021

8805 Milwaukee

TI-99/4A

HOME COMPUTER SPECIALISTS

WE HAVE WHAT YOU NEED FOR YOUR TI!

COMPETITION COMPUTER PRODUCTS

2629 W. NATIONAL AVE. MILWAUKEE, WIS. 53204
(near the Mitchell Park Domes)

STORE HOURS; MON THRU FRI 10-6   SAT 10-3

672-4010

BANKCARDS - CHECKS - DISCOVER CARDS WELCOME!

WOW! DS/DD DISKS .54 EACH!

GENUINE TI JOYSTICKS $10 PER PAIR!
(with this flyer only - regularly $29.95 - while supply lasts)

PICK UP YOUR COPY OF OUR CATALOG SOON

WE WILL BUY ANY TI HARDWARE OR SOFTWARE YOU NO LONGER NEED - CALL!

WE CUSTOM BUILD IBM COMPATIBLE COMPUTERS & TAKE TI ITEMS IN TRADE.

FEATURING PANASONIC & STAR MICRONICS PRINTERS FROM $189!

NEW AND USED TI99/4A COMPUTERS AVAILABLE!

EXPANSION SYSTEMS AVAILABLE - NEW AND USED!

* HUGE SOFTWARE INVENTORY - MORE IN STOCK THAN EVER BEFORE! *

CALL US FOR TECHNICAL HELP. WE WILL HELP YOU WITH YOUR PROBLEMS. WE WILL TRY TO MEET OR BEAT ANYBODY'S PRICES. REMEMBER THAT WE ARE HERE TO HELP IF YOU HAVE A QUESTION OR PROBLEM. WE WANT YOUR BUSINESS AND WE'LL PROVE IT!

OUR 21ST YEAR IN BUSINESS IN MILWAUKEE AT 27TH & NATIONAL!

TED, GENE, MIKE, JIM AND JERRY

1/88

NOISY KEYS

By Harold A. VanDusen
Milwaukee Area User Group

ARE YOUU BOTHERED BY NNOISY KEYS? My TI99/4A was very generous by providing more characters than I typed. I would often receive two or more U's or N's when I hit those particular keys. There appeared to be no way to gain access to the key contacts for cleaning so I bought one of the TI surplus keyboards available from Radio Shack (*Archer alpha-numeric keyboard cat no. 277-1023*).

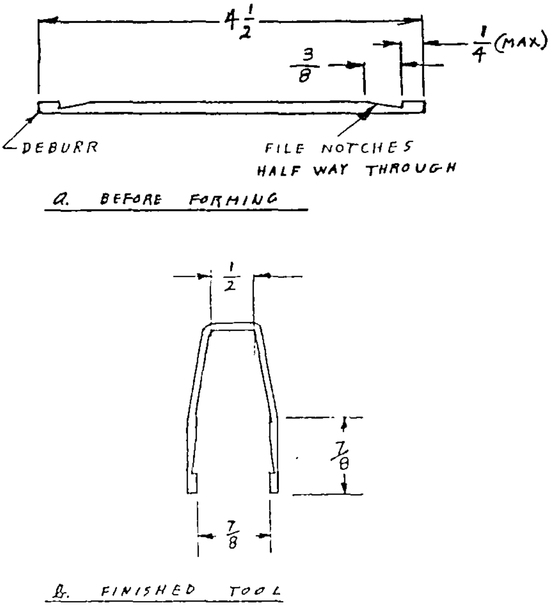
I did not use that keyboard as a replacement because I felt that (after sitting in stock for a number of years) it probably had corroded or dirty contacts and would be troublesome. In addition the console must be almost completely disassembeled to replace the keyboard; so, rather than inviting possible damage, I decided to leave *well enough* alone. I was able to study the replacement keyboard and find a way to gain access to the contacts.

The individual keys are removable from above, but a special tool is required. The keys are tapered and slippery such that they cannot be pulled free with fingers or pliers. It is necessary to grip the bottom dge, but they are too close to each other to allow much of a tool to inserted. There is sufficient space at the corners between adjacent s to allow passage of a 0.080" wire, especially when that key is ressed. This is the clue to the design of a special tool that can e used to pull individual keys off.

The tool is made of a 4.5" long piece of 0.080" dia. hard drawn steel wire as shown in figure 1. The wire was obtained from a wire coat hanger (The two piece type where the horizontal member is a cardboard tube). Many of the one piece coat hangers are made of a heavier wire which will not be satisfactory. File two notches as shown. the depth should be to the mid point of the wire diameter. I found it convinient to file the notches before bending the wire. File a slight round or taper contour to the wire ends to provide a smooth surface to ease its passage between the small gaps between the keys.

In operation the tool is placed over the selected key diagonally across its corners and forced downwards. It will cause the key to depress and the notches will hook the edges of the key. It can then be pulled free. The contacts can then be seen. This is illustrated in figure 2. CAUTION - I had a case where the key stuck too tightly into its contact housing and pulled the housing out with it. I was able to e-insert the housing but much care was needed to avoid damage to the contact assembly.

The contacts are recessed and it takes another special tool to clean them. I used a small flat file from an automotive ignition tool kit. This file was .045" thick and 5/16" wide. I ground the end down to 1/4" width to allow it to be inserted into the contact housing. This tool is described in figure 3. The file fits nicely between the contact faces while it bears on the plastic separator between the contacts. When the contact housing is depressed by pushing down on the file the contacts come into contact with the file and are burnished by movement of the file. Work the file up and down several times and then replace the key. the job is finished.



a. BEFORE FORMING

b. FINISHED TOOL

FIG-1 KEY REMOVAL TOOL
MATERIAL: .080 DIA. HARD DRAWN STEEL WIRE.

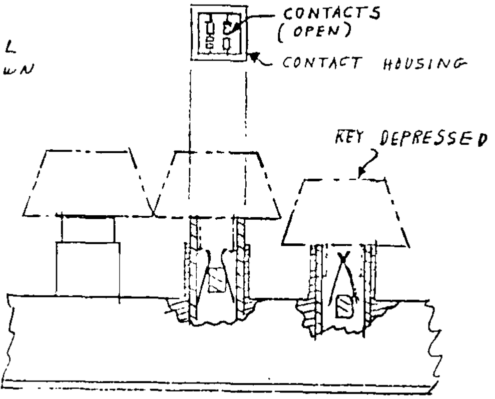


FIG. 2
KEYPAD DETAIL (REMOVEABLE KEY SHOWN IN DASHED LINES)

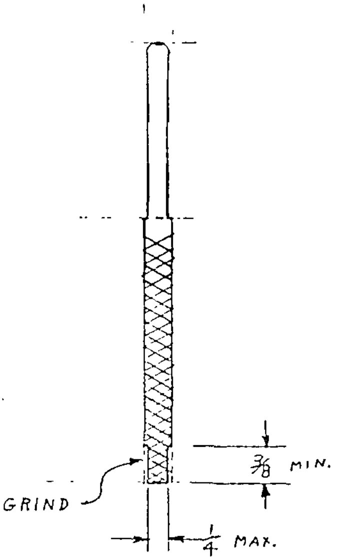


FIG-3 MODIFIED FILE
MATERIAL: 5/16" x .045" THICK IGNITION POINT FILE

SPEECH Part I

HARNESSING THE POWER OF SPEECH

by Craig Dunn

(Reprinted from the Central Texas 99/4A Users Group newsletter, Feb 1986.)

The TI Speech Synthesizer is an amazing little device. It was a breakthrough for the lower end (priced) computers. Unfortunately, many 99/4A owners still don't know how to access speech along with all its little features. Sure, a lot of games use speech to add interest and excitement, but the aplications of speech goes far beyond games.

One of the major features of the speech synthesizer is its ability to let you add speech to your programs. There are several ways to do this, including TI's Terminal Emulator II, XBASIC, and through the use of assembly language routines. XBASIC provides a rather limited vocabulary (unless you are using one of several recent utilities that give you unlimited speech in XB but that's another story). TE2 allows for unlimited speech directly from BASIC. This built-in text-to-speech capability of TE2 will be the focus of this article.

First, plug in the TE2 command module, turn on the computer, and select TI BASIC. Now type and run the following program:

```
100 OPEN #1:"SPEECH",OUTPUT
110 INPUT A$
120 PRINT #1:A$
130 GOTO 110
```

If you get an error, make sure you have the sppech synthesizer connected properly to the side port. Now we have a very simple text-to-speech editor. Line 100 contains the OPEN command needed to access TE2 speech capabilities. Line 120 sends the text strings that you type in to the text-to-speech interpreter, which then sends the info to the synthesizer. Experiment with this for awhile by typing in phrases, followed by an ENTER.

In the above example, you were in the default speech mode. This means that no commands have been sent to alter the voice. We can change the voice easily using the "//" command. The proper format is:

```
//PITCH SLOPE

ex. //34 118
```

The PITCH is a number between 0 and 63. A zero causes the speech synthesizer to wisper phrases. Pitches from 1 to 63 range from the highest pitched (1) to the lowest pitched (63). For best sound, figure the SLOPE using the following formula:

SLOPE = 32 x (PITCH/10)

Round this result to the nearest whole number. Now, when you enter the command along with these two numbers, it will appear that nothing has happened. But type in a simple phrase and press ENTER. You'll notice the change in voice. For example, at the prompt in our simple little speech editor, type "//55 176" and press ENTER. (Be sure to include a space between numbers.) Nothing happened, right? Well, now type something in and press ENTER. See how the voice changed? It became much deeper. Now try "//0 0" and press ENTER. Again, type in a short phrase. Another voice tone! Experiment with these and other PITCH/SLOPE combinations to get the feel of working with these.

Before we wrap up this tutorial, we'll take a look at the inflection symbols. The symbols are; "^" (carat), "_" (underline), and ">" (greater than). The "^", when placed in front of a word, indicates a primary stress point to the text-to-speech interpreter. Only one "^" may be used per string. The "_" is used to indicate a secondary stress point and may be used without limit through the string. The ">" will shift the stress points within a word. Experiment with all these to make words sound better and more human like. Remember, all inflection symbols must precede the word they are to affect.

Well I hope someone benefitted from this article. On a final note, remember that the text-to-speech interpreter is not perfect. Sometimes you might have to alter a words spelling drasticaly to make it sound right. Have fun!

SPEECH Part II

TURBO SPEECH

(or How to Speed up the Spoken Word)

by Stephen Shaw

(Excerpted from the TI99/4A Exchange TI*MES of Great Britain, Issue #6, Autumn 1984)

Now on to something really juicy. SPEECH. Old hat huh? Well, this information will give you speech in TI BASIC with the Mini Memory, or if you have XBASIC with 32K RAM, will give you speech just a mite faster than using CALL SAY which slows programs down no end.

For this information I am indebted to Neil Lawson who has been delving.

Speech requires either:

XBASIC with 32K memory
or: Mini-Memory

and: Speech Synthesizer

Program framework (For timing purposes):

```
 20 CALL INIT
 30 S=-27648
100 FOR I=1 TO 1000 :: NEXT I
110 PRINT "START......"
120 FOR X=1 TO 20
130 REM TEST ROUTINE HERE
140 FOR T=1 TO 30
150 PRINT ")";
160 NEXT T
170 NEXT X
180 PRINT "END......"
```

This standard routine sets up a framework to test our new routine in, and gives a basic time reference.

(NB: Times quoted are for MY system: yours may be different, but the ratios should be similar.)

Running the above program, with the loop in line 140 running 30 times as shown, takes 18.7 seconds from "START" to "END". Change line 140 to loop just 20 times and the timing is 12.7 seconds.

Now we can insert our two possibilities:

The first is available only in XBASIC:

```
130 CALL SAY("#THAT IS
     INCORRECT#")
```

Run the program again: If line 140 is looped 20 times, the time is 44 seconds. If line 140 is looped 30 times, the time is 50-seconds.

The time for the speech is constant, it adds about 21 seconds to the program.

Now for something different, (also works with Mini-Memory):

```
130 CALL LOAD(S,70,"",S,65,"",
S,72,"",S,70,"",S,64,"",S,80)
```

If you now run the program, it says the same thing as many times, but look at the timing:

```
If line 140 loops 20 times: 26.3 S
                  30 times: 26.5 S
```

We know that looping line 140 an extra 10 times adds 6 seconds... so where have those 6 seconds gone?

The CALL SAY routine holds everything up until it has finished speaking. But using the CALL LOAD equivalent, while the computer is speaking, it gets on with the next chore too. The "dead time" is used, and soaks up those 6 seconds.

Thus using the CALL LOAD equivalent, the computer speaks faster, and also permits your program to run more quickly if there is work for it to do between speech outputs.

That's the clever demonstration! (Impressed?) Now for the theroy.

References: Editor/Assembler Manual, pages 351, 355, 422 to 427

(Errata: The reference in para 1, page 355, should be to Section 22.1.4, not as printed in the manual.)

Address -27648 is the SPEECH WRITE address. We keep feeding it with bytes, and in due course the computer speaks. The bytes to feed to that address are found out as follows:

First, decide what you want to say from the standard vocabulary. Then look in the table (pp. 422-427) for the address of that word or phrase. "THAT IS INCORRECT" is given as 6816. That is Hexadecimal not a Decimal number. The four numbers are reversed, and become 6186.

Now we offset them by Hex 40 and feed them in. As we are dealing with decimals with our CALL LOAD, that means we add decimal 64 to each digit in turn:

```
(6+64)  (1+64)  (8+64)  (6+64)
  70      65      72      70
```

(If the numbers were Hex A-F these have a decimal value as follows:

A=10 B=11 C=12 D=13 E=14 F=15

Now we must indicate end of word by loading a zero, again offset, thus 0+64=64. Finally, instruct the computer to speak by loading Hex 50, Decimal 80.

Thus we have loaded, in order;

70,65,72,70,64,80

Check back to the listing. Note the way CALL LOAD has been used; a single command to load the same address with several different values.

To assist your experimentation, here are some Hex addresses from the manual. Remember to reverse them, translate to decimal and offset.

TEXAS INSTRUMENTS...	6696
WHAT WAS THAT.......	77E9
YOU WIN.............	7DDB
ANSWER..............	1913
CHOICE..............	1DA2
ELSE................	28B6
HELP................	3571
INSTRUCTIONS........	39BD
I WIN...............	37CF
NAME................	47C0
PLEASE..............	5093
THAT IS RIGHT.......	68FE
READY TO START......	56B3
AGAIN...............	17A5
CHECK...............	1D82
COMMAND.............	1F1A
GOODBYE.............	3148
HURRY...............	3757
I...................	3793
JOYSTICK............	3AED
NICE TRY............	49A5

This is not only a useful programming aid in its own right, but by demonstrating a part of the TI manual's sometimes complex instructions, it should assist you when you are ready to move on th FORTH or Assembly language proper.

CARE AND CLEANING OF DISK DRIVES
COURTLAND GRAY NUTMEG TI-99ERS

Load your program, load your data, save your data. Load another program. In all these operations you disk drive is revolving your disk at the rate of 5 times per second with the read/write heads in contact with the disk. During this contact the read/write heads pick up small particles of iron oxide which is the coating on the disk that contains our valuable information (either programs or data). The heads require periodic cleaning of these particles in order to maintain error-free operation.

The frequency of cleaning is dependent upon the frequency of use; every three months, six months or a year. The methods of cleaning are also many. There are "dry" cleaning disks, "wet" cleaning disks, "wet" cleaning disks (using a special disk and a solution) and disassembly of the drive from its box and cleaning with a swab and DENATURED alcohol. Regular isopropol alcohol is not recommended because it leaves a residue on the heads.

Considering the three methods available, the least desirable is the "dry" method as it is abrasive to the heads and leads to premature wear. The disassembly and swab method is the most efficient and the technique used by professional drive service companies. The simplest method and the one most of us would use is the "wet" disk solution. These are available as Disk Cleaning Kits, available through office supply houses and computer supply stores. These kits range in price from $6 to $30 and offer multiple cleanings which make the per-cleaning cost average between $.50 and $1.00.

Last, but not least, if you are transporting your drives, place the cardboard shipping protectors in the drive and close the door. Lacking the shiping protectors, put an old disk in the drive backwards. This prevents the heads from vibrating and misalignment.

RUN THIS PROGRAM WHILE CLEANING YOUR DRIVE

```
100 REM HEADCLEAN
110 REM HEAD CLEANING PROGRAM
120 REM !! TI Txtended BASIC only'!
130 REM by Milo Tsukroff NUTMET TI-99ers
140 CALL CLEAR
150 DISPLAY AT(3,1):" HEAD CLEANING PROGRAM": :"Run this
program while the cleaning disk is in the ":"drive."
160 DISPLAY AT(9,1):"To stop it, press and hold CLEAR (FCTN
4) until the":"prog ram stops."
170 DISPLAY AT(15,1):"DISK DRIVE: 1"
180 ACCEPT AT(15,13)SIZE(-1)VALIDATE(DIGIT):G
110 on error 200
200                                                     OPEN
#1:"DSK"&STR$(G)&".DUMMY_FILE",INTERNAL,SEQUENTIAL,VARIABLE
80
210 CLOSE #1
220 END
230 PRINT "RETRY #";
240 X=X+1
250 PRINT X
260 RETURN 190
```

IDS ############

* Catch It! *

This month's game is one that was from the Kansas City 99'er newsletter runs in Extended BASIC.

A THOUGHT FOR 1987

und in the TIC TAC Newsletter...thanx.

7730 S8/PRO News Views
-Dec-86 15:53:42
: #Thought for 1987
: Roy Bartee 71336,436
: ALL

L I EVER NEEDED TO KNOW I LEARNED IN
NDERGARTEN ... by Robert Faughum
nsas City Times Sept.17th 1986

Most of what I really need to know about w to live,and what to do and how to be, I arned in kindergarten. Wisdom was not at e top of the graduate school mountain but ere on the sandbox at the nursery school.

ese are the things I learned:

Share everything.
Play fair.
Don't hit people.
Put things back where you found them.
Clean up your own mess.
Don't take things that aren't yours.
Say your sorry when you hurt somebody.
Wash your hands before you eat.
Flush
Warm cookies and cold milk are good for you.
Live a balanced life.
Learn some and think some and draw and paint and sing sand dance and play and work every day some.
Take a nap every afternoon.
When you go into the world, watch for traffic, hold hands and stick together.
Be aware of wonder.

emember the little seed in the plastic cup. e roots go down and the plants go up, and body really knows how or why, but we are l like that.

LOADED TO THE COMPUSERVE TI FORUM

========= THANX, I NEEDED THAT ==========

```
100 ! CATCH "IT"
110 ! FAMILY COMPUTING
120 ! FEBRUARY 1987
130 ! TRANSLATED BY W. BLOOD
160 CALL CLEAR :: FOR T=1 TO
 12 :: CALL COLOR(T,16,1)::
NEXT T :: CALL SCREEN(5)!170
A$="I" :: C$=CHR$(32):: DN$=
"X" :: UP$="E" :: N1$=C$&C$
:: F$=RPT$("F",16)
180 CALL CHAR(136,F$&"FF7F3F
1F0F070301FF0000000000000000
"):: CALL COLOR(14,7,1)
190 CALL CHARPAT(73,E$,84,B$
):: CALL CHAR(128,E$&B$&F$):
: CALL COLOR(13,11,1):: I$=C
HR$(128)&CHR$(129)
200 P$=CHR$([illegible]RS(138)&C
HR$(137)&CHR$ [illegible] :: N2$=N1$
&N1$
220 DISPLAY AT(2,3)ERASE ALL
:"Welcome to CATCH IT!"
230 [illegible]AY AT(4,1):"Use the
 (";UP$;") and (";DN$;") key
s"
240 DISPLAY AT(5,2):"to move
 your ";CHR$(34);"catcher";C
HR$(34)
250 DISPLAY AT(6,6):"up and
down."
260 DISPLAY AT(8,1):"The goa
l is to catch "&CHR$(34)&I$&
CHR$(34)
270 DISPLAY AT(9,1):"before
it hits the right"
280 DISPLAY AT(10,1):"border
 of the screen."
290 DISPLAY AT(12,1):"Please
 select the level"
300 DISPLAY AT(13,1):"of dif
ficulty you prefer."
310 DISPLAY AT(15,1):"Would
you like"
320 DISPLAY AT(17,1):"(1) ea
sy,":"(2) moderate, or"
330 DISPLAY AT(19,1):"(3) ha
rd?":::" )"
340 ACCEPT AT(21,5)BEEP SIZE
(1)VALIDATE("123"):K$
350 DD=VAL(K$)*.5
360 RD=1 :: SC=0
380 DISPLAY AT(11,9)ERASE AL
L:"Get Ready! "
400 FOR DE=1 TO 1000 :: NEXT
 DE :: CALL [illegible]
410 CALL H[illegible] [illegible]3,130,28)
420 CALL [illegible] [illegible]3,130,28)
430 CALL [illegible] [illegible]2,136,30)
440 CALL [illegible] 24,2,136,30)
450 CALL V[illegible] [illegible]31,136,[illegible]
460 CALL V[illegible] [illegible]2,136,23)
470 CALL [illegible] [illegible]3,130,21)
480 CALL VC[illegible] [illegible]30,130,21)
490 DISPLAY AT(1,1):"Round:"
;RD
500 DISPLAY AT(1,13):"Score:
";SC
510 X=10 :: GA=4
520 Y=4 :: Z=INT(RND*16)+4 :
: GA=GA-(GA<100)
530 DISPLAY AT(X,24)SIZE(4):
P$
550 H1=Z :: HY=Y :: RANDOMIZ
E :: N=RND :: IF Y=22 THEN 6
90
560 Z=Z+DD*((N>.5)-(N<.5))
570 Y=Y+1 :: Z=Z-DD*((Z<4)-
Z>22))
580 DISPLAY AT(H1,HY)SIZE[illegible]
:N1$
590 DISPLAY AT(Z,Y)SIZE(2):I
$
610 CALL KEY(5,K,S):: IF S=0
 THEN 550
620 K$=CHR$(K)
630 IF K$<>UP$ AND K$<>DN$ T
HEN 550
640 H2=X :: X=X-(K$=DN$)+(K$
=UP$)
650 X=X-(X<4)+(X>20)
660 DISPLAY AT(H2,24)SIZE(4)
:N2$
670 DISPLAY AT(X,24)SIZE(4):
P$ :: GOTO 550
690 DISPLAY AT(H1,HY)SIZE(2)
:N1$
700 IF Z-X>2.5 OR Z-X<0 THEN
 Y=Y+2 :: GOTO 810
720 DISPLAY AT(Z,23)SIZE(1):
A$
730 FOR T=5 TO GA
740 CALL COLOR(2,RND*10+6,RN
D*5+1)
750 CALL SOUND(250,T*25+110,
0):: SC=SC+(0*DD
760 DISPLAY AT(1,19):SC :: N
EXT T
770 CALL COLOR(2,16,1)
780 DISPLAY AT(Z,23)SIZE(1):
C$
790 FOR DE=1 TO 200 :: NEXT
 DE :: GOTO 520
810 HY=Y :: Y=Y+.5 :: DISPLA
Y AT(Z,Y)SIZE(2):I$
820 DISPLAY A[illegible] SIZE(2):
N1$ :: IF Y>2[illegible] 310
840 RD=RD+1 :: IF RD<4 THEN
380
860 DISPLAY AT(8,5)ERASE ALL
:"Sorry, you, missed "&CHR$(
34)&I$&CHR$(34)
870 DISPLAY AT(10,9):"Three
times!"
880 DISPLAY AT(12,1):"Your s
core was";SC;"points."
890 DISPLAY AT(15,1):"Would
you like to"
900 DISPLAY AT(17,1):"(1) pl
ay again":"    at the same l
evel,"
910 DISPLAY AT(19,1):"(2) se
lect a new level, or"
920 DISPLAY AT(20,1):"(3) qu
it/re[illegible]n to BASIC?"
930 [illegible]LAY AT(22,1):"Enter
your choice."
940 DISPLAY AT(24,3):")"
950 ACCEPT AT(24,5)BEEP SIZE
(1)VALIDATE("123"):K$
960 IF K$="3" THEN CALL CL[illegible]
R :: END
970 IF K$="2" THEN CALL CLEA
R :: [illegible] 290
980 GO[illegible] 360
```

PROGRAMS THAT WRITE PROGRAMS
Part 2
by Jim Peterson

Last month I promised you something more useful, so here it is. This routine will come in very handy for formatting screen text into neat 28-column lines, and will save the text in program lines of DATA statements. When you are ready to save, type @@@ and enter as the last line, then NEW and MERGE DSK1.LINEFILE -

```
100 !LINEWRITER to aid in fo
rmatting screen text into 28
-column format and saving it
 as DATA program lines in ME
RGE format - by Jim Peterson
110 !strings containing comm
as and quotation marks will
be ACCEPTed, and converted t
o DATA statements which RUN
correctly even though they
120 !are not enclosed  in qu
otation marks!
130 CALL CLEAR :: OPEN #1:"D
SK1.LINEFILE",VARIABLE 163 :
: LN=30000
140 FOR R=1 TO 24 :: DISPLAY
 AT(R,1)SIZE(1):" " :: ACCEP
T AT(R,0)SIZE(-28):A$ :: IF
A$="@@@" THEN 180 :: B$=B$&C
HR$(200)&CHR$(LEN(A$))&A$
150 X=X+1 :: IF X/4=INT(X/4)
THEN 160 ELSE B$=B$&CHR$(179
):: GOTO 170
160 GOSUB 210 :: LN=LN+10
170 NEXT R :: X=0 :: CALL CL
EAR :: GOTO 140
180 IF B$="" THEN 200 :: IF
SEG$(B$,LEN(B$),1)=CHR$(179)
THEN B$=SEG$(B$,1,LEN(B$)-1)
190 GOSUB 210
200 PRINT #1:CHR$(255)&CHR$(
255):: CLOSE #1 :: END
210 PRINT #1:CHR$(INT(LN/256
))&CHR$(LN-256*INT(LN/256))&
CHR$(147)&B$&CHR$(0):: B$=NU
L$ :: RETURN
```

Oh - that puzzle in last month's article? Try creating those DATA statements with this LINEWRITER program!

Now, let's get down to business and learn how to do all this. First, let's write a program that will write a program to list the token codes that you need to use to write a program that will write a program -

```
100 OPEN #1:"DSK1.TOKENLIST"
,DISPLAY ,VARIABLE 163,OUTPU
T :: FOR N=129 TO 254 :: L1=
INT(N/256):: L2=N-256*L1
110 PRINT #1:CHR$(L1)&CHR$(L
2)&CHR$(131)&CHR$(N)&CHR$(0)
:: NEXT N
120 PRINT #1:CHR$(255)&CHR$(
255):: CLOSE #1 :: END
```

Key that in, RUN it, then enter NEW, then MERGE DSK1.TOKENLIST. Now LIST it and you will see a list of ASCII codes 129 through 254 and their token meanings. Delete lines 171 through 175, 185, 198, 226 through 231, and 242. Change the definition of 199 to QUOTED STRING, of 200 to UNQUOTED STRING, and 201 to LINE NUMBER, and add line 255 !END OF FILE.

You don't need all those exclamation points, so change the program to a DIS/VAR 80 file by LIST "DSK1.TOKENLIST". Then key in this little routine.

```
100 OPEN #1:"DSK1.TOKENLIST"
,INPUT :: OPEN #2:"PIO" !or
whatever
110 PRINT #2:CHR$(27);"N";CH
R$(6)
120 LINPUT #1:A$ :: PRINT #2
:TAB(10);SEG$(A$,1,4)&SEG$(A
$,6,255):: IF EOF(1)<>1 THEN
 120 ELSE CLOSE #1 :: END
```

RUN it, and print out a list of all the token codes. Keep it handy, you'll be needing it. Notice that every Extended Basic statement has its own ASCII token code - even the ones you perhaps never heard of, such as LET and GO. Notice also that every keyboard symbol which affects program execution, such as + and =, has its own ASCII token code which is NOT the same as its keyboard ASCII code. And notice that the double colon, used as a separator in Extended Basic multi-statement lines, has its own token.

Now, let's take a look at how a MERGE format program is put together. This routine will do that for you - and you will also find it very useful in debugging the MERGE programs you are going to write.

```
100 DISPLAY AT(3,5)ERASE ALL
:"D/V 163 FILE READER": :"
    by Jim Peterson": : :" T
o edit a file saved or":"cre
ated in MERGE format."
110 DISPLAY AT(12,1):"Output
 to? (S/P)S";"(S)creen":" (
P)rinter" :: ACCEPT AT(12,17
)SIZE(-1)VALIDATE("SP"):Q$
120 IF Q$="P" THEN DISPLAY A
T(14,1):"PRINTER? PIO" :: AC
CEPT AT(14,10)SIZE(-18):P$ :
: D=2 :: OPEN #2:P$
130 DATA ELSE,"::",!,IF,GO,G
OTO,GOSUB,RETURN,DEF,DIM,END
,FOR,LET,BREAK,UNBREAK,TRACE
140 DATA UNTRACE,INPUT,DATA,
RESTORE,RANDOMIZE,NEXT,READ,
STOP,DELETE,REM,ON,PRINT,CAL
L
150 DATA OPTION,OPEN,CLOS
UB,DISPLAY,IMAGE,ACCEPT,E
R,WARNING,SUBEXIT,SUBEND,
,LINPUT
160 DATA ,,,,,THEN,TO,STE
,",";",":",),(,&,,OR,AND,
,NOT,=,<,>,+,-,*,/,^,
170 DATA QUOTED STRING,UNCL
TED STRING,LINE NUMBER,EOF,A
BS,ATN,COS,EXP,INT,LOG,SGN,S
IN
180 DATA SQR,TAN,LEN,CHR$,PN
D,SEG$,POS,VAL,STR$,ASC,PI,R
EC,MAX,MIN,RPT$,,,,,,,NUMERI
C,DIGIT
190 DATA UALPHA,SIZE,ALL,USI
NG,BEEP,ERASE,AT,BASE,,VARIA
BLE,RELATIVE,INTERNAL,SEQUEN
TIAL,OUTPUT,UPDATE,APPEND
200 DATA FIXED,PERMANENT,TAB
,#,VALIDATE
210 DIM T$(126):: FOR J=1 TO
 126 :: READ T$(J):: NEXT J
:: E$(1)="LINE NOT CLOSED WI
TH CHR$(0)"
220 DISPLAY AT(16,1):"FILENA
ME? DSK" :: ACCEPT AT(16,14)
:F$
230 ON ERROR 240 :: OPEN #1:
"DSK"&F$,VARIABLE 163,INPUT
:: GOTO 250
240 DISPLAY AT(20,1):"I/O ER
ROR" :: ON ERROR STOP :: RET
URN 220
250 ON ERROR 260 :: LINPUT #
1:A$ :: X=ASC(SEG$(A$,1,1)):
: Y=ASC(SEG$(A$,2,1)):: IF X
=255 AND Y=255 THEN 410 ELSE
 270
260 PRINT #D:"FILE NOT CLOSE
D PROPERLY":"WITH CHR$(255),
CHR$(255) ?" :: STOP
270 PRINT #D:"LINE NUMBER":X
;"TIMES 256=";X*256;Y;"PLUS"
;Y;"=";X*256+Y
280 FOR J=3 TO LEN(A$)-1 ::
X=ASC(SEG$(A$,J,1))
290 IF X=201 THEN PRINT #D:X
;"LINE NUMBER" :: X=ASC(SEG$
(A$,J+1,1)):: Y=ASC(SEG$(A$,
J+2,1)):: J=J+2 :: PRINT #D:
X;"TIMES 256=";X*256;Y;"PLUS
";Y;"=";X*256+Y
300 IF X=199 THEN PRINT #D:
;"QUOTED STRING" ELSE IF X=
00 THEN PRINT #D:X;"UNQUOTE
 STRING" ELSE GOTO 360
310 J=J+1 :: X=ASC(SEG$(A$,
,1)):: PPINT #D:X;"OF";X;"C
ARACTERS"
```

```
320 ON ERROR 340 :: FOR L=1
TO X :: Y=ASC(SEG$(A$,J+L,1)
):: PRINT #D:Y;CHR$(Y):: IF
Y<32 OR Y>126 THEN PRINT #D:
"UNPRINTABLE CHAR - ERROR?"
330 NEXT L :: J=J+X :: GOTO
370
340 PRINT #D:"ERROR! INSUFFI
CIENT BYTES IN":"STRING" ::
IF ASC(SEG$(A$,LEN(A$),1))<>
0 THEN PRINT #D:E$(1)
350 ON ERROR STOP :: RETURN
250
360 IF X<129 THEN PRINT #D:X
;CHR$(X);" VARIABLE NAME" EL
SE PRINT #D:X;T$(X-128)
370 CALL KEY(0,K,S):: IF S=0
 THEN 390
380 CALL KEY(0,K2,S2):: IF S
2<1 THEN 380
390 NEXT J :: IF ASC(SEG$(A$
,J,1))=0 THEN PRINT #D:"0 EN
D OF LINE" ELSE PRINT #D:E$(
1)
400 GOTO 250
410 PRINT #D:X:X;"END OF FIL
E" :: CLOSE #1 :: STOP
```

PROGRAMS THAT WRITE PROGRAMS
Part 3

Jim Peterson

Let's start learning how to actually write a program that writes a program.

A MERGEd program is a D/V 163 file, so -

```
OPEN #1:"DSK1.(filename),VARIABLE
 163,OUTPUT
```

Every program line begins with a line number, of course. In MERGE format the line number, whether 1 or 32767, is squished into two characters. We don't need to get into how this is done, but you can accomplish it with CHR$(INT(LN/256))&CHR$(LN-256*INT(LN/256)), where LN has ben predefined as the line number.

To print a statement or command, anything that is represented by a token in the token list, just print the CHR$ of its token ASCII. For instance, the token for DATA is 147, so you would print CHR$(147).

To print a variable name, either numeric or string, jus enclose it in quotes, "A" or A$"

To print a value, or a strin which is not in quotation mark (such as in a DATA statement), o the word which follows a CALL you must print CHR$(200) followe by a token giving the number o characters to follow, such a CHR$(5) for a 5-letter word suc as CLEAR, then the value i quotes. For instance, the toke for CALL is 157, so CALL CLEAR i CHR$(157)&CHR$(200)&CHR$(5)&"CLE R" .

Similarly, tokens for paren theses are 183 and 182, so th variable name A(1) is "A"&CHR (183)&CHR$(200)&CHR$(1)&"1"&CHF$ 182).

A quoted string is handled i the same way except that it i preceded by token 199, so PRIN "HELLO" is CHR$(156)&CHR$(199 &CHR$(5)&"HELLO". Don't worr about the quotation marks, th computer will handle that.

If you need to refer to a lin number, as in GOTO 500, use toke 201 followed by the line numbe formula, thus CHR$(134)&CHR$(201 &CHR$(INT(500/256))&CHR$(500-256 INT(500/256)).

Don't print more than 16 characters in a record. You ca print multiple-statement XBasi lines, but be sure to use th double-colon token 130 as th separator, not two of the 18 colon tokens.

Each program line must end wit CHR$(0) as the end-of-lin indicator, and the last recor you print must be CHR$(255)&CHF (255) as the end-of-fil indicator.

If you get an I/O ERROR 25 whe you try to merge your program, i means that you left off the fina double-255. If the progra merges, but crashes when you ru it, you will probably be able t spot an obvious error in the lin when you LIST it. If the lin looks OK but gives you a DAT ERROR or SYNTAX ERROR, you lef off a CHR$(0) or gave the wron count of characters after toke 199 or 200. The progra published in Part 2 will help yo to track down these bugs.

Now let's write a program. What is the longest possible one-liner program?

Well, RANDOMIZE is the longest statement that can stand alone. It is represented by the single token 149, and to repeat it must be followed by the double-colon token 130. Since any line number will take two bytes, let's use a 5-digit line number. And don't forget that final CHR$(0). That still leaves us 160 of the 163 bytes, so we can repeat tokens 149 and 130 for 79 times, followed by a final 149.

```
100 OPEN #1:"DSK1.LONG",VARI
ABLE 163,OUTPUT
110 FOR J=1 TO 79 :: M$=M$&C
HR$(149)&CHR$(130):: NEXT J
:: M$=CHR$(254)&CHR$(254)&M$
&CHR$(149)&CHR$(0):: PRINT #
1:M$ :: PRINT #1:CHR$(255)&C
HR$(255)
120 CLOSE #1
```

RUN, NEW, MERGE DSK1.LONG and LIST - over 34 lines long! But that one-liner doesn't do anything, so try this one -

```
100 OPEN #1:"DSK1.LONG",VARI
ABLE 163,OUTPUT
110 FOR J=1 TO 52 :: M$=M$&C
HR$(162)&"X"&CHR$(130):: NEX
T J :: M$=CHR$(254)&CHR$(254
)&M$&CHR$(162)&"X"&CHR$(0)::
 PRINT #1:M$
120 PRINT #1:CHR$(255)&CHR$(
255):: CLOSE #1
```

Again RUN, enter NEW, then MERGE DSK1.LONG, then RUN. You'll get a message BREAKPOINT IN 32510 (don't ask me why!) but just enter RUN again.

HOCUS

Home Computer Users Spotlight
a monthly publication of the
Milwaukee Area 99/4 Users Group

MAY-1988

MILWAUKEE AREA USER GROUP
4122 GLENWAY WAUWATOSA WI 53222

President...D.Walden		5292173
Vice-Pres...J.Schroeder		2644735
Treasurer...P.Norton		4628954
Secretary...B.Kling		5295161
Librarian...E.VonDerEhe		5490593
Librarian...F.Pabian		3273618
Newsletter..G.Hitz		5350133
S.I.G.....Schroeder/Walden/Hitz		

Next Group Meeting - 2nd Saturday
June 11,1988 - 12 noon til 4 PM
Wauwatosa S & L - 7500 West State

North Sub-Meeting - 1st Tuesday
June 7, 1988 - 7 PM til 10 PM
Security S & L - 5555 N Pt Washington

South Sub-Meeting - 3rd Tuesday
May 17, 1988 - 7 PM til 10 PM
Franklin State Bank - 7000 So 76th

Membership Dues $10 - Family $15

<<<<< HOCUS NEWSLETTER INDEX >>>>>

HAPPY BIRTHDAY MILWAUKEE AREA TI USER GROUP

It was just 7 years ago when a small group of struggling pioneers met to form some sort of computer self-help unit in the void that was Texas Instruments at that time. Gene Hitz, Phil Norton, Ken Schmidt, Earl Schultz, Dean Cleveland and Lucretia Harmon were just the beginning, soon to joined by James Urmanski, Jim Steinhart, Mark Geer, Jim Vincent, Dave Zigler and the Browns. The beginning admittedly was pretty rough at times, with no support, no software and very little in the way of any hardware or peripherals. And besides, who even could possibly afford them, at prices prevalent then. We've gone through a lot of good and bad times since then. Remember the Crash of '83 (has it really been almost 5 years ago ?). Every one remembers what they were doing when the news reached them, and how they reacted.
Immediately our group doubled in membership and we couldn't find enough seating room at our meetings. After about a year though, the membership dropped back to the pre-crash years and has held steady since right up to today. Hopefully we can still squeeze a few more good days out of the old console. Mine is still getting just as much use as always, and is still perking along on all four burners, still giving me as much pleasure as when it was fresh out of the box. I'm very comfortable with it and it never ceases to amaze me with what it can do. If ever this group does disband for lack of interest, my old TI will be my faithful and trustwothy companion for many long years hence.