Questdata

Volume 3 Issue # 2 ©

INFESTATION

by
Phillip B. Liescheski III

INFESTATION is a computer modeling algorithm similar to LIFE. This program was inspired by an article written by Frank C. Hoppensteadt. From this article, the basic information and rules of INFESTATION were obtained. Unlike LIFE, it does not model growth. It demonstrates the propagation of disease in a forest or some other form of vegetation. Each tree can exist in one of three states. These states are green, infested, and defoliated. The term tree is used loosely as a unit of vegetation or an area containing some form of vegetation.

There are four basic rules
under which INFESTATION operates. These four rules are:

1) A green tree remains green as long as it is not infested.

2) An infested tree becomes defoliated in the next generation.

3) A defoliated tree becomes green in the next generation.

4) An infested tree will infect its four non diagonal, green neighbors in the next generation.

These rules determine the manner in which the disease propagates and thus dictates the type of patterns for each generation.

Since each tree can exist in any of three states, the SUPER COLOR OR GREMILIN video board with its color graphics ability is used to display the forest. A green or healthy tree is represented by a green square on the video grid. An infested tree is represented by a red square, while a defoliated tree is displayed as a yellow square. These colors are chosen since they subjectively represent these states. Also, with this color combination, the three states can be differentiated on a black and white television receiver.

In order to use INFESTATION, one must bootup the Super Elf with the Super BASIC interpreter. The Gremlin video board must be set for the Alphanumeric/Semigraphics-4 mode (The SUPER COLOR board can be set by adding a line of code to the beginning of the program to POKE the mode into the mode latch on the board). Next the BASIC program is typed in and it is executed. The pro gram will first print its identification and a symbol key. afterwards, it will request the ini tial state for each tree in the forest. Each tree is assigned unique row and column numbers. Each position is displayed and the state is then requested. One must enter G for green, I for in fested or D for defoliated. If another character is accidentally entered, an error message is displayed. The row and column numbers for this tree are reprinted and the request is made again until the proper character has been entered. After the initial pattern has been entered, the screen is cleared, and the initial pattern of the forest is displayed. The state of the forest for the next generation is determined according to the four rules. Afterwards, the forest of the next generation is displayed. For a ten by ten (10*10) forest, the time period for each gener ation is about a half minute.

The operation of the program is simple. Since the program is written in BASIC with several remark statements, a detailed discussion of its operation will not be made here. Basially, the information of state for each tree is contained in matrix P. As the initial pattern is entered, the Gremlin color code is assigned to each matrix element which represents a tree. A color code represents a specific state. In other words, an infested tree in the second row and third column of the forest grid is represented by the red color code #BF in the matrix element P(3,2). During the determination of the next generation, the pattern is temporarily stored in matrix G. Afterwards, the contents of the matrix G is transferred to matrix P. The contents of the matrix P is displayed by poking the code value of each element in the appropriate location of the video memory. The video screen is initially cleared by loading the black color code #80 into each location of the video memory between the addresses E000 and E200. Finally, this program will operate until the operator pushes the escape key or pulls the plug.

One thing which will be mentioned in detail is the dimension declaration statement. In almost all BASIC interpreters ranging from DEC to IBM, the size of each declared array must be expressed by a number, that is: DIM P(6,6). But, in Super BASIC, the size can be expressed by an arithmetic expression, that is: DIM P(N,N). This feature allows for greater flexibility. Finally, if the size of the forest grid is to be changed, one merely assigns the variable N a new value which is indicative of the new size. This change will not disrupt the operation of the program; however, the calculation of the video memory coordinates, X and Y, in lines 260 and 270 should be taken into consideration, if severe problems should result.

INFESTATION has been written in Quest Super BASIC V3.0. The algorithm is simple. The task of writing it in machine code or Tiny Basic should pose no problems. INFESTATION demonstrates the principles behind computer population simulation and also produces very interesting and colorful patterns.

REFERENCE———— Hoppensteadt, Frank C. "Mathematical Aspects of Population Biology.
" In MATHEMATICS TODAY: TWELVE INFORMAL ESSAYS, pp. 297–320.
Edited by Lynn Arthur Steen. New York: Springer-Verlag, 1978.

```
1 REM **
2 REM **INFESTATION
3 REM **
4 REM ** PHILLIP B. LIESCHESKI III **
5 REM **
6 DEFINT Z
7 REM *FOREST GRID SIZE
8 N=10
9 DIM P(N,N),G(N,N)
10 REM **
11 REM **REQUEST INITIAL FOREST PATTERN
20 PR "INFESTATION MODELING FOR A FOREST":PR
25 PR "SYMBOL TABLE:"
30 PR "G-GREEN=LIVING TREE"
40 PR "I-RED=INFESTED TREE"
50 PR "D-YELLOW=DEFOLIATED TREE":PR
60 PR "ENTER FOREST STATE PATTERN:"
70 PR "ROW","COLUMN","STATE"
80 FOR R=1 TO N
90 FOR C=1 TO N
100 PR. R,C,
109 REM *REQUEST EACH TREE STATE
110 INPUT A$
120 IF A$="I" P(C,R)=#BF:GOTO 160
130 IF A$="D" P(C,R)=#9F:GOTO 160
140 IF A$="G" P(C,R)=#8F:GOTO 160
150 PR "WRONG CHOICE":GOTO 100
160 NEXT C
170 NEXT R
180 REM **
181 REM **CLEAR SCREEN
190 FOR I=@E000 TO @E200
199 REM *LOAD VIDEO MEMORY W/ BLACK COLOR CODE
200 POKE(I,#80)
210 NEXT I
221 REM **DISPLAY FOREST PATTERN
230 FOR R=1 TO N
240 FOR C=1 TO N
249 REM *REFRESH TEMPORARY FOREST MATRIX
250 G(C,R)=#8F
259 REM *CALCULATE VIDEO MEMORY ADDRESS
260 X=C+11
270 Y=@E080+(R-1)*32
279 REM *PUSH COLOR CODE INTO VIDEO MEMORY
280 POKE(X+Y,P(C,R))
290 NEXT C
300 NEXT R
310 REM **
311 REM **DETERMINE PATTERN FOR NEXT GENERATION
320 FOR R=1 TO N
330 FOR C=1 TO N
339 REM *IF DEFOLIATED, MAKE GREEN
340 IF P(C,R)=#9F G(C,R)=#8F:GOTO 460
349 REM *IF NOT INFESTED, LEAVE ALONE
350 IF P(C,R)<>#BF GOTO 460
359 REM *IF INFESTED, DEFOLIATE
360 G(C,R)=#9F
370 REM *INFECT FOUR GREEN NEIGHBORS
380 IF C<2 GOTO 420
390 IF C>N-1 GOTO 420
400 IF P(C+1,R)=#8F G(C+1,R)=#BF
410 IF P(C-1,R)=#8F G(C-1,R)=#BF
420 IF R<2 GOTO 460
430 IF R>N-1 GOTO 460
440 IF P(C,R+1)=#8F G(C,R+1)=#BF
450 IF P(C,R-1)=#8F G(C,R-1)=#BF
460 NEXT C
470 NEXT R
480 REM **
481 REM **TRANSFER NEW GENERATION PATTERN
490 FOR R=1 TO N
500 FOR C=1 TO N
510 P(C,R)=G(C,R)
520 NEXT C
530 NEXT R
540 GOTO 220
999 END
```

Q*BUG

In a future column, we will be adding some two character "Shorthand" commands to the Super command table. Until we learn and memorize all of the new commands, it might be helpful to have a "HELP" routine which will list all of the commands in the command table. What follows is a machine language routine to do just that.

To use this routine, we will make a change in the command table on page 0500 and the execution table on page 0700. At this point, the changes will be temporary and we will make them permanent when the "Shorthand" commands are inserted into the command table.

A look at the command table at location 0500 thru 06D2 reveals several interesting pieces of information: Each command in the table consists of a leading control byte, the ASCII representation of the command word with bit 7 of the last byte changed to a 1, and a trailing byte which is the command token. When Super writes a program to memory, only the token byte is actually put in memory. Thus, only one byte of memory is used for each command.

As an example, the command "CLS" appears in the command table at location 0505 as:

24 43 4C D3 81

The 24 is the control character, 43 = "C", 4C = "L", and D3, which is really 53 with bit 7 set to 1, = "S". The 81 is the token byte for the command.

Incidentally, tokens A9, AC, AF and B0 are not presently used for statements in the command table. Also, the FUNCTION execution table, at location 07A0 thru 07D7, contains several unused tokens as well as six unidentified tokens.

The STATEMENT execution table starts at 0700 and runs thru 0763. Each statement command token "points" to a two byte address in the execution table that Super will vector to when running your program. Our job will be to set up a token for the "HELP" routine and install the address of the Machine Language routine in the STATEMENT execution table.

Super now includes two rather redundant commands – SFMON and FDMON. I feel it is very unlikely that anyone would have both a Floppy Disk drive AND a Stringy Floppy drive. Consequently, we will use one of these for our HELP command. Actually, we will change the command word to HELPP so that it has the same number of characters as SFMON (or FDMON) and will save us the job of realigning the command table.

At location 0573, change 53(S), 46(F), 4D(M), 4F(O) and CE(N) to 48(H), 45(E), 4C(L), 50(P), and D0(P), if you wish to drop SFMON; or, at location 057A, change 46 44 4D 4F CE to 48 45 4C 50 D0. Either way, the command "HELPP" is now in the command table.

Now, if you changed SFMON to HELPP, you must change the address for token 93 in the execution table. If FDMON was changed, the address for token 94 must be changed.

As mentioned previously, each token "points" to a two byte address in the execution table. Token 80 "points" to location 0700 and 0701 which show an address of 02F9. This is the actual routine Super will perform when a REM is encountered in a program. (Did you know that you can type a "!" for REM and Super will automatically convert it to REM ?)

Token 81 "points" to location 0702 and 0703, token 82 "points" to 0704 and 0705, etc..

Token 93 "points" to location 0726 and 0727 which contain address 0CA6 and token 94 "points" to location 0728 and 0729 which contains the address 0CA9.

Depending on which command you changed, change its address in the execution table to 0010. Thus, if you changed SFMON to HELPP, you would change location 0726 and 0727 to 0010. If you changed FDMON, change location 0728 and 0729 to 0010.

We will put the HELPP machine language routine on work page 0000. It will run from location 0010 thru 004F. The program follows:

```
ADDR.
====

0010  F8 05 B1  Set R.1 to point to location
                0501
  13  F8 01 A1   (First ASCII character in REM)
  16  88 73     Save R8.0 on stack
  18  F8 00 A8  Set line length counter (R8.0)
                to "00"
  1B  01        Get byte at location pointed to
                by R1
  1C  FF 80     Subtract 80 – Convert to probab-
                le ASCII character
  1E  3B 25     If DF = 0 (Probable ASCII)
                branch to 0025
  20  41        Get byte at location pointed to
```

```
                 by R1 - R1 + 1
 21  FF 80       Subtracr 80 - convert to probab-
                 le ASCII character
 23  30 2B       Branch to location 002B
 25  41          Get byte at location pointed to
                 by R1 - R1 + 1
 26  D4 35 8E    Call print routine - prints only
                 valid ASCII
 29  30 3C       Branch to location 003C
 2B  D4 35 8E    Call print routine
 2E  F8 20       Load "20" (space) to D register
 30  D4 35 8E    Print the space
 33  11          Increment R1 past the next token
                 character
 34  01          Get the byte at location pointed
                 to by R1
 35  FB FF       Test (XRI) for "FF" (end of
                 command table)
 37  32 4C       If test is true, branch to 004C
 39  11          Increment R1 past first control
                 character
 3A  18          Increment line length counter
                 for byte printed
 3B  18          Increment line length counter
                 for space printed
 3C  18            (Housekeeping)
 3D  88          Get R8.0 (line length counter)
 3E  FF 3D       Test for "3D" (61 characters
                 printed)
 40  3B 1B       If test not true,(less than 61)
                 branch to 001B
 42  01          Get next byte at location poin-
                 ted to by R1
 43  FF 80       Subtract 80
 45  33 20       If DF = 1, branch back to print
                 byte(continue word)
 47  D4 2F F9    Print CR/LF
 4A  30 18       Branch back to reset line length
                 counter
 4C  12 02 88    Restore R8.0
 4F  D5          Return
```

You may have to adjust the value of the byte at location 003F, up or down, to keep Super from splitting too many command words.

First, make a new master Super program tape (record at least twice). Now, you can use your HELPP command by typing "HELPP" at any time.

The next article will go into the creation of a simply "Shorthand" basic, moving the printer driver routine to a new location and opening up the command table thru location 06FF, and the final HELP routine.

DF DEMO

by

Randy Rinker

DF Demo is a program for those who would like to learn more about the 1802 processor that is the heart of your Super Elf. As the title suggests, this program shows how the DF register works.

I purposely kept this program easy to understand. What good would it do to show you how one thing works if you don't understand the rest of the program? Addresses 0000 H to 0008 H are for initialization. 0009 H to 0018 H are used to input two numbers. Push input after each number. Addresses 0019 H to 001B H subtract the first number from the second. At address 001C H is a 32 instruction that branches to a buzzer if both numbers are equal. At the heart of this program is the 3B (branch if DF=0). DF will equal 0 if a borrow occurs on subtract. This means the first number was larger. Addresses 0020 H to 0029 H display either 01 or 02 to show which number was larger and then branches back to the start. The buzzer starts at address 002A H. It is a typical routine and others like it can be found in "What the Machine is Thinking" in Questdata #2 or the Super Elf manual. You must reset and start over if the buzzer sounds.

I hope this program has given you some insight into the great Cosmac microprocessor. I am always willing to help fellow Cosmacians!

Program Listing:

```
Addr.    Data

0000     90 B2 B3                 0020     E0
0003     F8 1B A2                 0021     64 02
0006     F8 FF A3                 0023     30 00
0009     E2                       0025     E0
000A     3F 0A                    0026     64 01
000C     37 0C                    0028     30 00
000E     6C 64 22                 002A     7B
0011     E3                       002B     F8 FF
0012     3F 12                    002D     FF 01
0014     6C 64 23                 002F     3A 2D
0017     37 14                    0031     7A
0019     F0                       0032     F8 FF
001A     FF 'any' (stack)         0034     FF 01
001C     32 2A                    0036     3A 34
001E     3B 25                    0038     30 2A
```

NOTE:
If you change address 38 to 3F 2A and add 30 00 at address 3A, then the program will reset itself to address 00 00 if the input is held while the "buzzer" is sounding.

TEXT EDITOR

by

Fred Hannan

When I first agreed to write the "QBUG" column, I did not possess an easily useable Text Editor. I had dabbled with the program "Simple Line Editor" in the manual for Super Basic Version 1.4 but found it lacked several features that I wanted in an Editor. Also, at this time, I did not have a printer so any output I generated went to the CRT.

The first article I prepared was created as a very long string of "PRINT" statements in a Basic program. Although this worked after a fashion, it gave our editor, Paul Messenger, fits when he had to convert the Basic program to a form compatible with his word processing machine.

Paul asked our resident programming Guru, Ron Cenker, to prepare a simple line editor program that I could use to prepare future columns. Although this program also worked, it was even more rudimentary than the Super Basic " Simple Line Editor" program (sorry, - Ron). At this point, I decided to use the "Simple Line Editor" in spite of its faults. I made a few changes to the program to make it more adaptable to my needs but refrained from a complete rewrite due to the pressure of getting several columns written up and off to QUESTDATA.

Shortly thereafter, I made the decision to purchase a printer and started to look for a means of interfacing a printer to my ELF II. I also needed a program to print out a hard copy of the column, both for proofreading purposes and to accompany the tape when I sent it to QUESTDATA. Happily, John Ware in Fort Worth, Texas, was kind enough to provide the hardware interface details and a much modified version of the "Simple Line Editor" program which included a printer driver section.

After a bit of a struggle with the printer interface, due to a lack of electronic knowledge on my part, I had the printer up and running and merrily spewing out page after page of the column.

At this point, I decided to really dig in and make the Line Editor program useful for the preparation of the column and letter or memo writing. The accompanying program is the result of my labor - it is a greatly modified version of the original program and includes some of John Ware's modifications.

The program is menu driven and will perform the following operations:

1 - Enter new data from the keyboard
2 - Enter data from a tape
3 - List the data already entered (to the CRT)
4 - Present a proof type listing on the CRT with line #'s and prompt for the line # to be edited
5 - Prompt for the line # to be edited and perform edit
6 - Save data to tape
7 - A) Print a proof type listing with line #'s
    B) Print a draft (final) copy of:
      1) A letter
      2) A memo form
      3) A plain form (no heading)
8 - Add data to data already entered - will display last line previously entered
9 - Exit from editor
10 - Display the menu at any time

Although the program is liberally sprinkled with REM like statements explaining most of the functions, I will run thru the program with you, section by section.

Line 30, the first non-REM line, will clear the screen and initialize the variables A and A1. Line 40 establishes strings H and H1 which are used for the headings on the memo form when this is the desired output. Lines 60 thru 80 are used to set up the letterhead strings which the program will print at the top of the form. I have placed "x"'s in the program but you should put your name, etc. in these lines.

The main program starts at line 90 with a sign on message. Line 100 requests the entry of a date. This date is printed at the appropiate place on th. letter, memo, or plain form. If you do not want the date to appear on a form, merely press the RETURN (or ENTER) key. The program will assign a space to the date string Y. This is to avoid the abort of the program later on when it looks for the Y string.

Line 110 thru 150 give you the option of going immediately to the entry of new data or performing some other task. If either an uppercase or lowercase "N" is entered, the program will ask you if you wish to have the menu displayed. (All prompt reply scan be either upper or lower case letters when a letter is the required input). A C/R will default to a "Y" and the program goes into the data entry mode. Line 150 is an error trap typeline and will display an error mesage if an invalid input is attempted.

Lines 160 thru 200 give the user the option of displaying the menu or not. In the beginning, you may want to display the menu rather often until you memorize the option selection codes. A C/R will default to an "N" and the menu will not be displayed. A "Y" will display the menu, of course. The errortrap is at line 200.

Lines 210 thru 330 request the input of the option selection code and will go to the proper function upon receipt of the code. A C/R will default to option #1 - Enter New Data. The error trap for this routine is line 330.

Lines 340 thru 370 perform the data load from a tape routine. A prompt is issued requesting the page # on the tape which is to be loaded. Instructions for setting the tape deck are issued and when RETURN is pressed, loading of the data begins. There is no error trap for this routine so be careful in entering the prompt replies.

Lines 380 thru 490 represent the keyboard data entry function. The program issues two simple instructions concerning line length and the exit symbol, both of which can be changed to suit your needs. If you change the number of characters on a line, be sure to change line 440 which is the actual line length delimiter.

Lines 430 thru 450 prepare the line # counter for the first or next line of data, issues the prompt "Enter Data", sets the line length delimiter, and awaits the entry of data.

Line 460 tests for the entry of the exit symbol "™". If the symbol is received, the program goes to the exit routine atline 490.

Line 470 tests the length of the data line entered and, if more that 64 characters were entered, the line is ignored and a new line input requested.

Line 480 increments the line # counter if a proper line of data was entered and goes back to line 450 for the entry of more data.

Line 490 is the exit routine. The value of variable A is placed in variable A1 and the program goes back for entry of the next option selection code.

Lines 500 thru 540 are the list routine. A prompt is issued requesting the starting line #. You may start with line # 1 and get a full list of all the data or any line # and get a list of the data from that line to the end. If the line # entered does not exist in the data, an error message is issued and a new starting line # is requested. The entry of an "0" will exit the routine. Again, no error trap is included so be careful in entering the responses.

Lines 550 thru 650 cover the edit routine. A prompt is issued requesting the # of the line to be edited. If an improper line# is entered, an error message is issued and a request for a line # repeated. If a "0" is entered, the procedure will exit and go back to the option code entry line. If a valid line # is entered, this line will be displayed and a prompt option for changing the line is issued. A C/R will default to a "Y" and an "N" will cause the line # request prompt to be redisplayed.

If a C/R was entered, the line to be edited will be redisplayed and the input of the new line requested. The new input will appear directly below the old line so you can copy the old data and see the change in the data when you enter the change. You exit the procedure with the entry of a "0".

Lines 660 and 670 are a subroutine to print to the CRT the numbered list for the edit list routine. This is the response to option code #4 and will go to the edit routine after the list is displayed.

Lines 680 thru 710 are the data save to tape routine and are virtually the same as the data load from tape routine. I do not think any specific explanation of this function is necessary.

Lines 720 thru 1320 cover the various printer functions. Four different types of printer outputs are generated within this section and many of the lines contain EPSON printer control commands. These, of course, will have to be changed or eliminated for other brands of printers. Incidentally, the program assumes that the printer is set to print the first line at the fourth physical line on the form. Also, each form is printed in the "Block" style (no indentations).

Lines 740 thru 780 request an input to designate the type of printer output desired - a proof copy or a draft (final)copy. A C/R defaults to a "D" while a "P" will generate a proof copy, The error trap is line 780.

If a proof copy was requested, the program goes to line 790 which sets the output to go to the printer by the P/O (short-hand for POUT). The date is printed (Y string) and the line number and line are printed. Line 830 increments the line counter and, if 56 lines have been printed, calls the form feed subroutine at line 1500. The line counter is reset to 0 and the next line is printed on the following page.

When all the data has been printed, the program falls through to line 850 where the EPSON printer control commands are turned off. Line 860 turns the program printer output off, clears the screen and goes back to get the next option selection code.

If a draft copy was requested at line 740, the program goes to line 870 where a prompt requests an entry for the type of final copy to print. If a letter form is requested, the program goes to line 1270. If a plain form was selected, the program goes to line 1320. If a memo form was chosen, the program falls through to line 940.

At line 940, the names of the "TO" addressees are requested. Up to 10 names may be entered and a plain C/R will exit the"TO" addressee section. One note of interest - the word "TO" will be printed to the left of the first "TO" name only although all names will be printed as a list.

Line 990 similarily requests the names of the carbon copy (CC) addressees with the same rules and results as the "TO" addressees.

Line 1040 requests the name of the sender and line 1050 asks for the subject of the memo.

Lines 1080 and 1090 put the "TO","CC","FROM", and "SUBJECT" in front of the appropiate strings to form the proper line headings for each section of the memo.

Lines 1100 and 1110 print the heading in double width (chr$(14))

MENU

Line 1120 prints the date. Lines 1130 thru 1160 print the "TO" names. If only one "TO" name was entered, the program will exit this section after that one name was printed or after all "TO" names have been printed.

Lines 1170 thru 1210 do the same for the "CC" names. Line 1220 prints the sender name and line 1230 prints the subject. Line 1240 prints down three blank lines, sets the line counter for the lines already printed and goes to line 1330 to print the body of the memo.

Line 1270 starts the letter printing function. At line 1275, the users name is printed in double width, emphasized format. Line 1280 prints the street address and line 1290 prints the city,state, and Zip code.

Line 1300 prints the date, sets the line counter and goes to line 1330.

Line 1320 is the entry point for the plain form routine and merely prints the date, sets the line counter and goes to line 1330.

Line 1330 prints the body of any of the forms, counts the number of lines printed and, if 56 lines have been printed, calls the form feed subroutine, resets the line counter, and resumes printing the next line. When all data has been printed, the program goes to line 850 to exit the printer section.

The menu display subroutine is contained in lines 1380 thru 1490, the form feed subroutine in line 1400, and the error message subroutine in lines 1610 and line 1620.

A word about option selection code #8 is in order. This is the code to add data to the already existing data. The routine for this is included in lines 290 thru 295. Here, the last-line entered in the existing data will be displayed and the program then goes to line 430 for the entry of the new data.

The program is very long with all of the REM type statements included. You should eliminate as many of these as you see fit so that you program space is not wasted. Also, if you have entered and printed one set of data and wish to start another data entry session, you must exit the program completely and restart it in order to clear all of the strings. If, for some reason, you had to exit the program prematurely (to check remaining memory for a very long set of data for instance), you can reenter the program and not lose any existing data by using the command "RUN 100". This entry point bypasses the variable setting line and picks up where you left off as far as the existing data is concerned. You must reenter the date, however.

All in all, the program pretty much does all that I want it to do. However, I am sure each user will have their own likes and needs to add to the program. Some other features that you might want to add could be variable number of spaces between lines, the addition of the addressee's name for a more formal letter, automatic indentation of the first line of each paragraph, etc.. These are all easily done with just a little bit of imagination. In fact, if you do come up with some interesting variations, send them to QUESTDATA and we will fit them into a future column so others have a chance to include them in their program.

```
5 REM   Line Editor Program
10 REM   Adapted from Quest's "Simple Line Editor" Program
15 REM   Super Basic Version 1.4 manual
20 REM
25 REM   Initialize variables
30 CLS:A=0:A1=0
32 REM
35 REM   Set up headings for MEMO form
40 H$="MEMO":H$(1)="===="
42 REM
50 REM   Set up letterhead
60 N$(1)="xxxxxxxxxxxxxxxxxxxx": REM <-- Insert your name
70 N$(2)="xxxxxxxxxxxxxxxxxxxx": REM <-- Insert street address
80 N$(3)="xxxxxxxxxxxxxxxxxxxx": REM <-- insert city, state, zip
82 REM
85 REM   Main program
90 PRINT "*** LINE EDITOR ***"
92 REM
95 REM   Enter today's date
100 PRINT : INPUT "ENTER DATE (MMM. DD, YYYY) "Y$
101 IF Y$="" THENY$=" "
102 REM
105 REM  Do you want to start entering data or do something else
110 PRINT : INPUT "GO RIGHT TO ENTER NEW DATA ( [CR] or N )"D$
115 REM  An "N" or "n" will get you to the next prompt question
120 IF D$="N" GOTO 160
130 IF D$="n" GOTO 160
135 REM  Enter or Return key will default to "Y"
140 IF D$="" GOTO 420
145 REM  Invalid entry will cause an error message
150 GOSUB 1610: GOTO 110
152 REM
155 REM  Menu display option
160 PRINT : INPUT "DO YOU WISH TO SEE THE MENU ( Y or [CR] )"M$
165 REM  A "Y" or "y" will display the menu
170 IF M$="Y" CLS: GOSUB 1380: GOTO 210
180 IF M$="y" CLS: GOSUB 1380: GOTO 210
185 REM  Enter or Return key will default to "N"
190 IF M$="" CLS: GOTO 210
195 REM  Invalid entry will cause an error message
200 GOSUB 1610: GOTO 160
202 REM
205 REM  Time to pick the task - RETURN will default to ENTER
210 PRINT : INPUT "ENTER YOUR SELECTION ( [CR] = ENTER )"S$
215 REM  Option #1 = enter NEW data
216 REM     C/R defaults to "1"
220 IF S$="1" GOTO 380
222 IF S$="" GOTO 380
225 REM  Option #2 = load data from tape
230 IF S$="2" GOTO 340
235 REM  Option #3 = list data already entered
240 IF S$="3" GOTO 500
245 REM  Option #4 = dispaly an edit listing by line number
250 IF S$="4" GOSUB 660: GOTO 550
255 REM  Option #5 = edit a designated line
260 IF S$="5" GOTO 550
265 REM  Option #6 = save data to tape
270 IF S$="6" GOTO 680
275 REM  Option #7 = printing operations
280 IF S$="7" GOTO 720
285 REM  Option #8 = add data to data already entered
286 REM
287 REM                1) Display last line entered
288 REM
289 REM                2) Go to add data to old data
290 IF S$="8" PRINT : PRINT "  LAST LINE ENTERED:":A=A1
293 IF S$="8" PRINT : PRINT A$(A)
295 IF S$="8" PRINT : GOTO 430
297 REM  Option "M" or "m" = display the menu
300 IF S$="M" CLS: GOSUB 1380: GOTO 210
310 IF S$="m" CLS: GOSUB 1380: GOTO 210
315 REM  Option #9 = exit from program
320 IF S$="9" CLS: END
325 REM  Invalid entry will cause an error mesage
330 GOSUB 1610: GOTO 210
332 REM
335 REM  Load data from tape
340 CLS: PRINT "DATA LOAD MODE:"
342 REM  Which page of data to load?
350 PRINT : INPUT "WHAT PAGE # DO YOU WANT TO LOAD "P
355 IF P=0 THENP=1
360 PRINT : PRINT "REWIND TAPE TO START OF DATA.";
362 PRINT " SET RECORDER FOR PLAY."
363 INPUT "PRESS RETURN WHEN READY."Z$
365 DLOAD ,P
366 REM       (D/L) is my shorthand for DLOAD
370 CLS: GOTO 210
372 REM
375 REM  Enter new data from your keyboard
380 CLS: PRINT "KEYBOARD ENTRY MODE:"
382 REM  Only 64 characters allowed per line
390 PRINT : PRINT "64 CHARACTERS PER LINE MAXIMUM"
392 REM  I use the symbol " " " to exit the enter mode
400 PRINT "TYPE THE SYMBOL " TO EXIT"
402 REM  Allow some time to read the instructions
410 WAIT(70)
420 CLS
422 REM  Set up line number counter
430 A=A+1: PRINT "ENTER DATA": PRINT
432 REM  Establish line length limiter
440 C=64
442 REM  Enter line
450 INPUT A$(A)
452 REM  If """ typed, end of enter session
460 IF A$(A)="""A=A-1: GOTO 490
462 REM  If more than 64 characters entered, re-enter line
470 IF LEN(A$(A))>C PRINT "LINE TOO LONG  -  RE-ENTER": GOTO 450
472 REM  Valid line entered. Increment line number counter
480 A=A+1: GOTO 450
482 REM  All done
490 A1=A: PRINT : PRINT "PAGE COMPLETE": GOTO 210
492 REM
```

```
495 REM  List data entered
500 CLS: PRINT "LIST MODE:"
502 REM  Enter starting line # - entry of "0" will exit
510 PRINT : INPUT "START WITH WHICH LINE # "S
515 IF S=0 CLS: GOTO 210
520 IF S>A1 PRINT "THAT LINE # DOES NOT EXIST": GOTO 510
530 PRINT : FOR A=S TO A1: PRINT A$(A): NEXT
540 PRINT : PRINT " END OF LIST": GOTO 210
542 REM
545 REM  Editing mode
550 PRINT
552 INPUT "WHAT LINE # IS TO BE EDITED (ENTER 0 TO EXIT)"L
560 IF L=0 GOTO 210
570 IF L>A1 PRINT "THAT LINE # DOES NOT EXIST": GOTO 550
572 REM  Display selected line
580 PRINT " ";A$(L)
590 PRINT : INPUT "WANT TO CHANGE IT ( [CR] or N )"Z$
592 REM  Enter or Return will default to "Y"
600 IF Z$="N" GOTO 550
610 IF Z$="n" GOTO 550
620 IF Z$="" GOTO 640
622 REM  Invalid entry will cause an error mesage
630 GOSUB 1610: GOTO 550
635 REM  Redisplay line
640 PRINT " ";A$(L)
642 REM  Enter corrected line
650 INPUT A$(L): GOTO 550
652 REM
655 REM  Subroutine to print numbered line listing to screen
660 PRINT : FOR A=1 TO A1: PRINT A;" ";A$(A): NEXT
670 RETURN
672 REM
675 REM  Save data on tape
680 CLS: PRINT "DATA SAVE MODE:"
690 PRINT : PRINT "SET TAPE TO END OF LAST DATA RECORDED.";
692 PRINT " SET RECORDER TO RECORD."
700 INPUT "PRESS RETURN WHEN READY"Z$
710 DSAVE : CLS: GOTO 210
712 REM       (D/S) is my shorthand for DSAVE
714 REM
715 REM  Printing operations
720 CLS: PRINT "PRINT MODE:"
730 PRINT : PRINT "POSITION PAPER IN PRINTER"
732 REM  A proof copy will be a numbered hard copy list
734 REM    Enter or Return will default to "D"
740 PRINT : INPUT "DO YOU WANT A PROOF OR DRAFT COPY ( P or [CR] )"Z$
750 IF Z$="P" GOTO 790
760 IF Z$="p" GOTO 790
770 IF Z$="" GOTO 870
772 REM  Invalid entry will cause an error message
780 GOSUB 1610: GOTO 740
781 REM
782 REM  PROOF printing operation
790 POUT : REM   (P/O) is my shorthand for POUT
792 REM  Print date (Y$) - B is line counter for End Of Form
800 B=2: PRINT TAB(8);Y$: PRINT
```

```
802 REM  Set line number counter
810 FOR A=1 TO A1
812 REM  Print proof line
820 PRINT A;TAB(4);A$(A)
822 REM  Increment line counter - if EOF reached, do Form Feed
823 REM    Reset line counter
830 B=B+1: IF B=56 GOSUB 1500:B=0
840 NEXT
842 REM  The following is an EPSON printer command which turns
843 REM  off the Emphasized print mode and performs a Form Feed
850 PRINT CHR$(27,70);CHR$(12)
852 REM  Print job done
860 TOUT : CLS: GOTO 210
862 REM       (T/O) is my shorthand for TOUT
863 REM
865 REM  DRAFT (or Finished copy) printing operation
870 PRINT
872 INPUT "PRINT A LETTER, MEMO, OR PLAIN FORM (L/M/P)"P$
874 REM  If "L" or "P" is not selected, program falls thru to
875 REM  line # 940 which is start of MEMO printing
880 IF P$="L" GOTO 1270
890 IF P$="l" GOTO 1270
920 IF P$="P" GOTO 1320
930 IF P$="p" GOTO 1320
931 REM
932 REM  MEMO print operation
933 REM
935 REM  Enter parties to receive the memo - the "TO" will
936 REM  appear to the left of the first addressee only
940 PRINT : PRINT "ENTER 'TO' ADDRESSEES  ( [CR] = EXIT )"
942 REM  Up to 10 addressees may be entered
950 FOR X=1 TO 10
960 INPUT T$(X)
962 REM  The Enter or Return key will exit "TO" entry section
970 IF T$(X)=""X=X-1: EXIT 990
980 NEXT X
981 REM
982 REM  Enter parties to receive a "Carbon Copy" of the memo
983 REM  The "CC" will appear to the left of the first name
990 PRINT : PRINT "ENTER 'CC' ADDRESSEES  ( [CR] = EXIT )"
992 REM  Up to 10 addressees may be entered
1000 FOR Y=1 TO 10
1010 INPUT C$(Y)
1012 REM The Enter or Return key will exit "CC" entry section
1020 IF C$(Y)=""Y=Y-1: EXIT 1040
1030 NEXT Y
1032 REM Enter name of memo sender
1040 PRINT : INPUT "ENTER ADDRESSOR"F$
1042 REM Enter subject matter of memo
1050 PRINT : INPUT "ENTER SUBJECT"M$
1052 REM Set up headings
1060 U$="TO: ":G$="FROM: ":O$="SUBJECT: "
1070 B$="CC: "
1072 REM Assign headings to first name of each section
1080 V$(1)=U$+T$(1):J$=G$+F$:P$=O$+M$
1090 E$(1)=B$+C$(1)
```

```
1092 REM Set output to printer - P/O is shorthand for POUT
1093 REM The CHR$(14) is an Epson printer command to print
1094 REM the word "MEMO" and underscore in DOUBLE WIDTH
1100 POUT : PRINT : PRINT TAB(36);CHR$(14);H$
1110 PRINT TAB(36);CHR$(14);H$(1): PRINT
1112 REM Print the date
1120 PRINT : PRINT TAB(4);Y$: PRINT
1122 REM Print "TO" addressees
1130 PRINT : PRINT TAB(4);V$(1)
1132 REM If only one "TO" addressee, exit to print "CC" section
1140 FOR W=2 TO X: IF T$(W)="" EXIT 1170
1150 PRINT TAB(8);T$(W)
1160 NEXT W
1162 REM Print "CC" section
1163 REM If no "CC" addressees, (Y=0), go to next section
1170 IF Y=0 GOTO 1220
1172 REM Print "CC" addressees
1180 PRINT : PRINT TAB(4);E$(1)
1182 REM If only one "CC" addressee, exit to next section
1190 FOR V=2 TO Y: IF C$(V)="" EXIT 1220
1200 PRINT TAB(8);C$(V)
1210 NEXT V
1212 REM Print sender
1220 PRINT : PRINT TAB(4);J$
1222 REM Print subject
1230 PRINT : PRINT TAB(4);P$
1232 REM Space down to first line of memo - set line counter
1233 REM Go to line printing operation
1240 PRINT : PRINT : PRINT :B=14+(X-1)+(Y-1): GOTO 1330
1250 REM
1260 REM LETTER printing operation
1262 REM
1265 REM Set output to printer. CHR$(14) is Epson Double Width
1266 REM print command. CHR$(27,69) is Epson Emphasized print
1267 REM command.
1270 POUT : REM (P/O) is shorthand for POUT
1272 REM Print Name portion of letterhead
1275 PRINT TAB(29);CHR$(27,69);CHR$(14);N$(1)
1277 REM Print address portion of letterhead
1280 PRINT TAB(34);CHR$(27,69);N$(2)
1282 REM Print city, state, zip
1290 PRINT TAB(30);CHR$(27,69);N$(3)
1292 REM Print date, set line counter, go to print line section
1300 PRINT : PRINT : PRINT TAB(4);Y$: PRINT : PRINT :B=8
1310 GOTO 1330
1311 REM
1312 REM Plain form entry point
1315 REM
1317 REM Print date
1320 POUT : PRINT : PRINT TAB(4);Y$: PRINT :B=2
1321 REM
1322 REM Print lines
1330 FOR A=1 TO A1
1340 PRINT TAB(4);A$(A)
1350 B=B+1: IF B=56 GOSUB 1500:B=0
1352 REM Increment line counter - if EOF reached, do Form Feed
1353 REM   Reset line counter
1360 NEXT
1362 REM Job done - exit to turn off printer option
1370 GOTO 850
1372 REM
1375 REM Display menu subroutine
1380 PRINT : PRINT " *** MENU ***"
1390 PRINT " ============"
1400 PRINT : PRINT "ENTER - NEW DATA = 1 OR PRESS 'RETURN'"
1410 PRINT "        - TAPE      = 2"
1420 PRINT "LIST  -             = 3"
1430 PRINT "EDIT  - # LIST      = 4"
1440 PRINT "      - NO LIST     = 5"
1450 PRINT "SAVE  -             = 6"
1460 PRINT "PRINT -             = 7"
1470 PRINT "ADD   -             = 8"
1480 PRINT "QUIT  -             = 9"
1485 PRINT "DISPLAY MENU -      = M"
1490 RETURN
1492 REM
1495 REM FORM FEED subroutine
1497 REM  CHR$(12) is an Epson printer command for a Form Feed
1500 PRINT CHR$(12): RETURN
1510 REM
1550 REM Error message subroutine
1610 PRINT : PRINT "THAT DOES NOT COMPUTE. TRY AGAIN!"
1620 RETURN
1700 END
```

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

Publisher..............Quest Electronics
Editor......................Paul Messinger
QBUG Editor...................Fred Hannan
Proof Reading.................Judy Pitkin
Production...................John Larimer

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.

SUPER SOUND

by

Don Stevens

The music program given here is like McCormack's program (Quest vol. 1, nu. 7) in that tones are generated by the process of reading a number from a table of input data and then using this number in a counting routine to determine how long Q is on and off. In McCormack's program the number of machine cycles in a complete on-off cycle of Q is equal to 68+8*N, where N is the number read from the table. For the program presented here the number of machine cycles in a complete on-off cycle of Q is equal to N, and so the frequency of the note is C/8N, where C = clock frequency. The program can play notes three octaves higher than the McCormack program can with the same resolution. In fact, the program provides about twice the frequency resolution that the General Instruments AY-8910 chip provides (this chip is used in the Alf board) and also twice the frequency resolution that the RCA 1863 chip provides (this chip is used in the RCA VP-550 Super Sound Board). This Super Sound Program includes even harmonics in its audio signal and has a more pleasing timbre.

The data in the data table can use either one byte or two bytes to specify the frequency of the note. If the 3 highest order bits of the first byte are all equal to 1, then the remaining 13 bits of the first two bytes specify N as a 13 bit binary number. If the above condition is not met, then from this byte are extracted the tone number (from 1 to 12, 1 gives C, 2 gives C#, 3 gives D, 4 gives D#, 5 gives E, etc. to 12 gives B) from the first nibble and the octave (from 0 to 8) from the second nibble. First byte = 14 hex gives middle C or C4 (261.6 hz), first byte = 10 hex gives C0 (16.3 hz), first byte = 58 hex gives E8 (5327 hz). All the notes from C0 to E8 can be produced, a greater range than a piano produces.

The duration is determined by the value of the byte following the byte(s) which gives the tone. The duration is approximately equal to .0183 * byte value / (1-.00015*F), where F is the frequency (in hz) of the note being played. For ordinary music the duration is approximately .02 * byte value. The duration should be greater than the duration of 16 cycles of the note being played. If the first data byte = 01, a silent note is produced with duration determined by the second data byte (actually a 14khz tone is produced). If the first data byte = 00, the program jumps to the address contained in bytes B2 (high address byte) and B3 (low address byte) in the program.

The program is given in the listing and it fits in one page. The address of the data table in this listing is 00 F5 hex, with the high byte specified at 06 and the low byte specified at 09. Of course, the address of the data table may be changed as the user chooses. If the PC is R0 and not R3, change byte 00 from 93 (GHI3) to 90 (GHI0). If the program is loaded as listed at 00 00, it will play a certain tone repeatedly until the input key is held down. This causes the program to read new data (entered on the hex keypad) and play the corresponding tone.

The equal tempered musical scale is in general use today because it works equally in every key. With our Super Sound Program we can use a better scale for any given key. The program has a list of divisors (starting at byte DD) which specify the equal tempered scale by giving the approximate divisors for tones 1 through 12 in octave 0. To use a different divisor list, put the high byte of the address of the list in byte 03 and put the low byte of the address of the list in byte C4 of the program. The divisor list should not straddle a page boundary.

The program may be run on any page. For easy reference, here are the locations in the program where various addresses should be put:

	high byte	low byte
data table	06	09
exit address	B2	B3
divisor table	03	C4

```
00      93 BE F8 00 8C F8 00 B6    F8 F5 A6 E6 7B 06 64 7A
10      AC FF E0 3B B0 8C 7B FA    1F B7 7A 46 A7 46 B8 7B
20      97 F6 7A B9 87 76 A9 7B    29 29 7A 29 29 29 99 7B
30      F6 B9 7A 89 76 A9 76 7B    AD 99 7A F6 B9 89 76 7B
40      A9 8D 7A 76 AD 99 F6 7B    B9 89 7A 76 A9 8D 76 7B
50      F6 F6 7A F6 F6 F6 AD 7B    29 99 7A FC 01 B9 29 F8
60      99 7B AE 87 7A F6 7A 33    6D F8 99 30 6F F8 C4 5E
70      1E 7B 8D 32 7B 7A 2D F8    99 30 6F 7A F8 30 5E 1E
80      F8 83 5E 99 7B BA BB 89    AA AB 2A 9A 3A 8A 98 7A
90      7A 32 0C 2B 28 9B 9B 3A    93 89 FF 01 A9 99 7F 00
A0      B9 30 83 7A 46 B8 28 98    7B 98 98 7A 32 0C 30 A6
B0      8C C2 00 F8 7B FA 0F A8    7A 8C F6 F6 F6 7B FA 1E
C0      32 A3 7A FC DD FF 02 AC    4C B7 7B 0C A7 7A 88 32
D0      1D 97 7B F6 B7 7A 87 76    A7 7B 28 30 CD 35 72 32
E0      72 2F 9D 2C F1 2A 6B 28    0A 25 CB 23 AC 21 AB 1F
F0      C7 1D FF 1C 50 55 10 00    3F 00 26 26 26 6C 30 00
```

```
00        93   GHI3   PUT HIGH BYTE OF
01        BE   PHIE      PC IN RE.1
02        F8   LDI
03        00   00     HIGH BYTE OF DIVISOR
04        BC   PHIC   LIST
05        F8   LDI
06        00   00     DATA TABLE ADDRESS
07        B6   PHI6   HIGH BYTE
08        F8   LDI
09        LF   LF     DATA TABLE ADDRESS
0A        A6   PLO6   LOW BYTE
0B        E6   SEX6
0C   L1   7B   SEQ
0D        06   LDN6   GET NEXT NOTE
0E        64   OUT4   DISPLAY NOTE
0F        7A   REQ
10        AC   PLOC
11        FF   SMI    TEST WHETHER
12        E0   E0     TO USE
13        3B   BNF    TONE TABLE
14        LC   LC     OR NOT
15        8C   GLOC
16        7B   SEQ
17        FA   ANI    ELIMINATE HIGH
18        1F   1F     ORDER 3 BITS
19        B7   PHI7
1A        7A   REQ
1B        46   LDA6
1C        A7   PLO7
1D   L2   46   LDA6   GET DURATION
1E        B8   PHI8
1F        7B   SEQ
20        97   GHI7   COMPUTE R9
21        F6   SHR    AND
22        7A   REQ    ADJUST
23        B9   PHI9   LOOP
24        87   GLO7   TIMING
25        76   SHRC
26        A9   PLO9
27        7B   SEQ
28        29   DEC9
29        29   DEC9
2A        7A   REQ
2B        29   DEC9
2C        29   DEC9
2D        29   DEC9
2E        99   GHI9
2F        7B   SEQ
30        F6   SHR
31        B9   PHI9
32        7A   REQ
33        89   GLO9
34        76   SHRC
35        A9   PLO9
36        76   SHRC
37        7B   SEQ
38        AD   PLOD
39        99   GHI9
3A        7A   REQ
3B        F6   SHR
3C        B9   PHI9
3D        89   GLO9
3E        76   SHRC
3F        7B   SEQ
40        A9   PLO9
41        8D   GLOD
42        7A   REQ
43        76   SHRC
44        AD   PLOD
45        99   GHI9
46        F6   SHR
47        7B   SEQ
48        B9   PHI9
49        89   GLO9
4A        7A   REQ
4B        76   SHRC
4C        A9   PLO9
4D        8D   GLOD
4E        76   SHRC
4F        7B   SEQ
50        F6   SHR
51        F6   SHR
52        7A   REQ
53        F6   SHR
```

```
54        F6   SHR
55        F6   SHR
56        AD   PLOD
57        7B   SEQ
58        29   DEC9
59        99   GHI9
5A        7A   REQ
5B        FC   ADI
5C        01   01
5D        B9   PHI9
5E        29   DEC9
5F        F8   LDI
60        L9   L9
61        7B   SEQ
62        AE   PLOE
63        87   GLO7
64        7A   REQ
65        F6   SHR
66        7A   REQ
67        33   BDF
68        L3   L3
69        F8   LDI
6A        99   99
6B        30   BR
6C        L4   L4
6D   L3   F8   LDI
6E        C4   C4
6F   L4   5E   STRE
70        1E   INCE
71        7B   SEQ
72        8D   GLOD
73        32   BZ
74        L5   L5
75        7A   REQ
76        2D   DECD
77        F8   LDI
78        99   99
79        30   BR
7A        L4   L4
7B   L5   7A   REQ
7C        F8   LDI
7D        30   30
7E        5E   STRE
7F        1E   INCE
80        F8   LDI
81        L6   L6
82        5E   STRE
83   L6   99   GHI9   PLAY THE
84        7B   SEQ    NOTE
85        BA   PHIA
86        BB   PHIB
87        89   GLO9
88        AA   PLOA
89        AB   PLOB
8A   L7   2A   DECA
8B        9A   GHIA
8C        3A   BNZ
8D        L7   L7
8E        98   GHI8
8F        7A   REQ
90        7A   REQ
91        32   BZ     GO FOR
92        L1   L1     NEW NOTE
93   L8   2B   DECB
94        28   DEC8
95        9B   GHIB
96        9B   GHIB
97        3A   BNZ
98        L8   L8
99   L9   89   GLO9   ADJUST
9A        FF   SMI    LOOP
9B        01   01     TIMING
9C        A9   PLO9
9D        99   GHI9
9E        7F   SMBI
9F        00   00
A0        B9   PHI9
A1        30   BR     CONTINUE
A2        L6   L6     THE NOTE
CD   LD   7A   REQ
CE        88   GLO8   LOOP
CF        32   BZ     SHIFTS
D0        L2   L2     R7
D1        97   GHI7   BY OCTAVE
D2        7B   SEQ
```

```
A3   LA   7A   REQ
A4        46   LDA6   DO SILENT NOTE
A5        B8   PHI8   FIRST
A6   LB   28   DEC8   GET
A7        98   GHI8   THE
A8        7B   SEQ    DURATION
A9        98   GHI8   THEN
AA        98   GHI8   LOOP
AB        7A   REQ    FOR
AC        32   BZ     THE
AD        L1   L1     DURATION
AE        30   BR
AF        LB   LB
B0   LC   8C   GLOC   TABLE LOOK UP
B1        C2   LBZ    FOR TONE
B2        00   00     USERS CHOICE
B3        LG   LG     WHERE TO GO
B4        7B   SEQ
B5        FA   ANI
B6        0F   0F
B7        A8   PLO8   R8 HAS OCTAVE
B8        7A   REQ
B9        8C   GLOC
BA        F6   SHR
BB        F6   SHR
BC        F6   SHR
BD        7B   SEQ
BE        FA   ANI
BF        1E   1E
C0        32   BZ     IF TONE=0,
C1        LA   LA     TO SILENT NOTE
C2        7A   REQ
C3        FC   ADI
C4        LE   LE     LOW BYTE OF
C5        FF   SMI    DIVISOR LIST
C6        02   02
C7        AC   PLOC
C8        4C   LDAC
C9        B7   PHI7
CA        7B   SEQ
CB        0C   LDNC
CC        A7   PLO7
D3        F6   SHR
D4        B7   PHI7
D5        7A   REQ
D6        87   GLO7
D7        76   SHRC
D8        A7   PLO7
D9        7B   SEQ
DA        28   DEC8
DB        30   BR
DC        LD   LD
DD   LE   35   35     THIS
DE        72   72     .
DF        32   32     IS
E0        72   72     .
E1        2F   2F     THE
E2        9D   9D     .
E3        2C   2C     DIVISOR
E4        F1   F1     .
E5        2A   2A     TABLE
E6        6B   6B     .
E7        28   28     FOR
E8        0A   0A     .
E9        25   25     THE
EA        CB   CB     .
EB        23   23     EQUAL
EC        AC   AC     .
ED        21   21     TEMPERED
EE        AB   AB     .
EF        1F   1F     SCALE
F0        C7   C7     .
F1        1D   1D     .
F2        FF   FF     .
F3        1C   1C     .
F4        50   50     .
F5   LF   55   55     INTERACTIVE
F6        10   10     DATA TABLE
F7        00   00
F8   LG   3F   3F     TEST INPUT
F9        00   00     READ NEW
FA        26   26     TONE AND
FB        26   26     OCTAVE
FC        26   26     DATA
FD        6C   6C
FE        30   30
FF        00   00
```

DAY OF THE WEEK

by

Werner Cirsovius

Want to know the week day of 2,16,1978?

Type in the following short program into your ELF, input day, month, and year and observe the display. It shows "04" and means "Thursday".

How to run the program:

Start the ELF

Key in day : 16
Push input key, ELF displays : 10

Key in month : 02
Push input key, ELF displays : 02

Key in year : 78
Push input key, ELF displays : 4E
Push input key, ELF displays : 04

ELF stops

The generated code means:

00 : Sunday
01 : Monday
02 : Tuesday
03 : Wednesday
04 : Thursday
05 : Friday
06 : Saturday

Note that the ELF doesn't check for legal input of day, month or year.

Although I tested the program and it runs well, the range from X,X,1901 is not correct. It begins correctly with X,X,1905 but 12,31,1904 is correct. Can anyone find the bug?

```
0000 ;           ..
0000 ;           ..****************************************************
.
0000 ;           ..COMPUTE A DAY OF A WEEK FROM 1,1,1901 TO 12,31,1999
.
0000 ;           ..
0000 ;           ..WRITTEN BY W.CIRSOVIUS;
0000 ;           ..****************************************************
.
0000 ;           ..
0000 ;           ..GENERATED CODE:0 SUNDAY
0000 ;           ..               1 MONDAY
0000 ;           ..                 "  "
0000 ;           ..               6 SATURDAY
0000 ;           ..
0000 ;           ..------------------------------
0000 ;           ..
0000 ;           ..ALGORITHM FROM FOLLOWING BASIC PART
0000 ;           ..
0000 ;           ..IF M<=2 THEN M=M+12;Y=Y-1
0000 ;           ..C=INT((13*M-27)/5)+D+Y+INT(Y/4)-34
0000 ;           ..COD=C-INT(C/7)*7
0000 ;           ..
```

```
0000 ;                 0021 ..------------------------------
0000 ;                 0022 ..
0000 ;                 0023 ..WITH ; D DAY
0000 ;                 0024 ..         M MONTH
0000 ;                 0025 ..         Y YEAR
0000 ;                 0026 ..
0000 ;                 0027 ..------------------------------
0000 ;                 0028 ..
0000 ;                 0029 ..REGISTER ASSIGNMENT
0000 ;                 0030 ..
0000 ;                 0031 PC0=0
0000 ;                 0032 X=2
0000 ;                 0033 WRK=3
0000 ;                 0034 AC=4
0000 ;                 0035 ..
0000 ;                 0036 ..------------------------------
0000 ;                 0037 ..
0000 ;                 0038 ..CONSTANTS
0000 ;                 0039 ..
0000 ;                 0040 DISPL=4
0000 ;                 0041 HI=#F0
0000 ;                 0042 LO=#0F
0000 ;                 0043 ..
0000 ;                 0044 ..------------------------------
0000 ;                 0045 ..
0000 C4C4C4;           0046         NOP;NOP;NOP
0003 90B2B3;           0047         GHI PC0;PHI X;PHI WRK
0006 F86EA3;           0048         LDI A.0(DMY);PLO WRK     ..POINT TO DAY,MONTH,
.YEAR
0009 F873A2;           0049         LDI A.0(STCK);PLO X      ..POINT TO STACK
000C E3;               0050         SEX WRK
000D F803A4;           0051         LDI 3;PLO AC     ..SET COUNT
0010 3F103712;         0052 INPUT:  BN4 *;B4 *       ..WAIT FOR QUIT
0014 6C;               0053         INP DISPL     ..GET DECIMAL INPUT
0015 FA0FB4;           0054         ANI LO;PHI AC   ..SAVE LOW DIGIT
0018 03FAF0;           0055         LDN WRK;ANI HI ..MULTIPLY HIGH DIGIT TIMES TE
.N
001B F653;             0056         SHR;STR WRK        ..TIMES 8
001D F6F6;             0057         SHR;SHR ..TIMES 2
001F F453;             0058         ADD;STR WRK      ..TIMES 10
0021 94F453;           0059         GHI AC;ADD;STR WRK       ..INSERT LOW DIGIT
0024 248464;           0060         DEC AC;GLO AC;OUT DISPL  ..DISPLAY HEX
0027 3A10;             0061         BNZ INPUT
0029 3F29372B;         0062         BN4 *;B4 *        ..WAIT FOR START
002D F86FA3;           0063         LDI A.0(DMY+1);PLO WRK  ..POINT TO MONTH
0030 43FF03;           0064         LDA WRK;SMI 3    ..TEST IF M>2
0033 333D;             0065         BGE NOCOR
0035 F801;             0066         LDI 1    ..IF NOT:Y-1
0037 F573;             0067         SD;STXD
0039 F80C;             0068         LDI 12  ..M+12
003B F453;             0069         ADD;STR WRK
003D F86EA3;           0070 NOCOR:  LDI A.0(DMY);PLO WRK
0040 4313;             0071         LDA WRK;INC WRK
0042 F4FF22;           0072         ADD;SMI 34        ..D+Y-34
0045 52E2;             0073         STR X;SEX X
0047 03F6F6;           0074         LDN WRK;SHR;SHR ..INT(Y/4)
004A F47323;           0075         ADD;STXD;DEC WRK            ..V1=D+Y-34+INT(Y/4)
004D E303;             0076         SEX WRK;LDN WRK
004F FEF4FE;           0077         SHL;ADD;SHL        ..M*13
0052 FEF4;             0078         SHL;ADD
0054 FF1B52;           0079         SMI 27;STR X       ..M*13-27
0057 F8FFA4;           0080         LDI @-1;PLO AC
005A E272;             0081         SEX X;LDXA
005C FF0514;           0082         SMI 5;INC AC       ..V2=INT((13*M-27)/5)
005F 335C;             0083         BPZ *-3
0061 84F4;             0084         GLO AC;ADD         ..C=V1+V2
0063 FF07;             0085         SMI 7    ..COD=D-INT(D/7)*7
0065 3363;             0086         BPZ *-2
0067 FC0752;           0087         ADI 7;STR X
006A 64;               0088         OUT DISPL    ..DISPLAY CODE
006B 00C4C4;           0089         IDL;NOP;NOP
006E ;                 0090 DMY:    ORG *+3
0071 ;                 0091         ORG *+2
0073 ;                 0092 STCK:   END
```

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