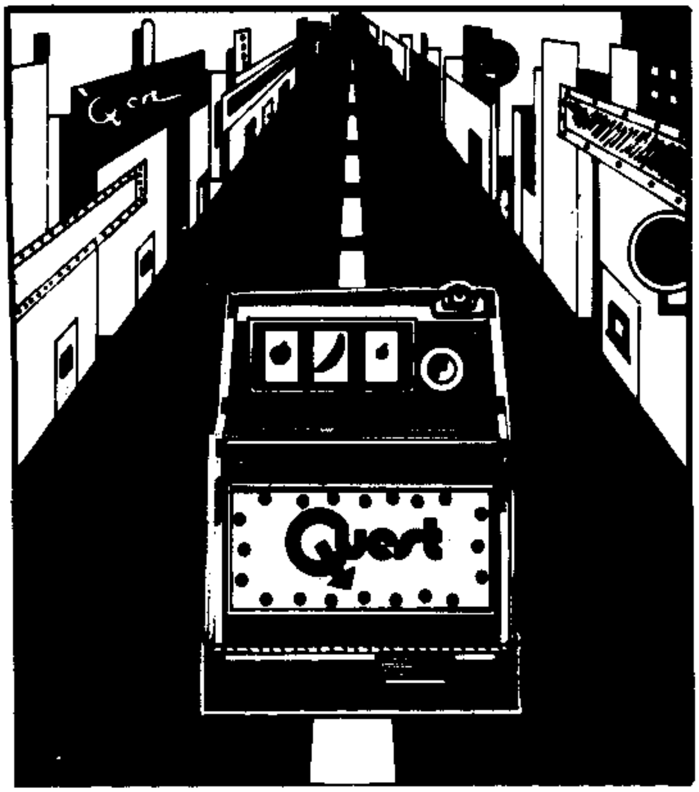
Questdata
Volume 2 Issue # 7 ©

ONE-ARMED BANDIT

by
L. Owen

This program is a game which I call "One-Armed Bandit". It was conceived after I had built my own home-brew 1802 system, and had to answer the inevitable question, "But, what does it do?". Through various rewrites, it has matured to its present form, in which it will run on the basic Super Elf with only 256 bytes of memory. It will also run on expanded memory systems, as long as it is not entered with R2 as the program counter. Note that any loading errors beyond address XX20 hex may be corrected using the Super Elf's basic monitor, even though the program starts at the beginning of the page. Also, experimentation with the tone sequences is facilitated by this feature.

This game is modeled after the infamous slot machine, which, when its handle is pulled, spins some dials, with the object being to match up two or more of the dials. This "round" of play costs the player a specific amount of money to play. As the Elf has only two digits which are easily controllable, the game will allow three pulls of the crank (Input switch depressions) before charging the player. Each "pull" spins one dial at a time, with the first being the "A"pples dial, the second being the "B"ananas dial, and the third being the "C"herries dial. These three pulls constitute one round of play, which costs one point to play. The game starts the player off with three points, and calculates and displays the new score after every round of play. Points are added for every match found amongst the dial spins, and also if any of the spins match certain "secret" internal numbers. The game continues until the score drops to zero, or climbs to ten. In the first case, the display will show FF for FFoowy, and a "Lose Tune" will be played. In the event of a win, the display will show BB for Beat the Bank, and a "Win Tune" will be played. In both cases, a new game may be started by depressing the Input switch again.



Also, when the dials spin, they go through the sequence 0,1,——8,9,0,1,etc., and a beep sound is generated for each increment. This action occurs when the Input switch is pressed, and stops when the switch is released.

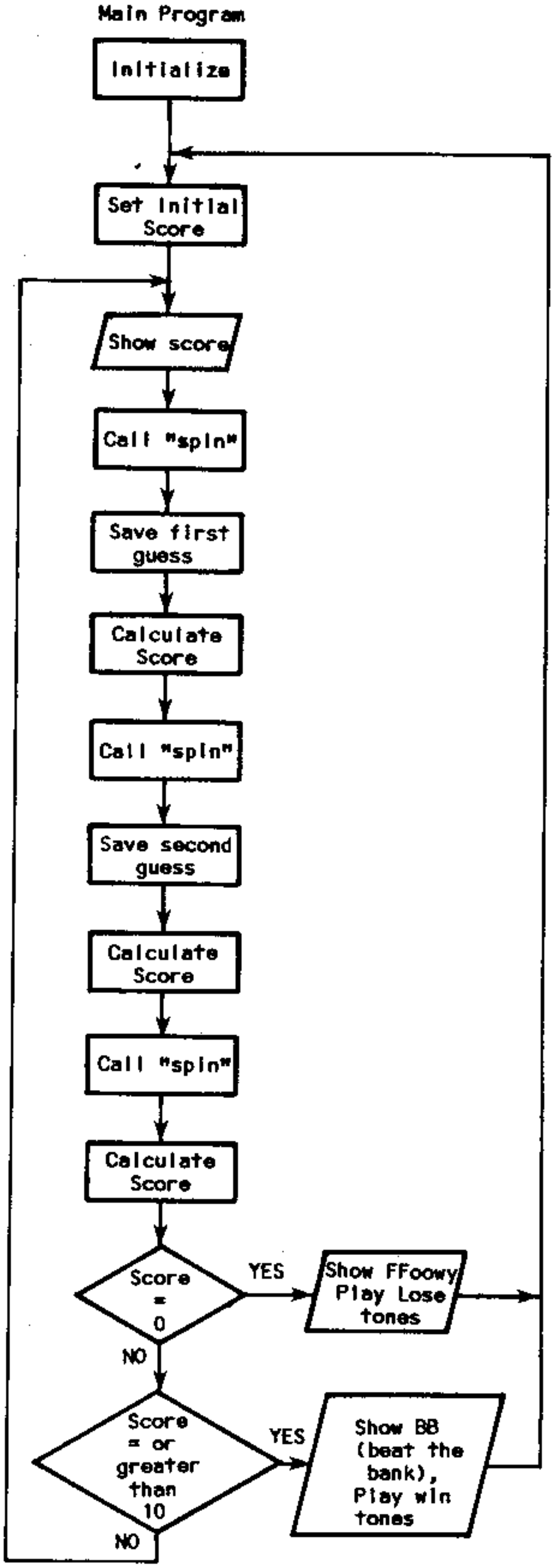
The sound effects are produced using a music program which was published in Popular Electronics, and which I rewrote in the form of a subroutine. To allow experimenting with the tunes, I am including the following table:

These notes are not exact, as they were originally calculated for a system running at 2MHz. However, they are adequate for this program and for experimenting.

At any rate, my family, friends, and I have had a great deal of fun with this program. I hope that your readers will too.

One-Armed Bandit Table

Note	1/4 Duration	1/2 Dur.	1 Dur.	2 Dur.	Pitch
D	24	49	93	--	12
C#	22	45	8B	--	14
C	20	41	83	--	15
B	1E	3D	7B	F6	17
A#	1D	3A	75	EA	19
A	1B	37	6E	DC	1B
G#	1A	34	68	D0	1D
G	18	31	62	C4	1F
F#	17	2E	5D	BB	22
F	15	2B	57	AE	24
E	14	29	52	A4	27
D#	13	27	4E	9C	2A
D	12	24	49	92	2D
C#	11	22	45	8A	30
C	10	20	41	82	33
B	0F	1F	3E	7C	37
A#	0E	1D	3A	74	3B
A	0D	1B	37	6E	3F
G#	0D	1A	34	68	43
G	0C	18	31	62	47
Rest	0B	16	2D	5A	C0 to FF



MUSIC TO MARCH BY

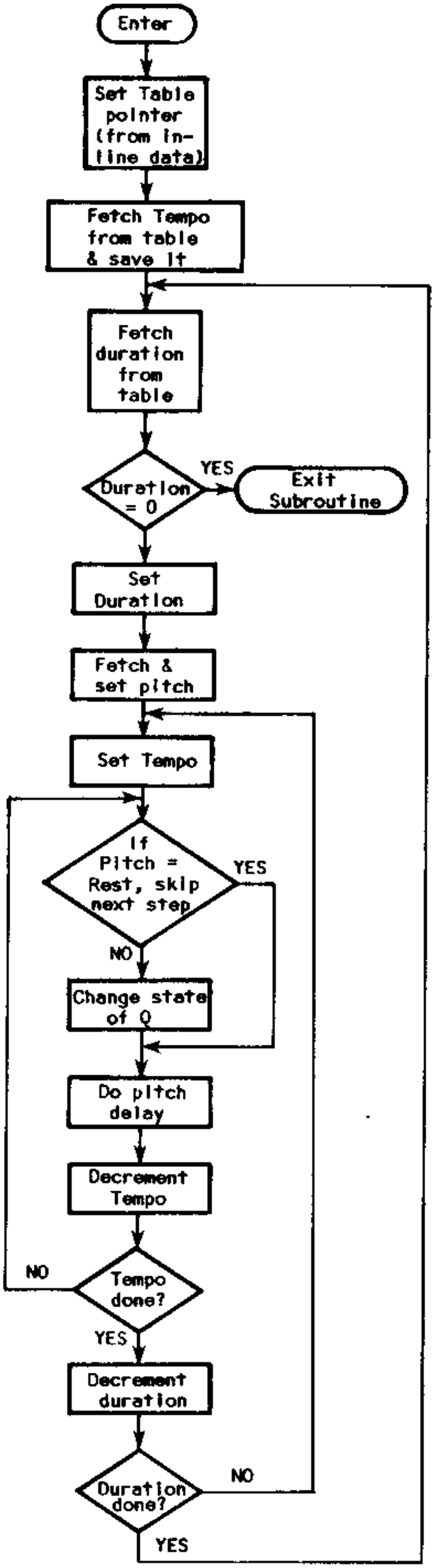
by
Dan Van Dyke

This program is designed to run on a basic Super Elf (256 bytes). The following program uses all of the available 1/4K or RAM and also Paul C. Moews music generator "Music Algorithm" program which appeared in Questdata Volume 1, Issue 10, page 6. The "Music Algorithm" must be loaded into location 00 thru location 2F. Be sure to put 23 into location 0001! The table of music (which is "When the Saints go Marching In") starts at location 30 and ends at location 98.

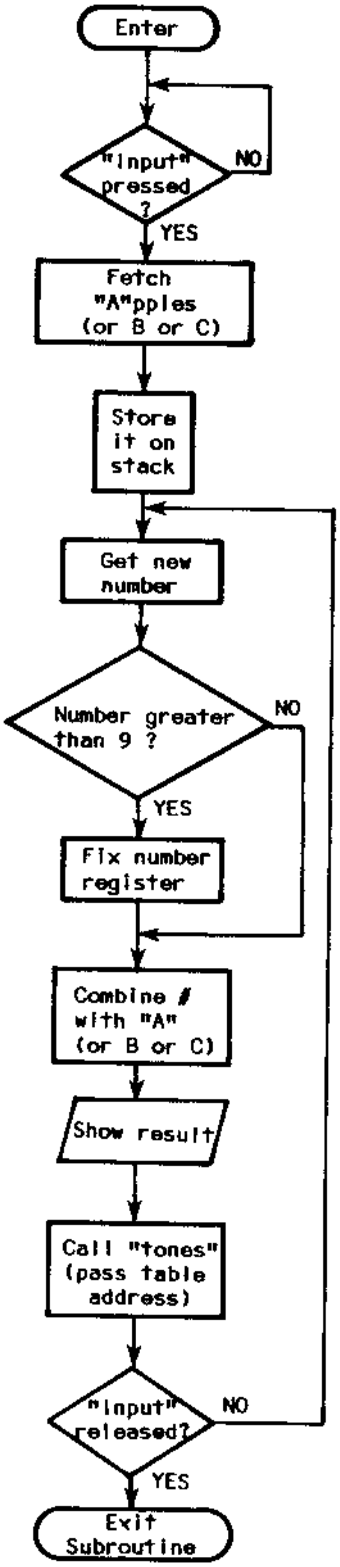
```
ADDR CODE                      ADDR CODE
0030 6B 01 42                  006C 6B 01 83
0033 55 01 52                  006F 6B 01 41
0036 50 01 57                  0072 55 01 A5
0039 47 02 27                  0075 47 01 C5
003C 6B 01 41                  0078 47 01 62
003F 55 01 52                  007B 50 02 06
0042 50 01 57                  007E 55 01 52
0045 47 02 27                  0081 50 01 57
0048 6B 01 41                  0084 47 01 C5
004B 55 01 52                  0087 55 01 A5
004E 50 01 57                  008A 6B 01 83
0051 47 01 C5                  008D 5F 01 93
0054 55 01 A5                  0090 6B 01 C4
0057 6B 01 83                  0093 00 01 1B
005A 55 01 A5                  0096 6B 01 41
005D 5F 01 D0
0060 55 01 52                  *0099-00FF
0063 55 01 52                       00 00 00
0066 5F 01 4A
0069 6B 01 4A
```

*This is done so unwanted tones will not be generated.

Sub. "tones"



Sub. "spin"



```
ADDR CODE    LABEL   OPCODE OPERAND COMMENT
0000 F8 00           LDI 00         Set hi byte of
0002 B2              PHI R2         registers
0003 B3              PHI R3
0004 B4              PHI R4
0005 B5              PHI R5
0006 B6              PHI R6
0007 F8 FF           LDI FF         Set stack pointer
0009 A2              PLO R2
000A E2              SEX R2
000B F8 0F           LDI 0F         Set main program
000D A3              PLO R3         counter
000E D3              SEP R3
000F F8 98           LDI 98         Set "Spin" program
0011 A4              PLO R4         counter
0012 F8 73           LDI 73         Set "Tones" pro-
0014 A5              PLO R5         gram counter
0015 F8 03   12      LDI 03         Set initial score
0017 AB              PLO RB
0018 8B      13      GLO RB         Show the score
0019 52              STR R2
001A 64              OUT 4
001B 22              DEC R2
001C D4              SEP R4         Call "Spin"
001D A0                             "A"pples data
001E 8A              GLO RA         Save 1'st number
001F 38              SKP            (Skip over mon-
0020 38                              itor's stack)
0021 AC              PLO RC
0022 FB 03           XORI 03        Add 1 to score if
0024 3A 27           BNZ to 1       1st guess = 3
0026 1B              INC RB
0027 D4      1       SEP R4         Call "Spin"
0028 B0                             "B"ananas data
0029 8A              GLO RA         Save 2nd guess
002A BC              PHI RC
002B FB 07           XORI 07        Add 1 to score if
002D 3A 30           BNZ to 2       2nd guess = 7
002F 1B              INC RB
0030 D4      2       SEP R4         Call "Spin"
0031 C0                             "C"herries data
0032 8A              GLO RA         Save 3rd guess
0033 52              STR R2
0034 FB 09           XORI 09        Add 1 to score
0036 3A 39           BNZ to 3       if 3rd guess=9
0038 1B              INC RB
0039 8C      3       GLO RC         Add 1 to score
003A F3              XOR            if 1st guess=3rd
003B 3A 3E           BNZ to 4
003D 1B              INC RB
003E 9C      4       GHI RC         Add 1 to score if
003F F3              XOR            2nd guess = 3rd
0040 3A 43           BNZ to 5
0042 1B              INC RB
0043 96      5       GHI RC         Add 1 to score if
0044 52              STR R2         1st guess = 2nd
0045 8C              GLO RC
0046 F3              XOR
0047 3A 4A           BNZ to 6
0049 1B              INC RB
004A 2B      6       DEC RB         Subtract 1 from
                                    score(can't play
                                    free)
004B 3F 4B   7       BN4 to 7       Wait for Input to
                                    be pressed &
                                    released
004D 37 4D   8       B4 to 8
004F 8B              GLO RB         if score=0, show
                                    FFooowy and play
0050 3A 61           BNZ to 9       "Lose" tune
0052 F8 FF           LDI FF
0054 52              STR R2
0055 64              OUT 4
0056 22              DEC R2
```

```
ADDR CODE    LABEL   OPCODE OPERAND COMMENT
0057 79              MARK
0058 D5              SEP R5
0059 B8                             (address of "Lose"
                                     tune)
005A 22      14      DEC R2
005B 3F 5B   10      BN4 to 10      Wait for input to
                                    be pressed &
                                    released
005D 37 5D   11      B4 to 11
005F 30 15           BR to 12       Go start a new
                                    game
0061 FF 0A   9       SMI 0A         If score less than
                                    10, continue game
0063 3B 18           BL to 13
0065 F8 BB           LDI BB         Score is 10 or
                                    more, so show BB
0067 52              STR R2         (Beat the Bank),
0068 64              OUT 4          play "Win" tune,
                                    and go start a new
0069 22              DEC R2         game
006A 79              MARK
006B D5              SEP R5
006C C2                             (address of "Win"
                                     tune)
006D 30 5A           BR to 14
```

Subroutine "Tones"

```
ADDR CODE    LABEL   OPCODE OPERAND COMMENTS
006F 7A      Exit    REQ            Insure Q is off
0070 E2              SEX R2         Return from sub-
                                    routine
0071 12              INC R2
0072 70              RET
0073 72      Enter   LDXA           Set Table Pointer
0074 A6              PLO R6
0075 46              LDA R6         Fetch Tempo
0076 B7              PHI R7
0077 46      18      LDA R6         Fetch Duration
0078 C6              LSNZ           Return from sub.
                                    if duration = 0
0079 30 6F           BR to Exit
007B A8              PLO R8         Else set duration
007C 46              LDA R6         Fetch and Set
007D A9              PLO R9         pitch
007E 97      17      GHI R7         Set Tempo
007F A7              PLO R7
0080 89      16      GLO R9         Skip next step if
                                    pitch = rest
0081 FC 40           ADI 40
0083 CF              LSDF
0084 CD              LSQ            Change state of Q
0085 7B              SEQ
0086 38              SKP
0087 7A              REQ
0088 89              GLO R9         Do pitch delay
0089 FF 01   15      SMI 01
008B 3A 89           BNZ to 15
008D 27              DEC R7         Do tempo delay
008E 87              GLO R7
008F 3A 80           BNZ to 16
0091 28              DEC R8         Do duration delay
0092 88              GLO R8
0093 3A 7E           BNZ to 17
0095 30 77           BR to 18       Go do next note
```

```
                    Subroutine "Spin"
ADDR CODE      LABEL   OPCODE OPERAND COMMENTS
0097 D3        Exit    SEP R3         Return from sub-
                                      routine
0098 3F 98     Enter   BN4 to Enter   Wait for input to
                                      be pressed
009A 43                LDA R3         Fetch "A"pples (or
                                      B or C) and save
009B 52                STR R2         on stack
009C 1A        20      INC RA         Fetch new number
009D 8A                GLO RA
009E FD 09             SDI 09         If greater than 9,
                                      fix it
00A0 33 A5             BGE to 19
00A2 F8 00             LDI 00
00A4 AA                PLO RA
00A5 8A        19      GLO RA         Combine it with
                                      "A" (or B or C)
00A6 F1                OR
00A7 22                DEC R2         Show result
00A8 52                STR R2
00A9 64                OUT 4
00AA 22                DEC R2
00AB 79                MARK           Call sub. "Tones"
00AC D5                SEP R5
00AD B2                               (address of "Spin"
                                       tune)
00AE 37 9C             B4 to 20       If input not rel-
                                      eased, continue
00B0 30 97             BR to Exit     Else return from
                                      subroutine

                    Table of Tones
  Spin Tones
00B2 01                               Tempo
00B3 08 27                            Duration and Pitch
00B5 14 C0                            Duration and Pitch
00B7 00                               End of sequence

                 Registers Used:

      P=3                       5=Call Tones
      X=2                       6=Used
      2=SP                      A.0=First number
      3=PC                      B.0=Score
      4=Call Spin               C=numbers
```

```
  Lose Tones

ADDR CODE    LABEL    OPCODE OPERAND COMMENTS
00B8 08                              Tempo
00B9 31 1F                           Duration and Pitch
00BB 29 27                           Duration and Pitch
00BD 20 33                           Duration and Pitch
00BF 62 47                           Duration and Pitch
00C1 00                              End of sequence

  Win Tones

ADDR CODE    LABEL    OPCODE OPERAND COMMENTS
00C2 08                              Tempo
00C3 1E 17                           Duration and Pitch
00C5 49 12                           Duration and Pitch
00C7 1E 17                           Duration and Pitch
00C9.18 1F                           Duration and Pitch
00CB 15 24                           Duration and Pitch
00CD 18 1F                           Duration and Pitch
00CF 52 1B                           Duration and Pitch
00D1 49 1F                           Duration and Pitch
00D3 00                              End of Sequence

   0000 F800 B2B3 B4B5 B6F8 FFA2 E2F8 0FA3 D3F8
   0010 98A4 F873 A5F8 03AB 8B52 6422 D4A0 8A38
   0020 38AC FB03 3A27 1BD4 B08A BCFB 073A 301B
   0030 D4C0 8A52 FB09 3A39 1B8C F33A 3E1B 9CF3
   0040 3A43 1B96 528C F33A 4A1B 2B3F 4B37 4D8B
   0050 3A61 F8FF 5264 2279 D5B8 223F 5B37 5D30
   0060 15FF 0A3B 18F8 B852 6422 79D5 C230 5A7A
   0070 E212 7072 A646 B746 C630 6FA8 46A9 97A7
   0080 89FC 40CF CD7B 387A 89FF 013A 8927 873A
   0090 8028 883A 7E30 77D3 3F98 4352 1A8A FD09
   00A0 33A5 F800 AA8A F122 5264 2279 D5B2 379C
   00B0 3097 0108 2714 C000 0831 1F29 2720 3362
   00C0 4700 081E 1749 121E 1718 1F15 2418 1F52
   00D0 1B49 1F00

               I/O Table:

   Q-Line for sound
   EF4 for "I"
   IN4 for Hex keyboard
```

ESP

by
Jess Hillman

Many QUESTDATA readers who want an ESP game to test their powers of paranormal preception without the necessity of a second player may find this program entertaining. Unlike the ESP test of QUESTDATA #12, the only players here are you and your Elf.

The program will run on basic or expanded Elf systems. I loaded the program for debugging purposes at Hex 0000, then moved it without modification to the page of memory beginning at Hex 0F00 and it worked fine.

Locations 000C-0029 provide the program key. Register F is continuously incremented until you push the input button. At that time the Elf will "guess" its number by retrieving Reg. F.0 and shifting the byte right six places, thus "selecting" a number from 0-3 (varying the number of shift instructions will, of course, change the size of the highest number chosen) and storing it in Reg. E.0. You then select the number you think Elf chose and enter it as a byte (00 to 03), and press and release "I".

If your selection matches Elf's, the Q light comes on and "AA" is displayed. Press the input button again and the current number of guesses is displayed and the Elf is ready to "select" its next secret number.

If your number fails to match, Elf will display "EE" and then wait for you to press the input key twice to go back and get another number. Then enter your next guess.

At the end of each run of ten numbers, the Elf will display "10" and wait for the input button to be pressed. Once pressed, it displays how may matches you've made out of the last ten tries and starts all over again.

Register assignments are simple: Register C keeps up with the number of matches you make; Register D is used to tally the total number of turns taken; Register E stores the Elf's number; and Register F is used as part of the pseudo-random number generating mechanism set up by the code in locations 000C—0018.

Do you have ESP? Blind luck should get you an average of 2 or 3 matches out of each run of 10. If you get any more than that, buy yourself a turban and start reading fortunes for a fee.

```
Registers Used:
     X=4
     P=0
     4=Stack Pointer
     C=Work Space
     D=Number of Guesses
     E=Work Space

ADDR CODE                               COMMENT
0000 E4 F8 A0 A4                        Register 4.0 is
                                        stack pointer
0004 F8 00 AC BC                        Initialize working
                                        registers
0008 AD BD AE BE
000C 1F                                 Increment Reg. F
                                        until
000D 3F 0C 37 0F                        INPUT pressed
0011 8F                                 Get lo byte Reg. F
0012 F6 F6 F6 F6                        Shift it left six
0016 F6 F6                              places for number
                                        between 0-3
0018 AE                                 Store it in Reg.
                                        E.0.
0019 F8 00 54 64                        Display "00" as
                                        ready signal
001D 3F 1D 37 1F                        Is INPUT pressed?
0021 6C 54                              Get byte and store
                                        it
0023 8E F3                              Retrieve Elf's
                                        "random" # and XOR
                                        it with player's
                                        number
0025 C6                                 Is D greater than
                                        0? Then skip..
0026 30 30                              If D is 0, then
                                        jump for match
0028 30 50                              If D not 0, go to
                                        Loc. 0050
0030 1C                                 Increment Reg. C
                                        and tell player
0031 F8 AA 54 64                        He got one right..
0035 7B                                 Turn on "Q", too..
0036 3F 36 37 38                        Wait for INPUT to
                                        be pressed, then
003A 7A                                 turn off "Q"
003B 1D 8D 54 64                        Increment turns
                                        Register and shows
                                        it
003F FB 0A                              Is it 10 Turns
                                        yet?
0041 CE                                 If D is 0, then
                                        skip
0042 30 0C                              If D not 0, go for
                                        another turn
```

```
ADDR CODE                               COMMENT
0044 30 5F                              Else go to end run
                                        routine
0050 1D                                 Increment turns
                                        register
0051 F8 EE 54 64                        Show that guess
                                        was wrong then
0055 3F 55 37 57                        wait for INPUT to
                                        be pressed
0059 8D                                 Get total number
                                        of turns
005A FB 0A                              Is it "10" turns
005C CE                                 If so, skip
005D 30 0C                              If not, go for
                                        another turn
005F F8 10 54 64                        Else, load "10"
                                        and show it
0063 3F 63 37 65                        Has input been
                                        pressed?
0067 8C 54 64                           Now show total
                                        matches out of 10
006A 30 00                              Then start all
                                        over again
```

```
0000 E4F8 A0A4 F800 ACBC ADBD AEBE 1F3F 0C37
0010 0F8F F6F6 F6F6 F6F6 AEF8 0054 643F 1D37
0020 1F6C 548E F3C6 3030 3050 0000 0000 0000
0030 1CF8 AA54 647B 3F36 3738 7A1D 8D54 64F8
0040 0ACE 300C 305F 0000 0000 0000 0000 0000
0050 1DF8 EE54 643F 5537 578D F80A CE30 0CF8
0060 1054 643F 6337 658C 5464 3000
```

QUESTDATA
P.O. Box 4430
Santa Clara, CA 95054

Publisher ....Quest Electronics
Editor ....Paul Messinger
Assistant to Editor ...Jeanette Johnson
Associate Editor ....Allan Armstrong
Contributing Editors   Ron Cenker
                                  Van Baker
Art and Graphics ....Holly Olson
Proof Reading ....Judy Pitkin
Production ....John Larimer
Circulation ....Sue Orr

The contents of this publication are copyright and shall not be reproduced without permission of QUESTDATA. Permission is granted to quote short sections of articles when used in reviews of this publication. QUESTDATA welcomes contributions from its readers. Manuscripts will be returned only when accompanied by a self addressed stamped envelope. Articles or programs submitted will appear with the authors name unless the contributor wishes otherwise. Payment is at the rate of $15 per published page. QUESTDATA exists for the purpose of exchanging information about the RCA 1802 microcomputer.

PSEUDORANDOM NUMBER GENERATOR WITH DIGITAL SOUND

by
Lester Hands

A random number generator can be a pretty tricky programmer's problem when outside signals cannot be relied upon (such as the pressing of an input switch), when numbers are needed in a hurry, or when several thousand numbers are needed in a few milliseconds. A pseudo-random number generator can come pretty close to fulfilling such exotic needs. While it does not produce truly random numbers, it can spew out a string of over 32,000 15-bit numbers that are non-repetitive. The following routine makes use of an idea from Don Lancaster's book, "CMOS Cookbook". Basically, it is a 15 stage shift register with the output of the last two registers exclusive-ored and fed back to the first stage. The program could easily be modified to a 22 stage shift register with feedback provided in the same way. Such a generator would have a sequence length of over 4 million!

This pseudorandom number generator (PNG) has the advantage of simplicity: the 15 stage shift register version requires one scratchpad register, shares the stack of the main program, and can be written in only 12 bytes. The only restriction is that the inital contents of the scratchpad register must be non-zero.

Assuming that R(2) is non-zero, and that a stack (with R(1) as pointer) has been already set up, the program looks like this:

```
92 FE 51 FE  Last 2 bits X-ORed together
F3 FE        shifted into DF
82 7E A2     DF shifted into 1st 8 bits
92 7E B2     Carry bit shifted into last
             7 bits
```

R(2) contains the pseudorandom number.

One obvious use of a PRG is in computer music—the possibilities are endless. To facilitate rapid writing of programs, I have written four subroutines that carry out the functions of register initialization, PNG, note length calculation (essentially a division routine), and tone production.

A basic program which produces a series of random tones of equal length is:

```
00 30 90
02 DA DB DC
05 30 02
```

The purpose of the note length calculator is to tell the tone producer the number of cycles to run, so that each tone is of the same length.

So that you can write your own modifications, here is a summary of the scratchpad registers used:

0 main program
1 stack pointer
2 shift register for PNG
3 duplicates 2
4 tone length (calculated in note length calculator)
5 used as a time delay counter, refreshed from 3 (used in the tone producer routine)
A used to call the PNG
B used to call the note length calculator
C used to call the tone producer

Here are some ideas for new variations. To increase the tone length, shift left the contents of R(4): 84 7E A4 94 7E B4.

The range of tones may be restricted with an AND immediate: 83 FA 0F A3 ("0F" may be any value).

The note length calculator routine may be used to calculate new pitches from the PNG:

```
00 30 90
02 DA DB
04 84 A3 DB
07 DC
08 30 02
```

(This one was for those who thought the original program was prejudiced in favor of the lower-pitched tones.)

Here's how to produce an unsteady tone: a steady tone routine looks like this—30 90 F8 40 A3 F8 01 A4 DC 30 02. We have bypassed the A & B routines and have specified fixed values in their place. Now if we use the PNG to modify the specified note value by changing only the least significant bit, the result is a note that varies slightly in pitch in a random fashion: 30 90 DA 83 F6 F8 20 7E A3 F8 01 A4 DC 30 02.

The following program sounds like the 1802 has laryngitis. You might want to figure out how it works--it makes use of several modifications already mentioned.

```
00 30 90
02 DA 83 F6 87 7E A3 F8 02 A4
0B DC
0C 26 86 3A 02
10 DA 83 FA 1E A7 A3
16 DB 84 7E A6 94 7E B6
1D 30 02
```

White noise can be made by changing Q according to the value of the least significant bit of R(3): 30 90 DA 83 F6 CF 7A 38 7B 30 02. If you're into novel sound effects, here's a realistic "choo-choo train": 30 90 F8 FF AF DA 83 F6 CF 7A 38 7B 2F 8F 3A 05 F8 0A BF 2F 9F 3A 13 30 02.

The frequency range of the white noise generator can be reduced by inserting a time delay before a new number is calculated. In this program, depressing the input switch (EF4) sounds like a low-pass filter has been added: 30 90 DA 83 F6 CF 7A 38 7B 3F 02 C4 C4 C4 C4 C4 C4 30 02. The volume can be decreased by increasing the time the Q is off: 30 90 DA F8 FF AF 83 7E CF 7A 38 7B 3F 02 7A 30 02.

Here's some final food for thought: this one produces a snake-like tone of slightly varying pitch and timbre: 30 90 DA F8 FF AF 83 7E CF 7A 38 7B A3 2F 8F 3A 06 30 02.

As you can see, the sound effects possible with a purely digital sound source (only one line) are endless. Applications are just as numerous; music, sound effects, and voice production are just a few.

```
Registers Used:
P=0
X=1
0=PC
1=Stack Pointer
2=Random #
3=Work
4=Used
5=Work
A=Call RND
B=Call LEN
C=Call
```

```
***********INITIALIZATION OF  REGISTERS**********
ADDR CODE    LABEL   OPCODE OPERAND COMMENTS
0090 F8 00   INIT:   LDI     00     Initialize high
                                    bytes of
0092 B1              PHI 1          R1
0093 BA              PHI A          RA
0094 BB              PHI B          RB
0095 BC              PHI C          RC
0096 F8 FF           LDI     FF     Initialize low
                                    byte of
0098 A1              PLO 1          R1
0099 E1              SEX 1          Make it the stack
                                    pointer
009A F8 AA           LDI     AA     Initialize low
                                    byte of
009C AA              PLO A          RA so it points
                                    to RND.
009D F8 BA           LDI     BA
009F AB              PLO P
00A0 F8              PHI 8
00A1 DB              SEP B          Call Note length
                                    calculator
00A2 AC              PLO C
00A3 82              GLO 2
00A4 F9 01           ORI     01
00A6 A2              PLO 2
00A7 30 02           BR      02     Return to program.

*********PSEUDORANDOM NUMBER  GENERATOR**********
ADDR CODE    LABEL   OPCODE OPERAND COMMENT
00A9 D0      EXIT 1: SEP 0          Return
00AA 92      RND:    GHI 2
00AB FE              SHL
00AC 51              STR 1
00AD FE              SHL
00AE F3              XOR
00AF FE              SHL
00B0 82              GLO 2
00B1 7E              SHL C
00B2 A2              PLO 2
00B3 A3              PLO 3
00B4 92              GHI 2
00B5 7E              SHLC
00B6 B2              PHI 2
00B7 30 A9           BR     EXIT 1:

**************NOTE LENGTH  CALCULATOR************
ADDR CODE    LABEL   OPCODE OPERAND COMMENTS
00B9 D0      EXIT2:  SEP 0          Return
00BA 83      LENCAL: GLO 3          Prevent division
                                    by zero
00BB 32 D2           BZ     FIXIT
00BD F8 FF           LDI    FF      Prepare scratchpad
00BF 51              STR 1
00C0 F8 00           LDI    00      Initialize
00C2 A4              PLO 4          register
00C3 B4              PHI 4          4.
00C4 83      LOOP1:  GLO 3          Division routine.
00C5 F5              SD
00C6 51              STR 1
00C7 14              INC 4
00C8 33 C4           BDF    LOOP1
00CA 84              GLO 4          Multiply by 2.
00CB FE              SHL
00CC A4              PLO 4          (Shift R4 left)
00CD 94              GHI 4
00CE 7E              SHLC
00CF B4              PHI 4
00D0 30 B9           BR     EXIT2   Finished
00D2 F8 FF   FIXIT:  LDI    FF
00D4 A4              PLO 4
00D5 F8 01           LDI    01
00D7 B4              PHI 4
00D8 30 B9           BR     EXIT2   Finished
```

```
*****************TONE PRODUCER******************
ADDR CODE     LABEL    OPCODE OPERAND COMMENTS
00DA D0       EXIT3:   SEP 0          Return
00DB 83       TONES:   GLO 3
00DC A5                PLO 5
00DD 3A E1             BNZ    LOOP2
00DF 30 EB             BR     CHECK
00E1 85       LOOP2:   GLO 5          Kill time.
00E2 25                DEC 5
00E3 3A E1             BNZ    LOOP 2
00E5 C5                LSNQ           Complement Q.
00E6 7A                REQ
00E7 38                SKP
00E8 7B                SEQ
00E9 31 DB             BQ     TONES   Continue if Q=1
00EB 24       CHECK    DEC 4
00EC 94                GHI 4
00ED 3A DB             BNZ    TONES   Continue if
00EF 84                GLO 4          R4 not = 00
00F0 3A DB             BNZ    TONES
00F2 30 DA             BR     EXIT 3  Finished
```

```
0090 F800 B18A BBBC F8FF A1E1 F8AA AAF8 BAAB
00A0 F8DB AC82 F901 A230 02D0 92FE 51FE F3FE
00B0 B27E A2A3 927E B230 A9D0 8332 D2F8 FF51
00C0 F800 A4B4 83F5 5114 33C4 84FE A494 7EB4
00D0 30B9 F8FF A4F8 01B4 30B9 D083 A53A E130
00E0 EB85 253A E1C5 7A38 7B31 DB24 943A DB84
00F0 3ADB 30DA
```

ELFWRITER

by
Richard Moffie

Elfwriter is a program that will let you write and display messages or other information on the T.V. screen with your Elf system. The displayed information can be saved on tape and then read back onto the screen for display or editing.

Elfwriter has these features:

1. The program will run on a basic Elf, as long as it has 1K of RAM and the 1861 video chip.

2. It is written for hex keyboard, but an ASCII keyboard could be used with only a few minor changes.

3. 16 characters per line are displayed with either 5 or 10 lines displayed at a time.

4. The basic program fits in 1/2K of RAM, so that all remaining memory (2 pages in a 1K system, 14 pages in 4K system) can be used for display.

5. There are some editing features: change display page up or down, erase line, erase display area, carriage return-line feed.

6. There is some unused space for adding features and special character patterns.

USING ELFWRITER

Load the program at 0000 - 01FF. When Reset, Run is pressed, Elfwriter is ready to use, with page 2 displayed (Pages 0 and 1 contain the program - Don't write onto them!). To clear the entire display area - all pages - enter "BB". To write, enter the ASCII code (20-5F) of the desired character and press Input key. Continue in this manner until you have written the desired information. At the end of a line, the program will begin writing at the beginning of the next line. When a page is full, the writing will begin at the top of the next page and the display will then show the new page.

To begin a new line, or to skip lines, enter "0D" (ASCII code for carriage return). To erase the current line, enter "EE" - a carriage return is automatic. To change the page being displayed press "0C" for the next higher page, or "DD" for the next lower page. To display messages after they are written, press Reset, Run and page 2 will be displayed.

A totally white screen means a non-existent page of memory is being displayed - go back (using DD) to a memory page that you can use. A display that does not show character patterns that you have written means you are on page 0 or page 1 (which contain the program). DO NOT TRY TO WRITE ON THESE PAGES.

If you "get lost" and don't know which page you are on, pressing Reset, Run will put you on page 2 and no information will be lost. When pages are changed, the writing begins at the beginning of the top line. To space or skip over letters already written, use "20" (ASCII space) to skip over characters.

If using the 10 line display format, you may only write on the top 5 lines displayed. Change page to write on the other 5 lines. The display format can be changed from 5 to 10 lines at any time by changing byte 00D9 as indicated in the listing.

TRANSFER LOGIC

The program is basically an expanded version of the original T.V. Typerwriter Jr. that fits in a basic 256 byte Elf, and uses much of the same logic. The program can be found in Elf of the Valley Newsletter - August 1978, or in more detail in Questdata #4.

The program stores portions of two characters in each byte. The 4 high bits store one character and the 4 low bits store the next character in sequence. Similarly, there are two characters displayed per byte across the screen to get 16 characters on a line of 8 bytes. In order to get the correct character and display it in the correct position, the Q line and E register are used as follows:

If Q=0 get high 4 bits   If Q=1 get low 4 bits
If RE=0 display on high 4 bits
If RE=1 display on low 4 bits

These are tested and cause the program to proceed to the correct sequences:

Q	RE	Jump to
0	0	016A
0	1	0165
1	0	0179
1	1	0174

The use of the above information along with the memory map should make the logic easy to understand and the program easy to modify as desired.

```
Registers Used:

X=3
P=2
0=Display Page Pointer (initially 0200)
1=Address of Video Interrupt Routine (00C3)
2=Stack pointer (00FF)
3=Main program pointer (0100)
6=Pointer to memory containing display page (00CB)
7=No. of bytes/line of characters - Used for erasing
  line (0028)
8=Counts erased bytes - for erasing display
9=No. of lines/character (0005)
A=Points to current byte in character being trans-
  ferred
B=Points to first byte in character being transferred
C=No. of bytes/line (0008)
D=Points to first byte in current display position
E=Flag for transfer logic
F=Points to current byte in current display position
```

```
ADDR CODE           COMMENT
0000 F8 00          Initialize registers
0002 A3 AE
0004 B1 B2
0006 B6 BA
0008 BB BE
000A F8 01
000C B3 F8
000E 02 B8
0010 BD BF
0012 56 F8
0014 C3 A1
0016 F8 FF
0018 A2 F8
001A CB A6
001C 30 C0
001E xx xx
0020 02 55          Character Patterns (20-47 for
0022 75 22          ASCII 20-2F, 48-6F for ASCII
0024 22 22          30-3F, 70-97 for ASCII 40-4F,
0026 00 01          98-BF for ASCII 50-5F)
0028 02 07
002A 61 22
002C 41 72
002E 00 01
0030 02 05
0032 72 70
0034 41 27
0036 07 02
0038 00 07
003A 34 20
003C 41 02
003E 20 04
0040 02 05
0042 75 20
0044 22 02
0046 40 24
0048 72 77
004A 57 77
004C 77 00
004E 10 47
0050 52 11
0052 54 41
0054 55 22
0056 27 21
0058 52 77
005A 77 71
005C 77 00
005E 40 12
```

```
0060 52 41
0062 11 51
0064 51 22
0066 27 20
0068 72 77
006A 17 71
006C 77 04
006E 10 42
0070 77 67  Fix letter B
0072 67 77
0074 57 75
0076 45 72  Fix letter 0
0078 15 54
007A 54 44
007C 52 25
007E 47 55
0080 77 64
0082 56 65
0084 72 26
0086 47 55
0088 55 54
008A 54 45
008C 52 25
008E 45 55
0090 75 67
0092 67 47
0094 57 65
0096 75 52
0098 77 77
009A 75 55
009C 55 76
009E 43 00
00A0 55 54
00A2 25 55
00A4 55 14
00A6 41 00
00A8 75 67
00AA 25 57
00AC 22 24
00AE 21 00
00B0 47 51
00B2 25 57
00B4 52 44
00B6 11 00
00B8 47 57
00BA 27 25
00BC 52 76
00BE 13 07
```

```
00C0 D3 72  Video Interrupt Routine
00C2 70 22
00C4 78 22
00C6 52 C4
00C8 C4 C4
00CA F8 02
00CC B0 F8
00CE 00 A0
00D0 80 E2
00D2 E2 20
00D4 A0 E2
00D6 20 A0
00D8 E2 20
00DA A0 3C
00DC D0 30
00DE C1 xx  Unused
00E0 xx xx
00E2 xx xx
00E4 xx xx
00E6 xx xx
00E8 xx xx
00EA xx xx
00EC xx xx
00EE xx xx
00F0 xx xx
00F2 xx xx
00F4 xx xx
00F6 xx xx
00F8 xx xx
00FA xx xx
00FC xx xx  Stack Area
00FE xx xx

0100 E2 69  Initialize and Reset Registers
0102 F8 10
0104 AD F8
0106 08 AC
0108 8D AF
010A F8 05
010C A9 F8
010E 20 AB
0110 3F 10  Keyboard Input Routine
0112 37 12
0114 6C FB  Test for special characters
0116 BB 32
0118 A0 F0
011A FB CC
011C 32 CA
011E F0 FB
```

```
0120 DD 32
0122 D0 F0
0124 FB EE
0126 32 B8
0128 F0 FB
012A 0D 3A
012C 40 8D
012E FA F0  Carriage Return/Line Feed
0130 FC 30
0132 AD 33
0134 D6 F8
0136 00 AE
0138 30 05
013A xx xx  Unused
013C xx xx
013E xx xx
0140 F0 FF  Routine to point to correct
0142 20 FF  character pattern - Converts
0144 10 B9  ASCII code to pattern location
0146 3B 4F
0148 8B FC
014A 28 AB
014C 99 30
014E 43 02
0150 FA 0F
0152 52 F0
0154 F6 52
0156 3B 5B
0158 7B 30
015A 5C 7A
015C 8B F4  Transfer logic - Selects left or
015E AA E2  right character and left or
0160 31 71  right display area, then
0162 8E 3A  transfers character to correct
0164 6A 0A  location.
0166 FA F0
0168 30 7E
016A 0A F6
016C F6 F6
016E F6 30
0170 7E 8E
0172 32 79
0174 0A FA
0176 0F 30
0178 7E 0A
017A FE FE
017C FE FE
017E EF F1
0180 5F E2
```

QUESTDATA
P.O. Box 4430
Santa Clara, CA 95054

A 12 issue subscription to QUESTDATA, the publication devoted entirely to the COSMAC 1802 is $12.
(Add $6.00 for airmail postage to all foreign countries except Canada and Mexico.)
Your comments are always welcome and appreciated. We want to be your 1802's best friend.

Payment
□ Check or Money Order Enclosed
Made payable to Quest Electronics
□ Master Charge No. ______
□ Bank Americard No. ______
□ Visa Card No. ______
Expiration Date: ______

Signature ______

□ Renewal □ New Subscription

NAME ______

ADDRESS ______

CITY STATE ZIP ______

```
0182 8F FC
0184 08 AF
0186 8A FC
0188 08 AA
018A 29 89
018C 3A 60
018E 8E FB
0190 01 AE
0192 3A 95
0194 1D 3A
0196 08 2C
0198 8C 32
019A B2 30
019C 08 xx
019E xx xx
01A0 F8 00          Blank screen subroutine
01A2 A8 F8
01A4 00 58
01A6 18 98
01A8 FB 10
01AA 3A A3
01AC F8 02
01AE B8 30
01B0 02 E2          Automatic linefeed at end of
01B2 8D FC          line
01B4 28 AD
01B6 30 33
01B8 F8 28          Erase line subroutine
01BA A7 8D
01BC FA F0
01BE AD A8
01C0 F8 00
01C2 58 18
01C4 27 87
01C6 3A C0
01C8 30 35
01CA 06 FC          Change page (higher)
01CC 01 56
01CE 30 DA
01D0 06 FF          Change page (lower)
01D2 01 56
01D4 30 DA
01D6 9D FC          Automatic change page when full
01D8 01 56
01DA BD BF
01DC B8 F8
01DE 00 AE
01E0 30 02
01E2 xx xx          Unused
01E4 xx xx
01E6 xx xx
01E8 xx xx
01EA xx xx
01EC xx xx
01EE xx xx
01F0 F8 00          Super Monitor entry
01F2 B0
01F3 A0
01F4 E3
01F5 70
01F6 00
```

Notes:

1. Locations marked xx are unused (except for stack-00FB - 00FF) and can be used for additions to program.
2. For 5 line display, leave 00D9 as 20. For a 10 line display, change it to 80.
3. Location 01A9 should contain the number of pages of RAM in hex. It is set for 4K in above listing. For 1K, set to 04, etc.

```
0000 F800 A3AE B1B2 B6BA BBBE F801 B3F8 02B8
0010 BDBF 56F8 C3A1 F8FF A2F8 CBA6 30C0 0000
0020 0255 7522 2222 0001 0207 6122 4172 0001
0030 0205 7270 4127 0702 0007 3420 4102 2004
0040 0205 7520 2202 4024 7277 5777 7700 1047
0050 5211 5441 5522 2721 5277 7771 7700 4012
0060 5241 1151 5122 2720 7277 1771 7704 1042
0070 7767 6777 5775 4572 1554 5444 5225 4755
0080 7764 5665 7226 4755 5554 5445 5225 4555
0090 7567 6747 5765 7552 7777 7555 5576 4300
00A0 5554 2555 5514 4100 7567 2557 2224 2100
00B0 4751 2557 5244 1100 4757 2725 5276 1307
00C0 D372 7022 7822 52C4 C4C4 F802 B0F8 00A0
00D0 80E2 E220 A0E2 20A0 E220 A03C D030 C100
00E0 0000 0000 0000 0000 0000 0000 0000 0000
00F0 0000 0000 0000 0000 0000 0002 0020 23BB
0100 E269 F810 ADF8 08AC 8DAF F805 A9F8 20AB
0110 3F10 3712 6CFB BB32 A0F0 FBCC 32CA F0FB
0120 DD32 D0F0 FBEE 32B8 F0FB 0D3A 408D FAF0
0130 FC30 AD33 D6F8 00AE 3005 0000 0000 0000
0140 F0FF 20FF 10B9 3B4F 8BFC 28AB 9930 4302
0150 FA0F 52F0 F652 3B5B 7B30 5C7A 8BF4 AAE2
0160 3171 8E3A 6A0A FAF0 307E 0AF6 F6F6 F630
0170 7E8E 3279 0AFA 0F30 7E0A FEFE FEFE EFF1
0180 5FE2 8FFC 08AF 8AFC 08AA 2989 3A60 8EFB
0190 01AE 3A95 1D3A 082C 8C32 B230 0800 0000
01A0 F800 A8F8 0058 1898 FB10 3AA3 F802 B830
01B0 02E2 8DFC 28AD 3033 F828 A78D FAF0 ADA8
01C0 F800 5818 2787 3AC0 3035 06FC 0156 30DA
01D0 06FF 0156 30DA 9DFC 0156 BDBF B8F8 00AE
01E0 3002 0000 0000 0000 0000 0000 0000 0000
01F0 F800 B0A0 E370 00
```

COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC

19 QUESTDATA
P.O. Box 4430
Santa Clara, CA 95054

ADDRESS CORRECTION REQUESTED

[redacted]