TOSIE Toronto Ohio Scientific Idea Exchange

June 26, 1983
Editors:
John Horemans
Paul Vail
Ed Maste

T.O.S.I.E.
P.O. Box 29
Streetsville,Ont.
Canada L5M 2B7

THE TOSIE PRINTOUT

Club News by Paul C.,SM (the SM stands for Still Moderator)
(or slightly mad?)

- Due to holidays,summer,etc the July meeting will probably be lightly attended but it WILL be held as usual on the last Sunday at noon at the same place. There will NOT be an August meeting.

- Well we have our new executive and surprise surprise, it hasn't changed all that much. I would like to thank Brian Scully for his term as librarian and welcome Paul Vail and Ed Maste as new newsletter editors. Please note the word 'editor', this is not the same as the word 'writer' or 'author' but guess what? All the articles this month are from John H., lets hope that Paul and Ed aren't stuck with the same situation when it is their turn to 'edit' the newsletter. And lets not forget to thank Bruce Fleming for his past editing efforts.

- Last month I asked to hear from any of our out-of-town members. To this point I don't believe we've heard from anyone. If you can't be bothered writing us why should we write this newsletter for you. If the active part of our club consists of the regulars at the meetings why do we even need a news letter, if people don't start sending us articles,questions,gossip or whatever then maybe we should just quit?

- We are still working on an OSI list of hardware and software sources. If you have had dealings with anyone regarding OSI related products please let us know.

- There was more stuff to go here this month but its after one in the morning and I can't remember so look for what ever it was next month.

Submitted Articles Shown below are the articles submitted this month by our members.

(its an old joke but this space is STILL empty!)

BASIC SPEED AND LINE NUMBERS

Key in the two little programs that follow. Note that they are identical except for one digit, yet their running speeds are vastly different.

A note in the May issue of Compute! helped explain the difference, and once again stresses the close relationship of PET BASIC and the OSI ROM version.

When BASIC encounters a statement that changes control to another line, e.g. GOSUBXXXX or GOTOXXXX, then it must somehow find that line of code. I have had the impression that BASIC began the search at the lowest line number, continuing line by line until the proper line was found. Thus authors have expounded the virtues of putting subroutines, especially the critical ones, at the top of the program. They were to be executed faster because of the lower line numbers.

This proves to be not always true. In the demonstration programs line 256 is found faster 255, even though they are the same number of lines from the beginning of the program.

It turns out that the BASIC routines do search from the beginning if the high byte of the line number is the same or lower than the high byte of the line to be found.

However, if the high byte is greater, then the search continues from the present line, without backtracking to the beginning.

Calculating the numbers to be used to take advantage is simple. Divide the present line number by 256, then add one to the integer, and multiply by 256. e.g. from line 40, type ?40/256. The computer responds with 0.15625 . The integer part is 0. Add one equals 1. 1*256=256.

Once again:
Your current line number is 1250. Key in ?1250/256. The computer responds with 4.8828, thus the high byte is 4. We use at least 5 as the high byte, or a line number 5*256 i.e. 1280 or greater.

Thus an opportunity presents itself to speed up BASIC just by a proper choice of line numbers. Hopefully this will add a little to our understanding of BASIC, and maybe help speed up a critical section of your program.

```
10 REM EXECUTES IN 19.8 SECONDS
15 REM AT 1 MHZ CLOCK
20 REM SIMULATE PROGRAM LINES
21 REM
22 REM
23 REM
24 REM
25 REM
26 REM
27 REM
28 REM
29 REM
30 REM
40 FORI=1TO6000:GOSUB255:NEXT
50 PRINT"DONE":STOP
255 RETURN
```

```
10 REM EXECUTES IN 14.2 SECONDS
15 REM AT 1 MHZ CLOCK
20 REM SIMULATE PROGRAM LINES
21 REM
22 REM
23 REM
24 REM
25 REM
26 REM
27 REM
28 REM
29 REM
30 REM
40 FORI=1TO6000:GOSUB256:NEXT
50 PRINT"DONE":STOP
256 RETURN
```

The Tosie Hacker #1 - Programmable Character Generator & more!

Features;
- 40 to 48 pin connector, board runs on the OSI 48 pin bus or the C1 40 pin bus while acting as an adapter between the two. (unbuffered)
- two AY-3-8910 programmable sound generators, if your not familiar with this chip visit an Arcade and listen.
- 1024 programmable 8x8 characters in addition to the original 256 OSI characters. (Read this line again,1024!)
- Optional external control will allow selection of any and or all of the 1280 characters on the screen at the same time!
- 1-2MHz dual speed clock with hard and soft control to select speed. This clock also allows switching speeds while running, without the CPU ever hanging up.
- Compatible with any video system using an OSI style character generator rom. e.g. C1,540,seb-3,etc.
- 12k (or more?) addressable space for 2716 Eproms and/or 6116 style static rams.
- what ever else I have time and or the space for.
- Low cost! this project is to be sold at cost.
- Estimmated cost $35.00 for bare board.
- Cost of populating this board could be substantial, but will work fine with limited parts leaving room for future expansion to the full features of the board

While this board is still in the development stages, the majority of the circuits have already been tested in the prototype. The date the board will be ready for sale is still uncertain, it is estimated that the first lot will be ready for delivery sometime in August. The demand for this board is going to be difficult to forcast, since this board is to be produced at cost there will be few boards produced in excess of those for which we have received anadvance deposit. If you don't order a board now you will have to wait 6 to 10 weeks before a second run will be ordered. If you have any questions call Paul C. 519-925-5561 or write to our club's address. The deposit $5.00 for orders can be paid at our meetings or mailed to our address, you can make cheques payable to T.O.S.I.E.

A REAL TIME CLOCK

Here is a no hardware (almost) real time clock for Superboard. As described here it will work with the OSI monitor only, but by using different memory locations to hold the hours, minutes, and seconds you can make it compatible with any system.

As a bonus, there is a demonstration of a method of tacking a machine language load right onto a BASIC load. Even more, there is a demo of an auto-run routine, and a way to have the computer go back to the Program after doing a LIST. Usually it gives you the O.K. message after doing a LISTing.

HOW IT WORKS:

A wire is run from the TV sync signal to then NMI (Non Maskable Interrupt-see a 6502 manual for more information) Pin of the 6502. Every sixtieth of a second this Pulse interrupts the Program and increments the clock as needed. You will not notice that this is being done, as the machine code does this very quickly. As a matter of fact, I did not even have to turn it off during cassette saves or loads, however, this might be a wise Precaution.

WHAT TO DO - HARDWARE:

Run a wire from U65 Pin 4 through an SPST switch to Pin 2 of the exPansion connector. Leave the switch OFF until the software is entered (otherwise the interrupt tries to execute a Program that isn't there yet). That must be the easiest modificatin to your machine you have ever done! If you can't find U65, a 74LS123 chiP, get helP with this work. It will only take a minute.

SOFTWARE:

Use the OSI or other monitor to enter the code as in the block dumP, or as you have modified it. TyPe in the BASIC, both the clock demonstration, which will Prove it all works, and the Portion at line 50000 on that makes an auto machine language loader.

You can test the clock (after all the code is in) by flipPing the switch to on , and tyPing RUN. Answer the questions to reset the time,and watch the time displayed on the screen. Of course you will want to develoP your own software to use this new feature on your machine. Maybe a timed skill test, checking the speed of a routine or other uses such as controlling the lights via a BSR or other interface are Possible.

To make a self loading coPy of this Program AND the machine code from $0130 to $0173, all you have to do is type: RUN 50000, Press record, and then Press RETURN. An autoload version of the Program will be made. Don't turn off the recorder until the OK message. You will not see anything happening on the screen while the ML is being written to taPe. The Basic routine used would be too slow.

Software exPlanation:

Memory locations $E5 to E0 are reserved for the real time clock. Put these elsewhere for ROMTERM or other monitors that use this area of memory.
$E4 and $E5 are used to temporarily save the X and A register during the interrupt.
$E3 stores the number of sixtieths of a second. Retrieve it by a PEEK to location 227 (decimal)
$E2 stores the number of seconds. PEEK location 226
$E1 stores the number of minutes. PEEK location 225
$E0 stores the number of hours. PEEK 224

The machine code at $0130 up, basically stores the A and X registers, increments $E3, the sixtieths counter. If this is 60, then it reset this to zero, but adds one to the seconds counter,...and so on for the minutes and hours.

I have added two Places to adjust the clock. I found that my clock was running quite slow. This is due to the fact that the vidio sync is only approximately a sixtieth of a second, as derived from the computer's crystal. The 01 at $0148 adds an extra sixtieth of a second every minute. You can increase this if your clock is a too slow. There is another 01 at $0153. This adds a sixtieth of a secondevery hour, and can be changed for really fine adjustments. With these two values at 01 my clock ran as accurately as my LCD wristwatch. Your crystal frequency may be a little different, so adjustments may be needed.

Exit from the routine is at $0164, by reloading the X and A registers,and doing an RTI, return from interrupt, instruction.

The Basic Program from 0 to 90 is quite straightforward. It is a demonstration to Put the clock on the screen. Line 0 makes sure the computer is no longer in SAVE. A B C and D are memory locations 224, 225, 226 and 227 that store hours, minutes, seconds and sixtieths. These are PEEKed, converted to strings, and then Poked onto the screen in line 90. A simple routine.

The autoloader routine is a little trickier. As you know, the normal response after LIST is for the computer to say OK. If we want angther statement to be executed, we will have to Point the vector at location 4 and 5 to a CONT command instead. This is what happens in line 50010. Let's start again.

50010The SAVE sets the cassete outPut Pointer, but before LIST is executed in hte next line, the Program is Pointed at the CONT code by POKEing the appropriate address into locations 4 and 5. The computer will now CONTinue insteadof saying OK.
50015LIST is executed, but the Program continues. First we will reset the Pointer back to it's original setting. (note- Change this for ROMTERM or you will crash)50020 Puts line 0 in. POKE 515,0 turns off the LOAD
50040This Puts the following commands onto the tape. They will be executed as if you typed them from the keyboard. If the LOAD flag is set, the cassette inPut can operate the computer as if it was

the keyboard. The machine doesn't know the difference. This gimmick saved OSI a lot of monitor space. Immediate commands executed from tape:
POKE 251,1 same a L , to set monitor load
The POKES to 11 and 12 set up the user vector to the monitor,which can be entered at $FE43. The X=USR(X) does that.

The rest of the program writes the ML to the tape. If you look you will see that it is identical to you entering the code manually-except that the entering device is the tape- via the above mentioned gimmick.

50050 set the memory to $0130, the / starts data entry.
50060 Data from 304 decimal ($0130 hex) to 372 ($0173) is to be read, converted to hex High and Low bytes by the calculations in 59119 and 50120, then written to the cassette port H byte firts, Low next, then a carriage return (13). These things are done in 50130 to 50150.

The last thing executed is a .0169G (see line 50090). Yes, exactly as if it came from the keyboard, it starts executing at $0169. Here is a RUN command executed from a ML program. The code at $0169 and $016B, turn off the Load thatt the monitor was in, and the next two lines of code, at $016E and $0171 set up and execute the RUN.

I know that this article is quite a lot to chew if you are a beginner. You may be able to work it through with the help of the various other resources that you have. If not, why not write me a note asking for clarification of this or that point. I'd be glad to explain the details of the above if someone asks. On the other hand, if you all have it figured out there is no sense my wasting valuable space.

Lastly, if you get this running properly, you can, and this would be the ideal solution, put the real time clockinto EPROM, and thus it would always be available.It will also avoid the shock you get if you turn on the interrupts (remember the switch?) before putting the software in. Your machine will behave rather strangely!You are forewarned! The cure of course is to flip the switch only after entering the software.

```
4
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0130  85 E4 86 E5 A9 3C A2 00 E6 E3 C5 E3 D0 26 86 E3
0140  E6 E2 C5 E2 D0 1E 86 E2 A2 01 86 E3 E6 E1 C5 E1
0150  D0 12 A2 01 86 E3 A2 00 86 E1 A9 18 E6 E0 C5 E0
0160  D0 02 86 E0 A5 E4 A6 E5 40 A9 00 8D 03 02 20 77
0170  A4 4C C2 A5
:Q0130
0130 85E4    STA $E4
0132 86E5    STX $E5
0134 A93C    LDA #$3C
0136 A200    LDX #$00
0138 E6E3    INC $E3
013A C5E3    CMP $E3
013C D026    BNE $0164
013E 86E3    STX $E3
0140 E6E2    INC $E2
0142 C5E2    CMP $E2
0144 D01E    BNE $0164
0146 86E2    STX $E2
0148 A201    LDX #$01
014A 86E3    STX $E3
014C E6E1    INC $E1
014E C5E1    CMP $E1
0150 D012    BNE $0164
0152 A201    LDX #$01
0154 86E3    STX $E3
0156 A200    LDX #$00
0158 86E1    STX $E1
015A A918    LDA #$18
015C E6E0    INC $E0
015E C5E0    CMP $E0
0160 D002    BNE $0164

0162 86E0    STX $E0
0164 A5E4    LDA $E4
0166 A6E5    LDX $E5
0168 40      RTI
0169 A900    LDA #$00
016B 8D0302  STA $0203
016E 2077A4  JSR $A477
0171 4CC2A5  JMP $A5C2
0174 60      RTS
```

```
 0 POKE515,0
 5 FORI=1TO30:PRINT:NEXT
 10 REM CLOCK DRIVER
 20 A=224:B=A+1:C=A+2:D=A+3:SC=53419
 22 INPUT"RESET TIME Y/N";N$
 23 IFN$="N"THEN 40
 25 PRINT:PRINT
 31 PRINT:INPUT"TIME IN H,M,S";H,M,S
 35 POKEA,H:POKEB,M:POKEC,S:POKED,0
 40 FORI=1TO30:PRINT:NEXT
 42 H$="TIME:":FORI=1TOLEN(H$)
 43 POKE53414+I,ASC(MID$(H$,I,1)):NEXT
 50 H=PEEK(A):H$=STR$(H):IFH<10THENH$=" 0"+RIGHT$(
H$,1)
 60 M=PEEK(B):M$=STR$(M):IFM<10THENM$=" 0"+RIGHT$(
M$,1)
 70 S=PEEK(C):S$=STR$(S):IFS<10THENS$=" 0"+RIGHT$(
S$,1)
 80 J=PEEK(D):J$=STR$(J):IFJ<10THENJ$=" 0"+RIGHT$(
J$,1)
 90 T$=H$+M$+S$+J$:FORI=1TO12:POKESC+I,ASC(MID$(T$
,I,1)):NEXT:GOTO50
 50000 REM ROUTINE TO RECOVER AFTER LIST
 50010 SAVE:POKE4,194:POKE5,165
 50015 LIST:POKE4,195:POKE5,168
 50020 PRINT:PRINT"0POKE515,0
 50030 REM TURN ON THE MONITOR
 50040 PRINT"POKE251,1:POKE11,67:POKE12,254:X=USR(
X)
 50050 PRINT".0130/";
 50060 A1=304:A2=372:GOSUB50100
 50090 PRINT".0169G":END
 50095 REM WRITES M/L TO TAPE
 50100 FORA=A1TOA2
 50110 OP=PEEK(A):H=INT(OP/16):L=OP-16*H
 50120 H=H+48-7*(H>9):L=L+48-7*(L>9)
 50130 WAIT61440,2:POKE61441,H
 50140 WAIT61440,2:POKE61441,L
 50150 WAIT61440,2:POKE61441,13
 50160 NEXTA:RETURN:
```

ME AND MY OSI - RAMBLINGS

All of us must have wondered a few times in the last few weeks why we are still at the keyboard of this primitive device called the OSI Superboard, C1, or C4. The loudly heralded birth of the ADAM computer from COLECO got me wondering again.

COLECO will get you a machine with tons of RAM (more than I ever could stuff into this machine), some sort of mass storage device (not a cassette I gather), and a LETTER QUALITY printer for a measly $800 US. Included in firmware is a word processing package! Now while us Canadian citizens will have to pay $1200 for this package, I can tell you that I have as much or more Canadian funds sunk into my OSI, and I still haven't started the disk controller board for the used disk I have.

With the possibility of getting an APPle with disk running for less than a Thousand, you can see that I sometimes think it is time to call it quits with my OSI.

ENOUGH of THAT NONSENSE! This machine is quite capable! It does have a relatively fast (for 8 bit) Processor. Most of all, I have learned quite a bit about the innards of this machine and it's Processor, and it would take a serious effort to get into another machine. Not that it could ever be so much of a challenge again to search out the documentation, and even the most elementary information. No one could be as elusive as OSI on that subject.

So here is the sum. I am staying with this machine for a while. It is still challenging, and that is one of the main reasons that I bought it. I can Play around with its innards. Things are easy to change and get at with the truckloads of TTL chips inside. I mean, how do you get the horizontal sync out of a large scale integrated chip? I'll stick it out for a while yet. I am still enjoying this machine.

Now, if only my Wintario number...Hmmm...

A MAILING LIST FOR SUPERBOARD

You can use the cassette based system to keep such a thing as a mailing list. There are a few disadvantages, and you should have at least 16k of RAM, but it can easily be done.

The trick is to read the whole file into memory, add, delete, etc as needed, then write a new copy to tape. It may be wise to save the copy from last time, in case of errors. The easiest way to do this is to keep two tapes, one for this run, and the one from last time.

The data can easily be stored onto tape by printing each field to tape, with a carriage return between. The BASIC input statement is handy for reading them back, but does introduce a few problems of its own.

For instance, commas and quotation marks end an input statement. This can be overcome by not allowing these, or by changing them to another character before storing them on tape, then reconverting them later.

The program below is quite short, yet is capable of sorting through your list alphabetically, saving and loading the file, deleting entries, and the like. It will keep about 30 names and addresses in an 8k machine. About 90 names fit onto a 16k machine. I have kept the TOSIE mailing list with this program, and printed the mailing labels with it.

If nothing else you could use it to keep your Christmas card list, print out the labels for them, and prove to your wife that there really is a use for this thing.

The program is self documenting and will give you directions on it's use. Note that a corrected BASIC 3, the infamous garbage collector chip, is required. Ask at the club meetings if you don't know about this one.

```
1 ME=160:GOTO14
2 X=USR(1):I=PEEK(531):I$=CHR$(I):RETURN
6 IFM<2THEN55
7 PRINT:PRINT"ALPHABETIZING":M=ME
8 IFM$(M,0)=""THENM=M-1:GOTO8
9 FORD=1TOM-1:IFM$(D,0)<=M$(D+1,0)THEN13
10 PRINT"*";:FORG=DTO1STEP-1:IFM$(G,0)<=M$(G+1,0)THEN12
11 FORH=0TO6:T$=M$(G,H):M$(G,H)=M$(G+1,H):M$(G+1,H)=T$:NEXT
12 NEXT
13 NEXT:GOTO55
14 DIMM$(ME,6),C$(6):D$="DONE":POKE11,0:POKE12,253:C$(1)="NAME"
16 C$(2)="ADDR":C$(3)="CITY":C$(4)="CODE
17 C$(5)="TEL#":C$(6)="NOTE":GOTO50
18 IFN<>1THEN29
19 IFM$(M,1)=""THENM=M-1:GOTO29
20 A$=M$(M,1):L=LEN(A$):FORI=LTO2STEP-1
21 IFRIGHT$(A$,1)=" "ORRIGHT$(A$,1)="."THENA$=LEFT$(A$,I-1)
22 IFRIGHT$(A$,1)<>" "ANDRIGHT$(A$,1)<>"."THENI=2
23 NEXT:L=LEN(A$):FORI=L-1TO2STEP-1
24 IFL<2THENM$(M,0)=A$:GOTO29
25 IFMID$(A$,I,1)=" "ORMID$(A$,I,1)="."THENM$(M,0)=RIGHT$(A$,L-I):I=2
26 NEXT
27 L=LEN(M$(M,0))
28 IFLEFT$(M$(M,0),1)=" "THENM$(M,0)=RIGHT$(M$(M,0),L-1):GOTO27
29 N=N+1:IFN>6THENPRINT:PRINT:M=M+1:PRINT"  ENTRY"M"?":PRINT:N=1
30 IFM>METHENPRINT"DONE"M:PRINT"OUT OF MEMORY":GOSUB7:GOTO121
32 A$=M$(M,N):PRINTC$(N)" "M$(M,N);
33 GOSUB2:IFI=17THENM$(M,N)=A$:GOTO6
34 IFI=13THENM$(M,N)=A$:PRINT:GOTO18
37 IFI=127THEN42
38 IFI=27THEN46
39 IFI<32ORI>90THEN33
40 PRINTI$;:A$=A$+I$:IFI=13THEN34
41 GOTO33
42 L=LEN(A$):IFL<2THENA$="":GOTO44
43 A$=LEFT$(A$,L-1)
44 PRINTCHR$(13)"                                    "CHR$(13);
45 PRINTC$(N)" "A$;:GOTO33
46 N=N-1:IFN=0THENN=6:M=M-2:GOTO48
47 PRINT:GOTO32
48 IFM<1THENM=0
49 GOTO29
50 FORI=1TO30:PRINT:NEXT:PRINT"     MAILING LIST":PRINT
51 PRINT:PRINTTAB(7)"John Horemans":PRINTTAB(7)"c1981":PRINT:PRINT
52 PRINT:PRINT"Use CTRL/@ for menu
53 PRINT"     ESC to reverse
54 PRINT"     RUB OUT to edit
55 PRINT:PRINT:PRINT:PRINT"  Enter"TAB(12)"Reload
56 PRINT"  Delete"TAB(12)"Correct
57 PRINT"  List"TAB(12)"Names":PRINT"  Printout"TAB(12)"Mailing
58 PRINT"  Save"TAB(12)"Quit":PRINT
59 REM
60 PRINTCHR$(13)"Your command Please? ";:GOSUB2:PRINTI$;
61 IFI$<>"E"THEN64
62 FORI=1TOME:IFM$(I,1)=""THENM=I-1:I=ME
```

```
63 NEXT:N=6:GOTO29
64 IFI$<>"R"THEN72
65 PRINT:PRINT:PRINT"Press PLAY then RETURN
66 X=USR(1):LOAD
67 INPUTX:IFX<>0THEN67
68 INPUTX:IFX<>255THEN68
69 FORM=1TOME:FORI=0TO6:INPUTM$(M,I)
70 IFM$(M,I)=D$THENM$(M,I)="":M=ME:I=6
71 NEXT:NEXT:POKE515,0:GOTO55
72 IFI$<>"D"THEN80
73 NA=1:GOSUB 126:IFNA=0THEN55
74 PRINT:PRINTM" "M$(M,1):PRINT:PRINT"ARE YOU SURE Y/N?":X=USR(1)
75 IFPEEK(531)<>89THEN55
76 PRINT:PRINT"DELETING "M$(M,1)
77 FORM=MTOME-1:IFM$(M,1)=""THEN79
78 FORI=0TO6:M$(M,I)=M$(M+1,I):NEXT
79 NEXT:GOTO55
80 IFI$<>"C"THEN82
81 GOSUB126:N=0:GOTO29
82 IFI$<>"N"THEN88
83 GOSUB122:PRINT:PRINT
84 K=0:FORI=1TO24:PRINTMTAB(5)M$(M,1):M=M+1
85 IFM>METHENI=24:NEXT:X=USR(1):GOTO55
86 NEXT:X=USR(1):IFPEEK(531)=17THEN55
87 GOTO84
88 IFI$<>"P"THEN97
89 PRINT:PRINT:PRINT" Adjust the Printer":PRINT
90 PRINT"    then Press RETURN
91 X=USR(1):SAVE:FORM=1TOME:IFM$(M,1)=""THEN96
92 PRINTM$(M,1)TAB(25)M$(M,2)TAB(50)M$(M,3)TAB(70)M$(M,4)CHR$(13);
93 PRINTM$(M,5)TAB(25)M$(M,6)CHR$(13)CHR$(13);
96 NEXT:POKE517,0:GOTO55
97 IFI$<>"M"THEN106
98 PRINT:PRINT:PRINT" Adjust the Printer":PRINT
99 PRINT"    then Press RETURN
100 X=USR(1):SAVE:FORM=1TOMESTEP2:IFM$(M,1)=""THEN105
101 PRINTM$(M,1)TAB(35)M$(M+1,1)CHR$(13);
102 PRINTM$(M,2)TAB(35)M$(M+1,2)CHR$(13);
103 PRINTM$(M,3)TAB(35)M$(M+1,3)CHR$(13);
104 PRINTM$(M,4)TAB(35)M$(M+1,4)CHR$(13);:PRINT
105 NEXT:POKE517,0:GOTO55
106 IFI$<>"S"THEN112
107 PRINT:PRINT:PRINT"Press RECORD then RETURN
108 X=USR(1):SAVE:FORI=1TO5:PRINT0:NEXT:PRINT255
109 FORM=1TOME:FORI=0TO6:IFM$(M,1)=""THENM=ME:I=7:GOTO111
110 PRINTCHR$(34)M$(M,I)
111 NEXT:NEXT:PRINTCHR$(34)D$:POKE517,0:GOTO55
112 IFI$<>"L"THEN120
113 GOSUB122
114 IFM<1ORM>METHEN55
115 PRINT:PRINT:FORI=1TO6:IFI=1THENPRINT"ENTRY #"M:PRINT
116 PRINTC$(I)" "M$(M,I):NEXT
117 X=USR(1):I=PEEK(531):IFI=17THEN55
118 IFI=27THENM=M-1:GOTO114
119 M=M+1:GOTO114
120 IFI$<>"Q"THENPRINTCHR$(13);:GOTO59
```

```
 121 PRINT:PRINT:PRINT"GET IN WITH A GOTO 55":END
 122 PRINT:PRINT:PRINT"FROM LETTER? ";:GOSUB2
 123 IFI$<"0"ORI$>"Z"THEN55
 124 PRINTI$;:FORI=1TOME:IFM$(I,0)>>I$THENM=I:I=ME
 125 NEXT:RETURN
 126 PRINT:PRINT:PRINT"WHICH NAME?":PRINT:PRINT"      ";:A$=""
 127 GOSUB2:IFI=17THENNA=0:RETURN
 128 L=LEN(A$):IFI=127ANDL>1THENA$=LEFT$(A$,L-1):GOTO132
 129 IFI=127ANDL<2THENA$="":GOTO132
 130 IFI=13THENPRINT:PRINT:GOTO133
 131 PRINTI$;:A$=A$+I$:GOTO127
 132 PRINTCHR$(13)"                    "CHR$(13)"     "A$;:GOTO127
 133 NA=1:FORI=1TOME:IFM$(I,1)=A$THENM=I:RETURN
 134 IFM$(I,0)=A$THENM=I:RETURN
 135 NEXT:NA=0:RETURN
READY
```

Rob Leathley
35 Trueman St
451-9760

Bob Stafford
8 Kline St
846-0797