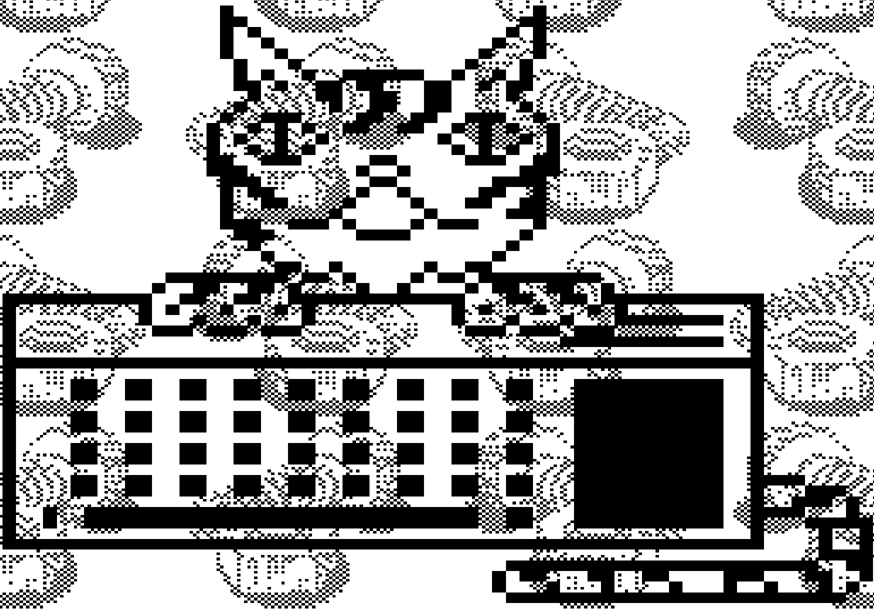
NUTS & BOLTS NUMBER 3 DOCUMENTATION 1987



TIGERCUB SOFTWARE

JIM PETERSON

NUTS & BOLTS

No. 3

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus OHIO 43213

This disk contains 140 utility programs which are recorded in MERGE format so that you can incorporate them into your own programs by simply typing -
MERGE DSK1. (and the program name). They have line numbers running from 21800 to 22640 so that they will not overwrite any of your program lines, and they have consecutive line numbers so that any number of them may be MERGEd into your program without interfering with each other, or with the 208 subprograms on Nuts & Bolts disks #1 and #2.
Almost all of them are in the format of subprograms, so that any values assigned to variable names within them will not affect variables of the same name in the body of your program, unless they are passed in the parameter list.

DATA statements in a main program can be read from a subprogram, and vice versa. Some of these routines contain DATA which is RESTORED and READ internally. After CALLing any such subprogram, be sure to RESTORE any main program DATA before reading it.
Program execution follows the last open ON ERROR, whether in the main program or in a subprogram. A few of these routines contain an ON ERROR as an essential part of the algorithm; it is cancelled by ON ERROR STOP before leaving the subprogram, but any active ON ERROR in the main program must be reactivated after CALLing such a subprogram.
Otherwise, these subprograms generally do not contain error trapping, which can be done more effectively before CALLing a subprogram.
Some of the routines contain a flag, immediately after the SUB, reading IF F=1 THEN (line number):: F=1 These speed up execution, after the first CALL, by skipping over the initialization, but they also prevent reinitializing for subsequent use. When it is necessary to have them reinitialize, the flag routine can usually be deleted. Or, the subprogram can be MERGEd into blank memory, renamed after the SUB, resequenced to some other high line numbers, SAVEd in MERGE format under the new name, MERGEd back into the program and CALLed by the new name.

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

ARROW

CALL ARROW will redefine the E, S, D and X keys to print their arrow symbols on the screen when keyed in with the CTRL key held down. BXB is CALLed internally, so must also be merged in. Type in the EXSD keys in this demo with CTRL down (will be invisible until the program is run).

```
100 CALL ARROW :: DISPLAY AT
(12,1)ERASE ALL:"PRESS E TO
GO UP, X TO GO  DOWN, S TO
GO LEFT, D TO GO RIGHT"
110 GOTO 110
```

BIGLOW

CALL BIGLOW(M$,R,C) will display text of M$ (which must be in lower case letters) in greatly enlarged true lower case letters, 14 per line, 2 rows per line, starting at row R, column C. ASCII 97-122 are redefined. Contains internal flag.

```
100 CALL CLEAR :: CALL BIGLO
W("this is a demo",2,3):: CA
LL BIGLOW("of the biglow",5,
3):: CALL BIGLOW("subprogram
",8,3)
110 GOTO 110
```

BIGWRITER

CALL BIGWRITER(S,C,M$) will change the screen to color S and display string M$ of up to 28 characters (but recommend a limit of 20) across the screen in magnified sprites of color C, as nearly horizontal as is possible without blanking out.

```
100 DATA TIGERCUB SOFTWARE,N
UTS & BOLTS,DISK NO. 3,100 S
UBPROGRAMS,IN MERGE FORMAT,R
EADY TO MERGE,INTO YOUR,OWN
PROGRAMS
110 CALL CLEAR :: FOR J=1 TO
 8 :: READ T$ :: CALL BIGWRI
TER(5,11,T$):: NEXT J :: RES
TORE :: GOTO 110
```

BLACKCHAR

CALL BLACKCHAR will redefine characters from 33 to 122 to a very heavy form, including true lower case. Credited to John Hedstrom.

```
100 FOR CH=32 TO 122 :: PRIN
T CHR$(CH);:: NEXT CH :: CAL
L BLACKCHAR
110 GOTO 110
```

DOUBLEHIGH

CALL DOUBLEHIGH("",0,0) should be CALLed first to redefine this character set because it RESTOREs internal DATA. Thereafter, CALL DOUBLEHIGH(M$,R,C) will display a line of up to 28 double-height characters at row R, column C. Text must consist only of capital letters and numerals. Characters 38-47 and 97-122 are redefined.

```
100 CALL DOUBLEHIGH("",0,0):
: CALL CLEAR
110 CALL DOUBLEHIGH("ABCDEFG
HIJKLMNOPQRSTUVWXYZ",5,1)::
CALL DOUBLEHIGH("0123456789"
,10,1)
120 CALL DOUBLEHIGH("BY PAUL
 DUNDERDALE",14,3):: CALL DO
UBLEHIGH("OF TISHUG SYDNEY",
18,3)
130 GOTO 130
```

FANCYFONT

CALL FANCYFONT will redefine the upper case letters to a stylized form. DATA is restored and read internally.

```
100 CALL CLEAR :: FOR CH=65
TO 90 :: PRINT CHR$(CH);:: N
EXT CH :: CALL FANCYFONT
110 INPUT M$ :: GOTO 110
```

FUNNYFONT

CALL FUNNYFONT will reidentify the upper case letters to a distorted style.

```
100 CALL FUNNYFONT :: DISPLA
Y AT(12,1)ERASE ALL:"ABCDEFG
HIJKLMNOPQRSTUVWXYZ" :: GOTO
 100
```

HALFCHAR

CALL HALFCHAR(M$,B$), where M$ is any string not over 136 characters long and not containing lower case letters or other ASCII above 90, will return it in B$ in half-width letters so that up to 56 characters may be printed on one line, or 64 if the DISPLAY subprogram is used. Not too legible on some TVs, better on a monitor. BXB is CALLed within the subprogram, must also be MERGEd in. ASCII from 91 up to 159 are reidentified as needed, one for each two characters. Internal DATA is RESTOREd. Slow to initialize but much faster when CALLed again. Has internal flag.

```
100 M$="! @ # $ % & * ( ) +
= / - , . < > ? ' 1 2 3 4 5
6 7 8 9 0 A B C D E F G H I
J K L M N O P Q R S T U V W
X Y Z " :: CALL HALFCHAR(M$,
B$):: PRINT B$
110 M$="TIGERCUB SOFTWARE NU
TS & BOLTS DISK NO. 3" :: CA
LL HALFCHAR(M$,B$):: PRINT B
$
120 INPUT M$ :: CALL HALFCHA
R(M$,B$):: PRINT B$ :: GOTO
120
```

HEBREW

CALL HEBREW will convert the upper case letters A through V to the Hebrew alphabet, clear the screen, and accept screen text from the keyboard in rows right to left until Enter is pressed. Contains internal flag.

```
100 CALL HEBREW
```

JAPANESE

CALL JAPANESE will redefine ASCII 33, 35-38, 40-43, 45, 47, 58, 60, 62, 64, 92, 94 and 96-124 to the Japanese KataKana alphabet. Diacritical marks are not included.

```
100 CALL JAPANESE
210 INPUT M$ :: GOTO 110
```

MONEY

CALL MONEY will reidentify CTRL C to the cent sign, CTRL L to the British pound symbol and CTRL Y to the Japanese yen symbol. BXB is CALLed internally so must also be MERGEd in.
Key in this example with CTRL C after the 79 and 89, CTRL L before the first 1 and CTRL Y before the second - they will be invisible.

```
100 CALL MONEY :: DISPLAY AT
(12,1)ERASE ALL:"The rate of
 exchange is 89C":"for L1 an
d 79C for Y1"
110 GOTO 110
```

OLDENGLISH

CALL OLDENGLISH will reidentify the numerals, upper case and lower case letters to a medieval form.

```
100 CALL CLEAR :: FOR CH=33
TO 122 :: PRINT CHR$(CH);::
NEXT CH :: CALL OLDENGLISH
110 GOTO 110
```

OLDSTYLE

CALL OLDSTYLE will redefine the upper case letters to an old stylized form. Credited to Neil Lawson.

```
100 CALL CLEAR :: FOR CH=65
TO 90 :: PRINT CHR$(CH);:: N
EXT CH :: CALL OLDSTYLE
110 GOTO 110
```

SUPERCHAR

CALL SUPERCHAR(K,R,C) where K is the ASCII of any character, normal or redefined, will fairly quickly display it magnified 64 times, with the upper left corner at row R, column C. Characters are up to 8 spaces wide and tall, therefore limited to 4 per row in 2 rows. Character 127 is redefined.

```
100 CALL CLEAR :: R,C=8 :: F
OR CH=33 TO 126 :: CALL SUPE
RCHAR(CH,R,C):: NEXT CH
```

SKINNY

CALL SKINNY(K,R,C) is similar to CALL SUPERCHAR except that the characters are only 4 spaces wide, so that 7 of them can be placed in a row. Upper case letters are 7 spaces high, permitting 3 rows; lower case letters are 5 spaces high, permitting 4 rows. Redefined characters may be up to 8 spaces high. K is the ASC of the character, R is the row of the upper edge of the character but upper case letters will be 1 space below this, lower case will be 2 spaces below. C is the column of the left edge. Characters 128-131 are redefined.

```
100 CALL CLEAR :: R,C=0 :: F
OR CH=65 TO 90 :: CALL SKINN
Y(CH,R,C):: C=C+4
110 IF C>25 THEN C=1 :: R=R+
8
120 NEXT CH :: CALL CLEAR ::
 C=0 :: R=-2 :: FOR CH=97 TO
 122 :: CALL SKINNY(CH,R,C):
: C=C+4
130 IF C>25 THEN C=1 :: R=R+
6
140 NEXT CH
150 GOTO 150
```

SPOOKY

CALL SPOOKY(A,B) will redefine the characters from ASCII A to B to a "spooky" form.

```
100 FOR CH=33 TO 122 :: PRIN
T CHR$(CH);:: NEXT CH :: CAL
L SPOOKY(33,122)
110 GOTO 110
```

SPRITETEXT

CALL SPRITETEXT(R,C,M1$,M2$) will convert the strings M1$ and M2$, of not more than 8 characters each, into 2 rows of magnified sprite characters displayed at dot-row R, dot-column C. Characters 128-139 are redefined, sprites #1-#4 are used. They may then be set in motion by a simultaneous CALL MOTION, recolored by CALL COLOR(#, redefined by a new CALL, etc.

```
100 CALL CLEAR :: CALL SCREE
N(5)
110 M1$="NUTS AND" :: M2$="B
OLTS #3" :: CALL SPRITETEXT(
10,10,M1$,M2$):: FOR D=1 TO
500 :: NEXT D :: CALL MOTION
(#1,5,5,#2,5,5,#3,5,5,#4,5,5
)
120 FOR T=1 TO 6 :: FOR J=1
TO 4 :: RANDOMIZE :: C=INT(1
5*RND+2):: ON (C=5)+2 GOTO 1
20,130
130 CALL COLOR(#J,C):: NEXT
J :: NEXT T
140 M1$="TIGERCUB" :: M2$="S
OFTWARE" :: CALL SPRITETEXT(
1,1,M1$,M2$):: GOTO 120
```

SQUAT

CALL SQUAT will reidentify the lower case letters to a wider form, rather crowded unless double-spaced.

```
100 CALL CLEAR :: FOR CH=97
TO 122 :: PRINT CHR$(CH)&" "
;:: NEXT CH :: CALL SQUAT
110 GOTO 110
```

THINLINE

CALL THINLINE(L) will read L lines (up to 6) of DATA from the main program, limited to upper case letters, numerals and common punctuation, and not over 7 characters and spaces per DATA item, and display them in greatly enlarged thin-line letters. DATA is read internally. ASCII 99 to 142 are reidentified.

```
100 DATA THIS IS,A DEMO,OF T
HE,THIN-,LINE,PROGRAM
110 RESTORE 100 :: CALL THIN
LINE(6):: FOR D=1 TO 300 ::
NEXT D :: CALL CLEAR
120 DATA THE,SECOND,TIME IS,
FASTER
130 RESTORE 120 :: CALL THIN
LINE(4)
140 GOTO 140
```

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

BACKDROP

CALL BACKDROP(F,B), where F is a foreground and B is a background color, will place a background on the screen consisting of ASCII 143, and color set 14. Text may then be superimposed on this background using DISPLAY AT, but be sure to follow each string with a semicolon. Each further CALL will cause the screen to change pattern. Try 11 and 12 for a shimmering golden screen. Contains an internal flag.

```
100 F=11 :: B=12 :: CALL BAC
KDROP(F,B):: DISPLAY AT(9,11
):"TIGERCUB";:: DISPLAY AT(1
2,11):"SOFTWARE";:: DISPLAY
AT(15,10):"NUTS & BOLTS";
110 CALL BACKDROP(F,B):: CAL
L KEY(0,K,S):: IF S<>0 THEN
STOP ELSE 110
```

BACKFORTH

CALL BACKFORTH(T,R) reads T number of DATA items from the main program and prints them to screen, centered, alternately left to right and right to left, on every other row starting at row R.

```
100 DATA THIS IS A DEMO,OF A
 PRINTING,ROUTINE,FROM THE,N
```

UTS & BOLTS #3 DISK
110 RESTORE 100 :: CALL BACK FORTH(5,8)

BURST and SLURP

CALL BURST(T,R) will read T number of DATA items from the main program and display them, starting in row R, spreading both ways from center.
CALL SLURP(T,R) with the same parameters will reverse the action to erase the lines. From routines by Roy Tamashiro.

```
100 DATA THIS IS A DEMONSTRA
TION,OF THE BURST SUBPROGRAM
,ON NUTS & BOLTS DISK #3,ADA
PTED FROM A ROUTINE,BY ROY T
AMASHIRO
110 RESTORE 100 :: CALL CLEA
R :: CALL BURST(5,5):: FOR D
=1 TO 200 :: NEXT D :: CALL
SLURP(5,5)
```

CURTAIN4

CALL CURTAIN4(CC) will very smoothly wipe the screen from left to right with color CC. Char set 14 is colored, characters 140-143 are redefined, sprites #1-#7 are called.

```
100 CALL CURTAIN4(14)
```

DIAGBAN

CALL DIAGBAN(M$,S,B,C) where M$ is any text, S is the screen color, B is the banner color and C is the sprite color, will display a banner diagonally across the screen and scroll M$ along it in enlarged sprite letters.

```
100 CALL CLEAR :: CALL DIAGB
AN("THIS IS A DEMONSTRATION
OF THE TIGERCUB NUTS & BOLTS
 DISK #3 DIAGBAN SUBPROGRAM"
,5,11,16)
```

DISPLAY

CALL DISPLAY(R,C,M$) where R and C are the beginning row and column and M$ is the text string, will simulate DISPLAY AT in full 32-column screen width, for use on monitors or TV screens which can display 32 columns.

```
100 CALL CLEAR :: M$="Now is
 the time for all good men t
o come to the aid of the par
ty." :: CALL DISPLAY(12,1,M$
)
```

DROPTITLE

CALL DROPTITLE(S,T,T$) will color the screen S and will drop the characters of T$, in magnified sprites of every color except S, downward from the top into a diagonal banner, with an audible clunK. Clear with CALL DELSPRITE(ALL).

```
100 CALL CLEAR :: CALL DROPT
ITLE(5,5,"NUTS & BOLTS #3")
110 GOTO 110
```

FLY

CALL FLY(M$,R) will cause the characters of M$ to zoom into position randomly from all sides and audibly, centered on row R.

```
100 CALL CLEAR :: CALL FLY("
TIGERCUB SOFTWARE",12)
```

JAWS

CALL JAWS(F,B) will place a pattern on the screen in F foreground and B backGround colors. Subsequent CALLs in a loop will animate the pattern. Text may be placed on the screen with DISPLAY AT (be sure to put a semicolon after the string) and will seem to float above the pattern. Has an internal flag.

```
100 CALL JAWS(16,5):: DISPLA
Y AT(9,11):"TIGERCUB";:: DIS
PLAY AT(12,11):"SOFTWARE";::
 DISPLAY AT(23,12):"Press an
y Key";
110 CALL JAWS(16,5):: CALL K
EY(0,K,S):: IF S=0 THEN 110
ELSE STOP
```

NEON

CALL NEON(M$,SC,COL,SP,T) will display text M$ of up to 28 characters in magnified sprites of color COL diagonally across a screen of color SC and will blinK the title T times at speed SP (try 1 to 5). Erase with CALL DELSPRITE(ALL).

```
100 CALL CLEAR :: CALL NEON(
"TIGERCUB",5,16,1,8)
```

NEON2

CALL NEON2 will put a border of colored lights around the screen, and each subsequent CALL will move the colors one step around. The subprogram BXB must be merged in before this CALL, because sets 11-16 are used for the colors. Has internal flag.

```
100 CALL CLEAR :: CALL SCREE
N(2):: FOR SET=2 TO 8 :: CAL
L COLOR(SET,16,1):: NEXT SET
 :: DISPLAY AT(8,10):"TIGERC
UB" :: DISPLAY AT(12,10):"SO
FTWARE"
110 FOR J=1 TO 200 :: CALL N
EON2 :: NEXT J
```

SCATTER

CALL SCATTER(M$,R) will randomly place the letters of M$ into position on row R, centered.

```
100 DATA THIS IS A,DEMONSTRA
TION OF,THE SCATTERPRINT SUB
PROGRAM,FROM THE,TIGERCUB SO
FTWARE,NUTS & BOLTS,DISK #3
110 CALL CLEAR :: R=3 :: FOR
 J=1 TO 7 :: READ A$ :: CALL
 SCATTER(A$,R):: R=R+2 :: NE
XT J
120 GOTO 120
```

SLIDE

CALL SLIDE(S,C,M$), where S is the screen color and C is the sprite color, will color the screen and display M$ of up to 28 characters in magnified sprites diagonally across the screen, and then rapidly slide them off. Contains internal flag. Requires Memory Expansion.

```
100 DATA TIGERCUB SOFTWARE,N
UTS & BOLTS,DISK NO. 3,SLIDE
 SUBPROGRAM
110 CALL CLEAR :: FOR J=1 TO
 4 :: READ M$ :: CALL SLIDE(
5,14,M$):: NEXT J :: RESTORE
 :: GOTO 110
```

SQUIRMY

CALL SQUIRMY will place a hypnotic squirming pattern on the screen, which will continue to squirm if the CALL is repeated. Text can be placed on the screen between CALLs, with DISPLAY AT, but be sure to follow the text with a semicolon. Characters 142-143 are redefined, char set 14 is colored. Has internal flag.

```
100 CALL CLEAR :: CALL SQUIR
MY
110 DISPLAY AT(5,6):"TIGERCU
B SOFTWARE";:: DISPLAY AT(8,
7):"NUTS & BOLTS #3";:: DISP
LAY AT(22,7):"PRESS ANY KEY"
;
120 CALL SQUIRMY :: CALL KEY
(0,K,S):: IF S=0 THEN 120
```

UPSCROLL

CALL UPSCROLL(M$,L) will scroll strings, of not more than 28 characters, up the lower L lines of the screen. At each CALL, M$ is displayed in row 24 and the previous L-1 strings are each moved one row higher, the uppermost disappearing.

```
100 DISPLAY AT(12,1)ERASE AL
L:"FILENAME? DSK" :: ACCEPT
AT(12,14)BEEP:F$ :: CALL CLE
AR
110 OPEN #1:"DSK"&F$,INPUT
120 LINPUT #1:M$
130 CALL UPSCROLL(M$,13)
140 IF EOF(1)<>1 THEN 120 EL
SE CLOSE #1 :: END
```

WALKING

CALL WALKING(L,R,S) will read L items of text from DATA in the main program and cause it to "walK" across the screen on alternate lines from row R downward, at S spaces per step.

```
100 DATA TIGERCUB SOFTWARE,N
UTS & BOLTS #3,WALKING TITLE
 SUBPROGRAM
110 RESTORE 100 :: CALL CLEA
R :: CALL COLOR(14,11,11)::
CALL HCHAR(9,1,143,224):: CA
LL WALKING(3,10,3)
```

WINDOW

CALL WINDOW(R,C,L,T,F), if F=0, will save the contents of, and clear, a window beginning at row R and continuing for T lines, from column C and L characters in length. Text can then be displayed in this window by DISPLAY AT (use a semicolon after each text to avoid blanking out the rest of the line).
CALL WINDOW again with F equal to 1 and any dummy values in the other parameters, will quickly restore the original text or graphics in the window.
After the first CALL with F equal to 0 has saved the window contents, it can be cleared very quickly by a CALL with F equal to 2 and any dummy values in other parameters.

```
100 CALL WINDOW(12,3,10,5,0)
:: DISPLAY AT(13,4):"TIGERCU
B";:: DISPLAY AT(15,4):"SOFT
WARE";:: FOR D=1 TO 500 :: N
EXT D :: CALL WINDOW(0,0,0,0
,2)
110 DISPLAY AT(13,4):"NUTS &
";:: DISPLAY AT(15,4):"BOLTS
 #3";:: FOR D=1 TO 500 :: NE
XT D :: CALL WINDOW(0,0,0,0,
1)
120 GOTO 120
```

>>>>>>>> FORMATTING <<<<<<<<

DELETE

CALL DELETE(M$(),J,T,N) where M$ is an array such as a TI-Writer text in tabular format, J is a subscript number, T is a tab position and N is the number of characters from the tab position to be deleted, will delete that part of the string. If placed in a loop, such as
FOR J=1 TO 100 :: CALL DELETE(M$(),J,30,5):: NEXT J, it will delete a column from a tabular file.

FILL

CALL FILL(M$) is the same as CALL READER except that it also inserts blanks between words as necessary to justify the right margin.

```
100 CALL CLEAR :: OPEN #1:"D
SK1.S" !(any readable file)
110 IF EOF(1)=1 THEN CLOSE #
1 ELSE LINPUT #1:M$ :: CALL
FILL(M$):: GOTO 110
```

JUSTIFY

CALL JUSTIFY(M$,R,C) will print M$ on the screen at row R, right-justified to column C providing the length of M$ is less than C.

```
100 CALL CLEAR :: C=25 :: FO
R R=1 TO 24 :: M$=RPT$("*",2
4*RND+1):: CALL JUSTIFY(M$,R
,C):: NEXT R
110 GOTO 110
```

JUSTIFY_N

CALL JUSTIFY_N(J,N,C,T,L,R,N$),
If J=1, T will be the tab setting to print N right-justified on column C.
If J=2, T will be the tab to print N with the decimal on column C.
If J=3, the decimal portion of N will be zero-filled or truncated to R places, and T will be the tab to print the string N$ right-justified on column C.
If J=4, the integer portion of N will also be zero-filled at right to L places, the decimal portion will be zero-filled or truncated to R places, and T will be the tab to print N$ right-justified on column C.

```
100 CALL CLEAR
110 INPUT "N? ":N :: INPUT "
J? ":J :: INPUT "C? ":C :: I
F J>2 THEN INPUT "R? ":R ::
IF J=4 THEN INPUT "L? ":L
120 CALL JUSTIFY_N(J,N,C,T,L
,R,N$)
130 IF J<3 THEN PRINT TAB(T)
;N ELSE PRINT TAB(T);N$
```

READER

CALL READER(M$) will display a string M$ on the screen without breaking words, from row 1 downward on successive CALLs, prompting and waiting for a keypress on line 24 and then returning to line 1.

```
100 CALL CLEAR :: OPEN #1:"D
SK (any DISPLAY file)",INPUT
110 LINPUT #1:M$ :: CALL REA
DER(M$):: IF EOF(1)<>1 THEN
110 :: CLOSE #1
```

REDUCE

CALL REDUCE(N(),Y,L) where N() is a numeric array of Y items, will reduce the largest value to L and all other values in proportion, so that they may be graphed on a screen or printer of L maximum width.

```
100 CALL CLEAR :: DIM N(24):
: FOR J=1 TO 24 :: N(J)=100*
RND :: NEXT J :: CALL REDUCE
(N(),24,28)
110 FOR J=1 TO 24 :: DISPLAY
 AT(J,1):RPT$(CHR$(30),N(J))
:: NEXT J
120 GOTO 120
```

>>>>>>>>> PLOTTING <<<<<<<<<

CIRCLE

CALL CIRCLE(RD,R,C,CH) where RD is the radius in row/column spaces, R and C are the row/column of the center of the circle, and CH is the ASCII character to be used, will plot an approximate circle on the screen. Maximum parameters for a full circle are RD=11, R=12 and C between 12 and 20.

```
100 CALL CHAR(42,RPT$("F",16
))
110 CALL CLEAR :: CALL CIRCL
E(11,12,14,42)
120 GOTO 120
```

CIRCLER

CALL CIRCLER(S,CH,R), where S is the screen color, CH is the character to be used for sprites, and R is a radius value between 25 and 87, will plot a perfect circle of sprites in all colors except the screen color. Magnified sprites can be used but may partially erase each other. For sprites of all one color, change the C in the CALL SPRITE parameter to the color value desired. Clear the screen with CALL DELSPRITE(ALL).

```
100 CALL CLEAR :: S=2 :: CAL
L CHAR(40,RPT$("F",64))
110 FOR J=87 TO 25 STEP -8 :
: :: CALL MAGNIFY(2):: CALL
CIRCLER(S,40,J):: S=S+1-ABS(
S=16)*15 :: CALL DELSPRITE(A
LL):: NEXT J
```

DOTPLOT LINEPLOT DIAGPLOT BOXPLOT CIRCPLOT

DOTPLOT is to be merged in; LINEPLOT, DIAGPLOT, BOXPLOT and CIRCPLOT are included within it, cannot be merged in separately. Developed from a routine by Dwight Klettke.
CALL DOTPLOT(X,Y) will place a single dot on the screen at Cartesian coordinates X,Y.
CALL LINEPLOT(X1,Y1,X2,Y2) will draw a horizontal or vertical line from coordinates X1,Y1 to X2,Y2. Do not use for diagonals.
CALL DIAGPLOT(X1,Y1,X2,Y2) will draw a diagonal line from X1,Y1 to X2,Y2. Do not use for horizontal or vertical lines.
CALL BOXPLOT(X1,Y1,X2,Y2) wil draw a rectangle with the lower left corner at X1,Y1 and upper right corner at X2,Y2.
CALL CIRCPLOT(R,X1,Y1) will draw a circle of R dots radius with axis at X1,Y1.
 All measurements are in dotrows and dotcolumns, as used for sprites, and based on a Cartesian grid. Characters are redefined from 143 downward, as needed.

```
100 CALL CLEAR :: CALL LINEP
LOT(-20,0,20,0):: CALL LINEP
LOT(0,-20,0,20)
110 CALL DIAGPLOT(-20,-20,20
,20):: CALL DIAGPLOT(-20,20,
20,-20)
120 CALL BOXPLOT(10,10,-10,-
10)
130 CALL CIRCPLOT(30,1,1)
140 GOTO 140
```

FASTPLOT

CALL FASTPLOT(R,C) will place a single pixel at dotrow R, dotcolumn C. It is faster than DOTPLOT but still slow, and it sKips over previous lines; also, R and C must be integers. From a routine by Peter BrooKs.

```
100 CALL CLEAR :: C=100 :: F
OR R=50 TO 70 :: CALL FASTPL
OT(R,C):: NEXT R :: FOR C=10
0 TO 170 :: CALL FASTPLOT(R,
C):: NEXT C
110 FOR R=70 TO 50 STEP -1 :
: CALL FASTPLOT(R,C):: NEXT
R :: FOR C=170 TO 100 STEP -
1 :: CALL FASTPLOT(R,C):: NE
XT C
120 GOTO 120
```

>>> JOYSTICK AND KEYBOARD <<

CHARMOVE

CALL CHARMOVE(R,C,CH<K,G) when placed directly after a CALL KEY(3,K,S) will move the character CH in the direction of the W, E, R, S, D, Z. X or C Key that is pressed, without erasing any text or graphics that it passes over. The current row and column are returned in R and C, the ASCII temporarily replaced is returned in G and can be used for coincidence checKing. Before entering this routine, CALL GCHAR(R,C,G) is necessary to obtain the initial value for G, and CALL HCHAR(R,C,CH) to initially place the character on the screen.
List this example before running it, to put something on the screen. The screen will flash every time the cursor runs over a C.

```
100 R,C=12 :: CH=143 :: CALL
 CHAR(CH,RPT$("F",16)):: CAL
L GCHAR(R,C,G):: CALL HCHAR(
R,C,CH)
110 CALL KEY(3,K,S):: CALL C
HARMOVE(R,C,CH,K,G):: IF G=6
7 THEN CALL SCREEN(11):: CAL
L SCREEN(8):: GOTO 120 ELSE
GOTO 120
```

DOODLE

CALL DOODLE(R,C,K) after a CALL HCHAR(R,C,CH) and a CALL KEY(3,K,S) will return in R and C the values to be reinput to the CALL_HCHAR to move the character in the direction of the W, E, R, S, D, Z. X and C Keys and doodle on the screen.

```
100 CALL CLEAR :: R,C=12
110 CALL HCHAR(R,C,42)
120 CALL KEY(3,K,S):: CALL D
OODLE(R,C,K):: GOTO 110
```

To run the cursor around the screen without doodling, change lines 100-110 to -

```
100 CALL CLEAR :: R,C,R2,C2=
12
110 CALL HCHAR(R2,C2,32):: C
ALL HCHAR(R,C,42):: R2=R ::
C2=C
```

JOYKEY

CALL JOYKEY(X,Y) when placed immediately after a CALL JOYST and with the two variables the same as used in the CALL JOYST, will accept Keyboard input of arrow Keys and WRZC Keys and convert the input to equivelant joystick values in X and Y.
Permits optional use of either joysticK or Keyboard, but results will vary depending on type of programming.

JOYMOUSE

CALL JOYMOUSE(M$) will permit movement of a pointer with joysticK #1 and will return in M$ the characters on the screen immediately to the right of the pointer, up to the first blanK space, when the fire button is pressed. List the program to put something on the screen before running this demo.

```
100 CALL JOYMOUSE(M$):: PRIN
T M$ :: GOTO 100
```

KEYJOY

CALL KEYJOY(K), when placed immediately after a CALL KEY (before any "IF S=0 THEN...) and using the variable used for Key in the CALL KEY, will accept input from joystick #1 and convert it to the equivelant ASCII of the Key. This permits optional use of either Keyboard or joysticK, but results will vary depending on type of programming.

KSPRITE

CALL KSPRITE(SP,K,T) placed immediately after a CALL KEY(3,K,S) will set a sprite #S (previously called in main program) in motion at speed T, in the direction K from the W, E, R, S, D, Z, X and C input.

```
100 CALL CLEAR :: CALL MAGNI
FY(2):: SP=1 :: CALL SPRITE(
#SP,42,11,50,50)
110 CALL KEY(3,K,S):: CALL K
SPRITE(SP,K,10):: GOTO 110
```

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

DEC_FRAC

CALL DEC_FRAC(D,B,C,F$), where D is any positive or negative number, integer or non-integer, will return the numerator in B, the denominator in C, and the fraction in string form in F$.

```
100 CALL CLEAR
110 INPUT "Decimal number? "
:D :: CALL DEC_FRAC(D,B,C,F$
):: PRINT F$;" ";B;"/";C ::
GOTO 110
```

MARKDOWN

CALL MARKDOWN(D,P,M) will return in M the marKdown price of an item regularly priced at P and discounted by D percent.

```
100 CALL CLEAR
110 INPUT "Regular price? ":
P :: INPUT "Discount percent
age? ":D :: CALL MARKDOWN(D,
P,M):: PRINT "MarKdown price
 is";M :: GOTO 110
```

MARKUP

CALL MARKUP(W,P,S) will return in S the retail price of an item with a wholesale price of W and a marKup percentage of P.

```
100 CALL CLEAR
110 INPUT "Wholesale price?
":W :: INPUT "MarKup percent
age? ":P :: CALL MARKUP(W,P,
S):: PRINT "Sale price is";S
 :: GOTO 110
```

MAXMIN

CALL MAXMIN(L,H), where L and H are two numeric values, will return the lower in L and the higher in H. This is a combination of the MAX and MIN functions.

```
100 INPUT "LOW? ":L :: INPUT
 "HIGH? ":H :: CALL MAXMIN(L
,H):: PRINT "LOW=";L:"HIGH="
;H :: GOTO 100
```

MEDIAN

CALL MEDIAN(N(),Y,M) where N() is a sorted numeric array containing Y items, will return the median in M.

```
100 CALL CLEAR :: DIM N(100)
:: FOR J=1 TO 99 :: N(J)=N(J
-1)+10*RND :: NEXT J
110 CALL MEDIAN(N(),99,M)::
PRINT M
```

MF
(Contains 20 subprograms)

MERGE DSK1.MF will merge in a set of 20 mathematical subprograms. These are all listed in the TI Extended Basic manual as DEFs, but they can be accessed faster as subprograms. The value is entered as X and returned as Y.
CALL SEC(X,Y) gives the se-cant. CALL CSC(X,Y) gives the cosecant. CALL COT(X,Y) gives the cotangent. CALL ARCSIN(X,Y) gives the inverse sine. CALL ARCOS(X,Y) gives the inverse cosine. CALL ARCSEC(X,Y) gives the inverse secant. CALL ARCCSC (X,Y) gives the inverse cosecant. CALL ARCCOT(X,Y) gives the inverse cotangent. CALL SINH(X,Y) gives the hyperbolic sine. CALL COSH (X,Y) gives the hyperbolic cosine. CALL TANH(X,Y) gives the hyperbolic tangent. CALL SECH(X,Y) gives the hyperbolic secant. CALL CSCH(X,Y) gives the hyperbolic cosecant. CALL COTH(X,Y) gives the hyperbolic tangent. CALL ARCSINH(X,Y) gives the inverse hyperbolic sine. CALL ARCCOSH(X,Y) gives the inverse hyperbolic cosine.

CALL ARCTANH(X,Y) gives the inverse hyperbolic tangent. CALL ARCSECH(X,Y) gives the inverse hyperbolic secant. CALL ARCCSCH(X,Y) gives the inverse hyperbolic cosecant and CALL ARCCOTH(X,Y) gives the inverse hyperbolic cotangent.

MOD

CALL MOD(A,B,C) will return in C the remainder from the division of B into A. This simulates the MOD function of some other Basic languages.

```
100 CALL CLEAR
110 INPUT "NUMBER? ":N :: IF
 N<>INT(N)THEN PRINT "WHOLE
NUMBER, PLEASE" :: GOTO 110
120 INPUT "DIVIDED BY? ":N2
:: IF N2<>INT(N2)THEN PRINT
"WHOLE NUMBER, PLEASE" :: GO
TO 120 ELSE IF N2>N THEN PRI
NT "LESS THAN";N;",PLEASE" :
: GOTO 120
130 CALL MOD(N,N2,R):: PRINT
 "REMAINDER IS";R :: GOTO 11
0
```

RANDOM

CALL RANDOM(A,B,X) will return in X a random integer between A and B, which may be either positive or negative numbers providing that A is the lesser (i.e., -10 is less than -9).

```
100 CALL CLEAR :: INPUT "Low
er number? ":A :: INPUT "Hig
her number? ":B :: IF B<A TH
EN 100
110 CALL RANDOM(A,B,X):: PRI
NT X;:: GOTO 110
```

ROUND

CALL ROUND(N,R), where N is a number to be rounded off and R is the number of places to be rounded to, will return the rounded number in N. If R is a positive number, N will be rounded to the right of the decimal; if it is negative, N will be rounded to the left of the decimal. Credited to Terry Atkinson.

```
100 CALL CLEAR
110 INPUT "NUMBER? ":N
120 INPUT "ROUND TO HOW MANY
 PLACES? ":R
130 CALL ROUND(N,R):: PRINT
N :: GOTO 110
```

TOL

CALL TOL(X,Y,L,M,V) will return a value of 1 for V if Y is not more than L less than X nor more than M more than X. Used to accept values where a known tolerance of error is acceptable.

```
100 CALL CLEAR :: INPUT "TRU
E VALUE? ":X :: INPUT "ACCEP
TABLE TOLERANCE BELOW? ":L :
: INPUT "ACCEPTABLE TOLERANC
E ABOVE? ":M
110 INPUT "INPUT VALUE? ":Y
:: CALL TOL(X,Y,L,M,V):: IF
V THEN PRINT "WITHIN TOLERAN
CE" ELSE PRINT "OUT OF TOLER
ANCE"
120 GOTO 110
```

TOLP

CALL TOLP(X,Y,L,M,V) will return a value of 1 for V if Y is not more than L percent less nor more than M percent more than X. Used to accept values where a known percentage of error is acceptable.

```
100 CALL CLEAR :: INPUT "TRU
E VALUE? ":X :: INPUT "ACCEP
TABLE PERCENTAGE OF TOLERANC
E BELOW? ":L :: INPUT "ACCEP
TABLE PERCENTAGE OF TOLERANC
E ABOVE? ":M
110 INPUT "INPUT VALUE? ":Y
:: CALL TOLP(X,Y,L,M,V):: IF
 V THEN PRINT "WITHIN TOLERA
NCE" ELSE PRINT "OUT OF TOLE
RANCE"
120 GOTO 110
```

TWOS

CALL TWOS(B$,T$) will return the twos complement T$ of the binary B$.

```
100 INPUT "BINARY NUMBER? ":
B$ :: CALL TWOS(B$,T$):: PRI
NT "TWOS COMPLEMENT IS ";T$
:: GOTO 100
```

VARIANCE

CALL VARIANCE(N(),Y,SV,PV) where N() is a numeric array and Y is the number of items in the array, will return the sample variance in SV and the population variance in PV. The standard deviation can then be determined by SQR(SV) or SQR(PV).

```
100 CALL CLEAR :: RANDOMIZE
:: DIM N(100):: FOR J=1 TO 1
00 :: N(J)=1*RND :: X=X+N(J)
:: NEXT J
110 CALL VARIANCE(N(),100,SV
,PV):: PRINT SV;PV

E BELOW? ":L :: INPUT "ACCEP
TABLE PERCENTAGE OF TOLERANC
E ABOVE? ":M
110 INPUT "INPUT VALUE? ":Y
:: CALL TOLP(X,Y,L,M,V):: IF
 V THEN PRINT "WITHIN TOLERA
NCE" ELSE PRINT "OUT OF TOLE
RANCE"
120 GOTO 110
```

>>>>>>> TIME AND DATE <<<<<<

BETWEEN

CALL BETWEEN(D1,D2,B) will return in B the number of days between Julian dates D1 and D2. To find the number of days between two calendar dates, use the CALJUL subprogram to convert them to Julian -

```
100 INPUT "1st Julian date?
":D1 :: INPUT "2nd Julian da
te? ":D2 :: CALL BETWEEN(D1,
D2,B):: PRINT B;"DAYS BETWEE
N" :: GOTO 100
```

CALJUL

CALL CALJUL(M,D,Y,JD) where M is a month number, D is a date and Y is a 4-digit year, will return the Julian date in JD in the form 87365 where the first two digits are the last digits of the year and the other digits are the numeric date.

```
100 CALL CLEAR
110 DISPLAY AT(3,1):"Month n
umber?" :: ACCEPT AT(3,15)SI
ZE(2)VALIDATE(DIGIT):M :: IF
 M<1 OR M>12 THEN 110
120 DISPLAY AT(5,1):"Date?"
:: ACCEPT AT(5,7)SIZE(2)VALI
DATE(DIGIT):D :: IF D<1 OR D
>31 THEN 120
130 DISPLAY AT(7,1):"Year?"
:: ACCEPT AT(7,7)SIZE(4)VALI
DATE(DIGIT):Y
140 CALL CALJUL(M,D,Y,JD)::
DISPLAY AT(10,1):"Julian dat
e is";JD :: GOTO 110
```

JULCAL

CALL JULCAL(JD,M$,D,Y), where JD is a Julian date in the 1900's in the form 87365, will return the month, day and year in M$, D and Y. This subprogram RESTOREs and READs internal DATA the first time it is CALLed.

```
100 CALL CLEAR
110 INPUT "Julian date?":JD
:: CALL JULCAL(JD,M$,D,Y)::
PRINT M$;D;Y :: GOTO 110
```

TIMEBAR

CALL TIMEBAR(T,K) will fill a time bar along the right edge of the screen at speed T until any key is pressed, then will return the ASCII of the key in K, or -1 if no key is pressed by the time the bar reaches the top. ASCII 136-143 are redefined. Contains internal flag.

```
100 CALL CLEAR
110 CALL TIMEBAR(1,K):: PRIN
T K :: GOTO 110
```

TIMESTART and TIMEREAD

CALL TIMESTART will start a timer; CALL TIMEREAD(M,S) will return the minutes in M and the seconds in S since it was started. Accurate for short periods, fairly accurate up to 10 minutes, 56 seconds depending on complexity of math operations performed by the computer in the interim. TIMEREAD is imbedded, does not have to be merged in. Uses sprite #1. Based on a routine by Ian Hakanson of TIUP.

```
100 CALL CLEAR
110 DISPLAY AT(12,1):"HOW MU
CH IS";INT(10*RND+2);"DIVIDE
D BY";INT(10*RND+2):: CALL T
IMESTART :: ACCEPT AT(13,1):
V :: CALL TIMEREAD(M,S)
120 DISPLAY AT(14,1):"RESPON
SE TIME=";M;"MINUTES";S;"SEC
ONDS" :: GOTO 110
```

>>>>> INPUT AND ACCEPT <<<<<

ACCEPTER

CALL ACCEPTER(N,M$) accepts either a numeric value N or a string M$ without crashing; if a string is input, N will have a value of 0. Contains ON ERROR. Example -

```
100 CALL CLEAR
110 CALL ACCEPTER(N,M$):: ON
 (N=0)+2 GOTO 120,130
120 A$=A$&M$ :: PRINT A$ ::
GOTO 110
130 NN=NN*10+N :: PRINT NN :
: GOTO 110
```

CALLNUM

CALL CALLNUM(R,C,L,N) will acept input of a number of exactly L digits in length, displaying it on row R, column starting at C, and return the value in N, without pressing Enter.

```
100 CALL CLEAR
110 DISPLAY AT(12,1):"TYPE A
 6-DIGIT NUMBER" :: CALL CAL
LNUM(12,22,6,N):: DISPLAY AT
(14,22):N :: GOTO 110
```

CALLTEXT

CALL CALLTEXT(R,C,L,M$) will accept a string of exactly L characters in length, at row R column C, and return it in M$, without pressing Enter.

```
100 DISPLAY AT(10,1)ERASE AL
L:"TYPE A 7-LETTER WORD"
110 CALL CALLTEXT(12,1,7,M$)
:: DISPLAY AT(14,1):M$ :: GO
TO 110
```

CHORD

CALL CHORD(P$,M$), where P$ is an input prompt (may be omitted if P$ is a null string), will accept LINPUT of M$ with a C chord instead of a beep.

```
100 P$="NAME? "
110 CALL CHORD(P$,M$):: PRIN
T M$ :: GOTO 110
```

CODE

CALL CODE(R,C,C$) will ACCEPT at row R, column C a string of integer values, such as printer control codes, separated by single spaces, and return them as an ASCII string C$. If any of the input is not numeric, "ERROR" will be flashed at R,C and C$ will be a null string.
Example - try inputting 27 52 27 14 27 71 27 45 1

```
100 OPEN #1:"PIO"
110 CALL CODE(23,1,C$):: IF
LEN(C$)>0 THEN PRINT #1:C$&"
THIS IS A TEST" :: GOTO 110
ELSE 110
```

KINPUT

CALL KINPUT(R,C,K) will simulate a blinking cursor at row R, column C, and will return in K the ASCII value of whatever key is pressed. Can be used for a single-character INPUT without the need to press Enter.

```
100 CALL CLEAR :: DISPLAY AT
(12,1):"QUIT? (Y/N)" :: CALL
 KINPUT(12,15,K):: IF K=89 T
HEN STOP ELSE 100
```

FIELD

CALL FIELD(R,C,P$,L,M$) where R and C are row and column, P$ is an input prompt, and L is the maximum number of characters acceptable, will display the prompt at row and column, followed by a white field of length L, and will ACCEPT a string input of up to that length and return it in M$. Char set 14 is colored white on white, ASCII 143 is blank.

```
100 CALL CLEAR :: CALL FIELD
(12,3,"YES OR NO?",3,M$)
```

FIELDN

CALL FIELDN(R,C,P$,L,N), where R and C are row and column, P$ is an input prompt, and L is the maximum number of characters acceptable, will display the prompt at row and column followed by a white field of length L, and will ACCEPT a numeric input (validated NUMERIC) of up to that field length and return it in N. Char set 14 is colored white on white, char 143 is blank.

```
100 CALL CLEAR :: CALL FIELD
N(12,3,"YEAR?",4,N)
```

LONGACCEPT

CALL LONGACCEPT(R,M$) will accept, at row R and subsequent rows, strings up to 28 characters long, and combine them into a string up to 254 characters long or until Enter is pressed before column 28. A guideline shows the remaining number of characters acceptable.

```
100 CALL CLEAR :: CALL LONGA
CCEPT(12,M$):: PRINT M$ :: E
ND
```

PRINTCODE

CALL PRINTCODE(P$) will accept numeric inputs between 0 and 99, until a negative number is input, and convert them into an ASCII string PS which can then be output to a printer or otherwise used to create ASCII strings. Example - try inputting
27,66,2,27,87,1,27,45,1,-1

```
100 CALL PRINTCODE(P$):: OPE
N #1:"PIO" :: PRINT #1:P$&"T
HIS IS A DEMONSTRATION OF TH
E NUTS & BOLTS #3 PRINTCODE
SUBPROGRAM"
```

SILENT

CALL SILENT(P$,M$), where P$ is an optional prompt, will accept LINPUT of M$ without the beep.

```
100 P$="NAME? "
110 CALL SILENT(P$,M$):: PRI
NT M$ :: GOTO 110
```

>>>>>> STRING HANDLING <<<<<

ADVERB

CALL ADVERB(M$,ADV$), where M$ is any adjective, will return the adverbial form in ADV$, in upper or lower case as input.

```
100 CALL CLEAR
110 INPUT "Adjective? ":M$ :
: CALL ADVERB(M$,ADV$):: PRI
NT "Adverb form is ";ADV$ ::
 GOTO 110
```

CONVERT

CALL CONVERT(A$,B$) will convert any lower case characters in A$ to upper case, and return the converted string in B$.

```
100 A$="This subprogram conv
erts any lower case letters
to upper case" :: PRINT A$
110 CALL CONVERT(A$,B$):: PR
INT B$ :: STOP
```

ED

CALL ED(M$,ED$) will add the proper past tense suffix to any verb M$ which has a standard past tense form, and return it in ED$, in upper or lower case as input. It cannot handle the many verbs, such as RUN, FLY, SEE, etc. which have an irregular past tense.

```
100 CALL CLEAR
110 INPUT "Verb? ":M$ :: CAL
L ED(M$,ED$):: PRINT "If thi
s is a regular verb,":"the p
ast tense is ";ED$ :: GOTO 1
10
```

ING

CALL ING(M$,ING$) will add the proper "ing" suffix to any verb M$ and return it in ING$ in upper or lower case as input.

```
100 CALL CLEAR
110 INPUT "VERB? ":M$ :: CAL
L ING(M$,ING$):: PRINT ING$
:: GOTO 110
```

LAST

CALL LAST(M$,T$,P) will return in P the starting position of the last occurrence of substring T$ in string M$; it is the opposite of the Extended Basic POS which returns the starting position of the first occurrence.
To find the last occurrence at some point before the end of the string, truncate the string before CALLing this subprogram. For instance, to find the last occurrence of " " in the first 50 characters of M$,

```
V$=SEG$(M$,1,50):: CALL LAST
```

```
(Y$," ",P)

100 M$="NOW IS THE TIME FOR
ALL GOOD MEN TO COME TO THE
AID OF THEIR PARTY" :: DISPL
AY AT(3,1)ERASE ALL:M$
110 DISPLAY AT(12,1):"STRING
 TO SEARCH FOR?" :: ACCEPT A
T(14,1):T$
120 CALL LAST(M$,T$,P):: DIS
PLAY AT(18,1):P :: GOTO 110
```

MAXMINSTR

CALL MAXMINSTR(L$,H$), where L$ and H$ are two strings, will return in L$ the lower in alphabetic rank and the higher in H$. This is the string equivelant of MAXMIN.

```
100 CALL CLEAR
110 INPUT "LOW? ":L$ :: INPU
T "HIGH? ":H$ :: CALL MAXMIN
STR(L$,H$):: PRINT "LOW IS "
;L$:"HIGH IS ";H$ :: GOTO 11
0
```

MID3

CALL MID3(X$,A,Y$) replaces one word in string X$ beginning at position A, with Y$ string of one or more words, even if strings differ in length. Example - try 6 with WAS, 1 with THAT, 9 with ANOTHER, 6 with WILL BE.

```
100 CALL CLEAR :: X$="THIS I
S A TEST"
110 PRINT X$ :: INPUT "Posit
ion? ":A :: INPUT "String? "
:Y$ :: CALL MID3(X$,A,Y$)::
```

SEG

CALL SEG(M$,S$,L,F,R$) is the equivelant of the LEFT$, MID$ and RIGHT$ functions of other BASIC languages.
If S$="L", R$ will be the leftmost L characters of M$.
If S$="R", R$ will be the rightmost L characters of M$.
If S$="M", R$ will be a string of L characters from M$ starting at position F.

```
100 M$="123456789" :: PRINT
M$ :: INPUT "L - M - or R? "
:S$ :: INPUT "SIZE? ":L :: I
F S$="M" THEN INPUT "FROM? "
:F
110 CALL SEG(M$,S$,L,F,R$)::
 PRINT R$ :: GOTO 100
```

TRANSLIT

CALL TRANSLIT(M$,CH,Y$) will redefine characters, in ASCII sequence starting with CH, to the characters contained in M$, and will return in Y$ the sequence of characters which will then print M$. If a character appears more than once in M$, it is only redefined once. Useful to transfer characters from one set to another so that sets can be used for graphics, color, etc.

```
100 M$="TIGERCUB SOFTWARE" :
: CALL TRANSLIT("TIGERCUB SO
FTWARE",33,Y$):: CALL CLEAR
:: PRINT Y$ :: FOR D=1 TO 50
0 :: NEXT D :: END
```

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

ARRAYFIND

CALL ARRAYFIND(L,M$(),P$,P) will rapidly find the subscript number P of P$ in a presorted array M$() of L items, or return a value of 0 in P if not found. This is a very fast binary search for large arrays, but they must be in alphabetic ascending sequence.

```
100 FOR J=1 TO 10 :: M$(J)=C
HR$(J+64):: NEXT J :: CALL C
LEAR
110 INPUT "LETTER TO FIND? (
A - J)":P$ :: CALL ARRAYFIND
(10,M$(),P$,P):: PRINT P ::
GOTO 110
```

ARRAYSORT

CALL ARRAYSORT(N,X,N$(,),F,@$) will perform a sort on element F of a 2-dimensional array N$(,) having N records of X elements each.
If @$="N" the sort will be numeric, but all data in that field must be numeric. Contains ON ERROR. If X is more than 10, the internal variable T$ in line 21817 must be DIMensioned.

```
100 DATA GEORGE,YOUNG,222 AR
LINGTON,ANNISTON,TEXAS,39845
110 DATA BILL,CHANDLER,56 CO
RTEZ ST.,DULUTH,MINNESOTA,55
081
120 DATA ROBERT,ANDREWS,RT #
4,DOVER,NEW YORK,09675
130 DATA GEORGIA,TRAIL,56 MA
GNOLIA,WILLIAMS,ALABAMA,4888
7
140 DATA HARRY,PAU,67 HAPALI
AU,LAUNA,HAWAII,98779
150 FOR J=1 TO 5 :: FOR K=1
TO 6 :: READ A$(J,K):: PRINT
 A$(J,K);" ";:: NEXT K :: PR
INT :: NEXT J :: PRINT
155 INPUT "SORT ON WHICH FIE
LD? (1-6)":F :: IF F<>INT(F)
OR F<1 OR F>6 THEN 155 :: IF
 F=6 THEN @$="N"
160 CALL ARRAYSORT(5,6,A$(,)
,F,@$)
170 PRINT :: FOR J=1 TO 5 ::
 FOR K=1 TO 6 :: PRINT A$(J,
K);" ";:: NEXT K :: PRINT ::
 NEXT J :: GOTO 155
```

CHECKFILE

CALL CHECKFILE(F,F$,Q$), where F is a file number and F$ is a filename to be opened for output, will first open the file for update in order to check if it already containd data; if so, an inquiry will be made as to whether the file should be opened and the response will be returned in Q$ as "Y" or "N". The input file will be closed and, if response was "Y", the output file will be opened.

```
100 INPUT "FILE NUMBER? ":F
:: INPUT "FILENAME? DSK":F$
:: CALL CHECKFILE(F,F$,Q$)::
 IF Q$="N" THEN 100 ELSE STO
P
```

CLOSEUP

CALL CLOSEUP(M$(),N) where M$() is a string array of N items, will close up the array by eliminating any null strings.

```
100 CALL CLEAR :: DIM A$(40)
:: FOR J=1 TO 26 :: A$(J)=CH
R$(J+64):: PRINT A$(J);" ";:
: NEXT J :: PRINT
110 DISPLAY AT(12,1):"DELETE
 WHICH LETTER?(A-Z or Enter)
" :: DISPLAY AT(14,1):CHR$(2
55)
120 ACCEPT AT(14,1)SIZE(-1)V
ALIDATE(UALPHA,CHR$(255)):D$
 :: IF D$=CHR$(255)THEN 140
130 FOR J=1 TO 26 :: IF A$(J
)=D$ THEN A$(J)="" :: GOTO 1
10
140 NEXT J :: FOR J=1 TO 26
:: PRINT A$(J);" ";:: NEXT J
 :: PRINT : :"CLOSING UP" ::
 CALL CLOSEUP(A$(),26)
150 FOR J=1 TO 26 :: PRINT A
$(J);" ";:: NEXT J
```

FIELDSAVE

CALL FIELDSAVE(FN,N,F$()) will read DATA items from the main program and combine them into an array of tabbed strings of up to 254 characters which can then be dumped to a printer in tabular format (within printer width limits), or sorted on any field by FIELDSORT, or selectively recovered from any field by FIELDPICK.
F$() is the array of strings to be created, N is the number of strings, FN is the number of tabbed fields in each string, and the array F() must contain the tab positions in sequence plus one more value indicating the maximum allowable length of the string. Data elements are padded with blanks to fill the space between tab positions, or are truncated if longer than the allowed space.
If the maximum string length desired is not more than 80, this array can more easily be set up using the TI-Writer Editor, in which case it should be SAVEd by PF with the C option, rather than SF.
Merge in FIELDSAVE, FIELDSORT and FIELDPICK before running this demo.

```
100 DATA JOHN,JONES,100 MAIN
 ST.,ANYTOWN,HOMESTATE,99999
,SLIM,RAMBLER,45 COLT RUN,EL
 PASO,TX,58465
110 DATA WILLIE,WASHINGTON,1
1 1/2 PEARL,HARLEM,NY,00133,
OLE,SVENSON,R.F.D 4 BOX 10,B
ARNESVILLE,MINN.,56556
120 DATA PEDRO,MARTINEZ,CORT
EZ PLAZA,CORPUS CHRISTI,TX,5
8180,SAMMY,PUKA,11 LOANA LAN
E,WAKIKI,HAWAII,99845
130 DATA 1,8,20,40,55,60,65
140 RESTORE 130 :: FOR J=1 T
O 7 :: READ F(J):: NEXT J ::
 RESTORE 100 :: CALL FIELDSA
VE(6,F(),6,F$())
150 DISPLAY AT(12,1)ERASE AL
L:"(1) FIELDSORT?":"(2) FIEL
DPICK?":"(3) PRINT?":"CHOICE
?" :: ACCEPT AT(15,9)SIZE(1)
VALIDATE("123"):C
160 ON C GOTO 170,180,220
170 DISPLAY AT(18,1):"SORT O
N POSITION 1,8,20,40, 55 OR
60?" :: ACCEPT AT(19,11):P :
: CALL FIELDSORT(6,F$(),P)::
 GOTO 150
180 DISPLAY AT(18,1):"PICK B
ETWEEN POSITIONS(1,8, 20,40,
55,60,65)?" :: ACCEPT AT(20,
1):P1 :: DISPLAY AT(20,5):"A
ND?" :: ACCEPT AT(20,10):P2
190 CALL FIELDPICK(F$(),6,P1
,P2,M$):: PRINT M$
200 PRINT "PRESS ANY KEY"
210 CALL KEY(0,K,S):: IF S=0
 THEN 210 ELSE 150
220 DISPLAY AT(20,1):"PRINT
TO?":"(1) SCREEN":"(2) PRINT
ER" :: ACCEPT AT(20,11)VALID
ATE("12")SIZE(1):PP :: PR=AB
S(PP>1):: IF FL=1 THEN 240 :
: FL=1
230 IF PP=2 THEN DISPLAY AT(
20,1):"PRINTER NAME?" :: ACC
EPT AT(21,1):PR$ :: OPEN #1:
PR$
240 FOR J=1 TO 6 :: PRINT #P
R:F$(J):: NEXT J :: GOTO 200
```

FIELDSORT

CALL FIELDSORT(N,F$(),P) will sort a tabbed file created by FIELDSAVE or by TI-Writer, of N records, on position P.

FIELDPICK

CALL FIELDPICK(F$(),N,P1,P2,M$) will return in M$ the substring between tab positions P1 and P2 of the subscript N of the tabbed array F$() created by FIELDSAVE.

FIELDMAKE

CALL FIELDMAKE(FN,F(),OP,OPF$,P$()) will open #OP as a DISPLAY, VARIABLE 254 file named OPF$ (include DSK and drive number in parameter). It will then display a sequence of FN input prompts from P$(), if any, and accept LINPUT of FN items of data which will be combined into a tabbed string, using tab values from F() in the same way as FIELDSAVE, and will print the resulting string to the disk. Input of "END" will terminate. The resulting file can be read into an array and manipulated by FIELDSORT and FIELDPICK. Since the file is opened in APPEND mode, it can be added to by this subprogram.

```
100 DATA FIRST NAME?,LAST NA
ME?,STREET ADDRESS?,CITY?,ST
ATE?,ZIP CODE?
110 RESTORE 100 :: FOR J=1 T
O 6 :: READ P$(J):: NEXT J
120 DATA 1,8,20,40,50,60,70
130 RESTORE 120 :: FOR J=1 T
O 7 :: READ F(J):: NEXT J
140 CALL FIELDMAKE(6,F(),1,"
DSK1.FIELDMAKER",P$())
```

FIND

CALL FIND(F$,H,M$,S) where F$ is a DISPLAY, FIXED disk file of text records in alphabetic sequence, H is the number of records and M$ is the string to be searched for, will perform a fast binary search and return in S the REC number of the record which begins with M$, or -1 if it is not found.

The first time this demo is run, put a blank disk in the drive, because it will create a file of 6760 records in 272 sectors.

Before running it again, delete the ! in line 100.

Records will be AA0 to ZZ9.

```
110 OPEN #1:"DSK1.TEST",FIXE
D 10,RELATIVE,OUTPUT
120 FOR J=65 TO 90 :: FOR K=
65 TO 90 :: FOR L=48 TO 57 :
: PRINT #1:CHR$(J)&CHR$(K)&C
HR$(L):: NEXT L :: NEXT K ::
 NEXT J
130 F$="DSK1.TEST" :: H=6760
140 INPUT "FIND? ":M$ :: CAL
L FIND(F$,H,M$,S):: PRINT TA
B(5);S :: GOTO 140
```

KEYSEARCH

CALL KEYSEARCH(F,F$) will open a file #F named F$; will offer options of output to screen, printer, or both; will accept up to 10 Key-words to be searched for, alone or only in combination with up to 10 secondary Key-words for each; and will perform a search for first match or all matches.

LONGSHELLT

CALL LONGSHELLT(N,N$(),T) will sort a string array N$() of N items into the numeric sequence of digits at the end of the string starting at position T - as for instance, the ZIP code at the end of an address. Numbers may vary in length but must begin at position T and all characters following must be numeric.

```
100 DATA ABC 123,XXX 999,YY
 567,B   323,KQK  89,GVG   1
,B     2
110 FOR J=1 TO 7 :: READ N$(
J):: NEXT J :: CALL LONGSHEL
LT(7,N$(),5):: FOR J=1 TO 7
:: PRINT N$(J):: NEXT J
```

OPENER

CALL OPENER(N,G$) will search 4 drives for any type of file named G$ and open it as file #N, or print "CAN'T OPEN" if not found. Contains ON ERROR.

```
100 INPUT F$ :: CALL OPENER(
1,F$):: INPUT #1:M$ :: PRINT
 M$ :: CLOSE #1 :: STOP
```

RECNUM and ENDFILE

CALL RECNUM(F,F$,N), where F is the file number and F$ is the filename of a file to be opened in FIXED, RELATIVE, UPDATE mode, will open the file and, if a new file, print 0 in REC 0 and return that value in N. Otherwise, N will be the value found in REC 0.

CALL ENDFILE(F,N) will print the value of N in REC 0 of file #F.

CALL ENDFILE is used to record the REC number of the highest record, in REC 0. CALL RECNUM is used to retrieve this number, before adding records to the file, and will also open a new file and check REC 0 without crashing.

```
100 DISPLAY AT(12,1)ERASE AL
L:"FILENAME? DSK" :: ACCEPT
AT(12,14):F$
110 CALL RECNUM(1,F$,N):: PR
INT N;"RECORDS" :: IF N=0 TH
EN 130
120 FOR J=1 TO N :: INPUT #1
,REC J:A :: PRINT A;:: NEXT
J :: PRINT
130 N=N+1 :: PRINT #1,REC N:
N :: CALL ENDFILE(1,N):: GOT
O 110
```

SWEEP

CALL SWEEP will delete all the files on a disk, unless protected. Use with caution! After MERGing in, delete the ! in line 22548. Contains ON ERROR.

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

BASIC

This is not a subprogram, and is line-numbered 1 to 8. If it is merged into the beginning of a program which will run only in Basic because it uses character sets 15-16 (unless BXB is used) or TE II speech, it will warn and abort if the program is run in Extended Basic. Possibly this will not work with all consoles and modules. Credited to Steve Chapman, Bill Wallbank

BXB

CALL BXB at the beginning of a program will permit character sets 0 through 16 to be used for colors and character redefinition. Even characters 24-31 can be redefined, including the cursor (30) and edge character (31); redefining others in this set may possibly affect program execution. Sprites can only use characters 32 through 143. A single CALL COLOR cannot be used for multiple char sets, and CALL COLOR cannot be used for sprites. This is an adaptation of John Behnke's VDPUTIL2. Requires Memory Expansion.

```
100 CALL CLEAR :: CALL BXB :
: CH=24 :: FOR J=0 TO 16 ::
PRINT J;:: FOR K=0 TO 7 :: P
RINT CHR$(CH+K);:: NEXT K ::
 PRINT :: CH=CH+8 :: NEXT J
110 FOR S=0 TO 16 :: CALL CO
LOR(S,2,S+3+(S>13)*13):: NEX
T S
120 FOR CH=24 TO 159 :: CALL
 CHAR(CH,"FF"):: NEXT CH
130 GOTO 130
```

FINISHED

CALL FINISHED will alert you to the completion of a lengthy routine such as a sort. Credited to Bill Knecht. Optional speech.

FREEZE

CALL FREEZE(X), if X=1, will freeze all sprite motion; if X=0, will release all sprites to whatever motion they have been programmed for. CALL FREEZE(1) can be used before creating sprites with motion, to hold them all motionless; then CALL FREZE(0) sets them all in motion simultaneously. Requires Memory Expansion.

```
100 CALL CLEAR :: DEF RAND=5
0*RND-50*RND :: CALL FREEZE(
1):: CALL MAGNIFY(2):: FOR J
=1 TO 28 :: CALL SPRITE(#J,4
2,INT(15*RND+2),100,100,RAND
,RAND):: NEXT J :: CALL FREE
ZE(0)
110 FOR D=1 TO 200 :: NEXT D
 :: CALL FREEZE(1):: FOR D=1
 TO 200 :: NEXT D :: CALL FR
EEZE(0):: GOTO 110
```

MENU

CALL MENU(W,P) will read W number of items from DATA statements in the main program (each must have a different initial letter), clear the screen and list them (double-spaced if not more than 10) with the initial letter in parentheses, then request and validate input of one of the initial letters and return its position in the menu in P, for use in ON P GOTO or ON P GOSUB.

```
100 DATA INPUT,OUTPUT,SAVE,M
ERGE,UPDATE,REVIEW,LIST,DELE
TE,PRINT,QUIT
110 RESTORE 100 :: CALL MENU
(10,P)
```

MENU2

CALL MENU2(W,P) is the same as MENU except that it numbers items (therefore more than one can have the same initial, but is limited to 9) and accepts choice without Enter.

```
100 DATA LOAD,LIST,SORT,SAVE
110 RESTORE 100 :: CALL MENU
2(4,P):: ON P GOSUB 120,130,
140,150 :: FOR D=1 TO 200 ::
 NEXT D :: GOTO 110
120 PRINT "LOADING" :: RETUR
N
130 PRINT "LISTING" :: RETUR
N
140 PRINT "SORTING" :: RETUR
N
150 PRINT "SAVING" :: RETURN
```

NOTES

NOTES is not a subprogram. It is line-numbered 1 to 6 and is intended to be merged into memory before writing a music program. It creates a 4-octave scale of notes begining with the frequency assigned to F in line 1, and then assigns these frequencies to mnemonic variables from A1 for A 1st octave, B1F for B flat 1st octave, B1 for B 1st octave, B1S for B sharp 1st octave, etc., to G4S for G sharp 4th octave. P gives a silent rest. Music can then be programmed in CALL SOUNDs using these mnemonic variables, and the key of the music can be changed by changing the value of F in line 1.

NUMST

CALL NUMST(N,N$), where N is any number, will return it in N$ as a string followed by the appropriate suffix st, nd, rd or th in superscript. Letters "dhnrst" are redefined.

```
100 CALL CLEAR
110 INPUT "NUMBER? ":N :: CA
LL NUMST(N,N$):: PRINT N$ ::
 GOTO 110
```

SUBFLAG

CALL SUBFLAG(F,V), if F is 0 will place the value of V in the subprogram. If F is 1 and V is not 0, it will obtain the value of V from the subprogram. This is useful for changing flags or other values, in subprograms, which are not in the parameter list.

```
100 CALL SUBFLAG(0,10):: CAL
L TEST(X):: PRINT X :: CALL
SUBFLAG(0,0):: CALL TEST(X):
: PRINT X :: CALL SUBFLAG(0,
30):: CALL TEST(X):: PRINT X
110 SUB TEST(X):: CALL SUBFL
AG(1,V):: IF V<>0 THEN X=V
120 SUBEND
```

NUTS & BOLTS DISK (No. 1) contains another 100 MERGE format subprograms. Contents include:
13 screen fonts - giant, stylized, slanted, enlarged, upside down, inverse, compressed numbers, Russian, slashed zero, etc.
10 screen wipes - Chameleon border and wipe, curtains, 4-way, etc.
8 pauses - key holds, stop and go, music while you wait, music while you read, etc.
3 programming aids - screen grid, check routine, kill quit.
9 data saving and reading routines including some little-known memory savers.
12 sorts and scrambles for both numeric and string data, inserting data, shuffling, etc.
And protection routines, printer aids, joystick and keyboard controls, math, music routines, etc.
Plus a tutorial on using subprograms, and 5 pages of documentation.

NUTS & BOLTS DISK No. 2 contains another 108 subprograms including:
20 character fonts and related routines - giant, enlarged, double-height, double width, script, sideways, underlined, etc.
21 screen displays - horizontal and vertical scrolling, centering, titling, etc.
3 joystick routines for 1 or 2 joysticks.
13 math routines including every conversion between binary, hex and decimal, and more.
6 very unusual graph routines, one for printer.
3 self-changing routines to permit use of a variable in a GOSUB, GOTO or RESTORE.
1 speech routine and 2 sound effect routines.
4 word processing subprograms - screen formatting, plural endings, replacing strings.
5 utilities - INIT check, instant color change, resets, reading memory size.
10 programming utilities to edit and save screens, print screens, call disk catalog, etc.
Also 4 file handling, 2 menu routines, 6 sorting routines for 2-dimensional arrays, etc.
With 10 pages of documentation.