VIPER

October - November 1982

Volume 4, Number 4 Journal of the VIP Hobby Computer Assn.
The VIPER was founded by ARESCO, Inc. in June 1978

****

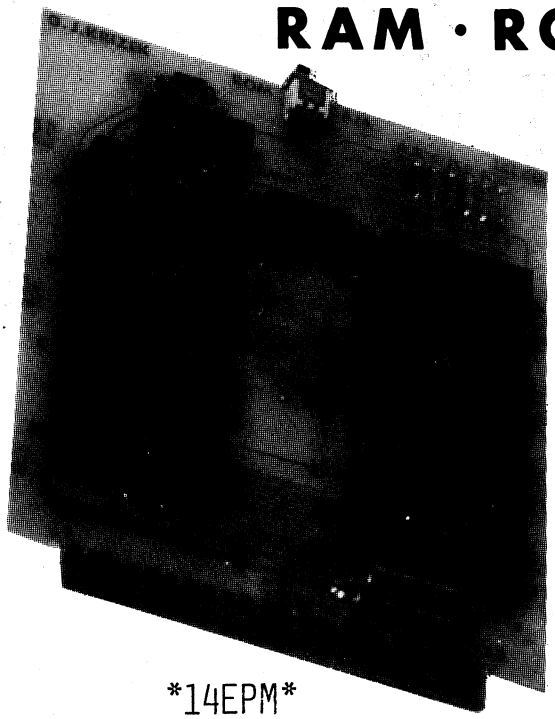
Contents

EDITORIAL

This issue, even with only two feature articles, nevertheless is so tight for space that this page is being used for the Editorial and Announcements. There are only two more issues of VIPER after this one in the 1982 membership year. And April 1st will be here before we know it. That date marks the end of the normal two-year term of yours truly as Director of VIPHCA. So according to our democratic proceoures, any member in good standing may nominate another member as Director.

I think most of you have been happy with yours truly in the job, and I'm willing to serve again, but we should afford the opportunity for a change, should you desire one. So, Nominations are open! If you want to nominate someone or yourself send in a letter or postcard. Anyone whose name is submitted will be contacted to see if the nomination is accepted by that person. The nominations will be posted in the next VIPER (4.05) and ballots sent to members, so that the results of the election can be printed in VIPER 4.06. Closing date for nominations is December 20, 1982.

RAM · ROM for VIP



14EPM

THE 14EPM GIVES YOU ALMOST INSTANT VP-701 FLOATING POINT BASIC. JUST SWITCH TO RUN, PRESS "1" ON HEX KEYPAD, AND PRESS "W" OR "C" ON YOUR ASCII KEYBOARD TO SELECT WARM OR COLD START. USES 5V 2716s.

BARE BOARD WITH DATA...$ 39.00
A&T LESS EPROMS.....$ 59.00
A&T W/ EPROMS PROGRAMED WITH DATA PROVIDED BY YOU*..$119.00
A&T W/ RCA's VP-701**...$139.00

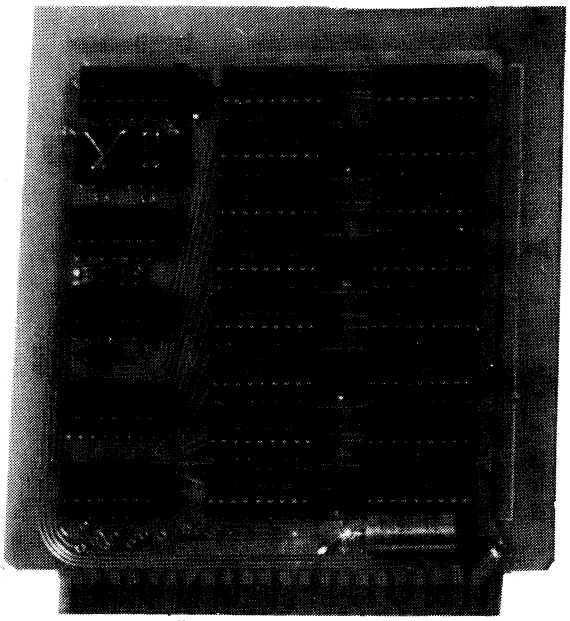
*DATA MUST BE ON VIP COMPATIBLE CASSETTE IN 2K BLOCKS.

**STATE YOUR HIGHEST RAM ADDRESS AND IF YOU WANT COLOR COMMANDS.

THE 8KRAM CARD GIVES YOU TWO 4K BLOCKS OF STATIC RAM ADDRESSABLE TO ANY 4K BLOCK IN VIP MEMORY. USES POPULAR 2114 RAMs.

BARE BOARD WITH DATA.$ 49.00
A&T LESS RAMS.....$ 79.00
A&T W/ RAMS.....$129.00

NOTE: 8KRAM CARD REQUIRES APROX 600mA FROM 5V LINE.



8KRAM

PACKAGE SPECIAL

2-8KRAM, 1-14EPM BARE BOARDS WITH DATA.....$ 99.00

2-8KRAM, 1-14EPM A&T W/ RAMS AND VP-701 BASIC.....$349.00

(CALIF. RESIDENTS ADD SALES TAX.)

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

A CHIP-8 Editor for the VIP

As attested by many previous articles about CHIP-8, using the monitor provided by the VIP is no way to enter and modify programs or data for CHIP-8. The editor enclosed is designed to solve the many drawbacks of the normal VIP monitor. This editor includes such features as the following to allow easier modification and entry of CHIP-8 programs: 2-byte display of address and data for easy reference to location in memory and data; arrow pointing to 2-byte word currently being edited; scrolling five lines of display; blinking cursor to signal location of nybble under editing; and automatic timed scrolling for easy checking of programs.

The editor, once it is typed in beginning at location $0D00, is invoked by changing the CHIP-8 word at $01FE to $1EEE, and entering the following sequence at $0EEE: 00 4B 1D 06. This will branch, upon running CHIP-8, to the editor. The entries at $0EEE are necessary to turn the display on, since the display-on instruction of CHIP-8 has been replaced by the branch to the editor. The editor may be saved as two blocks at location $0D00. When the editor is to be called, simply change the word at $01FE to $1EEE. Normal CHIP-8 may be run by rechanging this word to its original $004B word.

The format of the display used by the editor is as follows:

```
addr word
addr word
addr word
addr word <--
addr word
```

The top three lines and the bottom one display the surrounding words of the one currently under question. The "addr" indicates the address of the displayed word in the same line. The arrow shows which word is the one which may be modified. The arrow thus acts as a pointer to the current word.

The available commands in the editor are:

(E) Enter a CHIP-8 word at the location pointed to by the arrow. The first digit of the data word will blink; when the first digit of the new word is entered, the flashing cursor will move over and the next digit is entered. This is repeated three more times, until four digits, one word, has been entered. The display will scroll forward one line and the new word will be displayed one line above the new word at the arrow.

(A) Address Select. The first digit in the first column of the display will blink, just as in the "E" command; four new digits are entered, and the display will change to reflect the new addresses and the data contained therein. Note that the address is not limited to the first four kilobytes of RAM; any address may be specified. The CHIP-8 interpreter may thus be modified while running the editor, but this is not recommended procedure as the editor is written in CHIP-8.

(B) Back One Word. This command will scroll the display back one word, moving the byte previously under editing down one line, not changing the display line of the arrow, but rather just the word pointed to. The address and data columns will change to reflect this change in address.

(F) Forward One Word. This command will move the word previously editing up one line, in the same manner as the Back command. This occurs automatically in the Check Mode or when the Enter Data command is used.

(C) Check Mode. This command will automatically scroll the display forward one word every few seconds; the time delay byte is located at $0E09, the data byte of the VE=3C instruction. The editor will return to command mode when any key is pressed. This mode is useful for scanning entered programs for errors; first enter the address, press "C," and look at the display, the sheet of data; the display; the sheet; the display.....

(D) Do Program. This command will clear the screen and execute the CHIP-8 program at $0200. Note that the CHIP-8 interpreter itself is not modified with the changes made to it; only the initial sequence of cold start is changed. Therefore, all this command does is execute a GO 0200 command.

Useful Subroutines

Several subroutines have been incorporated in this editor, and, although they may or may not be of use in other programs, I have outlined the more useful ones here. The comments on the listing will also be of use if you are planning on implementing them in your programs.

ADD: 13 (decimal) bytes, at location $0E18. Totally relocatable. The CHIP-8 I pointer points to the byte to be incremented; an inline argument is the addend. This is a sixteen bit add, thus one is able to add to words and not simply bytes. If one wishes to simply add to a byte, the argument must be in the upper byte of the argument, with the lower byte being zero. Thus, to add $00FF to a byte at $0600, the following sequence may be used:

```
A600    ;I = $0600
0E18    ;call ADD
00FF    ;  argument to add
```

UNPACK: 24 (decimal) bytes at location $0E3A. Totally relocatable. The two byte word pointed to by "I" is unpacked into four bytes, each containing a zero in the high nybble, and a nybble of the data. These bytes are stored in CHIP-8 variables V0, V1, V2, and V3. Thus, if the memory word at "I" contained $1234, and this subroutine was called, the variables would be as follows:

```
V0 = $01
V1 = $02
V2 = $03
V3 = $04
```

This is used for the display of the address and data bytes on the screen. This is one of the methods of circumventing the lack of a CHIP-8 command to display ALL of a variable, and not just the low- order digit.

PACK: 22 (decimal) bytes at $0E52. Totally relocatable. This subroutine does the exact opposite as the UNPACK subroutine. The low- order nybbles of V0 through V3 are assembled into one two- byte word at a memory location which is determined by the setting of "I."

KEY: 31 (decimal) bytes at $0E68. Relocatable if branch addresses are changed or located on any page at address $XX68. This subroutine, in the manner of CHIP-8 decision instructions, is a skip. A skip over the next instruction occurs if no key is being pressed on the hex keypad. Otherwise, if a key is being pressed, the next CHIP-8 instruction is executed. This is the subroutine used by the Check Mode for the "break" state.

All of the above subroutines may be easily called by CHIP-8 with a "Branch to MLS" instruction. For example, to erase the screen if any key is pressed, use the KEY subroutine as follows:

```
0E68    ;CALL KEY
00E0    ;ERASE IF KEY PRESSED
```

If anyone does not wish to type this program in, and would prefer to have the program supplied on a cassette, send me a cassette at the following address, along with a dollar for postage and handling, and I will record the program on cassette. I am using a Sears recorder, and have found that Panasonic's units do not read my tapes. Yours may or may not- but then again, you can always type the program in.....

My address is:

William Lindley
21 Hancock St.
Bedford, MA 01730

CHIP-8 EDITOR
by Bill Lindley

```
MAIN:   0D00   1D06   ;BRANCH AROUND DATA
        0D02   0000   ;CURRENT ADDRESS HERE
        0D04   0000   ;CURRENT DATA HERE
        0D06   2D40   ;CALL FILL
        0D08   FF0A   ;
        0D0A   AD02   ;
        0D0C   3F0A   ;TEST ADDRESS CHANGE
        0D0E   1D14   ;
        0D10   2D96   ;CALL ADDR
        0D12   1D06   ;
        0D14   3F0B   ;TEST FOR BACK-UP
        0D16   1D1E   ;
        0D18   0E18   ;CALL ADD
        0D1A   FFFE   ; -2
        0D1C   1D06   ;
        0D1E   3F0C   ;TEST FOR CHECK SCAN
        0D20   1D26   ;
        0D22   2E00   ;CALL CHECK
        0D24   1D08   ;
        0D26   3F0D   ;TEST FOR DO PROGRAM
        0D28   1D2E   ;
        0D2A   00E0   ;ERASE SCREEN
        0D2C   1200   ;GO TO PROGRAM
        0D2E   3F0E   ;TEST FOR ENTER DATA
        0D30   1D36   ;
        0D32   2DA6   ;CALL ENTER
        0D34   1D06   ;
        0D36   3F0F   ;TEST FOR FORWARD
        0D38   1D08   ;
        0D3A   0E18   ;CALL ADD
        0D3C   0002   ; +2
        0D3E   1D06   ;
FILL:   0D40   00E0   ;ERASE SCREEN
        0D42   6B00   ;
        0D44   AD02   ;POINT TO CURRENT ADDR
        0D46   0E18   ;CALL ADD
        0D48   FFFA   ; -6
        0D41   1D52   ;LINE +6
LINE:   0D4C   AD02   ;
        0D4E   0E18   ;CALL ADD
        0D50   0002   ; +2
        0D52   6A00   ;
        0D54   0E3A   ;CALL UNPACK
        0D56   2D7C   ;GOSUB DISPLY
        0D58   AD02   ;
        0D5A   0E2F   ;CALL GET
        0D5C   0E3A   ;CALL UNPACK
        0D5E   2D7C   ;GOSUB DISPLY
```

```
          0D60  7B06  ;NEW LINE
          0D62  3B1E  ;FIND END OF LAST LINE
          0D64  1D4C  ;GOTO LINE
          0D66  AD02  ;
          0D68  0E18  ;CALL ADD
          0D6A  FFFE  ;  -2
          0D6C  6A2E  ;
          0D6E  6B12  ;
          0D70  AD76  ;POINT TO LEFT ARROW
          0D72  DAB5  ;DISPLAY IT
          0D74  00EE  ;RETURN
          0D76  2040  ;       (PATTERN
          0D78  BF40  ;          FOR
          0D7A  E000  ;       LEFT ARROW)
DISPLY:   0D7C  F029  ;
          0D7E  DAB5  ;DISPLAY IT
          0D80  7A05  ;
          0D82  F129  ;
          0D84  DAB5  ;
          0D86  7A05  ;
          0D88  F229  ;
          0D8A  DAB5  ;
          0D8C  7A05  ;
          0D8E  F329  ;
          0D90  DAB5  ;
          0D92  7A0A  ;
          0D94  00EE  ;RETURN
ADDR:     0D96  6A00  ;
          0D98  6B12  ;
          0D9A  AD02  ;
          0D9C  0E3A  ;CALL UNPACK
          0D9E  2DC4  ;GOSUB NEW
          0DA0  AD02  ;
          0DA2  0E52  ;CALL PACK
          0DA4  00EE  ;RETURN
ENTER:    0DA6  6A19  ;
          0DA8  6B12  ;
          0DAA  AD02  ;
          0DAC  0E2F  ;CALL GET
          0DAE  AD04  ;
          0DB0  0E3A  ;CALL UNPACK
          0DB2  2DC4  ;GOSUB NEW
          0DB4  AD04  ;
          0DB6  0E52  ;CALL PACK
          0DB8  AD02  ;
          0DBA  0E25  ;CALL PUT
          0DBC  AD02  ;
          0DBE  0E18  ;CALL ADD
```

```
        0DC0  0002  ; +2
        0DC2  00EE  ;RETURN
NEW:    0DC4  8400  ;
        0DC6  2DDE  ;GOSUB INPUT
        0DC8  8040  ;
        0DCA  8410  ;
        0DCC  2DDE  ;GOSUB INPUT
        0DCE  8140  ;
        0DD0  8420  ;
        0DD2  2DDE  ;GOSUB INPUT
        0DD4  8240  ;
        0DD6  8430  ;
        0DD8  2DDE  ;GOSUB INPUT
        0DDA  8340  ;
        0DDC  00EE  ;RETURN
INPUT:  0DDE  F429  ;
BLINK:  0DE0  6E08  ;
        0DE2  FE15  ;
        0DE4  DAB5  ;
NT0:    0DE6  0E68  ;CALL KEY - SKIPS IF NO KEY
        0DE8  1DF2  ;GOTO GI
        0DEA  FE07  ;
        0DEC  3E00  ;
        0DEE  1DE6  ;GOTO NT0
        0DF0  1DE0  ;GOTO BLINK
GI:     0DF2  DAB5  ;TEST FOR NUMBER DISPLAYED
        0DF4  4F00  ;
        0DF6  DAB5  ;ERASE IF ON
        0DF8  F429  ;    (SHOW THE
        0DFA  DAB5  ;     NEW NUMBER)
        0DFC  7A05  ;MOVE RIGHT
        0DFE  00EE  ;RETURN
CHECK:  0E00  AD02  ;
        0E02  0E18  ;CALL ADD
        0E04  0002  ; +2
        0E06  2D40  ;GOSUB FILL
        0E08  6E3C  ;SET TIMER DELAY VALUE
        0E0A  FE15  ;START TIMER
TK:     0E0C  0E68  ;CALL KEY
        0E0E  00EE  ;RETURN IF ANY KEY PRESSED
        0E10  FE07  ;    (CHECK FOR
        0E12  3E00  ;     TIMER DONE)
        0E14  1E0C  ;GOTO TK
        0E16  1E00  ;GOTO CHECK
```

```
                **ADD SUBROUTINE**
            **RA POINTS TO 1st ADDEND**
          **IN LINE ARGUMENT IS 2nd ADDNED**

ADD:    0E18  EA  SEX RA
           9  15  INC R5
           A  1A  INC RA   (ADD LOW BYTES)
           B  05  LDN R5   (RETURN THE SUM)
           C  F4  AD
           D  5A  STR RA
           E  2A  DEC RA
           F  25  DEC R5
        0E20  45  LDA R5   (ADD HIGH BYTES)
           1  74  ADC      (RETURN THE SUM)
           2  5A  STR RA
           3  15  INC R5
           4  D4  SEP R4   (RETURN)

                **PUT SUBROUTINE**
 **RA POINTS TO ADDRESS TO PUT DATA AT RA+2**

PUT:    0E25  4A  LDA RA   (SET
           6  BD  PHI RD     ADDRESS
           7  4A  LDA RA      TO PUT
           8  AD  PLO RD        DATA)
           9  4A  LDA RA   (STORE
           A  5D  SRT RD     1st BYTE)
           B  1D  INC RD
           C  4A  LDA RA   (STORE
           D  5D  STR RD     2nd BYTE)
           E  D4  SEP R4   (RETURN)

             ** GET SUBROUTINE **
  ** RA POINTS TO ADDRESS OF DATA TO GET **
          ** DATA GOES TO RA+2 **

GET:    0E2F  4A  LDA RA   (GET
          30  BD  PHI RD     ADDRESS
           1  4A  LDA RA      OF
           2  AD  PLO RD        DATA)
           3  4D  LDA RD   (MOVE
           4  5A  STA RD     1st BYTE)
           5  1A  INC RA   (MOVE
           6  0D  LDN RD     2nd
           7  5A  STR RA      BYTE)
           8  2A  DEC RA   (RESTORE RA)
           9  D4  SEP R4   (RETURN)
```

```
                ** UNPACK SUBROUTINE **
          ** RA POINTS TO DATA -2 BYTES **
         ** UNPACKED DATA GOES IN V0:V3  **

UNPACK:0E3A  F8  LDI      (POINT
          B  F1            TO
          C  A6  PLO R6     V1)
          D  4A  LDA R6   (GET 1st BYTE)
          E  56  STR R6   (STORE IN V1)
          F  F6  SHR
         40  F6  SHR
          1  F6  SHR
          2  F6  SHR
          3  26  DEC R6   (STORE
          4  56  STR R6     IN V0)
          5  0A  LDN RA   (GET 2nd BYTE)
          6  2A  DEC RA   (RESTORE RA)
          7  16  INC R6   (STORE
          8  16  INC R6     IN
          9  16  INC R6
          A  56  STR R6      R6)
          B  F6  SHR
          C  F6  SHR
          D  F6  SHR
          E  F6  SHR
          F  26  DEC R6   (STORE
         50  56  STR R6     IN V2)
          1  D4  SEP R4   (RETURN)

                ** PACK SUBROUTINE **
      **  RA POINTS TO 2 BYTES FOR RESULTS **
              ** INPUT FROM V0:V3 **

PACK:   0E52  E6  SEX R6   (GET
          3  F8  LDI
          4  F0               V0
          5  A6  PLO R6
          6  46  LDN R6        VALUE)
          7  FE  SHL
          8  FE  SHL
          9  FE  SHL
          A  FE  SHL
          B  F4  ADD        (ADD V1 VALUE)
          C  5A  STR RA     (STORE 1st BYTE)
          D  1A  INC RA     (GET
          E  16  INC R6        V2
          F  46  LDA R6          VALUE)
         60  FE  SHL
          1  FE  SHL
          2  FE  SHL
          3  FE  SHL
          4  F4  ADD        (ADD V3 VALUE)
          5  5A  STR RA     (STORE 2nd BYTE)
          6  2A  DEC RA     (RESTORE RA)
          7  D4  SEP R4     (RETURN)
```

```
          ** KEY SUBROUTINE **
    ** SKIPS IF NO KEY PRESSED, V4 ZEROED **
        ** IF KEY, THEN VALUE IN V4 **

KEY:   0E68  E6  SEX R6
          9  F8  LDI       (ADDRESS
          A  F4
          B  A6  PLO R6       V4)
          C  F8  LDI       (SET
          D  10               KEY SCAN
          E  56  STR R6           COUNT)
          F  06  LDN R6
         70  FF  SMI
          1  01
          2  56  STR R6
          3  62  OUT 2
          4  26  DEC R6
          5  E6  SEX R6    (NOP)
          6  36  B3
          7  7D              (GOTO DB)
          8  3A  BNZ
          9  6F              (GOTO LOOP)
          A  15  INC R5    (NO KEY, SO
          B  15  INC R5        SKIP AND
          C  D4  SEP R4           RETURN)
          D  F8  LDI       (SET
          E  81              R7 TO
          F  B7  PHI R7        POINT
         80  F8                  TO
          1  A1                     DEBOUNCE
          2  A7  PLO R7                ADDRESS)
          3  22  DEC R2    (DECREMNT STACK)
          4  D7  SEP R7    (DO DEBOUNCE SUB)
          5  12  INC R2    (RESTORE STACK)
          6  D4  SEP R4    (RETURN)

            **** END OF PROGRAM ****
```

ADVERTISEMENT

FOR SALE: 4K VIP with manual and video modulator, $50 plus shipping. VP-575 Expansion board, $25 plus shipping. VP-550 Super Sound board, $20 plus shipping. Model 15 TTY with spare motor and case of paper, $50 plus shipping. Radio Shack PC-1 with printer, manuals, program book, $225 plus shipping. All above in excellent condition. Call Jerry (213-338-2696) after 5PM California time.

Announcement

Programs in this VIPER will be available also from VIPHCA on a cassette for $2.00, which includes postage. Please send in a check (no cassette needed) and be sure to include your address.

Oktoberfest Reflexes

Udo Pernisz

A recurring feature of a local elementary school's fund raising activities is the Oktoberfest. It has a successful tendency towards imitating the more salient ambiente of Munich's colorful exhibition of attractions but lacks many of the circumferential and less conspicuous machines that nevertheless belong there.

Having always enjoyed determining one's sensomotoric reflexes with a mechanical device that usually can be found at Oktoberfests I decided to emulate a Response Time Test with my COSMAC VIP microcomputer. In the original set-up you feed the device a coin and wait for a green light to come on. At the same time this happens, a large clock hand starts moving from its center position clockwise and past fields that are approximately 1/10 sec apart. With a large button you try to stop the hand as fast as you can - it then points to some funny remarks about your nervous and other bodily systems just short of invectives.

Certainly this can be done with the VIP, and surely with sufficient timing accuracy. This is how:

The basis of the timing lies with the fact that the COSMAC VIP uses the Video Display Control chip CDP 1861 which sends interrupts to the microprocessor (the CDP 1802) every 1/60 s. This causes between 1 and 4 pages of RAM to be displayed on the monitor by a routine that is provided in the operating system. As will be seen in the analysis below, the time necessary to transfer bytes from RAM to the monitor screen is just slightly smaller than 1/120 sec. This allows to increment a counter just before the video routine, and right after it again, giving 120 counts per second. Thus 8.3 ms resolution is obtained for the time interval between the start and a stop signal, received at the I/O port or some sense line the processor can scan. Since the start signal would be issued by the computer its relation to the interrupt timing can be determined and accounted for in the program. The accuracy of determining the time interval that will have elapsed when the stop signal is received (as generated by the player) is then given by the duration of the display routine refreshing the monitor screen. Therefore, the timing accuracy is also 8.3 ms.

A two-page display format was chosen for the display of the game field which results in a vertical resolution of 64 lines. This format allows to include instructions for the game and still have the program fit into 4 kbytes of RAM.

With 64 lines vertically a quasi-analogous display of the elapsed time is obtained during the first second (which is the range of interest) by displaying a line segment, the time bar, on the monitor and moving it vertically by one line for each interrupt routine that the Video Controller chip requests.

The details of the program design considerations are determined by the system constants of the COSMAC VIP. The microprocessor operates with a crystal-derived clock frequency of 1.76064 MHz. The execution of a standard machine code instruction takes two machine cycles for fetch and execute each consisting of 8 clock cycles. During display the video interface issues 128 streams of 8 bytes each that are, in the 2-page mode, configured into 64 display lines. Each byte stream uses 14 machine cycles. All this gives a total of 1792 cycles for the display sequence. The interrupt routine itself uses 34 machine cycles (including the return routine).

This adds up to 1826 cycles. The 1/60 sec interrupt interval contains 1,760,640/60/8 = 3668 cycles leaving 1842 cycles = 921 machine code instructions for the timing program.

Since the display routine is equivalent to 1826/2 = 913 machine instructions one has left just 4 instructions in exactly half the interrupt interval of 1/120 seconds allowing to count (and also look for the STOP signal) every 1/120 seconds. Four instructions are enough to do this as the program listing shows.

The rest of the available time is for book-keeping, analog display of elapsed time and for time delay loops that do both the waiting and the exact positioning of the START signal that the computer will issue relative to the interrupt routine. The time flow chart in figure 2 shows these relations.

The machine language subroutine COUNT TIME both counts and converts the response time interval into seconds for display. Register RC counts multiples of the system unit time (1/120 s). It is preset to 01 (by incrementing RC at 041C), after the STOP button was checked for premature pressing. The preset allows for displaying the prompt (.. 00) and for the time spent in the first waiting loop (mem.loc. 0426 to 042B) before the stop button is first checked. In this way the desired timing accuracy of 8.3 ms is maintained through the start-up period before the timing loops begin. The stop button is checked at the end of the waiting loops (giving the player an average advantage of approximately 4 ms).

Other parts of the COUNT TIME subroutine will convert the timer count which is in units of 1/120 s into two numbers: the first one becomes the elapsed time in integral seconds while the second one has a value equal to the fractional time in centi-seconds. This allows " X.XX s " as display format. A rounding routine shifts the 8.3 ms accuracy to an error of +/- 4.2 ms which is less than half of the last digit displayed thus making the displayed time meaningful.

The time conversion first determines the number of integral seconds in the value of RC by subtracting 78_h which is 120_d. The remainder in RC (number of 1/120 s fractions totalling less than 120) has to be divided by 1.2 to arrive at the number that represents the same time in units of 1/100 s. This division is done by subtracting 1/6 of RC's value, say V, from V itself since V/1.2 = V - V/6. The division by 6 is performed using proper rounding with 03 (mem.loc. 047C). The result of this computation is stored directly into the CHIP-8 variable memory as V8 for the seconds, and as V9 for the centiseconds.

Another part of the subroutine COUNT TIME terminates counting after 2 to 8 seconds (software adjustable) if the STOP button was not hit (the numerical time display on the screen supports only 9.99 s) and - very important - prevents the timer bar from being written below the display pages as this would result in the self-destruction of the program.

The START and STOP buttons are connected to flag EF4 (inverted) through the small circuit shown in figure 3. It consists of four NAND gates that debounce the two switches. The Q-line is fed into one gate which resets the debounce circuits and puts the flag EF4(inverse) at HIGH. Another gate is used to prevent starting the game with the STOP button and vice versa stopping it with the START key (sequential interlock). This interlock circuit is reset each time the timer subroutine is entered.

This interface circuit is built up on an experimenter's board (Radio Shack Cat.No. 276-154) which plugs into the COSMAC VIP's I/O port that also provides the supply voltage for the two low-power Schottky ICs 74LS00.

After pressing the PAY key 0 (could be done by a coin-activated switch !) the program waits for the START button to be hit. It then enters a delay loop whose duration is randomly set between one and three seconds. The timing begins when the last word GO of the starting command READY, SET.. GO flashes on the screen. Simultaneously the timer bar starts moving upwards within the bar frame. Then one hits the STOP button as fast as possible to stop the action. Since the STOP button interlock is removed as soon as the START button activates the delay/timing routine, it is possible to "stop" the timer before it has even started. The program detects, however, whether STOP was pressed before the GO signal appeared (and the time bar started to move) and displays a "cheated" message.

The game ends with the time bar filling the bar frame up to the value of the response time, providing a quasi-analogous output of the time the player needed to react on the GO signal with pressing STOP. This time is also digitally displayed on the screen. In addition to this, a two-line comment is written out regarding the player's sensomotoric skills. This comment is selected from eleven texts depending on the 0.1 s interval into which the reaction time falls between 0.00 and)=1.00 s.

After a software-controlled display period of the time and comment the program returns to begin a new game. It may also be started by pressing key 0. While waiting for the PAY key 0 to be pressed (again) to start another game it alternates between displaying the rules and the game field.

The rules are entered into pages 8 and 9 by graphically decomposing the text into a display field of 64 x 64 pixels, see e.g. figure 4. These two pages may be separately changed by loading the data from tape into the two pages. Similarly, the comment texts, in pages 6 and 7, may be adapted to any occasion by replacing the memory contents accordingly. The format of the text requires two lines each five bits high and 4 bytes wide to be graphically filled with a bit pattern that appears as characters when displayed with CHIP-8's DXYN instruction. The data in memory locations 0609 to 060F govern this format. They should not be changed without understanding the workings. The texts start at 0610, each of the comments being 40 consecutive bytes long, all in one contiguous block. Thus, the "cheated" message is located at 0610 right after the steering data. The comment for a reaction time between 0.00 and 0.10 s starts at memory location 0650; between 0.10 and 0.20 s at 0690 and so on. The title of the game is stored in page 5 in memory locations 0508 to 0548. For details see the memory assignments in front of the program listing, and figures 5 and 6.

In its present version the program runs on a 4 kbytes machine. Since the instructions, the headlines, and comments are not essential to the response time determination itself, one could modify it to run on a 2 kbyte machine after adjusting the page data and eliminating the cycling between game page display and instruction page display. Page 5 would assume the role of page D and the 2-page game display would be from pages 6 and 7.

The program supporting the actual timing program is written in the interpretative language CHIP-8 that RCA makes available to the users of the VIP. The standard version has been modified by the addition of an interrupt routine that allows the display of two pages of RAM on the monitor. One possible implementation of such a modification has been suggested by Modla and Winsor in VIPER vol.1, issue 3, (October 1978), published at the time by ARESCO. The CHIP-8 interpreter modified for this 2-page interrupt routine has to be loaded into memory from 0000 to about 0250 - it does not use all of the third page. This prgram, however, starts at 0300 which makes its structure more transparent and, at the same time, provides space in page 02XX for individual program modifications.

Response Time Game

Machine: COSMAC VIP (RCA) with I/O port
Language: CHIP-8, 2-page display version
Author: Udo Pernisz

Memory Size: 4 kbytes of RAM
Allocation

Pages	Use
0, 1, 2	2-Page CHIP-8 Interpreter
3	CHIP-8 Program
4	Machine Language Subroutines + scale/frame data
5	Special Occasion Texts
6, 7	Twelve comments to the various response times
8, 9	Instructions
A, B, C	not used
D	partly used by CHIP-8 Interpreter
E, F	CHIP-8 Display Pages

Note: This listing does not contain pages 6 through 9. Data from which this program generates the comments, and the instructions, are user-supplied, see above and figures 1,4,5 and 6.

```
ADDRESS CODE    SYNTAX          :COMMENTS

0300    6800    V8=00           :define the PAY key: 0

                                DISPLAY INSTRUCTIONS AND WAIT FOR PAY KEY
0302    13A0    GO 03A0         :to draw the game field page (resumes @ 0320)
0304    0400    DO MLS @ 0400   :select page 8 & 9 for display: instructions
0306    65FF    V5=FF           :set (inner loop) timer
0308    6404    V4=04           :set (outer loop) timer
030A    F418    V4=TONE         :beep alert
030C    F515    TIME=V5         :start timer
030E    F307    V3=TIME         :
0310    E8A1    SKP;KEY.NE.V8   :check if pay key is pressed
0312    1330    GO 0330         :if so exit timer loops
0314    3300    SKP;V3.EQ.00    :
0316    130E    GO 030E         :loop back if time not up (inner loop)
0318    74FF    V4=V4+FF        :decrement outer loop counter
031A    3401    SKP;V4.EQ.01    :
031C    130C    GO 030C         :loop back if not done (outer loop)
031E    F418    V4=TONE         :beep alert if done using counter left-over

                                DISPLAY GAME FIELD AND WAIT FOR PAY KEY
0320    0404    DO MLS @ 0404   :select pages E and F as display: game field
0322    65FF    V5=FF           :set time
0324    F515    TIME=V5         :
0326    F507    V5=TIME         :
0328    4500    SKP;V5.NE.00    :
032A    1304    GO 0304         :go back to instructions if done
032C    E89E    SKP;KEY.EQ.V8   :check pay key
032E    1326    GO 0326         :loop back if not done
```

```
                                    MAIN ROUTINE
0330    C97F    V9=RND(7F)          :select random number 0 .. 7F =^ 2 s delay
0332    793F    V9=V9+3F            :add ca. 1 s as a base delay time
0334    0408    DO MLS @ 0408       :reset key debounce circuit & display pages E+F
0336    A54C    I=054C              :point to  "READY  SET ..."
0338    23C8    DO 03C8             :write text
033A    F915    TIME=V9             :set timer to the delay time
033C    F907    V9=TIME             :
033E    3900    SKP;V9.EQ.00        :
0340    133C    GO 033C             :loop if time not up
0342    A571    I=0571              :point to  " GO"
0344    23C8    DO 03C8             :write
0346    0410    DO MLS @ 0410       :main subroutine: real time display and count
0348    A54C    I=054C              :
034A    23C8    DO 03C8             :erase texts
034C    A571    I=0571              :
034E    23C8    DO 03C8             :
0350    A580    I=0580              :points to scratch pad for seconds display
0352    F465    V0:V4=MI            :read in steering variables
0354    8780    V7=V8               :V8 contains full seconds
0356    23E8    DO 03E8             :display digit for seconds
0358    A588    I=0588              :points to scratch pad for the 1/100 s display
035A    F465    V0:V4=MI            :
035C    8790    V7=V9               :V9 contains centiseconds
035E    23E8    DO 03E8             :display digits for centiseconds

                                    MESSAGE SELECTOR
0360    3800    SKP;V8.EQ.00        :check for t)=1 s
0362    1374    GO 0374             :
0364    6B0A    VB=0A               :prepare for separating time into 1/10 s pieces
0366    6A00    VA=00               :initialize table jump distance
0368    7909    V9=V9+09            :single out a 0.00 s time (STOPped before run)
036A    89B5    V9=V9-VB            :routine to divide by 10 begins here
036C    3F01    SKP;VF.EQ.01        :
036E    1376    GO 0376             :if time = zero
0370    7A14    VA=VA+14            :add (half) jump distance for each 0.1 s piece
0372    136A    GO 036A             :
0374    6ADC    VA=DC               :entry point for t )= 1.00 s
0376    A609    I=0609              :begin of messages table: steering variables
0378    F665    V0:V6=MI            :
037A    FA1E    I=I+VA              :point to applicable message part
037C    FA1E    I=I+VA              :(overcoming a variable range problem)
037E    23CA    DO 03CA             :display message (modified entry point!)
```

```
                                    AFTER-EVENT HOUSEKEEPING
0380    6800    V8=00               :reset the PAY key
0382    60FF    V0=FF               :set timer and define key F as program exit key
0384    6102    V1=02               :
0386    F015    TIME=V0             :
0388    F307    V3=TIME             :display resulting game field for some time
038A    E0A1    SKP;KEY.NE.V0       :check if program exit key is pressed
038C    1xxx    GO 0xxx             :wherever you have others program e.g. editors
038E    E8A1    SKP;KEY.NE.V8       :check pay key
0390    1300    GO 0300             :back to begin if pay key pressed
0392    3300    SKP;V3.EQ.00        :
0394    1388    GO 0388             :
0396    4100    SKP;V1.NE.00        :
0398    1300    GO 0300             :
039A    71FF    V1=V1+FF            :
039C    1386    GO 0386             :
039E    xxxx                        :(not used)

                                    TEXTS FOR GAME PAGES
03A0    0230    ERASE               :(relocated ERASE, was at 00E0 in 1-pg-
03A2    A698    I=0698              :scale          \ version of CHIP-8)
03A4    23C8    DO 03C8             :
03A6    A6A4    I=06A4              :scale numbering
03A8    23C8    DO 03C8             :
03AA    A6C9    I=06C9              :vertical bars of box
03AC    23C8    DO 03C8             :
03AE    A6D5    I=06D5              :horizontal bars of box
03B0    23C8    DO 03C8             :
03B2    A6E1    I=06E1              :decimal point and "sec"
03B4    23C8    DO 03C8             :

                                    OPTIONAL TEXT (OCCASION-ORIENTED)
03B6    A500    I=0500              :"OKTOBER"  and  "REACTION"
03B8    23C8    DO 03C8             :
03BA    A52F    I=052F              :"FEST"  and  "TIME"
03BC    23C8    DO 03C8             :
03BE    1320    GO 0320             :return into game field timing section
03C0    xxxx                        :(not used)
03C2    xxxx                        :
03C4    xxxx                        :
03C6    xxxx                        :

                                    TEXT DISPLAY SUBROUTINE
03C8    F665    V0:V6=MI            :read the steering data
03CA    8B20    VB=V2               :
03CC    8A10    VA=V1               :
03CE    6700    V7=00               :
03D0    DAB5    SHOW 5MI@VAVB       :uses display unit of size 5-by-8: suits text
03D2    F61E    I=I+V6              :position pointer to next group of bytes
03D4    7701    V7=V7+01            :increment horizontal block counter
03D6    9700    SKP;V7.NE.V0        :check against number of horizontal blocks
03D8    13DE    GO 03DE             :
03DA    8A34    VA=VA+V3            :move horizontally
03DC    13D0    GO 03D0             :to show another block
03DE    8B44    VB=VB+V4            :move vertically
03E0    75FF    V5=V5+FF            :decrement value of vertical blocks to do
03E2    3500    SKP;V5.EQ.00        :and check value
03E4    13CC    GO 03CC             :to reset horizontally to initial values
03E6    00EE    RETURN              :if done
```

```
                                        DIGIT DISPLAY SUBROUTINE
03E8    F733    MI=3DDE(V7)             :convert V7 which is the dummy
03EA    F01E    I=I+V0                  :position pointer to next group of bytes
03EC    F165    V0:V1=MI                :read two bytes (=digits)
03EE    F029    I=LSDP(V0)              :display the first one
03F0    D235    SHOW 5MI@V2V3           :
03F2    74FF    V4=V4+FF                :decrement digit counter
03F4    4400    SKP;V4.NE.00            :check
03F6    00EE    RETURN                  :if done with desired number of digits
03F8    F129    I=LSDP(V1)              :if not proceed to next digit
03FA    7206    V2=V2+06                :move horizontally
03FC    13F0    GO 03F0                 :to show the digit
03FE    xxxx                            :(not used)

ADDRESS OP-CODE LABEL   SYNTAX  OPERAND :COMMENTS


                * INSTRUCTIONS
0400    F8 08           LDI     08      :instruction pages, 08 and 09
0402    BB              PHI RB          :
0403    D4              SEP R4          :return

                * GAME FIELD
0404    F8 0E           LDI     0E      :display pages, 0E and 0F
0406    BB              PHI RB          :
0407    D4              SEP R4          :return

                * START
0408    7A              REQ             :prepares for next command which
0409    7B              SEQ             :resets the START/STOP switches
040A    F8 0E           LDI     0E      :selects games field pages for displ
040C    BB              PHI RB          :
040D    3F 0D   WTSRT   BN4     WTSRT   :wait for START, i.e. EF4 flag
040F    D4              SEP R4          :return if START button pressed

                * COUNT TIME
0410    F8 00           LDI     00      :
0412    BC              PHI RC          :
0413    AC              PLO RC          :RC - time counter set to zero

0414    F8 0F           LDI     0F      :
0416    BD              PHI RD          :
0417    F8 F2           LDI     F2      :
0419    AD              PLO RD          :RD points to time bar start location

041A    3F 50           BN4     SUMUP   :check for STOP pressed before start
041C    1C              INC RC          :pre-set timer to 1/120 s

041D    F8 04           LDI     04      :
041F    BA              PHI RA          :
0420    F8 97           LDI     97      :
0422    AA              PLO RA          :RA points to time bar symbol @ 0497
0423    EA              SEX RA          :RA = R(X) data pointer
```

```
0424    F0       MVBAR   LDX              :get symbol
0425    5D               STR RD           :put it into display page

0426    F8 94    WTLP1   LDI      94      :waiting loop part 1 counting 1/120 s
0428    AE               PLO RE           :
0429    2E       WAIT1   DEC RE           :
042A    8E               GLO RE           :
042B    3A 29            BNZ      WAIT1   :if not done

042D    3F 50            BN4      SUMUP   :exit loops if STOP button pressed
042F    1C               INC RC           :else count time, then
0430    00               IDL              :wait (interrupt from TV controller)
0431    3F 50            BN4      SUMUP   :check STOP button immed.after display
0433    1C               INC RC           :if still not pressed count time
0434    F8 94    WTLP2   LDI      94      :enter wait loop 2 for another 1/120 s
0436    AE               PLO RE           :
0437    2E       WAIT2   DEC RE           :
0438    8E               GLO RE           :
0439    3A 37            BNZ      WAIT2   :
043B    9C               GHI RC           :check counter for reaching upper time
043C    FB 01            XRI      01      :limit of ~ 2 s (up to 04 is sensible)
043E    32 50            BZ       SUMUP   :and exit loop if exceeded
0440    9D               GHI RD           :check time bar pointer address and
0441    FB 0D            XRI      0D      :stop writing into page when it points
0443    32 26            BZ       WTLP1   :below page E
0445    8D               GLO RD           :else decrement time bar pointer by 8
0446    FF 08            SMI      08      :which is the memory line-to-line dist.
0448    AD               PLO RD           :
0449    9D               GHI RD           :
044A    7F 00            SMBI     00      :
044C    BD               PHI RD           :new time bar position address back
044D    30 24            BR       MVBAR   :then go back to writing the time bar
044F    xx                               :(unused filler)

0450    F8 00    SUMUP   LDI      00      :
0452    AD               PLO RD           :RD.0 as counter for seconds
0453    F8 04            LDI      04      :
0455    BA               PHI RA           :RA is still data pointer R(X)
0456    F8 95            LDI      95      :
0458    AA               PLO RA           :RA points to mem.loc of lo-count of RC
0459    8C               GLO RC           :get counter value
045A    73               STXD             :store in scratch pad
045B    9C               GHI RC           :
045C    73               STXD             :
045D    72       SCNDS   LDXA             :gets datum 78-h = 120-d for determi-
045E    60               IRX              :nation of full seconds
045F    F5               SD               :
0460    73               STXD             :
0461    F0               LDX              :
0462    7F 00            SMBI     00      :
0464    73               STXD             :
0465    1D               INC RD           :counts full seconds
0466    FB FF            XRI      FF      :check if done
0468    3A 5D            BNZ      SCNDS   :
046A    72               LDXA             :if done adjust for too much subtracted
046B    60               IRX              :
046C    F4               ADD              :
046D    73               STXD             :
```

```
046E    5A              STR RA          :
046F    2D              DEC RD          :
0470    8D              GLO RD          :
0471    BD              PHI RD          :transfer seconds from RD.0 into RD.1
0472    F8 00           LDI     00      :re-initialize RD.0
0474    AD              PLO RD          :
0475    0A      CNTIS   LDN RA          :
0476    FF 06           SMI     06      :divide by 6 to get division by 1.2 ar
0478    5A              STR RA          :(1-1/6): convert 1/120 s to 1/100 s
0479    1D              INC RD          :
047A    33 75           BDF     CNTIS   :
047C    FC 03           ADI     03      :rounding routine
047E    33 83           BDF     FRQAJ   :
0480    2D              DEC RD          :
0481    38 00           SKP     00      :(two fillers)

0483    1A      FRQAJ   INC RA          :
0484    8D              GLO RD          :which is (1/6) of count as per M(R(A
0485    F5              SD              :subtract to get centiseconds
0486    AD              PLO RD          :and store back in RD
0487    96              GHI R6          :
0488    BA              PHI RA          :
0489    F8 F9           LDI     F9      :
048B    AA              PLO RA          :RA points to CHIP-8 variable V9
048C    8D              GLO RD          :get centiseconds and
048D    73              STXD            :store in V9
048E    9D              GHI RD          :get seconds and
048F    73              STXD            :store in V8
0490    D4              SEP R4          :return
0491    xx xx                           :(not used)

0493    78                              :=120-d: counts per integral seconds

0494    xx xx                           :storage for (1/60) sec from RC
0496    xx FF                           :symbol for the time bar
```

DATA STORAGE FOR TEXTS

```
0498    02 00 00 0B 0C 06 00    steering data; explained at 0609
049F    00 00 F0 00 00          SCALE
04A4    01 04 00 00 0C 06 05            st.d.
04AB    30 10 10 10 38          1
04B0    1C 14 08 14 5C          .8
04B5    1C 10 1C 14 5C          .6
04BA    10 14 14 1C 44          .4
04BF    1C 04 08 10 5C          .2
04C4    38 28 28 38 00          0
04C9    01 10 00 00 3F 02 00            st.d.
04D0    3F 00 00 00 00          horizontal bar of frame
04D5    02 10 00 08 05 0D 00            st.d.
04DC    C0 C0 C0 C0 C0          vertical bar of frame
04E1    04 20 26 08 00 01 05            st.d.
04E8    00 00 00 08 08          decimal point for centiseconds
04ED    00 00 00 00 00          space
04F2    00 07 04 03 07          "sec"
04F7    00 77 74 44 77
04FC    xx xx xx xx             (unused)
```

SPECIFIC TEXTS FOR THE OCCASION (here: Oktoberfest)

```
0500    04 1C 00 08 0E 02 05    steering data
        0E 0A 0A 0A 0E          "OKTOBER"
        AE C4 C4 A4 A4
        EE AA AC AA EE
        EE 8A CC 8C EA

        3B 2A 3B 32 2B          "RESPONSE"
        BB 2A 3A 2A AB
        BA 12 12 12 92
        EE AA AA AA EA

052F    02 24 07 08 0E 02 05    st.d.
        EE 88 CC 88 8E          "FEST"
        EE 84 E4 24 E4

        75 25 25 25 25          "TIME"
        F7 54 56 54 57

054C    03 1C 32 08 09 02 05
        EE A8 EC C8 AE          "READY,"
        EC AA EA AA AC
        A0 A0 E4 24 E8

        EE 88 EC 28 EE          "SET ..."
        E0 40 40 49 49
        00 00 00 02 20

0571    01 38 3B 08 00 01 00
        EE 8A 8A AA EE          "GO"
```

DATA AND SCRATCH PAD FOR SECONDS DISPLAY

```
0580    02 00 1E 26 01 xx xx xx     display control data for subroutine
0588    01 00 26 26 02 xx xx xx     at loc.03E8 and scratch pad for the
                                    3-digit decimal equivalent of sec &
                                    centiseconds
```

DATA STORAGE FOR COMMENTS

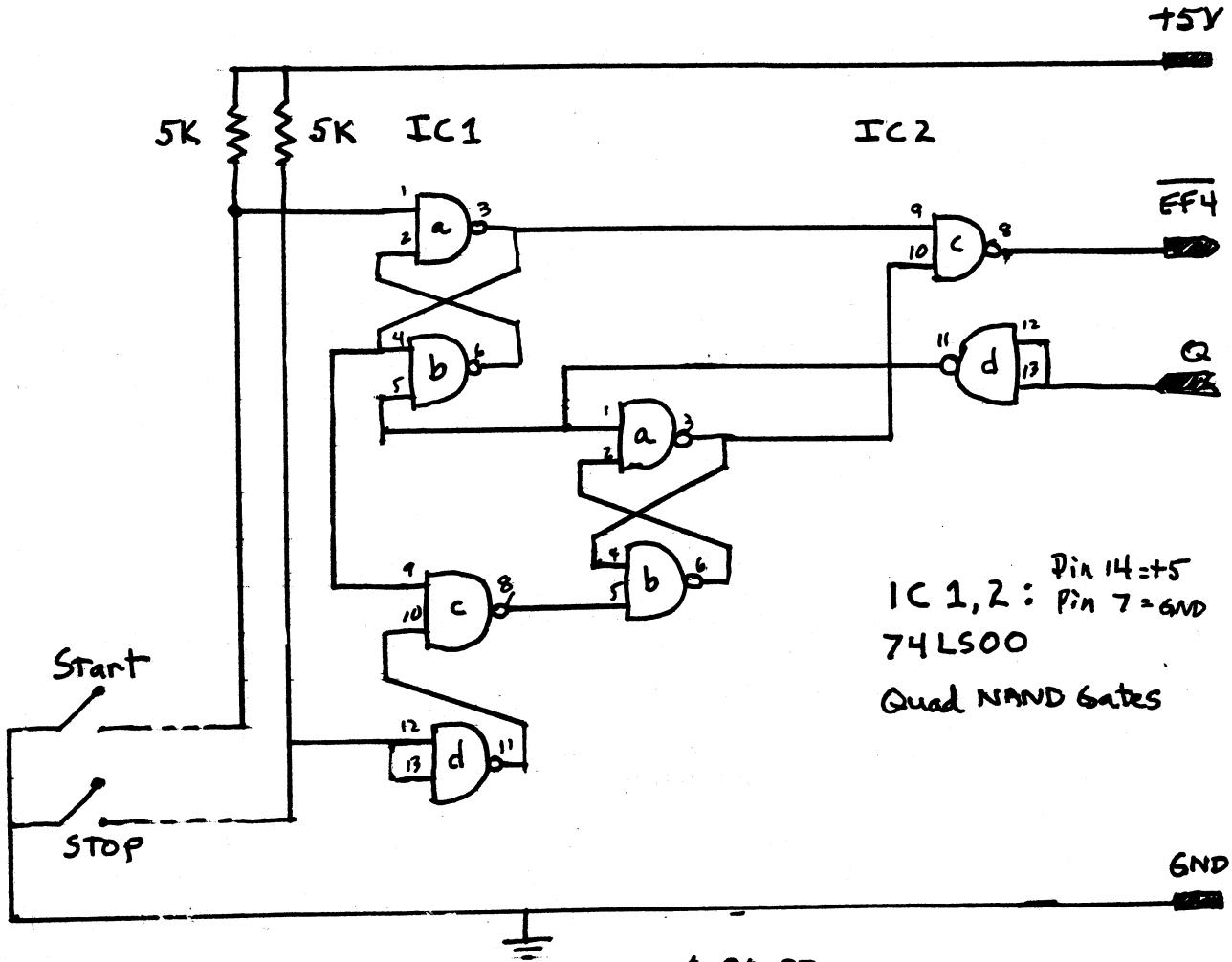
```
0600 .. 0608    (not used)
                    steering data definitions
0609    04      number of horizontal blocks
060A    1C      x-coordinate initial, upper left corner of block
060B    32      y-coordinate    "                   "
060C    08      horizontal increment
060D    09      vertical        "
060E    02      number of vertical blocks
060F    05      memory pointer increment

0610            comments (data user-supplied): 40 bytes/message, in two
                rows of four blocks, each block five bytes high.
```

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

FIRST CLASS

The interface for the START and STOP buttons plugs into the I/C port of the VIP and consists of two Quad NAND chips 74LS00. They are connected as two resettable switch-debounce circuits, IC1a,b and IC2a,b. Reset under program control is by the G-line which is inverted with IC2d. The STOP button is disabled by IC1c unless the START button is pressed first (IC1d inverts the STOP signal). The signals from the debounced buttons are connected to the microprocessor's flag line EF4(inverse) through the gate IC2c.



4.04.23