VOLUME 1 ISSUE 1 SEPTEMBER 1984

SUPER 99 MONTHLY

Welcome! We've enjoyed reading all the questions and comments you've sent in. The mail has been both very interesting and helpful.

We'll be trying to cover as wide a variety of topics as possible. Our articles will often build on a previous topic, introducing the basic concept first and then following up with more complex aspects. At other times we'll jump directly into the advanced material. Our hope is to thereby broaden the base of readers who can enjoy the technical information.

Thanks to you all for making this publication possible!

Immediately following the title of each article will be a "STANDARD" line. This line will relate to the "STANDARD KEY" found on page 12. Use this reference to determine the hardware and cartridges used to test the article. Note that brands are also referenced. As it is impractical to detail the compatability of all brands with each and every article, please be aware that some brands or even versions of a brand may not be compatible. The abbreviation "opt" means the item is optional.

Super 99 Disk Loader

STANDARD: 1A 2A 4B 5A 6A 7A 9A

Super 99 Disk Loader (listing on next page) will automatically run upon entering Extended BASIC, print to the screen a list of programs available on a disk, and either run a program on the disk or enter immediate mode.

The program allows you to select from up to 97 programs on any one of the 3 possible disk drives. Also, the program allows some other options such as disabling the QUIT key (reference the TI Forth Manual) and pre-selecting screen and character colors (assuming the program to be run does not change the pre-selections). This color control is especially useful for application programs used by more than one person if each user has a preference for a different color combination.

Super 99 Disk Loader uses 9 disk sectors as printed. There are no associated files to increase the number of sectors used. The program can easily be tailored to your own needs as resequencing or adding lines or statements are supported. Deleting lines or statements judiciously is also possible. Scrolling has been avoided to reduce eye-strain.

Only items listed as programs can be selected (the program also catalogs files). Selecting item 99 (END) will place the console in immediate mode. Don't forget to type "NEW" before starting to key a new program.

You must save the program using the name "LOAD", such as "DSK1.LOAD", to have it run upon entering Extended BASIC.

->

```
>1 ! SUPER 99 DISK LOADER
>2 ! COPYRIGHT 1984 SUPER 99
 MONTHLY
>100 ! *      SET-UP        *
>110 DIM A$(97),H(98),J(97),K
 (97)
>120 CALL INIT
>130 FOR I=1 TO 5 :: READ TYP
 E$(I):: NEXT I
>140 DATA "DIS/FIX","DIS/VAR"
 ,"INT/FIX","INT/VAR","PROGRA
 M"
>150 ! *       BODY         *
>160 GOSUB 2000
>170 ACCEPT AT(3,26)BEEP VALI
 DATE("01")SIZE(-1):Y
>180 ACCEPT AT(5,26)BEEP VALI
 DATE("123")SIZE(-1):X$
>190 ACCEPT AT(7,26)BEEP VALI
 DATE(DIGIT)SIZE(-2):SC
>200 ACCEPT AT(9,26)BEEP VALI
 DATE(DIGIT)SIZE(-2):FG
>210 ACCEPT AT(11,26)BEEP VAL
 IDATE(DIGIT)SIZE(-2):BG
>220 Y=Y*16 :: CALL LOAD(-318
 06,Y):: CALL SCREEN(SC):: FO
 R I=0 TO 14 :: CALL COLOR(I,
 FG,BG):: NEXT I
>230 OPEN #1:"DSK"&X$&".",INP
 UT ,RELATIVE,INTERNAL
>240 INPUT #1:A1$,U,U,V
>250 GOSUB 3000
>260 Q=Q+1
>270 INPUT #1:A$(Q),H(Q),J(Q)
 ,K(Q)
>280 IF LEN(A$(Q))=0 OR Q=97
 THEN 290 ELSE 260
>290 S=S+1 :: R=R+1
>300 IF ABS(H(S))=5 THEN B$="
    " ELSE B$="  "&STR$(K(S))
>310 T$=TYPE$(ABS(H(S)))&SEG$
 (B$,LEN(B$)-2,3)
>320 GOSUB 4000
>330 IF R=17 OR S=Q OR H(S+1)
 =0 THEN 340 ELSE 370
>340 R=0 :: GOSUB 5000
>350 ACCEPT AT(24,12)BEEP VAL
 IDATE(DIGIT," ")SIZE(-2):PRG
>360 IF PRG=98 THEN 370 ELSE
 IF PRG=99 THEN 999 ELSE PROG
 $=A$(PRG)
>370 IF PROG$<>"" THEN 380 EL
 SE IF S<Q AND H(S+1)<>0 THEN
 290 ELSE 350
>380 ! *       END          *
>390 CLOSE #1
>400 CALL PEEK(-31954,A,B)::
 GOSUB 1000 :: RUN "DSKX.1234
 567890"
>999 END
>1000 ! * RE-WRITE RUN LINE *
>1010 Z=A*256+B-65536 :: CALL
  PEEK(Z,A,B):: Z=A*256+B-655
 36
>1020 CALL LOAD(Z+29,LEN(PROG
 $)+5):: CALL LOAD(Z+33,ASC(X
 $))
>1030 FOR I=1 TO LEN(PROG$)::
  P=ASC(SEG$(PROG$,I,1)):: CA
 LL LOAD(Z+34+I,P):: NEXT I
>1040 IF LEN(PROG$)<10 THEN 1
 050 ELSE 1060
>1050 FOR I=LEN(PROG$)+1 TO 1
 0 :: CALL LOAD(Z+34+I,130)::
  NEXT I
>1060 RETURN
>2000 ! * PARAMETERS SCREEN *
>2010 DISPLAY AT(2,1)ERASE AL
 L:"QUIT KEY":"0 ENABLE 1 DIS
 ABLE       0"
>2020 DISPLAY AT(5,1):"SELECT
  MASTER DISK (1-3) 1": :"SCR
 EEN COLOR (1-16)        8"
>2030 DISPLAY AT(9,1):"FOREGR
 OUND COLOR (1-16)  2": :"BAC
 KGROUND COLOR (1-16)  1"
>2040 RETURN
>3000 ! *  CATALOG HEADER   *
>3010 DISPLAY AT(1,1)ERASE AL
 L:"DSK";X$;" - DISKNAME=";A1
 $:"AVAILABLE= ";V;" USED= ";
 U-V
>3020 DISPLAY AT(3,4):"FILENA
 ME  SIZE   TYPE":"   -------
 -  ----   ----"
>3030 RETURN
>4000 ! *    DISK CATALOG   *
>4010 DISPLAY AT(R+4,1):USING
  "##":S :: DISPLAY AT(R+4,4)
 :A$(S):: DISPLAY AT(R+4,14):
 J(S):: DISPLAY AT(R+4,19):T$
>4020 RETURN
>5000 ! * SPECIAL SELECTIONS*
>5010 DISPLAY AT(22,1):"98 NE
 XT SCREEN":"99 END":"SELECTI
 ON: 99"
>5020 RETURN
```

<u>Program Structuring in BASIC or Extended BASIC</u>

STANDARD: 1A 9A 2A (opt)

Quality programs usually require careful planning, sometimes known as

->

program development. There are many aspects of program development and in the next few issues we'll try to cover some of the more useful techniques.

We recommend beginning with flowcharting, which can be thought of as drawing a road map showing the connections of the sections of your program. Flowcharting will help you to think in a logical manner, which is necessary for a good understanding of computers. As a discussion of flowcharting would require more space than many of you would like devoted to the topic, you can find excellent books on the subject at most major book stores.

Next, learn to structure your programs. The bottome line to structuring is that a printed listing should have large segments that could be circled and labeled easily, thereby enabling you or another programmer to refer to the program later without problems in following the execution sequence of the program lines.

Here are the five sections of a structured BASIC or Extended BASIC program (in their proper sequence):

1. Set-up or initialization (May include DIM, OPTION BASE, CALL INIT, OPEN, etc.)
   This section is used to prepare the computer and/or peripherals for the remainder of the program.
2. Body (May include GOSUB, INPUT, ACCEPT, IF-THEN-ELSE, etc.)
   This section should be a brief outline of the sequence of the program, using statements such as GOSUB or ON GOTO to reference the detailed sections of the program. When using a menu, there may be a need for "sub-bodies", with each menu item requiring its own body.
3. End (May include END, CLOSE, etc.)
   This is simply the logical ending point of the program. Also, if the program runs another program, the RUN statement should be a part of the End section.
4. Subroutines (May include PRINT, DISPLAY, RETURN, etc.)
   Subroutines are used for detailed program steps and for repeated steps. Note that subroutines should follow the body of the program.
5. True subroutines or subprograms (May include SUB, SUBEND, etc.)
   Subprograms are user-defined CALL's and are available only in Extended BASIC. Subprograms are sometimes very useful and we hope to give a more detailed explanation of their usage in a future issue.

REM statements should precede each section or sub-section of your program to identify the purpose of that section. REM's can also be used for a variable directory to state the use of each variable used in the program.

Note that no reference has been made to GOTO statements as they are very seldom necessary and are usually considered to be a sign of a poorly structured program. Good structuring and wise use of IF-THEN-ELSE, ON GOTO, ON GOSUB, FOR-NEXT, GOSUB, etc. will usually preclude the need for GOTO.

Another common mistake is to exit a FOR-NEXT loop before the loop has finished incrementing. This is not a logical maneuver and there are situations in which some computers cannot handle such an action. To avoid this, use an IF-THEN statement to create a loop instead of FOR-NEXT.

Extended BASIC allows multiple statement lines. Good program structuring requires that a line should include only related statements.

Of course, we do not have unlimited memory and disk space, so we must sometimes write programs that are not perfectly structured. However, "unstructuring" should never be considered a substitute for clever protection coding.

A printer is an invaluable tool in writing a well-structured program. Without a printout, it is very

->

In Extended BASIC, the procedure is essentially the same except that you may want to make use of CALL CHARPAT and DISPLAY AT.

Don't blink when you run this program or you'll miss the screen changes!

An Introduction to TI FORTH

STANDARD: 1A 2C 4B 5A 6A 7A 9A

TI FORTH is a language that is unbelievably powerful! Your first question is probably whether FORTH is easy to learn. Our best answer is that there is a lot to learn to be a versatile programmer. In that regard, it is more difficult to learn than BASIC. However, some options are available in FORTH that are not readily available in other languages. Also, on the whole, FORTH is easier to use than Assembly Language.

FORTH is truly a language. It is based on a dictionary, into which you can easily add words. Words are defined until one word becomes the desired application. Words can reside in memory or be stored to disk on "screens". Screens can be used to define words or execute words. Words can also be executed, and thereby debugged, in immediate mode.

To use FORTH, you will need the following:

1. The items listed on the STANDARD LINE.
2. The TI FORTH System Diskette. <u>Do not use this diskette</u>! Make a backup using Disk Manager or similar software. It is easy to goof when first using FORTH and that might disable all or a part of the System Diskette.
3. The TI FORTH Manual, which is like most TI manuals in that it is a reference manual and uses few examples. However, Appendix C is a good cross-reference to a book of FORTH examples (see next item).
4. The book <u>Starting FORTH</u> by Leo Brodie.
5. At least one application diskette. To initialize this diskette for FORTH, follow these procedures:

A. Insert the Editor/Assembler cartridge.
B. Insert the FORTH System Diskette in drive 1.
C. Load and Run DSK1.FORTH
D. Type the following, pressing enter and waiting for the FORTH prompt "ok" after each of these lines (These steps ready the error screens, screens 4 and 5, for copying from the System Diskette):

```
-EDITOR
-SYNONYMS
EMPTY-BUFFERS
FLUSH
4 BLOCK DROP UPDATE
5 BLOCK DROP UPDATE
```

E. <u>Remove the System Diskette. Insert the new diskette</u>.
F. Proceed as in D. above:

```
0 FORMAT-DISK
FLUSH
4 EDIT
```

G. If you see, among other information, the following, then you have probably proceeded properly:

```
0 ( ERROR MESSAGES )
```

H. Press function 9 and proceed:

```
5 EDIT
```

I. Is screen 5 there? If not, start over at step A.

You are now ready to use FORTH. The article that follows this one does not store the words defined to disk. But, we wanted to be sure you knew how to set up your disk in case you get ahead of our pace and want to go ahead and store your words to disk. Next month we'll get into storing words and programs to disk.

Bit-mapped Drawing In FORTH

STANDARD: 1A 2C 4B 5A 6A 7A 9A

FORTH is much faster and often more versatile than Extended BASIC. Though usually not as fast as Assembly Language, it is usually easier to implement. For months (years?),

->

TI-99/4A users have heard a lot of talk about Assembly Language bit-map mode. Despite all this talk, few users have ever implemented bit-map mode with any reasonable degree of success. This article will show how easy this previously difficult application is in FORTH.

The procedure below will show you how to enter bit-map mode, draw (computer-assisted design anyone?), define words, define words from previously defined words, forget back to a previous word, etc. Refer to the TI FORTH Manual, especially chapter 6, for further information. We'll go without further explanation for now as the following was designed to allow you to explore (remember to press <enter> after each line and wait for "ok"):

```
-TEXT -GRAPH -SPLIT -SYNONYMS
:SUPER  CLS SPLIT ;
SUPER
: TLINE  8 0 248 0 LINE ;
: RLINE  248 1 248 100 LINE ;
: BLINE  247 100 8 100 LINE ;
: LLINE  8 99 8 1 LINE ,
: BOX  TLINE RLINE BLINE LLINE ;
BOX
: DIAGONAL  9 1 247 99 LINE ;
: PIXEL  34 34 DOT ;
DIAGONAL
PIXEL
DRAW 2
DTOG DIAGONAL PIXEL BOX
DIAGONAL BOX DIAGONAL BOX
SUPER DRAW 0
BOX DIAGONAL
FORGET SUPER
SUPER
BOX
TEXT
COLD
```

See if you can now create your own words and draw on the screen. Next month we'll cover storing to diskette and see if we can come up with a few FORTH surprises for you.

Support your local users group!

Horse Evaluator Using Multiplan

STANDARD: 1A 2D 4B 5A 6A 7A 9A

Computers have made their mark in the horse racing industry. This article proposes an improvement to data currently in use and is not presented as a wagering system.

The problem being approached is the reliability of the speed index. Many racing fans and horsemen assume that the speed index represents a percentage of the track record time. Actually, the speed index is an approximation of the number of lengths better or worse than the track record. This means that the speed index of one distance cannot be properly compared to the speed index of another distance. The spreadsheet we will detail will convert available figures to more useful ones and derive a speed figure that is a percentage of the track record.

Data commonly available on past performances includes the winning time, beaten lengths at the finish, and the speed index. What we want to derive is the time for the particular horse, the record time that the speed index was derived from, and an adjusted speed figure. These figures are usually not shown in racing papers, but can be calculated using figures that are given.

The spreadsheet will have input cells for the minutes, seconds and fifths of the winning time, the beaten lengths at the finish, and the speed index. Then there will be a dotted line which will be followed by the spreadsheet's calculations for the time of the particular horse, the record time of the distance and track at which the race was run, and the adjusted speed, which is the record time divided by the time. The spreadsheet assumes one speed index point equals 1/5 seconds or 1 length.

On the next page are complete instructions for building the spreadsheet, which can be expanded using the examples shown in the

->

Multiplan manual and/or future articles in this publication:

```
FORMAT   WIDTH 15 1 THRU 1     <ENTER>
GOTO     R5C1                  <ENTER>
ALPHA    MINUTES
CURSOR   DOWN TO R6C1
ALPHA    SECONDS
CURSOR   DOWN TO R7C1
ALPHA    FIFTHS
CURSOR   DOWN TO R9C1
ALPHA    BEATEN LENGTHS-
CURSOR   DOWN TO R10C1
ALPHA       (FINISH)
CURSOR   DOWN TO R12C1
ALPHA    SPEED INDEX           <ENTER>
CURSOR   DOWN TO R14C1
ALPHA    ---------------
CURSOR   DOWN TO R16C1
ALPHA    TIME
CURSOR   DOWN TO R17C1
ALPHA    TRACK RECORD
CURSOR   DOWN TO R18C1
ALPHA    S99 SPEED             <ENTER>
NAME     MINUTES  R5C2         <ENTER>
NAME     SECONDS  R6C2         <ENTER>
NAME     FIFTHS   R7C2         <ENTER>
NAME     BEATEN   R9C2         <ENTER>
NAME     INDEX    R12C2        <ENTER>
NAME     TIME     R16C2        <ENTER>
NAME     RECORD   R17C2        <ENTER>
GOTO     R16C2                 <ENTER>
VALUE    MINUTES*60+SECONDS+FIFTHS*.2+
         BEATEN*.2             <ENTER>
CURSOR   DOWN TO R17C2
VALUE    TIME-BEATEN*.2-20+(.2*(INDEX+
         INT(BEATEN+.5)))      <ENTER>
CURSOR   DOWN TO R18C2
VALUE    RECORD*100/TIME
WINDOW   SPLIT VERTICAL 2 YES <ENTER>
WINDOW   BORDER 1              <ENTER>
WINDOW   BORDER 2              <ENTER>
OPTION   NO                    <ENTER>
```

We selected the "no" option to avoid calculating the results of formulas after each input, thereby speeding up the input process.

```
Now input the time as 1:23².
Input beaten lengths as 16.
Input speed index as 42         <ENTER>
OPTION   YES                    <ENTER>
The spreadsheet should calculate:
TIME=86.6        TRACK RECORD=75
S99 SPEED= 86.6051
```

Using ERASE ALL in Extended BASIC

STANDARD: 1A 2A 9A

In beginning with Extended BASIC, one is likely to wonder why the DISPLAY statement has an ERASE ALL option when CALL CLEAR is available. The answer to the puzzling question lies in the functioning of the BASIC interpreter.

Your program is not stored in memory in quite the same way as it appears on the screen. The BASIC interpreter converts your words into characters (called "tokens") the computer can work with. So, although most beginners would think that ERASE ALL uses 9 bytes and CALL CLEAR 10 bytes, such is not the case. ERASE ALL uses only 2 bytes while CALL CLEAR chews up 8 bytes! Here is how the statements are stored in memory:

157	CALL
200	Next byte is no. of characters in screen word
5	Five characters in CLEAR
67	C
76	L
69	E
65	A
82	R

239	ERASE
236	ALL

Since most CALL CLEAR statements precede DISPLAY (PRINT can be changed to DISPLAY in this context), you'll probably soon be switching to the big byte saver - ERASE ALL.

DISPLAY AT in Console BASIC

STANDARD: 1A 9A

If you are about to "chunk" your computer because you hate to see the screen scroll up and you aren't using Extended BASIC, power up and forget your scrolling blues.

On the next page is a program that shows how easy it is to implement

->

what is known in Extended BASIC as DISPLAY AT. This prints characters to form words at any position on the screen that you choose. Granted, this is not nearly as fast as Extended BASIC, but it is less expensive and this routine will also work with the Terminal Emulator II where you would have full speech control available.

Here is the example program:

```
>100 CALL CLEAR
>110 FOR A=1 TO 5
>120 GOSUB 2000
>130 NEXT A
>999 END
>2000 READ WORD$,X,Y
>2010 FOR D=1 TO LEN(WORD$)
>2020 CALL VCHAR(X,Y-1+D,ASC(
 SEG$(WORD$,D,1)))
>2030 NEXT D
>2040 RETURN
>5000 DATA "DISPLAY AT TEST",
 1,3
>5010 DATA "12345678901234567
 890123456789012",3,1
>5020 DATA "THIS SHOULD APPEA
 R ON LINE 5",5,3
>5030 DATA "*",12,16
>5040 DATA "CENTER OF SCREEN"
 ,13,8
```

If you do not intend to use the graphics columns (1,2,31,32), you can change line 2020 to read "Y+1" instead of "Y-1" (Columns still begin at 1).

Using the TI-Writer Transliterate Commmand

STANDARD: 1A 2E 3A 4B 5A 6A 7A 9A 10A

The real purpose of using TI-Writer is to generate a written document using a printer. Therefore, it is very important to make full use of your printer.

TI-Writer is a very powerful tool once you've learned how to use it. One of the more complicated commands is the transliterate command, which allows you to send special instruction characters to your printer to perform printing tasks not otherwise possible. Not only does transliterate enable printer instructions, but it also provides a way around the built-in codes that TI-Writer uses. For example, the ampersand (ASCII 38) is used by TI-Writer to underscore. If you want to print "&", you must use transliterate. This also applies to the circumflex (caret) and the at sign. Occassional offenders include the asterisk and period. You may notice others in some situations.

Transliterate is just a fancy way of saying "I need a substitute character". The ampersand shown above can be printed by the following transliterate command, which uses ASCII 37 (the percent sign) for ASCII 38 (&) which is used by TI-Writer:

```
.TL 37:38
```

Now, each time "%" is encountered, the print-out will show "&".

To turn on a printer command, the procedure is similar. Most printer commands begin with the "escape code", which is ASCII 27. The numbers that follow the escape code instruct the printer to activate a particular capability. Although most printers use similar instruction codes, there is no industry standard. Because of these differences, do not assume that someone else's printer will work the same as yours. Study your printer manual carefully. Here is the transliterate command to select double-strike printing on a Gemini-15X PC, using the question mark as the substitute character:

```
.TL 63:27,71
```

In transliterating, be careful to not use a character already in use and be sure to turn printer commands off after their use. Underscore, superscript, etc. are normally not only turned on, but also off.

This entire publication was created using TI-Writer! We'll tell you more about how this was accomplished next month.

->

difficult to follow the structure of an entire program once it gets long.

This article shows one of many ways to structure a program. The most important factors are that the program be readable and functional.

So You Say You Can't Draw Graphics...

STANDARD: 1A 9A 2A (opt)

Many people would like to write educational or recreational programs, but get discouraged by their inability to draw graphics. There is a simple solution, as close as your nearest yarn shop. There you are likely to find many books with pictures drawn using dots, many being color-coded. These dots can be thought of as screen pixels. Simply use the Character Definition program in your BASIC Reference Manual or one of the many sprite (Extended BASIC only) editor programs available to convert the dots to pixels and obtain the Hex codes required by the CALL CHAR statement and you'll have the beautiful graphics to go with your beautiful text!

Cursor Pizzazz

STANDARD: 1A 2A 9A

If you are bored with the standard black on transparent cursor, Extended BASIC will allow you to use CALL COLOR on character set zero. Blue seems to be a pleasant alternative. Note that edge characters will also be affected.

Instant Screen Formatting

STANDARD: 1A 9A 2A (opt)

Innovative use of characters by manipulations of the character pattern table can produce some interesting effects.

For instance, CALL CLEAR places ASCII character 32 into every character position on the screen (768 positions per screen). If you redefine character 32, the new pattern will appear with each CALL CLEAR. Note that since ASCII 32 is the blank, if you will require blanks in subsequent program lines, you need to have a substitute blank until you restore the original character patterns.

Once the screen is filled with characters, the CALL CLEAR is no longer required. Simply using CALL CHAR will change the pattern of any character instantly.

Here is an example program in console BASIC:

```
>100 REM ** SET-UP **********
>110 A$="0000000000000000"
>120 B$="FF818181818181FF"
>130 C$="0038440408100010"
>140 CALL CHAR(63,A$)
>150 CALL CHAR(32,B$)
>160 REM ** BODY ************
>170 GOSUB 1000
>180 GOSUB 2000
>190 GOSUB 3000
>200 GOSUB 2000
>210 GOSUB 4000
>220 GOSUB 2000
>990 REM ** END *************
>999 END
>1000 REM * DRAW GRID *******
>1010 CALL CLEAR
>1020 PRINT "THIS?IS?A?GRID"
>1030 RETURN
>2000 REM * DELAY LOOP ******
>2010 FOR A=1 TO 1500
>2020 NEXT A
>2030 RETURN
>3000 REM * CLEAR SCREEN ****
>3010 CALL CHAR(32,A$)
>3020 PRINT "OOPS!":"THIS WAS
  A GRID"
>3030 RETURN
>4000 REM * SHOW SLOW HCHAR **
>4010 CALL CLEAR
>4020 CALL CHAR(63,C$)
>4030 CALL HCHAR(1,1,63,768)
>4040 RETURN
```

Note how slowly the CALL HCHAR statement works in comparison to the more direct CALL CHAR or CALL CLEAR.

->

Playing Tombstone City

STANDARD: 1A 2C 4B 5A 6A 7A 9A
Also available on cartridge

The TI game Tombstone City: 21st Century is an arcade-style game that is a lot of fun. Here are a few tips on how to increase your score:

1. Remember the obvious. Avoiding the loss of schooners and shooting tumbleweeds are keys to success but are easily forgotten with all those morgs attacking.

2. Keep the passageways to the safe area open. If a morg is on one side of the cemetary, attempt to exit on the other side rather than just shooting the morg.

3. Try using the keyboard instead of joysticks. Many game players find the keyboard to be much faster.

4. Try to set up the attacking morgs so that they can be shot while next to pairs of saguaros. This will limit the number of generating pairs of saguaros. This is especially useful in keeping the passageways to the cemetary open.

5. Watch the saguaros with white backgrounds. A morg is about to come from there! To avoid being suddenly caught by a newly generated morg, keep your schooner several "squares" away from all saguaros when possible.

6. Use the saguaros to block the morgs when to your advantage. However, be careful. This practice can result in more morgs than can possibly be eliminated without losing one or more schooners.

7. Fire at the morgs even before they are generated. This will limit the number of saguaros.

8. When you feel the game is hopelessly near ending, get points as quickly as possible. This will teach you to maintain a high average in case you ever compete with another player.

This article was based on the version that comes with the Editor/Assembler. The game is also available in cartridge form.

If you have tips on playing games or advice on which games are the best, please write to us. The time we have for playing games is very limited and we're sure that a lot of you are better game players than we are!

Beginning BASIC: Using RESEQUENCE

STANDARD: 1A 9A 2A (opt)

Many times you may feel that if you add four lines here and eight more there then your program will do just what you wanted it to do. But what happens if the place to insert the eight lines has only four line numbers available? Time to RESEQUENCE!

RESEQUENCE will rearrange your program line numbers so that all line numbers are equally incremented. The command is obviously only available in immediate mode, not while the program is running. Here is an example that will set the first line number equal to 100 and all subsequent line numbers 10 apart:

```
RESEQUENCE 100,10
```

RESEQUENCE can be shortened to RES. Also, since the values 100 and 10 shown above are the default values, the example could have used just RES.

RESEQUENCE will automatically change all references to line numbers. For instance, if you had GOSUB 1000 at line 100 and line 1000 is changed by the RESEQUENCE to line 800, line 100 will automatically be changed to GOSUB 800. References to non-existent line numbers will be assigned as 32767 with no error generated before the program is run.

->

99 POTPOURRI

News, Corrections, Updates, Editorials, Kudos, and Come-what-may

Texas Instruments, despite having withdrawn from the home computer field, is continuing to offer some support for the TI-99/4A. Much of the support is being channeled through local users groups.

Several months ago, TI released TI FORTH to the users groups. Though TI FORTH was not considered by TI to be a fully developed product, it was an essentially completed project and is now being enjoyed by many users.

In July, TI gave us all another surprise (another pleasant one for a change). Fixes for TI-Writer and Microsoft Multiplan were released to the users groups.

The fix to TI-Writer includes true lower case screen characters (with true descenders) and it eliminates some of the paper advances, most notably the unnecessary one that occurred prior to printing. Also, the fix offers defaults for serial printers ("RS232.BA=1200.LF" or "RS232.BA=4800.LF"). Our only negative criticism is that the new lower-case letters do not seem to be quite as easy to read on a television. Those of you with monitors and serial printers should find all aspects of the fixes to be real improvements. For parallel users, although it is a little inconvenient to change from the default during formatting, the inconvenience is outweighed by the improved paper advances.

The Multiplan fixes are designed to speed up your spreadsheeting. One of the most notable changes is an auto-repeat cursor feature. It makes moving from one cell to another much easier and faster. Other changes are more transparent, but we have not detected any bugs.

Though not likely to make us forget "Black Friday", at least TI's continuing support is helping us to be a little more forgiving.

Possibly the most exciting news of the summer is CorComp's release of their 9900 Disk Controller Card.

The CorComp Controller Card features several new statements that are available in any form of TI BASIC! These statements offer memory movement for most memory locations, including the Video Display Processor and 32K Card (yes, even from console BASIC!). The screen displays are the fastest we've seen! Approximately 40 screens of mixed characters can be displayed in one second (not a typo)! New Assembly linking statements are also included. The new statements are sure to revolutionize programming on the TI-99/4A!

Another fantastic feature is the support of double-density disks (DD disk drive is required). With double-sided/DD (DS/DD), 360K can be stored on one disk. The TI Controller Card supports only SD (limited to 180K) and the TI Disk Drive is SS/SD (limited to 90K).

The Disk Manager is on diskette instead of the cartridge format used by the TI Disk Manager II. The user's configuration can be stored on disk along with the CorComp Disk Manager. The Disk Manager uses many single keystroke commands, enabling fast work sessions. Also, all files on a disk can be marked for copy, rename, delete, and/or write protection change in a single operation. Most of the TI utilities are supported, with most sporting enhancements.

If you purchase the 9900 Disk Controller Card, it is very important that you read the Manual first. Here are our "better do it's":

1. Install the card. If it does not work, check to see if you have a CorComp RS-232 in a TI Expansion Box. If so, remove the RS-232. An addendum to the manual says to call CorComp if the Controller still does not work. The addendum

->

does not point out that you also need to call about the CorComp RS-232 if it was disabling the Controller Card. If it does, call CorComp to obtain authorization to return the RS-232 Card, which CorComp says will be quickly fixed and returned. Only older CorComp RS-232's should give this problem. Also, do not send the RS-232 to the address shown in the RS-232 manual. It should be sent to the address shown in the 9900 Disk Controller Manual. If you require express shipping, you will have to pay for it. Additionally, if you have difficulties and need to call CorComp, call after 1:30 PM Pacific time as there are no service technicians on duty prior to that time.

2. The 9900 Controller is designed to work with almost any 300 RPM disk drive and almost any software. This is possible due to available settings. Thorough testing is advised as DD is very sensitive. Use quality diskettes, be sure the disk head is clean, check the ground and voltage, use the maximum disk format (e.g. DS/DD), and test cartridge software (e.g. TI-Writer and Multiplan) and disk software. If you still have problems, read pages 4, 5, and 54 of the manual to determine if the disk head step and the diskette interlace are appropriate for your configuration. If all tests go well for 2 days, you are ready to store files permanently. We'll have more on testing the CorComp Controller next month.

Our Dealer of the Month is Steve Manuel of Investment in Knowledge, located here in Sulphur. Steve stays up on the availability of products and stands behind what he sells. If it were not for Steve, we may have missed our deadline for this issue.

If you know of a dealer who has a retail storefront, stocks TI-99/4A products, and is an outstanding merchant, drop us a line.

Bayou 99 Users Group offers the TI FORTH Manual and the TI FORTH System Diskette for $20 for non-members, $15 for members. Do not send money. C.O.D. only. There is no charge for shipping or C.O.D. for U.S. orders. Order from:

Bayou 99 Users Group
P.O. Box 921
Lake Charles, LA 70602

This month was a "get-acquainted" issue. If you want some changes or more of the same, please write! Next month we'll have all this introductory information out of the way and will be able to devote more space to programs. We hope you Multiplan beginners tried the spreadsheet this month even if you are not into horse racing as the basic techniques used will be used in many future Multiplan models. We'll be trying to cover different areas of interest each month, so let us hear from you so we can be sure to cover your favorite topic!

->

Super 99 Monthly is published monthly by Bytemaster Computer Services, 171 Mustang Street, Sulphur, LA 70663. Subscription rate in U.S. and possessions is $12.00 per year; all other countries $16.00 U.S. funds for surface mail. All correspondence received will be considered unconditionally assigned for publication and copyright and subject to editing and comments by the editors of Super 99 Monthly. Each contribution to this issue and the issue as a whole Copyright 1984 by Bytemaster Computer Services. All rights reserved. Copying done for other than personal archival or internal reference use without the permission of Bytemaster Computer Services is prohibited. Bytemaster Computer Servies assumes no liability for errors in articles.

STANDARD KEY

1	Computer	A	TI-99/4A
2	Cartridge	A	Extended BASIC
		C	Editor/Assembler
		D	Multiplan
		E	TI-Writer
3	RS-232	A	CorComp
4	Disk Drive	B	TEAC 55B
5	Expansion Box	A	TI
6	Disk	A	TI
	Controller	A	TI
7	32K Card	A	TI
9	Monitor or TV	A	TV & RF Modulator
10	Printer	A	Gemini 15-X PC

Note: This list will be adjusted monthly to relate to current articles. The reference symbols will always correspond to the same products.

Bytemaster Computer Services
171 Mustang Street
Sulphur, LA 70663