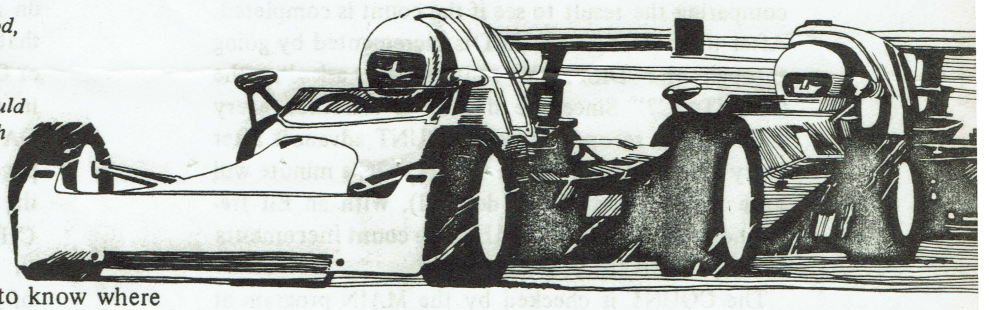
Questdata

Volume 1 Issue #10 ©

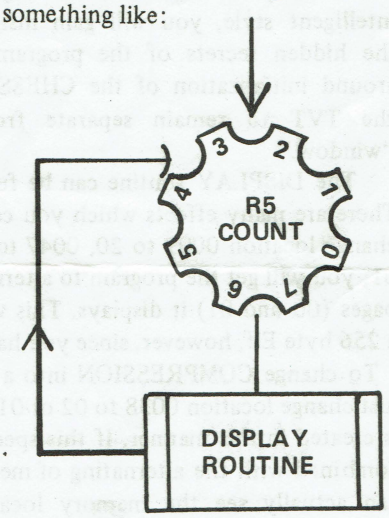
FLAGS AND COUNTERS

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth
–Robert Frost



TO place a tachometer in your racing car you do not have to know exactly how the engine works. However, you do have to know where to place the cog which keeps the count of the RPMs.

We are going to make two modifications to the DISPLAY routine which will give us a work of computer art entitled COMPRESSION. The first thing we will do is place a counter or cog in front of the DISPLAY routine. Each time the display routine goes through its cycle of refreshing the screen display it looks something like:



There really isn't a cog inside your microcomputer adding one to the COUNT in Register 5 each time through the DISPLAY routine, but this is a good way to think of the situation. The other modification we are going to make involves branching–choosing between two roads.

We have quickly jumped into the structure of this program (something programmers have been known to do), now let's take a few steps back to get the overall picture of what we are doing and why.

We are using a trick which has occasionally seen a college student through classes in programming. The trick—"you don't have to completely understand a section of programming code in order to work effectively with it"—will help us to modify the DISPLAY routine to suit our needs. For years programming students have copied sections of code the teacher has written on the blackboard and made only a few minor alterations to arrive at the solution to a new program.

The RCA DISPLAY routine, by itself, is a good example of one of the major programming techniques for the 1802. However, you need not understand everything about this important routine (the subtleties of R0 are difficult) in order to work with it.

Examples of display routines can be found in the 1861 Data Sheet, July 1977 *Popular Electronics*, Super Elf, Elf II, and COSMAC VIP manuals. QUESTDATA #2 has a detailed explanation of how a DISPLAY routine works. For most purposes you can take the basic structure of the routine and with a few modifications adapt it to your purpose. This is our approach to the computer art COMPRESSION.

COMPRESSION will work on 256 byte Elf, Expanded memory Elf, and VIP systems without modification. A counter is used to keep track of the number of times the DISPLAY process is performed. Thus, Register 5 is initialized to 0000. When the DISPLAY goes through its entire refresh process, the count in Register 5 will equal 0001. When the DISPLAY refreshes again, Register 5 will equal 0002, and so on. This counter is going to be useful in making decisions on whether to expand or compress the computer art.

Since the choice to compress or expand is a binary choice, we can use a flag or switch to determine when

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

we are on an expansion or compression cycle.

CHOOSING WHICH ROAD TO TAKE

The counters and flags we are going to use are not unlike the ones which signal that there is one lap left in the race or wave a car in for a pit stop. At every car race there has to be someone counting the number of laps the lead car has taken. The job reduces to the mechanical process of noting that the lead car has made another lap, incrementing a counter to this effect and comparing this with the number of laps in the race. When there is one lap left the flagperson waves a special flag and when the race is finished the flag is waved like crazy and the crowd mobs the driver. This process is like incrementing a register and comparing the result to see if the count is completed. After the computer COUNT is incremented by going through the DISPLAY routine, we ask "is the COUNT=3C?" Since the display is refreshed every 1/60 th of a second, and our COUNT advances after every refresh period; when COUNT=3C a minute will have passed (Hex 3C=60 decimal). With an Elf frequency rate of 1.789773 MHz, the count increments exactly 60.99 times each second.

The COUNT is checked by the MAIN program at locations 0037, 0038, 0039 and 003A. The code at this point asks whether COUNT=3C by Exclusive ORing it with 3C. If the two are equal then the result of the Exclusive OR will be zero. The code FB 3C does the comparison of the count with 3C and the decision to loop or continue is made by code 3A 36. A BRANCH back to location 36 is taken if there is no match (the D Register does not equal 00). If a match is made, then we know that 59/60 ths of a second has elapsed and we either expand or compress our art display.

The decision to expand or compress is really a binary decision. A flag can be considered a binary counter with the numbers 0=off and 1=on. The purpose of a flag is to send the flow of logic down a certain path. The flag is a very versatile and useful tool. When a programmer has painted himself into a programming corner, he will often use a flag to extricate himself. For example, a flag can be used to initialize a register or memory location in the middle of a program. The most obvious flag to use is the Q flip flop because it is preset to 0 on startup and it is easily tested and set.

The first flag example [A] works very much like a toggle switch which changes each time the program logic passes over its section of roadway. The next time the roadway is covered the switch will be thrown "on" if it is off and "off" if it is on. The switch is tested at the BRANCH point to determine which direction the program logic flow will take.

In [B] we have the next run through this section of code determined by the end of the branch line. At the end of the flag-not-on-route, the flag is turned on. And at the end of the flag-on-route, the flag is turned off. When the flag is tested at the BRANCH point it will take the opposite path of the route previously taken.

The last example [C] is what you might call a SILENT flag. Since using the Q flip flop will also turn the speaker on and off at 59/60 ths of a second, it will sound a lot like a grandfather clock. To remove the sound we will use a register flag. The program logic of this routine is exactly the same as [B]. That is, the flag is set or reset at the end of the section of code. Why is Register 8 not initialized in this routine? The Register can contain either a non-zero or 00 number on startup. In this particular instance the only harm that will be done is that it takes either the non-zero or 00 path on its first pass through the program, and in this case it does not really matter. In QUEST-DATA #9 there is an example at the bottom of page number 3 where it really does matter whether the flag is initially on or off. In the case of the CHESS buffer Register 9 was initialized very late in the program. The programmer had the choice of going back to the beginning of the program and pushing everything up 5 bytes to allow for initialization or use the Q as a flag. To initialize in this manner the Q was toggled to on once and not reset–hence, this section of code [0089 thru 008E] was never passed through again and the Q LED remained on. If you go thru this section of code like a computer and execute everything in its unintelligent style, you will gain insight into one of the hidden secrets of the programming cult. Flag around initialization of the CHESS buffer allowed the TVT to remain separate from the CHESS "window."

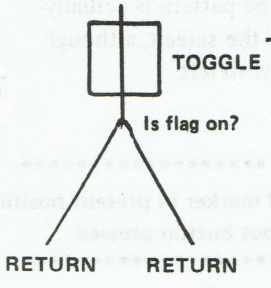
The DISPLAY routine can be fun to work with. There are many effects which you can create. If you change location 0009 to 20, 0047 to 00 and 004C to 01, you will get the program to alternate the memory pages (00 and 01) it displays. This will not work on a 256 byte Elf, however, since you have only page 00.

To change COMPRESSION into a blur of activity, just change location 0038 to 02 or 01. A strobe effect is created in this manner. If this speeded up effect is combined with the alternating of memory pages, you can actually see the memory locations which are not the same on both memory pages. This is much like the way astronomers discover new stars. Another variation of the COMPRESSION program is to run a solid line all the way through the "spring." You can then retitle the program INFLATION. Be warned that many parents, economists, politicians, etc. fail to see the humor of this little gag. If you avoid the gazes of economists while experimenting with INFLATION, you should have a lot of fun.

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

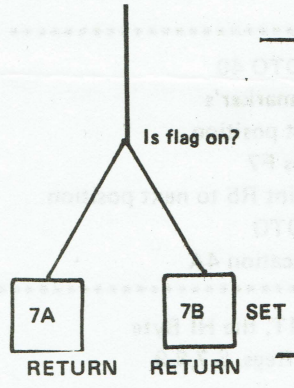
PROGRAM LISTING FOR COMPRESSION

LOCATION	CODE	COMMENTS
0000	90 B1 B2 B3	
0004	B4 B5 B6 A5	INITIALIZE R5 to 00 00
0008	F8 2B A6	Points to expansion-compression location
000B	F8 34 A3	MAIN program
000E	F8 5F A2	STACK location
0011	F8 18 A1	INTERRUPT routine
0014	D3	R3 becomes Program Counter
0015	15	increments COUNT every DISPLAY refresh
0016	72	
0017	70	
0018	22 78	
001A	22 52	
001C	C4 C4 C4	
001F	F8 00 B0	DISPLAY AREA LOCATION
0022	F8 60 A0	memory display starts at loc. 0060
0025	80 E2	
0027	E2 20 A0	
002A	E2 20 A0	Location 2B will alternate between contents
002D	E2 20 A0	20 and 80 to give compression-expansion effect
0030	3C 25	
0032	30 15	
0034	E2 69	
0036	85	Put COUNT in D Register to compare with 3C
0037	FB 3C	XOR with 3C to compare [Try 02 in location 0038]
0039	3A 36	If 3C then continue; Else loop back to 36
003B	F8 00 A5	Reset COUNT



003E	39 43	If Q not on BR to 43 to turn it on
0040	7A	If Q on "fall through" and turn it off
0041	30 46	
0043	7B	
0044	31 4B	Test Q [If Q is on then BRANCH to 4B]
0046	F8 80 56	Put 80 in location 002B to get COMPRESSION
0049	30 36	GOTO wait for COUNT to equal 3C
004B	F8 20 56	Put 20 in location 002B to get EXPANSION
004E	30 36	GOTO wait for COUNT to equal 3C

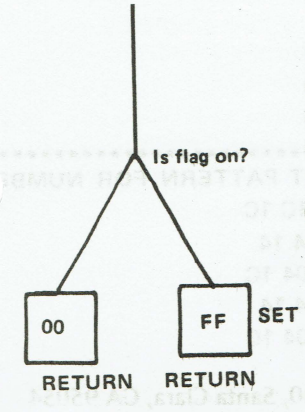
A



LOCATION	CODE	COMMENTS
003E	31 46	BRANCH to opposite of road not taken
0040	F8 80 56	Put 80 in location 002B to get COMPRESSION
0043	7B	Turn on Q SWITCH
0044	30 36	GOTO wait for COUNT to equal 3C
0046	F8 20 56	Put 20 in location 002B to get EXPANSION
0049	7A	TURN OFF Q-LED
004A	30 36	GOTO wait for COUNT to equal 3C

B

SILENT FLAG OPTION



LOCATION	CODE	COMMENTS
003E	8B 32 49	Is B.0=00? If yes, GOTO location 49
0041	F8 80 56	Put 80 in location 002B to get COMPRESSION
0044	F8 00 AB	Set SWITCH to 00
0047	30 36	GOTO wait for COUNT to equal 3C
0049	F8 20 56	Put 20 in location 002B to get EXPANSION
004C	F8 FF AB	Set fSWITCH to FF [or any non-zero hex number]
004F	30 36	GOTO wait for COUNT to equal 3C

C

(see next page for Display Pattern)

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

LOCATION	DATA 1	2	3	4	5	6	7	8
0060	00	00	00	FF	00	00	00	00
0068	00	00	00	FF	00	00	00	00
0070	FF	FF	FF	FF	FF	FF	FF	FF
0078	FF	FF	FF	FF	FF	FF	FF	FF
0080	FF	00	00	00	00	00	00	00
0088	FF	00	00	00	00	00	00	00
0090	FF	FF	FF	FF	FF	FF	FF	FF
0098	FF	FF	FF	FF	FF	FF	FF	FF
00A0	00	00	00	00	00	00	00	FF
00A8	00	00	00	00	00	00	00	FF
00B0	FF	FF	FF	FF	FF	FF	FF	FF
00B8	FF	FF	FF	FF	FF	FF	FF	FF
00C0	FF	00	00	00	00	00	00	00
00C8	FF	00	00	00	00	00	00	00
00D0	FF	FF	FF	FF	FF	FF	FF	FF
00D8	FF	FF	FF	FF	FF	FF	FF	FF
00E0	00	00	00	00	00	00	00	FF
00E8	00	00	00	00	00	00	00	FF
00F0	FF	FF	FF	FF	FF	FF	FF	FF
00F8	FF	FF	FF	FF	FF	FF	FF	FF

More fun with the Display Routine—RANDOM-8

[This program makes use of the DISPLAY Routine in issue number 2 of QUESTDATA. It is a random number generator with visual effects. Half the fun, according to author Ron Zoscak, is planning innovations to add to the DISPLAY Routine. Random 8 breaks into the DISPLAY Routine at location 2F. Note the jump and GAP starting at location 30. This gap allows you to use the full DISPLAY Routine or the Super Elf monitor to load your program.]

By Ron Zoscak

The object of Random 8 is to guess which number the pattern will stop under when the input button is pressed. The bit patterns for the numbers should be loaded into locations C8 through EF with the *PE* video program. The moving pattern under the number is read from the keyboard so that you can easily change to suit your tastes. The pattern is moved by incrementing Register 5 (R5) eight times, then resetting it to point to location F0. The pattern is actually moving from left to right across the screen, although it appears to be moving from right to left.

```
LOC.  CODE  MNEM.    REMARKS
002F  30    BR       GOTO INIT. then
0030  5D             MAIN program
*******************************************
GAP   GAP   GAP   GAP   GAP   GAP   GAP
*******************************************
0040  F8    LDI      MAIN--Put address
0041  F0    DATA     for starting position
0042  A5    PLO R5   of marker in Register 5
*******************************************
0043  F8    LDI      Set limit to
0044  08    DATA     how far the marker
0045  A7    PLO R7   can go
*******************************************
0046  F8    LDI      Put 00
0047  00    DATA     in Register 8
0048  A8    PLO R8   for erasing purposes
*******************************************
0049  E5    SEX R5   Put pattern from keyboard
004A  6C    INP 4    into location pointed to by R5
*******************************************
004B  F8    LDI      Delay Loop
004C  10    DATA
004D  A9    PLO R9
004E  29    DEC R9
004F  89    GLO R9
0050  3A    BNZ
0051  4E
*******************************************
0052  37    B4       Hold marker in present position
0053  52             if input button pressed
*******************************************
0054  88    GLO R8   Erase marker
0055  55    STR R5
*******************************************
0056  27    DEC R7   GOTO 40
0057  87    GLO R7   If marker's
0058  32    BZ       last position
0059  40             was F7
005A  15    INC R5   Point R5 to next position
005B  30    BR       GOTO
005C  4A             Location 4A
*******************************************
005D  90    GHI R0   INIT. the HI Byte
005E  B5    PHI R5   of Regs. 5,7,8,9
005F  B7    PHI R7
0060  B8    PHI R8
0061  B9    PHI R9
0062  30    BR
0063  40
*******************************************
                     BIT PATTERN FOR NUMBE
00C8  18 1C 1C 14 1C 1C 1C 1C
00D0  08 04 04 14 10 10 04 14
00D8  08 1C 1C 1E 1C 1C 04 1C
00E0  08 10 04 04 04 14 04 14
00E8  1C 1C 1C 04 1C 1C 04 1C
```

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

MOEWS MUSIC ALGORITHM

[Note: This music program by Paul Moews is similar to programs in his booklet, **Programs for the COSMAC Elf–Music and Games.** *It is also printed in the new software section of the QUEST Super Elf manual. It is reprinted here as a bonus feature. He has many valuable subroutines in* **Music and Games** *which include a random number generator, display, and number base conversion routines. There is an interesting routine to dump the contents of the COSMAC registers. "Morra," and Bridg-it are among the games in the booklet. For a lot of fun and excellent documentation of 256 byte Elf programs–***Music and Games** *is hard to beat. Paul wants to encourage QUESTDATA readers to work out their own music programs to submit to QUESTDATA using his Algorithm. For more music examples and some fun programming, Paul's booklet can be purchased from Quest Electronics for $2.50 plus 50¢ postage and handling.]*

By Paul C. Moews

The Elf is a computer which is particularly easy to program to produce music, all that is necessary is to attach a speaker to the Q line. Simple music can then be played by turning the Q line on and off to produce tones of the proper frequency and duration.

A number of different approaches to the necessary music program are possible. The program might be designed so that the notes of the tune to be played can be stored in a compact form; for example, five bits might be used to designate one of 32 notes and three bits to specify one of 8 note durations. This method would lead to a compact table of notes but would require a fairly complex program to decode the notes.

Another method, the one used here, is to write a very simple program which uses a more complex and less compact table of notes. More work has to be done to set up the table of note and more memory is used by each note as a trade off for this simple program. However, there are often advantages to a simple approach. One advantage is that the operation of the program is easy to understand. Another is that the program can easily be modified to suit other purposes.

In the program that follows the musical notes are stored in a table in memory; each note takes 3 bytes. The first bytes designates one of 255 frequencies, if the byte is 00 silence (a rest) occurs. The duration of the note is specified by the next two bytes. The program plays music by picking up the notes one after the other from the table and producing the right tone for the proper period of time. Tones are produced by turning the Q line on and off to produce a square wave. The delays while Q is held on or off are obtained by successively subtracting one from the D Register, i.e.:

```
        87 7B       load D from R(7).0, turn Q on
loc 1   FF 01       subtract one from D
        32 loc 1    to location 1 to subtract one
                    more if D is not equal to zero
        7A 87       turn Q off, load D again, etc.
```

For an Elf with a ½ x 3.579545 MHz crystal the instruction pair FF 01, 32 (loc 1) takes 0.00001788 seconds and if the D register is set to 107 (base 10) (6B(base 16)) for both Q on and Q off a note with a frequency of about 261 Hz is produced (middle C). By counting the number of times Q is turned on and off we can control the duration of the note.

In order to write music we have only to set up a table of values to be transferred to D to produce the proper frequencies and a table of the number of cycles to be counted to obtain notes of the right length. Such a table is provided below. Note that 256 (base 10) 100 (base 16) is added to the number of cycles in the duration count because the high order byte is the one checked to see if the proper number of cycles has been produced.

The note table can be used with many music sources (simple piano or recorder music is suitable.) Encoding is done by picking out the note (a group of three bytes) and placing it at the music starting point–location 30. The program to read the notes begins at location 00-2F.

Beethoven's Fifth symphony is an example of the encoding process. First the timing notes are chosen to give what seems an appropriate interval, in this case 1/8 notes are held for ¼ second, and ½ notes for 1 second.

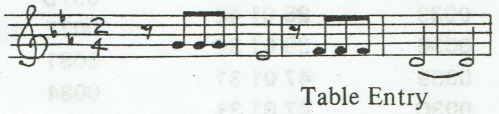


	Table Entry	
	NOTE	DURATION
1/8 rest	00	01 37
1/8 G	47	01 62
1/8 G	47	01 62
1/8 G	47	01 62
1/2 D #	5A	02 37
1/8 rest	00	01 37
1/8 F	50	01 57
1/8 F	50	01 57
1/8 F	50	01 57
1 D	5F	03 4C

First load the MUSIC PROGRAM into locations 00 through 2F. . . in LOCATION 0001 place 0A (the number of notes in this music example (base 16)).

The music program will play the 10 Beethoven notes if they are loaded into memory locations 30 through 4D, and the MUSIC GENERATOR occupies locations 00 through 30 (location 01 being 0A for the Beethoven Fifth sample).

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

MUSIC ALGORITHM

ADDRESS	CODE	NOTES
0000	F8 XX A8	Load number of notes to R(8).0 here 30 (base 16)
	(30)	for mystery tune or 0A (base 16) for Beethoven
0003	F8 00 BA	Starting address of note table to R(A)
0006	F8 30 AA	
0009	EA	Make R(A) the X register
000A	F8 01 BE	Loop for a short delay between
000D	2E 9E	notes - F8 02 BE, etc. for a longer
000F	3A 0D	delay - F8 80 AE 2E 8E, etc., for a
0011	F0 A7	Load first and subsequent notes to D and R(7).0
0013	64 28	Display note and decrement note count
0015	72 BC 72 AC	Load note duration (number of cycles plus 256 (base 10) to R(C)
0019	87	Load current note to D
001A	32 21	If D is not 00 its a rest—skip Q on loop
001C	7B	Turn Q on
001D	FF 01 3A 1D	Waiting loop for Q on count till D equals 00
0021	7A 87	Turn Q off, load note again
0023	FF 01 3A 23	Waiting loop for Q off
0027	2C 9C	Decrement cycle count, load R(C).1 to D
0029	3A 19	To location 19 to check for rest and begin next cycle if note is not finished
002B	88 3A 0A	If note is done comes here, load note count to D, back to 0A for next note if more notes to do
002E	30 00	or if done with all notes, start over
0030	- - -	TABLE OF MUSIC STARTS HERE

MYSTERY TUNE

LOCATION	CODE
0030	2F 01 4A
0033	35 01 42
0036	39 01 3D
0039	47 01 31
003C	47 01 31
003F	5F 01 25
0042	47 01 31
0045	47 01 31
0048	39 01 3D
004B	47 01 31
004E	39 01 3D
0051	2F 01 4A
0054	35 01 42
0057	39 01 3D
005A	35 01 42
005D	40 01 37
0060	40 01 37
0063	5F 01 25
0066	40 01 37
0069	40 01 37
006C	35 01 42
006F	40 01 37
0072	35 01 42
0075	2A 01 53
0078	2F 01 4A
007B	35 01 42
007E	39 01 3D
0081	47 01 31
0084	47 01 31
0087	5F 01 25
008A	47 01 31
008D	47 01 31
0090	39 01 3D
0093	47 01 31
0096	39 01 3D
0099	2F 01 4A
009C	35 01 42
009F	39 01 3D
00A2	35 01 42
00A5	39 01 3D
00A8	35 01 42
00AB	40 01 37
00AE	2F 01 4A
00B1	35 01 42
00B4	39 01 3D
00B7	47 01 31
00BA	47 01 31
00BD	47 01 C5

Reprinted by permission of Paul C. Moews
Copyright © 1978 by Paul C. Moews
All rights reserved.

Important Hardware Note

To attach a speaker to the Q line, as suggested on page 5 and page 8, you must use a buffer Amp. (A direct connection could zap your 1802!)

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

Publisher . . . . . Quest Electronics
Editor . . . . . .Bill Haslacher
Technical Coordinator . . . . . Jane McKennon
Circulation . . . . . Laurie Buzzell
Proofreading . . . . . Ken Brown

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. Subscriptions are $12 for this monthly publication.

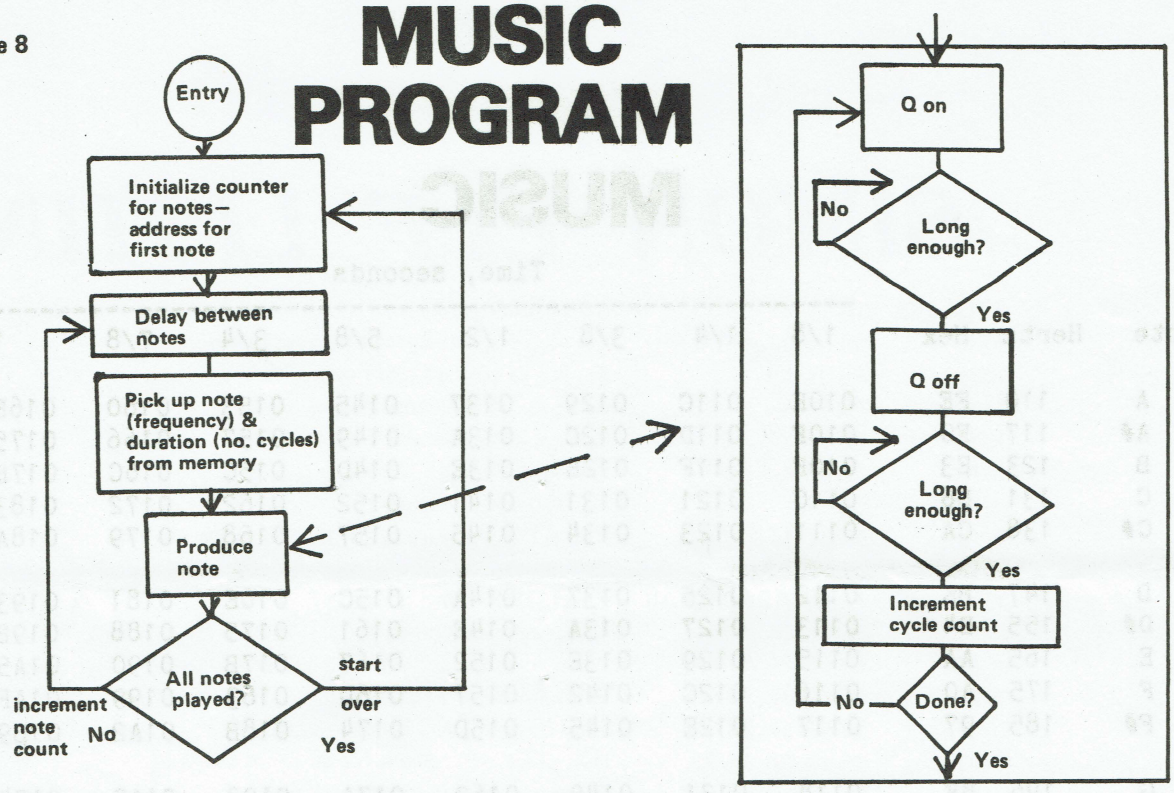
Quest Electronics Documentation and Software by Roger Pitkin is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.

MUSIC

Note	Hertz	Hex	Time, seconds 1/8	1/4	3/8	1/2	5/8	3/4	7/8	1
A	110	FE	010E	011C	0129	0137	0145	0153	0160	016E
A#	117	F0	010F	011D	012C	013A	0149	0157	0166	0175
B	123	E3	010F	011F	012E	013E	014D	015C	016C	017B
C	131	D6	0110	0121	0131	0141	0152	0162	0172	0183
C#	138	CA	0111	0123	0134	0145	0157	0168	0179	018A
D	147	BE	0112	0125	0137	014A	015C	016E	0181	0193
D#	155	B4	0113	0127	013A	014E	0161	0175	0188	019B
E	165	AA	0115	0129	013E	0152	0167	017B	0190	01A5
F	175	A0	0116	012C	0142	0157	016D	0183	0199	01AF
F#	185	97	0117	012E	0145	015D	0174	018B	01A2	01B9
G	196	8F	0118	0131	0149	0162	017A	0193	01AB	01C4
G#	207	87	011A	0134	014E	0168	0181	019B	01B5	01CF
A	220	7F	011C	0137	0153	016E	018A	01A5	01C1	01DC
A#	233	78	011D	013A	0157	0175	0192	01AF	01CC	01E9
B	247	71	011F	013E	015D	017C	019B	01BA	01D9	01F7
C(M)	261	6B	0121	0141	0162	0183	01A3	01C4	01E5	0205
C#	277	65	0123	0145	0168	018A	01AD	01D0	01F2	0215
D	294	5F	0125	014A	016E	0193	01B8	01DD	0202	0226
D#	311	5A	0127	014E	0175	019B	01C2	01E9	0210	0237
E	329	55	0129	0152	017B	01A5	01CE	01F7	0220	0249
F	350	50	012C	0157	0183	01AF	01DA	0206	0232	025E
F#	368	4C	012E	015C	018A	01B8	01E6	0214	0242	0270
G	394	47	0131	0162	0194	01C5	01F6	0227	0259	028A
G#	417	43	0134	0168	019D	01D1	0205	0239	026D	02A1
A	437	40	0137	016D	01A4	01DA	0211	0248	027E	02B5
A#	466	3C	013A	0175	01AF	01E9	0223	025E	0298	02D2
B	491	39	013D	017B	01B8	01F5	0233	0270	02AD	02EB
C	528	35	0142	0184	01C6	0208	024A	028C	02CE	0310
C#	559	32	0146	018C	01D2	0218	025E	02A3	02E9	032F
D	595	2F	014A	0195	01DF	022A	0274	02BE	0309	0353
D#	621	2D	014E	019B	01E9	0237	0284	02D2	0320	036D
E	666	2A	0153	01A6	01FA	024D	02A0	02F3	0347	039A
F	699	28	0157	01AF	0206	025E	02B5	030C	0364	03BB
F#	736	26	015C	01B8	0214	0270	02CC	0328	0384	03E0
REST	218	00	011B	0137	0152	016D	0189	01A4	01BF	01DA

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

MUSIC PROGRAM



SIMON ELF

By Richard Moffie

Last summer, Milton Bradley Inc. introduced an electronic game called Simon (TM of Milton Bradley) that became quite popular. Simon generated a sequence of tones and corresponding colors that must be remembered and matched by the player or players. As the number of tones gets higher and higher, it becomes more difficult to remember the correct sequence and do as 'Simon Says' without making an error and losing the game.

Here is a program that will let you 'Simonize' your Elf computer and make your Elf the life of the party. The program will run on the basic Super Elf or any similar system, provided you have a speaker attached to the Q line*. No additional memory is needed since the program is quite compact and takes only 165 bytes plus the stack area.

To play the game, run the program and push the INPUT switch. The computer will then generate one of 8 random tones and display the corresponding number (from 0 to 7). You must then press the correct key (matching the previously displayed number) to generate the same tone. If you are correct, 'charge' will be played and the number of tones in the sequence is displayed. If you choose the wrong key, a low buzz will sound and you must then start over with a single note (by pressing INPUT). When you enter the numbers, only one key need be pressed, so that if 05 is displayed, you need only enter the 5.

* See hardware note, page 6.

To play the next round (assuming you were successful), press INPUT and a sequence of 2 notes will be played, which you must then match in order. If these are correct, 3 notes will be played, then 4, 5, . . . until you can no longer remember and match the sequence. The stack area is large enough for 40 notes, but if you can get past 9 or 10, you're doing great! You can play this game alone or with a group where players alternate turns.

There are a couple of easy modifications to the program if desired: If you wish to have only 4 different tones rather than 8, just change bytes 1C and 55 from 07 to 03. Also, if you don't want each tone played as you enter it, change byte 51 from D4 to C4.

One of the interesting features of this program is the random tone generator. This is done by repeatedly incrementing Register 7 whenever the program is waiting for INPUT to be pushed, and happens tens of thousands of times a second, making the value of the last 3 bits (R7 is 'AND'ed with 07) impossible to determine and totally independent of any previous values. This has been used in many other programs to get random numbers, but happens to be ideal for this program since a new random number can be generated when INPUT is pushed to begin the new round.

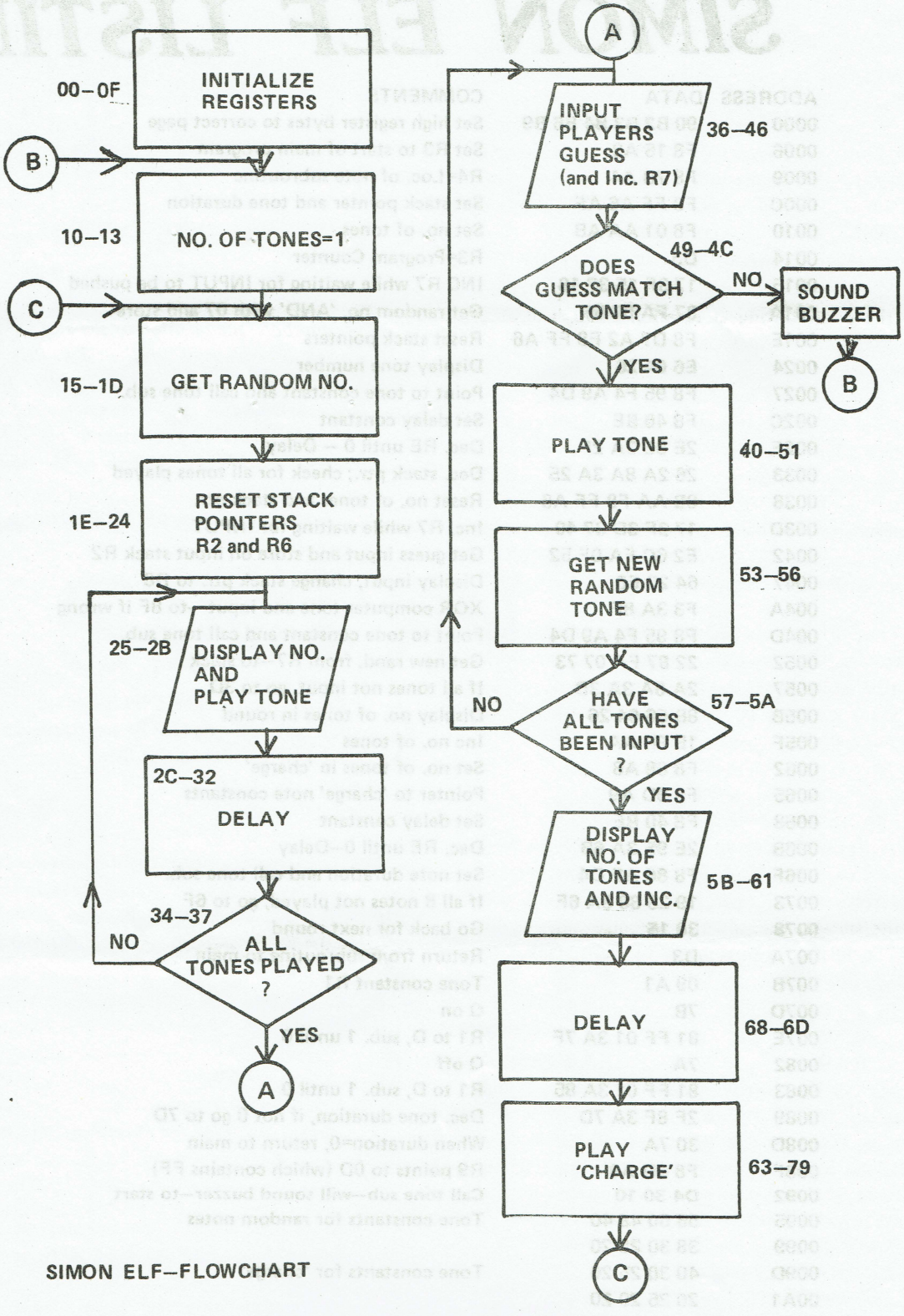
Try running and examining this program. It makes use of many of the 1802's features and shows again that a lot of programming can be done even with a limit of 256 bytes. With all the extra space left, you might even try adding a few features of your own.

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

SIMON ELF LISTING

ADDRESS	DATA	COMMENTS
0000	90 B2 B3 B4 B6 B9	Set high register bytes to correct page
0006	F8 15 A3	Set R3 to start of main program
0009	F8 7B A4	R4=Loc. of note subroutine
000C	F8 FF A6 AF	Set stack pointer and tone duration
0010	F8 01 AA AB	Set no. of tones
0014	D3	R3=Program Counter
0015	17 3F 15 37 18	INC R7 while waiting for INPUT to be pushed
001A	87 FA 07 56	Get random no. 'AND' with 07 and store
001E	F8 D7 A2 F8 FF A6	Reset stack pointers
0024	E6 64 26	Display tone number
0027	F8 95 F4 A9 D4	Point to tone constant and call tone sub.
002C	F8 40 BE	Set delay constant
002F	2E 9E 3A 2F	Dec. RE until 0 – Delay
0033	26 2A 8A 3A 25	Dec. stack ptr.; check for all tones played
0038	8B AA F8 FF A6	Reset no. of tones and stack ptr.
003D	17 3F 3D 37 40	Inc. R7 while waiting for INPUT
0042	E2 6C FA 0F 52	Get guess input and store on input stack R2
0047	64 22 E6	Display input, change stack ptr. to R6
004A	F3 3A 8F	XOR computer tone and input –to 8F if wrong
004D	F8 95 F4 A9 D4	Point to tone constant and call tone sub.
0052	22 87 FA 07 73	Get new rand. from R7–to stack
0057	2A 8A 3A 3D	If all tones not input, go to 3D
005B	8B 56 64 26	Display no. of tones in round
005F	1B 8B AA	Inc no. of tones
0062	F8 08 A8	Set no. of tones in 'charge'
0065	F8 9D A9	Pointer to 'charge' note constants
0068	F8 40 BE	Set delay constant
006B	2E 9E 3A 6B	Dec. RE until 0–Delay
006F	F8 80 AF D4	Set note duration and call tone sub.
0073	19 28 88 3A 6F	If all 8 notes not played, go to 6F
0078	30 15	Go back for next round
007A	D3	Return from subroutine to main
007B	09 A1	Tone constant R1
007D	7B	Q on
007E	81 FF 01 3A 7F	R1 to D, sub. 1 until 0
0082	7A	Q off
0083	81 FF 01 3A 85	R1 to D, sub. 1 until 0
0089	2F 8F 3A 7D	Dec. tone duration, if not 0 go to 7D
008D	30 7A	When duration=0, return to main
008F	F8 0D A9	R9 points to 0D (which contains FF)
0092	D4 30 10	Call tone sub–will sound buzzer–to start
0095	58 50 48 40	Tone constants for random notes
0099	38 30 28 20	
009D	40 30 25 20	Tone constants for 'charge'
00A1	20 25 20 20	
00B0-00D7		Input stack
00D8-00FF		Random tone stack

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



SIMON ELF–FLOWCHART

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

The Great COSMAC Cosmos Puzzle

By Mike Tope

The Great COSMAC Cosmos Puzzle is essentially the Elf computer version of the brain teaser "Teaser" program found in the September, 1974 issue of *PCC*, which is put out by the *Peoples Computer Company*. Although the brain teaser is fairly easy to understand, it is not by any means easy to solve. In fact, your initial attempt at the puzzle will be a good test of your patience as well as puzzle aptitude.

The program requires at least ½K of memory. One page (256 bytes) is for the program which is preferably to be the page starting at location 0000. The other required page is for the video display and has been set to the page starting at location 0400, but may be changed to any page by changing the operand found in location 0016. The optional playing field graphics may be loaded into the display page, but the extra work is only worthwhile if some sort of cassette interface can be used to prevent reloading every time one wishes to run the program. The Q line should also be connected to a speaker for the audio effects incorporated in the program.

Once the program has been loaded, start the program and the COSMAC Cosmos, consisting of 9 positions, will appear in the lower left hand corner of the playing field. Each position in the universe has a specific number as follows:

```
1 2 3
4 5 6
7 8 9
```

If the optional playing field graphics have been employed, the number of the corresponding position will be seen on the right hand side of the screen. Upon the start of the game the universe contains one star (*) at position 5 surrounded by eight black holes (0) as below:

```
0 0 0        1 2 3
0 * 0        4 5 6
0 0 0        7 8 9
```

By pushing the number of a star twice and then pushing the INPUT switch, the star will become a black hole. When the star becomes a black hole, huge gravitational waves are emitted through space so tell your friends that this is why the video display quavers or sways slightly from side to side. The quavering effect is only present if the Elf system is using a 3.58/2 crystal for the video timing.

The collapse of the star into a black hole also changes every other star in its galaxy or in other words, all black holes of the galaxy become stars and all stars become black holes. The galaxies for each star position are given below; knowledge of how the universe will look after a star is zapped is essential to sloving the puzzle.

```
1 x .    x 2 x    . x 3    x . .    . x .
x x .    . . .    . x x    4 . .    x 5 x
. . .    . . .    . . .    x . .    . x .

. . x    . . .    . . .    . . .
. . 6    x x .    . . .    . x x
. . x    7 x .    x 8 x    . x 9
```

numbers=the position of the star which was zapped upon depression of the INPUT switch.

x=the position of the stars or black holes which are changed because of the destruction of the stars.

. =the position of the stars or black holes which are not effected by the collapse of the zapped star.

For example suppose the universe is set in the pattern:

```
0 0 0        1 2 3
0 * 0        4 5 6
0 0 0        7 8 9
```

The player enters 5 via the keyboard and zaps the star by pressing the INPUT switch. The star died but four new stars appear in the positions 2, 4, 6, 8. The universe now looks like this:

```
0 * 0        1 2 3
* 0 *        4 5 6
0 * 0        7 8 9
```

To solve the puzzle zap the stars in a certain sequence in order to change the universe from this:

```
0 0 0        1 2 3    To this:    * * *        1 2 3
0 * 0        4 5 6                * 0 *        4 5 6
0 0 0        7 8 9                * * *        7 8 9
```

But not this:

```
0 0 0        1 2 3
0 0 0        4 5 6
0 0 0        7 8 9
```

Upon obtaining the correct pattern in the universe

```
* * *
* 0 *
* * *
```

the Q light will turn on and stay on. If the optional playing field graphics have been loaded, a special message will appear on the bottom of the screen. If the graphics have not been loaded, you will still notice that the universe positions have jumped up and the bottom of the screen has been filled with random

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

bits. The number of turns that it took to achieve the winning pattern will be displayed in the hex display. The lowest possible number of turns is 11 or 0B (hex).

HOW THE PROGRAM WORKS:

The execution of the program beginning at location 0000 sets up the registers for the video interrupt routine (used to refresh the video display). The actual video display routine is almost identical to the video program described in QUESTDATA number 3; therefore, it will not be dealt with here. Register (7) is set up to point at the various locations on the display page that correspond to the positions of the universe by finding these locations in Table II. Register (9) contains the present state of the universe. Each bit represents a position of the universe. If the bit for a particular universe position is equal to one, the position is determined to be a star. If the bit is zero, then it is a black hole. The high part of Register (9) contains the following star positions 1 2 3 4 6 7 8 9. The low part of Register (9) represents only position 5 or the center of the universe. Register (9.1) is called the universe register, while Register (9.0) is called the center register.

One bit is set in Register (A.1) which is used to examine the bit or position in the universe register. Upon the completion of the bit test, either the star pattern or the black hole pattern is placed in the display page at the location pointed to by the data found in Table II (throught the use of Register (7)). The single set bit in Register (A.1) is shifted to test the next position and the Register (7) is incremented so that it will point to where the next star or black hole pattern should be placed in the display page. After the entire universe register has been tested, the center register is tested as the appropriate pattern is placed upon the video page.

Now the program tests the universe register for the win condition. If the condition exists, the program tests the center register. If the center register is equal to a star a branch is made to the win routine; otherwise, the program waits for the user to press the INPUT switch. Once the INPUT switch has been pressed, it signals the entry of the position of the star to be zapped. The program tests the data entered from the keyboard to make sure it is a valid position in the universe that is not equal to zero and is not greater than nine.

The program now gets the valid star position test mask from Table I to see if the keyboard entry was that of a star. If the entry was not a star, the program jumps back to get another INPUT. The program now tests the INPUT to see if it was 1,3, 7, 9; if so, the center of the new universe will be inverted. The new universe mask is XOR'ed with the universe in order to form the new universe. The score, Register (8.0), is incremented. The program performs a frequency sweep via the Q line and then branches back to display the new universe.

The win routine turns on the Q light and puts 40 hex in the intra-page refresh start register. This causes the video interrupt routine to display the last ¾ of the 0400 page of memory and the first ¼ of the 0500 page; thus, the special message will appear on the bottom of the screen.

Before the win routine is encountered, the program appears to reset the intra-page refresh start Register (8.1) needlessly after every test for the win condition. The resetting was included only because the score is kept in the lower part of the register. After about 256 entries via the keyboard, the special message would begin to prematurely shift onto the screen. The resetting (found in location 006F) prevents this from happening. You may also notice a garbled sound during the frequency sweep. This is because of the delays due to the rather long video refresh process which constantly interrupts the operations of the main program.

So there you have it, the Great COSMAC Cosmos Puzzle. I hope it provides you and your friends many hours of enjoyment (frustration).

REGISTER ALLOCATION DATA

Register	Application
0	Refresh Pointer for Video
1	Interrupt Routine
2	Stack
3	Main
4	Temporary Memory Storage
5	Display Page Pointer
6	Star or Black Hole Pattern Pointer
7	Table pointer and used in freq. sweep
8.0	Score
8.1	Intra-page refresh start for video
9.0	Center of universe storage
9.1	Universe storage
A.0	Line Counter and used in freq. sweep
A.1	Test bit

Back issues of QUESTDATA (starting with issue No. 1) are available for $1.50 each. Programming knowledge is cumulative.

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

COSMAC Cosmos listing

LOC.	OP CODES	COMMENTS
0000	C0 00 03	Optional branch to monitor *
0003	90	Initialize registers
0004	B1 B2 B3	
0007	B4 B6 B7	
000A	F8 23 A1	Point to Interrupt routine
000D	F8 C6 A2	Point to Stack
0010	F8 3D A3	Point to Main
0013	F8 C7 A4	Point to temporary memory storage
0016	F8 04 B5	Point to Display page
0019	F8 00	put 00 in:
001B	A8	(1) score register
001C	B8	(2) intra-page refresh register
001D	B9	(3) universe register
001E	A9 19	Put 01 in center register
0020	D3	GOTO Main
0021	72 70	Return from Interrupt
0023	22 78	Video refresh subroutine
0025	22 52	
0027	C4 C4 C4	
002A	95 B0	
002C	98 A0	
002E	80 E2	
0030	E2 20 A0	
0033	E2 20 A0	
0036	E2 20 A0	
0039	3C 2E	
003B	30 21	
003D	E4	set X to temp. mem. storage
003E	69	Turn on video
003F	F8 EA A7	point R(7) to table II
0042	F8 80 BA	Prepare R(A.1) for . . .
0045	54	universe refister examine
0046	99	get R(9.1) univ. reg.
0047	F2	AND to test for star
0048	30 54	BR to display s./b.h.
004A	17	INC to next disp. pos. addr.
004B	07	Get disp. pos. addr. from table II
004C	32 6F	if loc.=00 exit to input processing
004E	9A	shift R(A.1)'s test bit . . .
004F	F6	to examine next univ. . . .
0050	BA	pos., but if all pos. . . .
0051	3A 45	have been examined . . .
0053	89	get center register
0054	32 5A	if the examined position . . .
0056	F8 C8	if=to 01; point . . .
0058	30 5C	to star pattern
005A	F8 D0	if=to 00; point to
005C	A6	black hole pattern
005D	07	get disp. pos. addr.
005E	A5	put in disp. page reg.
005F	F8 09 AA	set for count lines . . .
0062	2A	in s/bh pattern DEC count
0063	8A	get line count
0064	32 4A	if=00 br. to get next pos. (univ.)
0066	06	get part of s/bh pattern
0067	55	put in disp. page
0068	85	point to R(5) to . . .
0069	FC 08	next address to load . . .
006B	A5	part of s/bh pattern
006C	16	point R(6) to next part of . . .
006D	30 62	s/bh pattern; br. to display it
INPUT PROCESSING		
006F	B8	set intra-page refresh start to 00
0070	99	get univ. from R(9), to test for . . .
0071	FB FF	win condition by inversing it
0073	3A 78	to se if=00
0075	89	if so test center reg. for 00
0076	32 BB	and BR to win routine
0078	3F 78	wait for INPUT depressed
007A	37 7A	wait for INPUT released
007C	6C	get input from keyboard
007D	FA 0F	save lower nybble
007F	54	put in temp. mem. stor.
0080	32 78	if input=00 return
0082	FC F6	if input=09 return . . .
0084	32 78	to get next input
0086	04	get input and . . .
0087	FF 01	transform it into the
0089	FE	valid pos. test mask addr.
008A	FC D8 A7	for table I
008D	07	get valid pos. test mask
008E	3A 93	if pos. test mask=00 . . .
0090	89	test the center reg.
0091	3A 9D	BR to form new univ. if valid
0093	54	put pos. mask in temp. mam. stor.
0094	99	get univ. from R(9) . . .
0095	F2	AND to see if valid . . .
0096	32 78	star was selected
0098	07	test star pos. to see . . .
0099	FA A5	if≠to 1, 3, 7, 9 in which case . . .
009B	32 A1	skip the inversing of center
009D	89	get center reg.
009E	FB 01	inverse it
00A0	A9	put new center in R(9.0)
00A1	17	point R(7) to new univ. mask
00A2	07	get new universe mask from table I
00A3	54	put mask in temp. mem. storage
00A4	99	get univ. and XOR it to . . .
00A5	F3	the mask to inverse the . . .
00A6	B9	specific galaxy
00A7	18	INC score
00A8	F8 22 A7	prepare for freq. sweep
00AB	7B	Turn Q light on
00AC	87 AA	put R(7.0) in R(A.0)
00AE	2A	DEC R(A.0)
00AF	8A	get R(A.0)

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

LOC.	OP CODES	COMMENTS
00B0	3A AE	BR if ≠00
00B2	39 AB	BR to turn Q on if it is off
00B4	7A	turn Q off
00B5	27	DEC frequency
00B6	87	get R(7.0) if not=00 . . .
00B7	3A AC	BR for next frequency
00B9	30 3D	return to display new universe
WIN ROUTINE		
00BB	7B	Turn Q light on
00BC	88	get score
00BD	54	put in temp. storage
00BE	64	Display score on hex output
00BF	F8 40	set intra-page refresh
00C1	B8	register to 40
00C2	30 C2	wait here!
00C4		STACK AREA
00C5		
00C6		
00C7		Temporary Memory Storage Area

(PLEASE CONTINUE TO NEXT COLUMN)

LOC. OP CODES

STAR PATTERN 00C8 to 00CF= 92 54 38 FE 38 54 92 00

BLACK HOLE PATTERN 00D0 to 00D7= 00 38 44 44 44 38 00 00

Table I contains the Valid Star Position Test Masks followed by th... New Universe Masks:

00D8 to 00DF= 80 D0 40 E0 20 68 10 94

00E0 to 00E9= 00 5A 08 29 04 16 02 07 01 0B

Table II contains the addresses to the various universe positions to locate where they should appear on the video display.

00EA to 00F3= 40 41 42 80 82 C0 C1 C2 81 00

*Super Monitor users should change the contents of location 0003 to a 93 to enter the program from the monitor. Elf II owners must first change the first three locations to C0, F0, 00 to jump to their monitor for loading. Then hardware load the first three locations with C0, 00, 03 to start from reset.

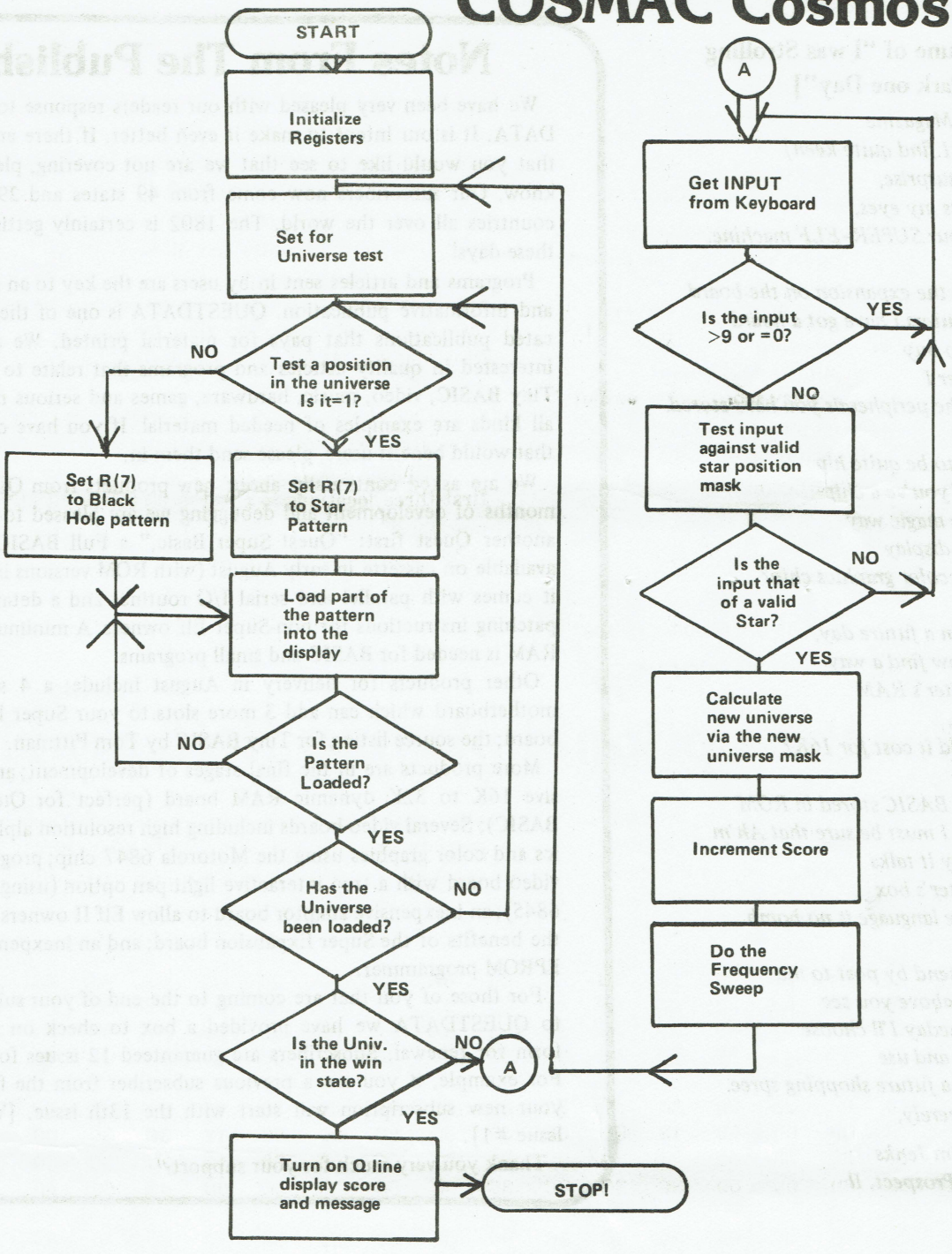
Listing for the optional playing field graphics

The following code should be loaded into the page of memory which will function as the video display buffer. The code below also loads into the first quarter page of memory following the display page. If for some reason your system does not have any memory immediately following the display page ignore the unloaded portion of the code and don't worry about it.

ADDRESS	DATA															
0400-0F	EE	EF	BB	8E	EE	FB	B8	00	8A	8A	AA	08	A8	AA	A0	00
10-1F	8A	8A	AA	08	A8	AA	A0	00	8A	EA	BA	08	AE	AA	B8	00
20-2F	8A	2A	AA	08	A2	AA	88	00	8A	2A	AA	08	A2	AA	88	00
30-3F	EE	EA	AB	8E	EE	AB	B8	00	00	00	00	00	00	00	00	00
0440-4F	00	00	00	00	38	3C	3C	00	00	00	00	00	18	0C	0C	00
50-5F	00	00	00	00	18	0C	0C	00	00	00	00	00	18	3C	3C	00
60-5F	00	00	00	00	18	30	0C	00	00	00	00	00	18	30	0C	00
70-7F	00	00	00	00	3C	3C	3C	3C	00	00	00	00	00	00	00	00
0480-8F	00	00	00	00	2C	3C	3C	00	00	00	00	00	2C	30	30	00
90-9F	00	00	00	00	2C	30	30	00	00	00	00	00	3C	3C	3C	00
A0-AF	00	00	00	00	0C	0C	34	00	00	00	00	00	0C	0C	34	00
B0-BF	00	00	00	00	0C	3C	3C	00	00	00	00	00	00	00	00	00
04C0-CF	00	00	00	00	3C	3C	3C	00	00	00	00	00	2C	2C	2C	00
D0-DF	00	00	00	00	0C	2C	2C	00	00	00	00	00	0C	3C	3C	00
E0-EF	00	00	00	00	0C	2C	0C	00	00	00	00	00	0C	2C	0C	00
F0-FF	00	00	00	00	0C	3C	0C	00	00	00	00	00	00	00	00	00
0500-0F	00	00	00	00	00	00	00	00	EE	EE	EE	EA	8E	EB	BB	80
10-1F	8A	A8	AA	4A	8A	4A	AA	00	8A	A8	AA	4A	8A	4A	AA	00
20-2F	8A	AE	EE	4A	8E	4A	AB	80	8A	AA	CA	4A	8A	4A	A8	80
30-3F	8A	AA	AA	4A	8A	4A	A8	80	EE	AE	AA	4E	EA	4B	AB	80

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

COSMAC Cosmos Puzzle



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

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

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

Signature _____

☐ Renewal ☐ New Subscription

NAME _____

ADDRESS _____

CITY STATE ZIP _____

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

Dear Sirs,

[Sung to the tune of "I was Strolling thru the Park one Day"]

I was reading Byte Magazine
(a publication that I find quite keen)
When quite to my surprise,
Your ad came 'cross my eyes,
Telling all about your SUPER-ELF machine.

I'm impressed with the expansion on the board
But of cheap computers I have got a hoard
So before I'm set to buy
Your ELF computer I
Would like to see the peripherals you have stored.

I find TV graphics to be quite hip
So I'd like to see if you've a chip
That could in some magic way
Add some color to display
I mean, have you a color graphics chip?

And I'm sure that in a future day,
I'll want to somehow find a way
To increase the 'puter's RAM
As cheaply as I can,
So how much would it cost for 16K?

I see that you have BASIC stored in ROM
But before I buy it I must be sure that Ah'm
Happy with the way it talks
In the ELF computer's box
So that I'm sure the language is no bomb.

So will you please send by post to me
Data on the things above you see
'Cause perhaps someday I'll choose
To buy, own, love, and use
Your computer on a future shopping spree.

Sincerely,

Jim Jenks
Mt. Prospect, Il

Notes From The Publisher

We have been very pleased with our readers response to QUESTDATA. It is our intent to make it even better. If there are subjects that you would like to see that we are not covering, please let us know. Our subscribers now come from 49 states and 29 different countries all over the world. The 1802 is certainly getting around these days!

Programs and articles sent in by users are the key to an interesting and informative publication. QUESTDATA is one of the few dedicated publications that pays for material printed. We are always interested in quality articles and programs that relate to the 1802. Tiny BASIC, video, music, hardware, games and serious routines of all kinds are examples of needed material. If you have other ideas that would benefit users, please send them in.

We are asked continually about new products from Quest. After months of development and debugging we are pleased to announce another Quest first: "Quest Super Basic," a Full BASIC. Initially available on cassette in early August (with ROM versions in the fall), it comes with parallel and serial I/O routines and a detailed set of patching instructions for non-Super Elf owners. A minimum of 12K RAM is needed for BASIC and small programs.

Other products for delivery in August include; a 4 slot S-100 motherboard which can add 3 more slots to your Super Expansion board; the source listing for Tiny BASIC by Tom Pittman.

More products are in the final stages of development; an inexpensive 16K to 32K dynamic RAM board (perfect for Quest Super BASIC); Several video boards including high resolution alpha/numerics and color graphics using the Motorola 6847 chip; programmable video board with a true interactive light pen option (using Motorola 6845); an inexpensive adaptor board to allow Elf II owners to obtain the benefits of the Super Expansion board; and an inexpensive 2708 EPROM programmer.

For those of you that are coming to the end of your subscription to QUESTDATA we have provided a box to check on the order form for renewal. Subscribers are guaranteed 12 issues for $12.00. For example, if you are a previous subscriber from the first issue, your new subscription will start with the 13th issue. [Volume 2 Issue #1].

Thank you very much for your support.

COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC

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

ADDRESS CORRECTION REQUESTED

BULK RATE
U.S. Postage Paid
QUEST
Electronics
Permit No. 649
Santa Clara, CA

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