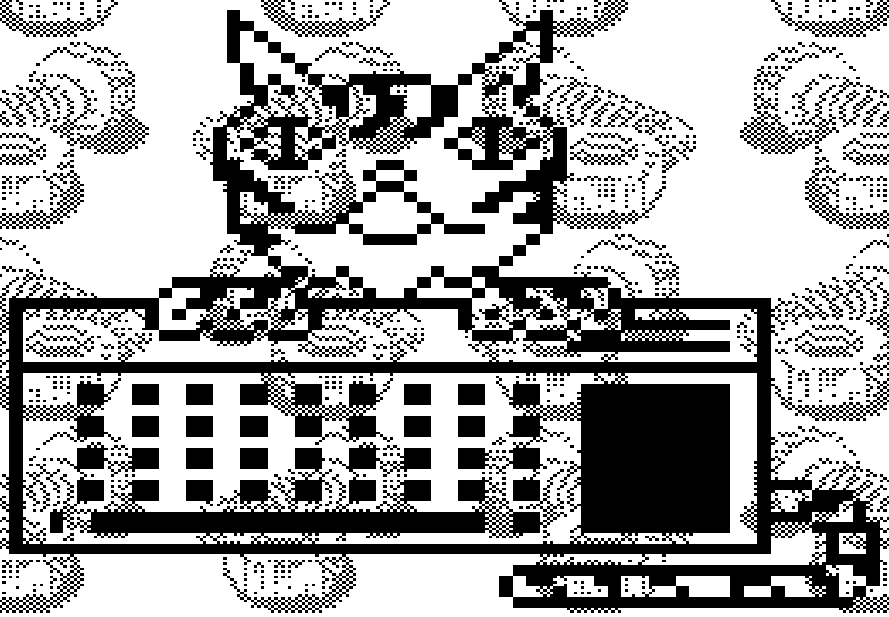
NUTS & BOLTS
NUMBER 2
DOCUMENTATION
1985



TIGERCUB
SOFTWARE
JIM PETERSON

NUTS BOLTS No. 2

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus, OH 43213

However, bona-fide purchasers of this disK are authorized without restriction to duplicate the individual subprograms contained on this disK for the purpose of incorporating them into their programs.

This disK contains 108 utility programs which are saved in MERGE format so that you can incorporate them into your programs by simply typing MERGE DSK1. (and the program name). For a catalog of them, type RUN "DSK1.CAT" . Almost all have line numbers running from 20710 to 21700 so that they will not overprint any of your program lines, and the line numbers are consecutive so that any number of them may be merged together, or with the subprograms on the NUTS & BOLTS DISK (No. 1).

All but a few are in the format of subprograms, so that any values assigned to variable names within them will not interfere with variables of the same name in the body of your program, unless they are passed in the parameter list. A few of them contain a CALL LOAD or CALL PEEK, and therefore require the Memory Expansion. Some of them contain DATA statements, so any DATA being read from or by your main program should first be RESTOREd - which is good programming practice, anyway.

A NUTS & BOLTS No. 1 disK is also available for $19.95 from Tigercub Software, containing another 100 utility subprograms plus a tutorial on using subprograms.

>>>>>> CHARACTER SETS <<<<<<

The subprograms HEAVYCHAR, GREEK, SMALLNUM, SCRIPT, SCRIPT2 and (on the first NUTS & BOLTS disK) CHARFACE, LARGECHAR, LOWERCASE, SLANT, and RUSSIAN all redefine characters from DATA statements, therefore overwrite any previos redefinitions. However, the subprograms BACKWARD, BIGLETTER, BIGSPRITER, HIGHCHAR, LINETEXT, SIDEWAYS, STENCIL, TALLCHAR, TALLTHIN,UNDERLINE, WIDECHAR and (on first NUTS & BOLTS disK) BIGCHAR, BIGCHAR2, MONGOLIAN and UPSIDEDOWN, internally disassemble and reassemble each character, and so can be used (except MONGOLIAN) to modify those in the previous paragraph (for instance, CALL SCRIPT :: CALL TALLCHAR) or to modify each other (CALL HEAVYCHAR :: CALL WIDECHAR). The resulting new character sets can then be saved as MERGE format DATA lines by the CHARKEEP subprogram, merged into any program, and recreated by RESTORE :: FOR CH=... TO ... :: READ CH$ :: CALL CHAR(CH,CH$):: NEXT CH

BACKWARD

CALL BACKWARD(A,B,C) will reidentify characters, starting with ASCII C, to the reversed form of ASCII A through B. Example:

```
100 CALL CLEAR
110 FOR CH=33 TO 126 :: PRINT CHR$(CH);:: NEXT CH :: CALL BACKWARD(33,126,33):: GOTO 110
```

BIGLETTER

CALL BIGLETTER(M$,R,C) will display a string M$ of up to 13 enlarged characters (14 if one of them is a space), of any character below ASCII 92, at row R (1 to 23), column C. Example:

```
100 CALL CLEAR :: INPUT "M$,R,C ":M$,R,C :: CALL BIGLETTER(M$,R,C)
110 GOTO 110
```

For a faster display, use SCREENSAVE to copy the resulting screen.

BIGSPRITER

CALL BIGSPRITER(F$,LN,A,B) will redefine ASCII characters A to B to 4 times normal size, to be used as a MAGNIFY(4) sprite, and will save their 64-byte hex code in a DATA statement, in line numbers starting with LN and incremented by 1, in a MERGE format file F$ which can then be used as described under MAGCHAR. Note that the characters may be previously redefined, so that you may create giant sprites of the GreeK letters or other special characters. Example:

```
100 CALL BIGSPRITER("DSK1.BIGSPRITE",48,48,57)
```

CHARKEEP

CALL CHARKEEP(F,F$,L,A,B) will open a file #F as disK/filename F$ and write a MERGE format program of DATA statements beginning with line number L containing the hex codes for ASCII A to B. This subprogram may be used after any subprogram which internally redefines characters, such as BACKWARD, to create DATA lines which can be MERGEd back in. Then both subprograms can be deleted and the characters redefined more rapidly by READ and CALL CHAR.

GREEK

CALL GREEK(C,CC) will redefine ASCII characters C to CC to the letters of the GreeK alphabet. Example:

```
100 CALL CLEAR :: FOR CH=65 TO 90 :: PRINT CHR$(CH);:: NEXT CH :: CALL GREEK(65,90)
110 GOTO 110
```

HEAVYCHAR

CALL HEAVYCHAR will redefine the numerals and upper case letters to a heavier form. Example:

```
100 CALL CLEAR :: FOR CH=48 TO 90 :: PRINT CHR$(CH);:: NEXT CH :: CALL HEAVYCHAR
110 GOTO 110
```

HIGHCHAR

CALL HIGHCHAR maKes all characters one pixel-row taller, quite rapidly. Example:

```
100 CALL CLEAR :: FOR CH=33 TO 126 :: PRINT CHR$(CH);:: NEXT CH :: CALL HIGHCHAR
110 GOTO 110
```

LINETEXT

CALL LINETEXT will reduce the size of numerals and put the numerals and lower case letters between two horizontal lines. Example:

```
100 CALL CLEAR :: DISPLAY AT(12,1):"abcdefghijKlmnopqrstuvwxyz" :: DISPLAY AT(14,9):"0123456789" :: CALL LINETEXT
110 GOTO 110
```

MAGCHAR

MAGCHAR is not a subprogram but a MERGE format program consisting of DATA statements containing the 64-byte hex codes for ASCII codes 32 to 90. It must be MERGEd into the BIGBANNER subprogram, and may be MERGEd into any other program which requires letters in the form of MAGNIFY(4) sprites. However, it takes 4307 bytes of memory. Therefore recommend that you clear

memory with NEW, MERGE DSK1.MAGCHAR, RES 32,1 to make the line numbers correspond to their ASCII characters, SAVE DSK1.MAGCHAR,MERGE and then MERGE it into the program and delete the line numbers of any unneeded characters.
Example:
```
100 CALL CLEAR :: CALL MAGNIFY(4):: FOR ch=32 TO 90 :: READ CH$ :: CALL CHAR(100,CH$):: CALL SPRITE(#1,100,2,100,100):: NEXT CH
```

SCRIPT

CALL SCRIPT will convert both upper and lower case letters into a modified script in which the lower case letters f, g, j, p, q, y and z do not have "tails" but are raised above the line in a modified form so that they can be displayed in a single print space.
Example:
```
100 CALL CLEAR :: FOR CH=65 TO 122 :: PRINT CHR$(CH);:: NEXT CH :: CALL SCRIPT
110 GOTO 110
```

SCRIPT2

CALL SCRIPT2(R,M$) will convert both upper and lower case letters to a true script form, with descenders, and display text M$ on alternate lines beginning at row R. This subprogram CALLs the SCRIPT subprogram, which must also be merged in.
Example:
```
100 CALL CLEAR :: M$="This is a demonstration of the SCRIPT2 subprogram from the Tigercub Nuts & Bolts #2 DisK" :: CALL SCRIPT2(5,M$)
110 GOTO 110
```

SIDEWAYS

CALL SIDEWAYS(A,B) will slowly redefine characters ASCII A to B by rotating them one quarter to the left. Example:
```
100 CALL CLEAR :: FOR CH=33 TO 126 :: PRINT CHR$(CH);:: NEXT CH :: CALL SIDEWAYS(33,126)
```
This takes 13 minutes for ASCII 33-126. For much faster initialization, but requiring more memory, type NEW, then MERGE DSK1.SIDEWAYS, then MERGE DSK1.CHARKEEP . Then enter
```
100 A=(first char desired):: B=(last char) :: CALL SIDEWAYS(A,B):: CALL CHARKEEP(1,"DSK1.SIDEKEEP",30000,A,B).
```
Place a disK with available space in drive 1, and RUN. The resulting program SIDEKEEP can be MERGEd into any program and characters defined quicKly by RESTORE ... FOR CH=... TO ... :: READ CH$ ...CALL CHAR(CH,CH$):: NEXT CH

SMALLNUM

CALL SMALLNUM reduces the numerals to the same size as the lower case letters.
Example:
```
100 CALL CLEAR :: PRINT "1234567890abc" :: CALL SMALLNUM
110 GOTO 110
```

STENCIL

CALL STENCIL(CH,K,S,C,R,CC) will display a character in stencil form as a MAGNIFY(4) sprite. CH is the ASCII of the character to be displayed, K is the ASCII to be redefined (it must be divisible by 4, and higher than any ASCII to be displayed), S is the sprite number, C is the sprite color, R and CC are the sprite row and column.
Example:
```
100 CALL CLEAR :: R=1 :: C=10 :: M$="TIGERCUB" :: K=96 :: FOR J=1 TO 8 :: CALL STENCIL(ASC(SEG$(M$,J,1)),K,J,16,R,C):: K=K+4 :: R=R+8 :: C=C+25 :: NEXT J
110 GOTO 110
```

TALLCHAR

CALL TALLCHAR(R,C,M$) converts ASCII 48-133 into double-height characters of ASCII 48-90, rather slowly, and prints a string (M$) of up to 28 characters at row (R), column (C). Subsequent CALLs display strings immediately. If punctuation is needed, predefine unused characters between 48 and 90 (do not use 60). For instance - CALL CHARPAT(43,CH$):: CALL CHAR(62,CH$) and then use > in the string when you need +. Example:
```
100 CALL CLEAR :: DISPLAY AT(23,1):"WAIT PLEASE"
110 DISPLAY AT(24,1):CHR$(130)&CHR$(108)&CHR$(116)&CHR$(127)&CHR$(32)&CHR$(123)&CHR$(119)&CHR$(112)&CHR$(108)&CHR$(126)&CHR$(112)
120 DATA THIS IS A DEMONSTRATION,OF THE TALLCHAR PROGRAM,FROM THE TIGERCUB SOFTWARE, DISK CALLED NUTS AND BOLTS,NUMBER TWO
130 DATA IT SETS UP SLOWLY,BUT PRINTS QUITE RAPIDLY>>
140 CALL CHARPAT(33,CH$):: CALL CHAR(62,CH$)
150 FOR R=2 TO 14 STEP 2 :: READ M$ :: CALL TALLCHAR(R,1,M$):: NEXT R
160 GOTO 160
```
For a faster display, use TALLPRINT or SCREENSAVE.

TALLPRINT

CALL TALLPRINT(R,C,M$) will display text M$ at row R, column C in the double-height characters created by TALLCHAR but much more quicKly (but using more memory). Type NEW, then MERGE DSK1.TALLCHAR, then MERGE DSK1.CHARKEEP. Type in
```
100 CALL TALLCHAR(1,1," "):: CALL CHARKEEP(1,"DSK1.TALLKEEP",21688,48,133)
```
Place a disK with sectors available in drive 1, RUN. Put Nuts&Bolts disK in drive 1, type NEW, then MERGE DSK1.TALLPRINT. Put the other disK backK in, type MERGE DSK1.TALLKEEP, then SAVE DSK1.TALLPRINT,MERGE. You now have the TALLPRINT subprogram ready to MERGE into any program.

TALLTHIN

CALL TALLTHIN(M$,X,C) will display text M$, of not more than 8 letters, starting at column C and in row X downward, in characters 8 spaces tall but only one space wide
Example:
```
100 CALL CLEAR :: CALL TALLTHIN("TIGERCUB",5,8)
110 GOTO 110
```

UNDERLINE

CALL UNDERLINE(F,T,C) will redefine characters from ASCII F to ASCII T into an underlined form, and place the redefinitions in characters starting with ASCII C. Example:
```
100 CALL CLEAR
110 CALL UNDERLINE(65,90,97) :: PRINT "PRESS enter WHEN READY"
120 CALL UNDERLINE(33,64,33) :: FOR CH=33 TO 64 :: PRINT CHR$(CH);:: NEXT CH
130 GOTO 130
```

WIDECHAR

CALL WIDECHAR(M$,R,C) will slowly redefine each character from ASCII 48 to 90 into a double-width character consisting of the ASCII of the character and an ASCII 43 higher, to be printed in 2 spaces. If punctuation below ASCII 48 will be needed, it may be predefined into unused characters.
The text M$ of up to 14 characters will then be displayed at row R, column C. Subsequent CALLs will display text immediately.
Example:
```
100 DATA TIGERCUB,4,7
110 DATA SOFTWARE,7,7,WIDECHAR,10,7,SUBPROGRAM,13,5
120 CALL CLEAR :: DISPLAY AT(4,7):"WAIT PLEASE"
130 READ M$,R,C :: CALL WIDECHAR(M$,R,C):: RESTORE 110 :: FOR J=1 TO 3 :: READ M$,R,C :: CALL WIDECHAR(M$,R,C):: NEXT J
140 GOTO 140
```
For a faster display, use WIDEPRINT or SCREENSAVE.

WIDEPRINT

CALL WIDEPRINT(M$,R,C) will display text M$ at row R, column C in the double-width letters created by WIDECHAR, but much more quickly. You must first type NEW, then MERGE DSK1.WIDECHAR, then MERGE DSK1.CHARKEEP. Type in
```
100 CALL WIDECHAR(" ",1,1)::
 CALL CHARKEEP(1,"DSK1.WIDEK
EEP",21665,48,133)
```
Put a disk with available space in drive 1, RUN. Then put Nuts&Bolts disk in drive 1, type NEW, then MERGE DSK1.WIDEPRINT, then put in the other disk and MERGE DSK1.WIDEKEEP, then SAVE DSK1.WIDEPRINT,MERGE . You now have the WIDEPRINT subprogram ready to merge into any program.

>>>>>>>>> DISPLAYS <<<<<<<<<

ALPHACHECK

CALL ALPHACHECK(Q$) will display at (23,1) "ARE YOU READY?" and if Q$="D" will respond to a lower case "y" by displaying at (24,1) "PUSH THE ALPHA LOCK DOWN!" or if Q$="U" will respond to upper case "Y" by "UNLOCK THE ALPHA LOCK!"
```
100 CALL CLEAR :: PRINT "TRY
 IT WITH ALPHA LOCK UP": : :
 :: CALL ALPHACHECK("D")
110 PRINT "NOW TRY WITH ALPH
A LOCK DOWN": : : :: CALL AL
PHACHECK("U"):: END
```

BACKGROUND

CALL BACKGROUND will fill the screen with a randomly patterned and colored background, change the upper case letters to white on the background color, read up to 10 DATA strings, each up to 28 characters long, from the main program and display them, properly centered, on alternate lines against the background. It then displays a flashing PRESS ANY KEY at the bottom, and wipes the screen with the pattern when any Key is pressed. The last DATA item for each screen of text must be END. Example:
```
100 CALL CLEAR :: RESTORE 12
0
110 CALL BACKGROUND :: RESTO
RE 130 :: CALL BACKGROUND ::
 CALL CHARSET :: CALL CLEAR
:: PRINT "CONTINUE PROGRAM"
120 GOTO 120
130 DATA THIS IS A DEMONSTRA
TION,OF THE SUBPROGRAM CALLE
D,BACKTEXT,IN THE,NUTS & BOL
TS #2 DISK,OF 100 UTILITY SU
BPROGRAMS,FROM,TIGERCUB SOFT
WARE,END
140 DATA IT PLACES A RANDOM
PATTERN,ON THE SCREEN,RANDOM
LY COLORED,CHANGES THE CHARA
CTERS,TO WHITE ON THE SAME,B
ACKGROUND COLOR
150 DATA READS UP TO 10 LINE
S,OF DATA AND DISPLAYS THEM,
PROPERLY CENTERED,ON ALTERNA
TE LINES,END
```

BIGBANNER

CALL BIGBANNER(M$) will scroll the text of M$ across a window in giant letters in the form of MAGNIFY(4) sprites. The MERGE format file MAGCHAR must be MERGEd into this subprogram (it is sequenced 21487-21545 to fit between lines 21486-21546 of this subprogram). Example:
```
100 CALL CLEAR :: M$="ABCDEF
GHIJKLMNOPQRSTUVWXYZ12345678
90!@#$%^&*()+=-/:;,.<>" :: C
ALL BIGBANNER(M$)
110 CALL BIGBANNER("THIS IS
A DEMONSTRATION OF THE BANNE
R SUBPROGRAM"):: STOP
```

BIGTITLER

CALL BIGTITLER(T,C) will read T (not over 6) DATA words from the main program (totaling not over 28 characters, and not over 15 characters per word) and display them as MAGNIFY(2) sprites in color C. Example:
```
100 DATA TIGERCUB,SOFTWARE,N
UTS,&,BOLTS
110 RESTORE 100 :: CALL CLEA
R :: CALL SCREEN(5):: CALL B
IGTITLER(5,16)
120 GOTO 120
```

BLINK

CALL BLINK(R,M$,J$) will display the string M$ at row R, column 1, and will blink the selected portion J$ (which must be all on one line) one time. Example:
```
100 M$="WHEN YOU HAVE READ T
HE      INSTRUCTIONS PRESS E
NTER TO  CONTINUE" :: J$="EN
TER" :: CALL CLEAR
110 CALL BLINK(7,M$,J$):: CA
LL KEY(0,K,ST):: IF ST=0 THE
N 110
120 J$="CONTINUE" :: GOTO 11
0
```

BOX

CALL BOX(R,M$) will display M$ of up to 26 characters at row R. On subsequent CALLs, each M$ is displayed on the next line, for up to 22 lines. Then, if the last M$ is "END", a box is drawn around the whole. Example:
```
100 DATA TIGERCUB,SOFTWARE,"
 ",156 COLLINGWOOD AVE.,COLU
MBUS OHIO 43213,END
110 CALL CLEAR :: RESTORE 10
0 :: FOR J=1 TO 6 :: READ M$
 :: CALL BOX(5,M$):: NEXT J
120 GOTO 120
```

CENTER

CALL CENTER(M$,C) will return in C the column number to be used in DISPLAY AT or in PRINT TAB to center M$ on the screen. Example:
```
100 DATA THIS IS,A TEST,OF T
HE,TIGERCUB,NUTS & BOLTS,CEN
TER SUBPROGRAM
110 CALL CLEAR :: FOR R=1 TO
 12 STEP 2 :: READ M$ :: CAL
L CENTER(M$,C):: DISPLAY AT(
R,C):M$ :: NEXT R
120 GOTO 120
```

CENTERING

CALL CENTERING(N), where N is the number of DATA items, will read up to 22 DATA items of up to 28 characters each and display them centered both vertically and horizontally, double-spaced if less than 13 lines. Example:
```
100 DATA THIS IS,A,DEMONSTRA
TION,OF THE,TIGERCUB,NUTS &
BOLTS,CENTERING SUBPROGRAM
110 DATA AND,IT ALSO WORKS,W
ITH MORE,THAN,A,DOZEN,LINES
120 CALL CLEAR :: CALL CENTE
RING(7):: FOR D=1 TO 800 ::
NEXT D :: CALL CLEAR :: REST
ORE 100 :: CALL CENTERING(14
)
130 GOTO 130
```

CURTAIN3

CALL CURTAIN3 will cover the screen with a colorful curtain of random color and pattern, read up to 12 DATA items of up to 32 characters each from the main program (the last DATA item must be END), and then seemingly display the text by drawing back the curtain both ways from center. Example:
```
100 DATA THIS IS A DEMONSTRA
TION,OF THE CURTAIN3 SUBPROG
RAM,FROM THE NUTS & BOLTS,DI
SK #2,SOLD BY TIGERCUB SOFTW
ARE,156 COLLINGWOOD AVE.
110 DATA COLUMBUS OHIO 43213
,END
120 CALL CLEAR :: CALL SCREE
N(5):: FOR S=1 TO 12 :: CALL
 COLOR(S,16,5):: NEXT S :: R
ESTORE 100 :: CALL CURTAIN3
```

ENTER

CALL ENTER will display "Press ENTER to continue" at bottom of screen, flash ENTER on and off until any Key is pressed, then delete itself. Example:
```
100 CALL CLEAR :: CALL ENTER
```

EXPLODE

CALL EXPLODE(M$,R,C,SP) will display a line of up to 28 characters of M$ at row R, column C, and will then send them flying in all directions one by one with an explosive sound of duration SP (1 to 25) or silently if SP is 0. Example:
```
100 CALL CLEAR :: CALL EXPLO
DE(12,5,"EXPLODING TITLE",1)
```

ERRORSHOW

CALL ERRORSHOW, when called from an error-trapping routine, uses the 5 highest-numbered sprites to flash an ERROR message on the screen without disturbing the screen display, then deletes them. Example:

```
100 INPUT "FILENAME? (Type s
ome garbage)":F$
110 ON ERROR 120 :: OPEN #1:
"DSK1."&F$
120 CALL ERRORSHOW :: RETURN
 100
```

FLIPFLOP

CALL FLIPFLOP(M$) will display a one-line string M$ in upper-case leters at the top of the screen and then flipflop it down the screen alternately upside down and right side up. ASCII 97-122 are redefined as upside-down characters, therefore may also be colored differently. Example:

```
100 FOR SET=9 TO 12 :: CALL
COLOR(SET,16,1):: NEXT SET
110 INPUT M$ :: CALL CLEAR :
: CALL FLIPFLOP(M$):: GOTO 1
00
```

LEFTRIGHT

CALL LEFTRIGHT(M$,R,T) will scroll a string M$ of up to 229 characters across the screen from left to right at row R within a solid border, repeated T times. ASCII 127 is redefined for the border. Example:

```
100 CALL CLEAR :: CALL LEFTR
IGHT(RPT$("  TIGERCUB SOFTW
ARE  ",5),10,2)
```

RIGHTLEFT

CALL RIGHTLEFT(M$,R,T) will scroll a string M$ of up to 229 characters across the screen from right to left within a solid frame, repeated T times. ASCII 127 is redefined for the frame. Example:

```
100 CALL CLEAR :: CALL RIGHT
LEFT("THIS IS A DEMONSTRATIO
N OF THE RIGHTLEFT SUBPROGRA
M FROM THE TIGERCUB NUTS & B
OLTS DISK #2  ",10,2)
```

SCROLLDOWN

CALL SCROLLDOWN will move the contents of the screen down by one row, the lowest row disappearing. By saving a screen of text or graphics as DATA statements, then displaying another screen, this routine will scroll a screen down from the top, shoving the other screen off the bottom. Example:

```
100 CALL CLEAR :: DIM M$(24)
:: FOR J=1 TO 24 STEP 2 :: M
$(J),M$(J+1)=RPT$(" ",INT(25
-J)/2+1)&RPT$("*",J):: PRINT
 M$(J):M$(J+1):: NEXT J
110 FOR J=24 TO 1 STEP -1 ::
 CALL SCROLLDOWN :: DISPLAY
AT(1,1):M$(J):: NEXT J :: GO
TO 110
```

TELETYPE

CALL TELETYPE(L,R) will read L items of DATA, up to 28 characters long, from the main program, and scroll them onto the screen from right to left on successive lines starting at row R. Example:

```
100 DATA THIS IS A DEMO OF T
HE TELE-,TYPE SUBPROGRAM FRO
M THE,NUTS & BOLTS NO. 2 DIS
K,SOLD BY TIGERCUB SOFTWARE,
156 COLLINGWOOD AVENUE
110 DATA COLUMBUS OHIO 43213
120 CALL CLEAR :: RESTORE 10
0 :: CALL TELETYPE(6,1)
130 GOTO 130
```

TITLER

CALL TITLER(M$,RR,CC,X) displays a string M$ of up to 8 magnified characters (below ASCII 96) at row RR, column CC, in color X. Characters from ASCII 96 upward are redefined, in 2 characters per sprite. Since the letters are composed of sprites, they can then be controlled by sprite commands. Example:

```
100 CALL CLEAR :: CALL TITLE
R("TIGERCUB",12,6,16)
110 FOR D=1 TO 500 :: NEXT D
 :: CALL MOTION(#1,0,-20,#2,
0,-20,#3,0,-20,#4,0,-20):: F
OR D=1 TO 5000 :: NEXT D
120 CALL DELSPRITE(ALL)
```

VERTPRINT

CALL VERTPRINT(R) will read up to 24 DATA items of up to 28 characters from the main program, then print them column by column, left to right, downward from row R. The last DATA item must be END. Example:

```
100 DATA "   THIS IS A DEMON
STRATION",,OF THE TIGERCUB N
UTS & BOLTS,,"    VERTPRINT
ROUTINE",,TO PRINT VERTICALL
Y LEFT,
110 DATA "        TO RIGHT",
END
120 CALL CLEAR :: CALL VERTP
RINT(5)
130 GOTO 130
```

WAVYTEXT

CALL WAVYTEXT(M$,S), where M$ is any text up to 27 characters long and S is the screen color, will display the text as a multicolored waving banner until any Key is pressed. It looKs best with spacing between the letters. Also try deleting the CALL MAGNIFY. Example:

```
100 CALL CLEAR :: CALL WAVYT
EXT("N U T S  &  B O L T S",
2)
```

ZOOP

CALL ZOOP(M$,R) will display a single-line string M$ at row R, centered, in letters which "zoop" into position. The letters are sprites until they stop moving, therefore can be colored, passed over graphics, etc. Try changing the -7 in CALL SPRITE to +50. The speed of the CALL MOTION can also be changed. Also try CALL MAGNIFY(2). Example:

```
100 CALL CLEAR :: CALL ZOOP(
"TIGERCUB SOFTWARE",12)
110 GOTO 110
```

>>>>>>>> JOYSTICKS <<<<<<<<<

JOYSPEED

CALL JOYSPEED(N,A) where N is the joystick number (1 or 2) and A is the number of a sprite, will control the motion of the sprite. Holding down the fire button will gradually increase the speed, releasing it will slow the sprite to a stop. The CALL must return to itself to continue the motion. Example:

```
100 CALL CLEAR :: CALL CHAR(
96,RPT$("F",64)):: CALL SPRI
TE(#1,96,5,92,124):: CALL MA
GNIFY(4):: CALL SPRITE(#2,96
,16,1,1)
110 X=INT(20*RND)-INT(20*RND
):: Y=INT(20*RND)-INT(20*RND
):: CALL MOTION(#2,X,Y)
120 CALL JOYSPEED(1,1):: CAL
L COINC(#1,#2,8,A):: IF A=-1
THEN 130 ELSE 110
130 Z=Z+1 :: DISPLAY AT(1,1)
:Z :: GOTO 120
```

JOYSPRITE

CALL JOYSPRITE(N,P,S) where N is the number of the joystick (#1 or #2), P is a sprite number, and S is a speed between 0 and 30, will control the motion of the sprite with the joystick. The CALL must return to itself to continue the motion. Example:

```
100 CALL CLEAR :: CALL CHAR(
96,RPT$("F",64)):: CALL SPRI
TE(#1,96,5,92,124):: CALL MA
GNIFY(4)
105 RANDOMIZE :: S=INT(30*RN
D):: CALL JOYSPRITE(1,1,S)::
 GOTO 105
```

TWOSPRITE

CALL TWOSPRITE(A,B) will enable two players to control sprites #A and #B with the two joysticks. Holding down the fire button will increase speed, releasing it will slow the sprite to a halt. The CALL must return to itself. Example:

```
100 CALL CLEAR :: CALL CHAR(
96,"FF"&RPT$("80",14)&"FFFF"
```

```
&RPT$("01",14)&"FF",100,RPT$
("0",14)&"0F0F"&RPT$("0",32)
)
110 CALL SPRITE(#1,96,5,92,1
24,#2,100,16,92,124):: CALL
MAGNIFY(4)
120 CALL TWOSPRITE(1,2):: CA
LL COINC(#1,#2,16,C):: IF C=
-1 THEN CALL SOUND(100,1000,
0)
130 GOTO 120
```

>>>>>>>>>>> MATH <<<<<<<<<<<

Base conversion routines are necessarily limited to a length of 255 digits in the lower-numbered base other than 10.

AMOUNT

CALL AMOUNT(P,R,Y,T,A) will compute the (A)mount resulting from P(rincipal invested at interest (R)ate in decimals (i.e. .0525) for (Y) years and compounded (T)imes annually. Example:

```
100 INPUT "PRINCIPAL":P :: I
NPUT "INTEREST RATE IN DECIM
ALS":R :: INPUT "YEARS":Y ::
 INPUT "TIMES COMPOUNDED":T
:: CALL AMOUNT(P,R,Y,T,A)
110 PRINT "$";A :: GOTO 100
```

ARCSIN

CALL ARCSIN(X,Z) gives the arcsine (inverse of sine) correctly even at right angles (arcsine of 1.000). Example:

```
100 CALL CLEAR
110 INPUT VALUE :: CALL ARCS
IN(VALUE,RESULT):: PRINT RES
ULT :: GOTO 110
```

BIN_DEC

CALL BIN_DEC(B$,D) will convert a binary number B$ of any length to its decimal equivelant D. Example:

```
100 CALL CLEAR
110 ACCEPT AT(12,1)VALIDATE(
"01"):B$ :: CALL BIN_DEC(B$,
D):: PRINT D :: GOTO 110
```

BIN_HEX

CALL BIN_HEX(B$,H$) will convert a binary B$ of up to 252 digits into the equivelant hexidecimal H$. Example:

```
100 INPUT "BINARY #? ":B$ ::
CALL CLEAR :: CALL BIN_HEX(B
$,H$):: PRINT H$ :: GOTO 100
```

BINMATH

CALL BINMATH(M$,B$) will perform a mathematical operation on two binary numbers in string M$ and return the binary result in string B$. The subprograms BIN_DEC and DEC_BIN are called by this subprogram, therefore must also be MERGEd in. M$ must be in the form "1011+101010", substituting binary numbers as desired. Symbols -, * and / can also be substituted for + but the intermediate decimal result of subtraction or division must be a positive integer or an error message will result. Example:

```
100 CALL CLEAR
110 INPUT M$ :: CALL BINMATH
(M$,B$):: PRINT B$ :: GOTO 1
10
```

CONVERTER

CALL CONVERTER(N$,F,T), where N$ is any positive integer in string format in any number base (F) from 2 to 16, will convert N$ into the string representation of itself in any number base (T) from 2 to 16. It is slower than the specialized conversion routines, and will crash on large numbers such as a 16-digit hex code. Example:

```
100 CALL CLEAR
110 INPUT "NUMBER? ":N$ :: I
NPUT "FROM BASE? ":F :: INPU
T "TO BASE? ":T :: CALL CONV
ERTER(N$,F,T):: PRINT "THE A
NSWER IS ";N$: : :: GOTO 110
```

DEC_BIN

CALL DEC_BIN(D,B$) will convert any positive decimal integer D to its binary equivelant B$. Example:

```
100 CALL CLEAR
110 INPUT D :: CALL DEC_BIN(
D,B$):: PRINT B$ :: GOTO 110
```

DECFRACT

CALL DECFRACT(D,F$,N,DV), where D is a numeric value, will give the fraction equivelant in the string F$, the denominator in N and the divisor in DV. Example:

```
100 CALL CLEAR
110 INPUT "DECIMAL? ":D :: C
ALL DECFRACT(D,F$,N,DV):: PR
INT F$;N;DV :: GOTO 110
```

DEC_HEX

CALL DEC_HEX(D,H$) will convert any positive decimal integer D (up to 98303) to its hexadecimal equivelant H$. Example:

```
100 INPUT D :: CALL DEC_HEX(
D,H$):: PRINT H$ :: GOTO 100
```

FACTOR

CALL FACTOR(N(),GCD) gives the greatest common denominator GCD of an array of numbers N(). Example:

```
100 INPUT "HOW MANY NUMBERS?
 ":Q :: CALL CLEAR
110 FOR J=1 TO Q :: INPUT N(
J):: NEXT J
120 CALL FACTOR(N(),GCD):: P
RINT "GCD=";GCD: : :: GOTO 1
00
```

HEX_BIN

CALL HEX_BIN(H$,B$) will convert any hexadecimal H$ (not over 63 digits of F) into its binary equivelant B$. Example:

```
100 INPUT "HEXADECIMAL #? ":
H$ :: CALL CLEAR :: CALL HEX
_BIN(H$,B$):: PRINT B$ :: GO
TO 100
```

HEX_DEC

CALL HEX_DEC(H$,D) will convert any hexadecimal number H$ (up to RPT$("F", 107) to its decimal equivelant D. Example:

```
100 CALL CLEAR
110 INPUT H$ :: CALL HEX_DEC
(H$,D):: PRINT D :: GOTO 110
```

HEXMATH

CALL HEXMATH(M$,H$) will perform mathematical operations on two hexadecimal numbers (maximum value 7FFD) expressed in M$ in the form "FF+1A" and return the hexadecimal result in H$. Subprograms HEX_DEC and DEC_HEX are called by this subprogram, therefore must also be MERGEd in. Symbols -, * or / can also be used for subtraction, multiplication or division. Maximum result is FFFF and negative or non-integer results will crash. Example:

```
100 CALL CLEAR
110 INPUT M$ :: CALL HEXMATH
(M$,H$):: PRINT H$ :: GOTO 1
10
```

>>>>>>>>>> GRAPHS <<<<<<<<<<

COLORGRAPH

CALL COLORGRAPH(N,A(),T$()) will display a vertical bargraph of up to 28 bars, accurate to 1/8 of a print space and as wide as space permits, in 6 colors, and with title above each bar. N is the number of records, A() is the array of values and T$() is the array of titles. Titles will be truncated as necessary. Example:

```
100 DATA 76,40,120,311,3,93,
112,TAX,DEPR,WAGE,SALES,PROF
IT,OVHD,MISCELLANEOUS
110 FOR J=1 TO 7 :: READ A(J
):: NEXT J :: FOR J=1 TO 7 :
: READ T$(J):: NEXT J :: CAL
L COLORGRAPH(7,A(),T$())
120 GOTO 120
```

GRAFPRINT

CALL GRAFPRINT(N,M$(),V()), where N is the number of records, M$() is the array of their titles and V() is the array of their values, will output a horizontal bar chart to a Gemini printer. For other printers, it may be necessary to change the "PIO" in line 20732 and perhaps the CHR$(175) in line 20735. Example:

```
100 DATA JANUARY,FEBRUARY,MA
RCH,APRIL,MAY,JUNE,JULY,AUGU
ST,SEPTEMBER,OCTOBER,NOVEMBE
R,DECEMBER
110 RESTORE 100 :: DIM M$(12
),V(12):: FOR J=1 TO 12 :: R
EAD M$(J):: V(J)=500*RND ::
NEXT J :: CALL GRAFPRINT(12,
M$(),V())
```

GRAPH

CALL GRAPH(G$(),G()) will display a colored horizontal bargraph of up to 18 values, accurate to 1/8 of a print space, with labels at the left (truncated to 3 spaces) and with the title and scale displayed above. G$() is the array of labels, with the graph title in G$(0), and G() is the array of values. Example:

```
100 CALL CLEAR :: DIM G(20),
G$(20)
110 DATA JAN,18.2,FEB,17.8,M
AR,21.3,APR,14.7,MAY,8.0,JUN
,22.9,JUL,16.7,AUG,18.1,SEP,
19.3
120 DATA OCT,26.2,NOV,26.2,D
EC,26.6,AVG,21.1,84,18.4,83,
22.1,82,24.0,81,21.6,80,27.0
,79,26.1
130 FOR I=1 TO 18 :: READ G$
(I),G(I):: NEXT I :: G$(0)="
RAINFALL 1985"
140 CALL GRAPH(G$(),G())
150 GOTO 150
```

MANYBARS

CALL MANYBARS(N,V()), where N is the number of values to be graphed (up to 56), and V() is the array containing their values, will display a vertical bar chart of narrow black and white lines on a horizontally lined ground. If you have a 32-column monitor, you may change the 56 in line 20725 to 64, and change C=3 to C=1, to display 64 values. For a one-color graph, change the 16's in line 20726 to 2's. Example:

```
10 RANDOMIZE
100 DIM V(56):: FOR J=1 TO 5
6 :: V(J)=INT(100*RND):: NEX
T J :: CALL MANYBARS(56,V())
110 GOTO 110
```

STACKGRAPH

CALL STACKGRAPH(N,V()), where N equals the number of values (not over 15) to be graphed, and V() is the array containing their values, will display a vertical 3-dimensional graph of colored blocks - more accurate if values do not vary too greatly. Example:

```
100 CALL CLEAR :: DIM V(15):
: FOR J=1 TO 15 :: V(J)=INT(
500*RND):: NEXT J :: CALL ST
ACKGRAPH(15,V())
110 GOTO 110
```

TRIGRAPH

CALL TRIGRAPH(N,V()) will graph an array V() of N values as a horizontal curve graph, using triangles which indicate increase or decrease by their slant even when the scale of the graph leaves them on the same line. Example:

```
100 DIM V(30):: X=50 :: FOR
J=1 TO 30 :: RANDOMIZE :: X=
X+5*RND-7*RND+1 :: V(J)=X ::
 NEXT J :: CALL TRIGRAPH(30,
V()):: GOTO 100
```

>>>>>> SELF-CHANGERS <<<<<<<

GOSUB

CALL GOSUB(N), where N is any line number in the program, will perform a GOSUB to that number, thus permitting a variable GOSUB. This subprogram MERGEs into lines 32762-32767 because its last line must be the last line of the program. Therefore it cannot be used in the same program as CALL GOTO or CALL RESTORE, which use the same line numbers. Example:

```
100 INPUT "101, 1001 OR 1000
1? ":N :: CALL GOSUB(N):: GO
TO 100
101 PRINT "LINE 101" :: RETU
RN
1001 PRINT "LINE 1001" :: RE
TURN
10001 PRINT "LINE 10001" ::
RETURN
```

GOTO

CALL GOTO(N), where N is any line number in the program, will rewrite line 32767 to read GOTO that line number, which can then be accomplished by GOTO 32767. This subprogram cannot be used together with CALL GOSUB, for the reason stated there, or with CALL RESTORE. Example:

```
100 INPUT "101, 1001 OR 1000
1? ":N :: CALL GOTO(N):: GOT
O 32767
101 PRINT "LINE 101" :: GOTO
 100
1001 PRINT "LINE 1001" :: GO
TO 100
10001 PRINT "LINE 10001" ::
GOTO 100
```

RESTORE

CALL RESTORE(N), where N is any DATA line number in the program, will RESTORE that line number, thus permitting a variable RESTORE. It cannot be used together with CALL GOSUB, for the reason stated there, or with CALL GOTO. Example:

```
100 INPUT "101, 1001 OR 1000
1? ":N :: CALL RESTORE(N)::
READ D$ :: PRINT D$ :: GOTO
100
101 DATA 101
1001 DATA 1001
10001 DATA 10001
```

>>>>>>>>>> SPEECH <<<<<<<<<<

SAY_NUM

CALL SAY_NUM(NR), when the Speech Synthesizer is attached, will speak any number between 0 and 999 correctly, rather than speaking each digit. Example:

```
100 CALL CLEAR :: INPUT "NUM
BER? (1-999)":NR :: CALL SAY
_NUM(NR):: GOTO 100
```

>>>>>> SOUND EFFECTS <<<<<<<

EFFECT

CALL EFFECT(N,T), where N is a number from 1 to 6, will call one of 6 unusual sound effects including (1)pow-pow (2)tremolo, (3)poing-a-poing (4)sonar, (5)sirens,(6)groan repeated T times. Example:

```
100 INPUT "EFFECT #(1-6),# O
F TIMES? ":N,T :: CALL EFFEC
T(N,T):: GOTO 100
```

SOUNDS

CALL SOUNDS(A,B,C,D), where A, B, C and D are any positive or negative numbers (out-of range numbers give silence) will produce a very wide variety of unusual sound effects. The sound is determined by the values of A, B, C and D, the length of the sound is determined by the difference between A and B and the difference between C and D. Experiment! Press any Key to stop. Example:

```
100 INPUT "A,B,C,D?":A,B,C,D
 :: CALL SOUNDS(A,B,C,D):: G
OTO 100
```

>>>>> WORD PROCESSING <<<<<<

FORMAT

CALL FORMAT(M$) will insert blanks as necessary in string M$ so that it will display on the screen without breaking a word at the end of a line. This is useful when concatenating strings. If insertions cause the string to exceed 255 characters a WARNING will print. Example:

```
100 INPUT M$ :: CALL FORMAT(
M$):: PRINT M$ :: GOTO 100
```

NUMTH

CALL NUMTH(N,N$), when N is any number between 1 and 99, will return in N$ the text form such as FIRST, NINETY-NINTH, etc. Example:

```
100 CALL CLEAR
110 INPUT "NUMBER(1-99)? ":N
 :: CALL NUMTH(N,N$):: PRINT
 N$ :: GOTO 110
```

PLURAL

CALL PLURAL(W$,PL$), where W$ is any noun (except those with irregular plural forms), will return the plural form in PL$) Example

(try CAT, WOMAN, KNIFE, BABY, BOY):
```
100 INPUT W$ :: CALL PLURAL(W$,PL$):: PRINT PL$ :: GOTO 100
```

REPLACE

CALL REPLACE(OLD$,M$,R$) will replace a string OLD$, in the string M$, with R$. Example:
```
100 INPUT "WHAT IS YOUR NAME ? ":R$ :: M$="THIS PROGRAM WAS WRITTEN FOR @ TO USE" :: OLD$="@" :: CALL REPLACE(OLD$,M$,R$):: PRINT M$ :: END
```
Try following this with the FORMAT subprogram.

>>>>>>>> UTILITIES <<<<<<<<

INITCHECK

CALL INITCHECK will check whether CALL INIT has already been called since entering Extended Basic and, if not, will CALL INIT. This prevents loss of assembly routines in memory, etc.

QUICKCOLOR

CALL QUICKCOLOR(F,B) will change all character sets to foreground color F and background color B, in about one second the first time and instantly thereafter. Input values 0 and 0 to return control to CALL COLOR, 0 and 0 again to return control to subprogram. Example:
```
100 F=2 :: B=16 :: CALL QUICKCOLOR(F,B):: FOR D=1 TO 50 :: NEXT D :: F=16 :: B=2 :: CALL QUICKCOLOR(F,B):: FOR D=1 TO 50 :: NEXT D :: GOTO 100
```

RESET

CALL RESET clears the screen, deletes all sprites, sets MAGNIFY at (1), sets screen and character sets to standard colors, restores all characters to standard patterns. Used before RUNning one program from another.

RESETLOAD

CALL RESETLOAD restores standard colors and characters, deletes sprites, RUNs a LOAD program if there is one on the disk.

STACKSIZE

CALL STACKSIZE(M) will return in M the approximate number of bytes available in stack memory - apparently sometimes more accurately than the SIZE command! Example:
```
100 DIM M$(100):: X$=RPT$("A",255)
110 X=X+1 :: M$(X)=X$ :: CALL STACKSIZE(M):: PRINT M;"type SIZE, then type CON" :: BREAK
120 GOTO 110
```

>>>>>>>>> GRAPHICS <<<<<<<<<

DCHAR

CALL DCHAR(R,C,CH,T) will display ASCII CH for T times, diagonally from upper left to lower right, beginning at row R, column C, and wrapping around as necessary. Example:
```
100 CALL CLEAR :: CALL DCHAR(20,20,42,40)
```

DRAWLINE

CALL DRAWLINE(R,A,B) will smoothly draw a horizontal line on row R from column A to column B, directly below any characters on the line. Example:
```
100 CALL CLEAR :: DISPLAY AT(10,1):"DRAW A LINE UNDER LINE" :: CALL DRAWLINE(11,10,13):: CALL DRAWLINE(15,1,32):: GOTO 100
```

LCHAR

CALL LCHAR(R,C,CH,T) will display ASCII CH for T times, diagonally from upper right to lower left, beginning at row R, column C and wrapping around as necessary. Example:
```
100 CALL CLEAR :: CALL LCHAR(12,16,42,100)
```

>>>>>> MISCELLANEOUS <<<<<<<

MOON

CALL MOON(Y,M,D,P), where Y is any year after 1581, M is the month number and D is the date, will return in P the ASCII (132-143 redefined) which will print the phase of the moon. Example:
```
100 CALL CLEAR
110 INPUT "YEAR,MONTH,DAY :":Y,M,D :: CALL MOON(Y,M,D,P):: PRINT CHR$(P):: GOTO 110
```

>>> PROGRAMMER UTILITIES <<<

ACCEPTDUMP

CALL ACCEPTDUMP(F,P$), where F is a file number and P$ is either a printer designation or a disk/filename, will dump any or all lines from screen to printer or disk. Press Enter for each line, then Enter again to print or Q to quit or any other Key to go to next line. Example:
```
100 PRINT "BREAK THE PROGRAM AND TYPE":"'LIST' FIRST TO GET SOME-":"THING ON THE SCREEN, THEN":"RUN AGAIN"
110 CALL ACCEPTDUMP(F,"PIO")
```

DATAWRITER

CALL DATAWRITER(F,F$,LN,N$) will open a file #F with disk/filename F$ and begin writing a MERGE format program of DATA statements beginning with line number LN, containing the value of N$, which may be either text or numeric. Each subsequent CALL will add another DATA DATA item; an input of END for N$ will close the file and reset the program.
NOTE!: This creates program lines of DATA statements which may be merged into another program, NOT a file of DATA which may be read directly. Example:
```
100 INPUT N :: CALL DATAWRITER(1,"DSK1.DATAFILE",1000,N) :: IF N=99999 THEN STOP ELSE 100
```

DISKCAT

CALL DISKCAT(D,F), where D is the disk drive number and F is any file number not currently open, will list the disk catalog to the screen, pausing at each screenfull until any Key is pressed.Example:
```
100 CALL CLEAR :: CALL DISKCAT(1,255)
```

EDITOR

CALL EDITOR(F,F$,L) will set up a screen on which you can perform full-scale editing to format a screenfull of text. If characters have been predefined, you can also use it to create graphics. You may move downward by pressing either Enter or FCTN X, upward by pressing FCTN E, and even wrap around top or bottom. These keys respond slowly, so hold them down until the cursor moves. You can also edit with FCTN S, D, 1, 2 and 3. When finished, type FCTN V in any blank space and Enter. The screen will be converted into a MERGE format program of DISPLAY AT lines. F is the file number and F$ is the disk/filename to be used, and L is the starting line number, which is incremented by 1.

EXTRACTOR

Will extract any selected series of lines from any program into which it is merged.
Unlike the others, this one is not in subprogram format, and is line-numbered 1 to 6. The program into which it is merged should have line numbers starting higher than 6. After it runs, you MUST save the resulting extracted lines by SAVE DSK1.(filename),MERGE and then MERGE DSK1.(same filename).

KEEPSHOW

CALL KEEPSHOW(A,B) will read rows A through B of a screen

into an array, while erasing them, if Enter is held down. CALL KEEPSHOW(A,B) again will then display those lines on the screen at rows A to B. Example - LIST to fill screen, then:

```
100 DISPLAY AT(24,1):"MOVE L
INE #? " :: ACCEPT AT(24,15)
:L :: CALL KEEPSHOW(L,L)
110 DISPLAY AT(24,1):"MOVE T
O LINE #?" :: ACCEPT AT(24,1
6):L :: CALL KEEPSHOW(L,L)::
 GOTO 100
```

LINESAVER

CALL LINESAVER(S$()) will save selected lines from the screen into an array S$(). Press Enter for each line, then Enter again to save or Q to quit or any other Key to go to next line. Subsequent CALLs continue adding to the same array unless its name is changed. Example (LIST something to the screen, RUN, save 5 lines, quit, save 5 lines, quit):

```
100 CALL LINESAVER(A$()):: C
ALL LINESAVER(B$()):: FOR J=
1 TO 5 :: PRINT A$(J):: NEXT
 J :: FOR J=1 TO 5 :: PRINT
B$(J):: NEXT J
```

LONGACCEPT

CALL LONGACCEPT(L,M$) will accept a string M$ of up to 255 characters beginning at row L. If L is 0, input will begin at row 1, continue downward for subsequent inputs, clear the screen after row 24 and return to row 1. WARNING - fast typing will cause skipped letters. FCTN S can be used to backspace and correct. Example:

```
100 CALL CLEAR
110 X=X+1 :: CALL LONGACCEPT
(0,M$):: A$(X)=M$ :: DISPLAY
 AT(20,1):A$(X):: GOTO 110
```

SCREENSAVE

CALL SCREENSAVE(F,F$), where F is a file number and F$ is is a disk/filename such as "DSK1.KEEPSCREEN", will read a 28x24 screen of graphics and/or text and convert it into a MERGE format program occupying 6 sectors. To use this routine, load any program, RUN it to the point where it displays a screen you wish to copy, and break it with FCTN 4. Put in a temporary line at that point with GOTO (its own line number) and RUN again to be sure you have the right point. Then change the temporary line to CALL SCREENSAVE(1,"DSK1.KEEPSCREEN") - or whatever. MERGE in SCREENSAVE. Be sure the disk has 6 sectors available, and RUN. Then type NEW, MERGE (dsk/filename) and RUN. The resulting program can be RESequenced, SAVEd again in MERGE format, MERGEd into any program.

NOTE: Any redefined characters in the program will be saved in redefined form even though they may not be used in the screen being saved.

STRINGEDIT

CALL STRINGEDIT(M$) will enable you to perform on-screen editing of string data. Note that when a string of characters is to be changed, the computer looks for the first occurrence of this string. In the following example, if you try to change IS to WAS, the computer will change THIS to THWAS instead. However, you can change THIS IS to THIS WAS. Example:

```
100 M$="THIS IS A TEST OF TH
E STRING EDITOR SUBPROGRAM"
110 CALL STRINGEDIT(M$):: DI
SPLAY AT(12,1):"AGAIN?" :: A
CCEPT AT(12,8):Q$ :: IF Q$="
Y" THEN 110 ELSE STOP
```

TRACEPRINT

TRACEPRINT is a utility to enable you to dump a screenfull of TRACE printout to a printer. It contains its own CALL. To use it, load the program to be TRACED and merge in TRACEPRINT. Add a temporary line such as 1000 TRACE wherever you want the TRACE to start, or just type TRACE to start from the beginning. RUN. When 20 lines have listed to the screen, BREAK with FCTN 4. Enter RUN 21305. Press Enter for each line, then press Enter at the prompt if you want to print the line, or any other Key if not. This routine can also be used to print any other text from the screen. Change the and printer designation as necessary.

>>>>>> FILE HANDLING <<<<<<<

ERRORTRAP

CALL ERRORTRAP(F,F$), where F is a file number and F$ is a disk/filename, will open a disk file and display a warning rather than crashing if drive is empty, door open, filename not on disk, etc. It will also CLOSE the "ajar" file so that it can be opened when the error is corrected. For other than DISPLAY VARIABLE OUTPUT files, change the first line as necessary. Example:

```
100 DISPLAY AT(12,1)ERASE AL
L:"OPEN DISK DRIVE DOOR, THE
N":"PRESS ANY KEY"
110 CALL KEY(0,K,S):: IF S=0
 THEN 110
120 CALL ERRORTRAP(1,"DSK1.T
EST"):: PRINT #1:"TEST" :: C
LOSE #1 :: END
```

READFILE

CALL READFILE will input any listable file from disk and list it to the screen, pausing on any Keypress and continuing on any Keypress, or will output it to a printer. This routine will perform a "blind" read of numeric and/or string records from DISPLAY or INTERNAL files, FIXED or VARIABLE. (The D/V 163 MERGE format files on this disk can be listed, but most of the characters are out of printable range).

READSTRING

CALL READSTRING(M$(),A,B,N) will read a MERGEd program created by STRINGSAVE and extract a value N from M$(A), position B, in the same way as from a 2-dimensional array M$(A,B) but more slowly. Example - MERGE the READSTRING subprogram into the SAVE program created by the STRINGSAVE example, and add these lines after the last line of the SAVE program -

```
1100 DISPLAY AT(12,1)ERASE A
LL:"INPUT TW0 VALUES SEPARAT
ED BY COMMA"
1110 INPUT "A (1-50),B(1-20)
 ":A,B :: CALL READSTRING(M$
(),A,B,N):: PRINT N :: GOTO
1110
```

STRINGSAVE

CALL STRINGSAVE(A,N(),F$,L,B) provides an extremely compact method of numeric data storage, much more memory-efficient than DATA statements, and sometimes preferable to disk files, although retrieval time may be slower. The entire console memory may be used for numeric data storage. N() is the numeric array from which data is read, A is the number of records in the array, F$ is the disk/filename of the MERGE format program to be created, L is the starting line number to be used in this program (incremented by 1). B is the number of records to be stored in each line; B must not exceed 31 and the total number of characters in the records to be stored in one line must not exceed 140. Retrieval time is faster when fewer records are stored in a line. Data is retrieved by the READSTRING subprogram. Example:

```
100 for J=1 to 1000 :: N(J)=
INT(1000*RND):: NEXT J :: CA
LL STRINGSAVE(1000,N(),1,"DS
K1.SAVE",1000,20)
```

Then NEW, MERGE DSK1.SAVE, LIST to see the results, add a line 999 DIM M$(50) and

merge in READSTRING and STRINGSORT to try it out.

```
>>>>>> MENU ROUTINES <<<<<<
```

MENU

CALL MENU(A,R,B) will read DATA items from the main program, display them beginning at row R with the initial letter in parentheses (each must begin with a different initial!), display at row 20 an instruction to type initial leter of choice. A is the number of DATA items to be read, B will be the sequence number of the selected routine. Example:

```
100 DATA MAKE NOISE,TURN SCR
EEN WHITE,RE-RUN,QUIT
110 CALL CLEAR :: RESTORE 10
0 :: CALL MENU(4,8,B)
120 ON B GOTO 130,140,150,16
0
130 CALL SOUND(500,500,5,-4,
5):: GOTO 110
140 CALL SCREEN(16):: GOTO 1
10
150 RUN 110
160 END
```

MOUSE

CALL MOUSE(L,P) will read up to 11 (the value of L) DATA items from the main program, display them on alternate lines of the screen, and display a pointer which may be moved up or down, or wrapped around, with either joystick. When it stops, it shifts to point to the nearest line. If the fire button is pressed, it will return in P the sequence number of that line. Example:

```
100 DATA START FILE,READ FIL
E,UPDATE FILE,SORT FILE,DELE
TE FILE,COMBINE FILES,PRINT
FILE,QUIT
110 RESTORE 100 :: FOR J=1 T
O 8 :: READ M$(J):: NEXT J :
: RESTORE 100 :: CALL MOUSE(
8,P)
130 CALL CLEAR :: CALL DELSP
RITE(ALL):: ON P GOSUB 140,1
50,160,170,180,190,200,210
135 FOR DELAY=1 TO 200 :: NE
XT DELAY :: GOTO 110
140 PRINT M$(1):: RETURN
150 PRINT M$(2):: RETURN
160 PRINT M$(3):: RETURN
170 PRINT M$(4):: RETURN
180 PRINT M$(5):: RETURN
190 PRINT M$(6):: RETURN
200 PRINT M$(7):: RETURN
210 PRINT M$(8):: RETURN
230 END
```

```
>>>> SORTS AND SHUFFLES <<<<
```

INDEXSORT

CALL INDEXSORT(A$(,),N,K,Q()) will sort a 2-dimensional array A$(,) of N records on element K and will return the sorted array in A$(Q()). The index array Q() must be DIMensioned for the same size as A$(,). This is by far the fastest, and the best when memory is available. Example:

```
100 CALL CLEAR :: DIM A$(20,
4),Q(20):: RANDOMIZE
110 DEF X$=CHR$(INT(26*RND+6
5))
120 FOR J=1 TO 20 :: FOR L=1
 TO 4 :: A$(J,L)=X$&X$&X$ ::
 NEXT L :: NEXT J
130 INPUT "SORT BY?(1-4)":K
140 CALL INDEXSORT(A$(,),20,
K,Q())
150 FOR X=1 TO 20 :: FOR L=1
 TO 4 :: PRINT A$(Q(X),L);"
";:: NEXT L :: PRINT :: NEXT
 X :: GOTO 130
```

INDEXSORTN

CALL INDEXSORTN(A(,),N,K,Q()) is the same as INDEXSORT, but will sort a numerical array. Example:

```
100 CALL CLEAR :: DIM A(20,4
),Q(20):: RANDOMIZE
110 DEF XX=INT(1000*RND)
120 FOR J=1 TO 20 :: FOR L=1
 TO 4 :: A(J,L)=XX :: NEXT L
 :: NEXT J
130 INPUT "SORT BY?(1-4)":K
140 CALL INDEXSORTN(A(,),20,
K,Q())
150 FOR X=1 TO 20 :: FOR L=1
 TO 4 :: PRINT A(Q(X),L);::
NEXT L :: PRINT :: NEXT X ::
 GOTO 130
```

STRINGSORT

CALL STRINGSORT(M$(),A,B) will sort strings which have been created by the STRINGSAVE subprogram. A is the number of strings M$(), B is the element to be sorted on. This is the equivelant of a 2-dimensional array sort, slow but requires no additional dimensioning of memory. Example: MERGE this routine and the READSTRING routine into the SAVE program created by the example of the STRINGSAVE routine, and -

```
1100 INPUT "SORT ON?(1-20)":
B :: CALL STRINGSORT(M$(),50
,B):: FOR J=1 TO 50 :: CALL
READSTRING(M$(),J,B,N):: PRI
NT N;:: NEXT J :: GOTO 1100
```

SWAPNUMBER

CALL SWAPNUMBER(J,S,K,A(,)) will sort on element K of a 2-dimensional numeric array A(,) of J records of S elements. See remarKs under SWAPSORT. Example:

```
110 FOR J=1 TO 9 :: FOR L=1
TO 7 :: A(J,L)=INT(90*RND+10
):: NEXT L :: NEXT J
120 INPUT "SORT BY?(1-7)":K
130 CALL SWAPNUMBER(9,7,K,A(
,))
140 FOR X=1 TO 9 :: FOR L=1
TO 7 :: PRINT A(X,L);:: NEXT
 L :: PRINT :: NEXT X :: GOT
O 120
```

SWAPSORT

CALL SWAPSORT(J,S,K,A$(,)) will sort on element K of a 2-dimensional string array of J records of S elements. This one handles strings of any length and arrays of any length with any number of elements, and does not require DIMensioning extra arrays, but is therefore very slow for large arrays. Best for arrays of few records with many elements per record, or when memory is at a premium. Example:

```
90 CALL CLEAR :: DIM A$(20,4
):: RANDOMIZE

100 DATA JONES,HOUSTON,TEXAS
,ANDERSON,DULUTH,MINN.,MARTI
NEZ,SAN DIEGO,CALIF.,GOLDSTE
IN,BROOKLYN,NEW YORK
110 FOR J=1 TO 4 :: FOR L=1
TO 3 :: READ A$(J,L):: NEXT
L :: NEXT J
170 INPUT "SORT BY?(1-3)":K
175 CALL SWAPSORT(4,3,K,A$(,
))
176 FOR X=1 TO J :: FOR L=1
TO 3 :: PRINT A$(X,L);" ";::
 NEXT L :: PRINT :: NEXT X :
: GOTO 170
```

TWOWAYSORT

CALL TWOWAYSORT(A$(,),N,K,Q(),Q$) is same as INDEXSORT except that the sort will be by ASCII if Q$="S" or numeric if Q$="N" (if non-numeric data is found, an error message is printed and the subprogram aborts without crashing). This is useful, for instance, in performing a ZIP code sort of a name/address file. Example:

```
100 CALL CLEAR :: DIM A$(5,6
),Q(5)
110 DATA JONES,JOHN J.,200 O
AK ST.,ANYTOWN,OH,43111,WILL
IAMS,BILL B.,150 ELM ST.,SOM
EWHERE,AL,38765
111 DATA SMITH,SAM X.,33 2ND
 AVE.,PODUNK,MI,42001,GOMEZ,
PEDRO L.,99 CHESTNUT ST.,AUS
TIN,TEX,67345
112 DATA OLSEN,SVEN S.,120 D
AIRY AVE.,FARMTOWN,MN,89333
120 FOR J=1 TO 5 :: FOR L=1
TO 6 :: READ A$(J,L):: NEXT
L :: NEXT J
130 INPUT "SORT BY?(1-6)":K
:: INPUT "SORT BY (S)tring o
r (N)umber? ":Q$
140 CALL TWOWAYSORT(A$(,),5,
K,Q(),Q$)
150 FOR X=1 TO 5 :: FOR L=1
TO 6 :: PRINT A$(Q(X),L);" "
;:: NEXT L :: PRINT :: NEXT
X :: GOTO 130
```