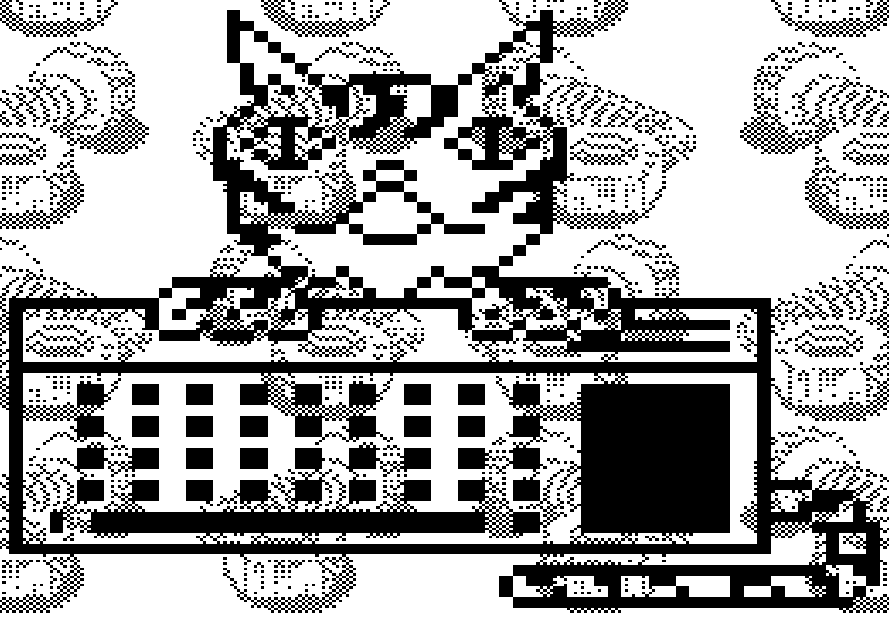
NUTS & BOLTS
NUMBER 1
DOCUMENTATION
1984



TIGERCUB
SOFTWARE
JIM PETERSON

NUTS & BOLTS

(No. 1)

TIGERCUB SOFTWARE
156 Collingwood Ave.
Columbus OHIO 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 own programs.

This disK contains 100 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 20,000 to 20,700 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.

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. When variable names are included in the parameter list only for the purpose of passing them into another subprogram, they contain the @ sign.

A few of these routines contain a CALL LOAD or CALL PEEK. These require the Memory Expansion, and possibly may not run properly on some models of the computer. Also, a few of them contain DATA statements; when using these, be sure to RESTORE any DATA statements in the main program before READing them.

The disK also contains a tutorial on the use of subprograms. To run this, type RUN DSK1.TUTORIAL . Also included as a bonus is the Tigercub Menuloader. It cannot be used to load these MERGEd subprograms, but can be transferred to your program disKs under the filename LOAD.

>>>>>>>>>TYPE FONTS<<<<<<<<<

BIGCHAR

CALL BIGCHAR(M$,CH,R,C) where M$ is text to be displayed, CH is lowest character to be redefined, R and C are starting row and column of print, will slowly display text of up to 14 characters in heavy enlarged letters. Allow 4 characters below ASCII 144 (and above the ASCII of any character in M$) for each character in M$. Will also enlarge previously redefined characters and can be used together with SLANT, MONGOLIAN, etc. Example -
```
100 CALL BIGCHAR("TI-99/4A",
91,10,8)
```

BIGCHAR2

CALL BIGCHAR(A$,CH,P$) where A$ is any character, CH is any ASCII divisible by 4 and not the ASCII of A$ or the following three, will enlarge A$ to a 64-bit size which can be displayed as a giant sprite by CALL MAGNIFY (4) and CALL SPRITE(.....,, and the 64-digit hex code is returned in P$. Example -
```
100 CALL CLEAR :: CALL MAGNI
FY(4):: FOR CH=65 TO 90 :: C
ALL BIGCHAR2(CHR$(CH),96,P$)
:: CALL SPRITE(#1,96,16,96,1
20):: NEXT CH
```

CHARFACE

CALL CHARFACE will reidentify the lower case characters ASCII 97-122 to a stylized upper case. Example -
```
100 PRINT "abcdefghijklmnopq
rstuvwxyz" :: CALL CHARFACE
```

CHARSET2

CALL CHARSET2 will restore the characters above ASCII 95 which are not restored by CALL CHARSET. Example - add
```
110 CALL CHARSET2
```
to the CHARFACE example above.

INVERSE

CALL INVERSE(M$,R,C) will turn the screen blue, display M$ at row R, column C, alternately flashing blue on white/white on blue until any Key is pressed. Any lower case text on the screen will change to upper case in reversed colors. Example -
```
100 CALL INVERSE("PRESS ANY
KEY",24,15)
```

LARGECHAR

CALL LARGECHAR converts all characters from ASCII 33 to 122 into an enlarged format which fills a print space. Example -
```
100 CALL CLEAR :: FOR CH=33
TO 122 :: PRINT CHR$(CH)&" "
;:: NEXT CH :: CALL LARGECHA
R
```

LOWERCASE

CALL LOWERCASE converts the lower case letters to true lower case and raises all the other characters to align them. Example -
```
100 CALL CLEAR :: FOR CH=33
TO 64 :: PRINT CHR$(CH);:: N
EXT CH :: FOR CH=65 TO 90 ::
 PRINT CHR$(CH)&CHR$(CH+32);
:: NEXT CH :: CALL LOWERCASE
```

MONGOLIAN

CALL MONGOLIAN will change the upper case letters ASCII 65-90 into an imitation "Mongolian"; CALL CHARSET will instantly restore them. Example -
```
100 CALL CLEAR :: PRINT "ABC
DEFGHIJKLMNOPQRSTUVWXYZ" ::
CALL MONGOLIAN :: CALL CHARS
ET
```

RUSSIAN

CALL RUSSIAN converts the upper case letters and the numerals to the Russian alphabet. FCTN W, R, T, F, G, A and Z are used for the extra letters. Example -
```
100 CALL CLEAR :: FOR CH=48
TO 90 :: PRINT CHR$(CH);:: N
EXT CH :: PRINT CHR$(91);CHR
$(92);CHR$(93);CHR$(123);CHR
$(124);CHR$(125);CHR$(126)
110 CALL RUSSIAN
```

SCRUNCH

CALL SCRUNCH(R,P$) will compress any positive integer R and return it in P$ with two digits in each print space - not very legibly on some TV screens. Reidentifies chars 91-143, therefore limited to 52 pairs of digits. Example-
```
100 CALL CLEAR :: CALL SCRUN
CH(1234567890,P$):: DISPLAY
AT(12,10):P$
110 GOTO 110
```

SLANT

CALL SLANT converts all upper case letters and numerals and punctuation to a slanted form. Example -
```
100 CALL CLEAR :: FOR CH=33
TO 90 :: PRINT CHR$(CH);:: N
EXT CH :: CALL SLANT
```

SLASH

CALL SLASH will put a slash through all the zeros while the program is running. Example -
```
100 PRINT "10000 0000 00000"
 CALL SLASH
110 GOTO 110
```

UPSIDEDOWN

CALL UPSIDEDOWN(A,B) will turn all characters between ASCII A and ASCII B upside down. Example -
```
100 CALL CLEAR :: FOR CH=49
```

```
TO 57 :: PRINT CHR$(CH);:: N
EXT CH :: CALL UPSIDEDOWN(49
,57)
110 GOTO 110
```

>>>>>>>>TEXT DISPLAYS<<<<<<<

ALTERNATE

CALL ALTERNATE(X) will read DATA, display it on the screen in alternate lines printed in color X, and will instantly flash from one screen to the next when any Key is pressed. DATA for the even-numbered screens must be in lower case (will print in upper case) using CTRL , for comma, CTRL A for period and no other punctuation. If numerals are needed on even-numbered screens, remove the ! in line 20460 and type CTRL B through CTRL K for digits 0 through 9. Screens are limited to 11 lines and the last DATA item for each screen must be X, the final DATA item to leave the subprogram must be XX. If a screen has fewer lines than the previous alternate screen, use blanK DATA " " for erasure. Example -

```
100 DATA THIS,IS,A,DEMONSTRA
TION,X,of,the,alternate,prin
ting,subprogram,X,FROM,THE,T
IGERCUB,NUTS & BOLTS,DISK,X
110 DATA sold,by,tigercub,so
ftware," ",XX
120 CALL CLEAR :: CALL SCREE
N(5):: CALL ALTERNATE(16)
```

COLORTEXT

CALL COLORTEXT displays text on a blacK ground, changing through all the colors until any Key is pressed. Example-

```
100 DISPLAY AT(10,1)ERASE AL
L:"THIS IS A DEMONSTRATION O
F":"THE COLORTEXT SUBPROGRAM
" :: CALL COLORTEXT
```

COLUMNIZER

CALL COLUMNIZER(D,N,P) will equalize the number of decimal places D and will right-justify by aligning the decimals in column P. Will handle both negative and positive numbers N. Example-

```
100 CALL CLEAR :: RANDOMIZE
110 N=1000*RND-1000*RND :: C
ALL COLUMNIZER(8,N,10):: GOT
O 110
```

CRAWL

CALL CRAWL(S,C$,R) will scroll text C$ across the screen from right to left in row R at speed S until any Key is pressed. Example -

```
100 CALL CLEAR :: CALL CRAWL
(10,"THIS IS A DEMONSTRATION
 OF THE CRAWL SUBPROGRAM - P
RESS ANY KEY",12)
```

FADE-IN

CALL FADE_IN will turn all upper case characters blanK and set charsets 1-4 to color 1 on 1 so that text can be PRINTed or DISPLAYed invisibly. CALL FADE_IN again will slowly fade in the text. Example -

```
100 CALL CLEAR :: CALL FADE_
IN
110 PRINT "THIS IS A TEST OF
 THE FADE- IN PROGRAM" :: CA
LL FADE_IN
120 CALL KEY(0,K,S):: IF S=0
 THEN 120 else 100
```

FLASH

CALL FLASH(A,B) will turn screen, border, foreground and backGround of charsets 1-12 to color A. After text is PRINTed or DISPLAYed invisibly, CALL FLASH(A,B) again will flash it on by changing the backGround color to B. Example -

```
100 CALL CLEAR :: CALL FLASH
(7,B):: FOR R=1 TO 24 :: PRI
NT "TESTING" :: NEXT R :: CA
LL FLASH(11,7)
110 GOTO 110
```

FLASH-ON

CALL FLASH_ON will CLEAR the screen, reidentify ASCII 65-90 to blanKs and change char sets 1-4 to color 1 on 1. After text has been PRINTed or DISPLAYed invisibly, CALL FLASH_ON again will flash it on instantly. Example -

```
100 CALL FLASH_ON :: FOR R=1
 TO 24 :: PRINT "TESTING" ::
 NEXT R :: CALL FLASH_ON
110 GOTO 110
```

FORMATTER

CALL FORMATTER will reformat text from DATA statements to be displayed on screen without words wrapping around. A DATA item P will cause a line to be sKipped. The last DATA item must be ZZZ. Example -

```
100 DATA CALL FORMATTER will
 reformat text from DATA sta
tements to be displayed on s
creen without words wrapping
 around.,ZZZ
110 RESTORE 100 :: CALL FORM
ATTER
```

JIGGLE

CALL JIGGLE(M$,R,C) will DISPLAY text M$ at row R, column C and "jiggle" it for attention until any Key is pressed. Example -

```
100 CALL CLEAR :: CALL JIGGL
E("PRESS ANY KEY",24,12)
```

MIRROR

CALL MIRROR(M$,R,C) will DISPLAY text M$ on the screen at row R, column C, in any number of lines, in unusual mirrored print. ASCII 97-122 are redefined, can be restored by CHARSET2. Example -

```
100 CALL CLEAR :: CALL MIRRO
R("THIS IS AN EXAMPLE OF MIR
ROR PRINTING",12,3)
110 GOTO 110
```

PRICE

CALL PRICE(P,Q,T,P$) where P=price, Q=quantity and T= sales tax percentage, will return in P$ the dollar amount preceded by $, rounded to nearest cent and zero-filled to two decimals. Example -

```
100 INPUT "price? ":P :: INP
UT "QUANTITY? ":Q :: INPUT "
SALES TAX PERCENTAGE? ":T ::
 CALL PRICE(P,Q,T,P$):: PRIN
T P$ :: GOTO 100
```

TITLE

CALL TITLE(S,M$) where S is the screen color, will display the text M$ of up to 28 characters diagonally across the screen in magnified sprites of every color except the screen color. Example -

```
100 CALL CLEAR :: CALL TITLE
(5,"THIS IS THE TITLE"):: GO
TO 100
```

TWOWAY

CALL TWOWAY(M,R,S) will read and print M number of DATA statements starting at row R with S line spacing, two ways from center. Example -

```
100 DATA THIS IS AN EXAMPLE,
OF THE TWO-WAY,PRINT SUBPROG
RAM,FROM THE NUTS & BOLTS DI
SK
110 CALL CLEAR :: RESTORE 10
0 :: CALL TWOWAY(4,2,2)
120 GOTO 120
```

>>>>>>>SCREEN WIPES<<<<<<<<<

BORDER

CALL BORDER(F,B) will give a screen color of F bordered on all sides with color B, with text in color B.
CALL WIPE will erase text from the screen without affecting the border. Example -

```
100 CALL BORDER(16,5)
110 DISPLAY AT(12,11):"BORDE
R" :: FOR D=1 TO 200 :: NEXT
 D :: CALL WIPE :: GOTO 110
```

CHAMELEON

CALL CHAMELEON puts a randomly designed and colored border around the screen. Redefines ASCII 128, colors set 13. It can be combined with BORDER.
CALL CHAMWIPE, after the above, alternately wipes the screen down or across with the same pattern. Example -

```
100 CALL CLEAR :: CALL CHAME
LEON :: DISPLAY AT(12,10):"C
HAMELEON" :: FOR D=1 TO 200
:: NEXT D :: CALL CHAMWIPE :
```

: GOTO 100

CURTAIN

CALL CURTAIN(S) will slowly and smoothly wipe the screen with color S from top to bottom. ASCII 120-143 are redefined. For a slower, smoother wipe, change line 20156 to -
FOR T=1 TO 8 (instead of 4) and CH$=RPT$("FF",T)&"0" instead of "FFFF"

CURTAIN2

CALL CURTAIN2(S) will slowly and smoothly wipe the screen from left to right with color S. ASCII 112-143 are redefined.
100 CALL CURTAIN2(5):: CALL SCREEN(5):: CALL CLEAR

FADE-OUT

CALL FADE_OUT will slowly fade out any upper case text. It can be restored instantly by CALL CHARSET. Example -
100 CALL CLEAR :: DISPLAY AT(12,1):"THIS IS AN EXAMPLE OF THE":"FADE-OUT SUBPROGRAM" :: CALL FADE_OUT :: FOR D=1 TO 200 :: NEXT D :: CALL CHARSET

OUTSIDE-IN

CALL OUTSIDE_IN will erase the screen from the edges inward. Example -
100 CALL HCHAR(1,1,42,768):: CALL OUTSIDE_IN

SECTIONS

CALL SECTIONS wipes the screen simultaneously outward in 4 sections with four colorful curtains.
Redefines ASCII 119,127,135 and 143, colors sets 11-14. Example -
100 CALL SECTIONS :: GOTO 100

UPDOWNWIPE

CALL UPDOWNWIPE will wipe the screen up and down from center with a colorful random pattern. Redefines ASCII 143, colors set 14. Example- 100 CALL UPDOWNWIPE

WIPES

CALL WIPES will clear the screen randomly in one of 4 directions with a random colorful curtain. Redefines ASCII 143, colors set 14. Example -
100 CALL WIPES :: GOTO 100

>>>>>>>>>>PAUSES<<<<<<<<<<

CALLBELL

CALL CALLBELL will pause until any Key is pressed, sound a bell when the Key is pressed, a different tone for each Key. Example -
10 CALL CALLBELL :: GOTO 10

HOLD

CALL HOLD will stop a program whenever any Key is pressed, until the Key is released. Example -
100 PRINT "HOLD DOWN ANY KEY" :: CALL HOLD :: GOTO 100

INTERLUDE

CALL INTERLUDE will play a 25-second interlude of randomly composed music.
100 CALL INTERLUDE

PAUSE

CALL PAUSE will stop a program when any Key is pressed and start it again when any Key is pressed. Example -
100 PRINT "HIT ANY KEY" :: CALL PAUSE :: GOTO 100

SHUTOFF

CALL SHUTOFF(T) will display "PRESS ANY KEY" and wait for Key input, return to title screen if no Key is pressed in T minutes. To avoid accidental erasure during programming, line 20589 is a REM. Delete the ! when programming is completed.

WAIT

CALL WAIT will display "PRESS ANY KEY" flashing until a Key is pressed.
100 CALL CLEAR :: CALL WAIT

WAITING

CALL WAITING(T) will display "PRESS ANY KEY" and after T seconds will ring an alarm and display "I'M WAITING", after each further T seconds will ring the alarm and flash the screen until any Key is pressed. Example -
100 CALL CLEAR :: CALL WAITING(10)

WAITMUSIC

Will play random music until any Key is pressed. Program it as follows -
100 CALL KEY(0,K,S):: IF S<>0 THEN 110 ELSE CALL WAITMUSIC :: GOTO 100
110 (continue program)

>>>>>>PROGRAMMING AIDS<<<<<<

SCREENGRID

CALL SCREENGRID in line 1 will put a temporary numbered grid on the screen to aid in programming graphics.
1 CALL SCREENGRID
2 GOTO 2

CHECK

When Keying in a program, start with temporary lines
1 CALL CHECK
2 GOTO 2
and type RUN whenever you have completed a screenfull of lines. The white numerals and punctuation make it much easier to spot errors.

KILLQUIT

CALL KILLQUIT at the beginning of a program disables the FCTN= so that you will not accidentally go back to the title screen, either while programming (once you have RUN it) or while using the program.

>>>>DATA SAVING, READING<<<<

CHARSAVE

CALL CHARSAVE(CH,R) where CH is an ASCII and R is a positive integer not in exponent format, will save the value of R in the redefinition of CH so that it can be passed to another program linked by a RUN statement and recalled by CALL CHARPAT(CH,CH$):: R=VAL(CH$). Example -

>MERGE DSK1.CHARSET2
100 DIM N(112):: FOR CH=32 TO 143 :: CALL CHARPAT(CH,CH$):: N(CH-31)=VAL(CH$):: NEXT CH :: CALL CHARSET :: CALL CHARSET2
110 FOR J=1 TO 112 :: PRINT N(J):: NEXT J
(insert copy disk)
>SAVE DSK1.TEST
(insert Nuts & Bolts disk)
>MERGE DSK1.CHARSAVE
100 FOR CH=32 TO 143 :: CALL CHARSAVE(CH,CH^2):: NEXT CH
110 RUN "DSK1.TEST"
(insert copy disk)
>RUN

CHARSAVE2

CALL CHARSAVE2(CH,N) works the same as CHARSAVE but will also transfer negative numbers, decimals, and numbers in exponential notation which must be recalled by the READCHAR subprogram. Example -
>MERGE DSK1.READCHAR
100 FOR CH=134 TO 143 :: CALL READCHAR(CH,N):: DISPLAY AT(CH-130,15):N :: NEXT CH
(insert copy disk)
>SAVE DSK1.TEST
(insert Nuts & Bolts disk)
>MERGE DSK1.CHARSAVE2
100 CALL CLEAR :: RANDOMIZE :: FOR CH=134 TO 143 :: N=10000*RND-10000*RND :: DISPLAY AT(CH-130,1):N :: CALL CHARSAVE2(CH,N):: NEXT CH
110 RUN "DSK1.TEST"
(insert copy disk)
>RUN

PACKING
(revised)

CALL PACKING(N,M@$(),T@$(),P

@$()) will store up to 2540 positive integers N of up to 64770 in value, sequentially in the same way as an array of N(10,254) but using only 3 bytes of stacK memory per number whereas an array would use 8 bytes of program memory. Even more storage is available by DIMensioning M@$(), T@$() and P@$().

The stored DATA is recovered by the READPACK subprogram. Example -

```
>MERGE DSK1.PACKING
>MERGE DSK1.READPACK
100 FOR N=1 TO 508 :: PRINT
N :: CALL PACKING(N,M@$(),T@
$(),P@$()):: NEXT N :: DISPL
AY AT(3,1)ERASE ALL:"DATA SA
VED FROM N(1,1) TO N(2,254)"
110 DISPLAY AT(12,1):"SUBSCR
IPT #? N(" :: ACCEPT AT(12,1
6)VALIDATE(DIGIT):P
120 DISPLAY AT(12,17):"," ::
 ACCEPT AT(12,18)VALIDATE(DI
GIT):PP :: IF PP<1 OR PP>254
 THEN 140 ELSE CALL READPACK
(P,PP,M@$(),T@$(),P@$(),N)
130 DISPLAY AT(14,1):"N("&ST
R$(P)&","&STR$(PP)&")="&STR$
(N):: GOTO 130
```

The data can also be saved to disK and recovered, much faster than by the usual way and requiring only 3 sectors per 254 numbers whereas an I/F file requires a sector for each 28 numbers. The data can be saved by this routine, where N represents the total number of values -

```
160 OPEN #1:"DSK1.PACKFILE",
VARIABLE 254,OUTPUT :: T=INT
(N/254)-(N/254<>INT(N/254)):
: PRINT #1:T
170 FOR J=1 TO T :: PRINT #1
:M@$(J):: PRINT #1:T@$(J)::
PRINT #1:P@$(J):: NEXT J ::
CLOSE #1 :: END
```

And recovered by -

```
100 OPEN #1:"DSK1.PACKFILE",
VARIABLE 254,INPUT
110 INPUT #1:T :: FOR J=1 TO
 T :: LINPUT #1:M@$(J):: LIN
PUT #1:T@$(J):: LINPUT #1:P@
$(J):: NEXT J :: CLOSE #1
```

PACKNUM
(revised)

CALL PACKNUM(N,F@$(),M@$(),T@$(),P@$(),D@$()) is the same as subprogram PACKING but will store both negative and positive numbers from -64770 to 64770 as well as non-integers, which are rounded to two decimal places. Uses 5 bytes of stacK per number. DATA is recovered by the READNUM subprogram. Example -

```
<MERGE DSK1.PACKNUM
<MERGE DSK1.READNUM
100 CALL CLEAR :: RANDOMIZE
:: FOR J=1 TO 20
110 N=64770*RND-64770*RND ::
 DISPLAY AT(J+2,1):N :: CALL
 PACKNUM(N,F@$(),M@$(),T@$()
,P@$(),D@$()):: NEXT J
120 FOR J=1 TO 20 :: CALL RE
ADNUM(1,J,F@$(),M@$(),T@$(),
P@$(),D@$(),N):: DISPLAY AT(
J+2,15):N :: NEXT J
```

To save to disK, add -

```
102 OPEN #1:"DSK1.PACKFILE",
VARIABLE 254,OUTPUT :: PRINT
 #1:1
103 PRINT #1:F@$(1):: PRINT
#1:M@$(1):: PRINT #1:T@$(1):
: PRINT #1:P@$(1):: PRINT #1
:D@$(1):: CLOSE #1
```

And to recover -

```
105 OPEN #1:"DSK1.PACKFILE",
VARIABLE 254,INPUT
106 INPUT #1:T :: FOR J=1 TO
 T :: LINPUT #1:F@$(J):: LIN
PUT #1:M@$(J):: LINPUT #1:T@
$(J):: LINPUT #1:P@$(J):: LI
NPUT #1:D@$(J):: NEXT J :: C
LOSE #1
```

READCHAR

CALL READCHAR(CH,N) will read the hex code of ASCII CH and recover the value N which was passed by the CHARSAVE2 subprogram. See example under CHARSAVE2.

READPACK

CALL READPACK(P,PP,M@$(),T@$(),P@$(),N) will recover the value N which was stored in string P position PP by the PACKING subprogram, in the same way as calling it from N(P,PP). See Example under PACKING.

READNUM

CALL READNUM(P,PP,F@$(),M@$(),T@$(),P@$(),D@$(),N) will recover the value which has been stored in string P, position PP, by the PACKNUM subprogram, in the same way as calling it from N(P,PP). See example under PACKNUM.

SAVESTRING

CALL SAVESTRING(A,B,M$(),A$) will select a substring A$ among B substrings stored in A number of strings M$(). This method will store far more than can be read into an array from DATA statements. Strings must contain an equal number of substrings which must be separated by lower case letters in sequence. Example -

```
100 CALL CLEAR :: M$(1)="aCU
PbSWORDcPURSEdGOLD COINeMIRR
ORf" :: M$(2)="aCAVEbCHESTcR
OOMdDUNGEONeCRYPTf"
110 RANDOMIZE :: A=INT(5*RND
+1):: CALL SAVESTRING(1,A,M$
(),A$):: B=INT(5*RND+1):: CA
LL SAVESTRING(2,B,M$(),B$)
120 PRINT "THE ";B$;" CONTAI
NS A ";A$: :: GOTO 110
```

SCREENSAVE

CALL SCREENSAVE(W$(),XX(),W,X) will read the values left on the screen by a previous program linKed by a RUN statement, and return them in W$(W) and XX(X). Previous program must terminate with FOR J=1 TO 24 :: PRINT "!" :: NEXT J and then PRINT statements to print up to 23 values along the left edge of the screen, such as PRINT A :: PRINT NAME$ :: FOR J=1 TO 9 :: PRINT N(J):: NEXT J etc. Strings must not contain spaces. The last PRINT must be PRINT "*" and then RUN. Place this CALL at the very beginning of the next program before clearing the screen. Example -

(insert Nuts & Bolts disK)

```
>MERGE DSK1.SCREENSAVE
100 CALL SCREENSAVE(W$(),XX(
),W,X):: FOR J=1 TO W :: PRI
NT W$(J);"'S SCORE WAS";XX(J
):: NEXT J
```

(insert copy disK)

```
> SAVE DSK1.NEXT

100 P$(1)="JACK" :: P$(2)="J
OE" :: P$(3)="CHARLIE" :: P(
1)=700 :: P(2)=840 :: P(3)=1
102
110 FOR J=1 TO 24 :: PRINT "
!" :: NEXT J :: FOR J=1 TO 3
 :: PRINT P$(J):P(J):: NEXT
J :: PRINT "*" :: RUN "DSK1.
NEXT"
>RUN

>>>>>>>>>>DISPLAYS<<<<<<<<<<
```

FLAG

CALL FLAG will clear the screen and unfurl an American flag from the staff outward. Example -

```
100 CALL FLAG
```

SPRITESHOW

CALL SPRITESHOW will set 10 randomly designed and colored intricately symmetrical giant sprites floating in all directions on a blue screen. ASCII 96-136 are redefined. Example -

```
100 CALL SPRITESHOW
```

CURSOR

CALL CURSOR(A) will redefine the cursor to the shape, normal or redefined, of ASCII A. Example -

```
100 CALL CURSOR(95)
```

TIGER

CALL TIGER will change the cursor to the Tigercub's head. Example -

```
100 CALL TIGER
```

TWODIE

CALL TWODIE(A,B) randomly displays two dice in the center of the screen and returns their value in A and B. Redefines ASCII 128, changes color of sets 9 and 13. Example -

```
100 CALL TWODIE(A,B):: IF A+
B=7 THEN DISPLAY AT(3,12):"C
RAPS!"
110 FOR D=1 TO 200 :: NEXT D
```

:: DISPLAY AT(3,10):" " :: GOTO 100

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

CALENDAR

CALL CALENDAR(Y,M,D,D$) where Y,M,D are the year, month and date of any year between 1583 and 3999, will return the day of the week in D$. Example -
100 INPUT "YEAR? ":Y :: INPUT "MONTH? ":M :: INPUT "DAY? ":D :: CALL CALENDAR(Y,M,D,D$):: PRINT D$ :: GOTO 100

CLOCK

CALL CLOCK(R,CC) will count off the seconds and display them at row R, column CC, fairly accurately until any key is pressed. Example -
100 CALL CLOCK(12,12)

MONTH

CALL MONTH(M,Y,D) will give in D the number of days in month M of year Y, including leap years. Example -
100 INPUT "NUMBER OF MONTH? ":M :: INPUT "YEAR? ":Y :: CALL MONTH(M,Y,D):: PRINT D;" DAYS" :: GOTO 100

>>>>>>>>>>>MUSIC<<<<<<<<<<<

MAJORSCALE

DIM N@(21) and CALL MAJORSCALE(K$,N@()) will set up a 3-octave major scale in whatever Key (A through G) is specified by K$. Can be used with PLAY or PLAYORGAN, for example -
>MERGE DSK1.MAJORSCALE
>MERGE DSK1.PLAY
100 DIM N@(21):: CALL MAJORSCALE("C",N@()):: CALL PLAY(999,N@(),21)

MINORSCALE

DIM N@(21) and CALL MINORSCALE(K$,N@()) will set up a minor scale as above, to be used with PLAY or PLAYORGAN. Example -
100 DIM N@(21):: CALL MINORSCALE("D",N@()):: CALL PLAYORGAN(200,N@(),21)

MUSIC

CALL MUSIC(K$,M$,B) will play music in the Key of K$ (A through G) with a tempo of B (1 to 4) from string M$ which is written in the format 2GGEGaFaG4E...where the numerals represent the comparative length of notes (only needed when the note length changes) and letters A through G represent names of notes, those in lower case being an octave higher. There is no provision for accidentals. Example -
100 CALL MUSIC("C","2GGEGaFaG4E2GccccbaG4G2GccccbaG4E2GGaGFED8C",3)

PLAY

CALL PLAY(T,N@(),X) plays random music from the scales created by the SCALE, MAJORSCALE or MINORSCALE subprograms. T is the duration (try 999 for a boogie beat, 99 for smoother music) and X is the number of notes (21 or 36) DIMensioned in the scale being played. The music is terminated by any Keypress. See example under MAJORSCALE.

PLAYORGAN

CALL PLAYORGAN(T,N@(),X) is used in the same way as PLAY but it plays notes with harmonic and bass accompaniment somewhat more slowly. See example under MINORSCALE.

SCALE

DIM N@(36) and CALL SCALE(K$,N@()) sets up a 3-octave chromatic scale beginning with whatever note (A to G) is specified by K$. Used in the same way as MAJORSCALE. Example -
100 DIM N@(36):: CALL SCALE("A",N@()):: CALL PLAYORGAN(200,N@(),36)

>>>>>SORTS AND SCRAMBLES<<<<<

DIFFERENT

CALL DIFFERENT(A,B,S,X) will pick a random number X from a sequence of A to B without selecting the same number again until after S times. Very useful to prevent the "stupid computer" occurrence of selecting the same question twice in a quiz, etc. Example -
100 CALL DIFFERENT(1,5,3,X):: PRINT X;:: GOTO 100

HIGHLOW

CALL HIGHLOW(T,N(),H,L) will return the high H and low L numbers in array N() of T numbers, without sorting. Example -
100 RANDOMIZE :: DIM N(100):: FOR J=1 TO 100 :: N(J)=INT(100*RND):: PRINT N(J);:: NEXT J :: CALL HIGHLOW(100,N(),H,L)
110 PRINT "HIGH=";H:"LOW=";L

HIGHSCRAM

CALL HIGHSCRAM(HN,N) will provide a random number N from any sequence beginning with 1 and ending with HN, which may be any number less than 256 or any greater number (to about 10,000) evenly divisible by more than 1 and less than 256. Each further CALL will give another random number, without duplication, until all are used. Example -

100 CALL CLEAR :: FOR J=1 TO 767 :: CALL HIGHSCRAM(767,X):: CALL HCHAR(INT(X/32)+1,X-INT(X/32)*32+1,30):: NEXT J
110 GOTO 110

LONGSHELL

CALL LONGSHELL(N,N$()) will sort an array of N number of strings N$(). One of the best sorts when the array may be anywhere from entirely random to entirely in sequence. Example -
100 CALL CLEAR :: DIM N$(20) :: FOR J=1 TO 20 :: FOR W=1 TO 3 :: N$(J)=N$(J)&CHR$(INT(26*RND+65)):: NEXT W :: DISPLAY AT(J+3,1):N$(J)
110 NEXT J :: CALL LONGSHELL(20,N$()):: FOR J=1 TO 20 :: DISPLAY AT(J+3,8):N$(J):: NEXT J

LONGSHELLN

CALL LONGSHELLN(N,NN()) as above, to sort numbers. Example -
100 CALL CLEAR :: DIM N(20):: FOR J=1 TO 20 :: N(J)=INT(100*RND):: DISPLAY AT(J+3,1):N(J):: NEXT J :: CALL LONGSHELLN(10,N())
110 FOR J=1 TO 20 :: DISPLAY AT(J+3,8):N(J):: NEXT J

NO-REPEAT

CALL NO_REPEAT(A,B,X) will randomly select a number from a sequence of A to B, without repeating, until all have been selected and will then reinitialize. Can be used as subscript numbers to select randomly from an array without repeating. Example -
100 CALL CLEAR :: R,C=1
110 CALL NO_REPEAT(1,15,X):: DISPLAY AT(R,C):X :: R=R+1 :: IF R<16 THEN 110 ELSE R=1 :: C=C+5 :: IF C<27 THEN 110 ELSE C=1 :: GOTO 110

QUICKSORT

CALL QUICKSORT(N,N$()) will sort an array of N number of strings N$(). It must first be DIMensioned for N+1. One of the fastest sorts if the array is entirely random, but extremely slow if it is almost in sequence. Example-
100 CALL CLEAR :: RANDOMIZE :: DIM N$(101):: R,C=4 :: FOR J=1 TO 100 :: FOR W=1 TO 3 :: N$(J)=N$(J)&CHR$(INT(26*RND+65)):: NEXT W
110 GOSUB 120 :: NEXT J :: CALL QUICKSORT(100,N$()):: R,C=4 :: FOR J=1 TO 100 :: GOSUB 120 :: NEXT J :: END
120 DISPLAY AT(R,C):N$(J);:

R=R+1 :: IF R=24 THEN R=4 :
: C=C+4 :: RETURN ELSE RETUR
N

QUICKSORTN

CALL QUICKSORTN(N,NN()) as above to sort numbers.
Example -
```
100 CALL CLEAR :: DIM N(101)
:: FOR J=1 TO 100 :: N(J)=IN
T(100*RND):: PRINT N(J);:: N
EXT J :: PRINT
110 CALL QUICKSORTN(100,N())
:: FOR J=1 TO 100 :: PRINT N
(J);:: NEXT J
```

RESORTSORT

CALL RESORTSORT(N,N$()), used as above; too slow for large random arrays but one of the fastest for restoring an almost-sequenced list. Example - change LONGSHELL to RESORTSORT in line 120 of the LONGSHELL example.

RESORTNUM

CALL RESORTNUM(N,NN()) as above, to sort numbers.

SHOEHORN

CALL SHOEHORN(N,A$,N$()) will insert A$ into its proper sequence in a sequenced array of N number of strings N$(), faster than resorting, and increase the value of N by 1. Example - Remove END from line 110 of the QUICKSORT example and add this line -
```
115 A$="JJJ*" :: CALL SHOEHO
RN(100,A$,N$()):: R,C=4 :: F
OR J=1 TO 101 :: GOSUB 120 :
: NEXT J :: END
```

SHOEHORN-N

CALL SHOEHORN_N(N,A,NN()) as above, to sort numbers.

>>>>>>>>PRINTER AIDS<<<<<<<<

SLASHZERO

CALL SLASHZERO will OPEN #1: "PIO" and initialize a Gemini printer to print slashed zeros. Example -
```
0000000000000000000000000000
```

READSCREEN

CALL READSCREEN(P$), where P$ is the designation of the printer, such as "PIO", will read the text on the screen and output it to the printer. Example -
```
100 CALL READSCREEN("PIO")
(or your printer name)
>MERGE DSK1.READSCREEN
>LIST
>RUN
```

>>>>>>>>>>KEYBOARD<<<<<<<<<< AND JOYSTICK CONTROL

CURSORCONT

Programmed as -
```
100 CALL KEY(3,K,S):: IF S=0
 THEN 100
110 CALL CURSORCONT(K,RR,CC,
CH)
120 IF K=81 THEN ..(optional
 for Q Key)ELSE 100
```
- where RR and CC are starting row and column and CH is ASCII of cursor, will move cursor in 8 directions with arrow Keys and W, R, Z and C Keys, and will stop at borders or move along them if struck diagonally, erasing the previous position, and will respond to the Q Key as programmed.

FASTJOY (revised)

CALL FASTJOY(J,C,R,Q) where J is the joystick number and C and R are the starting column and row, will return values of C and R to run the cursor around the screen rapidly without crashing at the borders, and will return a value of 18 for Q if the fire button is pressed. Example -
```
100 CALL CLEAR :: FOR CH=33
TO 137 STEP 8 :: CALL CHAR(C
H,RPT$("F",16)):: NEXT CH ::
 FOR S=1 TO 14 :: CALL COLOR
(S,S-(S>8),1)
110 NEXT S :: R,C=3 :: CH=33
 :: J=1
120 CALL HCHAR(R,C,CH):: CAL
L FASTJOY(J,C,R,Q):: IF Q=18
 THEN 130 ELSE 120
130 CH=CH+8 :: IF CH=145 THE
N CH=33 :: GOTO 120 ELSE 120
```

JOYSELECT

CALL JOYSELECT(JS) will inquire if joystick is to be used and, if so, require that Alpha Lock is up and that the fire button is pressed to identify the joystick in use, which can then be programmed with CALL JOYST(JS.... Example - Delete J=1 from line 110 of the FASTJOY example and add
```
90 CALL CLEAR :: CALL JOYSEL
ECT(J)
```

JOYWRAP (revised)

CALL JOYWRAP(J,C,R,Q) is used in the same way as FASTJOY but it wraps around at the borders. Example - substitute JOYWRAP instead of FASTJOY in line 120 of the FASTJOY example.

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

AVERAGE

CALL AVERAGE(T,N(),A) will give the average in A of an array N() of T numbers. Example -
```
100 CALL CLEAR :: RANDOMIZE
:: FOR J=1 TO 10 :: N(J)=INT
(100*RND):: DISPLAY AT(J+8,1
):N(J):: CALL AVERAGE(J,N(),
A):: DISPLAY AT(J+8,8):A
110 NEXT J
```

BINARYDEC

CALL BINARYDEC(B$,DN) will convert a binary number in string format B$ into the decimal value DN. Example -
```
100 DISPLAY AT(12,1)ERASE AL
L:"BINARY NUMBER?" :: ACCEPT
 AT(12,16)VALIDATE("10"):B$
:: CALL BINARYDEC(B$,D):: DI
SPLAY AT(15,1):"DECIMAL=";D
```

FACTORIAL

CALL FACTORIAL(N,F) will give the factorial in F of a number N. Example -
```
100 INPUT N :: CALL FACTORIA
L(N,F):: PRINT F :: GOTO 100
```

HEX-DEC

CALL HEX_DEC(H$,D) will convert any hexadecimal number H$ into its decimal equivelant D. Example -
```
100 DISPLAY AT(12,1)ERASE AL
L:"HEXADECIMAL NUMBER?" :: A
CCEPT AT(14,1)VALIDATE("0123
456789ABCDEF"):H$ :: CALL HE
X_DEC(H$,D)
110 DISPLAY AT(16,1):"DECIMA
L=";D
```

>>>>>>>>>PROTECTION<<<<<<<<<

FAILSAFE

CALL FAILSAFE at the beginning of a protected program will return to the title screen if protection has been removed. NOTE! To prevent accidents during programming, line 20435 is a REM. Delete the ! when ready to protect program.

PASSWORD

Replace the asterisKs in line 20233 with whatever password you choose, but type it in invisibly with the CTRL Key held down. Start your program with CALL PASSWORD(C@). RUN it. When the password is requested, type it in with the CTRL Key down. The value of C@ will print out. MaKe a note of it, then delete line 20235. Now, in your program give one or more Key variables a value which is its true value multiplied by the value of C@, divided by C@. For instance, if A=10 and C@ equals 720, write A=7200/C@. Now nobody can RUN the program unless they Know the password and type it in with CTRL down. If they LIST the program to find it, they will find it invisible, and if they delete it the program will not properly.

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

BELL

CALL BELL will ring a bell.

CARDS

DIM C$(13); also dimension CARD$ if more than 10 are to be dealt in a hand.
CALL CARDS(C$(),H,CARD$(),FLAG) will shuffle the deck. Subsequent CALLs will randomly select and print out in text the number of cards specified by H; if not that many cards remain in the deck, a value of 0 is given to FLAG and the next CALL will reshuffle the deck. Example -

```
100 CALL CLEAR :: DIM C$(13)
110 CALL CARDS(C$(),5,CARD$(
),FLAG):: IF FLAG=0 THEN PRI
NT "SHUFFLING"
120 PRINT :: GOTO 110
```

CHECKSAY

Put CALL CHECKSAY(A) at the beginning of a program containg speech, and a line
IF A=0 OR A=127 THEN..... before each CALL SAY to skip over it and avoid the silent delay if the Speech Synthesizer is not attached. If it is attached, A will equal 96. Example -

```
100 CALL CHECKSAY(A)
110 IF A=0 OR A=127 THEN 130
120 CALL SAY("HELLO")
130 STOP
```

COUNTER (revised)

CALL COUNTER(N,N$) will return a number N as a string N$ "1st", "2nd", 3rd", etc. Example -

```
100 FOR N=1 TO 100 :: CALL C
OUNTER(N,N$):: PRINT N$;:: N
EXT N
```

HIGHSCORE

CALL HIGHSCORE(S) after the final score S will DISPLAY the previous highest score of the game, or a flashing NEW HIGH SCORE and musical salute. Example -

```
100 CALL CLEAR :: RANDOMIZE
:: S=INT(1000*RND):: DISPLAY
 AT(12,1):"SCORE=";S :: CALL
 HIGHSCORE(S):: FOR D=1 TO 5
00 :: NEXT D :: GOTO 100
```

KEYBOARD

CALL KEYBOARD(K) will respond to CALL KEY(3,K,S) by printing CHR$(K) in the screen position corresponding to its Keyboard position so that the whole Keyboard may be used in games for firing, etc. Example -

```
100 CALL CLEAR
110 CALL KEY(3,K,ST):: IF ST
=0 THEN 110
120 CALL KEYBOARD(K):: GOTO
110
```

TAKETURNS

CALL TAKETURNS asks for number of players, and players' names. Subsequent CALLs will announce each player's turn by name. Example -

```
100 CALL TAKETURNS
110 CALL KEY(0,K,ST):: IF ST
<1 THEN 110 ELSE 100
```

*** NUTS & BOLTS No. 2 *****

Another Nuts Bolts DisK, No. 2, is also available from Tigercub Software for $19.95. It contains another 108 utility subprograms in MERGE format, line-numbered higher than those on this disK so that both can be used without over-writing.

Contents include -

20 character fonts and related routines including giant, enlarged, double-height, double-width, script and sideways and underlined characters, etc.
21 screen display routines including 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.
6 very unusual graphing routines, one for printer.
3 self-changing routines to permit use of a variable line number in GOSUB, GOTO or RESTORE.
4 word processing programs including formatting, plural endings, replacing strings.
15 programming utilities to edit and save screens, print screens, catalog the disK, INIT checK, instant color changes, resets, reading memory size, etc.
Also 4 file handling, 2 menu routines, 6 sorting routines for 2-dimensional arrays, speech, sound effects, etc.
With 10 pages of documentation including an example of the use of each subprogram.