99'er Online

October 1984 Edition

P.O.Box 11983
Edmonton, Alberta
Canada T5J 3L1

TO:

THE EDMONTON 99'er COMPUTER

USERS' SOCIETY

Date / /

New Membership Application ☐

Renewal of Membership ☐

Change of Address ☐

NAME _____
last first initial

ADDRESS _____

CITY _____ PROVINCE or STATE _____

POSTAL or ZIP CODE _____ TELEPHONE _____

Single or Family membership	$20	(12 months)	☐
	$15	(6 months)	☐
Student membership (must have student I.D. number)	$15	(12 months)	☐
	$10	(6 months)	☐
Out of Town (newsletter only)	$20	(12 months)	☐
	$15	(6 months)	☐

* Please note that all membership money should be in CANADIAN FUNDS.

* May we release your name, address and telephone number on a club roster, to companies and other clubs?

YES ☐ NO ☐

FOR OFFICE USE ONLY

Date Filed / / Membership Number _____

Amount Tendered $ _____ Authorized Signature _____

99'er ON LINE is the news letter of the Edmonton 99'er Computer User's Society published ten times a year. All material contained in this news letter may be published in other news letters provided that source and author are identified unless otherwise stated. We welcome correspondence from all TI User Groups and will extend the same source credit courtesy.

Mail news letter correspondence to:

BOB PASS - EDITOR
59 L[illegible]E CRES.
ST. ALBERT, ALBERTA
CANADA
T8N-2G6

All other correspondence should be addressed to:

EDMONTON 99'er COMPUTER USER'S SOCIETY,
P/O BOX 11983
EDMONTON, ALBERTA
CANADA
T5J-3L1

OFFICERS

PRESIDENT-----BILL CA[illegible]
VICE P-ES-----PAUL HELWIG
[illegible]-----EVAN SMITH
SECRETARY -----SUSAN LIVINGSTON

Worthy assistants; Bob Pass (Editor), Bob Burley (Cassette Library and electronics wizardry), Gord Bradlee (Printed Media Library), Tom Hall (Disc Library), and all of our members who contribute articles to this news letter!

DISCLAIMER

All information published in this news letter is, for the most part, the fruits of the labors of amateurs; therefore, we cannot guarantee that the information presented is always correct.

REGULAR MEETINGS

Regular meetings of the Edmonton User's Group are held on the second Tuesday of each month on the 8'th floor of the General Services building of the University of Alberta from 7:00 till 10:00 PM and are open to all members in good standing. Non-members may attend their first meeting free of charge.

The Executive Committee meets monthly also and members may attend these meetings as observers or to address a particular issue. Please arrange with one of the officers listed above if you wish to attend.

ADVERTISING

Commercial advertising space is available in this news letter at the following rates:

FULL PAGE ---- $20.00
HALF PAGE ---- $15.00
1/4 PAGE ---- $10.00

Please discuss your commercial needs with Paul Helwig at the next meeting or write to the P/O Box above.

Members may advertise their personal computer related items for free but are asked to limit their ads to about 20 words. Please mail your ads to the EDITOR'S ADDRESS,

see above, or hand it to him at the general meeting; ads received by the 15'th will appear in the next news letter.

MEMBERSHIP FEES

FAMILLY - 12 MONTHS - $20.00
" - 6 " - $15.00
STUDENT - 12 MONTHS - $15.00
" - 6 " - $10.00

WORKING WITH FORTH

by: Tom Hall and Michal Jaegermann

While most of us probably let our computers collect dust during the fabulous summer we had, a few of us continued to plug away at uncovering some of the many mysteries still to be discovered in the TI99/4A.

This summer the major thrust of our activity was with TI FORTH. After the usual number of blown disks and frayed nerves, we managed to come up with a couple of useful utilities for FORTH: a Fast Copy routine and a disk zapper. In the process of writing the disk zapper, we discovered a number of things about the way the TI disk is put together that we didn't know before, some of which we shared with you in the September issue. One thing interesting to note is that, unlike most other programming languages which access files, TI FORTH's I/O is on a sector-by-sector basis, not by files. It is possible to use FORTH to write a conventional file, but FORTH will quite happily look at any part of a disk, ignoring the fact that it either is or is not part of a file. This makes it quite easy to accidentally erase something that makes it necessary to go back and start again.

Another aspect of FORTH that puts it in a class by itself is it's portability -- that is, a routine written in FORTH on one computer is frequently adaptable to another CPU with minimal modifications. We discovered this for ourselves this summer when we downloaded a FORTH application written by Mr. Warren Ward in the 8086 version of FORTH, and we were able to get it running on the TI with almost no modifications at all.

One thing which we tackled fairly early on in our work with FORTH was the 64 column editor. We both agreed that the white letters on a black screen with a dark blue strip at the bottom left a lot to be desired. For those of you who use ordinary TV sets as monitors, one of the "facts of life" we have to put up with most of the time is something called "overscan". This condition causes the picture to appear to "bloom", and usually makes it difficult or impossible to see the far left and/or right columns on the screen. Depending on the type of TV you use, you may have a particular color combination which is both personally pleasing and which allows you to see the extreme edges of the screen.

In the 64-column editor there are five separate parameters which you can set individually: the color of the bit-map (or "tiny") character set, the background color of the upper portion of the screen, the color of the cursor, and the foreground and background colors of the split-screen strip at the bottom of your screen. The two FORTH words which accomplish all these things are VWTR and VFILL. VWTR is the command which, if you know much about TI's assembler, you will recognize as the same as the word that instructs a program to write to one of the registers in the VDP portion of your console's RAM. The word VFILL operates in a similar manner, except that it will write a specified character beginning at a specified location in VDP memory, and repeat that character through a specified range of consecutive memory

locations.

There are three ways that you can change these parameters of the 64-column editor. You can enter the commands for the color combinations you want manually from the keyboard, or you can write a word that will do the same thing, and store it in one of your screens on disk, or you can modify certain lines in the 64-column editor screens directly. The format of the three commands is as follows (in all cases it is assumed that the necessary values will be entered in hexadecimal format):

b 7 VWTR where "b" is the the background color of the upper portion of the editor screen

0 1000 f0 VFILL ("f" is the hex number corresponding to the desired color of the tiny character set)

1000 800 fb VFILL ("f" and "b" are the foreground and background colors of the strip at the bottom of the screen.

If you prefer, you can write a FORTH procedure using these routines, and simply execute the word or words to change the color of your editor. If you decide that you would like the change to be of a more permanent nature, you can make a couple of changes to Screen #23 on your system disk. Line #2 of that screen begins with a definition of CINIT, which is the word that sets up all the default colors of the 64-column editor every time the editor is called. Very carefully extend the definition of CINIT by adding your version of the VFILL and VWTR words mentioned earlier into the empty space on line #6 of screen #23. When you've done that, you should make sure to move the "; DECIMAL" from the end of the preceding line so that they fall after the modifications you have made. If you use the minimum of one space between words you should have no problem getting all of this to fit on the one line. The flashing cursor's color is defined by the statement in line #5: "0 1 F 5 0 SPRITE". The "F" is the hexadecimal word for 15, which represents white; simply change that value to whatever looks good to you with all the other changes you make, and that will take care of the cursor color! Once you have flushed these changes to disk, you have made a permanent change to your 64-column editor's appearance, and no matter how you subsequently modify these colors in immediate mode or from other applications, every time you type

the word ED" or [screennumber]
EDIT, you will have these colors back again!

GEMINI PRINTER OWNERS

As all of you Gemini owners are probably aware, your instruction book leaves us TI owners out in the cold. However, as reported in several news letters from the States, the Gemini folks have written an addendum to their manual specifically for TI/994A use. Included are demo programs and all the pin-outs to interface parallel or serial ports on the TI RS232 card. Write to the following address requesting this freebie:

STAR MICRONICS,
NO. 3 OLDFIELD,
IRVINE, CA 92714
(714) 768-4340

BEGINNING BASIC PROGRAMMING

by: Bob Pass

For the beginning programmer, one of the toughest concepts to understand is the use of variable names. This is especially true for those of us who had trouble with algebra in school or for those too young to have yet been exposed to it's symbolism.

To get started, you should read pages 13 to 17 of your Beginner's Basic manual which came with your computer. This month I will concentrate on numeric variables leaving string variables to a later article.

In BASIC, or for that matter any computer language, data that is to be processed must be stored in the machine's memory. The memory can be visualized as a system of pigeon holes with each pigeon hole having a unique identifcation. In computerese, these pigeon holes are called "Memory Locations" and the identification is called the "Address". The computer can retrieve or write information to these locations by using the address to index it's memory store. A computer with 16,000+ words (locations) of memory would present us as programmers with quite a problem if we had to keep track of where we had stored data. For example, if we had stored two numbers and wished to add them, we would have to know the addresses (pigeon hole box numbers) of the memory locations where the numbers were stored so we could tell the machine where to get the numbers to be added.

Fortunately, a short cut has ben provided in Basic which allows the computer to administer it's own memory without the programmer having to be concerned about it. This short cut utilizes variable names which are really tags that we are using to label some of those pigeon holes instead of using the memory address. For instance, suppose that we wished to write a program that would add two numbers and print the result. The values of the numbers would be dependant on some other routines in the program so we can't fix their value at the time the program is written. The first time we wish to make a reference to one of the numbers in the program, we will give it a name; you can pick any name you want following a few rules which you should review by reading page II-11 of your User's Reference Guide.

Let's give the first number the name "A" and assign a value to it. This can be done by using the LET, INPUT, READ, or (with X-BASIC) the ACCEPT AT statements. (Refer to the User's Reference Guide pages II-45, 59, and 61 for instructions on how to correctly use these statements). For instance the BASIC statement LET A=14 would cause the computer to take the following actions as it is setting up memory (that pause when you type "RUN"):

1. Set aside a location in memory to hold numeric data.
2. Place the name of this area and it's address in a table of variable names.
3. Write the value assigned (14) into the memory area.

Now, let's call the second number "B" but it's value will be dependant on the user's input. Therefore, we cannot assign a value to B using the LET statement. The INPUT statement will allow the user to assign a value to a variable while the program is running. The statement

```
INPUT "ENTER VALUE FOR B";B
```

would cause the computer to set up memory just as it does for the LET statement. However, because there is no value for B when the variable table and memory space are set up, the computer will automatically assign B a value of zero by writing that into the memory space reserved

for B. During program execution, when the computer comes to the INPUT statement, it will display the instruction "ENTER VALUE FOR B" and then wait for the user to type in a value. When the computer receives this value, it will look up the name "B" in it's variable table to obtain the address of the memory location reserved for the number called "B". The value entered by the user will now be written into that location.

The two numbers are now in their respective memory pigeon holes and they are tagged with distinctive names of our choice. Now we can manipulate these numbers simply by referring to their names rather than their actual value. For instance, to add the two numbers and display the answer we could use the statement

PRINT A+B

which would cause the computer to execute the following:

1. Look up the name "A" in the variable table and get the address of the memory space for "A".
2. Get the value of "A" and place it in an adding register in the arithmetic area of the computer.
3. Repeat the above steps for "B".
4. Add the two numbers and display the result on the screen.

The current values of A and B remain in their respective pigeon holes unchanged and can be used or modified many times under our control simply by referencing their names. I hope that this will give some of you a better idea of what variable names are and how the computer uses them. Don't let the concept frighten you as you will find that it is not all that difficult once you have experimented with it for a while. The best way to learn programming is to sit down and do it! In the near future, I will do a tutorial on string variables.

LITTLE GEM

This little gem, by Jim Peterson of Tiger Cub Software, was spotted in the Miami County Area User's Group news letter:

To obtain an arcade effect in your musical tones or single note music, structure your CALL SOUND command as follows:

CALL SOUND(D,N,V,N*1.01,V)

("*" is an asterisk).

DISK CATALOGING UTILITIES

by: Tom Hall

There are perhaps two generalities which can be made about the majority of serious computer hobbyists (and TI owners are certainly no exception!): (1) most eventually purchase a disk drive of some sort; and (2) having done that, sooner or later that person will be in the market for some means of keeping efficient and readily accessible records of just what he has stored on his disks. Depending on individual requirements and preferences, the hobbyist may be content with the listings he can produce using his DISK MANAGER module (assuming, of course, that he owns a printer), or he may want something which boasts a few more "bells and whistles." Also, for those poor souls who happen not to own a printer, there is the need to be able to store information about library contents on disk.

There are three such disk-cataloging packages that I would like to discuss here. The first two are from J K H

Software in Arlington, Virginia. The is called the MULTI-DISK INFORMER. It requires EXTENDED BASIC and 32K memory expansion, and retails for about $35 U.S. It has the capability of creating disk files to store information about your cataloged disks. It comes ready-made for a multi-drive system, and will print various summary statistics about the disks you are cataloging on your screen as the disks are being put through. One unique feature of this package is that it will allow you to exclude up to 8 filenames that you don't want appearing in the final printout (like, how many of your disks have a program called LOAD?), although these filenames will be included for the purposes of statistical information, and the printout will inform you of which disks these excluded programs are actually on. At the end of the cataloging process, after you've been given the option of saving all this information to one of the five pre-prepared files on disk (they come with the system and are already named and set up), you have the option of searching the file for a specified filename, and then you are asked if you want a printout. If you say no, the program abruptly ends, and that's that. One thing about the program I don't like is the fact that there is no provision for listing to the screen information that is already stored on disk; the only way to see what is in the file is to print it.

Another utility from the same company is called SUPER CATALOGER, and was actually the forerunner of MULTI-DISK INFORMER. This program likewise requires EXTENDED BASIC and 32K Memory Expansion, but, unlike MULTI-DISK INFORMER, SUPER CATALOGER does not create any files. It merely lets you keep cataloging your disks (until you run out of either disks or memory), and then sorts and prints the result. This program displays similar information on your screen as you catalog your disks, but that's all; once you've cataloged a disk, you won't see that information again unless you print it. About the only thing I can say for this program is that it has a very fast sorting routine and the printout makes efficient use of paper.

I have deliberately saved the best for last: MASTER DISK FILE, from Extended Software, also an American company. It's only system requirement is EXTENDED BASIC, and at $15 U.S. is, in my opinion, by far the best value for your money, and you don't even have to own a printer to make full use of it's capabilities. MASTER DISK FILE uses almost the entire 90K of a single-sided diskette as virtual memory, and has the capability of storing up to 120 disk titles and/or 1100 program titles. You can get all the information this package has to offer displayed either to your screen or to printer, and there are three listing options: you can have an alphabetical listing of disknames, showing the number of sectors used and available on each, as well as the number of files; or you can have an alphabetical listing of every file title displayed; or you can get a complete listing of every disk, arranged alphabetically by diskname, in a format similar to that which is produced by the DISK MANAGER.

The entire program is menu-driven, and you are also given the option of storing information about your system, such as whether you are running a one-, two-, or three-drive system, and the specifications of your printer, including any special control codes unique to your particular requirements. There is even a feature, especially useful for those without printers, which allows you to search the files for a specified disk or program title -- and lookup time is fast!! Less than 25 seconds from a running program!

FUNNIES

One of the advantages of being disorderly is that you are constantly discovering new things!

LETTER TO THE EDITOR

Dear Editor:
I think that two articles in last month's news letter do require some comments.

First, the article entitled "FLOPPY DISK STORAGE" by Thomas Island is a little misleading. It actually describes a well known method of making "floppies" into "flippies" to double the capacity of diskettes using single sided drives. Well, in the May/84 issue of "The Smart Programmer", Craig Miller of Miller Graphics gives a very emphatic DON'T to the procedure. He explains that on single sided drives there is a felt pressure pad used on the back side of the disk to keep the disk pressed against the read/write head. This pad eventually picks up minute specs of dust and grit which begins to grind into the disk surface. As long as this surface is never used to record data, no harm is done. Also, the inside of the disk jacket is lined with a soft fibrous material designed to pick up lint and dirt from the disk as it spins in the jacket. When you flip the disk, it will spin in the opposit direction and trapped dirt can be pulled off the lining back onto the disk where it further contaminates the felt pressure pad and/or the head. I realize that many people are using "flippies" and report no ill effects. Bear in mind though that the damage is accumulative over a period of time and the end result can be catastrophic. Any one contemplating this practice would be wise to check both sides of the coin.

Secondly, the article "RADIO SHACK DISK DRIVE" by the same author, Thomas Island. Some of the TI Disk Controller cards (PHP 1240) will support double sided drives; if you received a Disk Manager 2 module with your disk system, you may have double sided support for double sided drives. (Thus eliminating use of "flippies"! -Ed).

99'ers ON LINE OCT/84

I have been using non-TI drives (ie not Shuggart) which are double sided to successfully access both sides of my diskettes with out any problem for some time (barring some silly and easily repaired bug in TI FOR[illegible]. The Disk Controller will not support double density storage. However double density drives will work at the single density level. Right now there is a double density controller available from CorComp Inc. but this card is not compatible with the Foundation 128K Memory Expansion card. Apparently Fondation is working on a fix.

Now the only thing we will need is a Winchester Hard Disk system for the TI and I understand somebody is marketing that beast now!

FOR SALE

Eight 3'rd party games in original cassettes with documentation. eg. Strike Force 99, Super Frogger, etc. Call Paul at 476-0669 weekdays after 3:30.

LITTLE GEMS

This one from the Cental Iowa 99/4A User's Group. A new TI magazine called SUPER 99 MONTHLY will contain 12 pages published monthly starting this September at a cost of 12 bucks (US?). For further information or a subscription, write to:

BYTE[illegible] COMPUTER SERVICES
171 [illegible] STREET
SULPHUR, LA
70663

If one of our members does spring for the loot, how about a review for the rest of us?

NOTES FROM THE EXECUTIVE MEETING

Membership cards to be mailed out when printed. ----- Look for a Member of the Month profile soon in news letter ----- Watch for a Job Sheet where volunteers can fight APATHY ----- Official letterhead being ordered for business letters.

NEXT MEETING TUESDAY OCT 9'th AT 7:00

PROGRAM FEATURE

This month I am pleased to pass along a program written by one of our younger members. It is an interesting game featuring good graphics and requiring skill to master. I have tested the program and found it to be bug free as listed. Be very carefull when typing in the data statements. The attached listing is in 28 column format to make your on screen editing easier; what you see on the sheet is what you should see on the screen. It's also in larger print to make thinks as simple as possible!

Many thanks Paul for sharing this with us. Now for Paul's discription followed by the listing.

INSTRUCTIONS FOR "THRUSTER"

On the following pages you will find a print out for a game called THRUSTER. The game has been tested and there are no errors; if it does not work, check your typing. (Also, make sure ALPHA LOCK is **UP** -- ed). You will need the following to run this game:

-TI/994(A) computer
-TV set or monitor
-Joysticks (uses #1)
-TI Extended Basic Module
-Cassette or Disk storage device.

Just before you begin entering the program, enter "NUM" and the machine will auto-number your program lines for you. When you have finished typing the program, save it to cassette or disk before you "RUN" it. (It would be wise to save this one after about every 5 lines! The lines are so jammed full of coding that I would hate to retype just one of them if my system crashed during entry. -- ed).

After entering and saving, type RUN to play. There are two screens in this game and the object is to pick up all the diamonds on each screen without crashing into walls, floors, etc. Since you are limited to the number of men you have, you must be carefull on the Joystick. UP for thrusting, LEFT and RIGHT for steering. The fire button is not required and the DOWN position is ignored. It will take you many tries to complete both screens. When you do succeed in getting through, try modifying the program to increase the challenge by adding:

-more screens
-more objects to pickup
-moving objects to pick up
-music or improved graphics

GOOD LUCK AND ENJOY!
Paul Stahlke

```
100 REM THRUSTER
110 REM
120 REM PAUL STAHLKE
130 REM 5932-148 AVE
140 REM EDMONTON, ALBERTA
150 REM CANADA  T5A-1T9
160 REM (403) 476-0669
170 REM
180 REM EDMONTON 99'ER
190 REM COMPUTER USER'S
200 REM SOCIETY
210 REM
220 CALL CLEAR :: CALL SCREE
N(5):: FOR A=0 TO 14 :: CALL
 COLOR(A,16,1):: NEXT A
:: RANDOMIZE
230 DISPLAY AT(11,10):"THRUS
TER": :TAB(3);"WRITTEN BY PA
UL STAHLKE" :: FOR A=1 T
O 400 :: NEXT A
240 SC=0 :: D=27 :: P=1 :: M
=6 :: CALL SCREEN(2):: DISPL
AY AT(1,1)ERASE ALL:"HIG
H";HS;TAB(15);"SCORE";SC
250 CALL CHAR(96,"18181C3C1C
1C0818181838 3C3838101818183C
7E3C3C183C")
260 CALL CHAR(59,"0000001818
000000",61,RPT$("0",16)):: C
ALL COLOR(2,5,5)
270 IF P=1 THEN RESTORE 360
ELSE RESTORE 410
280 CALL SOUND(2400,110,30,1
10,30,900,30,-8,0):: FOR A=1
 TO 20 :: READ A$ :: DIS
PLAY AT(A+2,1):A$ :: NEXT A
290 CALL HCHAR(2,2,46,30)::
CALL HCHAR(23,2,46,30):: CAL
L VCHAR(3,2,46,20):: CAL
L VCHAR(3,31,46,20):: CALL H
CHAR(24,2,97,M)
300 S=1 :: T=0 :: CALL SPRIT
E(#1,98,14,17,121,S,T)
310 CALL JOYST(1,X,Y):: CALL
 POSITION(#1,R,C):: CALL GCH
AR(INT(R+7)/8,INT(C+7)/8
,H):: IF H=59 THEN 460 ELSE
IF H=46 THEN 480
320 IF (X=0 AND Y=0 AND S<10
)OR(X=0 AND Y=-4 AND S<10)TH
EN S=S+.5 :: CALL MOTION
(#1,S,T):: GOTO 310 ELSE IF
S>10 THEN S=S-.5 :: GOTO 310
325 REM ****************
326 REM DO NOT ENTER LINES
327 REM 325 TO 328;  FOR
328 REM PAGE APPEARANCES
```

```
330 IF Y=4 AND S>-10 THEN CA
LL PATTERN(#1,98):: CALL SOU
ND(-300,-7,10):: S=S-.5
:: CALL MOTION(#1,S,T):: GOT
O 310
340 IF X=-4 AND T>-10 THEN C
ALL PATTERN(#1,96):: CALL SO
UND(-300,-7,10):: T=T-.5
 :: CALL MOTION(#1,S,T):: GO
TO 310
350 IF X=4 AND T<10 THEN CAL
L PATTERN(#1,97):: CALL SOUN
D(-300,-7,10):: T=T+.5 :
: CALL MOTION(#1,S,T):: GOTO
 310 ELSE 310
360 DATA =;.;======;.===.;==
====;.=;=,==.========.===.==
======.===,==.==.......=
==.==...===.===,==.==.=;.;=.
===.==.=.=======
370 DATA ==.==.==.==.===.;=.
=.======;,==.==.==.==.===...
.=.==.....,==========.=
==.===;.==.;==;,==========.
===.====.==.====
380 DATA =======.==========
=.==...==,.....==.=========
==.=======,;=.=======...
.....==.=======,==.;=======.
=;=.;=======..==
390 DATA ==...==.===.===.===
=====.===,==.====.===.===...
....==.===,==.;===.=====
==.;=======.===,==.====.====
==..=======;.===
400 DATA ==...==.==.=======.
........==,==.====.==.=======
.=====.===,=======.==.==
=.===.;========,;=====;.=;.=
;=.=;=.========;,0
410 DATA ;=========;.===.;.=
===;=====,==========.===.=.
==========,==.......==.=
==.=.=........=,==.=====.==.
===.=.=.,=======
420 DATA ==.==.=;.==.===.=.;
.==...=.=,==.==....==.===.=.
..==.;.=.=,==.========.=
==.======.=.=.=,==.========.
===.;=====.=.=.=
430 DATA ==.........===....
.======.=,================
==========,=..=.........
.........=....=,========;.;=
==;.;===;.======
440 DATA =.=.....=.=.===.=.=
==.;=====,=.=.===.=.=.===.=.
=.=.....=,=.=.=.=.=.=.=.=.=
...=.=.=.;==;.=,=.=.=.=.=.=.
=.;==.=.=.==.=.=
```

COL 11

```
450 DATA =.=.;.=.=.=.=.==;.=
.=.==.=.=,=.=...=.=.=.=.==..
=.;.==.=.=,=.=====.=.=.=
===.==...==.===,;.====;.==;.
===;.;=====;.===,0
460 S,T=0 :: CALL MOTION(#1,
0,0):: FOR A=0 TO 20 STEP 5
:: CALL SOUND(-100,110,A
,-8,10):: NEXT A :: SC=SC+10
0 :: DISPLAY AT(1,15):"SCORE
";SC
470 D=D-1 :: IF D=0 THEN 540
 ELSE CALL HCHAR(INT(R+7)/8,
INT(C+7)/8,61):: GOTO 31
0
480 CALL DELSPRITE(#1):: FOR
 A=15 TO 0 STEP -3 :: CALL S
OUND(-99,-1,A,110,A):: N
EXT A
490 FOR A=0 TO 15 STEP 3 ::
CALL SOUND(-99,-1,A,110,A)::
 NEXT A
500 M=M-1 :: IF M=-1 THEN 51
0 ELSE IF M>-1 THEN CALL HCH
AR(24,2,97,M):: CALL HCH
AR(24,M+2,61):: GOTO 300
510 CALL CLEAR :: CALL SCREE
N(5):: FOR A=0 TO 14 :: CALL
 COLOR(A,16,1):: NEXT A
520 DISPLAY AT(12,10):"GAME
OVER": :"YOU HAVE FINISHED W
ITH A    SCORE OF";SC: :
TAB(6);"PLAY AGAIN? (Y/N)" :
: IF SC>HS THEN HS=SC
530 CALL KEY(3,K,S):: IF K=7
8 THEN CALL CLEAR :: END ELS
E IF K=89 THEN 240 ELSE
530
540 CALL HCHAR(INT(R+7)/8,IN
T(C+7)/8,61):: IF P=1 THEN P
=2 ELSE P=1
550 CALL DELSPRITE(#1):: FOR
 A=1 TO 200 :: NEXT A :: CAL
L SOUND(400,262,0):: CAL
L SOUND(200,196,0)
560 CALL SOUND(200,196,0)::
CALL SOUND(400,208,0):: CALL
 SOUND(200,196,0):: FOR
A=1 TO 150 :: NEXT A :: CALL
 SOUND(200,247,0)
570 D=27 :: FOR A=1 TO 90 ::
 NEXT A :: CALL SOUND(250,26
2,0):: FOR A=1 TO 200 ::
 NEXT A :: GOTO 270
```

COL 12