

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 GREMLIN 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 program will first print its identification and a symbol key. afterwards, it will request the initial 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 infested 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 generation 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. Basically, 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 locat-

ion 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 R&0 on stack
          18 F8 00 A8 Set line length counter (R.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
```

```

21 FF 80      by R1 - R1 + 1
              Subtracr 80 - convert to probab-
23 30 2B      le ASCII character
              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 R&O (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 R&O
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 appropriate 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 message 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 at line 490.

Line 470 tests the length of the data line entered and, if more than 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 similarly 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 appropriate 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.

```

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
288 REM
289 REM
290 IF S$="8" PRINT : PRINT " LAST LINE ENTERED:".A=A1
295 IF S$="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$="9" CLS: END
325 REM Invalid entry will cause an error message
330 GOSUB 1610: GOTO 210
332 REM
335 REM Load data from tape
340 CLS: PRINT "DATA LOAD MODE:"
342 REM Which page of data to load?
350 PRINT : INPUT "WHAT PAGE # DO YOU WANT TO LOAD "P
355 IF P=0 THENP=1
360 PRINT : PRINT "REWIND TAPE TO START OF DATA.:"
362 PRINT " SET RECORDER FOR PLAY."
363 INPUT "PRESS RETURN WHEN READY."Z$
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

```

```

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$(1)="xxxxxxxxxxxxxxxxxxxxx": REM --- Insert your name
70 N$(2)="xxxxxxxxxxxxxxxxxxxxx": REM --- Insert street address
80 N$(3)="xxxxxxxxxxxxxxxxxxxxx": REM --- Insert city, state, zip
82 REM
85 REM Main program
90 PRINT "**** LINE EDITOR ****"
92 REM
95 REM Enter today's date
100 PRINT : INPUT "ENTER DATE (MM, DD, YYYY) "Y$
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 "N"
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 = display 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

```



```

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$="" GOTO 640
622 REM Invalid entry will cause an error message
630 GOSUB 1610: GOTO 550
635 REM Redisplay line
640 PRINT " ";A$(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 "0"
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"FS$
1042 REM Enter subject matter of memo
1050 PRINT : INPUT "ENTER SUBJECT"MS$
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 :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 - # LIST = 3"
1430 PRINT "EDIT - NO LIST = 4"
1440 PRINT " - = 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

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.



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 $.0183 * \text{byte value} / (1-.00015*F)$, where F is the frequency (in hz) of the note being played. For ordinary music the duration is approximately $.02 * \text{byte value}$. 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 (GH13) to 90 (GH10). 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 BC 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	54	F6	SHR	A3	LA	7A	REQ	
01	BE	PHIE	PC IN RE.1	55	F6	SHR	A4		46	LDA6	DO SILENT NOTE
02	F8	LDI		56	AD	PLOD	A5		88	PHI8	FIRST
03	00	00	HIGH BYTE OF DIVISOR	57	7B	SEQ	A6	LB	28	DEC8	GET
04	BC	PHIC	LIST	58	29	DEC9	A7		98	GHI8	THE
05	F8	LDI		59	99	GHI9	A8		7B	SEQ	DURATION
06	00	00	DATA TABLE ADDRESS	5A	7A	REQ	A9		98	GHI8	THEN
07	B6	PHI6	HIGH BYTE	5B	FC	ADI	AA		98	GHI8	LOOP
08	F8	LDI		5C	01	01	AB		7A	REQ	FOR
09	LF	LF	DATA TABLE ADDRESS	5D	B9	PHI9	AC		32	BZ	THE
0A	A6	PLO6	LOW BYTE	5E	29	DEC9	AD		L1	L1	DURATION
0B	E6	SEX6		5F	F8	LDI	AE		30	BR	
0C	L1	7B	SEQ	60	L9	L9	AF		LB	LB	
0D	06	LDN6	GET NEXT NOTE	61	7B	SEQ	B0	LC	8C	GLOC	TABLE LOOK UP
0E	64	OUT4	DISPLAY NOTE	62	AE	PLOE	B1		C2	LBZ	FOR TONE
0F	7A	REQ		63	87	GLO7	B2		00	00	USERS CHOICE
10	AC	PLOC		64	7A	REQ	B3		LG	LG	WHERE TO GO
11	FF	SMI	TEST WHETHER	65	F6	SHR	B4		7B	SEQ	
12	E0	E0	TO USE	66	7A	REQ	B5		FA	ANI	
13	3B	BNF	TO NE TABLE	67	33	BDF	B6		0F	0F	
14	LC	LC	OR NOT	68	L3	L3	B7		A8	PLO8	R8 HAS OCTAVE
15	8C	GLOC		69	F8	LDI	B8		7A	REQ	
16	7B	SEQ		6A	99	99	B9		8C	GLOC	
17	FA	ANI	ELIMINATE HIGH	6B	30	BR	BA		F6	SHR	
18	1F	1F	ORDER 3 BITS	6C	L4	L4	BB		F6	SHR	
19	B7	PHI7		6D	L3	F8	BC		F6	SHR	
1A	7A	REQ		6E	C4	C4	BD		7B	SEQ	
1B	46	LDA6		6F	L4	5E	BE		FA	ANI	
1C	A7	PLO7		70	1E	INCE	BF		1E	1E	
1D	L2	46	GET DURATION	71	7B	SEQ	C0		32	BZ	IF TONE=0,
1E	B8	PHI8		72	8D	GLOD	C1		LA	LA	TO SILENT NOTE
1F	7B	SEQ		73	32	BZ	C2		7A	REQ	
20	97	GHI7	COMPUTE R9	74	L5	L5	C3		FC	ADI	
21	F6	SHR	AND	75	7A	REQ	C4		LE	LE	LOW BYTE OF
22	7A	REQ	ADJUST	76	2D	DECD	C5		FF	SMI	DIVISOR LIST
23	B9	PHI9	LOOP	77	F8	LDI	C6		02	02	
24	87	GLO7	TIMING	78	99	99	C7		AC	PLOC	
25	76	SHRC		79	30	BR	C8		4C	LDAC	
26	A9	PLO9		7A	L4	L4	C9		B7	PHI7	
27	7B	SEQ		7B	L5	7A	CA		7B	SEQ	
28	29	DEC9		7C	F8	LDI	CB		0C	LDNC	
29	29	DEC9		7D	30	30	CC		A7	PLO7	
2A	7A	REQ		7E	5E	STRE	D3		F6	SHR	
2B	29	DEC9		7F	1E	INCE	D4		B7	PHI7	
2C	29	DEC9		80	F8	LDI	D5		7A	REQ	
2D	29	DEC9		81	L6	L6	D6		87	GLO7	
2E	99	GHI9		82	5E	STRE	D7		76	SHRC	
2F	7B	SEQ		83	L6	99	GHI9	PLAY THE	D8	A7	PLO7
30	F6	SHR		84	7B	SEQ	D9	NOTE	D9	7B	SEQ
31	B9	PHI9		85	BA	PHIA	DA		28	DEC8	
32	7A	REQ		86	BB	PHIB	DB		30	BR	
33	89	GLO9		87	89	GLO9	DC		LD	LD	
34	76	SHRC		88	AA	PLOA	DD	LE	35	35	THIS
35	A9	PLO9		89	AB	PLOB	DE		72	72	.
36	76	SHRC		8A	L7	2A	DECA		32	32	IS
37	7B	SEQ		8B	9A	GHIA	E0		72	72	.
38	AD	PLOD		8C	3A	BNZ	E1		2F	2F	THE
39	99	GHI9		8D	L7	L7	E2		9D	9D	.
3A	7A	REQ		8E	98	GHI8	E3		2C	2C	DIVISOR
3B	F6	SHR		8F	7A	REQ	F4		F1	F1	.
3C	B9	PHI9		90	7A	REQ	E5		2A	2A	TABLE
3D	89	GLO9		91	32	BZ	F6	GO FOR	6B	6B	.
3E	76	SHRC		92	L1	L1	E7	NEW NOTE	28	28	FOR
3F	7B	SEQ		93	L8	2B	DECB		0A	0A	.
40	A9	PLO9		94	28	DEC8	E8		25	25	THE
41	8D	GLOD		95	9B	GHIB	EA		CB	CB	.
42	7A	REQ		96	9B	GHIB	EB		23	23	EQUAL
43	76	SHRC		97	3A	BNZ	EC		AC	AC	.
44	AD	PLOD		98	L8	L8	ED		21	21	TEMPERED
45	99	GHI9		99	L9	89	GLO9	ADJUST	EE	AB	AB
46	F6	SHR		9A	FF	SMI	EF	LOOP	1F	1F	SCALE
47	7B	SEQ		9B	01	01	F0	TIMING	C7	C7	.
48	B9	PHI9		9C	A9	PLO9	F1		1D	1D	.
49	89	GLO9		9D	99	GHI9	F2		FF	FF	.
4A	7A	REQ		9E	7F	SMBI	F3		1C	1C	.
4B	76	SHRC		9F	00	00	F4		50	50	.
4C	A9	PLO9		A0	B9	PHI9	F5	LF	55	55	INTERACTIVE
4D	8D	GLOD		A1	30	BR	F6		10	10	DATA TABLE
4E	76	SHRC		A2	L6	L6	F7	CONTINUE	00	00	
4F	7B	SEQ		CD	L7	7A	F8	THE NOTE	3F	3F	TEST INPUT
50	F6	SHR		CE	88	GLO8	F9	LOOP	00	00	READ NEW
51	F6	SHR		CF	32	BZ	FA	SHIFTS	26	26	TONE AND
52	7A	REQ		D0	L2	L2	FB	R7	26	26	OCTAVE
53	F6	SHR		D1	97	GHI7	FC	BY OCTAVE	26	26	DATA
				D2	7B	SEQ	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?

ELF stops

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

The generated code means:

How to run the program:

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

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

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 ;          0001 ..
0000 ;          0002 ..*****
.
0000 ;          0003 ..COMPUTE A DAY OF A WEEK FROM 1,1,1901 TO 12,31,1999
.
0000 ;          0004 ..
0000 ;          0005 ..WRITTEN BY W.CIRSOVIUS;
0000 ;          0006 ..*****
.
0000 ;          0007 ..
0000 ;          0008 ..GENERATED CODE:0 SUNDAY
0000 ;          0009 ..           1 MONDAY
0000 ;          0010 ..           " "
0000 ;          0011 ..           6 SATURDAY
0000 ;          0012 ..
0000 ;          0013 ..-----
0000 ;          0014 ..
0000 ;          0015 ..ALGORITHM FROM FOLLOWING BASIC PART
0000 ;          0016 ..
0000 ;          0017 ..IF M<=2 THEN M=M+12;Y=Y-1
0000 ;          0018 ..C=INT((13*M-27)/5)+D+Y+INT(Y/4)-34
0000 ;          0019 ..COD=C-INT(C/7)*7
0000 ;          0020 ..
    
```

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 add \$2.00)
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. _____
 Visa Card No. _____
Expiration Date: _____

NAME _____
ADDRESS _____
CITY _____ STATE _____ ZIP _____

Renewal New Subscription

Quest Electronics Documentation and Software by Roger Pflitsch is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.

```

0000 ;
0001 ;
0002 ;
0003 ;
0004 ;
0005 ;
0006 ;
0007 ;
0008 ;
0009 ;
0010 ;
0011 ;
0012 ;
0013 ;
0014 ;
0015 ;
0016 ;
0017 ;
0018 ;
0019 ;
0020 ;
0021 ;
0022 ;
0023 ;
0024 ;
0025 ;
0026 ;
0027 ;
0028 ;
0029 ;
0030 ;
0031 ;
0032 ;
0033 ;
0034 ;
0035 ;
0036 ;
0037 ;
0038 ;
0039 ;
0040 ;
0041 ;
0042 ;
0043 ;
0044 ;
0045 ;
0046 ;
0047 ;
0048 ;
0049 ;
0050 ;
0051 ;
0052 ;
0053 ;
0054 ;
0055 ;
0056 ;
0057 ;
0058 ;
0059 ;
0060 ;
0061 ;
0062 ;
0063 ;
0064 ;
0065 ;
0066 ;
0067 ;
0068 ;
0069 ;
0070 ;
0071 ;
0072 ;
0073 ;
0074 ;
0075 ;
0076 ;
0077 ;
0078 ;
0079 ;
0080 ;
0081 ;
0082 ;
0083 ;
0084 ;
0085 ;
0086 ;
0087 ;
0088 ;
0089 ;
0090 ;
0091 ;
0092 ;
0093 ;
0094 ;
0095 ;
0096 ;
0097 ;
0098 ;
0099 ;
0100 ;
0101 ;
0102 ;
0103 ;
0104 ;
0105 ;
0106 ;
0107 ;
0108 ;
0109 ;
0110 ;
0111 ;
0112 ;
0113 ;
0114 ;
0115 ;
0116 ;
0117 ;
0118 ;
0119 ;
0120 ;
0121 ;
0122 ;
0123 ;
0124 ;
0125 ;
0126 ;
0127 ;
0128 ;
0129 ;
0130 ;
0131 ;
0132 ;
0133 ;
0134 ;
0135 ;
0136 ;
0137 ;
0138 ;
0139 ;
0140 ;
0141 ;
0142 ;
0143 ;
0144 ;
0145 ;
0146 ;
0147 ;
0148 ;
0149 ;
0150 ;
0151 ;
0152 ;
0153 ;
0154 ;
0155 ;
0156 ;
0157 ;
0158 ;
0159 ;
0160 ;
0161 ;
0162 ;
0163 ;
0164 ;
0165 ;
0166 ;
0167 ;
0168 ;
0169 ;
0170 ;
0171 ;
0172 ;
0173 ;
0174 ;
0175 ;
0176 ;
0177 ;
0178 ;
0179 ;
0180 ;
0181 ;
0182 ;
0183 ;
0184 ;
0185 ;
0186 ;
0187 ;
0188 ;
0189 ;
0190 ;
0191 ;
0192 ;
0193 ;
0194 ;
0195 ;
0196 ;
0197 ;
0198 ;
0199 ;
0200 ;
0201 ;
0202 ;
0203 ;
0204 ;
0205 ;
0206 ;
0207 ;
0208 ;
0209 ;
0210 ;
0211 ;
0212 ;
0213 ;
0214 ;
0215 ;
0216 ;
0217 ;
0218 ;
0219 ;
0220 ;
0221 ;
0222 ;
0223 ;
0224 ;
0225 ;
0226 ;
0227 ;
0228 ;
0229 ;
0230 ;
0231 ;
0232 ;
0233 ;
0234 ;
0235 ;
0236 ;
0237 ;
0238 ;
0239 ;
0240 ;
0241 ;
0242 ;
0243 ;
0244 ;
0245 ;
0246 ;
0247 ;
0248 ;
0249 ;
0250 ;
0251 ;
0252 ;
0253 ;
0254 ;
0255 ;
0256 ;
0257 ;
0258 ;
0259 ;
0260 ;
0261 ;
0262 ;
0263 ;
0264 ;
0265 ;
0266 ;
0267 ;
0268 ;
0269 ;
0270 ;
0271 ;
0272 ;
0273 ;
0274 ;
0275 ;
0276 ;
0277 ;
0278 ;
0279 ;
0280 ;
0281 ;
0282 ;
0283 ;
0284 ;
0285 ;
0286 ;
0287 ;
0288 ;
0289 ;
0290 ;
0291 ;
0292 ;
0293 ;
0294 ;
0295 ;
0296 ;
0297 ;
0298 ;
0299 ;
0300 ;
0301 ;
0302 ;
0303 ;
0304 ;
0305 ;
0306 ;
0307 ;
0308 ;
0309 ;
0310 ;
0311 ;
0312 ;
0313 ;
0314 ;
0315 ;
0316 ;
0317 ;
0318 ;
0319 ;
0320 ;
0321 ;
0322 ;
0323 ;
0324 ;
0325 ;
0326 ;
0327 ;
0328 ;
0329 ;
0330 ;
0331 ;
0332 ;
0333 ;
0334 ;
0335 ;
0336 ;
0337 ;
0338 ;
0339 ;
0340 ;
0341 ;
0342 ;
0343 ;
0344 ;
0345 ;
0346 ;
0347 ;
0348 ;
0349 ;
0350 ;
0351 ;
0352 ;
0353 ;
0354 ;
0355 ;
0356 ;
0357 ;
0358 ;
0359 ;
0360 ;
0361 ;
0362 ;
0363 ;
0364 ;
0365 ;
0366 ;
0367 ;
0368 ;
0369 ;
0370 ;
0371 ;
0372 ;
0373 ;
0374 ;
0375 ;
0376 ;
0377 ;
0378 ;
0379 ;
0380 ;
0381 ;
0382 ;
0383 ;
0384 ;
0385 ;
0386 ;
0387 ;
0388 ;
0389 ;
0390 ;
0391 ;
0392 ;
0393 ;
0394 ;
0395 ;
0396 ;
0397 ;
0398 ;
0399 ;
0400 ;
0401 ;
0402 ;
0403 ;
0404 ;
0405 ;
0406 ;
0407 ;
0408 ;
0409 ;
0410 ;
0411 ;
0412 ;
0413 ;
0414 ;
0415 ;
0416 ;
0417 ;
0418 ;
0419 ;
0420 ;
0421 ;
0422 ;
0423 ;
0424 ;
0425 ;
0426 ;
0427 ;
0428 ;
0429 ;
0430 ;
0431 ;
0432 ;
0433 ;
0434 ;
0435 ;
0436 ;
0437 ;
0438 ;
0439 ;
0440 ;
0441 ;
0442 ;
0443 ;
0444 ;
0445 ;
0446 ;
0447 ;
0448 ;
0449 ;
0450 ;
0451 ;
0452 ;
0453 ;
0454 ;
0455 ;
0456 ;
0457 ;
0458 ;
0459 ;
0460 ;
0461 ;
0462 ;
0463 ;
0464 ;
0465 ;
0466 ;
0467 ;
0468 ;
0469 ;
0470 ;
0471 ;
0472 ;
0473 ;
0474 ;
0475 ;
0476 ;
0477 ;
0478 ;
0479 ;
0480 ;
0481 ;
0482 ;
0483 ;
0484 ;
0485 ;
0486 ;
0487 ;
0488 ;
0489 ;
0490 ;
0491 ;
0492 ;
0493 ;
0494 ;
0495 ;
0496 ;
0497 ;
0498 ;
0499 ;
0500 ;
0501 ;
0502 ;
0503 ;
0504 ;
0505 ;
0506 ;
0507 ;
0508 ;
0509 ;
0510 ;
0511 ;
0512 ;
0513 ;
0514 ;
0515 ;
0516 ;
0517 ;
0518 ;
0519 ;
0520 ;
0521 ;
0522 ;
0523 ;
0524 ;
0525 ;
0526 ;
0527 ;
0528 ;
0529 ;
0530 ;
0531 ;
0532 ;
0533 ;
0534 ;
0535 ;
0536 ;
0537 ;
0538 ;
0539 ;
0540 ;
0541 ;
0542 ;
0543 ;
0544 ;
0545 ;
0546 ;
0547 ;
0548 ;
0549 ;
0550 ;
0551 ;
0552 ;
0553 ;
0554 ;
0555 ;
0556 ;
0557 ;
0558 ;
0559 ;
0560 ;
0561 ;
0562 ;
0563 ;
0564 ;
0565 ;
0566 ;
0567 ;
0568 ;
0569 ;
0570 ;
0571 ;
0572 ;
0573 ;
0574 ;
0575 ;
0576 ;
0577 ;
0578 ;
0579 ;
0580 ;
0581 ;
0582 ;
0583 ;
0584 ;
0585 ;
0586 ;
0587 ;
0588 ;
0589 ;
0590 ;
0591 ;
0592 ;
0593 ;
0594 ;
0595 ;
0596 ;
0597 ;
0598 ;
0599 ;
0600 ;
0601 ;
0602 ;
0603 ;
0604 ;
0605 ;
0606 ;
0607 ;
0608 ;
0609 ;
0610 ;
0611 ;
0612 ;
0613 ;
0614 ;
0615 ;
0616 ;
0617 ;
0618 ;
0619 ;
0620 ;
0621 ;
0622 ;
0623 ;
0624 ;
0625 ;
0626 ;
0627 ;
0628 ;
0629 ;
0630 ;
0631 ;
0632 ;
0633 ;
0634 ;
0635 ;
0636 ;
0637 ;
0638 ;
0639 ;
0640 ;
0641 ;
0642 ;
0643 ;
0644 ;
0645 ;
0646 ;
0647 ;
0648 ;
0649 ;
0650 ;
0651 ;
0652 ;
0653 ;
0654 ;
0655 ;
0656 ;
0657 ;
0658 ;
0659 ;
0660 ;
0661 ;
0662 ;
0663 ;
0664 ;
0665 ;
0666 ;
0667 ;
0668 ;
0669 ;
0670 ;
0671 ;
0672 ;
0673 ;
0674 ;
0675 ;
0676 ;
0677 ;
0678 ;
0679 ;
0680 ;
0681 ;
0682 ;
0683 ;
0684 ;
0685 ;
0686 ;
0687 ;
0688 ;
0689 ;
0690 ;
0691 ;
0692 ;
0693 ;
0694 ;
0695 ;
0696 ;
0697 ;
0698 ;
0699 ;
0700 ;
0701 ;
0702 ;
0703 ;
0704 ;
0705 ;
0706 ;
0707 ;
0708 ;
0709 ;
0710 ;
0711 ;
0712 ;
0713 ;
0714 ;
0715 ;
0716 ;
0717 ;
0718 ;
0719 ;
0720 ;
0721 ;
0722 ;
0723 ;
0724 ;
0725 ;
0726 ;
0727 ;
0728 ;
0729 ;
0730 ;
0731 ;
0732 ;
0733 ;
0734 ;
0735 ;
0736 ;
0737 ;
0738 ;
0739 ;
0740 ;
0741 ;
0742 ;
0743 ;
0744 ;
0745 ;
0746 ;
0747 ;
0748 ;
0749 ;
0750 ;
0751 ;
0752 ;
0753 ;
0754 ;
0755 ;
0756 ;
0757 ;
0758 ;
0759 ;
0760 ;
0761 ;
0762 ;
0763 ;
0764 ;
0765 ;
0766 ;
0767 ;
0768 ;
0769 ;
0770 ;
0771 ;
0772 ;
0773 ;
0774 ;
0775 ;
0776 ;
0777 ;
0778 ;
0779 ;
0780 ;
0781 ;
0782 ;
0783 ;
0784 ;
0785 ;
0786 ;
0787 ;
0788 ;
0789 ;
0790 ;
0791 ;
0792 ;
0793 ;
0794 ;
0795 ;
0796 ;
0797 ;
0798 ;
0799 ;
0800 ;
0801 ;
0802 ;
0803 ;
0804 ;
0805 ;
0806 ;
0807 ;
0808 ;
0809 ;
0810 ;
0811 ;
0812 ;
0813 ;
0814 ;
0815 ;
0816 ;
0817 ;
0818 ;
0819 ;
0820 ;
0821 ;
0822 ;
0823 ;
0824 ;
0825 ;
0826 ;
0827 ;
0828 ;
0829 ;
0830 ;
0831 ;
0832 ;
0833 ;
0834 ;
0835 ;
0836 ;
0837 ;
0838 ;
0839 ;
0840 ;
0841 ;
0842 ;
0843 ;
0844 ;
0845 ;
0846 ;
0847 ;
0848 ;
0849 ;
0850 ;
0851 ;
0852 ;
0853 ;
0854 ;
0855 ;
0856 ;
0857 ;
0858 ;
0859 ;
0860 ;
0861 ;
0862 ;
0863 ;
0864 ;
0865 ;
0866 ;
0867 ;
0868 ;
0869 ;
0870 ;
0871 ;
0872 ;
0873 ;
0874 ;
0875 ;
0876 ;
0877 ;
0878 ;
0879 ;
0880 ;
0881 ;
0882 ;
0883 ;
0884 ;
0885 ;
0886 ;
0887 ;
0888 ;
0889 ;
0890 ;
0891 ;
0892 ;
0893 ;
0894 ;
0895 ;
0896 ;
0897 ;
0898 ;
0899 ;
0900 ;
0901 ;
0902 ;
0903 ;
0904 ;
0905 ;
0906 ;
0907 ;
0908 ;
0909 ;
0910 ;
0911 ;
0912 ;
0913 ;
0914 ;
0915 ;
0916 ;
0917 ;
0918 ;
0919 ;
0920 ;
0921 ;
0922 ;
0923 ;
0924 ;
0925 ;
0926 ;
0927 ;
0928 ;
0929 ;
0930 ;
0931 ;
0932 ;
0933 ;
0934 ;
0935 ;
0936 ;
0937 ;
0938 ;
0939 ;
0940 ;
0941 ;
0942 ;
0943 ;
0944 ;
0945 ;
0946 ;
0947 ;
0948 ;
0949 ;
0950 ;
0951 ;
0952 ;
0953 ;
0954 ;
0955 ;
0956 ;
0957 ;
0958 ;
0959 ;
0960 ;
0961 ;
0962 ;
0963 ;
0964 ;
0965 ;
0966 ;
0967 ;
0968 ;
0969 ;
0970 ;
0971 ;
0972 ;
0973 ;
0974 ;
0975 ;
0976 ;
0977 ;
0978 ;
0979 ;
0980 ;
0981 ;
0982 ;
0983 ;
0984 ;
0985 ;
0986 ;
0987 ;
0988 ;
0989 ;
0990 ;
0991 ;
0992 ;
0993 ;
0994 ;
0995 ;
0996 ;
0997 ;
0998 ;
0999 ;
1000 ;

```

COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC

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

ADDRESS CORRECTION REQUESTED

BULK RATE
U.S. Postage Paid
QUEST
Electronics
Permit No. 549
Santa Clara, CA

Quest Electronics Documentation and Software by Roger Patten is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.