similarly, the notations M(R(X)) and M(R(P)) refer to the registers which are in turn designated by the values in the X and P registers. If X=E, then M(R(X)) means the byte addressed by register RE. This would be equivalent to M(R(E)).

Without knowing anything else, you are ready to begin machine language programming! There is a world of information that I have left out, but too much at one time may be simply confusing to you especially if you are new at the game. Probably the only real way to truly learn a new language is to read it, write it and speak it, and I suggest you study other people's programs as much as you can and try some of your own.

CHIP-8 offers a good way to learn machine language too. You may call a machine language subroutine (MLS) from a CHIP-8 program with the instruction OMMM where the M's are replaced by the starting address of the routine. Check your VIP manual for which of the 1802 registers are available for use in MLS's. Remember to end with a $D4 byte to return control to CHIP-8 when your MLS is done. Try some experimenting -- perhaps just add two numbers together and put the answer in a known memory location (See exercise #1).

Just getting started is usually the hardest part of any new endeavor. Hopefully this bare treatment of the essentials will help. Good luck!

Project #1: Using the "write mode" of the VIP system monitor, set location $0300 to $04, and $0301 to $05. Write a machine language subroutine to be called from a CHIP-8 program which will add the two numbers together and store the answer at $0301 (thus the previous value at $0301 will be destroyed.) Run the program and using the "read mode" see if your MLS works by examining location $0301.

Project #2: Write a machine language subroutine that will verify a 16-byte block of memory addressed by I with the 16 bytes located at I+16, I+17, I+18... I+32. (I am using decimal numbers here.) If both 16-byte blocks are identical, set VF=0, if not then set VF=1. We will use this subroutine in a game to be included in a future VIPER. (Hint -- R6 addresses the CHIP-8 variables which are kept in memory locations. Do not change R6.1! To address variable VF, set R6.0 = FF.)

PIT STOP

Here's a new display setup for the VIP manual DOT DASH game. Just load that game into your VIP, enter the following new page 4 information and you're ready to play.

With the new display, you've got two long straight-aways where you can let the dot get up to full speed, something that is difficult to accomplish in the original version.

Another feature is a secret passage-way to the exit! This is not so obvious and I'm not to reveal the key here. After all, then it wouldn't be a secret. Can you find it?

PIT STOP LISTING

```
0400  FF FF FF FF FF FE 00 00 FF FF FF FF FF FE 00 00
  10  C4 63 00 00 01 82 00 00 EC 6F 00 00 03 C2 00 00
  20  ED FB 00 18 07 E2 00 00 ED 63 00 3C 0B D2 00 00
  30  FF FF 00 7E 0D B3 55 55 FF FF 00 BD 0E 72 AA AA
  40  0C 00 00 DB 0E 73 FF FF 0C 00 C0 E7 0D B3 FF FF
  50  0C 00 60 E7 0B D0 00 0F 0C 7F F0 DB 07 E0 00 0F
  60  0C 00 60 BD 03 C0 00 0F 0C 00 C0 7E 01 80 00 0F
  70  0C 00 00 3C 00 07 FE 0F 0C 00 00 18 00 0F FF 0F
  80  0C 1F FF FF FF FF FF 0F 0C 0F 8A 3E 22 23 FF 0F
  90  0C 0F AB 7E F6 AB 01 0F 0C 0F 8B 7E 36 A3 03 8F
  A0  0C 0F BB 7F B6 AF 01 0F 0C 0F BB 7E 36 2F 03 8F
  B0  0C 1F FF FF FF FF 01 0F 0C 00 00 00 00 00 03 8F
  C0  0C 00 00 00 00 00 01 0F 0C 00 00 00 00 00 03 8F
  D0  0D 55 55 55 55 55 01 00 0E AA AA AA AA AA 03 80
  E0  00 00 00 00 00 00 07 C0 00 00 00 00 00 00 0F E0
  F0  00 00 00 00 00 00 1F F0 AA AA AA AA AA AA BF FF
```

VIPER INDEX

COMPILED BY ANDREW SILLS

TINY BASIC
Boomerang Puzzle: 3.02.18
More Tiny BASIC Machine Language Subroutines, C.D. Smith: 2.08/09.43
TINY Comments, Carmelo Cortez: 2.06.11
VIP Tiny BASIC Machine Language Subroutine, Andrew Modla: 2.08/09.37
TUTORIAL
VIP Flow Charts, Phil Sumner: 3.01.11
VIPHCA Information: Vol. 3, Page 1

****

READER I/O and ADVERTISEMENT

William Lindley
c/o VIPER/VIPHCA

Dear Bill,

I appreciate your modifications of the PIPS programs EDITOR 21 and ASSEMBLER 3 for overlay operation with 4K of text memory. A problem appeared with the "Show Page N" function of the EDITOR 21. The obvious correction was to modify memory location 03DD to OF, the first page of text minus 1. This allowed text pages 1(ML1000) through F (ML1E00) to be accessed by the "Show Page N" function; ESCAPE,3,N. But,the last text page at ML1F00 was not directly accessable with this command. Of course,the ESCAPE,2 command while in page F, or typing past page F, will move into this last page of text. The 4K of memory gives not F pages of text, but 10 HEX (16 decimal) pages.

A better correction to the "Show Page N" function at ML03D3 is listed below. The sixteen text pages are renumbered 0 (ML1000) through F (ML1F00). This full HEX range of page numbers permits a simplification of the subroutine as originaly written by Tom Swan in PIPS. The renumbering of text pages has no other effect on the operation of EDITOR 21 and ASSEMBLER 3 as modified for overlay use. The following modification will not operate correctly with the Modification #2 to EDITOR 21 described in PIPS II, but does work with Modification #1 as outlined.

SHOW PAGE "N" Modification

Location	Data	Comment	
03D3	DC	SEP RC	; Do key scan in ROM for page #
03D4	FC	ADI	; Add 10H to reference correct
03D5	10		memory page (0=10,1=11,2=12,etc.)
03D6	B9	PHI R9	; R9 points to correct page
03D7	D5	SEP R5	; Return
03D8-03DF	00		; Not used

FOR SALE VP560 EPROM board $20, VP570 Memory Expansion $60, VP700 TINY BASIC board $22. Prices postpaid USA.

Brian H. Hudson
33½ Cerice Circle
Marietta, Ga. 30060

ONE PAGE MORSE CODE OUTPUT ROUTINES

By David Barber, WD8AJQ, Route 7, Defiance, Ohio 43512

It has been about 6 to 8 months since I started computer programing using the 1802 processor. As a blind computer hobbyist, being an amateur radio General Class operator gave me all the knowledge I needed to obtain a readout from my RCA VIP microcomputer.

My friend Herm Sasson, who lives in Defiance, Ohio, and taught me all my programing skills, wrote a monitor program for the VIP which outputs in Morse Code. The three programs I have written for this article use a programing concept that Herm has not yet tried in a Morse Code output routine.

These three programs all use the same data tables and operate in only one page of memory. The programs are:

1. Morse Code Keyboard program
2. Random Morse Code Generator program
3. Output Morse Code from (ASCII) buffer

I am providing a hex listing of these programs and all necessary comments so that you can run them right away. The output is via the onboard tone generator on the VIP. If you are using a computer that does not have this feature you will have to set it up to use these three programs. Simply tie an oscillator to the output of the Q line and you are ready to run these programs.

The programs each consist of three basic parts as follows:

1. The main program
2. The small data table (0061-7F) This table contains the one byte Morse Code character to be output.
3. The large data table (0080-FF) This table is used to access the small table and also contains the length of the character to be sent.

The programs function on the ability to get data out of the small table. This is then shifted left into the data flag, which is used to determine if a dot or dash is to be sent.

The following characters are available:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
1 2 3 4 5 6 7 8 9 0 .(period) ,(comma) /(slash) ?(question mark)
Vertical break End of Transmission End Transmission Block
(These last two are standard ASCII control characters)

Here is a list of the registers as used in all three programs:

Register(s)	Use
F	Input— Work space at 0060
E	Table access (both tables)
D,C	Morse Code loops
A	Output buffer (program 3 only)
7	Controls character length
3,2,1	Random number generation
0	Program counter

All empty spaces in the programs must be set to 00 or they will not function correctly. The following simple program will clear all RAM in memory so you will not need to enter all those 00's yourself. Run it before entering these programs.

```
0000    E1 90 73 30 00
```

1. Morse Code Keyboard program (All programs function with the tables listed seperately at the end of this article)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	F8	00	BF	BE	F8	60	AF	3F	07	37	09	EF	6B	30	1B	
0010												0F	FF	20	32	57
0020	0F	F9	80	AE	0E	32	00	FA	07	A7	0E	F6	F6	F6	F9	60
0030	AE	0E	38	0F	FE	5F	33	3D	F8	03	AD	30	40	F8	01	AD
0040	7B	F8	0C	BC	2C	9C	3A	44	39	51	2D	8D	3A	41	7A	30
0050	41	87	32	57	27	30	33	F8	28	BD	2D	9D	3A	5A	30	00

Speed is controlled by the byte at 0042
Input occurs at 000C
EF4 is tested at 0007 & 0009
Space between characters is determined by location 0058

2. Random Morse Code Generator program

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	23	36	05	30	00	90	B1	BE	B2	F8	7B	A1	F8	7E	A2	83
0010	52	E2	01	21	F4	12	F4	51	F8	7E	A1	01	21	73	01	21
0020	73	01	21	73	01	21	73	01	A3	F9	80	AE	0E	32	09	FA
0030	07	A7	0E	F6	F6	F6	F9	60	AE	0E	38	8F	FE	AF	7B	33
0040	45	F8	24	30	47	F8	0B	BD	2D	9D	3A	48	39	51	7A	30
0050	45	87	32	57	27	30	3B	F8	88	BD	2D	9D	3A	5A	30	09

Press key C (hex keypad) to start program.
EF3 is tested at location 0001 to determine keypress.
The dot length is at 0046.
The dash length is at 0042.
The space between characters is at 0058.

3. Output Morse Code from (ASCII) buffer

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	90	BF	BE	AA	F8	60	AF	F8	01	BA	4A	5F	8A	3A	1B	30
0010	0F											0F	FF	20	32	57
0020	0F	F9	80	AE	0E	32	0A	FA	07	A7	0E	F6	F6	F6	F9	60
0030	AE	0E	38	0F	FE	5F	33	3D	F8	03	AD	30	40	F8	01	AD
0040	7B	F8	0C	BC	2C	9C	3A	44	39	51	2D	8D	3A	41	7A	30
0050	41	87	32	57	27	30	33	F8	48	BD	2D	9D	3A	5A	30	0A

Speed is determined by location 0042.
Morse code output buffer starts at 0100. It is 256 bytes long.
The program stops when it reaches the end of the buffer.
The space between characters is determined by 0058.

Small Data Table (required by all three programs)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0060	ws	80	C0	E0	F0	FF	78	38	18	08	B8	50	68	D0	90	20
0070	40	AA	CC	30	E8	70	00	E8								

Large data table (also required by all programs)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0080	54															
0090								8C	2F							
00A0													9D		8D	64
00B0	B4	0C	14	1C	24	2C	34	3C	44	4C						95
00C0		09	33	5B	32	28	6B	3A	2B	29	0B	5A	53	B1	31	B2
00D0	73	7B	52	2A	B0	12	1B	0A	63	83	3B		64		AC	
00E0		09	33	5B	32	28	6B	3A	2B	29	0B	5A	53	B1	31	B2
00F0	73	7B	52	2A	B0	12	1B	0A	63	83	3B		AC			

COMMENTS

The value that determines the speed (@ 0042) in programs 1 and 3 should be no less than 03 nor greater than 0F. The first is probably too fast, while the last is too slow. The same applies to the dot length in program 2. the dash length should be 3 times the dot length.

The random number generator used in the second program is based on that presented in the KIM Manual and was converted by Herm Sasson. His address is:

Herm Sasson
618 N. Clinton St.
Defiance, Ohio 43512

The Morse Code characters used in these programs are those in common usage on the amateur radio bands. This program could be modified with a larger character table, a video display, and facilities to type into a buffer. A program similar to that described has been written in CHIP-8 by George Gadbois, W3FEY, and occupies 4K.

Finally, my experience with an expanded VP111 has been through material passed over the phone and hands-on experience at home. What I know about machine language programming of the 1802 indicates that it has more capabilities than just games and automation-type programs.

Have fun with these programs and let me know of any new programing techniques that come along.

Editorial

This issue of VIPER marks the end of the 1981 membership year for VIPHCA. For the most part, it has all gone fairly smoothly, and the VIPER has been well received. I'd like to thank the many people who have helped in the effort to keep VIPER going, especially those of you who have submitted articles and programs. There are still articles and programs waiting to be published, and these will be printed in Volume 4. But, as usual, I invite you to submit any material which you might like to share with other VIPHCA members.

The future of the VIP, however, is a bit uncertain, since RCA has, for all practical purposes, discontinued manufacturing the VIP and its accessories. There is still a pretty good inventory of VP-111 and VP-711 units and some of the plug-in accessories. They are now available at close-out prices and you should call Judy Warfel at RCA's VIP number (800-233-0094) for exact prices and info. Naturally, this is a disappointment for VIP users, since this means that the number of VIP hobbyists will be limited. And VIP hardware may no longer be available from RCA. This may mean that any future 1802 hobbyists will be using one of the ELF machines or a home-built design. Of course, RCA will probably still have the Micro-Board series of development units available, but these units are not priced for the typical hobbyist.

While lamenting the RCA decision to down-play the VIP line and emphasize the VP-3300 series of terminals, I can understand that the VIP line could not have been a great profit center for RCA. And during business recessions, unprofitable or low profit enterprises are not carried along as they might be in betters times. Also, there is competition from other companies offering newer and perhaps "improved" hobby computers. The $99 Sinclair ZX81 kit comes to mind, for example. A hobbyist buying a "first" machine on a limited budget would no doubt be attracted to the ZX81. Personally, I feel that with the VIP a person can learn <u>more</u> about how a computer works, how machine language operates, how computers connect to the outside world, and so on. And now there are computers by Radio Shack, Commodore, Atari, and so on, all aimed at the first-time computer buyer. With all the features these machines have, it sure doesn't make it easy to popularize the VIP!

Nevertheless, the VIP is an enormously appealing machine and truly amazing, considering its scale. Many people (including me) marvel at what can be done with "only" 4K of memory in the VIP. We have a lorge number of ham radio operators in our membership--much more that mere probability would allow--and I think that is because the VIP appeals to the electronic "hardware" tinkerer, the same type of person who might also be a ham. But VIPHCA will continue to exist as long as it is supported by its members.

Which brings us to the subject of 1982 membership. The VIPHCA treasury has a modest surplus, primarily because I was able to find a printer in busy Manhattan who is able to print the VIPER for about half the cost quoted by printers in my area. Fortunately, I work in Manhattan and pass by the printer's store every day. But unfortunately, that has meant that I have had to lug home on the bus about 80 pounds of paper in 2 large boxes each time an issue is printed! But it saved us a lot of money, and for that reason we will NOT have to increase dues, even in this day of ever-increasing prices. Dues for 1982 will be $12. The back of this VIPER contains a renewal from and a questionaire. I would like to get "feedback" from VIPHCA members in order to plan for the 1982 membership year. You can send in the back page, but all I really need is the information. As usual, make your check payable to VIP Hobby Computer Association, U.S. funds only. All checks must be drawn on a U.S. Bank or correspondent.

Yours,

Ray

Raymond C. Sills
Director, VIPHCA

VIP Hobby Computer Assn.
32 Ainsworth Avenue
East Brunswick, NJ 08816

FIRST CLASS

Membership Questionaire & Renewal Request

1. What type of 1802 system do you own?
   VIP ELF II Super ELF Studio II Homebuilt✓ Other ________

2. Is it O.K. to reveal your address to other VIPers, as in a membership list?
   Yes No

3. Do you have a favorite VIPER article/program? ________________

4. Any article/program you did not like? ________________

5. Would you to spend some of VIPHCA's money to give authors/programmers of published VIPER material a modest ($5 or so) honorium to cover the expenses of submitting the material?
   Yes No Don't care.

6. If you said yes, should we make it retroactive for Volume 3?
   Yes No Don't care.

7. Is this print format OK as far as you are concerned?
   Yes No Don't care.

8. If not, what style would you prefer? ________________
   (e.g. loose-leaf, even though it would cost more)

9. Should VIPER use large envelopes for mailing (and fewer pages, due to postal weight limitations)?
   Yes No Don't care.

10. Would you prefer a VIPER with fewer pages, but more frequent publication?
    Yes No Don't care.

Please indicate your preferences, and make sure your name and address are correct. You may remove this page and return it, or photocopy it or merely send in your answers on a plain sheet of paper, whatever is most convenient for you.