Questdata ©
Volume 1 Issue # 5

WHAT—THE MACHINE IS THINKING?

SOMETIMES mankind makes a tool that reshapes society. The industrial revolution can be seen as the substitution of mechanical muscle for human toil. There was a certain fear of the new machines which were taking over in the industrial revolution. The folk tale of Paul Bunyan vs. the railroad spike-driving machine is a revealing myth. Paul Bunyan beats the spike-driving machine, but dies of exhaustion at the end of the race.

There are two fears apparent in the Paul Bunyan myth. First, a lot of people were afraid of losing their jobs to the new machines. This, of course, was not an unjustified feeling since a lot of people were displaced by the new machines. Second, mankind likes to feel superior to its machines.

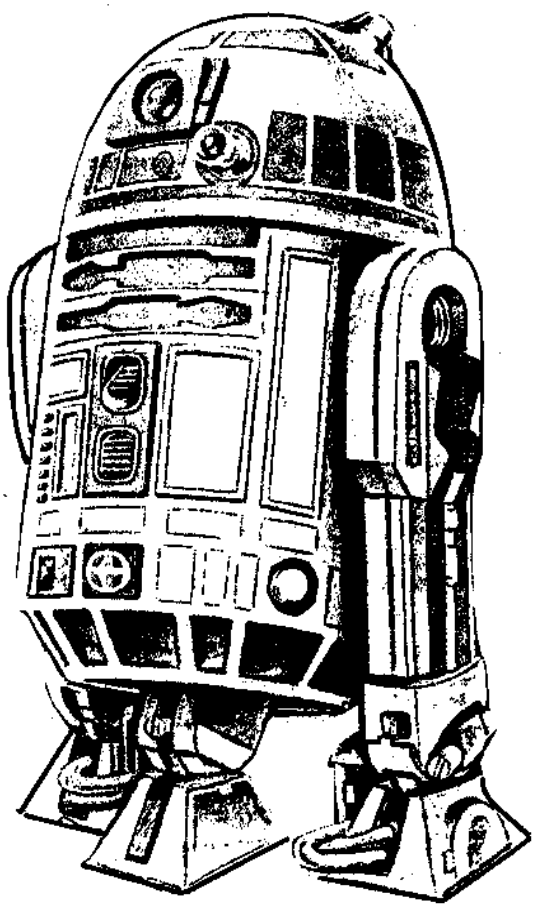
We are on the verge of a machine takeover which combines the muscle of industrial revolution machines with the automation of mental tasks. Myths of mad robots running amok abound in our culture. There are also myths of computers run by government officials for the express purpose of prying into the individual's freedom and rights of privacy. The real prospect for computers, however, appears to be a non-threatening one for mankind, at least for the present and forseeable future.

The movie Star Wars is an exception to the robot as nightmare creation treatment with its friendly R2D2 and C3PO. We are pretty far away from the creation of robots with the capabilities of this dynamic pair, but indications are that the creation of robots has not fallen into the hands of evil scientists bent on the destruction of earth.

Would you believe that at least some of the research work with real robots is being done with the COSMAC 1802? Let's take a brief look at some work being done at Yale with STOVER (Steerable Observing Vehicular Robot). QUESTDATA would like to thank John-Francis Mergen for information on this interesting exclusive report on this project.

STOVER THE YALE ROBOT

The STOVER Robot was developed by Stan Honey, Richard Kozlak, J-F Mergen, Frank Schely, and Professor J.G. Zornig as part of a project in the 1977–1978 academic year. The entire document of STOVER capabilities and electronics can be ordered through the Yale University Department of Engineering and Applied Sciences, Attn: Professor Zornig. A price for the 70 page document on STOVER has not been given to QUESTDATA. If you wish to inquire about



STOVER, it is wise to include a self- addressed, stamped envelope with a couple dollars to cover copying and handling of the documentation.

STOVER is a three-wheeled robot which is controlled by a radio link with a PDP-10 (now upgraded to PDP-20/50). The 1802 uses a special set of interface circuitry to read data from the robot, read data from a sonar, send data to the robot, and control a full duplex radio link. The schematic for these devices are in the STOVER report.

Funding for the project was by NBS, and the project will continue next year in order to improve and work all bugs out of the STOVER system.

A cross-assembler was written by John-Francis Mergen to facilitate the writing of COSMAC programs assembled by the PDP-20/50. Since the 1802 oriented cross-assembler was developed using government funds, it cannot be sold. However, as part of the documentation of STOVER it is available for the price of postage and handling.

STOVER has for its heart a COSMAC Super Elf which has been modified to suit the research project. The STOVER project is an interesting one since it is yet another step toward bringing the robot into the big leagues. Where do we want to go with robotics and just how far can we go with robotics?

VENUS PROBES

Christmas 1978 will bring you news of how the COSMAC space travelers are doing. These probes contain not one, but many 1802's. The idea is that if one system in a probe should fail there will be another to back it up—redundancy.

You can bet that these probes are performing some pretty sophisticated and complex functions out there in space—a lot of miles away from mother earth. It is probable that they are performing some functions as well as humans might. The 1802 even has some advantages over us. You don't need elaborate systems to take along and monitor oxygen when you have only a microcomputer to worry about.

At last there is an answer to those who ask, "What can you do with a microcomputer?" "Well Charley," you reply sagely, "you can land them on Venus and use them to explore the terrain . . . now can we play computer blackjack?" It is interesting to note, but only if you are not a diehard STAR TREK fan, that space exploration (going where no man has gone) will probably be done entirely with computer robots of fantastic and fascinating complexity. These microcomputers functioning as brains will have to be programmed to adapt. That is, they will have to be able to survive the unforseeable. What happens if they encounter a black hole? What happens, and one current physics theory has it this way, if the known universe is actually a black hole; and this robot escapes out of it for a look around? Or a Klingon vessel attempts to destroy it? Anyhow, such considerations make fascinating programming. Such a program might even incorporate that neat little anti-gravity routine you wrote for STAR TREK—for real.

Let's face it—the future is interesting. This is part of the reason why we are in the microcomputing hobby in the first place. The microcomputer is a kind of crystal ball or window to the future. Just because we enjoy our machines does not diminish their role of significance in the future.

FURTHER FUTURE POSSIBILITIES

Arthur C. Clarke in ***Profiles of the Future*** starts his fine work of non-fiction speculation with the statement that it "is impossible to predict the future." Clarke, however, has a good track record in future prediction. In 1945 he wrote up an article on how a synchronous satallite radio and television relay system might work. Today, the communication satellite is a reality. In 1947 he correctly predicted that a rocket would land on the moon in 1959.

In the back of ***Profiles of the Future*** is a time table for future events. He gives 1980 for planetary landings, for example. Some of the predictions about computers are interesting: 1970 translating machines; 1990 artificial intelligence; 2000 global library; 2020 robots; 2050 memory playback; 2080 machine intelligence exceeds man's. His chapter on the obsolescence of man concludes with humans being the link to intelligent robots. He does not find this to be a bleak prospect for mankind. "No individual exists forever; why should we expect our species to be immortal?" Man, said Nietzche, is a rope stretched between the animal and the superhuman--a rope across the abyss. That will be a noble purpose to have served."

Man, said Nietzche, is a rope stretched between the animal and the superhuman-a rope across the abyss.

About robots Clarke says, "It will be much smaller and neater than the walking jukeboxes or mechanized suits of armor which Hollywood presents . . ."

A number of scientists and laymen, according to Clarke, suffer from "a failure of nerve." He gives as examples numerous authorities who give scientific explanations of why airplanes can't fly and rockets will never reach the moon. These negative predictors did not reach far enough with their imaginations. Clarke says, "To predict the future we need logic; but we also need faith and imagination which can sometimes defy logic itself."

COSMAC USER's CLUB -- FINLAND

Of all microcomputer hobbyists in Finland some 60% use the RCA 1802 processor chip! Our hardware is rather similar in operation to the VIP, we use Chip-8 graphics together with machine language and Tiny Basic. A lot of hardware has also been developed, such as TV typewriter, EPROM programmer, etc. We would like to exchange both software and hardware ideas with other user's groups around the world. Please contact:

J-E Nystrom
Ulvilantie 2 C 27
FIN-00350 Helsinki 35
FINLAND

Back issues of QUESTDATA are available (beginning with issue number 1) at $1.50. Back issues ar mailed first class postage. Customers in foreign countries, other than Canada and Mexico, please send 50 cents additional for airmail with each issue ordered.

WHAT THE MACHINE IS THINKING

PROGRAM RELOCATION

Just for the fun of it let's take a look at programming from the viewpoint of the programmer. How does he write a program? What steps are involved?

Programmer: Ummm, let's see now, I wrote this whole thing in a lower part of memory (Loc. 0C-18) and now they want it to be moved up higher in memory. Logically speaking that leaves me with two alternatives... find other employment or move it. OK, so first thing is to look at the program itself a Q line tone which goes: 7B, C4, C4, C4, C4, C4, 7A, C4, C4, C4, C4, 30, C0. The first thing to look for is the 3X numbers because I know that those are conditional and unconditional branches and the jump to locations must be changed. If there were long branches then there would be CX type instructions to find. So in this tone generator we have a 7B which turns the Q LED on, and C4's which cause a time delay until the flow comes to 7A which turns Q off, and then some more C4 type delays follow until the 30 instruction is encountered. The 30 causes a jump back to location 0C. So if I change 0C to the first location of the place where we are going to move it then we have got the problem partially licked.

Now instead of using the Super Elf monitor or incrementing up to that new starting location using the DMA increment bootstrap, I am going to write a short routine to move it. This is the type of task that the register indirect mode of addressing is well suited to handle. The 1802's addressing scheme is register indirect, so this program demonstrates its effectiveness as well as solving the task.

The first thing we have to do is initialize the MOVE TO location. That is, we want to select a starting value and set it aside in one of the lower parts of the "pointer registers." Since the solution to this problem of moving a program up in memory requires fewer bytes to solve it on a basic 256 byte Elf, we will solve it first for the basic Elf and then show how it would work on an expanded memory COSMAC. The bytes F8, 20, A1 establish the starting location 20 in the lower half of register 1. What happens is that the 20 is put first into the D-register and then passed or handed off to the low byte of register 1. It is a kind of programming ritual to establish values that you know you will need in a program the very first thing. So after loading in the MOVE TO value we realize that it is equally important to establish a FROM value. F8, 0C, A2 will do this nicely and it gives the starting value (or where to find) of the routine we wish to move.

Now that we have taken care of the necessary "housekeeping"(we programmers have a word for everything). we can move on to the interesting part. The machine language instruction 42 is LDA 2. This is the instruction we want next. We really get two commands in one with LDA. With two you get egg roll. (No, that isn't a programmer term its restaurant talk.) See, with LDA you get both an LDA and an INC. The LDN part takes the location designated by register 2, namely 20 and puts its contents (in this case 7B) in the D-register. Then the INC part increments register 2, and thus leaves register 2 pointing at location 21.

We now have 7B in the D-register. What next? Well, we want to transport it to the MOVE TO location. This we must do with two instructions. There is not a combined instruction to do this simply because it is not something that is encountered very regularly in programming the 1802. There is something of this type in STXD which both stores and decrements and is extremely useful in stack handling. So what we do is first STR (51) and then INC (11). The STORE VIA N instruction takes the contents of the D-register and puts them in the location pointed to by register 1 which is location 0C. The next time we perform the loop we do not want register 1 to still be pointing to 0C or we will be continually writing over this location with the new bytes. Hence, we increment location 0C with INC (11). Now we are pointing at 0D and are all set for another round of moving.

Think of it this way. A moving truck must move all the houses on a certain block and move them to an identical culdesac in another neighborhood. This is a strange situation, but a good example, of the process involved. The truck packs up house number one and moves the stuff into house number one in the new area. The driver increments his map (crosses off this house), and goes after the next house. The process continues until... The trucker keeps going until he reaches the blue house with the green fence. Or the house at 760 Elm, or the fifth house if he has been told to stop moving at five houses.

Right, there are three possible ways to stop the moving process. The first way is to look for the blue house with the green fence. This is the easiest method for the programmer but it has some problems which we will cover a little later. In programming terms we "test the data." We use 3A (BNZ) for this test. The byte we are testing for is, of course, 00. If we find 00 then we stop the moving process; and if we don't then we want the moving to continue.

Page 4 shows the way this program looks.

LOC.	CODE	MNEM.	COMMENTS
00	F8	LDI	MOVE-TO starting
01	20		location
02	A1	PLO1	Put loc. in Reg. 1.0
03	F8	LDI	Program you are
04	0C		moving starts here
05	A2	PLO2	Put loc. in Reg. 2.0
06	42	LDA2	Load by R2 and INC
07	51	STR	Put D in appointed loc.
08	11	INC	Increment Reg. 1
09	3A	BNZ	Test for flag 00
0A	06		Branch if not flag
0B	00	HLT	Halt if 00 flag

SAMPLE PROGRAM TO BE MOVED TO LOC. 20

LOC.	CODE	MNEM.	COMMENTS
0C	7B	SEQ	Turn on Q-LED
0D	C4	NOP	Computer thumb
0E	C4	NOP	twiddling
0F	C4	NOP	. . .
10	C4	NOP	
11	C4	NOP	
12	C4	NOP	
13	C4	NOP	
14	C4	NOP	
15	7A	REQ	Turn Q-OFF
16	C4	NOP	
17	C4	NOP	
18	C4	NOP	
19	C4	NOP	
1A	C4	NOP	
1B	C4	NOP	
1C	C4	NOP	
1D	30	BR	Branch to Loc.
1E	20		changed by hand
1F	00		Stop code FLAG

This program will be moved to location 20 when run with the above program. To execute the program after the move has been made, load locations 00 and 01 with 30 20 and press run. You can check to see that the program is moved by the MEMORY PROTECT and INPUT increment method.

The problem with the blue house with the green fence approach to data moving is that there may be more than one blue house with a green fence, and the mover may stop moving before he reaches the correct house. Translating this physical analogy to the programming situation, you see that it is quite likely that any hex number that you choose will be used as part of the program or its incorporated data. We have picked an unusual program to load and have checked it to see if there are any 00's beforehand. If you make sure to do this, your program will be moved without any problems. There are ways around this problem, so that a flag type approach (which is what programmers would call it) could be used with a better degree of certainty. If you were to search for two 00's in a row, it is more unlikely that these characters would appear by chance. IBM uses a similar approach in its Job Control Language (JCL) in which the program searches for the control characters /* and /&. In some programming situations, the search for a control flag character is a good choice and a method to keep in mind.

ADDRESS METHOD

In our particular programming application—moving a program—it is better to choose the address method. This method is a sure thing and it doesn't take that many more bytes to perform. Each time the programming loop is performed the program checks for hex address 30. When the chosen address is reached, the program terminates.

LOC.	CODE	MNEM.	COMMENTS
00	F8	LDI	MOVE-TO starting location
01	20		is address 20
02	A1	PL01	Put loc. in Reg. 1.0
03	F8	LDI	Program you are
04	0F		moving starts here
05	A2	PLO2	Put loc. in Reg. 2.0
06	42	LDA2	Load by R2 and INC
07	51	STR	Put D in appointed loc.
08	11	INC	Increment Reg. 1
09	81	GLO1	R1.0 goes into D for
0A	FB	XRI	Address XRI testing
0B	30		is LOCATION=30?
0C	3A	BNZ	If it is not location 30
0D	06		then jump to 06
0E	00	HLT	Stop if it is location 30
0F	7B	SEQ	Turn on Q-load a simple
10	C4	NOP	program here
11	C4	NOP	to make sure things
12	C4	NOP	really are moved to
13	C4	NOP	location 30 —you can
14	7A	REQ	Turn off Q-never be sure
15	C4	NOP	of anything
16	C4	NOP	until you try it and
17	30	BR	discover for yourself
18	20		Branch to loc. 20

To run the moved program change locations 00 and 01 to 30 20 and run. By changing address locations and modifying the program you can relocate programs of any length and content.

COUNTING METHOD

In this method you set up a register to count the number of times the moving loop is made. When this counter reaches a preset number the moving process halts. The initialization part can be placed anywhere up towards the front of the program. Register 3 is used as the counter in this example.

LOC.	CODE	MNEM.	COMMENTS
00	F8	LDI	MOVE-TO starting location
01	20		is address 20
02	A1	PL01	Put loc. in Reg. 1.0
03	F8	LDI	Program you are
04	15		moving starts here
05	A2	PLO2	Put loc. in Reg. 2.0
06	F8	LDI	Initialize Reg. 3.0 to 00
07	00		
08	A3	PLO3	
09	42	LDA2	Load by R2 and INC
0A	51	STR	Put D in appointed loc.
0B	11	INC	Increment Reg. 1
0C	83	GLO3	Register 3.0 into D
0D	FC	ADI	ADD one to count
0E	01		
0F	A3	PLO3	Put count in R3.0
10	FB	XRI	Test the count
11	0B		Is count=12 hex?
12	3A	BNZ	If count is not up then
13	09		goto 09
14	00	HLT	Halt if count is up
15	AA		load data or program
16	BB		here — it can be AA
17	CC		or 01, 02 — just to
18	DD		get the idea of program
19	EE		relocation
1A	ETC. TO LOC. 1F (thus 12 dec. locations moved)		

Load 00 and 01 with 30 0F respectively. Push run and check the locations (starting with 20 hex) to see if program or test data has been moved using this method.

INDIGENOUS MUSIC—A DO-IT-YOURSELF APPROACH

By Bob Richie

SOUNDER delves, once again, into the largely unexplored frontier of "indigenous music." In the program VORTEX II (QUESTDATA no. 4) we generated a relatively simple sound or function which we found to be "indigenous to the 1802 microcomputer," hence the term—indigenous music. This program explores the concept of a more complex and variable format for indigenous music programs.

This program is designed to make it possible for the computer hobbyist to easily explore a small segment of the new musical frontier.

SOUNDER contains eight addresses which are to be filled in by the programmer. These addresses should conform to certain ranges or modes indicated in the program. Changing one or more of these "variables" will produce modifications in the musical sound. In some cases the changes will produce dramatic alterations in the musical content.

If you keep the tempo and cycle near 0A, it is recommended that the program be played in its entirety. Many times there are surprises hidden in toward the end of the score that are worth noting.

As you explore and experiment with the program you may want to keep track of the combinations of modifiers which rendered interesting results. A good way to organize your experimentation is by drawing a chart similar to CHART 1. By giving each set of modifiers an apt and colorful name, you will be able to easily recall what that particular combination sounds like. You are encouraged to send in any interesting combinations you feel valuable to QUESTDATA for publication.

Listening to the combination of modifiers labeled SONGBIRD, makes me wonder if there is a direct relationship between the songs of birds and arithmetics. What do you think? Have fun exploring new combinations—the fun part is discovering how each new modification changes the tune.

SOUNDER

LOC.	CODE	ACTION
00	90 A1	R(1).0=00
02	21	Dec. R1
03	81	D=R(1).0
04	A4	Put value in R(4).0
05	F8	Load Cycle
06	CYCLE 0A-30	
07	A2	Put value in R(2).0
08	22	Dec. R2
09	82	D=R(2).0
0A	32 02	Branch if D=00
0C	F8	Load Tempo
0D	TEMPO 01-10	
0E	83	Put value in R(3).1
0F		D=R(4).0
10	DIRECTION ↑76, 7E↓	
11	PERSPECTIVE ←33, 3B→	
12	17	Go to loc. 17
13	ALU F9, FA, FB, BC, FF	
14	OP CODE 00-FF	
15	ALU as defined at 13	
16	OP CODE	
17	A4	Put value in R(4).0
18	7A	Q=OFF
19	84	D=R(4).0
1A	A5	Put value in R(5).0
1B	25	Dec. R5
1C	23	Dec. R3
1D	85	D=R(5).0
1E	3A 1B	If ≠ 00 then goto 1B
20	93	D=R(3).1
21	32 08	Branch to 08 if D=00
23	31 18	If Q=ON goto 18
25	7B	Turn on Q
26	30 19	Branch to loc. 19

CHART 1. SOUNDER

FUNCTION	CYCLE	TEMPO	↑ ↓	← →	ALU	OP CODE CHOICE		
RANGE/MODE	0A-30	03-10	76 7E	33 3B	F9, FA, FB FC, FF	00-0E		
ADDRESS	06	0D	10	11	13	14	15	16
TITLE								
SONGBIRD	30	03	76	3B	FF	B7	FB	BF
WITCH								
DOCTOR	11	09	76	3B	F9	C9	FA	BF
ELF DUET	06	09	7E	33	FB	09	FA	6D
NOTE: INTRODUCTION TO ELF DUET LASTS 59 SECONDS.								
ELF DUET								
OVERTURE	06	09	7E	3B	FB	09	FA	6D

MULTIPLE PRECISION SUBTRACTION

By Ivan Dzombak

This program subtracts two four-byte numbers and puts the answer in memory locations 78-7B. To run it, enter the lowest order byte of the minuend, hit INPUT, and repeat up to the highest order byte (fourth one). After the first four have been entered, the Q-LED will come on. This indicates that the minuend bytes have been entered. Now, enter the subtrahend bytes, from the lowest to highest order. The Q-LED will extinguish after these four bytes have been entered. If the answer is negative, it will stay on. Keep pressing INPUT and the answer will be displayed (lowest to highest order). The negation routine (36-4A) takes the twos complement of the accumulator and sets Q to 1 if the answer is negative. Note the use of the SUBTRACT WITH BORROW instruction. Have fun.

LOC.	CODE	MNEM.	ACTION
00	E1	SEX R1	X=1
01	7A	REQ	RESET Q
02	F8 70	LDI	R1.0=70 POINTS TO
04	A1	PLO R1	MINUEND
05	F8 74	LDI	R2.0=74 POINTS TO
07	A2	PLO R2	SUBTRAHEND
08	F8 78	LDI	R3.0 POINTS TO
0A	A3	PLO R2	ACCUMULATOR
0B	38	SKP	SHORT SKIP
0C	7B	SEQ	Q=ON
0D	F8 04	LDI	R4.0=04 (LOOP 4
0F	A4	PLO R4	TIMES)
10	3F 10	BN4	WAIT FOR INPUT
12	37 12	B4	CLOSURE
14	6C	IMP 4	INPUT
15	64	OUT 4	DISPLAY; R(X)+1
16	24	DEC R4	R4-1
17	84	GLO R4	R4.0 INTO D
18	3A 10	BNZ	D≠00 THEN 10
1A	39 0C	BNQ	Q=OFF THEN 0C
1C	7A	REQ	Q=OFF
1D	F8 70	LDI	RESET POINTER
1F	A1	PLO R1	R1.0=70
20	02	LDN R2	M(R2) INTO D
21	F5	SD	M(R1-D) TO DF, D
22	53	STR R3	D TO M(R(3)) ACCUM
23	11	INC R1	INCREMENT ALL
24	12	INC R2	POINTERS
25	13	INC R3	. . .
26	F8 03	LDI	R4.0=03 (LOOP 3
28	A4	PLO R4	TIMES)
29	02	LDN R2	M(R2) INTO D
2A	75	SDB	M(R1)-D NOT DF
2B	53	STR R3	D TO M(R3)
2C	11	INC R1	INCREMENT ALL
2D	12	INC R2	POINTERS
2E	13	INC R3	. . .
2F	24	DEC R4	R(4)-1
30	84	GLO R4	R(4.0) INTO D
31	3A 29	BNZ	D≠00 THEN 10
33	E3	SEX R3	X=3
34	33 4B	BDF	DF=ON THEN 4B
36	F8 78	LDI	RESET POINTER
38	A3		(2's COMP IF NEG)
39	F8 00	LDI	D=00
3B	F7	SM	D-M(R3) TO D
3C	53	STR R3	D TO M(R3)
3D	13	INC R3	R3+1
3E	F8 03	LDI	R(4.0)=03
40	A4	PLO R4	. . .
41	F8 00	LDI	D=00
43	77	SMB	D-M(R3) NOT DF
44	53	STR R3	D TO M(R3)
45	13	INC R3	R3+1
46	24	DEC R4	R4-1
47	84	GLO R4	R(4) INTO D
48	3A 41	BNZ	D≠00 THEN 41
4A	7B	SEQ	Q ON (NEG. ANS.)
4B	F8 78	LDI	RESET ACCUM.
4D	A3	PLO R3	POINTER
4E	3F 4E	BN4	WAIT FOR
50	37 50	B4	INPUT
52	64	OUT 4	SHOW ACCUM; R3+1
53	30 4E	BR	GOTO LOC. 4E

BUG SQUASHER

[NOTE: Vortex II will run OK without the changes noted by reader Larry R. Baysinger—although this bug should be squashed for the sake of programming form. The bug is a typographical omission on the part of QUESTDATA. Bob Richie, the author of the Vortex II program had the 00 stop. Our apology to Bob Richie for the omission. What is really interesting about the letter quoted below are the ideas on coding format for publication.]

I very much enjoy QUESTDATA and look forward to each new issue. Although my 1802 microcomputer is a "homebrew" bus based machine, I configured the I/O, display, etc. so that most software written for the Elf/Elf-II will run without modification. Therein lies my one suggestion—how about establishing a convention for the format of software listings published in QUESTDATA?

To date there has been a mix of (1), what I assume is RCA's convention—grouping two or three related instructions relative to one address—as in the Morse code program on page 10 of the No. 4 QUESTDATA, and (2), what I feel is the more easily read listing of the 1802 phototimer program on page 6 of the same issue.

I have inclosed a copy of a programming form which I had printed locally at a "Jiffy-Print" house which explains my preference for the "single-instruction-per-line" format. This form is usable for either assembly or machine language programming—and provides for register mapping and comments. Incidentally, the "Vortex II" program has a bug—in that 32 (BZ)—18 (ADDR 18) instruction at address 05, 06 take you to address 18, for which there was no instruction listed in the original program. I assume an IDL 00 was to have been there—in which case the program will stop after one pass, or addresses 18 and 19 could have been a BR (30) 00 (ADDR 00) to endlessly loop the program.

One other comment. I very much like the inclusion of flowcharts with program listings.

Keep QUESTDATA going and growing.

Larry R. Baysinger

COMBINATION LOCK SHOWS USE OF MEMORY STACK

By Brandon Mathew

LOC.	CODE	MNEM.	ACTION
00	F8	LDI	Initialize "code" stack pointer R(1)
01	42		
02	A1	PLO	
03	F8	LDI	Initialize "input" data pointer R(3)
04	43		
05	A3	PLO	
06	E1	SEX	X = 1
07	F8	LDI	Get first code byte
08	01		
09	73	STXD	Put on stack R(1).0 = 42
0A	F8	LDI	Get second code byte
0B	02		
0C	73	STXD	Put on stack R(1).0 = 41
0D	F8	LDI	Get third code byte
0E	03		
0F	73	STXD	Put on stack R(1).0 = 40
10	F8	LDI	Get fourth code byte
11	04		
12	73	STXD	Put on stack R(1).0 = 3F
13	F8	LDI	Initialize loop counter
14	04		
15	A4	PLO	
16	E3	SEX	X = 3
17	3F	BN4	Wait for INPUT depressed
18	17		
19	6C	INP 4	INPUT data
1A	64	OUT 4	Show data
1B	37	B4	Wait for INPUT released
1C	1B		
1D	24	DEC	Decrement loop counter
1E	84	GLO	Get loop count R(4).0
1F	3A	BNZ	Check to see if loop is done
20	16		
21	F8	LDI	Set loop counter R(4).0 = 04
22	04		
23	A4	PLO	
24	11	INC	Increment stack pointer to point to "code"
25	E3	SEX	X = 3
26	23	DEC	Decrement data pointer to point to "input"
27	41	LDA	Load code from stack into D register
28	F3	XOR	Test to see if equal
29	3A	BNZ	If not zero goto error routine
2A	3A		
2B	24	DEC	Decrement loop counter
2C	84	GLO	Get loop count R(4).0
2D	3A	BNZ	Check to see if loop is done
2E	25		
2F	7B	SEQ	Set Q (INPUT was right)
30	E0	SEX	X = 0
31	64	OUT 4	Display "00"
32	00		
33	3F	BN4	Wait for INPUT depressed
34	33		
35	37	B4	Wait for INPUT released
36	35		

Here is a hex listing for a 4 bit combination lock. The object of this program is for the user to light the Q light by putting in the correct 4 byte combination. The Q output could be used to drive a solenoid driver which would open a door etc.... Now you might say that that seems impractical, but you know, you start writing software and.... Well anyway, here is a little information on how the program works. It all started as an exercise in using loops and stacks.

R(0) is used to set up the location of the various pointers. R(1) is used as a stack pointer. R(3) is used as a data pointer for input bytes. R(4) is used as a loop counter; to count the input loop and the compare loop. Locations 3A–3E are used to display "EE" for error and return to the top of the program. Once the user inputs the correct input a "00" is displayed and Q is set. When the input is pressed again Q is reset and the program loops to the begining. In case the code isn't obvious it is 01 02 03 04. If the programmer wants to change the combination, he can do so by changing locations 08, 0B, 0E, and 11.

Well, that's the program. The program is pretty straightforward and COSMACians being the brilliant people they are, ought to easily grasp it.

LOC.	CODE	MNEM.	ACTION
37	7A	REQ	Reset Q
38	30	BR	Start over
39	00		
3A	E0	SEX	X = 0
3B	64	OUT 4	Display "EE"
3C	EE		
3D	30	BR	Try again
3E	00		
3F	stack data code 4		
40	... code 3		
41	... code 2		
42	... code 1		
43	INPUT 1 stack		INPUT bytes stored
44	INPUT 2		in these 4 bytes
45	INPUT 3		
46	INPUT 4		

TV TARGET GAME USES VIDEO GRAPHICS

By Richard Moffie

Fire a bullet across the screen at a moving target. Only 256 bytes of memory are needed. A target moves vertically on the right side of the screen. To fire a shot, press and release the INPUT switch. The score is shown on the data display: first digit shows the number of shots and second digit shows the number of hits. Target speed can be changed by altering byte 45. Bytes 90–FF must be loaded with 00 since the program does not clear the game area.

The game uses the bottom part of the TV screen as a playing field, with the upper part being used as program storage. A target moves vertically downward at the right of the screen, returning to the top when it reaches the bottom. At the far left is a shot which can be fired by pressing the INPUT button briefly. There is a delay loop to slow the speed of the shot and target so that they can be seen. In addition, there is a counter (set to 0A hex in byte 45) so that the target moves slower than the shot - labelled delay counter 2 on the flowchart.

The target is always moving in the program (except between the time INPUT button is pressed and when it is released). When the INPUT is pressed, the Q line is turned on. This results in a click when the shot is fired, and another when the target is hit, but more important, it lets the computer know that the shot has already been fired so that each time through the loop, it will move the shot rather than wait for INPUT. R8 stores the score, with 10 being added each time a shot is fired, and 01 when a hit is made. In this way, the data display shows the number of shots (left digit) and the number of hits (right digit).

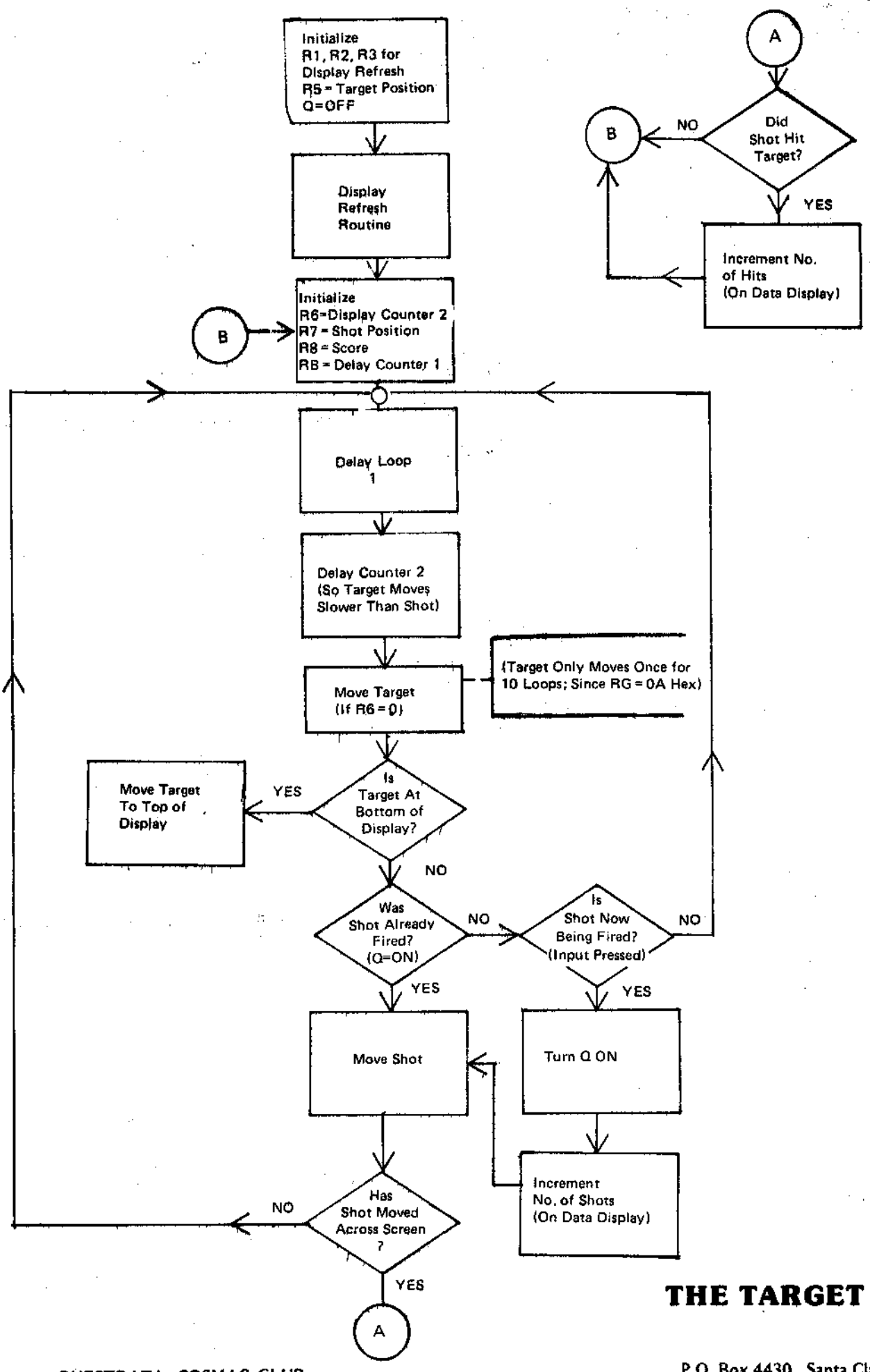
When the shot reaches the right side of the screen, its position is compared with that of the target, and if they are the same, a hit is recorded. If not, it is a miss; and the program resets for the next attempt (except for the score which records the attempt).

LOC.#	OP CODE/DATA	COMMENTS
00	F8 97 A5 7A C4	Initial shot position
05	F8 2D A3	Display refresh
08	F8 8E A2	
0B	F8 11 A1	
0E	D3	
0F	72 70	
11	22 78 22 52	
15	C4 C4 C4	
18	F8 7F AA	
1B	F8 00 A0	
1E	80 E2	
20	E2 20 A0	
23	E2 20 A0	
26	E2 20 A0	
29	3C 1E 30 0F	
2D	E2 E2 69	
30	F8 00 A6 A8	Initialize registers
34	F8 C0 A7	Initial shot position
37	F8 80 57 7A	
3B	F8 02 BB	Delay
3E	2B 9B 3A 3E	
42	16 86	
44	FD 0A 3A 5A	Change byte 45 to adjust target speed
48	F8 00 55 A6	Move target
4C	85 FC 08 A5	
50	FD FF 3A 57	If target at bottom of screen,
54	F8 97 A5	return to top position
57	F8 80 55	
5A	31 68	If shot fired, move shot
5C	3F 34 37 5E 7B	Check for shot fired (Q on=fired)
61	88 FC 10 A8 52	Increment shot counter
66	64 22	
68	07 F6 57 3B 74	Move shot
6D	F8 00 57 17	
71	F8 80 57	
74	87 FA 07	Is shot across screen?
77	FF 07 3A 3B	
7B	87 5A 85	Did shot hit target? (5A stores
7E	FF 00 3A 34	target position in byte 7F)
82	88 FC 01 A8 52	If so, increment hit counter
87	64 22 30 34	Display score, set up for next shot
8B	00 00 00 00 00	Stack
90-FF	00	Display Area (Set to 0—blank screen)

CONSTRUCTION HINTS FOR WORKING WITH TRANSISTORS

James C. Nicholson

Inasmuch as excessive heat is damaging to transistors, one can prevent damage during soldering by using small HEAT SINK clips. They may be purchased at your own local electronic supply or from NATIONAL CAMERA SUPPLY, 2000 W. Union Avenue, Englewood, Colorado 80110. The cost is nominal, under $1.00.



THE TARGET GAME

HOLIDAY FAVORITES

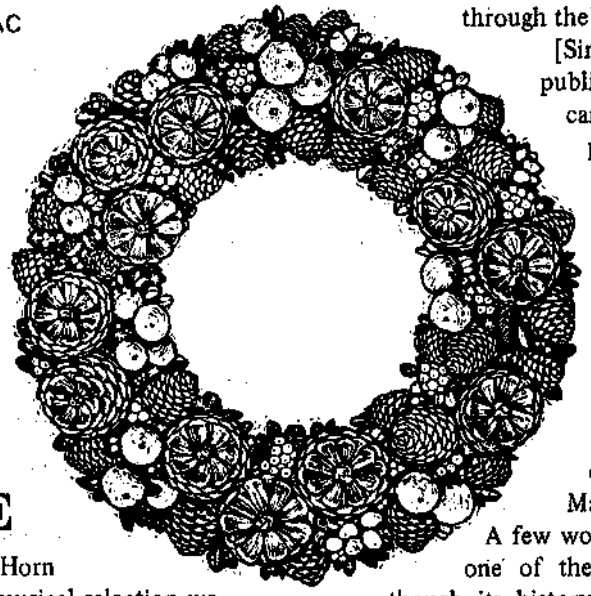
MYSTERY PROGRAM

By D.J. Lindberg

Here is a holiday program which uses the McCormick music program given in the February 1978 issue of *Popular Electronics*. You can run this program using the jump 30 68 in locations 00 and 01 or you can make the suggested alterations in the program given on page 10 of QUESTDATA No. 3.

We are not going to tell you the song but rather will let you put the program in and be surprised . . . unless you can figure the song just by reading the code.

CODE	← Begins at LOCATION AC
36 1B	
36 1B	
6E 1B	28 19
36 1B	38 19
36 1B	36 1B
6E 1B	36 1B
36 1B	18 1B
42 15	18 1B
28 24	42 15
31 1F	42 15
95 1B	19 19
38 19	31 1F
38 19	57 24
56 19	00 End



ENCORE

By Gerald M. Van Horn

For an encore to our holiday musical selection we have that old favorite—"Silent Night." This musical score was written for a one MHz XTAL clock. Modifications can be made to the tempo as noted in the February 1978 *Popular Electronics* music program by E.M. McCormick. The program uses 06 in location 77 for tempo. It is fun sometimes to adjust the tempo just to experiment with the music. You can run this program by using the jump 30 68 in locations 00 and 01. Extended memory is required to play the music in its entirety. Merry Christmas.*

MUSIC CODE

21 24	31 1F	18 1F	29 19
0C 1F	18 1F	21 24	0B 24
16 24	29 19	0C 1F	12 2D
36 2D	0F 1B	16 24	27 24
21 24	18 1F	24 2D	0A 2A
0C 1F	21 24	0B 4C	10 33
16 24	0C 1F	41 15	3A 3B
36 2D	16 24	21 15	00 End
41 15	24 2D	39 11	
21 15	0B 4C	10 15	
52 1B	31 1F	1B 1B	
3A 19	18 1F	4E 19	
1D 19	29 19	4A 12	
41 24	0E 1B	0B 4C	

* Note: 25 in location 77 is a good tempo for a super rendition of "Silent Night."

DECK THE HALLS

By Mark Wendell

When the editor of QUESTDATA subtly hinted that I write a music program for the Christmas issue, my mind was set in motion. The problem was to find sheet music to a popular Christmas carol. Once I had this taken care of (thanks to a local library), I had to decide which music program to use. This was no problem though, for the music program by Ed McCormick in the February 1978 issue of *Popular Electronics* had the widest note range of all the programs I have. I had to make a few modifications to the program for my own purposes, for example, in order to get plenty of room for the notes I relocated it from byte 68 to the beginning of the page (byte 00). In addition, I modified it to repeat once it has gone through the musical selection.

[Since *Popular Electronics* originally published the algorithm, QUESTDATA cannot publish the program without permission from *Popular Electronics*. We have contacted *Popular Electronics*, but have not received word from them yet. Ed McCormick has given his OK, but the copyright belongs to *Popular Electornics*. We will publish the entire data list. Those of you with 1K and 4K extended memory boards will, of course, have no problems with Mark's musical selection.]

A few words about "DECK THE HALLS," one of the most familiar Welsh carols. Although its history is obscure, it was picked up by Mozart in the 18th century for use in a duet for violin and piano.

Merry Christmas and a happy new COSMAC.

CODE			
23 15	0B 4C	0B 4C	25 15
0F 19	22 15	1F 1F	0F 19
1B 1B	0F 19	0E 1B	1B 1B
18 1F	1B 1B	1D 19	18 1F
16 24	18 1F	18 1F	16 24
18 1F	16 24	22 1B	18 1F
1B 1B	18 1F	0F 19	1B 1B
16 24	1B 1B	20 15	18 24
0C 1F	16 24	1B 1B	12 12
0E 1B	0C 1F	0E 1B	12 12
0F 19	0E 1B	0F 19	12 12
0C 1F	0F 19	20 15	12 12
25 1B	0C 1F	12 12	27 15
0C 1F	24 1B	14 0F	0F 19
16 24	0C 1F	2C 0D	1D 1B
14 27	16 24	28 0F	1F 1F
16 24	14 27	AB 12	2F 24
	16 24	20 15	FF 4C
		0B 4C	00 End

SYNTHESIZED MUSIC EFFECTS

Eugene E. Jackson

This music program sequentially displays data from memory, and at the same time converts the displayed data into tones with duty cycles and duration determined by the current byte.

Like the "Spirit of 76" (see QUESTDATA no. 3) program it can be modified to give a wide variety of effects that should be of special interest to synthesized music fans.

This program was originally designed to make use of the displayed data to control two 4/16 decoders which in turn control a 50240 top octave generator chip. The varied Q output is then used as a programmable envelope generator which varies audio output to an amplifier.

If the most significant digit of data sets the octave and effects; and the LSD sets the note and duration; the end result is a programmable organ.

LOC.	CODE	COMMENTS
00	F8 00 AA	RA=Start of display
03	F8 FF AB	RB=Number of displays
06	EA	Set X to RA
07	F0 A1 BF	R1=Duration RF=Pause
0A	64	Display current data
0B	81 A2 A3	R2=Off duration R3=On
0E	82 A4	duration; R4=Off Freq.
10	24 84	Q off loop
12	3A 10	
14	7B 12	Q on / increment off
16	83 A4	duration
18	24 84	Q on loop
1A	3A 18	
1C	7A 23	Q off / decrement on
1E	83	duration
1F	3A 0E	Tone finished?
21	2F 9F	Pause loop
23	3A 21	
25	2B 8B	Data count complete?
27	CE 30 07	
2A	00	
28	Display Pattern . . . to LOC. FF	
FF		

This program starts the display at location 00, proceeds to FF and stops. To repeat, change bytes 2A and 2B to 30/00. To display pattern only change byte at location 01 to 2B and byte at location 04 to D4. The pause cycle may be eliminated by replacing bytes at location 21 to 24 with C4's. Bytes at location 15 and loc. 1D may be changed to 22 and 13 or 22 and 23 for different effects on increment and decrement.

You can try arithmetic operations after the D=0 decisions are made to further change the output effects.

COMING ATTRACTIONS

- **QUESTDATA Delves into Hardware**
- **Interesting Arithmetic Routines Revealed**
- **Graphics Doodle Program for 256 Byte Elf**
- **Blackjack Program**
- **And Much, Much, More . . .**

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

Publisher . . . . . **Quest Electronics**
Editor . . . . . **Bill Haslacher**
Technical Coordinator . . . . . **Bill Thompson**
Programming Assistance . . . . . **Pam Gazlay**
Proofreading . . . . . **Ken Brown**

REGISTER ZAPPER

Here is a quick way to make sure that all the high bytes of your registers are initialized for use with extended memory COSMAC systems. It is also easy to remember:

LOC.	CODE	MNEM.	COMMENTS
0000	C0	LBR	Long Branch to Zero
0001	0F		Routine
0002	00		Can be anywhere
0F00	XX		First three bytes of
0F01	XX		program which the
0F02	XX		long branch took out
0F03	F8	LDI	Load 00 into the high
0F04	00		portion of all registers
0F05	B1	PHI1	except PC Reg.
0F06	B2	PHI2	00 into Reg. 2 high
0F07	B3	PHI3	00 into Reg. 3 high
0F08	B4	PHI4	00 into Reg. 4 high

Etc. – all the way to BF. . .
THEN BRANCH BACK TO PROGRAM
USING – C0 00 03.

THE GREAT SOFTWARE DEBATE GOES ON

The QUESTDATA article on software media generated quite a bit of reader interest. Reader Eugene E. Jackson writes, "I think floppy ROM is first choice with magnetic tape second. Everybody owns a record player." Floyd L. Oats says, "I would like to cast my vote in favor of publication in object code as the best media for transferring software. Even disk oriented systems are largely incompatible anymore, various manufacturers have their own 'secret' format for sectoring in order to discourage software theft. I don't think anybody knows how many different cassette tape standards are in existence today. The universal media is definitely the printed page."

Thanks for the input QUESTDATA people.

It is the policy of QUESTDATA not to change the programs received in any way. Programs are picked for inclusion in QUESTDATA because they show an original or creative approach to programming. It is true that there are sometimes shortcuts which the author of the program missed in his zeal to get the program completed. The QUESTDATA staff, even though aware of other ways of accomplishing an objective, will still choose to let the author's version of the program stand. QUESTDATA feels that it is the creativity and imagination of the program which count.

If a program does not run at all, then the author is contacted. QUESTDATA uses programs which are felt to be interesting, educational and inspirational. Programs which are good, but lacking in complete documentation, are put on a shelf and will be published when the QUESTDATA staff finds time to explore and document them. In conclusion, the more complete, legible, creative and talented programs find their way into print first.

Please send your programs and ideas to QUESTDATA but be aware that the better the story, ideas and legibility – the sooner it will see print.

GRAPHICS BOOKLET BY PAUL C. MOEWS

There are a lot of interesting things you can do with graphics with your basic 256 byte Super Elf or Elf II. This booklet shows you how to make your Elf into a kaleidoscope pattern generator and tickertape-like display system. You can also have a horse race on your video screen. The final two programs are a TV stop watch and a TV clock; the clock shows hours, minutes, and seconds and is completely setable. The booklet contains a discussion of graphic interrupt routines along with the eight programs.

This exciting new booklet can be yours for only $3.00 and 50 cents extra for postage and handling. Please make checks payable to Quest Electornics.

5

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

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

ADDRESS CORRECTION REQUESTED

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