

Images and Patterns for Tomorrow

FISH gotta fly, birds gotta swim. Eventually, given human intervention into things, this artistic arrangement is inevitable. With the advent of the microcomputer age, graphics and art are likely to be sliced and served in every conceivable way. What will be the new forms of art, design, architectural drawings, schematics, photographic images, animation, and graphics?

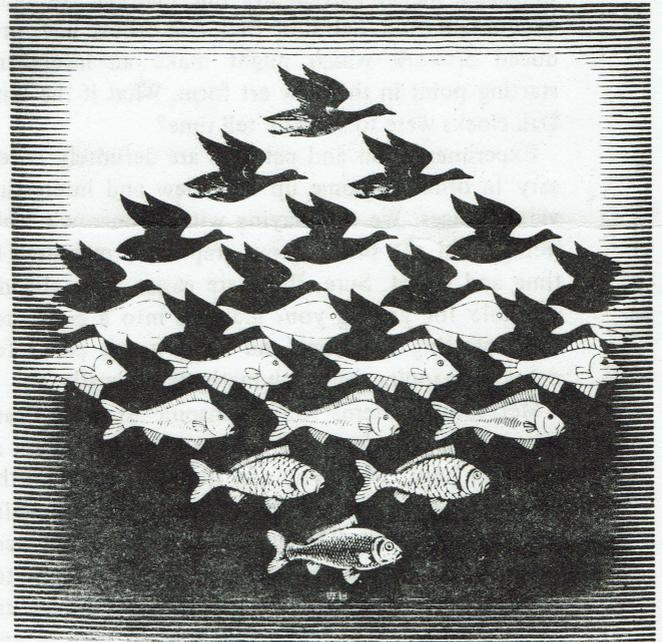
One of the first video art works was "Anti-Gravity Video." This particular piece consists of twenty TV sets hung face down from the ceiling. In order to view the show the viewer has to lie down—that is if he wants to see all the fish swimming across the screens. Such experimentation is not only playful fun, it marks the cutting of a new multifaceted diamond. Video art is here.

The flying-fish artwork is the creation of the foremost video artist, Nam June Paik. Paik is one of the first to venture forth into the new image-making process, and his views carry a lot of weight. In an interview in the recent Winter issue of *Video* he comments, "Home computers and synthesizers are an imaginative use of the TV screen. After Gutenberg invented the printing machine, he printed the Bible. Now that we have home video equipment, we have to create a purpose for it."

Some of the best works of computer art are destined to be done by computer hobbyists, "just for the fun and art of it." Wouldn't it be fun to take a Rube Goldberg cartoon and animate it. You would actually get to see A push B, and the rats running in the cages, and the ball rolling down the platform to . . . This project hasn't been tackled yet. True, there are obstacles in the way of accomplishing video art, but part of the fun of being a computer hobbyist has to be in overcoming obstacles.

One of the things blocking the way to computer art is that it requires a lot of memory. If you want really complex images you will have to have memory—4K makes a good start. Good things have been done with only 256 bytes, however. Paul Moews has shown in his booklet on graphics that there are infinite kaleidoscopic images and billboard images to be made. One thing you don't have to worry about in computer arts is the plunking down of a lot of money for canvas, paint and brushes.

And you sure don't have to worry about drawing a straight line any more. Or circle, ellipse or polygon. Just plot it out on graph paper and it will magically and perfectly appear on the TV screen, when fed into



Sky and Water I, woodcut by M. C. Escher

your willing COSMAC. Another interesting way to accomplish the task is to plot a function. An example of this ingenious technique is given by Michael Tyboriski in *PATTERNS* on page 9 of this *QUESTDATA*. Today, the prophecy of the matchbook cover is at last true, "Even you can become an artist."

So you are an artist. What are you going to do with your new found talents? You are freed of the handicap of not being able to draw a straight line, but you will soon discover that the techniques and details an artist must pay attention to are as demanding as the art of programming. Perhaps you always suspected they don't just dash out Disney cartoons. Planning, strategy and above all, patience are required in an artist.

There are many kinds of art begging to come to life on the TV screen. M. C. Escher makes a good starting point. The *Sky and Water I* woodcut is compelling. The images of bird and fish could blend into all kinds of new patterns if actually flying and swimming. What would happen if the school of fish switched directions? What kind of pattern will the birds make when they flap their wings? Perhaps this woodcut might lend itself to animated Moiré patterns. Note: Moiré patterns are a really interesting special effect which occurs when many thin lines are brought close together—the eye gets all caught up in them. If, even as stills, Escher's work can show the depth and fabric of life, think what they can do when given animation.

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

Recommended reading for M. C. Escher insight is *The Magic Mirror of M. C. Escher* by Bruno Ernst. Escher's art explores mathematical themes, such as Moebius strip and other concepts. About Escher's image experimentation, Ernst says, "To see an Escher print is to discover the infinite possibilities of the human imagination."

Escher's work, with its almost mathematical relationships, makes particularly fine material for video experimentation. Salvador Dali, and others, have produced artwork which might make an interesting starting point in this new art form. What if the limp Dali clocks were to actually tell time?

Experimentation and patience are definitely necessary in order to come up with new and interesting video images. We are playing with tomorrow's technology and the price of our graphics is measured in time and effort. Sure, there are easier ways already available for getting your graphics into a computer than plotting functions and using graph paper for memory mapping, but such methods are costly.

Metacolor Systems (855 Samsome St., San Francisco) rents time on its computer graphics system at \$150 an hour. What you can do, if you live in the vicinity of their studio, is take them a black-and-white or color photo or slide, even a 35mm film or videotape of the subject you wish to have the computer reshape. Once a video camera has digitized the picture it can then be made to overlap itself, turn in any direction, and do other interesting tricks. The new image can expand or diminish, or blur to give the illusion of depth. You have, no doubt, noticed that the MOVIE OF THE WEEK on ABC uses overlapping, expanding, and wondrous graphics.

Television networks are using more and more computer generated logos and attention getters. It is a good bet that you will shortly see these computer effects on TV commercials. Computer renderings also have interesting possibilities for the printed page when they are made into photographs.

Genigraphics, a division of General Electric, offers a higher resolution computer graphics system. This system is not as popular with the graphics consumer. Why? Because an image must be mathematically described to the computer, and this, of course, means a lot of time spent by a "computer image specialist." This new video image manipulator no doubt commands a tidy sum for this work.

Enough examples for a moment, the point is that there are computers out in the real world doing interesting graphics, but we are not a part of this computer action. Must we stand on the sidelines and watch the real action? Are our machines that different from the expensive corporate computers? This is a question which you should decide for yourself. QUESTDATA would like to see reader inputs of interesting software that will make image manipulation possible for COSMACians. Perhaps we can achieve blurring

effects by exposing part of the TV image for more or less time than other parts? Interesting effects can be achieved by switching images (areas mapped in memory) at a fast rate.

What kinds of software does it take to rotate and diminish an object at the same instant? QUESTDATA is grateful to its authors who have been generous enough to share their knowledge with others. There is so much to be done in computer graphics that it is difficult to pick a starting point. One interesting, and probably difficult software area, is the shaping and trueing of images made with a light pen. Could a light pen image of a circle be drawn rough and trueed up in software? Could a triangle be made true and then blurred and rotated? Knowledge in these areas, gained through persistence and hard work, just might have commercial value to some graphics studio. We hope you choose to share such knowledge with fellow COSMAC hobbyists.

Interesting graphics ideas abound in the new scanachrome technique. Imagine taking a computerized airbrush with the three primary colors linked to N-line COSMAC control. By scanning the airbrush in a pattern similar to the TV electron beam sweep, you could create interesting paintings. Is there some way we might make fantastic 14 foot murals like the 3M people who use this new system commercially? Drive your van right up and let me paint a memory mapped Star Wars scene on it, the 3M people might say some day very soon.

The Center for Professional Advancement (P.O. Box H, East Brunswick, New Jersey 08816) will be giving a three day intensive course for those interested in computer aided mechanical design and drafting. The cost of the course for, "technical personnel working with, or interested in, mechanical design, drafting, and engineering graphics, including 2D and 3D graphics design, drawing and documentation," is \$450. This course is to be given in East Brunswick, New Jersey, on March 26-28. The course is primarily concerned with plotters and other equipment which is outside the budget of computer hobbyists. Is it possible to produce an X-Y type plotter, a pen mechanism that can be motorized along two coordinates, for less than \$1,500? If so, QUESTDATA would like to hear about your homebrew system.

The Institute of Electrical and Electronics Engineers will be sponsoring a tutorial on Computer Graphics along with other sessions of interest to computer personnel. Student rates for tutorials are \$25, member rates \$50, and non-member rates \$65. Quoting from the literature on the session for graphics, "This tutorial is an introductory course in computer graphics. It will address fundamental issues in hardware and software, with an emphasis on decision-making in the acquisition, implementation, and use of these systems. Topics will include hardware for plotters, line-drawing

CRTs, master video, and input devices; software for communications, display primitives, transformations, clipping, perspectives, data structures, and hidden-line/hidden-surface removal. Also discussed will be commercial and turn-key systems, and graphics standards. Examples will be given of applications in engineering, research, education, and the film making business." Comcon 79 will be held in San Francisco on February 26-March 1. If you cannot attend the sessions, digests of the material will be made available. More Comcon information is available from: IEEE Computer Society, 5855 Naples Plaza, Suite 301, Long Beach, CA 90803.

Here are some words on electronic media from Alan Watts, a man perhaps best remembered for his interpretations of Zen Buddhist philosophy. He wrote the following as the forward to a catalogue for an exhibition of electronic art organized by Oliver Andrews, Professor of Sculpture at the University of California, Los Angeles:

"By reason of electronics and automation we are moving—to the consternation of the Protestant conscience—into an age when there will hardly be any distinction between work and play. Mankind has to face the moral shock of realizing that masochistic work will be obsolete, for the slaves will no longer be people but machines, watched and tended by swinging and fascinated engineers. Art will therefore cease to be a propaganda calling attention to misery. It will use all the facilities of electronic technology to create and exuberant splendor which has not been seen since the days of Persian miniatures and arabesques, medieval stained glass, the illuminated manuscripts of the Celts, the enamels of Limoges, and the jewelry of Cellini."

"The wheel extends the foot. Brush, chisel, hammer, and saw extend the hand. But electric circuitry extends the brain itself as an externalization of the nervous system, and will therefore perform wonders of art (that is, of playful patterns of energy) which have not heretofore been seen."

BAY AREA COSMAC GROUP

A group of COSMAC users is now forming in the San Francisco Bay Area. The person to contact if you live in the Bay Area is Eugene E. Jackson, 3637 Snell Avenue, Sp. 385, San Jose, CA 95136. Eugene's phone number is (408) 224-0837. Eugene has an abiding interest in hardware, and is also the author of "Elf Talk" in QUESTDATA No. 5. He would like to get as many people together as possible to talk about hardware, software and schedule guest speakers. If you live in the Bay Area, he would very much like to have you write or call him.

1K GRAPHICS

By Alan Wallace

This program allows you to display 1K bytes of memory on an expanded memory COSMAC system for more detailed graphics. If the byte at location 001F is 01, the program itself will not appear on the screen. With the display program off the screen, you can produce and manipulate pure graphics in a 64 x 128 display using the RCA 1861 graphics chip.

The monitor in my system is located at F0 00 and the display program allows jumping to the monitor with C0 F0 00. The monitor listing given in *Popular Electronics* March 1978 put me onto the PROM monitor with tape I/O. I continued to use the transistor amplifier approach to taping until I realized how easily it spoiled tapes. When I switched to the OP AMP method, I was able to record programs more quickly and accurately. Have fun exploring the 1K display capabilities of your extended memory Elf.

LOC.	OP CODE/DATA	COMMENTS
0000	30 03 C4	Can be used to jump to monitor
03	90 B1 B2 B3 B4	Initialize registers
08	F8 26 A3	R3.0=(MAIN)
0B	F8 00 B2	R2=(STACK)
0E	F8 3F A2	
11	F8 18 A1	R1.0=Interrupt
14	D3	SEP 3
15	72 70	INTRET: Restore D, X and P
17	C4 22 78	INT: Entry Point, Stack
1A	22 52	Manipulation
1C	E2 E2	NOP for timing
1E	F8 00 B0	Can use F8 01 B0 if you
21	F8 00 A0	wish display routine off screen
24	30 15	Branch INTRET
26	E2 69	X=2; turn TV on
28	3F 28	Wait for INPUT
2A	37 2A	to be depressed
2C	6C B4	Input data to R4.1
2E	3F 2E	Wait for INPUT
30	37 30	to be depressed
32	6C A4	Input goes to R4.0
34	3F 34	Wait for INPUT
36	37 36	to be depressed
38	6C	Input data
39	54 14	and store via 4 and INC 4
3B	30 34	Repeat storage process
3D	00 00 00	Stack
40	DISPLAY AREA 1K DISPLAYED	

WHAT THE MACHINE IS THINKING

Part of the power of the 16 x 16 register matrix within the 1802 microprocessing unit (MPU) is its ability to store the addresses of all 65K of memory. In the basic Elf, the need for the upper registers is not readily apparent. In the basic 256 byte COSMAC, the upper portion of the registers make nice temporary holding places for data or timing loops. The timing loop for blinking the Q-LED at a visible rate, refer to QUESTDATA #3, depends upon decrementing all 16 bits of a register. Other methods of long delays, such as cascading registers, are also possible. But it is when you have extended the memory of your Elf past 265 bytes that you begin to see the need for having the registers 0 through F sixteen bits in length. The fact that you can store the numbers 00 00 through FF FF hex in these registers is put to real use when it comes to extended memory relocation.

The process of putting a number in the high portion of Register 1 is similar to placing data in the low part of that Register. To place the hex number 01 in the lower part of Register 1 we write the following code: F8 01 A1. That is, first we place the number in the D-Register, using the load immediate instruction, then we transfer the number to the low part of Register 1 using PLO (Put Low Register 1 - A1). A good way to keep the machine code for Putting Low and Putting High separate in your mind is to consider that A is lower down in the alphabet than B. This relationship similar to Get Low 8N and Get High 9N (where N is any of the 16 registers 0-F). When we want to put the address of something on page 01 of memory (the basic 256 bytes of the COSMAC being page 00), we write F8 01 B1. We then designate the lower portion of memory: F8 00 A1 (Thus Register 1 contains 01 00 and can be used to point to the address where we wish to start loading data).

When we want to move data or program on an extended memory COSMAC, even from one part of page 00 to another part of page 00, it is necessary to put 00 in the high part of the pointer register. There is garbage in the high part of registers, just as there is garbage (unwanted numbers) in the memory of your microcomputer when it is first turned on. There are certain registers which are automatically reset to zero upon pressing RESET and RUN. They are: I, N, X, P, Q (the one bit latch), and R(0)—this is why the program starts execution at 00 00 (upon startup the Program Counter is automatically Register 0).

If you wish to write a program on page 00 and move it to another page of memory (page 04 is used in the example) the extended memory mover below can be used. You could use this mover to change just part of a program written on a high page of memory. It is not necessary to change the locations after the 3X branch instructions when you move things to a different page

of memory. This is because the 30 instructions are Short Jumps or jumps within the same page. This page relative feature is fine, as long as you remember to change long jumps or use coding tricks for making your long jumps relative jumps. One way to accomplish complete relativity of location, hence easily relocated programs, is by using a Pseudo Program Counter—a technique described in the RCA User Manual for the CDP1802 (MPM-201). Any program you write after location 19 and through 00 FF, as in this example, will run perfectly on the page to which it is transferred. To begin a program which has been moved to a new page, you can use the long jump C0 04 1A (the example relocates to page 04 1A but can be used to relocate anywhere in 65K).

LOC.	CODE	MNEM.	COMMENTS
0000	F8	LDI	
0001	04 ←		HIGH move to start
0002	B1	PHI1	
0003	F8	LDI	
0004	1A ←		LOW move to start
0005	A1	PLO1	
0006	F8	LDI	
0007	00 ←		HIGH from address
0008	B2	PHI2	
0009	F8	LDI	
000A	1A ←		LOW from address
000B	A2	PLO2	
000C	42	LDA2	This part loads and
000D	51	STR5	deposits data in new
000E	11	INC1	loc. and inc. pointer
000F	91	GHI1	Puts R1.1 into D for
0010	FB	XRI	compare with STOP loc.
0011	04 ←		HIGH ending STOP loc.
0012	3A	BNZ	loop until 04; then cont.
0013	0C		
0014	81	GLO1	Puts R1.0 into D for
0015	FB	XRI	compare with STOP loc.
0016	FF ←		LOW ending STOP loc.
0017	3A	BNZ	loop until FF; then HLT
0018	0C		
0019	00	IDL	HALT

Next month it is back to the basic 256 byte Elf for some more experiments. The logic instructions and ways to manipulate them to make them do what you want them to do, will be our topic. Should be fun and interesting.

DIRECT DECIMAL WITH THE 1802

By F. L. Oats

There are basically two ways in which decimal numbers may be treated in a microprocessor-based system. One widely used scheme is to convert input binary coded decimal to hex, treat the hex number internally, then convert the hex back to binary coded decimal for output. This method is usually restricted to low precision numbers (eight significant decimal digits or less) and gets very awkward when dealing with high precision or variable precision applications.

The other method bypasses the code conversion to and from hex, and pretty well keeps the numbers in decimal all the way through the calculation. This generally requires less command code, and puts high precision within the reach of everybody. In spite of the fact that the 1802 has no special flags or commands for handling decimal numbers, it is still possible and practical to work direct decimal with the 1802. We will now see how.

In order to work this scheme, decimal numbers must be represented in memory as shown in Figure 1. Each digit will occupy an entire byte of memory with the digit itself contained in the lower four bits (least significant), and a zero in the upper four bits (most significant). This representation has a fringe benefit when using ASCII code for input/output—it eliminates the digit pack and unpack operations required for two-digits-per byte environments.

The program presented here works with fifteen significant decimal digits, and has an extra byte reserved for sign, overflow and decomp purposes. Figure 2 shows an entire decimal quantity, or operand, as it would appear in memory. The addresses, hex 10 thru hex 1F, correspond to operand A on line 1 of Figure 3. The byte labelled 03 on line 1 of Figure 3, for example, would reside in memory address 001D. Note that the least significant digit resides in the highest memory address of the quantity. Later, we will see how to increase or decrease the number of significant digits (it's easy).

To work the direct decimal add and multiply operations, we will employ a two phase correction technique as shown in Figure 3. Phase I will be applied to one of the operands, and consists of adding hex F6 to each digit of the affected operand. In figure 3, line 1 shows the decimal number 1369 as it would appear in memory, line 2 shows phase I being applied to the number, and line 3 shows the result of phase I. Line 4 shows the number 2349 in memory as the second operand, and line 5 shows the result of adding lines 3 and 4 together in internal (hex) code.

Line 5 is obviously not ready to be output at this point, but we can salvage this number by applying phase II. Phase II looks for a carry out of the digit being treated into the next significant digit. If the

carry takes place, it says that the digit does not need further correction and stores the digit, unaltered, into the answer. The carry is then propagated to the next column. If the carry does not take place, it says that the number is bad and needs phase II correction—hex F6 is subtracted from the digit and the result is placed in the answer. Carry to the next significant digit is inhibited.

The machine language code for the add algorithm is presented in Table I. Operand A is assigned memory locations 0010 thru 001F, operand B is assigned 0020 thru 002F, and the result goes into 0030 thru 003F. They are placed near the beginning of memory for the sake of those who do not have monitors or operating systems. The program will first perform phase I correction to the entire operand A—all fifteen—before operand B is ever examined. This finishes up lines 1 and 2, figure 3, and puts us on line 3 with all fifteen digits as shown in the figure. Now we work with LSD first, taking the FF on line 3, adding it to the 09 on line 4, and getting the 08 with a carry out on line 5. We now examine the carry out, find that there is in fact a carry out, and put the 08 in our answer without phase II correction. We now take this carry and add it to the next digit on line 3 (the hex FC) plus the corresponding digit on line 4 (the 04) and we get an 01 with a carry out. Line 3 thru 7 are performed on each digit before proceeding to the next digit.

**Each digit will occupy an entire byte
of memory with the digit itself contained in
the lower four bits (least significant),
and a zero in the upper
four bits (most significant).**

The subtract algorithm, believe it or not, is actually simpler than the add because phase I is not required. Figure 4 shows the general flow of a subtract operation. Here again, the absence of a carry out invokes phase II correction and causes a borrow from the next position by inhibiting carry out.

When we reach the F9 on line 3, we add that to the 03 on line 4 (along with a carry from the previous position) and get an FD with no carry out. The absence of a carry out invokes phase II correction and hex F6 is subtracted from the FD, and we store the result, 07, in the answer. We also suppress carry into the next position.

After processing an entire decimal quantity, we need to look at DF one last time. If it is set after an add operation it indicates that an overflow has occurred and appropriate action should be taken. If it is RESET after a subtract, it indicates that the answer is in complimentary form (the answer is negative) and must be decomplimented. One way to decompliment or decomp, as it is often called, is to force a 01 into that extra byte on the most significant end, clear out (00) the other 15 bytes, and subtract the bad answer from this quantity. In other words, place 01 in memory address 0010 and 00 in address 0011 thru 001F. Move the bad answer to operand B memory area and perform another subtract. There is an easier way which I will leave to the reader to discover.

Looking at the program listing in Table 1, addresses 0000 thru 000C are used to point R8 to operand A, R9 to operand B and RA to the answer. Address 0040 and address 004F place a hex 0F in RB, used to define the number of significant decimal digits we are working with. By allocating more memory per operand, and changing the number loaded into RB (and re-locating some code), we can work with practically any number of decimal digits. (Didn't I say it was easy?)

In Table II, the subtract program is shown starting at memory location 0044. The addresses 0000 thru 0043 are exactly the same as the add program (and MUST be loaded). Notice that in the subtract program, RB is loaded only once—at address 0040—so to change the number of digits per operand you need only change addresses 0041 from hex 0F to hex ever-how-many-places-you-want.

Each program will turn on the Q-LED when finished. This should tip you off to any catastrophic load errors you might encounter. The Q-LED should come on instantly with these programs and if it doesn't, check your program code.

It is beyond the scope of this presentation to go into multiplication and division of decimal numbers. They are, of course, performed by repeated addition and subtraction as presented here. If you are still with us, you can no doubt add your own racing stripes to the programs. However, before you start figuring up the time required for that multiply operation, let me point out that phase I need be applied only once for the entire multiply. Once the operand is pre-corrected, it should remain that way throughout the calculation. (Don't let your stack pointer get ahead of it).

Hopefully, you now have at least some vague idea of how to work direct decimal with the 1802. If you are into high precision stuff, maybe you can utilize some of these concepts in your application. Have fun with it.

CAUTION: The Musical Keyboard program on this page involves a hardware mod suggested by the author Kirk Bailey. Any modifications of Quest products published in Questdata or otherwise will void the warranty for the product. Quest also will not be responsible for any damage caused by modification to any other hardware as a result of Questdata articles.

MUSICAL KEYBOARD

Kirk D. Bailey

This musical keyboard program demonstrates the use of masks, logic operation, and data tables. It turns an Elf into an organ, and with a minor modification, gives true organ function. The data lookup table contains the data necessary for the cycle duration loop, and simplifies the use of the machine. The Q-line is toggled by a four-byte sequence which, since it does not use addressed branches, can be used in other locations as it is. The logic sequence is:

STEP 1. LOGICAL AND to clear to 0 the .1 nibble.

STEP 2. ARITHMETIC ADD with 50 to form composite data byte.

The resulting data byte is loaded into the data table pointer (RD) which, as a result of this logic sequence, is pointing at the byte in memory containing the necessary data to generate the tone selected by the keyboard.

The program will wait for FLAG 4 to go to 0V, then generates tones as long as EF4 is at 0V. This can be replaced by C4's at locations 04 and 05 to get constant tones. On the Super Elf, the keyboard decoder chip (74C923) has a data available chip strobe pin, DA, which is pin number 13. If you run a jumper to P3 on the 44 pin or 50 pin busses on the Super Elf, you will have taken care of the tone makers hardware requirements. Changing addresses 04 from 3F to 36, in the program, you will get tones when the keyboard is depressed, and only then. You may want to change the size of R13 (47K in the Super Elf) to a larger size, if necessary. Good luck and 73's.

LOC.	CODE	MNEM.	COMMENTS
0000	E2	SEX	Set X=2
01	F8 FF	LDI	Pointer to loc. FF
03	A2	PLO2	Pointer in R2.0
04	3F (36)		See text use 36 or Br. EF3 with
05	04		hardware modification only
06	6C	IMP 4	Input from Bus
07	FA 0F	ANI	Logical And to clear
09	FC 50	ADI	ADD to form composite
0B	AD	PLOD	Resulting byte to RD.0
0C	0D	LDND	Get data from table
0D	FF 01	SMI	Tone delay
0F	3A 0D	BNZ	Br. back if delay not up
11	CD	LSQ	Long skip if Q=1
12	7B	SEQ	Set Q=1
13	38	SKP	Short skip
14	7A	REQ	Turn off Q
15	30 04	BR	GOTO 04 (get new tone)

TABLE OF TONE DATA

0050	84 77 6A 64 59 4E 45 40 39
0059	33 2F 29 25 20 1E 1A (last data in loc. 5F)

(Please turn to page 11 for a birds eye view of the pouts of the MM74C922 IC chip.)

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

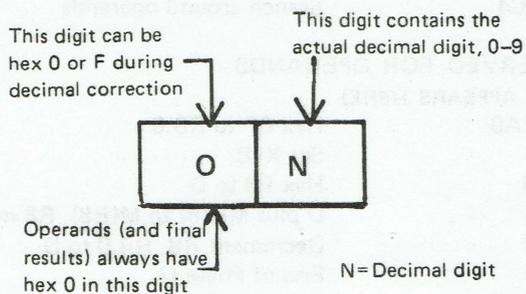


FIGURE 1. Representation of decimal numbers in memory. A single digit occupies an entire byte of memory.

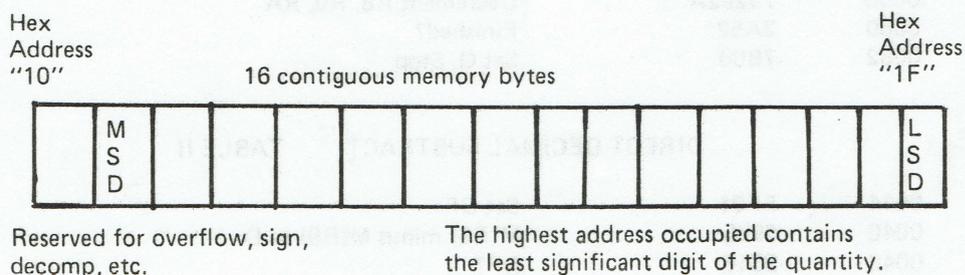


FIGURE 2. A single operand occupies 16 bytes of contiguous memory space, and can have 15 significant digits. This size is easily changed (see text).

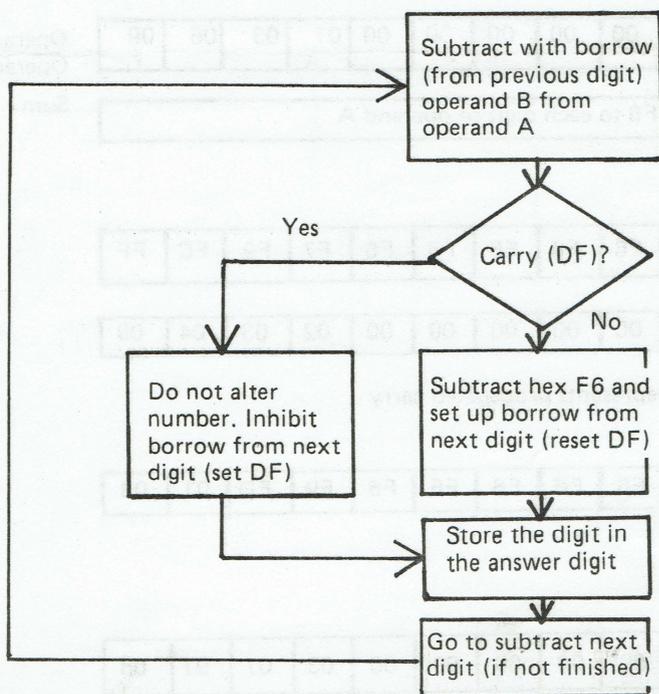


FIGURE 4. General subtract flow.

DIRECT DECIMAL ADD TABLE I

Address	Code	Action
0000	90B8B9BA	Clear R8.1, R9.1, RA.1
0004	F81FA8	Hex 1F to R8.0
0007	F82FA9	Hex 2F to R9.0
000A	F83FAA	Hex 3F to RA.0
000D	3040C4	Branch around operands
0010		
thru	RESERVED FOR OPERANDS	
003F	(ANSWER APPEARS HERE)	
0040	F80FAB	Hex 0F to RB.0
0043	E8	Set X=8
0044	F8F6	Hex F6 to D
0046	F473	D plus M(R8) to M(R8), R8 minus 1
0048	2B8B	Decrement RB, RB.0 to D
004A	3A44	End of Phase I?
004C	F81FA8	Hex 1F to R8.0
004F	F80FAB	Hex 0F to RB.0
0052	0974	M(R9) plus M(R8) to D
0054	335A	Carry (DF)?
0056	FFF6	D minus hex F6
0058	FC00	Reset DF
005A	5A	Store result in M(RA)
005B	2B8B	Decrement RB, RB.0 to D
005D	28292A	Decrement R8, R9, RA
0060	3A52	Finished?
0062	7B00	Set Q, Stop

DIRECT DECIMAL SUBTRACT TABLE II

0044	FF01	Set DF
0046	0975	M(R8) minus M(R9) to D
0048	334E	DF?
004A	FFF6	D minus hex F6
004C	FC00	Reset DF
004E	5A	Result to M(RA)
004F	28292A	Decrement R8, R9, RA
0052	2B8B	Decrement RB, RB.0 to D
0054	3A46	Finished?
MSD 0056	7B00	Set Q, Stop

LSD

OPERAND A

00	00	00	00	00	00	00	00	00	00	00	00	01	03	06	09
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Operand A = 1369
Operand B = 2349

PHASE I

Add hex F6 to each digit of operand A															
---------------------------------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Sum = 3718

MODIFIED OPERAND A

00	F6	F7	F9	FC	FF										
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

OPERAND B

00	00	00	00	00	00	00	00	00	00	00	00	02	03	04	09
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Represents propagated carry

INTERMEDIATE RESULT
MODIFIED OPERAND A + B

00	F6	F9	FD	01	08										
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

PHASE II

DECIMAL ANSWER

00	00	00	00	00	00	00	00	00	00	00	00	03	07	01	08
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

FIGURE 3. Direct decimal add.

Copyright Electronics Documentation and Software by Request. All Rights Reserved under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.

PATTERNS

Michael Tyborski

Here is a graphics program that I wrote for the COSMAC Elf. I hope that the readers of QUESTDATA find it as intriguing as I have.

This program uses the CDP 1861 video IC to plot various semi-animated patterns on a video monitor or TV with RF modulator. Interestingly, these images can become quite complex, depending upon the special mathematical equation used to define new points. As written, they will be circular in nature.

PATTERNS requires a minimum of 1¼K RAM for successful execution. This program should be partitioned with the program in page 00 and the display refresh buffer on a page boundary. These requirements are a result of various steps taken to minimize the programs length.

PATTERNS is extremely easy to use. Simply load and run the program, after connecting the computer to a video monitor. The direct video method is preferable as a better display will be obtained. The results will amaze almost anyone.

Many more patterns may be created by writing different point subroutines based upon new equations. As an incentive, ten additional equations are included. If you choose to use them, please note that upon entry, the data pointer (R6) is pointing to 'X' and that the stack or R2 should be used for storing intermediate results.

In addition, interested users may also desire to experiment with the display update speed or density. Simply change the value of the delay count at M(0086) to whatever is desired. This also holds true for the point counter at M(0045).

BIBLIOGRAPHY

Anderson, D. John, *Serendipitous Circles*, **Byte**, II (August 1977)

Kellerman, Eduardo, *Serendipitous Circles Explored*, **Byte**, III (April 1978)

PLOTTING EQUATION USED IN PATTERNS

$$\text{ADDRESS} = A0 + X/8 + Y \cdot 8$$

$$\text{BIT POSITION} = X \cdot 7$$

WHERE A0 IS THE START ADDRESS

ADDITIONAL PATTERN EQUATIONS

- | | |
|---------------------------|-----------------------------------|
| 1. $X := X + 1$ | 6. $X := X - (2 \cdot X \cdot Y)$ |
| $Y := Y + X$ | $Y := Y + (X/3)$ |
| 2. $X := X - (Y/3)$ | 7. $X := X - (2 \cdot Y)$ |
| $Y := Y + (X/1.5)$ | $Y := Y + (2 \cdot X)$ |
| 3. $X := X - (2 \cdot Y)$ | 8. $X := X - (Y/8)$ |
| $Y := Y + (X/1.1)$ | $Y := Y - (X/8)$ |
| 4. $X := X - (Y/2)$ | 9. $X := X + (Y/2)$ |
| $Y := Y + (X/4)$ | $Y := Y - (X/2)$ |
| 5. $X := X - Y$ | 10. $X_{\text{OLD}} = X$ |
| $Y := Y + (X/2)$ | $X := X - (Y/2)$ |
| | $Y := Y + (X_{\text{OLD}}/2)$ |

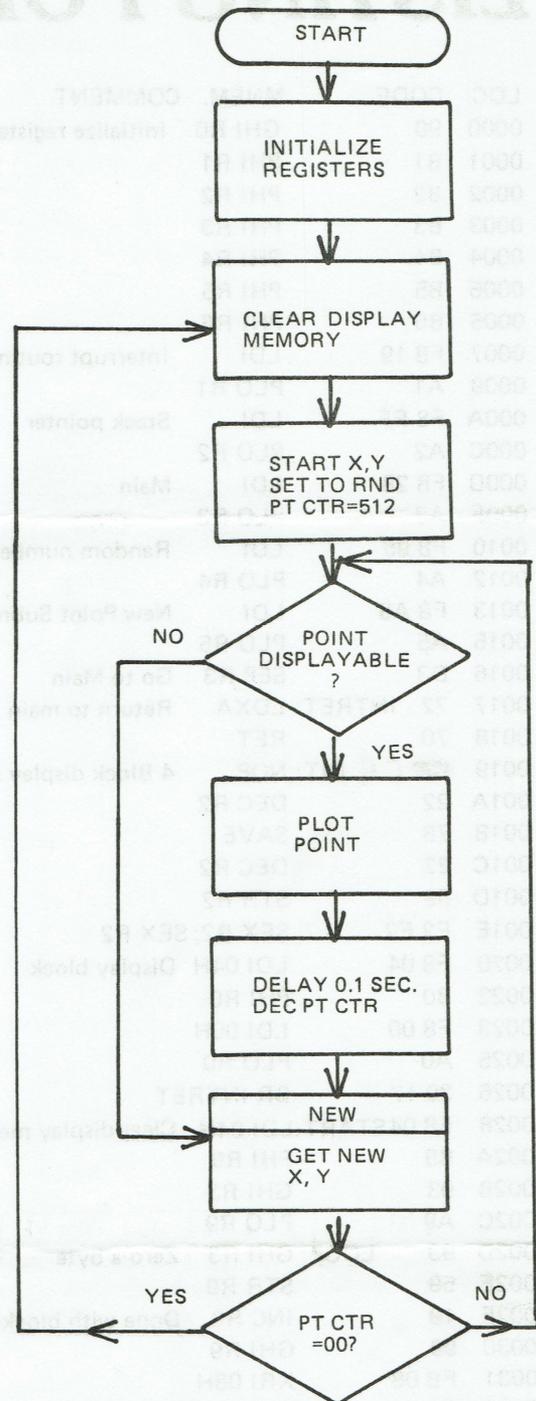


Figure 1. MAIN PROGRAM ALGORITHM

REGISTER	APPLICATION
0	Refresh Pointer
1	Interrupt Routine
2	Stack Pointer
3	Main Program Counter
4	Random Number Subroutine
5	New Point Calculation Routine
6	Data Pointer
7	Point Counter
8	Point Address
9	Utility

TABLE 1: REGISTER ALLOCATION DATA

LISTING FOR PATTERNS

LOC.	CODE	MNEM.	COMMENT
0000	90	GHI R0	Initialize registers
0001	B1	PHI R1	
0002	B2	PHI R2	
0003	B3	PHI R3	
0004	B4	PHI R4	
0005	B5	PHI R5	
0006	B6	PHI R6	
0007	F8 19	LDI	Interrupt routine
0009	A1	PLO R1	
000A	F8 FF	LDI	Stack pointer
000C	A2	PLO R2	
000D	F8 28	LDI	Main
000F	A3	PLO R3	
0010	F8 95	LDI	Random number subroutine
0012	A4	PLO R4	
0013	F8 A8	LDI	New Point Subroutine
0015	A5	PLO R5	
0016	D3	SEP R3	Go to Main
0017	72	INTRET: LDXA	Return to main
0018	70	RET	
0019	CA C4	INT: NOP	4 Block display format
001A	22	DEC R2	
001B	78	SAVE	
001C	22	DEC R2	
001D	52	STR R2	
001E	E2 E2	SEX R2: SEX R2	
0020	F8 04	LDI 04H	Display block
0022	B0	PHI R0	
0023	F8 00	LDI 00H	
0025	A0	PLO R0	
0026	30 17	BR INTRET	
0028	F8 04	START: LDI 04H	Clear display memory
002A	B9	PHI R9	
002B	93	GHI R3	
002C	A9	PLO R9	
002D	93	LOOP: GHI R3	Zero a byte
002E	59	STR R9	
002F	19	INC R9	Done with blocks?
0030	99	GHI R9	
0031	FB 08	XRI 08H	
0033	3A 2D	BNZ LOOP; no	
0035	E6	SEX R6	Activate video IC
0036	99 F8 EE	INP 9	
0037	F8 EE A6	LDI Y	Set X, Y to random
0039	A6 69	PLO R6	Initial values
003A	D4	SEP R4	Call RND
003B	FC 0B	ADI 0BH	
003D	FA 7F	ANI 7FH	Get Y in display range
003F	73	STXD	
0040	D4	SEP R4	Get random X
0041	FA 3F	ANI 3FH	
0043	56	STR R6	

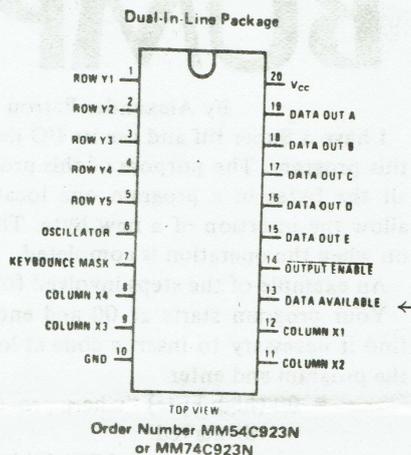
LOC.	CODE	MNEM.	COMMENT
0044	F8 02	LDI 02H	Set point counter 0200H
0046	B7	PHI R7	
0047	F8 04	PLOT: LDI 04H	Set R9 to starting
0049	A9	PLO R9	Block of display
004A	F8 ED	LDI X	Is X out of range?
004C	A6	PLO R6	
004D	F0	LDX	
004E	FF 41	SDI 41H	
0050	33 8D	BPZ NPOINT; yes	
0052	72	LDXA	Compute X/8
0053	F6 F6 F6	SHR; SHR; SHR	
0056	52	STR R2; Save in stack	
0057	F0	LDX	Is Y out of range?
0058	FF 81	SDI 81H	
005A	33 8D	BPZ NPOINT; yes	
005C	F0	LDX	Compute Y * X
005D	FE	SHL	Times 2
005C	3B 61	BNF TIME4; Overflow?	
0060	19	INC R9	Yes
0061	FE	TIME4: SHL	Times 4
0062	3B 65	BNF TIME 8; Overflow?	
0064	19	INC R9	Yes
0065	FE	TIME8: SHL	Times 8
0066	3B 69	BNF ADD; Overflow now?	
0068	19	INC R9	Yes
0069	E2	ADD: SEX R2	Add X/8+Y * 8
006A	F4	ADD	
006B	3B 6E	BNF NC	Any carry
006D	19	INC R9	Yes
006E	A8	NC: PLO R8	Set R8 to point addr.
006F	89	GLO R9	
0070	B8	PHI R8	
0071	F8 80	LDI 80H	Set bit position
0073	52	STR R2	
0074	26	DEC R6	Compute X * 7 and
0075	06	LDN R6	put in counter R9
0076	FA 07	ANI 07H	
0078	A9	PLO R9	
0079	32 82	BZ DISP	
007B	F0	BIT: LDX	Shift 80H required
007C	F6	SHR	Number of times
007D	52	STR R2	
007E	29	DEC R9	Done?
007F	89	GLO R9	
0080	3A 7B	BNZ BIT; no	
0082	08	DISP LDN R8	Activate desired bit
0083	F1 - F3	OR	
0084	58	STR R8	
0085	F8 02	LDI 02H	Delay
0087	B9	PHI R9	
0088	29	DLY: DEC R9	
0089	99	GHI R9	Time up?
008A	3A 88	BNZ DLY; no	

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

```

008C 27      DEC R7  Decrement point counter
008D E6  NPOINT: SEX R6  Restore X to R6
008E D5      SEP R5  Calculate new X, Y
008F 97      GHI R7  Required points plotted
0090 3A 47   BNZ PLOT; no
0092 30 28-35 BR START; Begin again
* * * * *
RANDOM NUMBER SUBROUTINE * * * * *
0094 D3  EXITRSEP R3  Return
0095 86      RND: GLO R6  Save data pointer
0096 52      STR R2
0097 F8 EF   LDI rndnum; compute 5 times
0099 A6      PLO R6  Old random number
009A F0      LDX
009B FE FE   SHL; SHL
009D F4      ADD
009E FC 02   ADI 02H
00A0 56      STR R6  Set as new rndnum
00A1 A9      PLO R9
00A2 02      LDN R2  Restore data pointer
00A3 A6      PLO R6
00A4 89      Glo R9  Return to Main
00A5 30 94   BR EXITR; with rndnum in D
* * * * *
NEW POINT CALCULATION SUBROUTINE * * * * *
00A7 D3  EXITN: SEP R3  Return
00A8 16      NEW: INC R6  Compute Y/2
00A9 06      LDN R6
00AA F6      SHR
00AB FB FF   XRI FFH Negate result
00AD 26      DEC R6
00AE F4      ADD  XNew=X+(-Y/2)
00AF 56      STR R6
00B0 16      INC R6  YNew=Y+XNew/2
00B1 F6      SHR
00B2 F4      ADD
00B3 56      STR R6
00B4 30 A7   BR EXITN
    
```

PIN OUTS FOR HARDWARE MODIFICATION (see page 6)



NOTE: A typographical error on page 6 has been found. PIN 13 IS THE DATA AVAILABLE.
 WARNING: The Super Elf Warranty is voided in the event that the product has been misused, damaged, or modified.

Back issues of QUESTDATA are available for \$1.50 each, beginning with issue No. 1. Programming knowledge is cumulative.

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

Publisher Quest Electronics
Editor Bill Haslacher
Technical Coordinator Bill Thompson
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.

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: _____

NAME _____

ADDRESS _____

Signature _____

CITY _____ STATE _____ ZIP _____

Please start my subscription with Issue No. _____

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

BYTE BUMPER

By Alexander Petrou

I have a Super Elf and use its I/O device numbers in this program. The purpose of this program is to move all the bytes in a program one location forward to allow the insertion of a new byte. The Q light comes on when the operation is completed.

An example of the steps involved follows:

Your program starts at 00 and ends with 60. You find it necessary to insert a code at location 40. Start the program and enter:

- 00 (high byte) "where process STOPS" is displayed.
- 60 (low byte) "of STOP" is displayed.
- 00 (high byte) "of START of bumping process" is displayed.
- 40 (low byte) "of START" is displayed and Q light comes on.

If you examine memory, bytes 41 thru 61, you will see the contents of what was previously in locations 40 thru 60.

The program takes care of incrementing jump or branch instructions. It does this by detecting 3X. Care must be taken, therefore, to make sure that it does not increment a code following a data code, eg.: If you look at the instructions F8 30 A2 you will find A2 incremented to A3.

The program also looks to see whether the jump or branch location needs to be incremented. For example, if you had an instruction branch to start (30 00), the program will not increment the location code.

I have this program on ROM (1702A) and do a long Branch to the starting address of the program - C0 80 00.

The operation is a handy one to have in your programmer's toolbox.

LOC.	CODE	COMMENTS
00	F8 00 BC	Load high byte R.C
03	BD	and high byte R.D
04	F8 FE AC	Load Low byte R.C
07	F8 FC AD	Load low byte R.D
0A	3F 0A	Wait for INPUT
0C	ED 6C BF	X=D, load R1.F via keyboard
0F	64	Display keyed numbers
10	37 10	Wait for INPUT release
12	3F 12	Wait for INPUT
14	6C AF 64	Load R0.F via keyboard & display
17	37 17	Wait for INPUT release
19	3F 19	Wait for input
1B	EC	Set X to C
1C	6C BE 64	Load R1.E via keyboard & display
1F	37 1F	Wait for INPUT release
21	3F 21	Wait for INPUT
23	6C AE 64	Load R0.E via keyboard & display
26	2C 2C	Bring R.C to first loc.
28	30 40	Br. to subroutine at loc. 40
2A	2F EC	Decrement R.F and set X to C
2C	9F F7	Subtract high byte to and from addr.
2E	32 33	If result=0 branch to loc. 33
30	2F 30 28	If result≠0 branch to 28
33	1C EC	Inc. R.C and set X to C
35	8F F7	Subtract low byte to and from address
37	32 3D	If result=0 branch to end loc. 3D
39	2F 2C	Dec. R.F and R.C
3B	30 28	and branch back to 28
3D	7B 30 3E	Turn Q on and halt
40	31 4D	Branch if Q is on to loc. 4D
42	4F 5F	Get from mem. pointed by R.F and increment and store in new address
44	FA F0	and mask high byte bits
46	FF 30	Subtract 30 from D
48	3A 2A	If D≠0 branch back to 2A
4A	7B	Turn Q on before
4B	30 2A	Branch back to main program
4D	7A	Q off
4E	1F 1F 1F	Decrement back to br. loc.
51	1C EC	Increment R.C & set X=C
53	0F F7	Subtract br. loc. from low byte 'to' addr.
55	3B 5E	If br. loc. is smaller than 'to' addr. go 5E
57	0F FC 01	If not add 1 to br. loc.
5A	5F 2C 2F	and store back in br. loc.- dec. R.C & R.F
5D	C8	Skip next two instructions
5E	2F 2C	Decrement R.C & R.F
60	30 2A	Br. back to main program

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

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

ADDRESS CORRECTION REQUESTED

44807
Steve Brune
408 E. Wadsworth Hill MTU
Houghton, MI. 49931

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

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