NH99'ers User Group

New Hampshire 99'ers User Group
PO Box 5991, Manchester, NH 03108

Newsletter

July 1989
Vol. 7, No. 7

CLUB NEWS
by Paul Bendeck, President

NO MEETINGS UNTIL SEPTEMBER!

Club meetings have been suspended for the summer months, July and August, since many people are away on vacation. Meetings will resume with the September 18 meeting. This decision was arrived at by a consensus vote of the members attending the June 19 meeting. Please mark September 18 on your calendars now so you don't forget.

During the summer recess, we will continue to send out a newsletter each month to keep people informed of what is happening. For those of you on standing order for MicroPendium magazine, the club will continue ordering these at least through September. If you would like to get the June, July, or August issues, contact Vince Demers (603-424-3538) in Merrimack, NH.

Elections were scheduled for the June meeting. However, due to the light member turnout and lack of any candidates willing to run for president, it was decided to postpone elections until September. This situation raises a number of critical questions for our club. In order to continue as a club, we need members to support the club by attending meetings, giving demos, making presentations, teaching tutorials, etc. At present, there are about 10-15 members who attend meetings regularly. We need more than that to realistically sustain the club.

In september we must decide whether or not to continue the club. If enough people show up and there is sufficient interest, we will take nominations for new club officers and have an election. Otherwise, we will begin steps to dismantle the club. Club assets will be donated to charity, as per our charter.

Please think about all of this over the summer. Do you want the NH99'ers User Group to continue as a club? What are you willing to contribute to the club in order to keep it going? What do you want to get out of the club? Are you willing to run for office, help with the newsletter or software library? Bring us your ideas, comments and suggestions to the September 18 meeting.

SPEECH

By Andrei Derksen, Netherlands

This article ("SPEECH") talks about some experimenting I did with the speech synthesizer.

The files starting with "DEF..." are the merge files listed in the back of the Extended Basic manual about speech. They go with the program "SUFFIXES" (also in the back of the manual). Suffixes is the ready-to-use program, the merge-files are needed for other things you might want to do with speech. This is (more or less) explained in the article. As for "S--T", well, read the article and run the program. Even the title is self-exolanetory.

"FRONTTRUNC" lets you remove bytes of the beginning of a speech word, More or less the reverse of the program Suffixes.

Those of you who own a Speech Synthesizer will know that although it might not always be very useful, it can be fun. Lately I have been playing around with it. In this article I would like to tell you about my findings.

As you might know, there are several ways to add ress your Speech Synthesizer. As far as TI is concerned, you have Extended Basic and the Terminal Emulatator. In Extended Basic there is a built-in vocabulary. These words and phrases can be accessed by using the CALL SAY and CALL SPGET statements. If you use TE II, you can have the Speech Synthesizer pronounce every word you want it to. Even foreign languages, albeit with the restriction that you will always have to deal with the American accent built in to it.

However, if you have Extended Basic, 32K and a disk drive you don't need the TEII module to be able to have the II say anything you like it to. There are programs available on disk which accomplish the same task as the TEII module.

I knew that with the TEII module, you can change the difference in pitch between the beginning of the sentence and the end. I also knew you can make the voice sound higher or lower. So I started wondering if it would also be possible to do the same with this Extended basic program. The best way to find out of course is to look at the basic listing. Most of the program is written in assembler, so that is of no help to us. But there is one very interesting line. In my version, this is line 150. You might have a different version, but it should be more or less the same. This line reads CALL LINK("SPEAK",B$,43,128) This line hands over control to the assembler program called "SPEAK", and feeds it the string B$ (the sentence you want it to say), and the values 43 and 128. These two values are particularly interesting. The first value (43) regulates now hign the voice sounds. The second determines the difference in pitch between the beginning and the end of the sentence. When you start changing these values, interesting things happen. If you increase the first value, the voice will become lower. If you decrease it, it will become higher. Setting it to zero will not, as you might expect, lock-up the computer. Try it. Ever heard a whispering computer before? As far as the second value is concerned, decreasing it will make the speech sound more and more like one would expect from a computer. If one sets it higher than 128, the pitch difference will start to become unnatural. By the way, you can run and break the computer without any problem. However it would be a good idea to skip the first line. This should read something like CALL LOAD(DSK1.xxx) The files loaded by this statement only need to be loaded once every time you use the program. Loading them every time you run the program is a waste of time.

Even in Extended Basic it is possible to have the computer say more than just the built-in words. Take the program Burt+Ernie. Whoever made that must have spent a lot of time on the matter. It isn't completely clear to me how it was done, but this much I do know. When listing the program, after a while you come across some kind of coding. Only, this is not coding. Type in the following line into Extended Basic: CALL SPGET("#TEXAS INSTRUMENTS#",A$) :: PRINT A$

Looks familiar? Also notice the length of the string. To check that, type PRINT LEN(A$) (After having entered the CALL SPGET). The coding is in fact a long string of ascii chars which enable the Speech Synthesizer to pronounce the word. To see the actual ascii codes use this short program:

```
10 CALL SPGET("#TEXAS INSTRUMENTS#",A$) 20 A=LEN(A$)
30 FOR I=1 TO A 40 PRINT ASC(SEG$(A$,I,1)) 50 NEXT I
```

It takes many more characters for the Speech Synthesizer to pronounce the word than for us to spell it. What the programmer of Burt+Ernie did was to rearrange these chars in such a way as to make new words. The work this must have been is absolutely mind bogling. Another thing I don't understand yet, is how the programmer managed to get these speech strings into DATA statements.

One thing you can do yourself is use the program in the back of the Extended Basic manual. (If you never managed to take the time to enter it, it is included on this disk). What this program does is to truncate the end of any word in the vocabulary to the length you want it, and then add one of the provided endings. Using this program I made a small program which makes the computer pronounce a four-letter word not in the vocabulary (included on this disk). Another thing I discovered is that the first two bytes of any speech string are always 96 and 0. The third byte is always the length of the remaining string, i.e. the total length minus 3 (bytes). So, you could create your own speech strings. But don't expect anything meaningful. It will probably sound like a lot of burps and whistles. Have fun!

TIPS FROM THE TIGERCUB

#57

Tigercub Software
156 Collingwood Ave.
Columbus OH 43213

I am still offering over 120 original and unique entertainment, educational and utility programs at just $1.00 each, or on collection disks at $5.00 per disk.

The contents of the first 52 issues of this newsletter are available as ready-to-run programs on 5 Tips Disks at $10 each.

And my three Nuts & Bolts Disk, $15 each, each contain over 100 subprograms for you to merge into your own programs to do all kinds of wonderful things.

My catalog is available for $1, deductable from your first order (specify TIGERCUB catalog).

xxxxxxxxxxxxxxxxxxxxxxxxxxx
TI-PD LIBRARY

I have selected public domain programs, by category, to fill over 200 disks, as full as possible if I had enough programs of the category, with all the Basic-only programs converted to XBasic, with an E/A loader provided for assembly programs if possible, instructions added and any obvious bugs corrected, and with an autoloader by full program name on each disk. These are available as a copying service for just $1.50 postpaid in U.S. and Canada. No fairware will be offered without the author's permission. Send SASE for list or $1, refundable for 9-page catalog listing all titles and authors. Be sure to specify TI-PD catalog.
xxxxxxxxxxxxxxxxxxxxxxxxxxx

I like little programs that load quickly and do just what I want to do at the moment. And one of the things I wanted to do quickly was to find phone numbers. So, I used FUNLWEB to create a little file -

SMITH,JOHN (999) 111-2222
BUSH, GEO. (000) 123-1234
GHADDAFI, O. (666)66-6666

and all my other frequently called numbers. I SAVEd it as DSK1.PHONELIST and wrote this little routine to use it.

```
100 CALL CLEAR
110 OPEN #1:"DSK1.PHONELIST"
,INPUT
120 DISPLAY AT(12,1):"LAST N
AME?" :: ACCEPT AT(14,1):N$
130 LINPUT #1:M$ :: IF POS(M
$,N$,1)<>0 THEN DISPLAY AT(1
6,1):M$ :: RESTORE #1 :: GOT
O 120
140 IF EOF(1)<>1 THEN 130
150 DISPLAY AT(16,1):"NAME N
OT FOUND" :: RESTORE #1 :: G
OTO 120
```

Now actually, that was all I needed, (even though it did take several seconds to find a name at the end of the file), and it was easy enough to load the file into FUNLWEB when it needed updating. But, programmers are never satisfied, so I decided to write a self-contained program -

```
100 CALL CLEAR
200 DATA "ALDA, ALAN 888-999
9"
201 !@P-
300 DATA "BUSH, GEORGE 111-1
111"
400 DATA "PRESLEY, ELVIS 000
-0000"
499 !@P+
500 DISPLAY AT(12,1):"LAST N
AME?" :: ACCEPT AT(14,1):N$
600 READ M$ :: IF POS(M$,N$,
1)<>0 THEN DISPLAY AT(16,1):
M$ :: RESTORE 200 :: GOTO 50
0
700 ON ERROR 800 :: GOTO 600
800 DISPLAY AT(16,1):"NAME N
OT FOUND" :: RESTORE 200 ::
GOTO 500
```

That funny thing in line 201 turns off the prescan and speeds up initialization. This routine is no faster than the last, but can be updated by editing the program itself. It is limited to about 500 records due to the least-known and greatest weakness of the TI, that string storage is limited to console memory.

But, computer users are paranoid about speed, so I decided to put my data into a pre-loaded array with self incrementing subscript numbers, and find the data by a binary search.

```
100 !QUICKFINDER by Jim Pete
rson
200 DIM D$(50):: GOTO 300 ::
 D$(),X :: !@P-
300 X=X+1 :: D$(X)="ALDA, AL
AN (999) 666-1234"
400 X=X+1 :: D$(X)="BUSH, GE
ORGE (111) 111-1111"
500 X=X+1 :: D$(X)="GHADDAFI
, OMAR (999) 456-1234567"
600 X=X+1 :: D$(X)="KHOMEINI
, AYATOLLAH (666) 666-6666"
700 !@P+
800 INPUT "NAME? ":M$
900 IF M$>D$(X)THEN PRINT "N
OT FOUND":"CLOSEST IS":D$(X)
:: GOTO 800
1000 IF M$<D$(1)THEN PRINT "
NOT FOUND":"CLOSEST IS":D$(1
):: GOTO 800
1100 H=X :: S=INT(X/2)
1200 S$=D$(S):: IF POS(S$,M$
,1)=1 THEN 1700
1300 S$=D$(S+1):: IF POS(S$,
M$,1)=1 THEN S=S+1 :: GOTO 1
700
1400 IF S$>M$ THEN H=S :: S=
INT(H/2):: GOTO 1600
1500 S=S+INT((H-S)/2)
1600 IF S=S2 THEN 1800 ELSE
S2=S :: GOTO 1200
1700 PRINT D$(S):: GOTO 800
1800 PRINT "NOT FOUND":"CLOS
EST ARE"
1900 IF D$(S2)>M$ THEN PRINT
 D$(S2-1):D$(S2+1):: GOTO 80
0
2000 PRINT D$(S2+1):D$(S2+2)
:: GOTO 800
```

Note that in this case the records must be in alphabetical sequence. New records can be inserted in intermediate line numbers, in alphabetic sequence, always preceded by X=X+1 :: D$(X)= . Obsolete records can be deleted, and records can be corrected in place if the correction does not change the alphabetic sequence.

This idea did not work out as well as I hoped. The maximum number of records is less than 300, for the reason mentioned above, and this leaves so little free memory that even a binary search is slow. However, for a smaller file this is perhaps the best method.

For a large file, the best method is certainly a fixed sequential disk file, accessed by a binary search routine. But, that requires other routines to delete, add or change records, and had best be the subject of another Tips.

There is apparently a mistaken belief that sprites cannot be used together with my BXB routine. Not so - you can use all 28 of them! However, you cannot change their color with CALL COLOR(#,N). The only other limitations of BXB that I can think of, are that a single CALL COLOR cannot be used for multiple character sets and a single CALL CHAR can only reidentify one character. CALL CHARPAT cannot return the hex code of an ASCII above 143 because those ASCII's were not supposed to be available in Extended Basic.

I have used BXB on hundreds of Basic-only programs and have had only

two rare problems. If the program contains multiple line feed colons ::::::, the computer may rearrange them into pairs of double colons :: :: and lock up. Or, if the colons are before the text, as in PRINT :"something" you may get a puzzling error message.

Also on rare occasions you might get an error message indicating the subprogram was called from a line containing a CALL CHAR, if the programmer had inadvertently put more than 16 characters in the hex code. Basic just ignores any extra characters, and XBasic uses them to reidentify the following ASCII, but BXB crashes.

From the T*I*M*E*S newsletter from England, here is an extremely useful bit of assembly which should be assembled as ALPHA/O and placed on the disk of every joystick program, or imbedded in it with ALSAVE.

```
       DEF   ALPHA
* save old R12
ALPHA  MOV   R12,@>FFFC
* 9900 CRU base=0
       CLR   R12
* signal alphalock key line
       SBZ   21
* check alphalock other side
       TB    7
* jump if state=on
       JNE   STATE
*  state=off
       SETO  @>FFFE
* as off skip next line
       JMP   JUMPA
*  state=on
STATE  CLR   @>FFFE
* stop sending to alpha key
JUMPA  SBO   21
* restore R12
       MOV   @>FFFC,R12
* standard XB return now
* clear error for basic
       SB    @>837C,@>837C
* return to calling program
       B     @>0070
       END   ALPHA
```

Now, put this in the first lines of the joystick program -

```
1 ! by M. Gikow, Andover
    MA  August 1988
2 ! used with ALPHA/O,
    will detect whether
    Alpha Lock is up (A=
    255) or down (A=0)
3 CALL CLEAR :: CALL INIT ::
 CALL LOAD("DSK1.ALPHA/O")
4 CALL LINK("ALPHA")!! CALL
PEEK(-1,A):: IF A=0 THEN DIS
PLAY AT(12,1):"RELEASE ALPHA
LOCK" :: GOTO 4 ELSE CALL CL
EAR
```

I published this one in the C.O.N.N.I. newsletter. Barry Traver, picked it up and put it in the TI Forum in Computer Shopper, but their typesetter garbled it, so here is how it was supposed to be -

According to the TI-Writer Reference Guide, page 77, when you select the PrintF command, then type C and space once and then the device name, any control characters with ASCII less than 32 are removed before the file is printed.

With Funlweb, at least, this is not quite true. A carriage return character, ASCII 13, or a line feed character, ASCII 10, at the end of a line is actually not deleted but is changed to the space bar character, ASCII 32. This can be proved by running this little routine -

```
100 OPEN #1:"DSK1.(Filename)
",INPUT
110 LINPUT #1:M$ :: PRINT M$
:LEN(M$):: IF LEN(M$)>0 THEN
 PRINT ASC(SEG$(M$,LEN(M$),1
))
120 CALL KEY(0,K,S):: IF S=0
 THEN 120 ELSE 110
```

Therefore, when a file is Filled/Adjusted and the line feed characters are stripped with the C option, the lines are one character longer than they appear to be. An apparently blank line also contains ASCII 32.

Since these characters are blank, they normally do no harm. However, they can create problems when records are read into programs for multiple column printing or concatenation of strings. In these cases, this routine can be used to strip out any ASCII below 33 at the ends of records.

```
100 DATA INPUT,OUTPUT
110 FOR J=1 TO 2 :: READ J$
:: DISPLAY AT(12,1)ERASE ALL
:J$&" FILENAME?":"DSK" :: AC
CEPT AT(13,4):F$(J):: OPEN #
J:"DSK"&F$(J),UPDATE :: NEXT
 J
120 LINPUT #1:M$
130 IF ASC(SEG$(M$,LEN(M$),1
))<33 THEN M$=SEG$(M$,1,LEN(
M$)-1):: IF LEN(M$)>0 THEN 1
30
140 PRINT #2:M$ :: IF EOF(1)
<>1 THEN 120 :: CLOSE #1 ::
CLOSE #2
```

Attention all newsletter editors! If you are going to print my Tips (or anything else that contains program listings!) through the Formatter, PLEASE first replace and transliterate the ampersand, asterisk, period, carat and "@" sign!

Print this one through the Formatter and see why -

```
100 A=A!264 :: @=1
110 PRINT "1 . . . 2 . . . 3
. . .4 . . . 5 . . . 6 . . .
7 . . . 8 . . . 9 . . . 0"
120 M$=M$&A$&B$&C$ :: K=K^3
```

Here's how you do it. Load the above in the Editor, position the cursor at the beginning of the 1st line, hit FCTN 9, type RS and Enter, then /&/)/ and Enter. At the prompt, type A. Now get the cursor back to the beginning, repeat the above with /$/!/, and then /./\/ and /^/~/ and /@/(/ and the file should now look like this -

```
100 A=A!264 :: (=1
110 PRINT "1 \ \ \ 2 \ \ \ 3
 \ \ \ 4 \ \ \ 5 \ \ \ 6 \ \
 \ 7 \ \ \ 8 \ \ \ 9 \
\ \ 0"
120 M$=M$)A$)B$)C$ :: K=K~3
```

Now use FCTN 8 to open 5 lines at the top and add this transliteration -

```
.TL 92:46
.TL 123:64
.TL 124:42
.TL 125:38
.TL 126:94
```

Save the result, go to the Formatter and print it.

If my multi-column Printall program (Tips from the Tigercub #45) won't run on your Epson-compatible printer, try changing line 250 to -

```
250 ACCEPT AT(12,3)VALIDATE(
"123")SIZE(1):P :: IF P=2 TH
EN PRINT #1:CHR$(27);CHR$(77
)ELSE IF P=3 THEN PRINT #1:C
HR$(15)
```

You might also need to change the 136 in line 280 to 132.

If your printer offers the elite condensed option, you might want to add -

:" (4) ELITE CONDENSED" to line 240, change the VALIDATE string in 250 to "1234", add ELSE IF P=4 THEN PRINT #1:CHR$(27);CHR$(77);C HR$(15) to the revised line 250 and add +(P=4)*160 to the first statement in line 280.

Memory almost full,

Jim Peterson

TI-PD public domain software for the TI-99/4A computer, $1.50 per disk postpaid (minimum 8 disks please); number of sectors filled is indicated in parentheses). For a 9-page catalog listing all titles and authors, send $1 which is deductable from first order. (specify TI-PD catalog) Offered as a copying service only, without warranty other than that copies are equal to the original. Make checks payable to Tigercub Software (no credit card orders). Send to Tigercub Software, 156 Collingwood Ave., Columbus OH 43213.

600. Sam Moore Jr. Music #1 (341)
601. Sam Moore Jr. Music #2 (343)
602. Sam Moore Jr. Music #3 (348)
603. Sam Moore Jr. Music #4 (337)
604. Bill Knecht Hymns (334)
605. Christmas Music (318)
606. Holiday Music (339)
607. Great Songs by Bill Knecht (351)
608. Music by Bill Knecht (295)
609. March Music (329)
610. Tigercub Country Music (356)
611. Christmas Sing-Along (352)
612. J. Stephen Foster Music #1 (332)
613. J. Stephen Foster Music #2 (317)
614. Bach Music Programs (338)
615. Sing-Along Music (351)
616. Some of the Best Music (343)
617. Classical Music (340)
618. Assorted Music (346)
619. Chopin's Polonaise (280)
620. Assorted Music #2 (351)
621. Hamilton UG Music Package #1 (346)
622. Assorted Music #2 (349)
623. A Diskfull of J.S. Bach (345)
624. Assorted Music #4 (358)
625. Assorted Music #5 (347)
626. J.S. Bach Music (340)
627. Assorted Music #6 (354)
628. Some of the Very Best (349)
629. Ollie Hebert's Music (340)
630. Gregory Rashall Music Master (287)
631. Assorted Music #7 (352)
632. Sonata for Pianoforte (222)
633. Sonata for Pianoforte DS/SD
634. Strange Music (337)
635. Chuck Berry Tunes (218)
636. Christmas Songs w/Graphics (310)
637. Assorted Music #9 (337)
638. Classical Music #2 (338)
639. Assorted Music #10 (347)
640. Marches and College Songs (336)
641. Another Sing-Along (345)
642. Sing-Along Music #2 (236)
643. Christmas Music #2 (351)
644. Christmas Sing-Along #2 (340)
645. Classical Music #3 (352)
646. Assorted Music #11 (350)
647. Music Doodlers and Tinytunes (302)
648. Rhapsodie in Blue (287)
649. Assorted Music #8 (354)
650. Christmas Music w/Graphics 2 (357)
651. Sorgan II (145)
652. Christmas Music w/Graphics 3 (352)
653. Pop Demo V1.1 (225)
654. Christmas Music w/Graphics 4 (255)
655. Assorted Music #12 (349)
701. Musical Education (350)
702. Musical Education #2 (318)
703. Musical Education #3 (188)
710. American Flags (360)
711. Flags of the World (345)
712. Geography - U.S. States (341)
713. Geography - U.S. States #2 (212)
714. World Geography (179)
730. American History (48)
750. Alphabet w/Speech (343)
751. Children's Programs w/speech (357)
752. Alphabet for Preschool (329)
753. Children's Prog. w/Speech #2 (335)
755. Shapes, Colors, Directions (173)
760. Spelling (324)
770. Vocabulary and Reading (293)
780. Preschool Math (341)
790. Elementary Addition, Subtract (257)
791. Addition & Subtraction (337)
796. Multiplication, Division (348)
797. Multiplication, etc. (224)
800. Higher Math (355)
801. Higher Math #2 (228)
810. Typing Practice (223)
815. Morse Code Teacher (155)
820. Health (354)
821. Health #2 (145)
830. Physics (111)
840. Nature (277)
850. Chemistry (277)
860. Astronomy (342)
861. Astronomy #2 (304)
870. Religion (346)
871. Religion #2 (42)
890. Teacher's Helpers (203)
900. Home Utilities (351)
901. Home Utilities #2 (342)
902. Home Utilities #3 (350)
907. Screen Drawing, Doodling (160)
909. High-Resolution Drawing (287)
910. Charts & Graphs (178)
912. Calculators & Converters (345)
913. Calculators & Convert. #2(147)
915. Financial Math (339)
916. Financial Programs (356)
918. Checkbook Programs (203)
920. Business Programs (146)
950. Genealogy
970. Astrology, Numerology etc. (171)
980. Radio Utilities (220)
990. Sports Programs (329)
1100. Character & Sprite Editors(254)
1101. Programmer's Utilities (346)
1102. Sorts, Scrambles, Searches (228)
1105. Auto-loaders (217)
1106. Disk Catalogers (268)
1107. Character Sets etc. (353)
1110. Assembly Utilities (357)
1111. Assembly Utilities, Routines(328)
1112. New Horizon Assembly Util. (269)
1119. Hardware Utilities (169)
1120. Sound Effects (197)
1130. Disk Labels & Jackets (284)
1131. Gemini Printer Utilities (224)
1132. Word Processing Utilities (182)
1133. Banners, Graphs, etc. (203)
1135. Speech Utilities & Demos (355)
1140. Music Composers (288)
1141. Assembly Music Compiler (265)
1145. Telecommunications Aids (342)
1150. Programming Tutorials (348)
1160. Assembly Tutorials #1 (231)
1161. Assembly Tutorials #2 (289)
1162. Assembly Tutorials #3 (357)
1163. Assembly Tutorials #4 (358)
1164. Assembly Tutorials #5 (340)
1300. Mathematical Games (133)
1301. Brain Games #1 (344)
1302. Brain Games #2 (345)
1303. Brain Games #3 (352)
1304. Brain Games #4 (352)
1305. Two-Player Brain Games (335)
1306. Brain Games #5 (345)
1307. Master Mind (322)
1310. Memory Games (235)
1315. Sargon Chess (155)
1320. Mazes #1 (342)
1321. Maze Games #2 (346)
1322. Maze Games #3 (338)
1330. Hangman Games (335)
1331. Wheel of Fortune #1 (249)
1332. Wheel of Fortune #2 (248)
1333. Word Games (310)
1340. Games by Roland Trueman (333)
1350. Card Games #1 (352)
1351. Card Games #2 (348)
1352. Card Games #3 (94)
1356. Dice Games (354)
1360. Board Games (321)
1361. Bingo (73)
1362. Checkers (238)
1363. Board Games #2 (287)
1367. Gambling Games (237)
1381. Bowling (289)
1382. Golf (138)
1383. Billiards, Boxing, etc. (250)
1400. Adventure Disk #1 (360)
1401. Adventure Disk #2 (306)
1402. Adventure Disk #3 (329)
1403. Adventure Disk #4 (324)
1415. Hamurabi Games (268)
1416. Text Games #1 (313)
1417. Text Adventures (340)
1425. Graphics/Text Adventures (354)
1426. Graphics/Text Adv. #2 (322)
1427. Graphics/Text Adv. #3 (325)
1430. Road Race Games (356)
1431. Keyboard Maneuvering (349)
1432. Road Crossing Games (344)
1433. Road Crossing Games #2 (175)
1434 Keyboard Games (347)
1435. Keyboard Maneuvering #2 (354)
1436. Slot Machines (343)
1437. Keyboard Games #2 (353)
1438. Keyboard Games #3 (343)
1440. QBert Games (288)
1445. King Kong Type Games (351)
1455. Assembly Games (231)
1456. Assembly Games #2 (346)
1460. Children's Programs (345)
1461. Fun Games for Kids (353)
1462. Easy Games for Kids (346)
1470. Great Games (342)
1471. Assorted Games #1 (348)
1472. Assorted Games #2 (343)
1473. Texas Games Medley w/speech(346)
1474. Sea Battle Games (329)
1475. Joystick Games (342)
1476. Joystick Games #2 (355)
1477. Joystick Games #3 (346)
1478. Joystick Games #4 (338)
1479. Two-Player Joystick Games (353)
1480. Two-Player Keyboard Games (353)
1481. Joystick Games #5 (345)
1500. Kaleidoscopes & Displays (262)
1501. Sprite Displays (200)
1505. Poetry, Prose & Nonsense (128)
N

TI-WRITER PROBLEMS
from SFV 99er Times

PUNCTION

Typists will always use two spaces after a punction mark ending a sentence. TI-writer, for some strange reason, does things a little different. For example:

The period(.) - TI-Writer will always put 2 spaces after every period that has been followed by a single space. This is fine if the period is at the end of the sentence. But what if you are using an abbreviation within a sentence? The formatter will put 2 spaces here also, but you properly only want one. What you need to do in this case is use the required space symbol (^) after the period of an abbreviation. This will give you the desired one space when using the formatter. (A period followed by no space will appear as just that.)

The exclamation and question marks (!),(?) - In these cases the formatter will not automatically give you 2 spaces as it properly should. To make your document look correct you will need to add one space and one required symbol (^).

THE PERIOD AND DECIMALS

The formatter thinks that any line which begins with a period is a formatter command and will delete the whole line. If by chance your document contains a value such as (.10) and the wraparound caused by Fill and Adjust of the formatter puts it at the beginning of the line, the whole line will disappear. To correct this you could put a zero in front of your decimals (0.10).

ASTERISK AND NUMBERS

If you are printing out of the formatter and your document contains an astrisk followed by two or more numeric digits, the asterisk and the two digits will disappear. For instance, A(*)256 becomes A6. What's happening here is that TI-Writer program misinterprets the astrisk and two digits as an instruction to input data from a "value file", as in mail merge. This is described on page 111 of the TI Instruction book. To correct this problem, you will need to type two astrisks followed by two dummy numbers, then the actual digits. For example, type A(**25256 to print A(*)256.

REQUIRED SPACE

If you tie words together for the purpose of underlining (&) or overstriking (@) with the required space (^), the Fill and Adjust of the formatter will leave gaping blanks in your lines. If you tie too many together, the line will extend beyond the right margin. It would be better to put a seperate (&) or (@) in front of each word. Be sure to include the spaces between the words. If you want a (^) to appear in your text, you will need to transliterate it(see page 107 of the TI Instruction book). The(&) and the(@) are typed twice in succession to get them to print.

OTHER PROBLEMS

Other problems have been noted in TI-Writer that cause erratic and destructive commands, but they are not fully documented.

TI WRITER MNEMONIC (MEMORY) TRICKS

CTRL	MNEMONIC	FUNCTION	ALTERNATE
A	ADVANCE DOWN	ROLL DOWN	F4
B	BACK UP	ROLL UP	F6
C	COMMAND MODE	COMMAND MODE	F9
D	(RIGHT ARROW)	RIGHT ARROW	FD
E	(UP ARROW)	UP ARROW	FE
F	FLYAWAY CHARACTER	DELETE CHARACTER	F1
G	GET A HOLE FOR CHAR	INSERT CHARACTER	F2
H	HOP BACK TO LAST	LAST PARAGRAPH	C6
I	INDENT	TAB	F7
J	JUMP TO NEXT	NEXT PARAGRAPH	C4
K	KILL TO END OF LINE	DELETE TO END OF LINE	--
L	LEAP HOME	HOME CURSOR	--
M	MAKE NEW PARAGRAPH	NEW PARAGRAPH	C8
N	NO MORE LINE	DELETE LINE	F3
O	OPEN BLANK LINE	INSERT BLANK LINE	F8
P	PAGE BEGINNING	NEW PAGE	C9
R	REFORMAT	REFORMAT	C2
S	(LEFT ARROW)	LEFT ARROW	FS
T	TAB BACK	BACK TAB	--
U	(USED FOR SPECIAL CHARACTER MODE)		
V	VEER TO LEFT	CURSOR TO LINE START	--
W	WORD TAB	WORD TAB	C7
X	(DOWN ARROW)	DOWN ARROW	FX
Y	YANK MARGIN CONTROL	LEFT MARGIN RELEASE	--
Z	ZIP BACK	OOPS!	C1
-	----------	SCREEN COLOR	C3
-	----------	DUPE LINE	C5
-	----------	NEXT WINDOW -->	F5
-	----------	WORD WRAP	C0

NOT ONLY DO MOST WORD PROCESSING PROGRAMS RECOGNIZE THE FUNCTION OR CONTROL KEYS USED WITH NUMBERS BUT ALSO MANY HAVE CONTROL/LETTER COMBINATIONS.

REPRINT FROM ROM - USERS GROUP OF ORANGE COUNTY, CA.

THE THEORY OF DARK

By Earl Raguse
(Newt Armstrong, Scientific consultant)

Dark is the natural state of things. To make anything visible to the human eye, you must remove Dark. If you doubt this, go into a windowless dark-tight room, where no Dark can escape. You will not be able to see anything. Now if the sun is up, and you open a small crack in the room, some of the dark will leak out and you will be able to see large objects. If you make the crack large enough, the sun will suck up all the dark and you can then see things clearly.

Many people believe in Light theory; just as they do in Ben Franklin's wrong way theory of electrical current. We however, think that Dark theory is the more reasonable. Rumor is that Al Einstien privately subscribed to Dark theory, but publically used Light theory, because it made it easier to explain his Univeral Relativity Theory to skeptical scientists of that day.

Our sun is the largest darksucker in our solar system. That's how it gets the energy to radiate all that heat and ultra-violet rays. The early Greeks had discovered that lighted candles had some dark sucking capacity, but it took Thomas Edison to invent a good artificial darksucker.

Dark travels in straight lines. Dark cannot penetrate solid objects and thus they make good dark shields. To prove this, look under any car, at high noon, with the sun directly overhead, and you will see a patch of dark that the sun is unable to suck up. Stand with your back to the sun, and you will see that you too are a good dark shield. You will have trapped a patch of dark which can't get around your fat frame to one of the universe's best darksuckers, the sun.

Dark travels at 186,000 mi/sec. Thats why, when you switch on the artificial darksucker (sometimes erroneously called a light) it sucks up the dark so fast that you can't really time it, after all in a 30 foot square room with the darksucker at center ceiling, its less than 22.67 feet to the most remote corner, and with a velocity of 186,000 mi/sec, it only requires about 23 pico seconds for dark to go from the corner to the darksucker. Most people hardly notice the delay.

Dark is made up of all possible frequencies in the subtractive mode, such that each frequency cancels another, thus the frequency of true dark is zero (ie dc). Pure white is total absence of dark, that takes a powerful darksucker like the sun or a good arclamp. Some artificial darksuckers have a frequency bias, and do not suck up all dark, and hence leave the illusion of color.

Certain darksuckers, like neon and sodium lamps eg, although apparently, energy efficient, are primitive with respect to frequency purity, and thus appear to be colored. Any color can be created by subtraction of the correct frequencies.

The proliferation of outdoor artificial darksuckers in our cities, eg Los Vegas, make it very difficult for astronomical observation, they pollute the night sky which in its natural state is void of darksucking, thus enabling high powered astronomical telescopes to detect faint points of dark loss caused by far off galaxies of darksuckers like our sun.

One of the most remarkable darksuckers is modern television; the best of which have a normally black (very dark) screen when unpowered. They incorporate electronics modules, and computers in some cases, and have the ability to convert radiated TV signals so as to suck dark from discrete points on the screen in varying amounts and frequency such that a colored picture appears on the normally dark screen.

What will they think of next. We have heard that a new super-computer will use dark logic. This does not require perfection of room temperature super-conductivity or the expense of super-cooling as is the current state of the art. This new computer will be very fast, extremely energy efficient and so cheap that you can afford one for every program you use, hence saving all that boot up time.

Dark logic is very simple. The basis of which is:

Dark = Not Light
Light= Not Not Not Dark

See how much simpler Dark logic is.

We know that Light theory is a lot of bull, and probably a conspiracy also. Astronomers speak of Black Holes. Now you know this can't be true, how can anything with so much mass be a hole? Actually these things are good Dark Radiators, (light suckers, to the under educated) and are the true source of dark in this universe

Newt Armstrong, a psuedo scientific and computer wizard, has graciously offered to promigate this obscure branch of dark knowledge at the weekly Garage SIG, along with other sagacious information. Call the author for exact time, address and directions for getting there. Like the true academician that he is, Newt will charge no fees before you are hooked on the theory.

This may not be a continuing series.

(Ed note: Some people spend most of their lives in the dark.)

SCHEDULE OF MEETINGS

There will NO club meeting in July or August. The next club meeting is scheduled for Monday September 18 starting at 6:30 PM. Meetings are held the third Monday of each month at the Science Enrichment Encounter (SEE) Center, 324 Commercial Street, Manchester, NH. Below is a list of dates for upcoming meetings.

September 18
October 16
November 20
December 18

NH99'ers User Group
PO Box 5991
Manchester, NH 03108

EDMONTON USER'S GROUP
PO BOX 11983
EDMONTON, ALBERTA
CANADA T5J-3L1