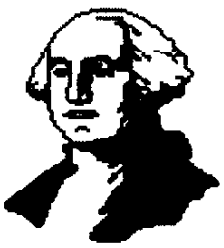
TI-D-BITS

PHILADELPHIA AREA USERS GROUP NEWSLETTER
COVERING THE TI99/4A
AND MYARC 9640 COMPUTERS

FEBRUARY 1993

Volume 13 Number 2

Happy Valentine's Day



The Philadelphia Area TI-99/4A Users' Group meets twice a month. On the second and forth saturday of the month at the Church of the ATONEMENT, 6200 Greene St. Germantown (Corner of Greene St and Walnut Lane) at 10 A.M. We invite anyone that is interested in the TI-99/4A to visit us. Stop in and see what is available to you for your TI and how membership can benefit you!

Current executive board consists of:

PRESIDENT
Norm Sellers..... 215-353-0475

VICE PRESIDENT
Allan Silversteen. 215-885-7910

SECRETARY
Tim Coyne..... 215-947-5881

TREASURE
Don Arsenault..... 215-368-0446

Committees consists of:

TI-D-BITS
Ralph Field..... 215-362-2534
Tim Coyne..... 215-947-5881

LIBRARY
Rich Mascaro..... 215-441-4060

MEMBERSHIP
Ralph Field..... 215-362-2534

EDUCATION
Barry Traver
Allan Silversteen

EQUIPMENT
Allan Silversteen

PROGRAM
(OPEN)

REMEMBER to be considerate when calling any of the above people. Limit your calls to the early evening hours. (6pm to 9pm)

The opinions expressed herein are those of the individual authors and are not necessarily those of the Philadelphia Area TI-99/4A Users' Group or its officers. Nor is the Philadelphia Area TI-99/4A Users' Group or any of its officers responsible for any damage, inconvenience, or loss which may result as a consequence of the use of any written material herein.

TI-D-BITS is published monthly by the Philadelphia Area TI-99/4A Users' Group, c/o Ralph E. Field, 603 N. Broad St., Lansdale, Pa. 19446. All material herein may be reprinted freely by other non-profit User Groups, (unless otherwise stated), as long as proper credit is given to both source and author. Contributions are encouraged, but no payment is made. Editorial, advertising, and classified copy MUST be in by the LAST day of the previous month. You can either mail your copy to: TI-D-BITS, The Philadelphia Area TI-99/4A Users' Group, c/o Ralph E. Field, 603 N. Broad St., Lansdale, Pa. 19446 or send it via modem by contacting Ralph E. Field at (215)-362-2534. If your piece contains any, diagrams, charts, or code, send a paper copy AT FINAL PUBLICATION SIZE.

The editor of TI-d-Bits or the executive board of The Philadelphia area TI-99/4a Users' Group reserve the right to reject any material submitted for publication for any reasons.

The Philadelphia Area TI-99/4A Users' Group's program library is available to all active members at NO CHARGE for copying to your disk. A charge of $2.00 per disk is made for club supplied disks for members. Non members may obtain copies of the library for a fee of $5.00 per disk. A catalog of the library's contents is given to all new members upon request and updates will appear in this publication from time to time. To obtain material from the library, contact the librarian for the best procedure to obtain your requests.

****

N O T I C E

THIS WILL BE YOUR LAST COPY

OF TI-D-BITS IF YOU HAVEN'T

RENEWED YOUR MEMBERSHIP

PROGRAMMING 4-DIMENSIONAL GRAPHICS

by Jim Peterson

Those of you who remember your first lesson in geometry are aware that a straight line has only one dimension, that of length. Ignoring the necessary breadth of one pixel, this can be programmed on the TI by CALL HCHAR(12,1,95,32).

Now, if you fix that one-dimensional line at one end and rotate the other, you will describe a circle, which is of course a two-dimensional figure having length and breadth. This too is easily programmed on the TI using its built-in SGN function.

Proceeding in logical sequence, if you fix that two-dimensional circle at two points and rotate it, you will describe a three-dimensional globe having length, width and breadth. The programming of this will require a slightly more complex algorithm and the radius should be limited to 14 units, since the TI-99/4A screen has only 29 planes.

Proceeding further in logical sequence, if you fix this 3-dimensional globe at three points and rotate it, you will obviously describe a four- dimensional figure. The algorithm required here is somewhat beyond the limits of my high-school gerometry, so I will leave it to some other programmer. The first one to publish this routine will have performed a valuable service to the TI community.

The more observant among you will have detected an apparent fallacy in my line of reasoning. It is impossible, you say, to fix an object at three points and still be able to rotate it. That is a valid argument, and it is perhaps theoretically impossible to describe a 4-dimensional object having perfect symmetry in all four dimensions.

However, it is not necessary to fix one point of a line in order to rotate the other. You may vary the point of fixing during rotation, alternately fix one point and then the other, move both points simultaneously, etc., and thereby create an infinite variety of two-dimensional objects. You might even rotate both points in a third plane, in either the same or opposite directions, and thereby convert a single-dimensional line into a three dimensional cylinder or opposing cones.

Similarly, it is not necessary to maintain the two points on a circle in a fixed position while rotating it. Note that it is not even necessary that the points be opposite, nor that they be moved only in a two-dimensional plane. It is only necessary that they maintain their relative distance from each other.

Therefore, the same obviously holds true for the rotation of an object having three dimensions.

I am sure that some young genius will soon take advantage of this technique to create some truly mind boggling graphics on our TI screen.

****

DEBUGGING

When you have finished writing a program, the next thing you should do is to run it. And, very probably, it will crash!

Don't be discouraged. It happens to the very best of programmers, very often.

So, the next thing to do is to debug it. And you are lucky that you are using a computer that helps you to debug better than some that cost ten times as much.

There are really three types of bugs. The first type will prevent the program from running at all - it will crash with an error message. The second type will allow the program to run, but will give the wrong results.

And the third type, which is not really a bug but might be mistaken for one, results from trying to run a perfectly good program with the wrong hardware, or with faulty hardware. As for instance, trying to run a Basic program, which uses character sets 15 and 16, in Extended Basic.

First, let's consider the first type. The smart little TI computer

makes three separate checks to be sure your program is correct. First, when you key in a program line and hit the Enter key, it looks to see if there is anything it can't understand - such as a misspelled command or an unmatched quotation mark. If so, it will tell you so, most likely by SYNTAX ERROR, and refuse to accept the line.

Next, when you tell it to RUN the program, it first takes a quick look through the entire program, to find any combination of commands that it will not be able to perform. This is when it may crash with an error message telling you, for instance, that you have a NEXT without a matching FOR, or vice versa.

And finally, while it is actually running and comes to something that it just can't do, it will crash and give you an error message - probably because a variable has been given a value that cannot be used, such as a CALL HCHAR(R,C,32) when R happens to equal 0.

The TI has a wide variety of error messages to tell you when you did something wrong, what you did wrong, and where you did it wrong. But, it can be fooled! For instance, try to enter this program line (note the missing quotation mark).

```
100 PRINT "Program must be saved in:"merge format."
```

And, sometimes you may be told that you have a STRING-NUMBER MISMATCH when there is no string involved, because the computer has tried to read a garbled statement as a string.

Also, the line number given in the error message is the line where the computer found it impossible to run the program; that line may actually be correct but the variables at that point may contain bad values due to an error in some previous line.

If the error occurs in a program line which consists of several statements, and you cannot spot the error, you may have to break the line into individual single statement lines. This is the easiest way to do that - Be sure the line numbers are sequenced far enough apart. Bring the problem line to the screen, put a ! just before the first ::, and enter it. Bring it back to the screen with FCTN 8, retype the line number 1 higher, use FCTN 1 to delete the first statement and the ! and ::, put a ! before the first ::, and continue. Then, when you have solved the bug, just delete the ! from the original line and delete all the temporary lines.

Pages 212-215 of your Extended Basic manual list almost all the error codes, and almost all the causes of each one - it will pay you to consult these pages rather than guessing what is wrong.

You may create some really bad bugs when you try to modify a program that was written by someone else - especially if you add any new variable names or CALLs to the program. Your new variable might be one that is already being used in the program for something else, perhaps in a subscripted array. I have noticed that programmers rarely use @ in a variable name, so I always tack it onto the end of any variable that I add to a program.

Also, the program that you are modifying may have ON ERROR routines, or a prescan, already built in. The ON ERROR routine was intended to take care of a different problem than the one you create, so it could lead you far astray - you had better delete that ON ERROR statement until you are through modifying.

The prescan had better be the subject of another lesson, but if the program has an odd-looking command !@P- up near the front somewhere, it has a prescan built in. And if so, if you add a new variable name or use a CALL that isn't in the program, you will get a SYNTAX ERROR even though there is no error. One way to solve this is to insert a line with !@P+ just before the problem line, and another with !@P- right after it.

When a program runs, even though it crashes or is stopped by FCTN 4 or a BREAK, the values assigned by the program to variables up to that point will remain in memory until you RUN again, or make a change to the program, or clear the memory with NEW.

This can be very useful. For instance, if the program crashes with BAD VALUE IN 680, and you bring line 680 to the screen and find it reads
@ CALL HCHAR(R,C,CH
just type PRINT R;C;CH and you will get the values of R, C and CH at the time of the crash. You will find that R is less than 1 or more than 24, or C is less than 1 or more than 32, or CH is out of range.

In Extended Basic, you can even enter and run a multi- statement line in immediate mode (that is, without a line number), if no reference is made to a line number. So, you can dump the curent contents of an array to the screen by
FOR J=1 TO 100::PRINT A(J);: : NEXT J
- or you can even open a disk file or a printer to dump it to.

You can also test a program by assigning a value to a variable from the immediate mode. If you BREAK a program, enter A=100 and then enter CON, the program will continue from where it stopped but A will have a value of 100.

You can temporarily stop a program at any time with FCTN 4, of course (the manual says SHIFT C, but it was written for the old 99/4), and restart it from that point with CON. Or you can insert a temporary line at any point, such as 971 BREAK if you want a break after line 970. Or, you can put a line at the beginning of the program listing the line numbers before which you want breaks to occur, such as 1 BREAK 960,970,980 Note that in this case the program breaks just BEFORE those listed line numbers. You can also use BREAK fol- lowed by one or more line numbers as a command in the immediate mode.

The problem with using BREAK and CON is that BREAK upsets your screen display format, resets redefined characters and colors to the default, and deletes sprites. So, it is sometimes better to trace the assign- ment of values to your variables by adding a temp- orary line to DISPLAY AT their values on some unused part of the screen. If you want to trace them through several statements, it will be better to GOSUB to a DISPLAY AT. And if you need to slow up the resulting display, just add a CALL KEY routine to the subroutine.

Sometimes, your program will appear to be not flowing ing through the sequence of lines you intended (perhaps because it dropped out of an IF statement to the next line!) and you will want to trace the line number flow. This can be done with TRACE, either as a command from the immediate mode or as a program statement, which will cause each line number to print to the screen as it is executed. If used as a command, it will trace everything from the beginning of the program, so it is usually better to insert a temporary line with TRACE at the point where you really want to start. Once you have implemented TRACE, the only way to get rid of it is with UNTRACE.

TRACE has its limitations because it can't tell you what is going on within a multi-statement line, and it will certainly mess up any screen display. Sometimes it is better to insert temporary program lines to display line numbers. I use CALL TRACE( ) with the line number between the parentheses, and a subprogram after everything else

```
AT(24,1):X :: SUBEND
```

Some programmers use ON ERROR combined with CALL ERR as a debugging tool, but I can't tell you much about that because I have never used it ON ERROR can give more trouble than help if not used very carefully, and I cannot see that CALL ERR gives any information not available by other means.

Sometimes you can debug a line by simply retyping it. It is only very rarely that the computer is actually interpreting a line differently than it appears on the screen, but retyping may result in correcting a typo error that you just could not see. In fact, most bugs turn out to be very simple errors.

When you are debugging a string-handling routine, don't take it for granted that a string is really as it appears on the screen - it may have invisible characters at one or both

ends. Try PRINT LEN(M$) to see if it contains more characters than are showing; or
PRINT "*"&M$&"*" to see if any blanks appear between the asterisks and the string.

There is no standard way to debug a program. Each problem presents a challenge to figure out what is going wrong, to devise a test to find out what is really happening.

Don't debug by experimenting by changing variable values just to see what will happen, etc. Even if you succeed, you will not have learned what was wrong so you will not have learned anything - and if your program contains lines that you didn't understand when you wrote them, you will have real problems if you ever try to modify the program. (Believe me, I speak from experience!)

****

GENEVE TRICKS

Taken from MICROpendium

This item is excerted from a column by Tom Arnold that appeared in the Channel User Group newsletter. We do not know which version of MY-Word it relates to.

A rather unusual feature of MY-WORD that I discovered is really neat. Our Myarc programmers are at it again, hiding little features in the programs for us to find.

Do the following: Load MY-WORD, then type "H" select one of the options (anyone will do). Actually, you could just type "EK" for example, and you would accomplish the same thing. Now press ENTER and your help screen appears. Type CTRL-3, your Computer will start to play a tune! I think it is the Little Fugue by Beethoven that appeared for a TI a few years ago. I think it was written in Forth. Anyway, it is a nice tune. You can control it by the number you choose. CTRL-1 playes it the fastest while CTRL-7 is the slowest.

****

ASGARD OFFICE MOVES

Taken from MICROpendium

New address for Asgard Software and Asgard Peripherals is 1423 Flagship Dr., Woodbridge, VA. 22192

Phone numbers are (703) 491-1267, 7-10 p.m. EST; (716) 778-9104. 11 a.m. -7 p.m. EST.

Chris Bobbitt of the company notes that the phone company accidentally disconnected service Dec 23, so customers calling after that date heard a message that the phone was disconnected, with no forwarding number.

****

USING REPLACE STRING FOR LENGTHY BOLDFACE

Taken from MICROpendium

This item from Chick DeMarti, is excerpted from the newsletter of LA 99ers.

Ever want to make an entire line or paragraph bold or underlined when using TI-Writer? Get tired of dozens of @ to bold-face a sentence?

Here's what to do: Put the cursor at the beginning of the sentence. Then go to the Replace String (RS) command and type in the following: / /@/

When the prompt (All,Yes,No,Stop) appears, select "Yes". The will be placed before each word. This method is in lieu of using transliteration codes.

****

TIPS FROM THE TIGERCUB

No. 67

Tigercub Software
156 Collingwood Ave.
Columbus, OH 43213
****

My three Nuts & Bolts disks, each containing 100 or more subprograms, have been reduced to $5.00 each. I am out of printed documentation so it will be supplied on disk.

My TI-PD library now has well over 500 disks of fairware (by author's permission only) and public domain, all arranged by category and as full as possible, provided with loaders by full program name rather than filename, Basic programs converted to XBasic, etc. The price is just $1.50 per disk(!), post paid if at least eight are ordered. TI-PD catalog #5 and the latest supplement is available for $1 which is deductible from the first order.

In a MICROpendium article, Jerry Stern remarked that it would be quite difficult to write a program that would accept input of a formula and then use the formula. He also thought such a program would be very slow.

No programmer could resist a challenge like that, so -

```
100 DISPLAY AT(1,3)ERASE ALL
:"PROGRAMMABLE CALCULATOR":"
":"   V1.1 by Jim Peterson"
:: CALL INIT
110 DISPLAY AT(5,1):" Input
 any mathematical    formula
 in the form of a    valid B
ASIC statement, usingA for t
he value to be calcu-"
120 DISPLAY AT(9,1):"lated a
nd B thru F for the  values
to be input.":" Examples -
":"   A=(B-C)^D+7":"   A=B-C
+C%.1-C%.0575":"   A=INT(ABS
(B-C))-PI"
130 DISPLAY AT(17,1):" To c
hange the formula,    enter
0 for all values."
135 DISPLAY AT(20,1):"This v
ersion can handle FOR/NEXT l
oops, IF THEN ELSE,   MAX, M
IN and <>"
140 DISPLAY AT(24,7):"PRESS
ANY KEY" :: DISPLAY AT(24,7)
:"press any key" :: CALL KEY
(0,K,S):: IF S=0 THEN 140 EL
SE CALL HCHAR(7,1,32,18*32)
150 A$="" :: DISPLAY AT(8,1)
ERASE ALL:"FORMULA?" :: LINP
UT F$ :: ON WARNING NEXT
160 DATA ),102,(,103,-,190,+
,193,-,194,*,195,/,196,^,197
,ABS,203,ATN,204,COS,205,EXP
,206,INT,207,LOG,208
170 DATA SGN,209,SIN,210,SQR
,211,TAN,212,PI,221
175 DATA ::,130,FOR,140,TO,1
77,NEXT,150,STEP,178,IF,132,
THEN,176,ELSE,129,MAX,223,MI
N,22,<,191,>,192,",",179
180 RESTORE 160 :: FOR J=1 T
O 32 :: READ X$,W
190 P=POS(F$,X$,1):: IF P<>0
 THEN F$=SEG$(F$,1,P-1)&CHR$
(W)&SEG$(F$,P+LEN(X$),255)::
 GOTO 190
200 NEXT J :: J=0
205 P=POS(F$," ",1):: IF P<>
0 THEN F$=SEG$(F$,1,P-1)&SEG
$(F$,P+1,255):: GOTO 205
210 IF J=LEN(F$)THEN 240 ::
J=J+1 :: Z$=SEG$(F$,J,1):: I
F POS(".0123456789",Z$,1)=0
THEN A$=A$&Z$ :: GOTO 210
220 N$=N$&Z$ :: Z$="" :: IF
J=LEN(F$)THEN 230 :: J=J+1 :
: Z$=SEG$(F$,J,1):: IF POS("
.0123456789",Z$,1)<>0 THEN 2
20
230 A$=A$&CHR$(200)&CHR$(LEN
(N$))&N$&Z$ :: N$="" :: GOTO
 210
240 A$=A$&CHR$(130)&CHR$(136
)&CHR$(0):: GOSUB 330 :: CAL
L HCHAR(12,1,32,250)
250 W=0 :: IF POS(A$,"B",1)<
>0 THEN DISPLAY AT(12,1):"B=
?" :: ACCEPT AT(12,5):B :: W
=W+B
260 IF POS(A$,"C",1)<>0 THEN
 DISPLAY AT(13,1):"C=?" :: A
CCEPT AT(13,5):C :: W=W+C
270 IF POS(A$,"D",1)<>0 THEN
 DISPLAY AT(14,1):"D=?" :: A
CCEPT AT(14,5):D :: W=W+D
280 IF POS(A$,"E",1)<>0 THEN
 DISPLAY AT(15,1):"E=?" :: A
CCEPT AT(15,5):E :: W=W+E
290 IF POS(A$,"F",1)<>0 THEN
 DISPLAY AT(16,1):"F=?" :: A
CCEPT AT(16,5):F :: W=W+F
300 ON ERROR 310 :: GOTO 320
310 CALL SOUND(400,110,0,-4,
0):: DISPLAY AT(12,1):RPT$("
",250):: DISPLAY AT(24,5):"
INVALID FORMULA" :: RETURN 1
50
320 IF W=0 THEN 150 :: GOSUB
 350 :: DISPLAY AT(18,1):"A=
";A :: GOTO 250
330 CALL PEEK(-31952,A,B)::
CALL PEEK(A*256+B-65534,A,B)
:: C=A*256+B-65534
340 FOR J=1 TO LEN(A$):: CAL
L LOAD(C+J-3,ASC(SEG$(A$,J,1
))):: NEXT J :: RETURN
350 !**********************
**************************
**************************
**************************
```

This method can also be used for the iterative calculator which I published in Tips #65. Just delete lines 100-140, 280-320 and 350 of the above and substitute-

```
100 DISPLAY AT(3,1)ERASE ALL
:"ITERATIVE CALCULATOR V1.1"
:"":"     by Jim Peterson" :
: CALL INIT
110 DISPLAY AT(7,1):" Will
solve difficult equations s
uch as A=X^X-SQR(X)  by iter
ation."
120 DISPLAY AT(11,1):" Inpu
t any mathematical    formul
a in the form of a    valid
BASIC statement, usingA for
the known value and X"
130 DISPLAY AT(15,1):"for th
e value to be deter-  mined.
":" Examples - ":"   A=X^X-
SQR(X)":"   A=SQR(X^X)"
140 DISPLAY AT(20,1):" To c
```

```
hange the formula,  enter
0 for value to calcu- late."

280 DISPLAY AT(12,1):"A=?" :
: ACCEPT AT(12,5):C :: DISPL
AY AT(16,5):"" :: IF C=0 THE
N 150

350 X=1 :: GOSUB 380
351 IF A<C THEN DISPLAY AT(1
4,5):X :: Y=X :: X=X*2 :: GO
SUB 380 :: GOTO 351 ELSE 353
352 IF A>C THEN DISPLAY AT(1
8,5):X :: Y=X :: X=X/2 :: GO
SUB 380 :: GOTO 352
353 IF A=C OR A=B THEN DISPL
AY AT(14,5):"" :: DISPLAY AT
(18,5):"" :: DISPLAY AT(16,5
):X :: GOTO 280 ELSE B=A ::
Z=(ABS(X-Y))/2 :: Y=X
354 IF A<C THEN X=X+Z :: DIS
PLAY AT(14,5):X ELSE X=X-Z :
: DISPLAY AT(18,5):X
355 GOSUB 380 :: GOTO 353
```

Here's a little-known peculiarity of TI XBasic -

```
100 ACCEPT AT(1,1):M$ :: IF
M$="" OR ASC(M$)<32 THEN 100
```

Now, if you press Enter, which is a null string or "" you would expect execution to go back to 100 - but it tries to find the ASCII of a null string, and crashes!

You must write IF M$="" TH EN 100 ELSE IF ASC(M$)<32 TH EN 100 .

And another peculiarity that caused me an hour of total frustration while trying to debug a program - it is well known that CALL KEY in mode 3, CALL KEY(3,K,S), will cause all subsequent INPUT or ACCEPT AT to be in upper case; but what it actually does is internally depress the Alpha Lock, so that ASCII 97 through 122 are read as 65 through 90 - and it disables character sets above 8, ASCII above 95, so that you cannot INPUT or ACCEPT even the printable characters ASCII 96 or 123 through 126, or any FCTN or CTRL input with an ASCII above that.

If you only use the Triton Super Extended Basic module for running programs, not writing them, you may not be aware of some of its most useful features. For example if you are answering an input prompt by typing something shorter over the default on the screen, you don't have to blank out the remaining characters - just use CTRL C. Take a look at page 8 of the manual for other useful features.

In a recentt Tips, I gave a method for reading the entire 13- or 14-digit number which the TI has in memory, by printing it to disk in internal format and reading it back. If I had read the Extended Basic manual more carefully, I would have known a simpler method. If you know where the decimal point will be, just use an IMAGE 14 characters long. Try this - PRINT USING ".#### ##########":17/19.07

If you don't know where the decimal will be, this subprogram will do the job if the number is within the range of -9,999,999,999 to 9,999,999,999; otherwise it will be in exponential notation as usual.

```
100 CALL CLEAR
110 ACCEPT AT(10,1):X :: CAL
L FULLNUM(12,1,X):: GOTO 110
20000 SUB FULLNUM(R,C,X):: P
=POS(STR$(X),".",1):: IF X>9
999999999 OR X<-9999999999 O
R P=0 THEN DISPLAY AT(R,C):X
 :: SUBEXIT
20010 DISPLAY AT(R,C):USING
RPT$("#",P-1)&"."&RPT$("#",1
4-P):X :: SUBEND
```

I worked this one up from a routine in the Swedish newsletter "Programbiten". It will convert to/from any base from 2 to 36.

```
100 CALL CLEAR :: CALL SCREE
N(2):: FOR S=0 TO 12 :: CALL
 COLOR(S,16,2):: NEXT S :: X
$="0123456789ABCDEFGHIJKLMNO
PQRSTUVWXYZ"
110 DISPLAY AT(3,5):"BASE CO
NVERTING" :: DISPLAY AT(10,1
):"From which base?":"":"To
which base?"
120 ACCEPT AT(10,18)VALIDATE
(DIGIT)SIZE(-2):A :: IF A>36
 OR A<2 THEN 120
130 ACCEPT AT(12,16)VALIDATE
(DIGIT)SIZE(-2):B :: IF B>36
 OR B<2 THEN 130
140 DISPLAY AT(14,1):"Number
?" :: ACCEPT AT(14,9)VALIDAT
E(SEG$(X$,1,A)):C$
150 FOR I=LEN(C$)TO 1 STEP -
1 :: D$=SEG$(C$,I,1):: IF AS
C(D$)>57 THEN E=ASC(D$)-55 E
LSE E=VAL(D$)
160 F=F+(E*A^(ABS(I-LEN(C$))
)):: NEXT I
170 FOR J=INT(LOG(F+0.5)/LOG
(B))TO 0 STEP -1 :: G=INT(F/
B^J):: F=F-G*B^J
180 IF G>9 THEN H$=H$&CHR$(G
+55)ELSE H$=H$&STR$(G)
190 NEXT J :: DISPLAY AT(20,
1):H$ :: H$="" :: GOTO 120
```

I have finally replaced my faithful Gemini 10X printer with the new NX1020R and it promptly gave me a problem until I tracked down a serious flaw in its logic. The manual fails to warn you emphasized print cannot be used in combination with condensed print. This is also true of other printers. If you try that combination with them, they condense but do not emphasize. The NX1020 gives me emphasized print but it is not condensed!

The Coco column in Computer Monthly had a contest to write the shortest program to figure first class post-

age. My one-liner is not as short but does a better job.

```
100 INPUT "OUNCES? ":A :: PR
INT .23*(INT(A)-(INT(A)<>A))
+.06 :: GOTO 100
```

Here's how that works. The rate is .29 for the first ounce and .23 for each additional ounce, so we can just multiply ounces by .23 and then add .06 more to the total. However, partial ounces count as full ounces. INT(A) strips off any decimal portion so .23*INT(A) multiplies by the ounces not including the decimal part, if any. (INT(A)<>A) compares A to the integer of A. If they are different, INT(A)<>A has a "truth" value of -1 and a double negative is a plus so 1 is added to the number of ounces to be multiplied by. Otherwise it has a "false" value of 0 so nothing is added.

A self-styled financial adviser has been making the headlines lately by claiming that anyone can become a financial wizard by buying a $19 compound interest calculator. Save yourself $19

```
100 CALL CLEAR :: ON WARNING
 NEXT
110 DISPLAY AT(12,1):"A sum
of $      invested at      %
interest for      years compou
nded      times per      year w
ill become"
120 DATA 12,11,5,13,1,4,13,2
0,2,14,12,3
130 FOR J=1 TO 4 :: READ A,B
,C :: ACCEPT AT(A,B)VALIDATE
(NUMERIC)SIZE(C):N(J):: NEXT
 J
140 FOR J=1 TO N(3)*N(4):: N
(1)=N(1)+N(1)*N(2)/100/N(4):
: NEXT J :: DISPLAY AT(16,1)
:USING "$#######.##":N(1):: R
ESTORE 120 :: GOTO 110
```

But don't believe the answers you get. Such calculations are worthless unless taxes and inflation are considered. Try this one -

```
100 CALL CLEAR :: ON WARNING
 NEXT
110 DISPLAY AT(12,1):"A sum
of $      invested at      %
interest for      years compou
nded      times per      year w
ith tax rate of      %"
120 DISPLAY AT(16,1):"and av
erage inflation rate of
 % will have a buying power
of"
130 DATA 12,11,5,13,1,4,13,2
0,2,14,12,3,15,23,4,17,4,4
140 FOR J=1 TO 6 :: READ A,B
,C :: ACCEPT AT(A,B)VALIDATE
(NUMERIC)SIZE(C):N(J):: NEXT
 J
150 N(2)=N(2)/100 :: N(5)=N(
5)/100 :: N(6)=N(6)/100
160 FOR J=1 TO N(3)*N(4):: I
=N(1)*N(2)/N(4):: I=I-I*N(5)
/N(4):: N(1)=N(1)+I
170 N(1)=N(1)-N(1)*N(6)/N(4)
180 NEXT J :: DISPLAY AT(19,
1):USING "$#######.##":N(1)::
 RESTORE 130 :: GOTO 110
```

By the first method, $1000 invested at 7% for 10 years compounded quarterly would double in value to $2001.60. By the second method, if the interest was taxed at 15% it would still be worth $1950. But if you factor in an average inflation rate of 4% that $1950 would only have a buying power of $1305 - if the price of bread today was $1 per loaf and that price remained constant in relation to wages, you could buy 1000 loaves today or invest the money and buy 1305, not 2001, ten years from now.

I know that this formula is oversimplified, but there is no way to calculate accurately anyway, since future rates of taxes and inflation cannot be predicted.

In Tips #65 I described a method of using DSKU to make the Funlweb Formatter recognize FCTN A, C and Z instead of &, @ and * to avoid garbled program listings. Jan Alexandersson in Sweden says that can be very dangerous. I should have mentioned that you should make the changes to a separate copy of Funlweb which you should not use to print text formatted by others, and you should not distribute text formatted with these alternative codes and those who use the version of TI-Writer which TI sold overseas should not use this method at all, because it uses those FCTN keys for special letters of foreign languages.

```
*************************************

   LOCAL TEMP1 C 40
   LOCAL TEMP2 C 40
   LOCAL TEMP3 C 40
   LOCAL BLNK C 1
   LOCAL ANS N 3 0
     CLEAR
   REPLACE TEMP1 WITH "%E                ";
     ! "  Exp. Date " ! XP
     WRITE 10,3,TEMP1
   REPLACE TEMP2 WITH TRIM(FN) ! " ";
           ! MI ! " " ! LN
     WRITE 12,3,TEMP2
     WRITE 14,3,SA
   REPLACE TEMP3 WITH TRIM(CT) ! " ";
           ! ST ! ". " ! ZP
     WRITE 16,3,TEMP3
  WRITE 22,1,"   Number of Labels"
 READ 22,22,ANS
WRITE 22,1,"                              "
IF ANS > 0                                "
  DO DSK2.PR-LBLS1
ENDIF
 CLEAR
RETURN
*
* DISPNA1     Save as  DISPNA1/C
* *******            11/29/88
*
 *********************************

SET PAGE=000
SET LINE=80
WHILE (ANS > 0)
  WHILE (ANS > 0)
     PRINT TEMP1
     PRINT BLNK
     PRINT TEMP2
     PRINT SA
     PRINT TEMP3
     PRINT BLNK
    REPLACE ANS WITH ANS - 1
    WRITE 22,4,"  Labels To Go =",ANS
    WAIT 1
WRITE 22,4,"                              "
  ENDWHILE
    WRITE 22,4,"More? How many? "
   READ 22,22,ANS
WRITE 22,4,"                              "
ENDWHILE
 CLEAR
RETURN
*            Version 1.02  11/29/88
* PR-LBLS1   Save as PR-LBLS1/C
* ********  Multiple Label Print
*
 *********************************
```

```
CLEAR
COLOR WHITE MAGENTA
WRITE 10,8,"DataBase should be open."
SORT OFF
TOP
EDIT
WRITE 10,8,"DataBase is not closed!"
COLOR WHITE DARK-BLUE
RETURN
*
* EDFL1            Save as EDFL1/C
* *****      EDIT A File  12/02/88
*
 *********************************

CLEAR
COLOR WHITE MAGENTA
WRITE 10,8,"DataBase should be open."
SORT OFF
APPEND
WRITE 10,8,"DataBase is not closed!"
COLOR WHITE DARK-BLUE
RETURN
*
* APFL1           Save as APFL1/C
* *****       APPEND To  12/02/88
*
 *********************************

SET TALK ON
CLOSE ALL
SET HEADING ON
SET RECNUM ON
COLOR WHITE MAGENTA
WAIT 5
CLEAR
DISPLAY STATUS
RETURN
*
* FIN1       Save as  FIN1/C
* ****  Finish Program 11/10/88
*
 *******************************
```

We have previously covered everything in this system. There are a couple of tricks that might be of interest. In INFSCR1 the WHILE loop will not accept a value less than zero or greater than 5, and in PR-LBLS1 there is a WHILE within a WHILE, and they both use the same variable (ANS). I have not typed in complete CFs. I used FunnelWeb to edit CFs we covered earlier, and I deleted and added some lines and then saved them under a new name. Read over the old tutorials. You will find it all there, with explanations. I searched for MM in the label CF, because it is the easiest and least confusing for the user. We created the MM field in tutorial 4. In January 1989 I plan on starting with the new features of TI-Base Version 2.0. I'm sure I will recover some of this old stuff in an effort to compare Ver. 1.02 with Ver. 2.0. If you're lost, don't dispair. Keep those questions and tips coming in. And, I'd like to thank Jerry Keisler of the PARIS 99/4A UG. He gave me several good tips. THANKS JERRY.