Questdata
Volume 2 Issue # 4 ©

PUT LIFE INTO YOUR COMPUTER

By
Ray Tully

This program is based on the "Life" game devised by Charles Conway, and described in the October, 1970 issue of "Scientific American" (p. 120). The game is initialized by placing into a grid a pattern of "cells" (i.e., bits on the video display). The cells then live, reproduce, and die according to these three "genetic laws": 1. Survivals - every cell with 2 or 3 neighbors survives to the next generation: 2. Deaths - an isolated cell, or one overcrowded with 4 or more neighbors, will not survive to the next generation: 3. Births - an empty grid location with exactly 3 neighbors will contain a cell in the next generation. From a home computerist's point of view, the object of the game is to devise starting patterns that will evolve into as spectacular a series of succeeding generations as possible.

How to Run the Program

First, the starting pattern (first generation) must be drawn on the display. Jump to the "Pattern Initialization Program" at 0060. Either load 30, 60, at 0000 and press and release "R" and "G" or use Quest Super Monitor to execute at 00D0. Page 02 will be erased and displayed. It is divided into six columns of bytes (see diagram under "How the Program Works"). First, using the hex keypad enter the column in which you want to begin drawing, by typing the data on the hex keypad and pressing and releasing "I" (01-06). Next, enter the data for the drawing. The data will be entered into the display vertically. Each column is 30 bytes deep; when you reach the bottom, Q turns on. You can then enter another column number and continue the drawing, if desired. When the drawing is finished, stop the program and jump to the Life program, 0100. Either load C0, 01, 00 at 0000 or use option 00 of Quest's Super Monitor to execute at 00E0. This will display the initial drawing, and then succeeding generations. The program calculates about 33 generations each minute on the 30x48 grid.

The program requires a minimum of 1K bytes of memory, and an 1861 graphics chip. The programs and subroutines are on pages 00 and 01, and the two generations in storage at any given time are on pages 02 and 03.

How The Program Works

Each bit in memory has eight nearest neighbors, numbered as follows:

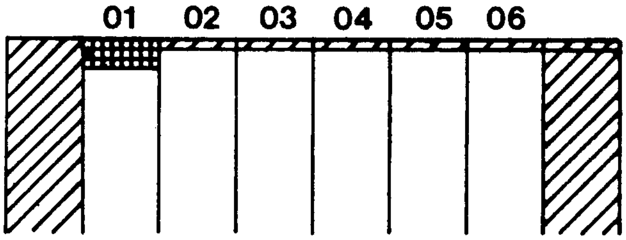
1	2	3
4		5
6	7	8

where the shaded block is the bit being tested. Consider next a column of three bytes in memory, pointed to by R(5), R(6), and R(7):

								← R (5)
L	M	M	M	M	M	M	R	← R (6)
								← R (7)

where M(R(6)) contains the particular bit under test. There are three generalized positions for this bit: left end of byte (L), middle of byte (M), or right end of byte (R). The routine for counting the number of neighbors of a given bit is thus divided into three sections, depending on whether the bit is in L, M, or R.

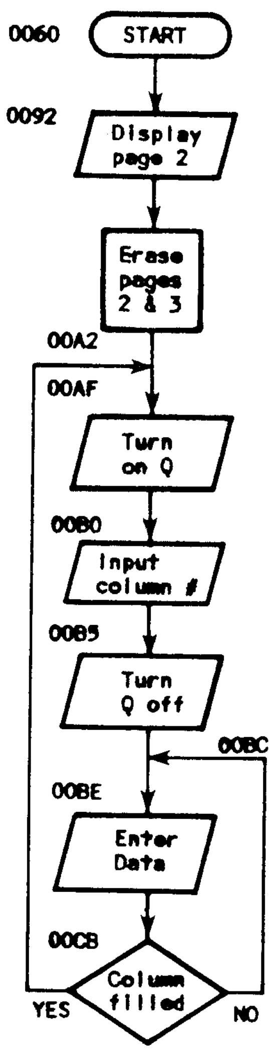
The starting diagram (first generation) is made on page 02. The top and bottom lines of the page, and the far left and right column of bytes, are kept blank (for ease of programming). R(5), R(6), and R(7) are initialized to the upper left corner:

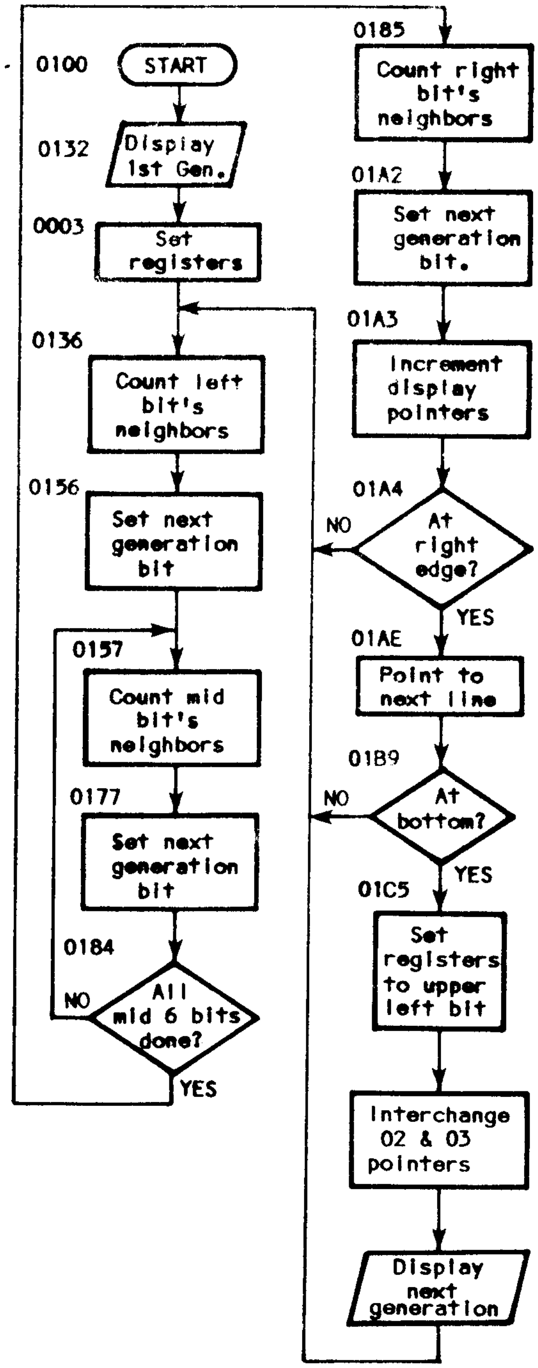


with the solid bit at "L" in M(R(6)) being the first one under test.

The neighbors are counted by placing the appropriate bytes into the Data Register, shifting the individual on or off bits into DF, and incrementing R(B).0 for each one that is on (by calling the subroutine at 0050 with a DE instruction). After counting the neighbors, the state of the bit under test in M(R(6)) is determined (by shifting into DF). If it is on, then Q is turned on. A branch is then made to the "Subroutine to Set Next Generation" at 002C (DC instruction), whose job it is to set the bits for the next generation, depending on the state of the bits in the first. The bits in the next generation are pointed to by M(R(9)). They are set by placing M(R(9)) into the accumulator after having set DF to 0 or 1, shifting DF into the byte, and copying back into M(R(9)).

For each byte, the procedure is to test the leftmost bit (0136-0156), the middle six bits (by passing through the loop 0157-0184 six times), and the rightmost bit (0185-01A2). The current generation pointers (R(5)-R(7)) and next generation pointer (R(9)) are then incremented appropriately, and a test is made to see if the process is finished (01A4-01BA). If so, then the next generation is displayed and made into the current generation, and vice versa.





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

REGISTER INITIALIZATION

ADDR	CODE	COMMENT
0000	C0	
0001	00 60	Load 0060 at first then 0100 (see text)
0003	F8 01 BD A5	
0007	F8 19 AD	
000A	F8 02 5D B5 B6 B7	
0010	F8 11 A7	
0013	F8 00 AB BE BC	
0018	F8 09 A6 A9	
001C	F8 31 AC	
001F	F8 03 B9	
0022	F8 51 AE	
0025	C0 01 36	Return to main program

REGISTER ASSIGNMENTS

R(5)-Points to neighbors #1-3 of current generation.
R(6)-Points to bit under test, and neighbors #4 and 5.
R(7)-Points to neighbors #6-8 of current generation.
R(9)-Points to next generation.
R(B).0-Holds count of neighbors.
R(C)-Points to subroutine to set next generation.
R(D)-Points to location in video routine which contains page of memory being displayed. This page is initialized at 02 for the first generation.
R(E)-Points to subroutine to count neighbors of bit.

SUBROUTINE TO SET NEXT GENERATION

ADDR	CODE	COMMENT
002C	7A F8 00 AB	Turn Q off, reset neighbor count to 00
0030	D3	Return to main program
0031	8B FB 02 32 4C	If neighbors=02 go to Status Quo
0036	8B FF 02 33 43	If count>1, go test if =3

SET NEXT GENERATION BIT TO 0

ADDR	CODE	COMMENT
003B	F8 00 F6	Set DF to 0
003E	09 7E 59 30 2C	Shift DF into M(R(9)), return

TEST FOR MORE THAN 3 NEIGHBORS

ADDR	CODE	COMMENT
0043	8B 7D 03	Test for count >03
0046	3B 3B	If so, set next generation to 0

SET NEXT GENERATION BIT TO 1

ADDR	CODE	COMMENT
0048	F8 FF	Set DF to 1
004A	30 3D	Go shift DF into M(R(9))

STATUS QUO (NO CHANGES TO NEXT GEN)

ADDR	CODE	COMMENT
004C	31 48	If Q=1, go set next generation bit to 1
004E	30 3B	Else, set to 0

SUBROUTINE TO COUNT NEIGHBORS

ADDR	CODE	COMMENT
0050	D3	Return
0051	3B 50	If DF = 0, return
0053	1B 30 50	Else increment count, return

STANDARD VIDEO ROUTINE TO DISPLAY 1 PAGE OF MEMORY

ADDR	CODE	COMMENT
0060	90 B1 B2 B3 B4	
0065	F8 91 A3	
0068	F8 8F A2	
006B	F8 71 A1	
006E	D3 72 70	
0071	22 78 22 52	
0075	C4 C4 C4	
0078	F8 02 B0	
007B	F8 00 A0	
007E	80 E2	
0080	E2 20 A0	
0083	E2 20 A0	
0086	E2 20 A0	
0089	3C 7E	
008B	30 6F	
008D	00 00 00 00	
0091	E2 61	

ERASE PAGES 02 AND 03

ADDR	CODE	COMMENT
0093	E5	X = 5
0094	F8 03 B5	
0097	F8 FF A5	R(5) points to area to be erased
009A	F8 00 73	Enter 00, work your way down
009D	95 FB 01	Continue erasing until you hit page 01
00A0	3A 9A	

ENTER DATA INTO DESIRED COLUMN

ADDR	CODE	COMMENT
00A2	F8 02 B5	R(5) points to column "0" = 0209
00A5	F8 08 A5	
00A8	E6	X = 6
00A9	F8 00 B6	
00AC	F8 FF A6	M(R(6)) stores input data from keys
00AF	7B	Q on to prompt entry of desired column #

ADDR	CODE	COMMENT
00B0	3F B0	Enter column#, Push I
00B2	6C 64 26 7A	Store and display column #, turn Q off
00B6	85 F4 A5	Add column # to R(5).0
00B9	37 B9	Release I
00BB	E5	Reset X to 5
00BC	3F BC	Enter data for pattern, push I
00BE	6C 64	Write data into display area, R(X)+1
00C0	85 FC 07 A5	R(X)+7 to point to next byte down column
00C4	F6 F6 F6 FB 1F	You're at bottom if R(X).0>=F8 (1F if shifted)
00C9	37 C9	Release I
00CB	3A BC	Not yet at bottom?
00CD	30 A2	If yes, go select another column

Super Monitor Entry Point for Data Entry

ADDR	CODE	COMMENT
00D0	F8 60	Make R0 point
00D2	A0	to program.
00D3	F8 90	Make R2 point
00D5	A2	to scratch.
00D6	93	Get 00
00D7	B0	Do R0.1
00D8	B2	Do R2.1
00D9	52	Put 00 in scratch
00DA	70	X=0,P=0, enable interrupts

Super Monitor Entry Point for Display

ADDR	CODE	COMMENT
00E0	F8 01	Make R0 point
00E2	B0	to program.
00E3	B2	R2 too.
00E4	F8 30	Do R2.0
00E6	A2	Do R2.0
00E7	93	Get 00
00E8	A0	Do R0.0
00E9	52	Store 00 in scratch
00EA	70	X=0,P=0, enable interrupts

Standard Video Routine to Display 1 Page of Memory

ADDR	CODE	COMMENT
0100	90 B1 B2 B3 B4	This routine is essentially that published in Questdata #2,p.12
0105	F8 31 A3	
0108	F8 2F A2	
010B	F8 11 A1	
010E	D3 72 70	
0111	22 78 22 52	
0115	C4 C4 C4	
0118	F8 02 B0	
011B	F8 00 A0	
011E	80 E2	
0120	E2 20 A0	
0123	E2 20 A0	
0126	E2 20 A0	
0129	3C 1E	
012B	30 0F	
012D	00 00 00 00	Stack area
0131	E2 61	Turn on TV
0133	C0 00 03	Go initialize registers

ROUTINE FOR LEFTMOST BIT

ADDR	CODE	COMMENT
0136	05 FE DE	Shift neighbor #2 into DF, count if on
0139	FE DE	Shift neighbor #3 into DF, count if on
013B	07 FE DE FE DE	Repeat with neighbors #7 and 8
0140	25 05 F6 DE 15	Repeat with neighbor #1, restore R(5)
0145	26 06 F6 DE 16	Repeat with neighbor #4, restore R(6)
014A	27 07 F6 DE 17	Repeat with neighbor #6, restore R(7)
014F	06 FE 3B 54 7B	Q on if bit on
0154	FE DE	Count neighbor #5
0156	DC	Set next generation

ROUTINE FOR MIDDLE 6 BITS

ADDR	CODE	COMMENT
0157	F8 06 A8	Set R(8).0 to count middle 6 bits
015A	05 AA	Store M(R(5)) in R(A).0
015C	06 BA	Store M(R(6)) in R(A).1
015E	07 BB	Store M(R(7)) in R(B).1
0160	8A FE DE FE DE FE DE	Count neighbors #1-3
0167	9B FE DE FE DE FE DE	Count neighbors #6-8
016E	9A FE DE	Count neighbor #4
0171	FE 3B 75 7B	Q on if bit on
0175	FE DE	Count neighbor #5
0177	DC	Set next genertion
0178	8A FE AA	Shift R(A).0 left
017B	9A FE BA	Shift R(A).1 left
017E	9B FE BB	Shift R(B).1 left
0181	28 88 3A 60	Decrement count repeat loop until = 0

ROUTINE FOR RIGHTMOST BIT

ADDR	CODE	COMMENT
0185	05 F6 DE F6 DE	Count neighbors #1 and 2
018A	07 F6 DE F6 DE	Count neighbors #6 and 7
018F	06 F6 3B 94 7B	Q on if bit on
0194	F6 DE	Count neighbor #4
0196	15 16 17	Point to column of 3 bytes to right
0199	05 FE DE	Count neighbor #3
019C	06 FE DE	Count neighbor #5
019F	07 FE DE	Count neighbor #8
01A2	DC	Set next generation
01A3	19	Advance byte pointer in next generation

TEST IF AT RIGHT MARGIN OR PAGE END

ADDR	CODE	COMMENT
01A4	86 FE FE FE FE FE	If byte pointer R(6).0, when shifted left by 5 bits, equals "E0", then you are at the edge. Else continue looping
01AA	FB E0	
01AC	3A 36	
01AE	15 15 16 16 17 17	Advance pointers to next line
01B4	19 19	Do same for next generation page
01B6	86 FB F9	If R(6).0 points to F9, you are at the end of the page. Else continue loop
01B9	3A 36	

DISPLAY NEXT GENERATION

ADDR	CODE	COMMENT
01BB	F8 01 A5	Reset R(5)
01BE	F8 09 A6 A9	Reset R(6) and R(9)
01C2	F8 11 A7	Reset R(7)
01C5	0D FB 02	If video routine displays page 02, change it to 03. Else change back to 02
01C8	3A D5	
01CA	F8 03 B5 B6 B7 5D	Set current generation pointers to page 03
01D0	F8 02 B9	Set next generation pointer to page 02
01D3	30 36	Go back to the beginning
01D5	F8 02 B5 B6 B7 5D	Set current generation pointers to page 02
01DB	F8 03 B9	Set next generation pointer to page 03
01DE	30 36	Go back to the beginning

```
0000 3060 00F8 01BD A5F8 19AD F802 5DB5 B6B7
0010 F811 A7F8 00AB BEBC F809 A6A9 F831 ACF8
0020 03B9 F851 AEC0 0136 0000 0000 7AF8 00AB
0030 D38B FB02 324C 8BFF 0233 43F8 00F6 097E
0040 5930 2C8B 7D03 3B3B F8FF 303D 3148 303B
0050 D33B 501B 3050 0000 0000 0000 0000 0000
0060 90B1 B2B3 B4F8 91A3 F88F A2F8 71A1 D372
0070 7022 7822 52C4 C4C4 F802 B0F8 00A0 80E2
0080 E220 A0E2 20A0 E220 A03C 7E30 6F00 0000
0090 00E2 61E5 F803 B5F8 FFA5 F800 7395 FB01
00A0 3A9A F802 B5F8 08A5 E6F8 00B6 F8FF A67B
00B0 3FB0 6C64 267A 85F4 A537 B9E5 3FBC 6C64
00C0 85FC 07A5 F6F6 F6FB 1F37 C93A BC30 A200
00D0 F860 A0F8 90A2 93B0 B252 7000 0000 0000
00E0 F801 B0B2 F830 A293 A052 7000 0000 0000
00F0 0000 0000 0000 0000 0000 0000 0000 0000
0100 90B1 B2B3 B4F8 31A3 F82F A2F8 11A1 D372
0110 7022 7822 52C4 C4C4 F802 B0F8 00A0 80E2
0120 E220 A0E2 20A0 E220 A03C 1E30 0F00 0000
0130 00E2 61C0 0003 05FE DEFE DE07 FEDE FEDE
0140 2505 F6DE 1526 06F6 DE16 2707 F6DE 1706
0150 FE3B 547B FEDE DCF8 06A8 05AA 06BA 07BB
0160 8AFE DEFE DEFE DE9B FEDE FEDE FEDE 9AFE
0170 DEFE 3B75 7BFE DEDC 8AFE AA9A FEBA 9BFE
0180 BB28 883A 6005 F6DE F6DE 07F6 DEF6 DE06
0190 F63B 947B F6DE 1516 1705 FEDE 06FE DE07
01A0 FEDE DC19 86FE FEFE FEFE FBE0 3A36 1515
01B0 1616 1717 1919 86FB F93A 36F8 01A5 F809
01C0 A6A9 F811 A70D FB02 3AD5 F803 B5B6 B75D
01D0 F802 B930 36F8 02B5 B6B7 5DF8 03B9 3036
```

JINGLE BELLS????

We at Questdata realize it's a little early to be thinking about Christmas, but it does seem to "pop up" before you know it. Therefore, this is a little reminder, to all you creative geniuses, that we need "holiday type" programs. We need them in the near future in order that we may review them for publication.

Many thanks,

QUESTDATA STAFF

ANNOTATED BIBLIOGRAPHY

by
Richard H. Johnson

Beringer, John "Disassembler for the 1802," Kilobaud/ Microcomputing, #43, p. 196 (July 1980).
Source listing of a 2 K disassembler. Generates a video display of output.

Bronstein, Martin "A Hex Keyboard with Applications for the 1802 Elf," Dr. Dobb's Journal, 5, #3, p. 40 (1980).

Bunn, John R., "Memory-Checking Program for the 1802," Kilobaud/Microcomputing, #44, p. 162 (August 1980).
Flow chart and machine language listing.

Cotter, Robert J., "Interfacing the Elf II," Kilobaud, #24, p. 40 (December 1978).
Hex address displays and 61-byte operating system.

Cotter, Robert J., "Programming the 1802," Kilobaud/ Microcomputing, #27, p. 122 (March 1979).
Integer addition, subtraction, and multiplication.

Cotter, Robert J., "The Amazing 1802: D/A and A/D Applications," Kilobaud, #20, p. 102 (August 1978).
Elvish hardware and software for 8-bit D/A (digitial-to-analog) conversion, 12-bit D/A, and 8-bit A/D.

Cotter, Robert J., "The Elf EPROMmer," Kilobaud/ Microcomputing, #38, p. 76 (February 1980).
Hardware and software for programming and verifying 2708 EPROM's.

Devaux, Edward C., "PIXIE Animation Program," Popular Electronics Magazine, July 1977, p. 42.
Rolling and shifting display under keyboard control.

Dolce, Larry "COSMAC Double Play," Kilobaud/ Microcomputing, #29, p. 66 (May 1979).
Review of the RCA COSMAC VIP.

Dolce, Larry "RCA's VIP Tiny BASIC," Kilobaud/ Microcomputing, #40, p. 111 (April 1980).
Review of Tiny BASIC on ROM for the RCA COSMAC VIP.

Duntemann, Jeff "Onward with the COSMAC Elf," Kilobaud/Microcomputing, #26, p. 66 (February 1979).
Hardware: Multipage expansion of basic Elf and memory address display. Software: Register display program and EHOPS-65K, a 58-byte operating system for an expanded Elf.

Feldman, Michael A. and Fay A. Dion, "Computerized Climate Control," Kilobaud/Microcomputing, #26, p. 38 (February 1979).
Hardware and 1/4 K software for control of a thermostat for an entire week.

Floersch, Gene "1802 EPROM Programming," Kilobaud/ Microcomputing, #39, p. 146 (March 1980).
Hardware and software for programming and verifying Intel 2716 EPROM's.

Haberhern, William J. Jr., "RCA Tries Again . . . with the 1802," Kilobaud, #2, p. 90 (February 1977).
Describes the 1802-based UC1800 microcomputer manufactured (briefly?) by Infinite, Inc.

Harris, Gregory "Elf Meets a New Friend," Kilobaud/ Microcomputing, #43 p. 54 (July 1980).
Interfacing basic Elf to SS-50 bus.

Hudspeth, Forrest and James Champ, "Put Something Super in Your Life," Kilobaud/Microcomputing, #28, p. 100 (April 1979).
Review of the Quest Super Elf.

Hutchinson, Thomas E. "Modify Your COSMAC Elf," Kilobaud, #23, p. 108 (November 1978).
Addition of sophisticated keyboard hardware which includes capability to jump to a desired location with the 1802 in the load mode.

Hutchinson, Thomas E. "The Cosmac Connection Part 1", 73 Amateur Radio Magazine, Jan. 1979 p. 102; Part 2, Feb. 1979 p. 106
Schematics, Flowcharts and Programs for an 1802 based machine as a electronic keyer.

McCormick, Edward M. "A Tic-Tac-Toe Game for your Elf Computer," Popular Electronics Magazine, November 1978, p. 98.
Requires 1K of memory. Describes addition and use of light pen.

McCormick, Edward M. "How to Upgrade a Basic Elf Microcomputer," Popular Electronics Magazine, February 1978, p. 65.
Hardware: Cassette I/O and 20-mA current loop. Programs: Memory-to-cassette, cassette-to-memory, teletype I/O, memory prenumbering, music, frequency counter, interval timer.

McCormick, Edward M. "Promable 1 K Operating system for RCA Elfs," Dr. Dobb's Journal, 2, #9, p. 34 (1977).

McCormick, Edward M. "Utilities & Music on the COSMAC Elf," Dr. Dobb's Journal, 2, #9, p. 30 (1977).

Melton, Henry "The 1802 Op Codes," Byte Magazine, 4, #6, p. 146 (1979).
Gives op codes in a 16 x 16 table.

Martin, Meyer "Expanding the Elf II," Popular Electronics Magazine, March 1978, p. 62. (PE Reprint #41101).
Hardware: ROM monitor, N-line decoder, parallel I/O port (note: 8212's can be substituted for 1852's). Software: Machine language listing only no source listing, of Netronics 256-byte prommable monitor program.

Meyerle, George "Microprocessor Applications for the 1980's . . . It's a whole New Ball Game!," Popular Electronics Magazine, May 1980, p. 41.
Part 1 of a microprocessor training course. Part 1 discusses the 1802, other microprocessors to be discussed in the future. Not very useful for anyone already familiar with 1802 programming.

Moews, Paul C. "Programs for the COSMAC Elf: Graphics," published by Paul C. Moews.
128-, 64-, and 32-byte display routines; 48-byte variable resolution display routine, horse race game, pattern generator program, stop watch program, twelve-hour clock program. Written for basic 1/4 K Elf. 23 pages.

Moews, Paul C. "Programs for the COSMAC Elf: Interpreters," published by Paul C. Moews.
Demonstration interpreter for basic 1/4 K Elf. Source listing of CHIP-8 interpreter for 1 1/4 and 4 K Elves. Machine language listing of relocatable CHIP-8 interpreter for 4 K Elf. Moews has added several instructions to the original RCA CHIP-8 interpreter for the VIP such as multiplication and division. 31 pages.

Moews, Paul C. "Programs for the COSMAC Elf: Music and Games," Published by Paul C. Moews.
Music, display subroutine, hex-to-decimal and decimal-to-hex subroutines, register examine subroutine, dice program, random number generator, "Morra" game, Bridgit game, reaction time program, and Tic-Tac-Toe. Written for basic 1/4 K Elf. 44 pages.

Petty, Bob "Putting the 1802 on the S-100 Bus," Kilobaud/Microcomputing, #30, p. 68 (June 1979).
Wirewrapping of an 1802 CPU board for the S-100 bus. Uses the RCA UT4 monitor on ROM.

Petty, R.W. "1802 PILOT," Kilobaud/Microcomputing, #31, p. 78 (July 1979).
Machine language listing only of a 2K PILOT interpreter. Includes a text editor and uses subroutines in the UT4 monitor ROM.

Petty, R.W. "Tiny Text Editor for the 1802," Kilobaud/Microcomputing, #36, p. 116 (December 1979).
Machine language listing only of text editor used in Petty's 1802 PILOT. Program occupies 0000 - 02E6 and uses subroutines in UT4 monitor ROM.

Pittman, Tom "DOTS, Part 1," Kilobaud/Microcomputing, #26, p. 84 (February 1979).
TV-Typewriter approach developed by author of Tiny Basic. Pointer, mask, and data tables for variable-width ASCII characters. Flow charts for program.

Pittman, Tom "DOTS, Part 2," Kilobaud/Microcomputing, #28, p. 34 (April 1979).
Source listing and detailed discussion of 826-byte long TV-typewriter program. Displays 7 lines with an average of 18 characters per line for Tiny Basic programs. All ASCII characters plus carriage return, line feed, form feed, space, and backspace control codes.

Popiel, Glen A. "Elfish Ideas," Kilobaud/Microcomputing, #34, p. 154 (October 1979).
Hardware and software for use of a UART to speed up serial I/O.

Pottinger, Hardy J., "Double Play," Kilobaud/Microcomputer, #29, p. 70 (May 1979).
Another review of the RCA COSMAC VIP.

Price, Gary H., "Clean Starts for COSMAC 1861 Video Output," Dr. Dobb's Journal, 5, #7, p. 14 (1980).

RCA Publication MPM-201B. "CDP1802 User Manual"

RCA Publication VP-710. "Game Manual,"
Programs in CHIP-8. Blackjack, biorhythm, pinball, bowling, plus ten more.

RCA Publication VP-720. "Game manual II,".

RCA Publication MPM-920A. "Instruction Summary for the CDP1802 COSMAC Micro-processor,"

RCA Publication VP-311. "VP-711 Instruction Manual,"
For VIP computer. Documents original CHIP-8 language. Includes twenty video games.

RCA Publication VP-320. "VP-711 User Guide Manual,"
For VIP computer. Discusses CHIP-8 usage.

Redstone, Allan "Analog-to-Digital Conversion," Kilobaud/Microcomputing, #40, p. 28 (April 1980).
Hardware: 8-bit A/D and D/A conversion; use of potentiometers for analog input. Software: A/D conversion routine and Etch-a-Sketch program.

Shroyer, Donald R. "Elf Video from Netronics," Kilobaud/Microcomputing, #41, p. 142 (May 1980).
Review of Netronics Video display board kit.

Shroyer, Donald R. "Play 'Space Battle' On Your Video Monitor," Popular Electronics, June 1980, p. 61.
Players can move space ships up or down. Requires push button switches attached to EF3 and EF4 for firing.

Thurm, J.H. "A COSMAC CDP1802, CDP1854 Monitor," Kilobaud/Microcomputing, #41, p. 132 (May 1980).
Hardware: Use of CDP1854 UART. Software: Source listing of EC-Bug, an 856-byte prommable monitor.

Tully, Ray "Elfin Echoes," Kilobaud/Microcomputing, #35, p. 127 (November 1979).
Printed circuit board pattern for monitoring Elf II cassette I/O.

Wasserman, Paul "A Floating Point Subroutine Package for the 1802," Dr. Dobb's Journal, 4, #7, p. 17 (1979).
Machine language listing only, no source listing. Requires 2K of memory; written for location 0100 - 08FF, but instructions given for relocation. User must provide code for the Standarad Call and Return Technique (SCRT) given in RCA's "CDP1802 User Maual". Range: 10-39 to 10+38. Operations: Addition, subtraction, multipliction, division, and finding square roots. 42 variables. Typo: the second

Weisbecker, Joseph A., "A Practical, Low-Cost, Home/School Microprocessor System," Computer magazine, August 1974, p. 20; reprinted in "Dr. Dobb's Journal of Computer Calisthenics & Orthodontia, 2, #5, p. 34 (1977).
Discusses the FRED system, a 1K prototype of the RCA COSMAC VIP.
calling address in Fig. 3 should read 02E9.

Weisbecker, Joseph A. "A Simplified Microprocessor Architecture," Computer magazine, March 1974, p. 41.
Discusses the COSMAC microprocessor design philosophy.

Weisbecker, Joseph A. "Build the COSMAC 'Elf,' Part 1," Popular Electronics Magazine, August 1976, p. 33 (PE Reprint #40857).
Construction of basic Elf; description of architecture; introduction to programming.

Weisbecker, Joseph A. "Build the COSMAC 'Elf,' Part 2," Popular Electronics Magazine, September 1976, p.37 (PE Reprint #40858).
Hardware: LED bus display, parallel I/O port, hex keyboard. More on programming.

Weisbecker, Joseph A. "Build the COSMAC 'Elf,' Part 3," Popular Electronics Magazine, March 1977, p. 61 (PE Reprint #40859).
ETOPS-256, a 32-byte operating system for basic 1/4 K Elf with toggle switches or for basic 1/4 K Super Elf. EHOPS-256, a 74-byte operating system for basic 1/4 K Elf with scanned hex keyboard. Memory expansion to 1 1/4 K.

Weisbecker, Joseph A. "Build the PIXIE Graphic Display," Popular Electronics Magazine, July 1977, p. 41 (PE Reprint #40870).
Addition of 1861 video display to basic Elf.

Weisbecker, Joseph A. "COSMAC VIP, the RCA Fun Machine," Byte Magazine, 2, #8, p. 30 August (1977).

For availability of magazine reprints and back issues write to:

Dr. Dobb's Journal
1263 El Camino Real
Box E
Menlo Park, CA 94025

Popular Electronics Reprints
P.O. Box 278
Pratt Station
Brooklyn, NY 11205

Byte Magazine
70 Main St.
Peterborough, NH 03456
Attn: Back Issues

Kilobaud/Microcomputing Back Issues Catalog
Microcomputing
Peterborough, NH 03458
Toll-free telephone number: 1-800-258-5473

The Paul C. Moews booklets and several of the RCA publications are available from Quest Electronics. The RCA publications may also be ordered directly from RCA Microcomputer Products
Customer Service
New Holland Avenue
Lancaster, PA 17604

Note from Editor:

If you come across any discrepancies or omissions, please let us know and we will publish corrections in a future issue. Your participation is greatly appreciated by the Questdata staff.

BIORHYTHM

by
Gary Gehlhoff

Recently there has been some controversy concerning the use of biorhythms as they pertain to our daily lives. I decided to find out how they related to me by writing a program to calculate the current level of each biorhythm and the composite of the three.

Two inputs are required; first your birth date (month/day/year), and second, today's date (month/day/year). The three biorhythm levels along with the composite are then the output.

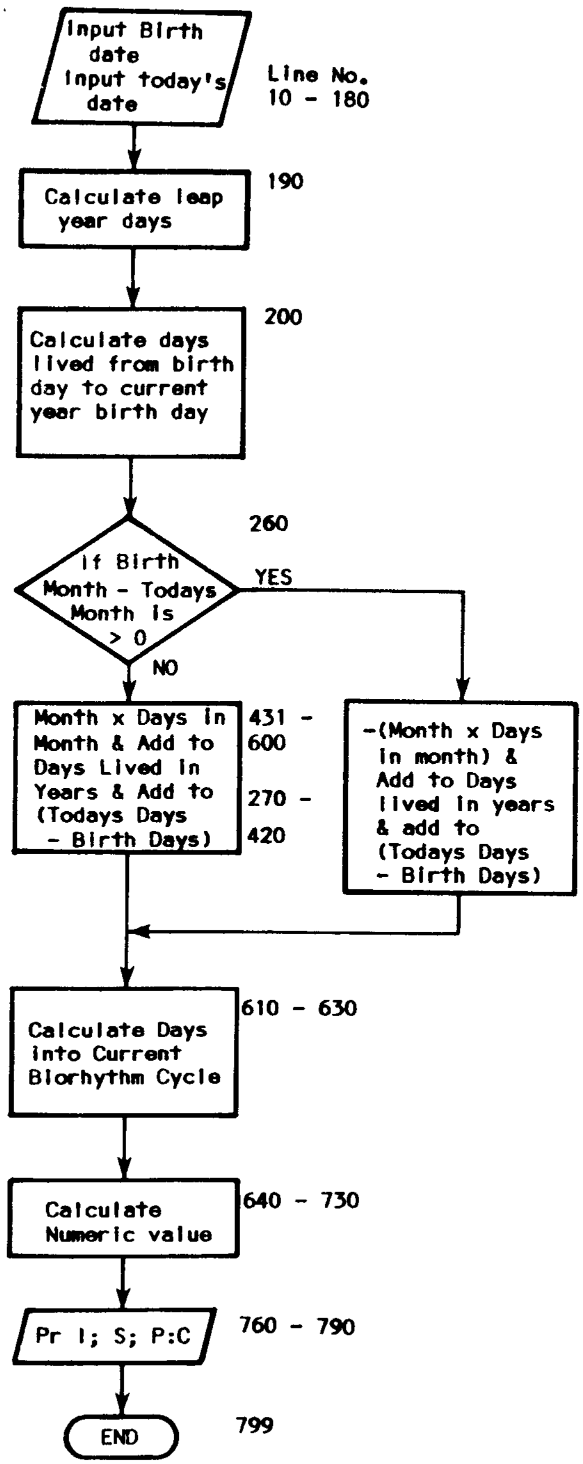
Lines 10 thru 180 are the input section. Line 190 calculates the number of leap year days since your birth day that are to be added. Line 260 determines whether the current month is before of after your birth date. Lines 210 thru 600 calculate the total number of days lived. Lines 610, 620, & 630 determine the number of days in each biorhythm cycle. Lines 650 thru 730 assign a numerical value to each one of the cycles (numbers are expressed as integers rather than decimals, as in traditional biorhythm plots). Finally, lines 760 thru 790 are the output.

Also included are a flow diagram and check examples.

CHECK EXAMPLES

BIRTH DATE	2-22-46	BIRTH DATE	2-22-46
TODAYS DATE	1-15-79	TODAYS DATE	2-23-79
DAYS LIVED		DAYS LIVED	
YEARS	12045	YEARS	12045
LEAP YEAR	8	LEAP YEAR	8
DAYS	.37	DAYS	1
	12015		12054
I= 12015/33 = 364 R3		I= 12054/33 = 365 R9	
J= +9		J= 22	
S= 12015/28 = 429 R3		S= 12054/28 = 430 R14	
T= +12		T= 0	
P= 12015/23 = 522 R9		P= 12054/23 = 524 R2	
Q= +5		Q= 4	
C= 26		C= 26	
Variables		Variables	
B= 12045		B= 12045	
L= 8		L= 8	
W= -28		W= 0	
F= -7		F= 1	
A= 12015		A= 12054	

```
10   REM BIORHYTHM------PROGRAM
20   PR "BIORHYTHM"
30   PR .
40   PR "BIRTH DATE";
50   PR "------MONTH-";
60   INPUT M
70   PR "-------DAY---";
80   INPUT D
90   PR "-----YEAR---";
100  INPUT Y
110  PR
120  PR "TODAYS DATE"
130  PR "------MONTH-";
140  INPUT N
150  PR "-------DAY---";
160  INPUT E
170  PR "-----YEAR---";
180  INPUT Z
190  L=(Z-Y)/4
200  B=(Z-Y)*365
210  REM
215  U=0
220  F=0
230  W=0
240  O=M-N
250  REM
260  IF O>0 GOTO 431
270  IF M=N THEN GOTO 415
280  IF M=1 THEN W=W+31
290  IF M=3 THEN W=W+31
300  IF M=5 THEN W=W+31
310  IF M=7 THEN W=W+31
320  IF M=8 THEN W=W+31
330  IF M=10 THEN W=W+31
340  IF M=12 THEN W=W+31
350  IF M=4 THEN W=W+30
360  IF M=6 THEN W=W+30
370  IF M=9 THEN W=W+30
380  IF M=11 THEN W=W+30
390  IF M=2 THEN W=W+31
400  M=M+1
410  GOTO 270
415  F=E-D
420  A=L+B+W+F
430  GOTO 610
431  N=N-1
432  M=M-1
433  IF M=0 THEN M=12
434  IF N=0 THEN N=12
435  IF M=N THEN GOTO 590
436  IF M=1 THEN W=W-31
437  IF M=3 THEN W=W-31
438  IF M=5 THEN W=W-31
439  IF M=7 THEN W=W-31
440  IF M=8 THEN W=W-31
500  IF M=10 THEN W=W-31
510  IF M=12 THEN W=W-31
520  IF M=4 THEN W=W-30
530  IF M=6 THEN W=W-30
540  IF M=9 THEN W=W-30
550  IF M=11 THEN W=W-30
560  IF M=2 THEN W=W-28
570  M=M-1
580  GOTO 433
590  F=E-D
600  A=L+B+W+F
610  P=A-(A/23)*23
620  S=A-(A/28)*28
630  I=A-(A/33)*33
640  REM ASSUME AVG PERSON
650  IF I>=1 IF I<=8 THEN J=I*3
660  IF I>=9 IF I<=24 THEN J=19+(10-I)*3
670  IF I>=25 IF I<=33 THEN J= -24+(I-25)*3
680  IF S>=1 IF S<=7 THEN T=S*4
690  IF S>=8 IF S<=21 THEN T=24-(S-8)*4
700  IF S>=21 IF S<=28 THEN T= -24+(S-22)*4
710  IF P>=1 IF P<=5 THEN Q=P*2
720  IF P>=6 IF P<=17 THEN Q=11+(6-P)*2
730  IF P>=18 IF P<=23 THEN Q= -12+(P-17)*2
740  C=J+T+Q
750  PR
760  PR "INTELLECTUAL-",J
770  PR "SENSITIVITY--",T
780  PR "PHYSICAL-----",Q
790  PR "COMPOSITE----",C
799  END
```



SUPER ELF CASSETTE TAPE READER

by
Van C. Baker

If you have a need to read cassettes generated by the Super Monitor, or wish to read Quest-supplied cassette software, but need to relocate the tape data or program in a location other than that from which the tape was originally created, use the program listed below. To use the routine, load the program into memory, noting that it may be located beginning on any page boundary. Also note that location XX01 (where "XX" represents the page number) must contain the byte defining the page in which the routine is located. The program listed below, for example, runs in page zero, hence, byte 0001 is "00".

The program assumes flag line EF3 is used for the cassette serial input; if your system differs from this convention, patch in the appropriate EFn conditional branch instructions at XX38,XX3F,XX4F,XX8E,XX95 and XXA2.

To use the program, proceed as follows:

1. Execute the program using your monitor or other means. It does not matter what register is the program counter when the program is entered.

2. Note that "AA" will be displayed on the hex display. Enter the high byte of the starting address into which the tape contents are to be loaded. Press the "I" key on the hex keypad.

3. Enter the low byte of the starting address. Press the "I" key.

4. Enter file number to be read (01-FF). Press the "I" key.

5. Start the recorder (on playback).

As the tape advances, the current file number being skipped (if the file entered in 4 was greater than 1) will be displayed until the requested file is reached. As the contents of the tape are being loaded into the requested memory locations, the hex display will rapidly flicker. When the tape has been read, "AA" will appear on the hex display. At this point, you may load another tape by proceeding with step 2 above. Since the Super Monitor tapes have a record defining the total number of bytes on the tape, it is necessary only to input the starting address for the cassette load; the program does the rest. Use caution, however, to avoid loading the cassette data over the tape read program.

If "EE" should appear on the display while the tape is being read, it indicates that a read error (e.g., a parity error) occured. The address of the byte at which the error occured can be determined by examining locations XXFE and XXFF (High and low address bytes, respectively). To recover from a read error, press the "I" key and proceed from step 2 above.

```
0000          ;***************************************
0000          ;
0000          ;          MANUAL CASSETTE LOAD ROUTINE
0000          ;
0000          ;                 BY V C BAKER
0000          ;
0000          ;      * THIS ROUTINE READS A STANDARD
0000          ;      * "SUPER MONITOR" CASSETTE,
0000          ;      * LOADING THE DATA INTO USER-
0000          ;      * DESIGNATED MEMORY AREA.
0000          ;
0000          ;
0000          ;      * TO USE:
0000          ;      *
0000          ;      *      (1)  LOAD THE FOLLOWING
0000          ;      *      ROUTINE INTO MEMORY.
0000          ;      *      ALTHOUGH THE ROUTINE
0000          ;      *      LISTED HERE STARTS AT 0000
0000          ;      *      (HEX), IT MAY BE LOCATED
0000          ;      *      ANYWHERE AS LONG AS IT
0000          ;      *      BEGINS ON A PAGE BOUNDARY,
0000          ;      *      I.E., AT XX00. PATCH IN
0000          ;      *      THE ACTUAL PAGE NUMBER AT
0000          ;      *      BYTE XX01.
0000          ;      *
0000          ;      *      (2)  EXECUTE THE PROGRAM
0000          ;      *      USING ANY REGISTER FOR THE
0000          ;      *      PROGRAM COUNTER.
0000          ;
0000          ;      *      (3)  WHEN "AA" IS DIS-
0000          ;      *      PLAYED ON THE HEX DISPLAY,
0000          ;      *      ENTER THE FOLLOWING USING
0000          ;      *      THE HEX KEYPAD:
0000          ;      *
0000          ;      *        (A)  HIGH BYTE (MSH) OF
0000          ;      *        STARTING ADDRESS INTO
0000          ;      *        WHICH TAPE CONTENTS ARE
0000          ;      *        TO BE LOADED.
0000          ;      *
0000          ;      *        (B)  PRESS THE "I" KEY.
0000          ;      *
0000          ;      *        (C)  LOW BYTE (LSH) OF
0000          ;      *        STARTING ADDRESS.
```

```
ADDR CODE     LABEL   OPCODE OPERAND COMMENT
0000          ;       *
0000          ;       *         (D)  PRESS THE "I" KEY.
0000          ;       *
0000          ;       *         (E)  TAPE FILE NUMBER
0000          ;       *         (1-FF)
0000          ;       *
0000          ;       *         (F)  PRESS THE "I" KEY.
0000          ;       *
0000          ;       *       (4)  START THE RECORDER
0000          ;       *       ON PLAYBACK.
0000          ;       *
0000          ;       *       (5)  AS THE TAPE IS READ,
0000          ;       *       ITS CONTENTS WILL BE
0000          ;       *       LOADED INTO THE DESIGNATED
0000          ;       *       MEMORY LOCATIONS.  SINCE
0000          ;       *       THE STANDARD SUPER MONITOR
0000          ;       *       TAPES CONTAIN RECORDS
0000          ;       *       GIVING THE NUMBER OF BYTES
0000          ;       *       STORED ON THE TAPE, THE
0000          ;       *       ROUTINE WILL STORE THE
0000          ;       *       ENTIRE CONTENTS OF THE
0000          ;       *       TAPE AT CONSECUTIVE MEMORY
0000          ;       *       LOCATIONS.  CAUTION MUST
0000          ;       *       BE USED TO ASSURE THAT THE
0000          ;       *       CASSETTE ROUTINE ITSELF IS
0000          ;       *       NOT WRITTEN OVER AS THE
0000          ;       *       TAPE IS LOADED!  IF FILE
0000          ;       *       2 OR GREATER WAS SPECIFIED
0000          ;       *       IN STEP 3-K, THE CURRENT
0000          ;       *       FILE NUMBER BEING SKIPPED
0000          ;       *       WILL BE DISPLAYED UNTIL
0000          ;       *       THE REQUESTED FILE IS
0000          ;       *       REACHED.  THE DISPLAY WILL
0000          ;       *       FLICKER AS THE TAPE IS
0000          ;       *       BEING READ.
0000          ;       *
0000          ;       *       (6)  WHEN THE TAPE HAS
0000          ;       *       BEEN SUCCESSFULLY READ,
0000          ;       *       "AA" WILL APPEAR ON THE
0000          ;       *       HEX DISPLAY.  IF "EE"
0000          ;       *       APPEARS, A READ ERROR
0000          ;       *       OCCURED  (E.G., A PARITY
0000          ;       *       ERROR).  TO RECOVER FROM
0000          ;       *       THE ERROR, REWIND THE
0000          ;       *       TAPE, PRESS THE "I" KEY,
0000          ;       *       AND START OVER FROM STEP 3
0000          ;       *         THE ADDRESS OF THE BYTE
0000          ;       *       BEING READ WHEN THE ERROR
0000          ;       *       OCCURED CAN BE DETERMINED
0000          ;       *       BY INSPECTING MEMORY LOC-
0000          ;       *       ATIONS XXFE (FOR THE HIGH
0000          ;       *       BYTE OF ADDRESS) AND XXFF
0000          ;       *       (FOR THE LOW BYTE).
0000          ;
0000
0000 CODE     LABEL   OPCODE OPERAND COMMENT
0000                  ORG    #0000    ; Start of program
0000          START:  EQL    #0000
0000          ONECT:  EQL    #000D    ; One-bit timing
0000                                  ; value
0000          LDRCT:  EQL    #000A    ; Leader one's
0000                                  ; timing
0000          ;
0000          ;
0000 F8 00    GO:     LDI    START    ; Load page number
0002 B3               PHI    R3       ; for this routine
0003 F8 07            LDI    INIT     ; initialization
0003                                  ; address
0005 A3               PLO    R3
0006 D3               SEP    R3       ; SEP to PC = R3
0007 93       INIT:   GHI    R3       ; Initialize
0007                                  ; registers.
```

```
ADDR CODE     LABEL   OPCODE OPERAND COMMENT
0008 BB               PHI    RB       ; RB = BOTM2 PC.
0009 B8               PHI    R8       ; R8 = DI PC.
000A B6               PHI    R6       ; R6 = ERR PC.
000B B5               PHI    R5       ; R5 = HEXIN PC.
000C B2               PHI    R2       ; R2 = Stack
000C                                  ; pointer
000D          ;
000D          ;
000D          ;
000D F8 6A            LDI    BOTM2    ; BOTM2 address
000F AB               PLO    RB
0010 F8 8A            LDI    DI       ; DI address
0012 A8               PLO    R8
0013 F8 7A            LDI    ERR      ; ERR address
0015 A6               PLO    R6
0016 F8 FF            LDI    #FF      ; Stack at XXFF
0018 A2               PLO    R2
0019 E2               SEX    R2       ; SP = R2
001A F8 71            LDI    HEXIN    ; HEXIN address
001C A5               PLO    R5
001D          ;
001D          ;       * BEGIN MAIN PROGRAM
001D          ;
001D          ;
001D          ;       * DISPLAY "AA" ON LED'S
001D          ;
001D F8 AA    BEGIN:  LDI    #AA      ; Load "AA"
001F 52               STR    R2       ; Store it
0020 64               OUT    4        ; Output it.
0021 22               DEC    R2       ; Reposition SP.
0022          ;
0022          ;       * GET STARTING ADDRESS FROM
0022          ;       * KEYPAD
0022          ;
0022 D5               SEP    R5       ; Get MSH of
0022                                  ; address
0023 BE               PHI    RE
0024 D5               SEP    R5       ; Get LSH
0025 AE               PLO    RE       ; Start address
0025                                  ; in RE.
0026          ;
0026          ;       * GET FILE NUMBER FROM KEYPAD
0026          ;
0026 D5               SEP    R5       ; Get file number
0026                                  ; (1-FF).
0027 A4               PLO    R4       ; Save in R4.
0028          ;
0028          ;       * SET TIMING VALUE FOR A ONE BIT
0028          ;
0028 F8 0D            LDI    ONECT
002A B9               PHI    R9       ; "1" TIMING VALUE
002A                                  ; IN R9.1
002B          ;
002B          ;
002B F8 00            LDI    #00      ; Initialize
002B                                  ; current file
002B                                  ; number
002D          ;
002D          ;
002D 52       FTST:   STR    R2       ; Save current
002D                                  ; file number
002E 64               OUT    4        ; Display it.
002F 22               DEC    R2
0030 84               GLO    R4       ; Get requested
0030                                  ; file number.
0031 F3               XOR             ; Check if equal
0031                                  ; to current.
0032 32 4F            BZ     RDF      ; Branch if so.
0034          ;
0034          ;       * CHECK FOR LEADER
0034          ;
```

```
ADDR CODE     LABEL    OPCODE OPERAND COMMENT
0034 F8 0A    LDFND:   LDI    LDRCT    ; Load test value
0034                                   ; for leader.
0036 B7                PHI    R7       ; Save in R7.1
0037 DB       OZTST:   SEP    RB       ; Check for abort
0037                                   ; (BOTM2)
0038 36 37             B3     OZTST    ; Find data pulse
0038                                   ; transition
003A 99                GHI    R9       ; Test pulse width
003B FF 01    PTIM:    SMI    #01      ; is it too long
003B                                   ; for a one?
003D 3B 47             BNF    ZER      ; JMP if so.
003F 3E 3B             BN3    PTIM     ; else loop till
003F                                   ; pulse is over.
0041 97                GHI    R7       ; This bit is a
0041                                   ; "1". Count the
0042 32 37             BZ     OZTST    ; ones till LDRCT
0042                                   ; of them are
0044          ;                        ; found. (Verify
0044                                   ; leader located.)
0044 27                DEC    R7
0045 30 37             BR     OZTST
0047 97       ZER:     GHI    R7       ; the recieved bit
0047                                   ; must be a "0".
0048 3A 34             BNZ    LDFND    ; Is leader done
0048                                   ; yet?
004A 02                LDN    R2       ; Yes. Increment
004A                                   ; file counter and
004B FC 01             ADI    #01      ; then go check
004B                                   ; file number to
004D 30 2D             BR     FTST     ; see if it is the
004D                                   ; one requested.
004F          ;
004F          ;        * REQUESTED FILE FOUND
004F          ;
004F 3E 4F    RDF:     BN3    RDF      ; Wait for end of
004F                                   ; leader bit.
0051          ;
0051          ;        * NOW READ TAPE FOR START ADDRESS
0051          ;
0051 D8                SEP    R8       ; Read (But
0051                                   ; ignore) start
0051                                   ; address.
0052 D8                SEP    R8       ; Finish reading
0052                                   ; start address.
0053          ;
0053          ;        * NEXT, READ NUMBER OF BYTES
0053          ;        * STORED ON TAPE
0053          ;
0053 D8                SEP    R8
0054 B6                PHI    R6
0055 D8                SEP    R8
0056 A6                PLO    R6
0057          ;
0057          ;        * SET UP FOR DATA READ
0057          ;
0057 26       CNTSET:  DEC    R6       ; Set up byte
0057                                   ; counter
0058 96                GHI    R6       ; so that when
0059 FC 01             ADI    #01      ; all bytes have
0059                                   ; been read.
005B B6                PHI    R6
005C          ;
005C D8       RDOPT:   SEP    R8       ; Read a data byte
005D 5E                STR    RE       ; Store it in
005D                                   ; memory
005E 1E                INC    RE       ; Increment memory
005E                                   ; address
005F 8E                GLO    RE
0060 52                STR    R2
0061 64                OUT    4        ; Display low-
0061                                   ; order address.
0062 22                DEC    R2       ; Reposition SP
0063          ;
0063          ;        * CHECK IF FINISHED
0063          ;
```

```
ADDR CODE     LABEL    OPCODE OPERAND COMMENT
0063 26                DEC    R6       ; Decrement byte
0063                                   ; counter
0064 96                GHI    R6
0065 3A 5C             BNZ    RDOPT    ; Loop back if not
0065                                   ; done
0067 30 1D             BR     BEGIN    ; Else, go display
0067                                   ; "AA".
0069          ;
0069          ;
0069          ;
0069          ;***************************************
0069          ;***************************************
0069          ;
0069          ;****** BOTM2 ROUTINE -- ABORT THE READ
0069          ;
0069 D3       BTMRET:  SEP    R3       ; Return
006A 3F 69    BOTM2:   BN4    BTMRET   ; Return if I-Key
006A                                   ; is not in
006C 37 6C    WAIT:    B4     WAIT     ; software "de-
006C                                   ; bounce" step.
006E 30 00             BR     GO       ; Abort. Start
006E                                   ; all over
0070          ;
0070          ;***************************************
0070          ;***************************************
0070          ;
0070          ;******HEXIN ROUTINE -- HEX KEYPAD INPUT
0070          ;
0070 D3       HEXRET:  SEP    R3       ; Return
0071 3F 71    HEXIN:   BN4    HEXIN    ; Wait for I-key
0071                                   ; in.
0073 6C                INP    4        ; Get input byte
0074 64                OUT    4        ; Display it
0075 37 75    WAIT2:   B4     WAIT2    ; Wait till key
0075                                   ; released
0077 22                DEC    R2
0078 30 70             BR     HEXRET   ; Return
007A          ;***************************************
007A          ;***************************************
007A          ;
007A          ;****** ERR ROUTINE -- ERROR "PROCESSOR"
007A          ;
007A F8 EE    ERR:     LDI    #EE      ; Load "EE"
007C 52                STR    R2
007D 64                OUT    4        ; Display "EE"
007E 22                DEC    R2
007F 8E                GLO    RE       ; Get low address
007F                                   ; byte
0080 73                STXD
0081 9E                GHI    RE       ; Get high address
0081                                   ; byte
0082 52                STR    R2       ; Save error
0082                                   ; address on stack
0083 3F 83    LOOP:    BN4    LOOP     ; loop "till I-key
0083                                   ; is pressed".
0085 37 85    LOOP2:   B4     LOOP2
0087 30 00             BR     GO       ; Start all over
0088                                   ; again
0089          ;***************************************
0089          ;***************************************
0089          ;******DI ROUTINE --READ A BYTE FROM TAPE
0089          ;
0089 DB       DIRET:   SEP    RB       ; Test for abort
0089                                   ; and return
008A F8 08    DI:      LDI    #08      ; Set up counters
008C A7                PLO    R7
008D A9                PLO    R9
008E 36 8E    LHTRAN:  B3     LHTRAN   ; Wait for end of
008E                                   ; current
0090          ;                        ; High input level
0090 99                GHI    R9       ; Test for bit
0090                                   ; value.
0091 FF 01    BTST:    SMI    #01
0093 3B 9A             BNF    AHZER
0095 3E 91             BN3    BTST
```

```
ADDR CODE    LABEL    OPCODE OPERAND COMMENT
0097         ;
0097 27               DEC    R7
0098 30 A4            BR     DNCHK    ; Got a "1" Bit.
009A F8 00   AHZER:   LDI    #00      ; Got a zero bit.
009A                                  ; Test
009C FC 01   TLTST:   ADI    #01      ; For excessive
009C                                  ; pulse width.
009E 3B A2            BNF    HLTRAN
00A0 30 7A            BR     ERR      ; Error
00A2 3E 9C   HLTRAN:  BN3    TLTST
00A4         ;
00A4 89      DNCHK:   GLO    R9       ; Got a byte?
00A5 32 AD            BZ     PARCHK   ; check parity if
00A5                                  ; so.
00A7         ;
00A7 02               LDN    R2       ; Not full byte
00A7                                  ; yet.
00A8 7E               RSHL            ; Insert bit into
00A8                                  ; buffer
00A9 52               STR    R2
00AA 29               DEC    R9
00AB 30 8E            BR     LHTRAN
00AD         ;
00AD 87      PARCHK:  GLO    R7       ; Check for even
00AD                                  ; parity.
00AE F6               SHR
00AF 02               LDN    R2       ; Put data value
00AF                                  ; into D-Reg.
00B0 3B 89            BNF    DIRET    ; Return if parity
00B0                                  ; okay.
00B2 30 7A            BR     ERR      ; Call error
00B2                                  ; routine
00B4
```

Note from the Editor:

The preceeding assembly listing is the output of the Quest Editor Assembler.

```
0000 F800 B3F8 07A3 D393 BBB8 B6B5 B2F8 6AAB
0010 F88A A8F8 7AA6 F8FF A2E2 F871 A5F8 AA52
0020 6422 D5BE D5AE D5A4 F80D B9F8 0052 6422
0030 84F3 324F F80A B7DB 3637 99FF 013B 473E
0040 3B97 3237 2730 3797 3A34 02FC 0130 2D3E
0050 4FD8 D8D8 B6D8 A626 96FC 01B6 D85E 1E8E
0060 5264 2226 963A 5C30 1DD3 3F69 376C 3000
0070 D33F 716C 6437 7522 3070 F8EE 5264 228E
0080 739E 523F 8337 8530 00DB F808 A7A9 368E
0090 99FF 013B 9A3E 9127 30A4 F800 FC01 3BA2
00A0 307A 3E9C 8932 AD02 7E52 2930 8E87 F602
00B0 3B89 307A
```

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

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

 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.

CHECKBOOK RECKONING

by
Gary Gehlhoff

The program title has included the word "reckoning" because when it comes to the end of the month that's my day of reckoning (Or I'm in the RED again).

The program uses the standard method of checkbook balancing, by subtracting the outstanding checks from the sum of the balance shown on the bank statement and the deposits not shown on the statement.

Each input requires the dollars number to be separated by a comma from the cents number. To signal the program that the list of "Outstanding checks" or the list of "Unshown deposits" is complete a 00,00 should be entered.

The program is straightforward with only minor subtleties around creating decimal division (lines 200,320, and 460) and outputting a number that's less than 10 cents (Line 480).

```
10   D=0
20   C=0
30   M=0
40   P=0
50   N=0
60   O=0
70   Q=0
80   R=0
90   PR "CHECKBOOK"
95   PR
100  PR "CHECKS"
110  PR "OUTSTANDING"
120  PR "DOLS, CTS"
130  PR "$";
140  INPUT D,C
150  IF D=0 THEN IF C=0 THEN GOTO 190
160  M=M+D
170  P=P+C
180  GOTO 130
190  M=M+ P/100
200  P=P- P/100 * 100
210  PR "------"
220  PR "TOTAL $";M;".";P
230  PR
240  PR "DEPOSITS NOT SHOWN"
250  PR "$";
260  INPUT D,C
270  IF D=0 THEN IF C=0 THEN GOTO 310
280  N=N+D
290  Q=Q+C
300  GOTO 250
310  N=N+ C/100
320  Q=Q- Q/100 * 100
330  PR "------"
340  PR "TOTAL $";N;".";Q
350  PR
360  PR "BAL FROM STATEMENT"
365  PR "$";
370  INPUT O,R
380  PR
390  PR "CURRENT BALANCE"
400  D=N+ O-M
410  C=R+ Q-P
420  IF C>=0 THEN GOTO 450
430  D= D-1
440  C= C+100
450  D= D+ C/100
460  C=C- C/100 * 100
470  PR "$";D;".";
480  IF C<10 THEN PR "0";
490  PR C
495  PR
499  END
```


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

ADDRESS CORRECTION REQUESTED