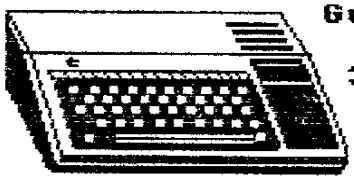
GUILFORD 99'ERS NEWSLETTER

SUPPORTING THE TEXAS INSTRUMENTS TI-99/4A COMPUTER



GUILFORD 99'ERS UG
3202 CANTERBURY DR
GREENSBORO NC
27408

TO:

VOLUME 9 NUMBER 1 JANUARY - MARCH 1992

, Pres. Bob Carmany, Newsletter Ed (855-1538)
Tony Kleen, Sec/Treas (924-6344) Bill Woodruff, Pgm/Library (228-1892)
BBS: (919)621-2623 --ROS

The Guilford 99'er Users' Group Newsletter is free to dues paying members (One copy per family, please). Dues are $12.00 per family, per year. Send check to:Tony Kleen c/o 3202 Canterbury Dr., Greensboro, NC 27408. The Software Library is for dues paying members only. (Bob Carmany Ed)

OUR NEXT MEETING

DATE: December 3,1991 Time: 7:30 PM. Place: Glenwood Recreation Center, 2010 S. Chapman Street.

Program for this meeting will be our Christmas program --sounds of the season, good cheer, and the "odd" surprise thrown in for good measure. Be sure to be there and bring a "munchie" with you and we'll share the lot with everyone!

MINUTES

The November meeting of the Guilford 99'er Users's Group was held on Tuesday the 12th at the Glenwood Recreation Center. There were five members present.

The Treasurer reported that we have $137.49 in the treasury. We donated $25 to the American Heart Association in memory of our late President, George von Seth.

There were not minutes from last month's meeting since there was no business transacted at that meeting. We just went right into the program (re: The GRAMKRACKER). Thanks to Roy West and Mac for their insight. Thanks for Bob for his overview of Funnelweb.

Bob brought it to our attention that The Hunter Valley UG has been resurrected. Contact Geoff Phillips, New Hunter Valley Computer Club, PO Box 347, Warners Bay, NSW 2282, Australia for details. Other news is that the periennial final (?) version of Funnelweb 4.40 is out with yet another version promised.

The program for the evening was presented by Tony Kleen. He demonstarted how to use TI-Base to keep track of installment loans. Once again, an excellent program.

Respectfully submitted,

Tony Kleen Sec/Treas

EDITOR'S NOTES

The December issue marks the end of the monthly newsletters for the Users' Group. Input for the newsletter has dwindled to near nothing and it is virtually impossible to continue at the current schedule. Beginning in January, the newsletter will be published quarterly and it is anticipated that the length will be increased as well. Maybe with a couple of months between issues, we can get enough to fill 18 or so pages. Let's hope!

Remember, your annual dues are due and collectable in January as well. Bring your $$$ to the December or mail your check to the UG mailing address on the newsletter. Come back for another year --- we ain't finished yet!

In fact, read this next bit if you think your TI is primitive when compare to the (ugh!) IBM.

UP YOURS---IBM

By Bob Carmany

I ran across this little "gem" while going through our newsletter at work. MultiMate is the word processor that is used frequently in the IBM PS/2 world. Just read the instructions as they are written for the IBM and compare them with those in the next section for F'WEB and the TI.

"Say, for example, that you are creating a document and would like it to contain text that already exists in another, previously created document. Instead of typing the text into the new document, you can simply copy it from the existing document. Here is how. . .

Using the "Create a Document" or "Edit a Document" commands from the MultiMate Main Menu, place on the screen the document into which you wish to copy the text (I will call this the target document).

Next, use the arrow keys to move the cursor in the target document to the location where you want the copied text to appear.

Hold down the SHIFT key and tap the F8 key. The target document will temporarily disappear from the screen and a document directory will appear. The words "EXTERNAL COPY" should appear in the upper right-hand corner of the screen.

Use the F4 and F5 keys to change drives and paths as needed to locate the document from which you wish to copy the text (I will call this document the source document).

Use the arrow keys to move the red highlight to the source document file name and tap the F10 key. The first page of the source document should appear on the screen. The words "START COPY WHERE" should appear in the upper right-hand corner of the screen.

Use the arrow keys to move the cursor to the beginning of the text that you want to copy and tap the F10 key. The words "COPY WHAT?" should appear in the upper right-hand corner of the screen.

Use the arrow keys to move the cursor to the end of the text that you want to copy and tap the F10 key. The source document should then disappear from your screen and the target document should reappear with the copied text in the correct location."

Now, for the TI version of the same operation using the same terminology:

Load the target document into F'WEB. Hit FCTN-9 and LF. The format for "external copy" is [start line] [end line] [after line] DSKx.filename. The first two entries can be omitted if you wish to copy the entire source text file. You can also use "E" for the third entry if you wish for the source text to be added to the end of the target file.

It's a lot easier that all of the "garbage" we went through on the IBM. Here are a couple of examples:

0001 0100 0200 DSKx.Filename copies lines 0001 through 0100 after line 200 in the text.

0050 0075 E DSKx.filename copies lines 0050 through 0075 at the end of the target file.

E DSKx.filename copies the entire source file at the end of the target file.

UP YOURS----IBM!!!

```
100 REM ********************
110 REM * BARS AND GRAPHS *
120 REM ********************
130 REM BBY JOHN GUNTER
140 REM HOME COMPUTER MAGAZI
NE
150 REM VERSION 4.3.1
160 REM TI BASIC
170 CALL CLEAR
180 PRINT TAB(7);"BARS AND G
RAPHS": : : : : : :
190 INPUT "USE A PRINTER (Y/
N)?":P$
200 IF (P$<>"Y")&(P$<>"N")TH
EN 190
210 IF P$="N" THEN 290
220 PRNT=1
230 INPUT "PRINTER DEVICE NA
ME?":DEV$
240 GOTO 290
250 CALL KEY(0,K,S)
260 IF (S=0)+(K<>13))THEN 25
0
270 CALL CLEAR
280 RETURN
290 PRINT TAB(10);"PLEASE WA
IT": :
300 PRINT TAB(12);"GOTTA RES
T"
310 DIM CO(32)
320 DIM A$(11)
330 A$(1)="00000000000007E"
340 A$(2)="0000000000007E7E"
350 A$(3)="000000000007E7E7E"
360 A$(4)="00000007E7E7E7E"
370 A$(5)="0000007E7E7E7E7E"
380 A$(6)="00007E7E7E7E7E7E"
390 A$(7)="007E7E7E7E7E7E7E"
400 A$(8)="7E7E7E7E7E7E7E7E"
410 A$(9)="0101010101010101"
420 A$(10)="0101010101010FF
"
430 A$(11)="FF01010101010101
"
440 CALL SCREEN(13)
450 CALL COLOR(9,5,1)
460 CALL COLOR(10,9,1)
470 CALL COLOR(11,11,1)
480 CALL COLOR(12,15,1)
490 COL(1)=5
500 COL(2)=9
510 COL(3)=11
520 COL(4)=15
530 FOR X=95 TO 158 STEP 8
540 FOR CH=1 TO 8
550 CALL CHAR(X+CH,A$(CH))
560 NEXT CH
570 NEXT X
580 CALL CHAR(42,"935539FF39
5593FF")
590 CALL CHAR(91,"0000000000
0000FF")
600 CALL CHAR(92,A$(9))
610 CALL CHAR(93,A$(10))
620 CALL CHAR(94,A$(11))
630 CALL CLEAR
640 SET=0
650 CALL VCHAR(1,7,92,21)
660 FOR X=1 TO 21
670 CALL HCHAR(X,8,91,25)
680 NEXT X
690 CALL HCHAR(21,8,93,24)
700 FOR TIC=5 TO 20 STEP 5
710 CALL HCHAR(TIC,7,93)
720 NEXT TIC
730 CALL HCHAR(1,7,94)
740 ST$="MAXIMUM DATA VALUE
? "
750 GOSUB 770
760 GOTO 1190
770 D1$=D$
780 D$=""
790 FOR ST=1 TO LEN(ST$)
800 CALL HCHAR(24,2+ST,ASC(S
EG$(ST$,ST,1)))
810 NEXT ST
820 KI=1
830 CALL SOUND(100,800,2)
840 CALL HCHAR(23,7,42)
850 CALL KEY(0,K,S)
860 IF S=0 THEN 850
870 IF K=13 THEN 1140
880 IF K=7 THEN 1080
890 IF K=8 THEN 980
900 IF K=9 THEN 930
910 IF K<>32 THEN 1030
920 IF (ST$="SIDE LABEL ? ")
+(ST$="BOTTOM LABEL ? ")+(ST
$="LEGEND 1")+(ST$="LEGEND 2
")THEN 1030
930 IF M+1>32 THEN 840
940 CALL HCHAR(21,M,93)
950 M=M+1
960 CALL HCHAR(21,M,42)
970 GOTO 840
980 IF M-1<8 THEN 840
990 CALL HCHAR(21,M,93)
1000 M=M-1
1010 CALL HCHAR(21,M,42)
1020 GOTO 840
1030 IF KI>25 THEN 840
1040 CALL HCHAR(23,7+KI,K)
1050 D$=D$&CHR$(K)
1060 KI=KI+1
1070 GOTO 840
1080 CALL SOUND(300,300,2)
1090 FOR ES=KI TO 1 STEP -1
1100 CALL HCHAR(23,7+ES,32)
1110 NEXT ES
1120 D$=""
1130 GOTO 820
1140 CALL HCHAR(23,1,32,32)
1150 CALL HCHAR(24,1,32,32)
1160 IF D$<>"" THEN 1180
1170 D$=D1$
1180 RETURN
1190 FOR X=1 TO LEN(D$)
1200 IF (ASC(SEG$(D$,X,1))<4
8)+(ASC(SEG$(D$,X,1)))>57)THE
N 1240
1210 NEXT X
1220 IF (LEN(D$)<1)+(LEN(D$)
>5)THEN 1240
1230 GOTO 1260
1240 GOSUB 1870
1250 GOTO 740
1260 MD=VAL(D$)
1270 SCA=(INT(MD/20)*20)+20
1280 IF INT(MD/20)<>MD/20 TH
EN 1300
1290 SCA=SCA-20
1300 DIV=100
1310 LA$=STR$(SCA)
1320 FOR MK=1 TO 21 STEP 5
1330 FOR LAB=1 TO LEN(LA$)
1340 CALL HCHAR(MK,2+LAB,ASC
(SEG$(LA$,LAB,1)))
1350 NEXT LAB
1360 DIV=DIV-25
1370 SCA1=SCA*(DIV/100)
1380 LA$=STR$(SCA1)
1390 NEXT MK
1400 ST$="SIDE LABEL ? "
1410 GOSUB 770
1420 IF LEN(D$)<25 THEN 1440
1430 D$=SEG$(D$,1,24)
1440 FOR LAB=1 TO LEN(D$)
1450 CALL VCHAR(LAB,2,ASC(SE
G$(D$,LAB,1)))
1460 NEXT LAB
1470 ST$="BOTTOM LABEL ?"
1480 GOSUB 770
```

```
1490 R=22
1500 Y=7
1510 GOSUB 1530
1520 GOTO 1570
1530 FOR LAB=1 TO LEN(D$)
1540 CALL HCHAR(R,Y+LAB,ASC(
SEG$(D$,LAB,1)))
1550 NEXT LAB
1560 RETURN
1570 M=0
1580 ST$="COLOR?1-BLU,2-RED,
3-YEL,4-GY"
1590 GOSUB 770
1600 IF (ASC(D$)<49)+(ASC(D$
)>52)+(LEN(D$)>1)THEN 1580
1610 C=VAL(D$)-1
1620 IF SET=0 THEN 1770
1630 CH=127
1640 CH1=135
1650 SE=SET-1
1660 CST=13+SET-1
1670 IF CST<17 THEN 1790
1680 CST=CST-1
1690 SE=SE-1
1700 CALL SOUND(300,300,2)
1710 D$="COLOR FULL-ENTER ,ER
ASE OR END"
1720 Y=1
1730 R=24
1740 GOSUB 1530
1750 ST$=""
1760 GOTO 1800
1770 CH=95
1780 CH1=103
1790 ST$="ENTER DATA"
1800 CALL HCHAR(21,M,42)
1810 CALL HCHAR(21,M-1,93)
1820 GOSUB 770
1830 IF D$<>"C" THEN 1900
1840 IF SET=0 THEN 1580
1850 SET=SET+1
1860 GOTO 1580
1870 CALL SOUND(300,300,2)
1880 ST$="BAD DATA-TRY AGAIN
"
1890 RETURN
1900 IF D$="END" THEN 2330
1910 IF D$<>"ERASE" THEN 196
0
1920 CALL HCHAR(21,M,93)
1930 CALL VCHAR(2,M,91,19)
1940 CALL HCHAR(1,M,95)
1950 GOTO 1770
1960 IF (D$="P")&(PRNT=1)THE
N 2560
1970 IF D$="NEW" THEN 630
1980 FOR X=1 TO LEN(D$)
1990 D=ASC(SEG$(D$,X,1))
2000 IF (D<47)+(D>57)THEN 20
30
2010 NEXT X
2020 GOTO 2050
2030 GOSUB 1870
2040 GOTO 1820
2050 D=VAL(D$)
2060 IF D>SCA THEN 2080
2070 GOTO 2100
2080 GOSUB 1870
2090 GOTO 1820
2100 BAR=D/(SCA/20)
2110 HT=INT(BAR)
2120 RE=BAR-HT
2130 TP=1+INT((RE*8)+.5)
2140 IF TP<9 THEN 2160
2150 TP=8
2160 SE=C
2170 CALL CHAR(CH+TP+(SE*8),
A$(TP))
2180 CALL CHAR(CH1+(SE*8),A$
(8))
2190 CALL VCHAR(21-HT,M,CH1+
(SE*8),HT)
2200 CO(M)=C
2210 IF HT>19 THEN 2250
2220 CALL HCHAR(20-HT,M,CH+T
P+(SE*8))
2230 CALL VCHAR(1,M,91,19-HT
)
2240 GOTO 2260
2250 CALL HCHAR(21-HT,M,CH1)
2260 IF M<32 THEN 2310
2270 CALL SOUND(300,300,2)
2280 ST$="DATA FULL"
2290 GOSUB 770
2300 GOTO 1790
2310 M=M+1
2320 GOTO 1790
2330 CALL SOUND(500,800,2)
2340 ST$="LEGEND 1"
2350 GOSUB 770
2360 D2$=D$
2370 ST$="LEGEND 2"
2380 GOSUB 770
2390 D3$=D$
2400 D$=D2$
2410 R=23
2420 Y=7
2430 GOSUB 1530
2440 D$=D3$
2450 R=24
2460 Y=7
2470 GOSUB 1530
2480 CALL KEY(0,K,S)
2490 IF S=0 THEN 2480
2500 IF K=80 THEN 2560
2510 IF K=78 THEN 630
2520 IF K=83 THEN 2540
2530 GOTO 2480
2540 CALL CLEAR
2550 END
2560 OPEN #1:DEV$
2570 FOR X=1 TO 24
2580 FOR P=1 TO 32
2590 CALL GCHAR(X,P,Z)
2600 IF Z<96 THEN 2620
2610 Z=35
2620 IF (Z<91)+(Z>94)THEN 26
40
2630 Z=95
2640 PRINT #1:CHR$(Z);
2650 NEXT P
2660 PRINT #1:""
2670 NEXT X
2680 CLOSE #1
2690 GOTO 1820
```

TIBase Topic - TUTORing #02
by Tony Kleen, Guilford TI99er Users Grp

This article deals with creating a database, adding records to a database, and editing records that already exist on a database.

CREATE

Databases may be created via the CREATE directive

CREATE (filename) i.e CREATE ADDRESS

If no device is included in the file-name, the default device defined by DATDISK as shown in the status display will be used.

The status may be displayed via the DISPLAY STATUS directive.

The default device may be changed via the SET directive.

i.e. SET DATDISK=DSK2.

Once the create directive has been entered, an editor will allow the record structure to be defined. Up to 17 fields of (C)haracter, (D)ate, (N)umerical or (X)printer control code attributes may be defined.

HELP (F7) is available to describe the function key usage.

(*)

Some CREATE examples:

1a) CREATE DSK.DATA.ADDRESS

1b) SET DATDISK DSK.DATA
CREATE ADDRESS

2a) CREATE DSK2.ADDRESS

2b) SET DATDISK DSK2
CREATE ADDRESS

The first two examples create a database, called ADDRESS, on a diskette named DATA. The last two examples create a database, called ADDRESS, on the 2nd disk drive, no matter what the name of the diskette might be.

The examples '1a' and '1b' produce the same results, as do '2a' and '2b'. If I have a diskette named DATA on my 2nd disk drive, all four examples will produce the same results.

3.2.1 CREATING A DATA-BASE

Creating a data-base consists of defining the content of each record in terms of names, data-types, and sizes. This allows TIBase to create a database file and be able to associate the names that you define with the values that you enter. The databases will be a collection of the records you create. The following data-types may be used in creating a database:

(C)haracter - The width specifies the number of characters to be contained in the field.

(N)umeric - The width specifies the total number of digits including sign and decimal. Note that a sign is mandatory, which means the minimum size is 2 for a numeric type. The decimals specify how many digits are placed to the right of the decimal point.

(D)ate - The width is always 8, which includes the slashes between month/day/year.

he(X) - The width must be even. Values are entered as HEX digits and represent control codes to be transmitted to the printer.

Database names may be up to 8 characters long.

There must be a formatted diskette the selected device.

Once the 'CREATE (database)' direct has been entered, an interactive display will allow the definition of the database structure. You may enter up to 17 fields of (C)haracter, (N)umeric, (D)ate, or he(X) data types.

Once the structure is defined, EXECUTE , <F8>, will continue the process of creating the database. At the completion of this, you will be asked if you wish to input data, at this time. If so, an interactive display will allow data to be entered. If you do not wish to enter data, the database will simply be available, on the named device, for later use.

Any 5 databases may be active at any time. Each database may be active on a different device, if desired.

Database limitations are:
255 characters per field;
17 fields per record;
16129 records per databases.

Now, we're all familiar with the fact that a database is nothing more than a collection of records; that each record is a collection of field(s); that each field is a collection of character(s). Right? If not, you know it now!

I'm going to create a database of addresses, giving it the name ADDRESS. The first process of the CREATE directive is to define the field(s), the number of character(s) in that field, and the character type for that field. This will be the database's STRUCTURE. Again, the database's structure is nothing more that its collection of field(s); plus each field's length and character type. So, we'll be describing the collection of field(s) that our database will possess on each and every record in its database.

Enough verbage, let's actually enter a CREATE directive and see what it give

us. After entering the directive CREATE ADDRESS at the .dot prompt, the following screen is displayed:

```
arrows to move, enter to advance

FIELD  DESCRIPTOR  TYPE  WIDTH  DEC

  1    _________   _     ___    __
  _    _________   _     ___    __
  _    _________   _     ___    __
  _    _________   _     ___    __
  _    _________   _     ___    __
  _    _________   _     ___    __
  _    _________   _     ___    __
  _    _________   _     ___    __
  _    _________   _     ___    __
  _    _________   _     ___    __
  _    _________   _     ___    __
  _    _________   _     ___    __
  _    _________   _     ___    __
  _    _________   _     ___    __
  _    _________   _     ___    __
  _    _________   _     ___    __
  _    _________   _     ___    __
  _    _________   _     ___    __
```

It is expected that you enter a value, and then hit 'enter' to advance to the next value. DESCRIPTOR is nothing more than the 'field name'. TYPE and WIDTH have been described previously. DEC is the number of decimal places for a (N)umeric type of field. After you have entered a field's values, that is, after you enter the last value on a line, TIBase will advance to the next line.

What follows is my completed structure for the ADDRESS database. I have completed entry of my six fields, and have entered a <F8>, function 8, to inform TIBase of my completion. TIBase then creates an empty database on the diskette, and then prompts you as to whether or not you wish to enter data at this time.

```
FIELD  DESCRIPTOR  TYPE  WIDTH  DEC

  1    RECNUM       N     006    00
  2    ADDR_LN_1    C     024
  3    ADDR_LN_2    C     024
  4    CITY         C     018
  5    STATE        C     002
  6    ZIPCODE      C     010

do you wish to input data now? (Y/N):
?
```

If you wish to populate your database at this time, simply enter (Y)es. TIBase will then prompt you with it's APPEND interactive editor, which, for my database, looks like the following:

```
                APPEND

RECNUM      _____                    000
ADDR_LN_1   ____________________
ADDR_LN_2   ____________________
CITY        ____________
STATE       __
ZIPCODE     ________

000 1 ADDRESS 00000/00000eof
```

Looks very similar to the STRUCTURE, doesn't it? Each field of the record is presented in the order you described that field to the STRUCTURE. To the right of each field is an entry area for you data. The size of this area is dependent upon the size of the field, as you entered it on the STRUCTURE. If the size of the field is greater than 26, then, as you enter data for that field, the editor will tab to the right 10 characters to allow you to continue entering data. That's the reason for the three zeroes to the right of the RECNUM entry. As you enter data, that number is incremented to show you what character number you are entering. This number is not of much value if your field sizes are less than 26 characters, but is necessary when enter data for field sizes greater than 26. As you enter data, you can use the arrows to move left and right, and the insert and delete keys.

3.2.4 APPENDING DATA

As each record is filled with data, either an EXECUTE, <F8>, or an ENTER at the last field will cause the data to be stored and another blank record be made available for appending data.

When the APPEND session is complete, an ESCAPE, <F9>, will return to the main screen.

(*)

Now then, if you don't wish to populate the data base after you have entered the structure, you simply answer (N)o to the 'do you wish to input data now?' question; and the CREATE session will return to the main screen.

At this point, you have CREATEd a database, and may or may not have populated that database with data.

USE

Once a data-base has been created it is made active via the USE directive.

USE (filename) i.e. USE DSK2.ADDRESS

If no device is specified, the default device in DATDISK will be utilized. The USE directive will cause the named data-base to become active in the currently selected slot. If a data-base is already active in the current slot, either a CLOSE must be issued to close the active data-base or another slot must be selected.

When you have completed using the data-base, it may be closed via the CLOSE directive.

Note that all databases must be closed prior to terminating the TI-BASE program or data may be lost.

The QUIT directive will close all active databases prior to terminating TI-BASE.

----------(*)

After you have CREATEd a database, this is the only way you can USE that database, ie, issue the USE directive. Seems logical enough, right?

If you don't specify a specific disk drive number, or disk name, TIBase will default to the DATDISK that you have previously specified. That discussion on 'slots' will be covered later, in the SELECT discussion. Right now, we're doing everything in slot #1.

I probably should have covered this earlier. When you wish to end your TIBase session, you enter the QUIT directive. This directive will CLOSE any opened databases prior to exiting the TIBase application software.

You'll see an example of the USE directive after we introduce APPEND and EDIT.

\\

APPEND

To add data to an existing data-base the APPEND directive is utilized:

APPEND (BLANK) i.e. APPEND

APPEND BLANK will add a blank record to the data-base which may then have data replaced or edited into it.

APPEND will allow the user to input data interactively from the keyboard.

----------(*)

APPEND cannot be used unless you have an active database. Remember, databases are made active by the USE directive.

If you enter APPEND, TIBase will place you in the APPEND interactive editor. This is the same editor presented to you by answering (Y)es to the 'do you wish to input data now?' after you have CREATEd you data base's STRUCTURE. We covered this just above in the CREATE section.

APPEND BLANK will add JUST ONE record to the database. You do not get placed into the APPEND interactive editor. All the field values are space filled.

\\

3.2.4 APPENDING DATA

APPEND BLANK will append a blank record to the current database and return immediately. This allows blank records to be appended in the command language, and data to be replaced into the records programmatically.

----------(*)

I'll cover APPEND BLANK when we get to command files, which is considered programming.

\\

EDIT

The EDIT directive allows the user to enter data interactively.

EDIT (n) i.e. EDIT 5

The specified record will be retrieved and presented for editing. If no record number is specified, the current record will be retrieved.

After editing the record, it may be saved by an execute (F8) or by an ENTER at the last field. Paging (F5) or (F6) simply retrieves new records and does not save them.

When the desired data has been edited ESCAPE (F9) will return to the main screen. HELP (F7) is available to define the allowable function keys.

----------(*)

EDIT cannot be used unless you have an active database. Remember, databas are made active by the USE directive.

If you enter EDIT, TIBase will place you in the EDIT interactive editor. EDIT editor is very similar to the APPEND editor. With APPEND, you're always adding a new record to the end of the database. With EDIT, you're always modifying an old record, one that already exists. Since you're editing existing records, this editor allows you to 'page forward', <F5>, or to 'page backward', <F6>.

EXECUTE, <F8>, and ESCAPE, <F9>, function the same as in the APPEND editor.

Okay, now that I've introduced CREATE, USE, APPEND, and EDIT; let's give an example:

We're already done the following:

```
SET DATDISK DSK.DATA
CREATE ADDRESS
```

We've also entered the database's STRUCTURE, which can be reviewed in the above CREATE section. When I created this structure, I answered (N)o to the 'do you wish to input data now?' question. I now wish to enter data. Here's what's needed:

```
USE ADDRESS
APPEND
```

After entering the APPEND directive, I'm presented the APPEND interactive editor. At this point, I enter the following records. Remember, the APPEND editor presents me another blank record after I enter the last field on each record. To conclude entry, I simply ESCAPE, <F9>, the APPEND editor.

\\

APPEND

```
RECNUM      000000                    006
```

```
ADDR_LN_1  3239 High Cliffs Road___  021
ADDR_LN_2  ________________________  000
CITY       Pfafftown_____________   009
STATE      NC                       001
ZIPCODE    27040-9552               009


        000 1 ADDRESS 00000/00000eof
-------------------------------------------

\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
             APPEND

RECNUM     000001                   006
ADDR_LN_1  279 Sara Lane_________   013
ADDR_LN_2  Apartment #B-9________   014
CITY       Winston-Salem____        013
STATE      CN                       001
ZIPCODE    27106____                005

        000 1 ADDRESS 00000/00001eof
-------------------------------------------

\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
             APPEND

RECNUM     000002                   006
ADDR_LN_1  4874 Spainhour Mill Road 024
ADDR_LN_2  ________________________  000
CITY       Tobaccoville_____        012
STATE      NC                       001
ZIPCODE    27045____                005


        000 1 ADDRESS 00001/00002eof
-------------------------------------------
```

At this point, I have entered the ESCAPE, <F9>, to exit the APPEND interactive editor. This returns us to the .DOT prompt. Our database is still in use: we have not yet CLOSEd it nor have we QUIT.

If you notice above, I have incorrectly entered CN for Winston-Salem. We all know this city resides in North Carolina (NC). We need to edit our record.

At the .DOT prompt, I enter EDIT, to enter the EDIT interactive editor. This presents us with the 'current' record, the last record we accessed. In this case, the 'current' record is the last record we APPENDed:

```
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
             EDIT

RECNUM     000002                   000
ADDR_LN_1  4874 Spainhour Mill Road
ADDR_LN_2
CITY       Tobaccoville
STATE      NC
ZIPCODE    27045


        000 1 ADDRESS 00002/00003eof
-------------------------------------------
```

We want the record previous to this one. We have two ways to get to this record. We could ESCAPE and EDIT 1; or we could simply PAGE BACKWARD, <F5>. I take the simplest method, <F5>.

```
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
             EDIT

RECNUM     000001                   000
ADDR_LN_1  279 Sara Lane            000
ADDR_LN_2  Apartment #B-9           000
CITY       Winston-Salem            000
STATE      NC                       001
ZIPCODE    27106


        000 1 ADDRESS 00001/00002eof
-------------------------------------------
```

What I've done here is hit the ENTER key four times to get to the STATE field entry. Then I keyed over my mistake. At this point, I would want to EXECUTE, <F8>, my change.

After I have EXECUTEd my change, the EDIT editor presents me with the next record. At this point, I have made my change, and am ready to ESCAPE, <F9>.

After I have ESCAPEd, I am once again presented the .DOT prompt. I have now CREATEd and populated my ADDRESS database. I am now ready to QUIT, quite literally.

END of DISCUSSION. We have CREATEd, USEd, APPENDed, and EDITed our ADDRESS database. What more could we possibly hope to do? LOTS!! Later.

FOOTNOTE: (*) These sections were copied from Dennis Faherty's TUTOR diskette, or the TIBase Users Guide.

LOWERCASE

by Bob Carmany

Here is a short program that will produce a series of true lowercase letters in your programs instead of the "small caps" of the normal TI character set. When you RUN the program, the character set will appear across the bottom of the screen. If you want to use the characters in one of your own programs, simply erase line 100, 180, and 190 and resequence the remaining lines so that they are at the beginning of your program. Then, whenever you select a lowercase letter, it will appear as the corresponding letter in the demo instead of a small capital letter.

```
100 CALL CLEAR :: FOR U-33 TO 94 :: CALL CHARPAT(U,U$):: CALL CHAR(U,SEG$(U$,3,1 4)):: NEXT U

110 CALL CHAR(97,"000038043C243C",98,"20203824242438",99,"00001824202418",100,"0 4041C2424241C")

120 CALL CHAR(101,"00003C243C203C",102,"081410103C101",103,"00001C24241C0418",10 4,"20203824242424")

130 CALL CHAR(105,"0010001010101",106,"000800080808083",107,"20202428302824",108 ,"3010101010101")

140 CALL CHAR(109,"00007854545454",110,"00003824242424",111,"00001824242418",112 ,"0000382424382020")

150 CALL CHAR(113,"00001C24241C0404",114,"00003C2420202",115,"00003C203C043C",11 6,"00103810101018")

160 CALL CHAR(117,"0000242424241C",118,"0000242424181",119,"00005454545428",120, "00004428102844")

170 CALL CHAR(121,"000024241408102",122,"00003C0810203C")

180 PRINT "abcdefghijklmnopqrstuvwxyz"

190 FOR DE=1 TO 2000 :: NEXT DE :: END
```

, Pres. Bob Carmany, Newsletter Ed (855-1538)
Tony Kleen, Sec/Treas (924 6344) Bill Woodruff,Pgm/Library (228-1892)
BBS: (919)621-2623 --ROS

The Guilford 99'er Users' Group Newsletter is free to dues paying members (One copy per family, please). Dues are $12.00 per family, per year. Send check to:Tony Kleen c/o 3202 Canterbury Dr., Greensboro, NC 27408. The Software Library is for dues paying members only. (Bob Carmany Ed)

NEXT THREE MEETINGS

DATE: January 7, 1992 Time: 7:30 PM. Place: Glenwood Recreation
February 4, 1992 Center, 2010 S.
March 3,1992 Chapman Street

EDITOR'S NOTES

This issue marks the beginning of a 'new' endeavor. As you might have noticed, we have listed the meeting dates for the first three months of 1992. That also means that the newsletter will be coming out on a quarterly basis as well. Hopefully, the newsletter will be a bit thicker since there will be additional time to gather material. Of course, it also means that you won't be getting your newsletter quite so often.

I had an interesting phone conversation with a fellow in Green Bay, Wisconsin the other day. It seems that he ordered one of the Quest cards last year and had just now gotten around to putting it together. The reason that he called was, of course, because he was having trouble getting it on-line. After some checking, changing a jumper here and there, and finally checking all of his solder joints, he was convinced that it might never run properly. Luckily, he followed up the call with a letter. The problem was that he had one of those "antique" Foundation 128K cards in his system. Once it was removed, everything went just fine. It seems that there is a memory contention between the teo of them. When I last heard from him, he was marvelling at the sheer speed of his new Quest RAMdisk.

RETROSPECTIVE

By Tony McGovern

The last issue of the HV-99 Newsletter is an appropriate time for some retrospection, particularly on the early days. I suppose the first thing to ponder on is how a bunch of computer hobbyists in Australia ever came to get mixed up with Texas to such an extent. In our case we have never been there, the nearest being New Mexico on several occaisions over the years. Last time in the US we lived through much of the year in what was a Texas culture camp in the summer -- the Colorado Chataqua of Boulder right at the foot of the mountains. In those days EEs I had known years before as analog circuit designers were mucking around at home with computer kits called KIM-1 and the like, and fully packaged computers were around but still expensive. Our friend Ez in Boulder even had a Commodore PET down in the basement and the kids played their first computer game on that. Some would say Will has never stopped. There was some pre-publicity about a consumer level computer from TI, but no real detail.

Several years later back in Australia, Atari games machines were all the

rage among the junior set, and the pressure was on for one of those for Xmas '82. With the usual importer and retailer ripoffs these were hideously expensive ($275 if I remember corrctly) and I said to myself --no way, for that sort of money it has to be a genuine computer even if it did cost more. The first flirtation was with a Dick Smith Wizzard, but that proved so hopeless in the keyboard department tha I returned it for a refund (it took more of an arguement than Dick Smith advertising would have you believe). I wonder if any of these are still in use? It had a TI 9918 VDP in it though. Then there was a magazine writeup of the new TI-99/4A computer with a BASIC that sounded quite comprehensive, and games cartridges. Xmas was getting close and things were getting desperate. The only known supplier was the then Myer store in Sydney. So that was how we ended up spending twice as much as for an Atari. The selection of a cartridge to buy, and they were scarce that summer - the computer had cost so much that we could only afford one anyway - came to a choice over the phone of Hangman or Munchman. Not knowing what either one was, I chose Munchman which turned out to be one of their best games. [Ed note: The seasons are reversed in the Southern Hemisphere so Xmas falls in their summer]

So off to Queensland for Xmas with Nana. The TI was a big hit, even if not quite an Atari, but various cousins had those anyway. They couldn't program anything. so that equalized it a bit. The TI marketing department had it right there, one cartridge is never enough, so a desperate search started around the Brisbane area. The holy grail was the mythical Extended BASIC, not yet descended from the high plains of Texas. A trek to the north of Brisbane, at an under-the-house (that Queensland all in one word concept) computer workshop yielded Hunt the Wumpus and an unusual cartridge called a Mini-Memory. Another expedition went this time to Ipswich, the town where my mother came from. So there just a couple of doors along from the E. Bostock & Sons building (funny - I don't think the kids have ever come to terms with having a great-great-grandfather called Enoch) we found A-Maze-ing and the dreaded Car Wars. I wonder what it is that makes one cartridge an all-time favorite like A-Maze-ing? The real sleeper of this bunch was the Mini-Mem.

On return to Newcastle we found there was indeed a dealer in town, and they had a bunch of notice slips from someone called Brian Rutherford, which started our association with what was to eventually came to be the Hunter Valley 99ers. Then came several months of waiting for Extended BASIC. When it finally arived it proved to be a very fine implementation of BASIC, well in advance of what was available on other personal computers until quite recent times, even if slow. A language worthly of some serious programming. One of the things that I found very disappointing was the failure of most published XB programs to use the most powerful feature of the language, the user-defined subprogram. It was like the writers of these programs felt constrained to limit themselves to the parts of XB also found in the primitive BASICs of most other machines instead of using all the power of XB. This set off writing a set of XB programming tutorials, initially in the Sydney group newsletter, and continued in the HV99 Newsletter when our group went its seperate way. These have turned up in many newsletters around the world, even in Swedish translation.

Mini-mem, however, allowed access to the assembly language and even at the level of the Line-by-Line Assembler it was clear that there was something well worth pursuing. I had been intending to learn to program at assembly language level, but the 9900 structure and the assembly language looked so much more satisfying to and engineer's outlook that it became the one of choice. Nine years later it still looks good and is still a pleasure to work with. The device independent I/O system still looks more like a real engineered job than most, even today. We were waiting for the 99/8 to appear before making a bigger investment, when Black Friday rolled around. After a little soul-searching, the elegence of TMS-9900 code won out, and we bought

New-Tech's demo /4A system and ordered E/A and TI-Writer.

The next big programming event was the release from TI, with source code as well, of FORTH, a language that promised speed not attainable with XB. When it finally leaked northwards from Sydney, it captured the imagination of several people - jusy ask Joe and Richard about it - and what is more the source code revealed all sorts of essential information that had not been in the E/A manual. After a brief flirtation with FORTH we went back to mastering Assembler. The great thing about Assembler is that nothing and no-one gets between you and the raw power of the computer. I see the real problem with higher level languages as being the addition of an obscuring layer of someone else's understanding of how the machine should best be used, no matter how helpful it might be a lot of the time. Writing in Assembler on the 99/4A is doubly rewarding because the 99-4A BASICs do not use the 16-bit TMS-9900 directly, but are written in the infamous GPL, an interpreted language simulating an imaginary 8-bit processor. The double interpretation just kills the execution speed, so the improvement obtainable is huge.

So what have we seen in the way of hardware advances over the years? Incremental advances came by way of DSDD capable floppy disk controllers from CorComp and later Myarc --in a sense just a quantative improvement that TI would have made anyway in time. The first real change, the one that has made much of what the machine is today, was the Horizon RAMdisk and its developments. The programmable auto-booting in particular has meant the 99/4A remains the easiest of all computers to use, and RAMdisks have saved untold disk drive wear and tear for assembly programmers. On other machines it was (and still is mostly) a big deal to have a RAMdisk that survives normal system reset, let alone one that usually survives programs running wild, or even the machine power being turned off. The more recent great event has been the introduction of 80 column units based on the Yamaha V9938 development of the original TI VDP. Firstly this came as the clumsy Mechatronics unit from Germany, then as the more refined DIJIT PE-Box card from San Diego, and now the OPA TIM from Toronto, which looks to be the best of all. Some might vote for the Myarc HFDC and hard disk capability to join this select company, but this device needs another round of developments to fix its flaws, but it looks like it will never get it. Others may well have their own favorite devices of the many produced. The one area which has not yet been satisfactorily resolved with universal agreement is that of RAM extenders. The E/A "super-cart" with its extra 8K is universally recognized, but pre-empts all other cartridges. TI did have in the works a memory expansion system of general applicability, but this was never taken up, a real tragedy for 99/4A users.

I have not included the Myarc Geneve 9640 in this list. It really has been a development off to one side, and kept that way by the non-communicative, closed-system policies of Lou Phillips and the little circle of software developers who were never quite up to delivering the goods. Didn't they learn anything from the TI debacle with all its corporate resources behind it?

The intervening years have been largely filled with the development of the Funnelweb system. There were a few other programs along the way. The first, done as a learning exercise for XB was a pure XB game with sprite handling code so finely tuned to TI XB that Myarc XB-II, which was supposed to be faster cannot run it. The next major one was a multi-column lister for XB programs that grew out of the tutorial articles, and showed just how helpful XB subprograms are to good program writing. Then came the first versions of Funnelweb, born out of fear of the exorbitant rip-off pricing of TI products by Imagic. It was clear by then that TI-Writer was the main reason for keeping with the 99/4A and all of a sudden that GROM cartridge looked very fragile. In retrospect, not a single cartridge has ever gone bad on us, but TI's life sentence on everyone of cartridge swapping forever did

not appeal to us either. So after an Easter long weekend of poring over my disassembly of the Editor, the essential ideas clicked into place --how the code overlay system worked, and the magic little code patch that allowed a full ShowDirectory to be incorporated. What amazes (and depresses me too) is what we found a couple of years later when the original TI source code for the Editor finally leaked to Newcastle, was just how little had been done to it in the meanwhile. The same problem is still bedevilling the TI-99 community -- the original TI source code rarely leaked to those willing and able to do something with it. Assembler and Formatter source availability would have made a great deal of difference to Funnelweb development. Now that Gary Bowser at OPA in Canada has officially licensed TI code, it is to be hoped the programming skills there are adequate in quality and quantity to do something wirthwhile. For most of us it is now getting too late in their piece to care anymore anyway.

So where has it all led to? The HV99 group was fun while it lasted, which was in fact for much longer than anyone would have dared predict at the time of its formation, and its influence has been felt way beyond all expectations. Now the time has come to wind up the HV99 group without it suffering lingering decay, so we all remember it in its prime. 99/4A support by way of the Funnelweb system will continue for a while yet here, but inevitably involvement with newer and brighter things will bring active program development to an end, even if the machine itself remains in use here. Will has now been off on the Amiga for several years, and is still an assembler freak. It is clear in retrospect that the professional level of the E/A facilities, the organized 99/4A system structure, and the elegant TMS-9900 16-bit instruction set all combined to give a training ground for the later computer that was second to none, and far superior to that available on other personal computers of the time. It was a home mini-computer in disguise! This is the basis of the best advice I can give to anyone continuing with their TI or just starting with a newly bequeathed one. It was and is an excellent computer system, and remains an excellent self-education tool in programming and getting to understand computers. There isn't much point coming to it new as a package user --PCs and the like have overwhelmed that scene.

Perhaps the wheel has turned full circle. Now that advanced users are making use of the Yamaha 9938 VDP in their TI-99s, development work at Funnelweb Farm an the Amiga will soon be turning to a new and powerful generation of Texas Instruments graphivs co-processors, stating with a 60 Mhz TMS-34010.

RAVE

A company in Connecticut is marketing two replacement keyboards for the TI. Both keyboards feature standard key layouts which eliminate two-key functions like cursor control, FCTN keystrokes, and "keyfront" keys such as ",?,\, and [. The Alpha Lock function will not interfere with joystick operation on any version of the TI-99. This hardware will work with all software up to simultaneous keystrokes such as FCTN/CTRL/SHIFT. NO P-BOX IS REQUIRED.

Both keyboards have coiled cords for connection to the computer console so you can move your keyboard around without problems from the side connector. You will have one-key entry of word processing commands such as TAB, BACK TAB, DELETE, BACKSPACE, and HOME when using TI-Writer. There is no indication as to how these things will hook up to the 4A. The information I have says that the keyboard interface installs in place of the standard keyboard in the computer console with NO soldering required. The interface has provisions for user upgrade to include SYSTEM RESET and

LOAD INTERRUPT capability from the keyboard.

As near as I can determine, you remove the 4A keyboard from the console and install their interface which connects to the keyboard. The 99/84 keyboard enhancement has an 84 key enhanced version of an IBM PC/AT layout. There is an LED indicator for the CAPS LOCK and NUM LOCK keys. It will also have an adjustable keyboard tilt. There are 10 function keys along the left side for FCTN1-10 and they can be used for CTRL1-10 when SHIFTed. You can optionally use overlay strips with the number key row as we're doing now with the 4A. Price for this keyboard will be $149.95.

It's a nice looking keyboard and has the 10-key on the right hand side. The model 99/101 is the deluxe package. It is a heavy duty 101 key layout. There are FOUR separate key assignment modes which allow the user to have TWO separate command/programming modes. These modes can place the often used double quote (") in the unshifted key position for ease of use, for example. The QUIT key may be implemented as a single keystroke or may be defeated depending upon the mode selected. There are TWO separate editor modes also. One implements word processing keys for IT-Writer, the other for use with Editor/Assembler.

You have dedicated cursor control and a ten-key on the right. The function keys are arranged in a row at the top of the keyboard, allowing for direct use of standard overlay strips. This keyboard will have a 25-foot coiled cord as an option ($9.95). The 99/101 will sell for $164.95. Availability is expected to be about October 1 for both units. For more information, contact: Rick @ (203)242-4012 or John @ (203)872-9272, both after 6 pm Eastern time. These guys can also do custom key assignments such as Dvorak or Multiplan for a minimum of $15. RAVE 99, 23 Florence St., Bloomfield, CT 06002

PRESCAN-IT

One of the annoying things about TI Extended Basic is the long wait you have between typing in RUN and when the program starts. This is because XB scans the program and sets up memory space for variables, arrays, and data. This wait can be considerable if you have a long program with a lot of arrays and variables. The pre-scan commands are a bit obscure and unless you are an extremely disciplined programmer, they are hard to use. Well, Asgard has just released PRE-SCAN IT! to help us out. You won't find the pre-scan commands listed or discussed in the Extended Basic book, they're in the supplement that came with the book and you really don't get much help as to how to use them.

Using the PRE-SCAN IT! program (hereafter referred to as PSI), a relatively inexperienced XB programmer (such as myself) can use these commands to not only speed up execution time, but your program may end up shortened and even run faster. First, load the XB program you want to work on and RES it so that the line numbers start at 100, then save it in MERGE format. Next run the PSI LOAD program which presents you with the options of using XB in a 16K system configuration, in a generic configuration, or with 32K. Once you select the version which best suits your system, you are asked for the input and output file names. Then, you can customize your pre-scan for whichever version of XB you will be using. There are files for the Mechatronics XB, for Myarc's new XB, and for Craig Miller's revisions. There is also a provision for any new CALLs which may be added to XB at a future date.

Next, you can change up to 5 constants from a numerical value to special punctuation symbols to save memory space (for example changing OPEN #1:DSK1.FILE to OPEN #@:DSK1.FILE). You also have the option of deleting REMs. The program will then write a MERGE file containing the

proper pre-scan syntax. How does it work? Well, the program is written in XB, so it is a bit slow. As a test, I ran my BBS program through it and it took an hour, but the BBS is 80+ sectors long with a large number of variables. However, before using PSI, it took about 45 seconds for the BBS to start once I had told it to RUN. After using PSI, it took 6 seconds!!!! I also got about 5 sectors back.

This program really works fantastic. It is written by J Peter Hoddie who is responsible for getting DM1000 and Fast-Term to sit in GRAM1-2 of GK among other things. So, you probably think that an extremely useful program like this costs an arm and a leg? WRONG....Asgard is selling PSI for only $10!!!! That is one fantastic value. To get your copy, SMAIL Chris at TI9720 or contact an Asgard dealer. This is a MUST HAVE program for everyone.

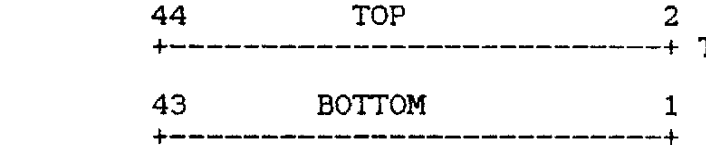
SPEECH MOD

Before I got a Triple Tech card, one thing that plagued my system was the speech synthesizer hanging on the side. Most programs would crash due to the poor connection. My solution was to just take the thing out of there and not be bothered. But some programs require that the speech box be hooked up and will not operate without it. Well, you can buy a Triple Tech card if you want the clock and buffer, but I realize that all of you are not as affluent as myself (HA). There has been much talk about putting the speech card in the P Box, but it's not as simple as just plugging in the card (although I have seen breadboard jobs done).

Here's how to put the speech synthesizer where it belongs...in the console. First, take the circuit board out of the shell and remove the big female connector. This is done by carefully prying up each contact as you hit it with a low-wattage soldering iron. Some type of solder-sucker would be very useful whether it is braid or a solder-sucker.

Next, you'll need to take your console apart so that you can mount the card on top of the motherboard case. I won't tell you how to take the console apart since you've probably done it by this time. It is pretty easy if you just pay attention. Get some thin cardboard (like what they package with shirts) and some double-stick tape so you can insulate the circuit board from the shield on the motherboard.

You'll need a piece of 8-conductor ribbon cable and another wire of equal length. Cut the length to whatever is needed to fit the speech board inside where there will be no obstructions when the case is reassembled. You will need to split the ribbon cable a couple of inches or so to do the soldering. Make sure you have neat ends on the cable....neatness counts. Now, refer to the diagram below for the pin designations on the speech board. The pins are labeled as if you were viewing the circuit board from the side, looking at where you took the connector off.



The top pins are even numbered and the

Connect the ribbon cables to the following pins: TOP 2 12 34 36 38 40 42 44 BOTTOM 1 3 5 19 35 37 39 43 Connect the odd wire to #21 which is the big pad on the bottom where there were 4 wires going to it. This is the earth ground. Now all you have to do is connect these wires to the side connector of the console in the places they would have gone if the speech box were still there. REMEMBER, neatness counts!!!!

Once you're through, use the double-stick tape to mount the circuit

board to the motherboard shield and make sure you use some kind of insulation so that it doesn't short out. Reassemble the console and give it a try. Everything should work as if the speech synthesizer were plugged in the side. NOTE: I take no responsibility for any screw-ups anybody makes doing this..it does work, but don't blame me if you fry your console from sloppy workmanship.

DISK DRIVE POWER SUPPLY

By Ken Hamai

This project requires some skills and knowledge in electronics assembly. Incorrect assembly could result in burning up your disk drive. If you are not sure how to connect the parts, contact me or somebody who can help. In any event, neither I or the ROM will cover you for any damages or losses resulting from the use of this power supply as suggested by this article and you are using it solely at your own risk.

PARTS LIST

1. Radio Shack 277-1016 power supply chassis
2. Radio Shack 273-1511 12.6volt, 3amp transformer
3. 1/2 amp fuse and holder
4. SPST bat switch
5. 5 ft. of lamp cord and plug
6. Three 6 inch lengths of 20-22ga stranded wire, different colors
7. Male plug for disk drive power connector
8. Small piece of heat shrink tubing 1/16th inch size

All of the above items except for item 7 are available at Radio Shack stores. The disk drive connector plug is available from one of our advertisers, R and D Electronic Supply, 100 E. Orangethrope Ave. Anaheim, CA 92801.

When you purchase the power supply board, you will note that it comes with instructions on a suggested wiring scheme. These instructions also recommend the use of an 18volt transformer and 2amp fuse. The reason I recommend the other transformer is because the power supply does not have to work so hard to regulate the output voltages and the lower amperage fuse gives quicker response to an overload. An added plus is that the 12.6volt transformer costs less.

Step 1. - Connect the transformer and the power supply as shown in the Radio Shack diagram that comes with the power supply.

Step 2. - Double check your connections and then plug in your supply to an outlet to test it out. Be sure to turn on the power supply switch located on the board (see fig. 3). Using a suitable volt meter and fig. 2, check that you get the indicated output voltages when you test the +5 and +12 pins and ground. The voltages MUST be pretty close. DO NOT use the power supply if you find it is off by 1/2 volt or more, especially on the +5 volt pin. If the voltages are way off, I suggest you return the board and get another one.

Step 3. - Disconnect the power to the supply and carefully bend the -5 volt pin out of the way or cut it completely off. Then solder one of the 6 inch lengths of 20-22ga wire to each of the remaining pins. Use a piece of heat shrink tubing over each soldered connection for insulation. Use yours or a friend's blow dryer at the High setting to shrink the tubing.

Step 4. - Referring to fig. 1, assemble the three wires you soldered to the pins into the power connector for the disk drive. Double check your wiring and test the connector with your voltmeter to be sure that you have the wires in the correct socket positions.

That's all there is to the wiring. If you connected up your disk drive now, it should work.

One more thing, I have not included plans for a cabinet for the disk drive and power supply. You will need to build one to hold your components together. For my demo model, I attached the transformer, fuse holder, off-on switch, and disk drive to a piece of plywood and covered the whole thing with a piece of cardboard to keep the fingers and dust out. Since various models of 5-1/4 inch floppy disk drives can be used, I suggest you take your own measurements for the design of the cabinet.

Using the above and one of the low power demand disk drives you should be able to set up a double side double density drive AND this power supply for less than $50.00, including the cardboard and nails.

LINGO

One of the more interesting aspects of my last trip to Australia (and the impending one as well) is the difference in language between the two countries. Allegedly, we both speak the same language but there are enough differneces between the Aussie lingo and our American variety to make things a bit troublesome. Everyone knows that "fair dinkum" means "genuine" in some instances but not always exactly!

"Pulling the scab off a tinny" translates into pulling the tab off a can of beer. Or, you can "plow the froth off a couple" and accomplish the very same thing. The phrase "to go walkabout" means that you are going on a journey to explore new areas. From there, it can get a bit more complicated!

"Fish and chips" are what we call fish and french fries. Potato chips (what you may have wrongly surmised from the first phrase) are "crisps". A "biscuit" is a cookie and a "scone" is a bisquit.

A "bum" is the side of your anatomy that you sit on --which gives an entirely new meaning to the American phrase "a bum' rush". Just imagine what colorful phrases can be concocted from innocent American slang!

Through it all, I'll probably have a "bonzer" time and come back with all sorts of information about new developments for the TI. After all, amongst the lot is a fair dinkum plethora of new gear. Look for a report at the March meeting!

FWEB INTERFACES

By Tony McGovern

At various times the HV99 Newsletter has contained articles on how to interface programs to the Funnelweb system. As the TI-99/4A computer contains much of its system in ROM of one persuasion or another, the Funnelweb system has to overlay this, but at the same time it unifies functions that TI assigned to seperately sold modules. Like any other substantial operating system it contains data items, memory pointers, and various utility routines, in this case held together by a reference block at the top of high memory. This has grown somewhat haphazardly over the years, but at this stage of the history of the TI-99/4A computer, plans for complete tidy up and rewrite have been shelved although almost all functional improvements and more, previously intended for that grand

revision have been incorporated in Vn 4.40 of Funnelweb. So here is the state of the art for the swansong of the HV99 group.

The information and facilities available include BLWP vectors for a variety of services, indirect BL routine pointers, and various data, pointer, and flag words and bytes. Specifications of various useful BLWP and BL routines and flags, are specified in FEWB/REPT in the Funnelweb documents. The XOP instruction is poorly supported on the 99/4A, and is left for application programs to use.

A) BLWP UTILITY ROUTINE VECTORS

(1) DSRLNK EQU >FFD4

This provides a DSRLNK function equivalent to that in the Ti-Writer Editor but corrected to be compatible with multiple RS232 cards, and with entry data recording as in the E/A DSRLNK. It is for file access only and does NOT take a following DATA 8 or DATA >A instruction. This seemed like a good idea in pre-historic times when Funnelweb was only a Ti-Writer interface and has stuck for reasons of consistency ever since. Never mind, this sort of thing happens all the time --MS-DOS smells like CP/M and Intell's 1486 microprocessor still suffers gigantic hangovers from the 8088.

After the usual setup of a PAB in VDP and PAB pointer in PAD, call DSRLNK as

```
BLWP @DSRLNK
JEQ error handler
```

Further error trapping from the PAB error bits and GPL status is up to the programmer. Funnelweb utilities regualary use this DSRLNK for subprogram access by the code sequence

```
DSRTYP  EQU  >7A
        MOV  @DSRLNK+2,R11
        INCT @DSRTYP(R11)
        BLWP @DSRLNK
        JEQ  DSERR

        DECT @DSRTYP(R11)    sucess exit

DSERR   DECT @DSRTYP(R11)    failure exit
```

The offset value DSRTYP will be maintained at this value. All exits must restore the offset value correctly.

(2) KSCANA EQU >FFD0

This is an anhanced KSCAN routine which may be used in place of the normal E/A utility. It provides some extra functions transparently setting processor status bits, which may then be tested by various conditional Jump instructions as required. There are 6 condition bits available, and the tests check these seperately.

```
BLWP @KSCANA

JNC --> <escape> --> Jump
```

JOP --> <No key> --> Jump

JNO --> <proc'd> --> Jump

JEO --> <Oldkey> --> Jump

JGT --> <Enter> --> Jump

JH -- <Redo> --> Jump

<Escape> senses both <Fctn-9> and Ctrl-C> and <proceed> checks both <FCTN- > and <CTRL-A>. Finally an unconditional JMP maybe used to loop on the keyscan after one or more other tests.

(3) FILENT EQU >FFCC

This provides a single line Editor typically used for entry of file and device names. and may be used in text mode as well from Vn 4.40 onwards. The flashing cursor has auto-repeat with acceleration, and the timing parameters are set on entry to Funnelweb. Editing keys are as described for filename entry. FILENT waits on entry until the previous key is released. Typical calling sequence is

```
BLWP  @FILENT
DATA  <scrpos>
BYTE  <init offset>
BYTE  <length>
```

The initial offset gives the initial position of the cursor in the entry window, with null string being the first position. The CPU buffer is assigned after entry to the routine. immediately following the workspace. The workspace is at >8302 in the PAD area, so the maximum buffer length can extend at most to the end of the DSR transient area. A calling sequence that allows another workspace/buffer area to be assigned is

```
      BLWP @FWFILN
      DATA <scrpos>
      DATA <ofs,len>

FILENT EQU  $
       LI   R11,FEWSP          Workpace for BLWP vector in R11
       MOV  @FILENT+2,R12      FILENT code pointer in R12
       MOV  *R14,@FILNT1       Screen pointer
       MOV  *R14,@FILNT2       Line window data
       BLWP R11                Execute F'web FILENT in FEWSP
FILNT1 DATA >0                 Data written in
FILNT2 DATA >0                 Also
       RTWP

FWFILN DATA TRANWS,FILNT       User "FILENT" BLWP vector
       DATA >0,>0,>0,>0,>0     R11-R15 of transfer workspace
TRANWS EQU  $->20
FEWSP  BSS  >20
FILBFR BSS  <as needed>
```

This is a lot simpler than writing your own routine, and may be necessary if you are already using the PAD area. It is possible to be even trickier in the transfer code but that will do.

Several auxiliary flag locations are associated with use of FILENT.

```
LOWCAS EQU  >FF22       Lowercase allowed
HEXDIG EQU  >FF24       HEX digits only
ESFLAG EQU  >FF60       <escape> flag/pointer
```

When LOWCAS is null, the normal condition, all entries are converted to upper case as in the Editor SF/LF. FILENT always resets LOWCAS to null on exit. SETO of LOWCAS before calling FILENT allows lower case alpha characters to slip through. If HEXDIG is set before entry only valid hex digits can be netered and deletes produce "0" instead of blanks. Write a valid hex entry on the screen first as this is not checked.

If ESFLAG is null the the <escape> key is ignored internally and the key value passed out. SETO of ESFLAG is reserved for internal purposes. Any other value is treated as an <escape> address pointer. First instruction in the <esc> routine should be a LWPI as the <esc> sensing occurs in the internal CURSEOR (see later) routine and will exit with the FILENT registers set.

(4) DELSPR EQU >FFC8

This one just writes byte >D0 to VDP memory at >300 to shut off the sprite list in Graphics Mode I with normal E/A table positions. It is useful in quick cleanups after switching back from Text Mode. Just call as is.

(5) VMBWD EQU >FFBA

VMBWD gives a VMBW function from in-line data, which saves program space when using fixed value data. Null length is just ignored. Call as

```
BLWP @VMBWD
DATA <scrpos>
DATA <CPU data>
DATA <length>
```

The primary Funnelweb workspace is at >FF7C and R9 of this workspace always points to VMBWD so that it may be called from this workspace as BLWP *R9 making a 4 word total call.

(6) VMBRD EQU >FFB6

This is the VDP version and is essentially similar to VMBWD.

(7) VSBRD EQU >FFB2

The calling sequence for this is

```
BLWP @VSBRD
DATA <scrpos>
```

and it returns the byte in VDP memory at <scrpos> to the MSB of R0 (not the usual R1 in E/A code).

(8) VFILL EQU >FFAE

This fills a block of VDP RAM with the byte value in the MSB of R0. The Funnelweb main workspace FWREGS EQU >FF7C maintains R8 as a pointer to

this so it may be called as BLWP *R8 etc. Standard calling sequence is

```
LI   R0,<MSB value>
BLWP @VFILL
DATA <scrpos>
DATA <length>
```

```
(9) VMBWR  EQU  >FED4

(10)VMBRR  EQU  >FED0

(11)VSBWR  EQU  >FECC

(12)VSBRR  EQU  >FEC8
```

This bunch of VDP utilities is equivalent to the standard E/A utility set in usage. The VM routines check for and ignore null length values in R2.

```
(13)VSTRW  EQU >FEC4
```

This writes the body of a string with leading length byte to VDP. Call as

```
LI   R0
BLWP @VSTRW
DATA <scrpos>
```

If the length byte of the string is null then the call is ignored.

```
(14)CURSOR  EQU  >FEC0
```

The CURSOR routine is called internally by FILENT, but may also be called externally. The calling sequence is

```
LI   R6,<scrpos>
BLWP @CURSOR
```

CURSOR flashes at the screen position specified in the calling R6 and returns the key-value in the MSB of the calling R2, and this is left in raw state at console KEYRT (>8375). See FILENT for a discussion of <escape>. Normally CURSOR provides the up-case function for FILENT, so the LOWCAS flag is effective here also. It is not cleared however except by the <escape> path. If you want to use CURSOR externally with hex digit validation, you will need to supply your own validation routine. A table of hex digits "01 . .EF" is provided at

```
HXTAB  EQU  >FEA4
```

The cursor flash rate and auto-delays are slaved indirectly to the VDP vertical interrupt rate. On entry to Funnelweb the number of console keyscans that occur in an interval of 14 vertical interrupts is counted and stored at

```
REPETS  EQU  >FEE4
```

This does not account for the difference between PAL and NTSC consoles, but this will be minor compared to possible Geneve to 99/4A differnce.

(14)DSRREN EQU >FEBC

DSRREN is intended to give a direct DSR re-entry from saved values, as is done in the E/A object loader, or more comprehensively in LINEHUNTER. It re-enters the last DSR ROM at the same entry point directly without having to search for it. All PAB information necessary must be supplied as normal. It is used in DiskReview (80-col) as the new speedup for Vn 4.40 of the Myart file viewer. It is of course not necessary to set the DSRTYP again for sub-program (eg. sector read) re-entry as this is only needed to find the entry point of the original search.
The relevent DSR values are stored in a 4 word block starting at

SAVENT EQU >FF46

on every full DSRLNK call.

(15)QCODE EQU >FEB8

This is mostly of use internally after the AID key has been detected in the path with SETO of ESFLAG (>FF60). It checks the value at AIDFL (>FF3A) and if null returns directly, else it branches to the address pointed to by AIDFL. Usually this is a QD test/load/branch routine.

(16)SETGRD EQU >FEB4

This sets the GROM address using the system GROM pointer in GPL R13, so that Module Library banking is supported. Call as

```
LI   R0,<GROM address>
BLWP @SETGRD
```

(B) INDIRECT BL ROUTINE POINTERS

All of these routines are conveniently call by

```
LI   @BLPTR,R11
BL   *R11
```

together with whatever register and/or in-line data is appropriate for the particular BL call.

(1) CFILE# EQU >FFA6

This takes the trailing DATA item >0n00 where "n" is the number of file buffers to be set aside in VDP as would be done with CALL FILES(n) from BASIC. The routine is compatible with V-9938 based systems, and rewrites MAXMEM (>8370) TO SUIT. A new VDP header is written for Ti/CorComp disk controllers and VDP cleared above that to the top of the disk DSR area. No range checking is done on "n".

(2) KEY EQU >FFA4

This keyloop enables interrupts, follows the previous prescription for <escape> and ESFLAG (>FF60), and calls QDCODE if AID is pressed. The key return is placed in the MSB of R2 and left unchanged at KEYRT (>8375).

(3) RESTR EQU >FFA2

This does various housekeeping tasks to set up the E/A Graphics Mode I for the Funnelweb Central Menu screen. It resets the VDP registers to E/A default, rewrites the color table, clears the screen, calls DELSPR, restores the CHR(>81) pattern to underline, and restores the key-unit to 5.

(4) RDDEV EQU >FFA0

The RDDEV routine handles the details after screen entry of a file or device name, saving the device name to CPU RAM, and building a PAB in VDP for DSRLNK. RDDEV is called externally as

```
MOV  @RDDEV,R11          Get pointer
BL   *R11                 Go to routine
DATA <screen ptr>         Entry pointer on screen
DATA <cpu ram bufr>       CPU RAM buffer for name
DATA <vdp PAB adr>        VDP address for PAB
DATA <cpu PAB data>       PAB data storage address
```

The first data item is the VDP pointer for the filename as entered on screen. RDDEV parses this up to the first space and enters it with leading length byte at the CPU RAM buffer pointed to by the second data item. It then builds a PAB in VDP at the address in the third data item. The CPU RAM buffer may well be a continuation of the PAB data, but need not be. While at it, RDDEV clears the GPL status byte (>837C) and loads SCNAME (>8356) for DSRLNK.

(5) QDLOAD EQU >FF3C Uses R0, R1, R2, R10, R12

System files with 2 character filenames in E/A program file format may be loaded without executing by QDLOAD. A typical calling sequence is

```
MOV  @QDLOAD,R11          Get pointer
BL   *R11                 Execute
DATA 'UL'                 2-char filename
DATA ULPOS                Load position in CPU
DATA ULLEN                Load length
JMP  <error return>
```

Successful loads step over the error branch. The normal E/A header is ignored in favor of the load target position and length. The maximum file length allowed in >1100 which is long enough for QD or QF. The file is loaded from the Funnelweb boot path as for other system files.

(4) FILCLN EQU >FEDC Uses R0, R1, R10, R14

This routine cleans up the mailbox (>A000 - >A050) area, for internal system use and may be subject to change in function or auxiliary conditions. Some not already mentioned which may be useful are

```
NAMBUF  EQU  >FF62
```

The space from NAMBUF to FWREGS (>FF7C) is used for a filename buffer by the various loaders. It is long enough to accomodate length byte and 25 character names (DSK.volname.filename or whatever else fits). Increase in this was the one change that would have forced major incompatibilities with previous versions, and so it has been left as is for better or for worse.

BTLN EQU >FF58

This points to the length byte of the boot pathname and all references to system filenames are by offsets from this value. The Editor printer device name, as a survival from much earlier versions, though part of this block, has its own pointer at EDPRNT EQU >FF1E. All these path/file and device names are stored with leading length byte, unlike the workfile name in the mailbox at >A000 which is stored without length byte as required by the Editor.

INCOL EQU >FF26

The word value here is an index range 0-9 into the 10 byte table of color bytes for use in VReg #7, and pointed to by COLRS EQU >FF1C. Standard Funnelweb system practice is always to use this table as the source of color combinations, and to use INCOL as the index. this helps avoid unexpected and jarring color changes on return.

CPUDEL EQU >FEE2

The value here estimates CPU speed as seen from normal expansion RAM by counting a loop against the VDP interrupt timer. It is handy as a value fo CPU delay timing of beeps and bloops. Use REPETS (see earlier) for keyloop timing.

```
100 CALL CLEAR :: PRINT "
      'CRUNCH'" :: PRINT ::
 PRINT "       by Tim MacEac
hern" :: PRINT "       PO Bo
x 1105" :: PRINT "       Dar
tmouth, NS"
110 PRINT "       B2Y 4B8 Ca
nada" :: PRINT :: PRINT :: P
RINT :: PRINT :: FOR DELAY=1
 TO 1000
120 NEXT DELAY :: CALL CLEAR
 :: PRINT :: PRINT " Purpose
:" :: PRINT "Shorten Basic &
 XBASIC" :: PRINT "programs
by substituting" :: PRINT "s
ingle character variable" ::
 PRINT "names for longer nam
es and"
130 PRINT "frequently used c
onstants." :: PRINT :: PRINT
 " To run:" :: PRINT "(1) En
sure that your program" :: P
RINT "does not have lines nu
mbered" :: PRINT "1 to 10 (o
r so). Statements"
140 PRINT "at these lines ar
e generated" :: PRINT "by CR
UNCH if you move a" :: PRINT
 "constant into a variable."
 :: PRINT "(2) Save your pro
gram in" :: PRINT "merge for
mat to disk."
150 PRINT "(3) Run CRUNCH, p
roducing a"
160 PRINT "new merge format
file."
170 PRINT "(4) Enter NEW in
XBASIC and" :: PRINT "merge
in the new program." :: PRIN
T :: FOR DELAY=1 TO 3000
180 NEXT DELAY :: CALL CLEAR
 :: DEF PC(X)=((X=36)+(X=34)
-(X>47)*(X<58)-(X>64)*(X<91)
)<>0

190 !DEF BL$(Z$)=SEG$(" ",1,
PC(ASC(SEG$(L$,LEN(L$),1)))*
PC(ASC(Z$)))&Z$
200 DEF BL$(Z$)=Z$ :: DEF YE
S=POS(ANSWER$&"Y","Y",1)=1 :
: DEF NO=YES=0 :: OPTION BAS
E 1 :: DIM TABLE$(2,500),TAB
LE(3,500):: PHASE=1 :: INPUT
 "ENTER FILENAME : ":FN$
210 IF POS(FN$,".",1)<>0 THE
N 230
220 FN$="DSK1."&FN$
230 IF PHASE=1 THEN NTAB=0
240 DIM TOKEN$(255):: FOR I=
255 TO 255 :: TOKEN$(I)="$"&
STR$(I)&"$"
250 NEXT I :: FOR J=256 TO 2
56 :: READ I,S$ :: IF I<0 TH
EN 280
260 TOKEN$(I)=S$
270 NEXT J
280 DEF CHAR(J)=ASC(SEG$(S$,
J,1)):: CHE$=CHR$(255)&CHR$(
255)
290 OPEN #1:FN$,VARIABLE 163
,INPUT ,DISPLAY :: IF PHASE=
1 THEN 380
300 PRINT :"TOTAL SAVINGS AC
HIEVED WILL BE";DOSAVE;"BYTE
S." :: INPUT "ENTER NEW FILE
NAME FOR MERGEFILE: ":FN2$ :
: IF POS(FN2$,".",1)=0 THEN
FN2$="DSK1."&FN2$
310 OPEN #2:FN2$,VARIABLE 16
3,OUTPUT,DISPLAY

320 ! SET CONSTANT VARS
330 FOR I=1 TO NTAB :: IF TA
BLE$(2,I)="" OR TABLE$(1,I)=
"0" OR TABLE$(1,I)=(CHR$(34)
&CHR$(34))THEN 350
340 IF (ASC(TABLE$(1,I))>=AS
C("0")AND ASC(TABLE$(1,I))(=
ASC("9"))OR ASC(TABLE$(1,I))
=ASC(".")OR ASC(TABLE$(1,I))
=34 THEN GOSUB 360
350 NEXT I :: GOTO 380
360 IXX=IXX+1 :: IF POS(TABL
E$(1,I),CHR$(34),1)=0 THEN T
$=TABLE$(1,I)ELSE T$=SEG$(TA
BLE$(1,I),2,LEN(TABLE$(1,I))
-2)
370 S2$=CHR$(INT(IXX/256))&C
HR$(IXX-256*INT(IXX/256))&TA
BLE$(2,I)&CHR$(190)&CHR$(200
+(T$<>TABLE$(1,I)))&CHR$(LEN
(T$))&T$&CHR$(0):: PRINT S2$
 :: PRINT #2:S2$ :: RETURN
380 FOR I=1 TO 99999 :: LINP
UT #1:S$ :: S2$=S$ :: IF S$=
CHE$ THEN 650

390 !L$=STR$(ASC(S$)*256+ASC
(SEG$(S$,2,1)))
400 COMMAND=ASC(SEG$(S$,3,1)
):: NOTDATA=COMMAND<>147 AND
 COMMAND<>158 AND COMMAND<>1
54 AND COMMAND<>131 :: FOR J
=3 TO LEN(S$)-1 :: C=ASC(SEG
$(S$,J,1)):: IF (C=200)+(C=1
99)THEN 560
410 IF C<>201 THEN 440

420 !L$=L$&BL$(STR$(CHAR(J+1
)*256+CHAR(J+2)))
430 J=J+2 :: GOTO 630
440 IF C>127 THEN 520
450 FOR JJ=J+1 TO 999 :: C=A
SC(SEG$(S$,JJ,1)):: IF (C>12
7)+(C=0)THEN 470
460 NEXT JJ
470 T$=SEG$(S$,J,JJ-J)

480 !L$=L$&BL$(T$)
490 IF NOT NOTDATA THEN 510
500 GOSUB 1070 :: IF PHASE=2
 AND TABLE$(2,P)<>"" THEN S2
$=SEG$(S2$,1,J-1+LEN(S2$)-LE
N(S$))&TABLE$(2,P)&SEG$(S2$,
JJ+LEN(S2$)-LEN(S$),999)
510 J=JJ-1 :: GOTO 630

520 REM TOKENS
530 !L$=L$&BL$(TOKEN$(C))
540 IF C=130 THEN COMMAND=AS
C(SEG$(S$,J+1,1))
550 NOTDATA=COMMAND<>147 AND
 COMMAND<>158 AND COMMAND<>1
54 AND COMMAND<>131 :: GOTO
630
560 LN=ASC(SEG$(S$,J+1,1)):
 T$="" :: IF C<>199 THEN 580
570 T$=CHR$(34)
580 T$=T$&SEG$(S$,J+2,LN)&T$

590 !L$=L$&BL$(T$)
600 IF NOT NOTDATA OR COMMAN
D=138 OR ASC(T$)>=ASC("A")TH
EN 620
610 GOSUB 1070 :: IF PHASE=2
 AND TABLE$(2,P)<>"" THEN S2
$=SEG$(S2$,1,J-1+LEN(S2$)-LE
N(S$))&TABLE$(2,P)&SEG$(S2$,
J+LN+2+LEN(S2$)-LEN(S$),999)
620 J=J+LN+1

630 NEXT J :: PRINT S$ :: IF
 PHASE=2 THEN PRINT #2:S2$ :
: PRINT S2$
640 NEXT I
650 CLOSE #1 :: IF PHASE=2 T
HEN PRINT #2:CHE$ :: CLOSE #
2 :: PRINT "FINISHED, SAVING
";DOSAVE;"BYTES" :: STOP

660 REM PRINT STRING TABLE
670 INPUT "LISTING VARIABLES
 AND       CONSTANTS. ENTER
MINIMUM    OCCURENCE COUNT T
O LIST :":MC :: PRINT : :"ST
RING OCCURENCES" :: FOR I=1
TO NTAB :: IF TABLE(1,I)<MC
THEN 690
680 PRINT TABLE(1,I);TAB(8);
TABLE$(1,I)
690 NEXT I :: INPUT "PRESS E
NTER TO CONTINUE ":T$
700 PRINT "SINGLE CHARACTER
VARIABLE   NAMES NOT YET IN
USE": : :: SN$="" :: S$="" :
: FOR II=1 TO 2 :: FOR I=ASC
("A")TO ASC("Z")):: T$=CHR$(I
)&S$ :: FOR J=1 TO NTAB :: I
F T$=TABLE$(1,J)THEN 720
710 NEXT J :: PRINT T$;" ";:
: SN$=SN$&CHR$(I)
720 NEXT I :: S$="$" :: IF I
I=1 THEN NN$=SN$ :: SN$=""
730 NEXT II :: PRINT :: PRIN
T "DO YOU WANT TO MODIFY THE
" :: INPUT "PROGRAM? ":ANSWE
R$ :: IF NO THEN RUN "DSK1.L
OAD"

740 REM !!!SHORTEN PROGRAM !
!!
750 REM PRINT POTENTIAL SAVI
NGS
760 PRINT "POTENTIAL SAVINGS
" :: PRINT :: TOTSAV=0 :: FO
R I=1 TO NTAB :: GOSUB 910
770 IF SAV>0 THEN PRINT "CHA
NGING ";TABLE$(1,I);" SAVES"
;SAV;"BYTES"
780 TOTSAV=TOTSAV+SAV
```

```
790 NEXT I :: PRINT :"MAXIMU
M TOTAL SAVINGS THAT  YOU CA
N MAKE BY RENAMING    VARIAB
LES AND CONSTANTS IS  ";TOTS
AV;"BYTES." :: INPUT "CONTIN
UE? ":ANSWER$ :: IF NO THEN
RUN "DSK4.FWEB"
800 PRINT :"SELECTING ELEMEN
TS TO RECODE": :: DOSAVE=0 :
: FOR I3=1 TO NTAB :: J=0 ::
 BEST=0 :: FOR I=1 TO NTAB :
: GOSUB 910 :: IF SAV>BEST T
HEN J=I :: BEST=SAV
810 NEXT I :: IF J=0 THEN PH
ASE=2 :: GOTO 290
820 TABLE(1,J)=0 :: NUMVER=P
OS(TABLE$(1,J),"$",1)=0 AND
POS(TABLE$(1,J),CHR$(34),1)=
0 :: IF NUMVER AND LEN(NN$)=
0 THEN 900
830 IF NOT NUMVER AND LEN(SN
$)=0 THEN 900
840 PRINT "CHANGING ";TABLE$
(1,J);" SAVES";BEST;"BYTES"
:: INPUT "DO YOU WANT TO DO
IT? ":ANSWER$ :: IF YES THEN
 870
850 INPUT "DO YOU WANT TO DO
 MORE? ":ANSWER$ :: IF YES T
HEN 900
860 PHASE=2 :: GOTO 290
870 PRINT "SELECT FROM THE A
VAILABLE   LETTERS:" :: DOSA
VE=DOSAVE+BEST :: IF NUMVER
THEN PRINT NN$ :: AV$=NN$ EL
SE PRINT SN$ :: AV$=SN$
880 PRINT "ENTER SELECTION:
" :: ACCEPT AT(23,19)VALIDAT
E(AV$)SIZE(1):TABLE$(2,J)::
AV$=SEG$(AV$,1,POS(AV$,TABLE
$(2,J),1)-1)&SEG$(AV$,POS(AV
$,TABLE$(2,J),1)+1,999)
890 IF NUMVER THEN NN$=AV$ E
LSE SN$=AV$ :: TABLE$(2,J)=T
ABLE$(2,J)&"$"
900 NEXT I3 :: PHASE=2 :: GO
TO 290
910 IF SEG$(TABLE$(1,I),1,1)
>="A" THEN 990

920 REM CONSTANT
930 IF ASC(TABLE$(1,I))=34 T
HEN 970
940 REM NUMERIC CONSTANT
950 SAV=(LEN(TABLE$(1,I))+1)
$(TABLE(1,I)-1)-8-14-10$(TAB
LE$(1,I)="0"):: SAV=MAX(0,SA
V):: IF SAV<=0 THEN 1060
960 GOTO 1060

970 REM STRING CONSTANT
980 SAV=MAX(0,(LEN(TABLE$(1,
I))-2)$(TABLE(1,I)-1)-9-8-12
$(TABLE$(1,I)="")):: GOTO 10
60

990 REM VARIABLE
1000 IF SEG$(TABLE$(1,I),LEN
(TABLE$(1,I)),1)="$" THEN 10
40

1010 REM NUMERIC VARIABLE
1020 SAV=MAX(0,(LEN(TABLE$(1
,I))-1)$TABLE(1,I)):: IF SAV
<=0 THEN 1060
1030 GOTO 1060

1040 REM STRING VARIABLE
1050 SAV=MAX(0,(LEN(TABLE$(1
,I))-2)$TABLE(1,I)):: IF SAV
<=0 THEN 1060
1060 RETURN

1070 REM TABULATE STRINGS
1080 GOSUB 1090 :: GOSUB 117
0 :: RETURN
1090 P=ROOT
1100 IF P<>0 THEN 1140
1110 ROOT=1
1120 NTAB=NTAB+1 :: TABLE$(1
,NTAB)=T$ :: IF P<>0 THEN TA
BLE(3+(TABLE$(1,P)>T$),P)=NT
AB
1130 P=NTAB :: RETURN
1140 IF TABLE$(1,P)=T$ THEN
RETURN
1150 IF TABLE(3+(TABLE$(1,P)
>T$),P)=0 THEN 1120
1160 P=TABLE(3+(TABLE$(1,P)>
T$),P):: GOTO 1100

1170 REM
1180 TABLE(1,P)=TABLE(1,P)+1
 :: RETURN
1190 DATA 129,ELSE
1200 DATA 130,"::"
1210 DATA 131,"!"
1220 DATA 132,IF
1230 DATA 133,GO
1240 DATA 134,GOTO
1250 DATA 135,GOSUB
1260 DATA 136,RETURN
1270 DATA 137,DEF
1280 DATA 138,DIM
1290 DATA 139,END
1300 DATA 140,FOR
1310 DATA 141,LET
1320 DATA 142,BREAK
1330 DATA 143,UNBREAK
1340 DATA 144,TRACE
1350 DATA 145,UNTRACE
1360 DATA 146,INPUT
1370 DATA 147,DATA
1380 DATA 148,RESTORE
1390 DATA 149,RANDOMIZE
1400 DATA 150,NEXT
1410 DATA 151,READ
1420 DATA 152,STOP
1430 DATA 153,DELETE
1440 DATA 154,REM
1450 DATA 155,ON
1460 DATA 156,PRINT
1470 DATA 157,CALL
1480 DATA 158,OPTION
1490 DATA 159,OPEN
1500 DATA 160,CLOSE
1510 DATA 161,SUB
1520 DATA 162,DISPLAY
1530 DATA 163,IMAGE
1540 DATA 164,ACCEPT
1550 DATA 165,ERROR
1560 DATA 166,WARNING
1570 DATA 167,SUBEXIT
1580 DATA 168,SUBEND
1590 DATA 169,RUN
1600 DATA 170,LINPUT
1610 DATA 176,THEN
1620 DATA 177,TO
1630 DATA 178,STEP
1640 DATA 179,","
1650 DATA 180,";"
1660 DATA 181,":"
1670 DATA 182,")"
1680 DATA 183,(
1690 DATA 184,&
1700 DATA 186,OR
1710 DATA 187,AND
1720 DATA 188,XOR
1730 DATA 189,NOT
1740 DATA 190,=
1750 DATA 191,<
1760 DATA 192,>
1770 DATA 193,+
1780 DATA 194,-
1790 DATA 195,*
1800 DATA 196,/
1810 DATA 197,^
1820 DATA 202,EOF
1830 DATA 203,ABS
1840 DATA 204,ATN
1850 DATA 205,COS
1860 DATA 206,EXP
1870 DATA 207,INT
1880 DATA 208,LOG
1890 DATA 209,SGN
1900 DATA 210,SIN
1910 DATA 211,SQR
1920 DATA 212,TAN
1930 DATA 213,LEN
1940 DATA 214,CHR$
1950 DATA 215,RND
1960 DATA 216,SEG$
1970 DATA 217,POS
1980 DATA 218,VAL
1990 DATA 219,STR$
2000 DATA 220,ASC
2010 DATA 221,PI
2020 DATA 222,REC
2030 DATA 223,MAX
2040 DATA 224,MIN
2050 DATA 225,RPT$
2060 DATA 232,NUMERIC
2070 DATA 233,DIGIT
2080 DATA 234,UALPHA
2090 DATA 235,SIZE
2100 DATA 236,ALL
2110 DATA 237,USING
2120 DATA 238,BEEP
2130 DATA 239,ERASE
2140 DATA 240,AT
2150 DATA 241,BASE
2160 DATA 243,VARIABLE
2170 DATA 244,RELATIVE
2180 DATA 245,INTERNAL
2190 DATA 246,SEQUENTIAL
2200 DATA 247,OUTPUT
2210 DATA 248,UPDATE
2220 DATA 249,APPEND
2230 DATA 250,FIXED
2240 DATA 251,PERMANENT
2250 DATA 252,TAB
2260 DATA 253,#
2270 DATA 254,VALIDATE
2280 DATA -1,-1
```