POP TRONICS MAR 77 MEMTEST

WISCONSIN COMPUTER SOCIETY
NEWSLETTER

Volume #2, Issue #8 September 1977 Don Stevens, Editor

****

MEETING NOTICE

Our Meeting will be held at 1:00 p.m., Saturday, September 3, 1977, at the Waukesha Technical Institute (room 202 - Admininstration Bldg.)

PROGRAM AGENDA

WCTI will give us a tour of their Computer facilities which is used for Administration and Student purposes. Some of the components in their computer system are Burroughs B-6700 Computer, 1.2 megabyte core, 500 megabyte disc, 2 card readers, 2 printers, and 28 data communication ports.

****

LOOKING FOR AN EXCELLENT BUY ON A ASR-33 TELETYPE complete with Modem?? Contact the writer at once. Only one (1) left.

****

Just received Volume 1, No. 1 of JUDGE, Joint Users of Digital Group Equipment NEWSLETTER. 20 pages of info on software and good info on Digital Group Systems. Price is $12.00 per year.

****

Radio Shack offers the following Computer System for only $600.00: Z-80 computer board, 4K RAM and ROM, 12" CRT Display, Keyboard, and Cassette Tape Recorder.

****

FREE - FREE copies of COMPUTER NOTES from MITS available at the meeting. Limited supply.

****

Newsletter Info

Please send your info for the Newsletter to:

Don Stevens

Don Stevens, Editor
Wisconsin Computer Society
P.O. Box 159
Sheboygan Falls, Wisc. 53085

6800 MOVER PROGRAM

Here is a handy little routine to move programs or data around in memory. I have used it to relocate programs and as an editor.

The program is designed to run with a MINIBUG II monitor. If you have a MIKBUG you will need to change:

E040	CONTRL	to	E0E3
E0D9	BADDR	to	E047
E130	PDATA1	to	E07E
A00F	ADDR	to	A00C

To relocate this program change the address in lines 10, 15 and 20. In fact MOVER will move itself.

```
                        NAM     MOVER   6800
                        OPT     S,O
1FA0                    ORG     $1FA0
     E040       CONTRL  EQU     $E040
     E0D9       BADDR   EQU     $E0D9
     E130       PDATA1  EQU     $E130
     A00F       ADDR    EQU     $A00F
     A01F       STADDR  EQU     $A01F
     A021       ENDADR  EQU     $A021
1FA0 CE 1FE3    MOVER   LDX     #START  POINTER TO START STRING
1FA3 BD E130            JSR     PDATA1  PRINT IT
1FA6 BD E0D9            JSR     BADDR   GET STARTING ADDRESS
1FA9 FE A00F            LDX     ADDR    LOAD ADDRESS
1FAC FF A01F            STX     STADDR  STORE IT
1FAF CE 1FEA            LDX     #STOP   POINTER TO STOP STRING
1FB2 BD E130            JSR     PDATA1  PRINT IT
1FB5 BD E0D9            JSR     BADDR   GET STOP ADDRESS +1
1FB8 FE A00F            LDX     ADDR    LOAD ADDRESS
1FBB FF A021            STX     ENDADR  STORE IT
1FBE CE 1FF1            LDX     #GOTO   POINTER TO DESTINATION STRING
1FC1 BD E130            JSR     PDATA1  PRINT IT
1FC4 BD E0D9            JSR     BADDR   GET DESINATION ADDRESS
1FC7 FE A01F    NEXT    LDX     STADDR  GET STARTING ADDRESS
1FCA BC A021            CPX     ENDADR  COMPARE TO END ADDRESS
1FCD 27 11              BEQ     END     DONE? GOTO END
1FCF A6 00              LDA A   0,X     GET CHARACTER
1FD1 08                 INX             POINTER  1
1FD2 FF A01F            STX     STADDR  STORE POINTER
1FD5 FE A00F            LDX     ADDR    GET DESINATION ADDRESS
1FD8 A7 00              STA A   0,X     STORE CHARACTER
1FDA 08                 INX             POINTER  1
1FDB FF A00F            STX     ADDR    STORE POINTER
1FDE 20 E7              BRA     NEXT    GET NEXT CHARACTER
1FE0 7E E040    END     JMP     CONTRL  RETURN TO MONITOR
1FE3 53         START   FCC     6,START
1FE4 54
1FE5 41
1FE6 52
1FE7 54
1FE8 20
1FE9 04                 FCB     4
1FEA 20         STOP    FCC     6, STOP
1FEB 53
1FEC 54
1FED 4F
1FEE 50
1FEF 20
1FF0 04                 FCB     4
1FF1 20         GOTO    FCC     6, GOTO
1FF2 47
1FF3 4F
1FF4 54
1FF5 4F
1FF6 20
1FF7 04                 FCB     4
                        END
```

```
CONTRL  E040
BADDR   E0D9
PDATA1  E130
ADDR    A00F
STADDR  A01F
ENDADR  A021
MOVER   1FA0
NEXT    1FC7
END     1FE0
START   1FE3
STOP    1FEA
GOTO    1FF1
```

"6800 MOVER PROGRAM" by T.D. Farnsworth appears courtesy Southern Florida Computer Group Newsletter, I/O, May 1977.

GOOD THINGS TO READ

Electronic Design - Sept. 1, 1977 (1) Time Stretcher Circuit speeds access in Z-80 microprocessor. (2) Revised data-interface standards permit data rate and longer

permit faster data rate and longer cables. New chips & RS232 adapters simplify their use.

EDN - August 20, 1977 (1) Microprocessor software programs bit-rate generator (MC-14411 Motorola). (2) Transform micro processor development systems into low cost LSI testers.

Computer Design - August 1977 Multiplexer System reduces cost of Terminal Interfacing.

Electronic Design - August 16, 1977 (1) Cut your processor's computation time by storing info in tables. (2) Get 32 times the bit rate instead of 16 from a programmable baud generator. (3) Simplify analog/computer interfacing.

EDN - August 5, 1977 (1) Interrupts add power, complexity to micro-computer system design. (2) Lookup tables provide logarithmic calculations.

****

I have been informed that the Memory Tester Program listed in the March 1977 issue is quite good.

****

Texas Instruments has introduced the TI Programmer (Hexadecimal and Octal calculator/converter for computer programmers). Priced at$49.95

117 vac Micro Control

BY HANK OLSON - HOMEBREW COMPUTER CLUB NEWSLETTER - MAY 13,1977

If one really wants to power a lamp, solenoid, or similar small ac load, a full-wave circuit is necessary. A typical circuit, using the same Monsanto MCS-1 photo-coupler, is shown in *Figure 2*. The addition of the bridge rectifier allows the SCR portion of the photo-coupler to operate across the + and − terminals of the bridge, where the pulsating dc is always of the correct polarity. If one is driving a small solenoid, motor, or other ac load which has appreciable inductance, it is wise to put a thyrite varistor (or one of the newer MOV types) across it. This will absorb inductive-kick transients that could otherwise exceed the SCR breakdown voltage of the MCS-1. Note that the MCS-1 is rated at 250 mA, and so these circuits are only good for about 30 W ac loads; but that is adequate for many small jobs such as hammer-drivers.

Note that using the MCS-1 as shown in *Figure 2* (or as in *Figure 1*) will not result in a system that has zero-voltage turn-on or zero-current turn-off, as is the case when using a true solid-state relay. This fact can cause a variety of transient and load current in-rush problems that may require attention. However, for simple ac line control, the MCS-1 (and its higher numbered MCS-series relatives) offers an inexpensive way to go.

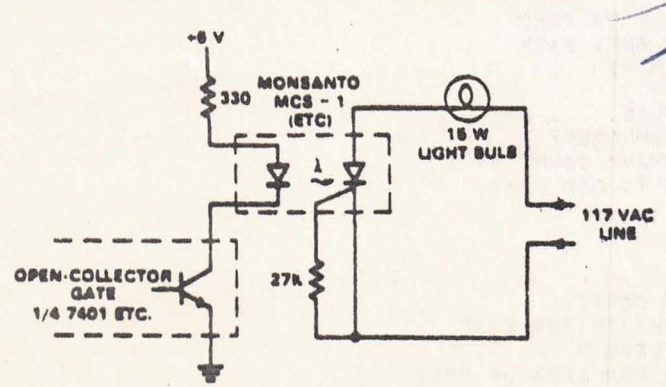

Fig. 1 Simple Half-Wave Control of ac with SCR-Photocoupler

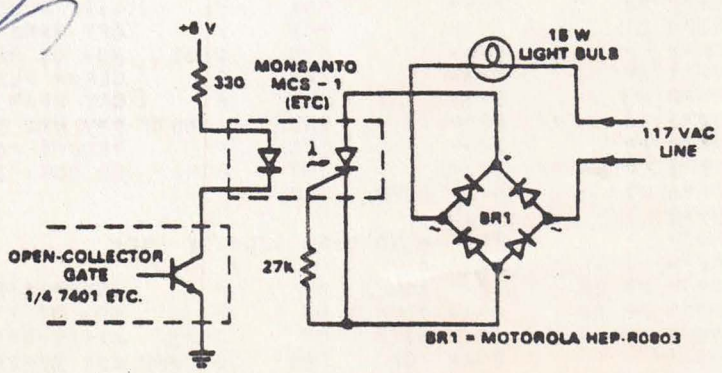

Fig. 2 Full-Wave Control of ac with SCR-Photocoupler

Newett Awl's Screen Sort

This little program does what is called a bubble sort. The idea behind the bubble sort is that each byte in the sorted area is compared to the next higher byte. If the lower byte has a higher value than the next higher byte, the two are swapped in memory. Then you look at the next two and do the same thing until you have passed through the whole sort area. After the first pass you know that the last byte in the sort area is higher than any other byte, so you don't need to sort it again. Consequently you can set up to pass over the sort area again, but you can shorten the sort area by one at the high end. If you keep track of whether there were any swaps during the pass, you can quit when no swapping takes place.

I have located this routine in low memory, so you don't need much to play with it. I have used the VDM screen area for the sort area, so if your eyes can be synced to the screen you can see exactly what is going on. Try putting a whole mess of garbage on the screen, then executing the program at location 0000. Your screen may never be the same!

Those of you who are adventuresome and mischievious might try to modify line 0047 to read 0047 SIZE DW, 2047 ARBITRARY LENGTH and 0048 TOP EQU, 0200H USE TV DAZZLER AREA. Then if you have Lichen Wang's Kaliedoscope, assemble this program so that it is above 0900 (HEX) and get the Kaliedoscope program running and then stop and run the sort program.

I've purposely coded the program so that it will sort any length field by manipulating the SIZE and TOP. A little practice and immagination should do the rest. I sorted the train program (naturally), but it broke my heart to see the little engine so badly and quickly torn apart. A couple of program notes: at line 40, the zero flag is been set or reset by the ADD B in line 37. As we all know but often forget, the DCX H at line 39 has no effect on the the accumulator flags.

At line 25, I just arbitrarily stuff register B with the last byte we looked at. This is an interesting use for the flag. At line 37, all I need to know is whether or not swapping occured. If the test at line 20 tells me that a byte in register A is greater than the byte in memory, I know for sure that it is not zero, so I can safely use it for a simple flag.

```
0000                0001 ;*********************************************
0000                0002 ;                                             *
0000                0003 ; NEWETT AWL'S SCREEN SORT FOR THE SIGHT-     *
0000                0004 ;              OF YOUR LIFE!                  *
0000                0005 ;                                             *
0000                0006 ;                                             *
0000                0007 ;*********************************************
0000                0008 ;
0000 2A 3F 00       0009 SORT   LHLD   SIZE   GET SIZE OF SORT AREA
0003 7D             0010        MOV    A,L    SEE IF SIZE IS
0004 B4             0011        ORA    H      ..ZERO
0005 CA 3A 00       0012        JZ     DONE   IF YES, LEAVE
0008 22 3D 00       0013 CON1   SHLD   COMPCT SET UP COMPARISON COUNT
000B 22 3B 00       0014        SHLD   ITMCNT RESET ITEM COUNT
000E 01 00 00       0015        LXI    B,0    CLEAR B/C
0011 21 00 CC       0016        LXI    H,TOP  POINT TO TOP OF AREA
0014 11 01 CC       0017        LXI    D,TOP+1  POINT TO TOP+1 OF AREA
0017 1A             0018 CON2   LDAX   D      GET ITEM B FROM AREA
0018 BE             0019        CMP    M      COMPARE TO ITEM A
0019 D2 21 00       0020        JNC    CON3   GO CON3 IF A<B
001C 4E             0021        MOV    C,M    GET A TO REG. C
001D 77             0022        MOV    M,A    PUT B WHERE A WAS
001E 79             0023        MOV    A,C    PUT A IN REG A
001F 12             0024        STAX   D      PUT A WHERE B WAS
0020 47             0025        MOV    B,A    USE BYTE FOR SWAP FLAG
0021 23             0026 CON3   INX    H      INCREMENT A POINTER
0022 13             0027        INX    D      INCREMENT B POINTER
0023 E5             0028        PUSH   H      KEEP MEMORY REFERENCE
0024 2A 3B 00       0029        LHLD   ITMCNT GET ITEM COUNT
0027 2B             0030        DCX    H      DECREMENT IT
0028 22 3B 00       0031        SHLD   ITMCNT RESTORE ITEM COUNT
002B 7D             0032        MOV    A,L    TEST IF
002C B4             0033        ORA    H      ..ITEM COUNT IS ZERO
002D E1             0034        POP    H      GET MEMORY REF. BACK
002E C2 17 00       0035        JNZ    CON2   NO, GO DO MORE
0031 AF             0036        XRA    A      CLEAR FLAGS
0032 80             0037        ADD    B      GET SWAP FLAG
0033 2A 3D 00       0038        LHLD   COMPCT GET NEW ITEM COUNT
0036 2B             0039        DCX    H      REDUCE COMPARE COUNT BY 1
0037 C2 08 00       0040        JNZ    CON1   GO CON1 IF IT DID SWAP
003A C9             0041 DONE   RET
003B                0042 ;
003B                0043 ; WORKING STORAGE AREA
003B                0044 ;
003B 00 00          0045 ITMCNT DW     0      INNER ITEM COUNTER
003D 00 00          0046 COMPCT DW     0      NO. OF ITEMS IN THE LIST
003F FF 03          0047 SIZE   DW     1023   ARBITRARY LENGTH
0041                0048 TOP    EQU    0CC00H USE SCREEN FOR AREA OF SORT
0041                0049 ;
```

Newett Awl's Screen Sort appears courtesy of the Homebrew Computer Club Newsletter, May 1977.

```
10 REM PIXMOX - FOR MITS BASIC AND PROC. TECH. VDM
11 REM VDM ACTION SIMULATION IN WHICH THE VDM 'MOON MAN'
12 REM BUILDS SOMETHING TO YOUR SPECIFCATIONS
20 VB=28672: REM VDM BASE ADDRESS (7000 HEX)
21 REM
22 REM DEFINE THE FUNCTIONS FOR CALCULATING VDM ADDRESS
23 REM
24 REM THEY ARE USED AS 'POKE FNP(0),CHR' WHERE FNP(0)
25 REM RETURNS AN ADDRESS ON THE VDM BASED ON Y AND X
26 REM
30 DEFFNP(Z)=VB+64*INT(Y)+X
35 DEFFNT(Z)=VB+64*TY+TX
40 DEFFND(Z)=VB+64*INT(Y+DY)+X+DX
45 DEFFNG(Z)=VB+64*GY+GX
50 FORI=1TO16:PRINT:NEXT:REM BLANK THE SCREEN
55 REM IF YOUR SCROLL ROUTINE USES VDM HARDWARE SCROLL
56 REM INSERT 'OUT XXX,0' WHERE XXX IS THE CONTROL PORT
60 POKEVB+960,32:REM BLANK THE CURSOR (LOWER LEFT)
65 REM DETERMINE SIZE, NUMBER OF CHARACTERS IN THE PIC.
70 FOR Y=0 TO 15
72 READ L$:REM READ A LINE
73 IF L$="END"THEN 90:REM HAVE READ THEM ALL
74 IFWD<LEN(L$)THENWD=LEN(L$):REM SAVE WIDEST LINE WIDTH
76 FORI=1TOLEN(L$):REM COUNT NON-BLANK CHARS
77 IF MID$(L$,I,1)<>" "THEN NB=NB+1
78 NEXT I:NEXTY
80 PRINT"PICTURE IS TOO BIG(TOO MANY LINES)":END
90 REM DONE READING TABLE - CENTER THE PICTURE
92 LM=INT((64-WD)/2):REM LEFT MARGIN
94 TM=INT((16-Y)/2):REM TOP MARGIN
96 REM DIMENSION TABLES
97 DIM B(NB,2):REM Y, X, CHAR OF BLOCKS STILL TO BUILD
98 DIM G(NB,2): REM Y,X OF 'GOAL' PICTURE
100 REM READ THE DATA TABLE TO GET THE CHARS, SCATTER THEM
102 RESTORE:CN=0:REM CHAR NUMBER
110 FOR L=0 TO 15
112 READ L$:REM GET A LINE
113 IF L$="END"  THEN 200:REM DONE
114 FORI=LEN(L$)TO1STEP-1
116 IF MID$(L$,I,1)=" " THEN 142
118 CN=CN+1:REM BUMP CHAR NUMBER
120 G(CN,0)=L+TM:REM Y POSITION IN GOAL TABLE
122 G(CN,1)=I+LM:REM X POSITION IN GOAL TABLE
124 G(CN,2)=ASC(MID$(L$,I,1)):REM CHAR
125 REM CALCULATE RANDOM Y,X POSITIONS
126 Y=INT(16*RND(1)):X=INT(64*RND(1))
128 REM IF THE LOCATION IS NOT EMPTY, TRY AGAIN
130 IF PEEK(FNP(0))<>32 THEN 126
132 REM THE LOCATION IS EMPTY
134 REM STORE CHAR, UPDATE THE TABLE
136 B=ASC(MID$(L$,I,1)):REM BLOCK THE STORE
138 B(CN,0)=Y:B(CN,1)=X:B(CN,2)=B
140 POKE FNP(0),B
142 NEXT I:REM NEXT CHAR IN LINE
150 NEXT L:REM NEXT LINE OF INPUT
200 REM
201 REM PLACE THE BUILDER ON THE SCREEN
202 REM
210 RESTORE:REM POINT TO START OF DATA TABLE
220 REM CALCULATE RANDOM Y,X FOR BUILDER
```

```
230 Y=INT(2+12*RND(1)):X=INT(2+60*RND(1))
240 REM MAKE SURE NO BLOCK IS ALREADY THERE
250 IFPEEK(FNP(0))<>32THEN230
260 REM STORE THE BUILDER
270 POKE FNP(0),7
280 G=NB:REM BUILD FROM BOTTOM UP
295 REM
296 REM MAJOR PROGRAM LOOP - GET A GOAL Y,X VALUE
297 REM THEN MOVE A BLOCK THERE
298 REM
300 IF G=0 THEN 7000:REM WE ARE DONE
310 GY=G(G,0):GX=G(G,1):GC=G(G,2):REM GOAL Y, X, CHAR
311 G=G-1
315 V=7:REM INITIALIZE CHAR OF THE BUILDER
320 REM SEE IF THERE IS A BLOCK THERE
325 REM CALCULATE DY DX TO MOVE TOWARD BLOCK LOCATION
330 DY=SGN(GY-Y):DX=SGN(GX-X)
340 IF FNP(0)=FNG(0) THEN 400:REM NO BLOCK THERE
350 IF PEEK(FND(0))<>32THEN 380:REM CAN'T MOVE
355 REM WE CAN MOVE TOWARD BLOCK LOCATION, DO SO, LOOP.
360 POKE FNP(0),32:Y=Y+DY:X=X+DX:POKEFNP(0),7:GOTO330
375 REM REM CAN'T CONTINUE -
380 IF FND(0)<>FNG(0)THEN GOSUB 8000:GOTO330
381 POKE FNP(0),32:Y=Y+DY:X=X+DX
382 FOR I=1TONB:IF(Y=B(I,0))AND(X=B(I,1))THEN386
383 NEXT:PRINT"CAN'T FIND  "Y;X"IN BLOCK TABLE B.":STOP
385 V=PEEK(FND(0))
386 V=PEEK(FNP(0)):GOSUB 9000:B(I,0)=Y:B(I,1)=X:V=7
389 REM GET OUT FROM UNDER
390 DY=1-INT(3*RND(1)):DX=1-INT(3*RND(1))
391 IF PEEK(FND(0))<>32 THEN 390
392 Y=Y+DY:X=X+DX:V=7:POKE FNP(0),V:GOTO 330
399 REM
400 REM FIND A BLOCK TO MOVE
401 POKE FNP(0),160:POKE FNP(0),7
405 V=7
410 FOR I=1TONB
415 IF B(I,2)=GC THEN 430
420 NEXT
425 PRINT "CAN'T FIND'"CHR$(GC)"' BLOCK TO MOVE":STOP
430 B(I,2)=0: TY=B(I,0): TX=B(I,1)
432 POKE FNT(0),128+GC:POKE FNT(0),GC
435 DY=SGN(TY-Y):DX=SGN(TX-X)
440 IF FND(0)=FNT(0) THEN 500:REM FOUND BLOCK
445 IF PEEK(FND(0))<>32 THEN GOSUB 8000: GOTO 435
450 POKE FNP (0),32
455 Y=Y+DY:X=X+DX
460 POKE FNP(0),7:GOTO 435
499 REM
500 REM FOUND BLOCK, CRAWL UNDER IT
510 POKE FNP(0),32:REM BLANK SELF
520 Y=Y+DY:X=X+DX
530 V=GC:REM MOVE THE GOAL CHAR
540 POKE FNP(0),V
600 REM MOVE THE BLOCK INTO PLACE
610 DY =SGN(GY-Y):DX=SGN(GX-X)
620 IF FNP(0)=FNG(0) THEN 700:REM BLOCK IN PLACE
630 IF PEEK(FND(0))<>32 THEN GOSUB 8000:GOTO 600
640 POKE FNP(0),32
650 Y=Y+DY:X=X+DX:POKE FNP(0),V
```

```
660 GOTO 600
700 REM BLOCK IS IN PLACE
710 REM MOVE OUT FROM UNDER IT
715 DY=-1:DX=0:IF PEEK(FND(0))=32 THEN740
720 DY=1-INT(3*RND(1)):DX=1-INT(3*RND(1))
730 IF PEEK(FND(0))<>32 THEN 720
740 Y=Y+DY:X=X+DX:POKE FNP(0),7
750 GOTO 300
7000 REM DONE BUILDING HOUSE - JUST WALK AROUND
7010 REM UNTIL CONTROL-C IS TYPED
7014 V=7:REM MAKE THE MOON MAN CHAR
7020 GOSUB 9000
7030 GOTO 7020
8000 REM ATTEMPT TO GO AROUND OBSTACLE
8010 REM LOGIC IS TO GO RIGHT, THEN LEFT,
8014 REM OR LEFT, THEN RIGHT
8018 REM IF EITHER IS BLOCKED, TRY THE OTHER
8020 IF RND(1)> .7 THEN 9010:REM OCASSIONALLY RANDOM
8022 IF RND(2)> .5 THEN 8110:REM RIGHT, THEN LEFT
8030 REM MOVE LEFT, THEN RIGHT
8032 GOSUB 8500:REM LEFT
8034 IF SU THEN 8400:REM LEFT OK, TRY RIGHT, RETURN
8100 REM TRY RIGHT, THEN LEFT
8110 GOSUB 8400:REM TRY RIGHT
8120 IF SU THEN GOSUB 8600
8130 GOTO 8500:REM TRY MOVING LEFT, THEN RETURN
8400 REM RIGHT TURN
8405 OY=DY:OX=DX:REM SAVE OLD
8410 T=DX:DX=-DY:DY=T
8412 IF PEEK(FND(0))<>32THEN8900:REM SHOW FAILURE
8420 POKE FNP(0),32:REM BLANK OLD POSITION
8425 Y=Y+DY:X=X+DX:POKE FNP(0),V:REM SHOW NEW POSITION
8430 SU=-1:REM SHOW SUCCESS
8440 RETURN
8500 REM TRY LEFT TURN
8505 OY=DY:OX=DX:REM SAVE OLD
8510 T=DX:DX=DY:DY=-T
8515 IF PEEK(FND(0))<>32 THEN 8900:REM FAILURE TO MOVE
8520 GOTO 8420:REM MOVE, SHOW SUCCESS
8600 REM RANDOMLY MOVE 1 MORE IN THE SAME DIRECTION
8610 IF RND(1)> .5 THEN RETURN
8620 IF PEEK(FND(0))<>32 THEN RETURN
8630 POKE FNP(0),32:REM BLANK OLD
8640 Y=Y+DY:X=X+DX:POKE FNP(0),V:RETURN
8900 REM FAILURE TO MAKE TURN
8910 DY=OY:DX=OX:REM RESTORE OLD VALUES
8920 SU=0:REM SHOW NO SUCCESS
8930 RETURN
9000 REM RANDOM MOVER, V=VALUE BEING MOVED
9010 DY=1-INT(3*RND(1))
9020 DX=1-INT(3*RND(1))
9040 L=64*RND(1)*RND(1)*RND(1)
9050 IF PEEK(FND(0))<>32 THEN RETURN
9060 REM OPEN PLACE, MOVE TO IT
9070 IF (Y+DY)<0 OR (Y+DY)>15 THEN 9010
9080 IF (X+DX)<0 OR (X+DX)>63 THEN 9010
9090 POKE FNP(0),32:REM OLD BLANK
9100 Y=Y+DY:X=X+DX:POKE FNP(0),V
9110 L=L-1
9120 IF L>0 THEN 9050
```

```
9130 IF RND (1)> .5 THEN RETURN
9140 REM TURN RIGHT OR LEFT. GO AGAIN
9145 IF RND(1)>.5THEN9200
9150 REM GO RIGHT
9155 T=DX:DX=-DY:DY=T:REM CALCULATE RIGHT TURN
9160 GOTO 9040
9200 REM GO LEFT
9205 T=DX:DX=DY:DY=-T:REM CALC LEFT TURN
9210 GOTO 9040
9999 REM HERE IS THE PICTURE FOR THE MOON MAN TO DRAW..
10000 DATA "        ^"
10010 DATA "    -  / \"
10020 DATA "   ! !/   \"
10030 DATA "   ! /     \"
10040 DATA "   !/       \"
10050 DATA "   /   ---   \"
10060 DATA " !    !-!-!    !"
10065 DATA " !    !-!-!    !"
10070 DATA " !     ---     !"
10080 DATA " !    !   !    !"
10090 DATA " |    |  o|    |"
10100 DATA " !---!---!---!"
10110 DATA "END"
```

****

HOBBY COMPUTER KIT OFFERED BY RCA. Priced at $275.00 in kit form, the VIP is a complete computer on a printed circuit card, offering a powerful, uncluttered, complete operating system in only 4K bits of ROM. VIP's output directly interfaces with a monochrome CRT display or, when used with an FCC-approved modulator, a TV receiver. Programs can be generated and then stored in an audio cassette tape recorder for easy retrieval and use. The VIP features a single 8-1/2 x 11" PC card with the CDP1802 microprocessor, 2,048 byte RAM using 4K-bit static RAMs, single-chip graphic video display interface, built-in hexadecimal keyboard.

****

Itaca Audio, Box 91, Ithaca, N.Y. 14850, offers S100 Z-80 Board for $35.00

****

Questionaires returned to date:

8080 Systems:

Don Senzig
Milt Krauthoff
Alfred Bode
Peter Davis
Tom Artzberger
Jeff Smith
Tod Templin
Dave Saar
Darrell Wood
Tom Doyle
Larry Leranth
Joseph Leair
Darryl Uchitil

Z-80 Systems

Steve Ujvarosy
Robert Senzig
Tom Carlton

TI-990 System

William Mack

6800 Systems

Julian Jetzer
Stephen Heinecke

KIM-1 6502

John Geiger