VIPER

July - August 1983

Volume 5, Number 2

Journal of the VIP Hobby Computer Assn.

The VIPER was founded by ARESCO, Inc. in June 1978

****

PIPS for VIPS IV - Part 2

This issue of VIPER continues the serialization of Tom Swan's PIPS for VIPS IV. There are still many more parts to come, since the original manuscript ran over 180 pages. And I've taken the liberty of changing the order of the material. Most of it is in complete "sections," although there are occasional references to parts which have not yet been published. Changing the order of the material was done to try to get things arranged in fairly equal size "chunks" and to send them out at the appropriate time. (One of the programs is called "A Binary Tree for the Holidays!)

Cassettes of PIPS IV will be available and the cost for the ten programs on the tape will be $5. You may send your check to VIPHCA at the usual address. Each program on the tape is complete, with the appropriate CHIP-8 interpreter, where necessary.

News & Views

Those of you who are considering getting a memory expansion board for your VIP from one of VIPER's advertisers might do well to place your order fairly soon. George Gadbois, a VIPHCA member and one of our advertisers, told me that prices on the 6116 chips have been going UP lately and the chips are in short supply. Not too surprising, considering the demand for such a useful memory chip. Many computers and memory boards are using the 6116, as a casual glance in most any computer magazine will tell you.

George also mentioned that RCA still has some VIP computers in stock, so you might also want to consider getting one, for spare parts, if nothing else. Or maybe for use as a dedicated controller for something around the house. If you use a computer to manage your heating system, there is a pretty good chance you could take a tax deduction or even a credit, assuming the system will save evergy when in use.

The price war in home computers has really heated up, and I suspect that within a year or so we may have some casulties in the form of companies which get out of the business altogether or even go belly-up. Right now, Commodore, with its 64K machine selling at less than $200 (I paid that much for a 2K VIP Kit five years ago) looks like it may really capture the home market. I have a VIC-20 here, and it is a nice machine. One smart move by Commodore was to make most of their accessories (printers, disk drives, etc.) fully compatible with each of their computers. If you get a disk drive for the VIC it will work with the '64, so you won't have to replace everything when you upgrade the system. Too bad the Commodore machines use a 6502 processor, rather than something with more power, like the 1802! And the price cutting many not be finished. Word is that there's room to cut prices another 25 to 50 per cent. Amazing.

By the way, I'd like to call your attention to an article by VIPHCA member Jim Brousseau in the July issue of Microcomputing Magazine (page 88) which describes how to build a simple cassette interface for the VIC-20. Well done, Jim, and it's nice to see your article in a large circulation magazine!

73,

Ray

ATTACK OF THE MICROMEN

A column of menacing MICROMEN advances towards you. Your only defense is an unlimited supply of cream puffs which you may throw at the descending devils. MICROMEN, you see, can only be stopped by direct hits of gooey sweet cream-filled pies, in other words, "just desserts" will work. When you hit a MICROMAN, (I wonder if I should have made them "MICROPERSONS"?) it will retreat but continue to march on from that point. If you manage to "cream" 25 MICROMEN before any of them reach your side, you win. Otherwise, well... I hate to tell you, but... the consequences are... oh, I can't say it... it isn't very pretty. Good luck. You'll need it.

Keys 4 and 6 move your pie thrower back and forth. Key F fires a cream puff straight up. (Be sure to hold Key F down as it may not respond to only quick taps.) Key 0 restarts new games without having to reset the run switch.

PROGRAM DESCRIPTION

The advantage of animating computer displays with page switching will be more apparent with a comparison. To deactivate the page switching, make the following changes.

```
00E9  00
0201  FC 03 -- Deactivate page switching.  Reload
0218  00    --   interpreter from tape to reactivate
```

The final display messages won't be shown properly but this will show you the normal CHIP-8 way of animating things. Page switching, though you don't see it happening, makes a game like this possible. You can improve most arcade game graphics with the technique while retaining all the other game instruction features of a CHIP-8 program.

Reload the program after making the above temporary changes. This will reset the page switching routines to their previous values.

The tape supplied with this book matches the listing printed here for ATTACK OF THE MICROMEN. All variables are set for a beginner level of play which may not be very challenging especially if you are experienced with handling the VIP hex pad. (And an expert cream puffer, no doubt!)

You may make changes to the following CHIP-8 instructions for more or less difficult versions of the program. The instructions that may be modified are listed here in the form "HHNN" where "HH" is a hex digit <u>not</u> to be changed and "NN" is a value which may be set by you to a new value. For a better understanding of the program, look up these same locations in the listing where you will find additional comments on the purpose of these commands.

0512 60NN -- Starting VY coordinate of MICROMEN. NN may be any value 00 to $68 with greater values

placing the robots closer to the bottom at the start of a game.

0578 70NN -- Amount that MICROMEN advance downward
057C 70NN -- when changing horizontal direction. Should be kept reasonable probably in the range $01 to $10. Both instructions are usually set to the same value for NN, but they do not have to be equal.

058A 70NN -- Set NN to any value in the range $F0 to $FF. This is the amount that a robot will jump back when hit. A value of $FF causes a small jump back (1 bit position) and $F0 causes the greatest jump (16 bit positions) back. Values less than $F0 may seem to work but could cause the program to crash.

059E 45NN -- Set NN to the winning score. Remember you are working in hexadecimal but the score will be displayed in decimal digits. So for a winning score of 100, for example, set NN to the hex value $64. Any value may be programmed in place of NN.

Here are my combinations for intermediate and advanced levels of play. These are only suggestions and you may adjust the program using the above information to any level of difficulty

you want. Make these changes using the ROM "write mode" of your operating system, then save eight pages on a fresh tape. Or simply make the changes before playing new games.

When saving new versions, it's best to use a fresh cassette. Do not overwrite the original programs supplied with this book. If something were to happen during writing to the tape, for instance if the power goes off, you may lose your version and have to hand load the entire listing. This is a good rule to remember for all your purchased tapes, and you may want to break off the record enable tabs on the back of each cassette to be sure you don't accidentally erase a program.

INTERMEDIATE PLAY LEVEL

0578 - 7004 - Move down by 4
057C - 7004 - " " "
*058A - 70F4 - Move back by 12
059E - 4528 - 40 hits to win

ADVANCED PLAY LEVEL

0578 - 7004 - Move down by 4
057C - 7004 - " " "
058A - 70FC - Move back by 4
059E - 4532 - 50 hits to win

*Note, 70F8 seems good here too.

ADDING SOUND EFFECTS

If you have an RCA Simple Sound Board, you may make the following additions to a copy of ATTACK OF THE MICROMEN to add sound effects to the game. The special CHIP-8 interpreter

supplied on the tape contains the ability to output tone values to the Simple Sound Board. You must hand load the following modifications to an original copy of ATTACK OF THE MICROMEN using the VIP system monitor "write mode." These modifications are not provided on the taped version of the game.

Two "patches" are added to give sound to the program. A patch is a section of a program which is added anywhere in available memory space. Usually it is a sequence which is to be executed in the middle somewhere of other code but which would require rewriting the entire program just to fit the patch in. Instead, a jump is executed to the patch in place of some instruction in the main code. Then a jump back to the program is performed following the patch after also having executed the original instruction that was replaced by the jump to patch command.

Programs with many patches should probably be rewritten especially if future modifications will be made. Patching should be used only as a limited means for either debugging a program design or to add a capability which the programmer is sure won't interfere with the program's execution. Use of an assembler program such as the CHIP-8 Assembler in PIPS FOR VIPS II allows patches to be directly inserted into the main code.

MARCHING SOUND EFFECT

```
0540            1650   PTCH1 -- Jump to sound effects patch #1
--
0650  PTCH1:  74FC   ;V4-4   -- Subtract 4 from V4 sound variable
  52          6FF0   ;VF=F0  -- Set utility variable to $F0 value
  54          84F1   ;OR     -- OR with V4 to cycle in range $F0-$FF
  56          B4A7   ;OUT    -- Output tone value to port
  58          6F01   ;VF=1   -- Set utility variable to 01 value
  5A          FF18   ;TONE   -- Sound tone for length in variable VF
  5C          6004   ;V0=4   -- Patched instruction from main code
  5E          1542   ;RET    -- Jump back from patch
```

HIT SOUND EFFECT

```
059C            1660   PTCH2
--
0660  PTCH2:  6E10   ;VE=10  -- Set utility variable VE to pitch
  62          BEA7   ;OUT    -- Output VE value to sound board
  64          6F02   ;VF=2   -- Set utility variable VF to length
  66          FF18   ;TONE   -- Sound tone for length in VF
  68          159E   ;RET    -- Jump back from patch
```

Note that variables V4 and VE are used and changed by these patches. Also, there is no patched instruction to execute in the HIT SOUND EFFECT. This is because the jump instruction at $059C replaces the original hit tone instruction which is no longer needed.

The MARCHING SOUND EFFECT may look a bit complex. I wanted a low tone to cycle around four values so that it would not sound musical, but rather approach the effect of robot boots stomping through the field -- a field sloppy with the remains of spent cream puff ammunition. Well, maybe it doesn't sound quite like that. You'll have to use your imagination a little.

All the first routine does is to subtract four from the value in V4 at location $0650. Logically ORing this value with hex $F0 keeps the resulting value within range. Anything ORed with $F0 will be at least at large as $F0. For this reason there is no need to initialize V4 at the beginning of the program though by not doing so, I am breaking one of my own rules about variable handling.

You may increase the tone length at line $0658 by setting VF to a higher value. This will produce a more musical sound though I like the effect of a shorter tone burst.

The HIT SOUND EFFECT simply beeps. A more elaborate routine is certainly possible and there is plenty of programming space available. I didn't want to slow up the game for this, however, so a simple beep was the choice. I don't know about you, but to me this frequency sounds like metal striking metal. I suppose that would be the pie pan.

POINTS OF INTEREST

Only one X coordinate needs to be kept to remember the horizontal location of all six MICROMEN. No matter where they are positioned, they all move as a block so the X coordinates for robots #2-6 may be easily found by adding eight to the X coordinate for robot #1. (See $05B0) This value is kept in variable VB which has no other purpose in the program.

Variable VA is always set to either 1 or $FF. This value

is added to the X coordinate in VB at line $056A before displaying the robot row. Adding 1 moves the row 1 bit position to the right while adding $FF moves the X coordinate one position to the left. This is because adding $FF is the same as subtracting 1 from any value. When the X coordinate reaches the limits 00 or $10, VA is changed to its opposite value and the row will go the other way.

Keeping track of the vertical positions of each robot is not quite so simple but is "mucho mas interesante." (much more interesting." Might as well throw in a free Spanish lesson once in a while for a change of pace.)

Each Y coordinate could have been kept in a separate CHIP-8 variable. But that would have tied up six variables and there weren't that many available anyway. Instead, each Y coordinate is kept in an array in memory at locations $07FA to $07FF. An "array" is just another way of saying "list" or "collection," but the word array is more appropriate especially when the position of an item in the list tells us something about that item. In this case, each position in the array corresponds to one of the robots. To find the Y coordinate for the third robot, a variable is set to the value stored at the third position in the array. The fourth Y coordinate is at the fourth position, etc.

If we give the array the single letter name "P", then we can express the values stored at locations in the array with

the expression "P(n)" where "n" is the number of the position in the array. A common convention is to let P(0) be the first item of the list, P(1) the second, P(2) the third, etc. Therefore P(5) would be the last or sixth value in the array P.

We already know the address of the start of the array. This is usually called "the base address" and in the ATTACK OF THE MICROMEN game the base address is $07FA. To find any value in the array, the program only needs to add the value of "n" to the array base address with the instruction sequence AMMM to set "I", then FX1E to add VX to "I". An FX65 instruction will then set VX equal to the value of P(n).

When a program needs to keep track of a lot of variable information, an array such as this may be the best storage device. Actually all CHIP-8 variables V0 to VF are also kept in a similar array managed by the CHIP-8 interpreter. When you examine CHIP-8's variables during program debugging, you are actually looking at the byte values in the interpreter's variables array. For the versions of CHIP-8 supplied with this book, the variables array is at $02F0 to $02FF. Other versions of CHIP-8 locate the array elsewhere but the concept is identical.

The loop at $0570 to $05B6 takes bytes from the Y coordinate array, uses those values to position each MICROMAN on the display, then replaces the possibly changed value into the array. V3 is used to find new positions in the array. V3 is the equivalent of "n" in the above explanation, and is first set to

zero at $056C then incremented at $05B2 for each pass through the loop to access each of the six Y coordinates.

Hundreds of MICROMEN could be kept track of in this way and only when V3 equals six does the loop terminate at $05B4 with the SKIP instruction located there. Of course there isn't room on the display for that many MICROMEN and thank goodness for that!

One other thing about using arrays. They should usually be initialized, most often to all zero bytes. In this program the initialization to zeros will cause the MICROMEN to all start out at the top of the display (VY=00). If the bytes you will be inserting into array positions won't ever be zeros during the run of your program, initializing an array is also a good way to find empty positions.

The array in ATTACK OF THE MICROMEN is initialized to all zeros with the instructions at $0512 to $051E. The instruction at $0518 does two things. 1) The value of V0, in this case 00, is stored in the array position addressed by "I"; 2) "I" is automatically incremented by one. By looping six times, all six bytes from $07FA to $07FF are set to zero.

USING THE PAGE SWITCHER

Notice the instruction pair at $053C marked by the label "LOOP." First the display is flopped to view the opposite display page, then an 0200 ERASE instruction is executed to

clear the off screen page preparing it to accept new display information. The order of these two instructions is very important. If you reverse them, you would be forever viewing a blank display.

Throughout this discussion and in the following program listing, the DXYN instructions are commented "Show MICROMEN" or "Print text" etc. Remember that all the action is taking place behind the curtain. Only when the next 0216 CALL FLOP instruction is executed will the work of the DXYN's be seen.

Because two display pages are being used, the usual way of animating a CHIP-8 display by redisplaying a figure on top of itself is complicated. You may still do this but will need to keep careful track of all X,Y coordinates. It is far easier to simply clear the entire display, then reshow all display figures in their new positions before flopping to the other page.

I hope you have an opportunity to use page switching for your next program. We'd love to see the results of your efforts at the VIPER and invite you to send them in. Good luck! May all your cream puffs find their targets!

ATTACK OF THE MICROMEN

VARIABLE ASSIGNMENT

V0 - Various uses
V1 - " "
V2 - " "
V3 - " "
V4 - Not used
V5 - Score (# of hits)
V6 - VX for cream puffs
V7 - VY " " " (when = 00, then no puffs thrown)
V8 - VX for player's pie thrower (called shooter in listing)
V9 - VY " " " " (" " " ")
VA - VX adder for MICROMEN, either 01 or FF, do not change
VB - VX for MICROMEN display (VY's in array @ 07FA-07FF)
VC - General purpose VX and other uses
VD - " " VY " " "
VE - Not used
VF - Hit indicator, some utility uses

MEMORY MAP

0000 - 029F -- CHIP-8 hi-res interpreter with MESSAGER & page switching
02A0 - 02EF -- CHIP-8 work space and stack
02F0 - 02FF -- CHIP-8 variables
0300 - 04FF -- ASCII character set bit patterns
0500 - 0649 -- MICROMEN program
064A - 06FF -- Available space
0700 - 07E7 -- ASCII encoded messages
07E8 - 07F9 -- Available space
07FA - 07FF -- 6 byte VY MICROMEN array
0800 - 0BFF -- Display page #1
0C00 - 0FFF -- Display page #2

SYMBOL TABLE

BEGIN - 0500	PUFF - 0638	TRS2 - 059E
C-3DD - 0624	SHOM1 - 0570	TRS3 - 05D0
LOOP - 053C	SHOOT - 0632	TRS4 - 05D8
LOSE - 05CA	TIME - 062A	TRS5 - 05EA
MICRO - 063A	TIMER - 0628	TRS6 - 0604
NUMB3 - 060A	TRS1 - 0518	WIN - 05CE

5.02.13

PROGRAM LISTING

```
        ; **************
        ; * ATTACK OF  *
        ; *    THE     *
        ; *  MICROMEN  *
        ; **************

        ; C.1980 T.SWAN

        ;INITIALIZE

0500  BEGIN: 0200  ;CLEAR -- Erase one off screen page (1K)
  02         0216  ;FLOP  -- Flip to view that cleared page
  04         0200  ;CLEAR -- Erase the other page (1K)

  06         6500  ;V5=0  -- Initialize score to 00
  08         6700  ;V7=0  --      "     cream puffs VY coordinate
  0A         6810  ;V8-10 --      "     pie thrower VX coordinate
  0C         697A  ;V9=7A --      "      "     "    VY     "
  0E         6A01  ;VA=1  --      "     direction adder for MICROMEN
0510         6B00  ;VB=0  --      "     VX coordinate - MICROMEN

        ;RESET VY ARRAY

  12         6000  ;V0=0  -- Set a utility variable to 00
  14         6106  ;V1=6  -- Set V1 to loop count of 6
  16         A7FA  ;ARRAY -- Point "I" to MICROMEN VY array
  18  TRS1:  F055  ;PUT   -- Store a zero byte at "I" (I+1)
  1A         71FF  ;V1-1  -- Decrement the loop counter
  1C         3100  ;SK=0  -- When loop count is zero, skip
  1E         1518   TRS1  -- Else jump to reset next array byte

        ;DO TITLE

0520         A700   ASCI1 -- Point "I" to ASCII title codes
  22         6C00  ;VC=0  -- Set VX, VY in VC, VD for displaying
  24         6D00  ;VD=0  --   first text line
  26         0244  ;MSGR  -- Call MESSAGER.  Print text @ VC,VD
  28         DCD8  ;SHOW  -- DXYN for use by MESSAGER program
  2A         6D10  ;VD=10 -- Move VY coordinate down
  2C         0244  ;MSGR  -- Print next line of text
  2E         DCD8  ;SHOW  --    "          "       "
0530         6D78  ;VD=78 -- Move VY coordinate down
  32         0244  ;MSGR  -- Print "by-line" of text
  34         DCD8  ;SHOW  --    "          "       "
  36         0216  ;FLOP  -- Call MLS to show text (switch pages)
  38         60FF  ;V0=FF -- Set V0 to timer value
  3A         2628   TIMER -- Do sub to delay start of game

        ;END INITIALIZE
```

```
      ;MOVE SHOOTER

  3C  LOOP:  0216  ;FLOP   -- Switch view to "static" page
  3E         0200  ;CLEAR  -- Erase off screen page
0540         6004  ;V0=4   -- Use V0 to test for key 4 and 6
  42         E0A1  ;SK≠K4  -- If not Key 4 (≠V0) skip next
  44         78FE  ;V8-2   -- If Key 4 pressed, reduce shooter VX
  46         6006  ;V0=6   -- Use V0 to test for Key 6
  48         E0A1  ;SK≠K6  -- If not Key 6 (≠V0) skip next
  4A         7802  ;V8+2   -- If Key 6 pressed, increase shooter VX

      ;TEST BOUNDARIES

  4C         48FE  ;SK≠FE  -- If shooter VX is not in the
  4E         6800  ;V8=0   --   range of 00 to 38, readjust to
0550         483A  ;SK≠3A  --   keep from wrapping around sides
  52         6838  ;V8=38  --

      ;SHOW SHOOTER

  54         A632   SHOOT  -- Set "I" pointer to shooter bit patterns
  56         D896  ;SHOW   -- Display shooter at V8, V9

      ;MOVE CREAM PUFF

  58         4700  ;SK≠0   -- If puff VY is not zero, then a puff has
  5A         1562   LINK   --   been shot and it must be readjusted
                            --   upwards.  Else, jump to 0562
  5C         77F8  ;V7-8   -- Reduce cream puff VY by 8 bits
  5E         A638   PUFF   -- Set "I" to bit pattern for cream puff
0560         D672  ;SHOW   -- Display at V6,V7 (Note - there are no
                            --   MICROMEN there yet so hits are looked
      ;SHOW MICROMEN        --   for later.)

  62 *LINK:  4B00  ;SK≠0   -- When MICROMEN VX variable VB is
  64         6A01  ;VA=1   --   either 00 or 10, then VA is reset
  66         4B01  ;SK≠10  --   to make them go the other
  68         6AFF  ;VA=FF  --   direction back and forth

  6A         8BA4  ;VB+VA  -- Add direction in VA to VX coordinate in VB
  6C         6300  ;V3=0   -- Set index variable V3 to 00
  6E         81B0  ;V1=VB  -- Move MICROMEN VX from VB to V1
                            --   (V1 will change and this saves VB value)
0570  SHOM1: A7FA  ;ARRAY  -- Set "I" to MICROMEN VY storage array
  72         F31E  ;I+V3   -- Add value of V3 to "I"
  74         F065  ;GET    -- Let V0 = the byte addressed by "I"
  76         4B00  ;SK≠0   -- When VB is either 00 or 10, then
  78         7004  ;V0+4   --   MICROMEN will move down
  7A         4B10  ;SK≠10  --   by 04.  Change increment in
  7C         7004  ;V0+4   --   locations 0578 & 0576 for a greater
                            --   advance and a more difficult game
```

*NOTE: Label "LINK" shown for reference only -- see PIPS FOR VIPS II -- CHIP-8 Assembler

```
      ;SHOW A MICROMAN

  7E          A63A   MICRO -- Set "I" to MICROMAN bit patterns
0580          D10F   ;SHOW  -- Display one MICROMAN at V1, V0
  82          3F01   ;HIT!  -- If hit a cream puff, skip next
  84          159E    TRS2  -- Else jump to continue -- no hit

      ;SCORE HIT

  86          7501   ;V5+1  -- Add 1 to score each hit
  88          D10F   ;ERASE -- Erase MICROMEN by redisplaying
  8A          70F8   ;V0-8  -- Reduce VY to move back (Try F0-FF)
  8C          8F0E   ;SHL   -- Test MSB of V0 by shifting into VF
  8E          4F01   ;SK +  -- If VF=1, then past top of screen
0590          6000   ;V0=0  --   so reset VX in V0 to 00

  92          D10F   ;SHOW  -- Redisplay knocked back MICROMAN
  94          A638    PUFF  -- Set "I" to cream puff bit pattern
  96          D672   ;ERASE -- Display to erase the puff that hit
  98          6700   ;V7=0  -- Reset puff VY to allow next firing
  9A          6F02   ;VF=2  -- Set VF to tone value
  9C          FF18   ;TONE  -- Sound tone to signal a hit

  9E  TRS2:   4519   ;SK≠19 -- If score not 25 decimal, skip
05A0          15CE    WIN   --   Else go to win sequence
  A2          6F68   ;VF=68 -- Set utility variable to 68 hex
  A4          8F05   ;VF-V0 -- Subtract MICROMAN VY coordinates in V0
  A6          4F00   ;SK<68 -- Skip next if positive
  A8          15CA    LOSE  --   Go to lose sequence when a
                      --     MICROMAN goes below VY=68

      ;RESET ARRAY VY

  AA          A7FA   ;ARRAY -- Set "I" pointer to same VY array
  AC          F31E   ;I+V3  -- Again, add V3 to index into array
  AE          F055   ;PUT   -- Replace possibly adjusted MICROMAN
                      --     VY for next time
05B0          7108   ;V1+8  -- Add 8 to VX for next MICROMAN
  B2          7301   ;V3+1  -- Also add 1 to array index for next VY
  B4          3306   ;SK=6  -- When index is 6, then skip
  B6          1570    SHOM1 --   Else loop back to display all
                      --     six MICROMEN

      ;CHECK FIRE
      ;BUTTON

  B8          3700   ;SK=0  -- If cream puff VY=0, then skip to test
                      --     for fire
  BA          153C    LOOP  -- Else jump back.  Only one pie at a time!
  BC          60DF   ;V0=F  -- Let V0=0F to test if Key F pressed
  BE          E09E   ;SK=KF -- If Key F is pressed, skip next to fire
05C0          153C    LOOP  -- Else jump back.  Key F not pressed
```

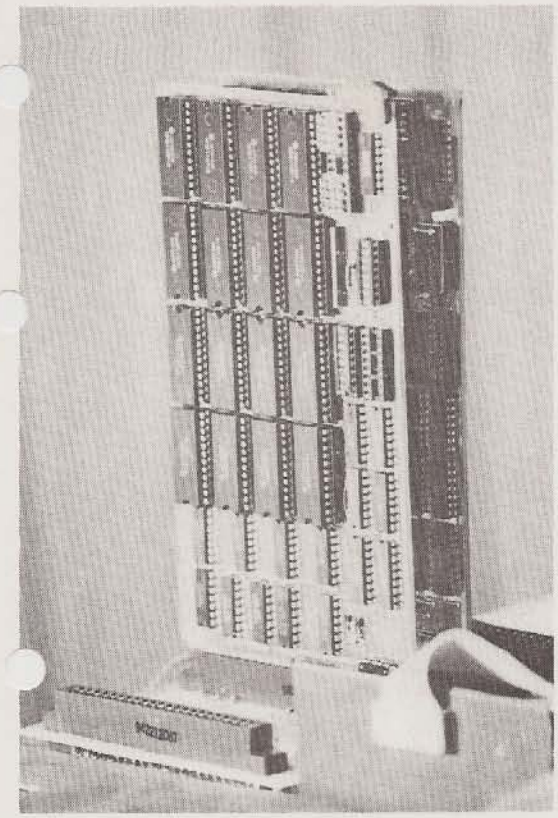
```
      ;FIRE CREAM
      ;PUFF!

05C2          8680  ;V6=V8 -- Let cream puff VX equal shooter VX
  C4          7603  ;V6+3  -- But add 3 to find tip of shooter
  C6          6778  ;V7=78 -- Set cream puff VY to hex 78
  C8          153C   LOOP  -- Jump back to do it all again

      ;END MAIN LOOP

      ;LOSE JUMPS HERE

  CA  LOSE:   A72B  ;ASCI2 -- On loss, set "I" to appropriate message
  CC          15D0   TRS3  -- Jump to start message display

      ;WIN JUMPS HERE

  CE  WIN:    A763  ;ASCI3 -- On win, set "I" to appropriate message

05D0  TRS3:   0200  ;CLEAR -- Erase off screen to prepare for message
  D2          6C00  ;VC=0  -- Set up VX, VY coordinates in VC, VD
  D4          6D00  ;VD=0  --   for message display
  D6          6004  ;V0=4  -- Let V0 = loop count of 4 (lines)

  D8  TRS4:   0244  ;MSGR  -- Call MESSAGER MLS to show
  DA          DCD8  ;SHOW  --   one line of text
  DC          7D08  ;VD+8  -- Increase VY coordinate
  DE          70FF  ;V0-1  -- Decrement the loop count in V0
05E0          3000  ;SK=0  -- Skip next when loop count in 00
  E2          15D8   TRS4  -- Jump to do 4 lines of text - either
                           --   the win or lose message
  E4          A79B  ;ASCI4 -- Set "I" to general message
  E6          6D28  ;VD=28 -- Set VY coordinate in VD (VC not changed)
  E8          6005  ;V0=5  -- Let V0 = loop count of 5 (lines)

  EA  TRS5:   0244  ;MSGR  -- Call MESSAGER MLS to print
  EC          DCD8  ;SHOW  --   one line of text
  EE          7D10  ;VD+10 -- Add 10 hex to VY for line spacing
05F0          70FF  ;V0-1  -- Decrement the loop count in V0
  F2          3000  ;SK=0  -- When loop count is zero, skip next
  F4          15EA   TRS5  -- Jump to display 5 lines -- score and
                           --   restart messages at screen bottom
      ;SHOW SCORE

  F6          6C19  ;VC=19 -- Let VC = VX for displaying score
  F8          6D40  ;VD=40 --  "  VD = VY      "          "
  FA          8050  ;V0=V5 -- Transfer score to V0 to pass to sub
  FC          260A   NUMB3 -- Do sub to insert score at VC, VD

      ;RESTART ON
      ;KEY - 0

  FE          0216  ;FLOP  -- Switch to view messages & score
0600          0200  ;CLEAR -- Erase off screen page for clean restart
  02          6000  ;V0=0  -- Let V0 = 0 to test for Key 0
  04  TRS6:   E09E  ;SK=K0 -- Skip next on Key 0 pressed
  06          1604   TRS6  -- Loop back if not Key 0
  08          1500   BEGIN -- Jump back to restart on Key 0

      ;END MAIN PROGRAM                    5:02.17
```

```
        ; *********
        ; * NUMB3 *
        ; *********

        ; V0,V1,V2 USED
        ; INPUT : VC,VD
        ;   VALUE IN V0
        ; OUTPUT: NUMB
        ;   SHOWN

  0A  NUMB3: A624   C-3DD -- Set "I" to 3-byte work area
  0C         F033  ;DECIM -- Convert V0 into 3 decimal digits @ "I"
  0E         F265  ;GET   -- Let V0,V1,V2 equal those digits
0610         F029  ;SET I -- Set "I" to ROM bit pattern V0 value
  12         DCD5  ;SHOW  -- Display digit in V0
  14         7C05  ;VC+5  -- Increase VX for next digit
  16         F129  ;SET I -- Set "I" to ROM bit pattern V1 value
  18         DCD5  ;SHOW  -- Display digit in V1
  1A         7C05  ;VC+5  -- Increase VX for next digit
  1C         F229  ;SET I -- Set "I" to ROM bit pattern V2 value
  1E         DCD5  ;SHOW  -- Display digit in V2
0620         7CF6  ;VC-0A -- Reset VX to original value
  22         00EE  ;RET   -- Return from subroutine

  24  C-3DD: 0000  ;RM03  -- Three bytes needed for workspace
  26         0000  ; "    --   (fourth byte not used)

        ;END NUMB3 SUB

        ; *********
        ; * TIMER *
        ; *********

        ; V0=VALUE

  28  TIMER: F015  ;TI=V0 -- Let Timer = value passed in V0
  2A  TIME:  F007  ;V0=TI -- Let V0 = current timer value
  2C         3000  ;SK=0  -- When timer is zero, skip to exit
  2E         162A   TIME  -- Else loop to recheck timer
0630         00EE  ;RET   -- Return from subroutine (V0=0)

        ;END TIMER SUB

        ; ************
        ; *    BIT   *
        ; * PATTERNS *
        ; ************

0632  SHOOT: 10 10 10 38 7C FE -- Pie thrower shooter bits

0638  PUFF:  8080              -- Cream puff bits

063A  MICRO: 3E 3E 2A 2A 3E 14 1C 00 -- MICROMAN bits
             3E 3E 1C 14 00 36 36 00 --   "       "

        ;END BIT PATTS.

064A - 06FF -- NOT USED -- May be set to all zeros
```

5.02.18

```
        ; *****************
        ; * ASCII ENCODED *
        ; *   MESSAGES    *
        ; *****************

0700  ASCI1: 5D 41 54 54 41 43 4B 20 -- Title codes
  08         4F 46 20 20 54 48 45 5B --
0710         00 20 20 20 4D 49 43 52 --
  18         4F 4D 45 4E 00 20 20 62 --
0720         79 20 54 4F 4D 20 53 57 --
  28         41 4E 00

072B  ASCI2: 20 59 4F 55 20             -- Losing message codes
0730         48 41 56 45 20 42 45 45 --
  38         4E 00 20 20 53 54 4F 4D --
0740         50 45 44 20 49 4E 54 4F --
  48         00 56 41 4E 49 4C 4C 41 --
0750         20 50 55 44 44 49 4E 47 --
  58         00 20 20 20 20 20 20 20 --
0760         62 79 00

0763  ASCI3: 20 59 4F 55 52             -- Winning message codes
  68         20 42 52 49 4C 4C 49 41 --
0770         4E 54 00 20 20 50 49 45 --
  78         2D 4D 41 4E 53 48 49 50 --
0780         00 20 20 20 20 20 48 41 --
  88         53 00 4F 55 54 20 43 52 --
0790         45 41 4D 20 50 55 46 46 --
  98         45 44 00

079B  ASCI4: 20 20 54 48 45             -- Score, and restart
07A0         20 4D 49 43 52 4F 4D 45 --     message
  A8         4E 21 00 20 20 20 59 4F --       codes
07B0         55 20 53 43 4F 52 45 44 --
  B8         00 20 44 45 4C 49 43 49 --
07C0         4F 55 53 20 48 49 54 53 --
  C8         00 20 20 50 52 45 53 53 --
07D0         20 4B 45 59 20 30 00 46 --
  D8         4F 52 20 41 4E 4F 54 48 --
07E0         45 52 20 47 41 4D 45 00 --

07E8 - 07F9 -- NOT USED -- May be set to all zeros

07FA - 07FF -- MICROMAN VY STORAGE ARRAY -- Initial values are
                                              not important
```

NOW - - - 64K MEMORY AVAILABLE FOR YOUR VIP ! ! ! !



Introductory Special . . .

VSP626 32K ROM/RAM card with 4K of 6116 RAM installed plus one VSP001 expander/adapter: $149 + shipping

Same fully stuffed with 32K of 6116 RAM: $249 + shipping

Your VSP626 will be delivered set up for the lower half of memory space, i.e., ØØØØ-7FFFH. Memory addresses 7ØØØ-7FFF can be inhibited to permit use of on-board VIP RAM with your VSP626.

Prices subject to change without notice. Pennsylvania residents add 6% sales tax.

VIP Consultants:

George S. Gadbois, P.E.
David E. Van Zandt
P.O. Box 7062
Lancaster, PA 17604

Two VSP626 boards and VSP001 expander/adapter installed in VIP for 64K memory capability.

The VSP626 32K ROM/RAM card is a versatile memory module having on-board address latches and decoders. Address latches and data lines are buffered to minimize loading of the VIP system expansion bus. The VSP626 card contains 16 24-pin sockets which can be populated with 2716 EPROMs or 6116 CMOS RAMs or mixed ROM and RAM. A VSP001 expander/adapter is required to use the VSP626 memory card with the VIP.

The VSP001 bus expander/adapter provides one socket for expansion cards using the VIP system bus and four sockets for the VSP626 memory cards and other function cards to be announced soon. No modification of your VIP is required to add one VSP626 card for memory addresses ØØØØ to 7FFFH. Expansion of VIP memory above 8ØØØH requires that the VIP system ROM be removed and a new operating system be installed in your VSP626 memory card.

We are currently working on a new operating system for the VIP using ASCII keyboard input. We will be offering the new operating system installed in a VSP626 card for full 64K capability in your VIP in the near future.