8+Ø3 (Ø51)
Upstate

UPSTATE UPSTATE UPSTATE UPSTATE UPSTATE UPSTATE UPSTATE UPSTATE UPSTATE UPSTATE

OUR NEXT MEETING will be on Thursday
MARCH 19,1987 AT 7:30 p.m.

Our APRIL meeting will be
APRIL 16,1987 at 7:30 p.m.

PLACE: CAPITAL DISTRICT PSYCHIATRIC CENTER

New Scotland Ave. Next to Albany Medical Center

Topics for March:
Chuck Eacy will show us the some utilities for FAST-TERM.
Al Smith should receive his GRAM-PACKER utilities to demo.
The Software Library Club should be back from down south.
I hope to arrange an XB demonstration of some type.

A NOTE to other Users Groups: The articles printed in the Upstate Newsletter may be reprinted if proper credit is given to the author and to the Upstate New York 99/4 Users Group.

UPSTATE 99/4A USERS GROUP
P.O. BOX 13522
ALBANY, N.Y. 12212

VOL V , NO. 3 March 1987 Allison Smith, EDITOR 439-4860

THE POWER OF "CALL KEY"

By Steve Langguth Ozark 99'er Users Group

The CALL KEY command in Basic and Extended Basic is one whose complete power may not be appreciated by many programmers. This article and list of examples is an attempt to explain some of the "hidden" capabilities of the CALL KEY statement so that you can get the most out of it in your own programs.

The information in this article was collected from several sources including : an excellent summary of the CALL KEY options, written by Joyce Corker of Waltham, Mass. (the examples that make up the second half of this article are completely hers) which has appeared in several other newsletters recently: and an article by Glenn Davis in the January 1985 edition of the MSP 99 Newsletter.

CALL KEY, as implemented on the TI 99/4A has six possible modes in which to operate. These modes are summarized below.

CALL KEY(0,KEY,STATUS)

When the mode specified is "0", the keyboard is scanned in the same mode it was in previously. (The normal Basic mode is Mode 5 --see below-- so when a CALL KEY(0,K,S) statement is used in Basic or Extended Basic, we are really telling the computer to scan using Mode 5 "just like you were doing before".)

CALL KEY(1,KEY,STATUS)

Mode 1 scans the left side of the keyboard only.

CALL KEY(2,KEY,STATUS)

Mode 2 scans the right side of the keyboard only.

CALL KEY(3,KEY,STATUS)

Mode 3 is the "99/4" mode. In this mode values for upper case letters are returned in "KEY" even if a lower case letter is pressed. (In other words, in this mode it doesn't matter whether the ALPHA LOCK key is up or down, all you get is upper case letters.)

This mode is particularly useful where upper case letters are important. For example, it is recommended that disk file names be in all upper case letters. By putting a CALL KEY(3,K,S) statement before the INPUT or ACCEPT statement, the name typed in by the user will be all in upper case letters. (TI Writer uses this mode when accepting file names.)

CALL KEY(4,KEY,STATUS)

Mode 4 (Pascal Mode) allows upper and lower case letters and all control and function keys. However, some of the "codes" are different than in Basic. For example, FCTN 4 will not "break" a program on an INPUT or ACCEPT statement, FCTN S will not backspace, etc. This is because these combinations of key strokes generate different codes in this mode than in Basic. (See the appendix in the User's Reference Guide.)

CALL KEY(5,KEY,STATUS)

Mode 5 is normal Basic mode and allows for both upper and lower case letters.

EXAMPLES

Below are several examples of how some of the modes described can be put to use.

Yes or no answers using CALL KEY 0

```
100 CALL CLEAR
110 PRINT "Y OR N?"
120 CALL KEY(0,K,S)
130 IF K=78 THEN 170
140 IF K<>89 THEN 120
150 PRINT "YES"
160 GOTO 180
170 PRINT "NO"
180 END
```

Space bar or ENTER answers using CALL KEY 5

```
    100 DISPLAY AT(3,3)ERASE ALL:
        "PRESS SPACE BAR TO CONTINUE" :
        "PRESS ENTER TO PRINT"
    110 FOR DELAY=1 TO 600 ::
        NEXT DELAY
    120 CALL KEY(5,K,S)
    130 IF K=32 THEN PRINT "SPACE BAR
        PRESSED" :: GOTO 150 ELSE
        IF K<> 13 THEN 120
    140 PRINT "ENTER WAS PRESSED"
    150 END
```

Alphabet answers that are forgiving of wrong case using CALL KEY 3

```
    100 DISPLAY AT(3,3)ERASE ALL:
        "PRESS R TO REPEAT" :
        "PRESS P TO PRINT"
    110 FOR DELAY=1 TO 600 ::
        NEXT DELAY
    120 CALL KEY(3,K,S)
    130 IF K=82 THEN PRINT "HERE YOU
        WOULD GOTO YOUR REPEAT
        SUBPROGRAM" :: GOTO 150 ELSE
        IF K<>80 THEN 120
    140 PRINT "HERE YOU WOULD GO TO
        YOUR PRINT SUB"
    150 END
```

Accessing Function and Control Keys using CALL KEY 5

```
    100 DISPLAY AT(3,3)ERASE ALL:
        "PRESS CONTROL KEY AND COMMA"
    110 FOR DELAY=1 TO 600 ::
        NEXT DELAY
    120 CALL KEY(5,K,S)
    130 IF K=128 THEN PRINT "CONTROL
        AND COMMA PRESSED" ELSE 120
    140 END
```

or

```
    100 DISPLAY AT(3,3)ERASE ALL:
        "PRESS FUNCTION 8"
    110 FOR DELAY=1 TO 600 ::
        NEXT DELAY
    120 CALL KEY(5,K,S)
    130 IF K=6 THEN PRINT "FUNCTION
        8 PRESSED" :: GOTO 140 ELSE
        120
    140 END
```

As you can see, the CALL KEY command gives you a great deal of control over the input you are accepting.

Using "PRINT USING" with your printer

One of the more obscure statements available with TI Extended BASIC is one called PRINT USING. Even more obscure is the fact that this statement can be used to format variables and constants that will be dumped to your printer. The Extended BASIC manual, on page 150, shows several examples of how PRINT USING can be used to format data for screen display, but nary a word of how to do the same with open files. It can be done, and is much more powerful than you may realize.

Any discussion of PRINT USING will require an understanding of the IMAGE statement, so if you are not familiar with it, you better brush up on it first. The PRINT USING statement uses IMAGE in one of two ways, either with a string expression, or a line number reference. I prefer the latter, as it allows for more flexibility, but since these different methods are explained in the manual, I will limit this to a few simple examples that are not shown in the manual.

```
100 TCOST=19.55
110 IMAGE ##.##
120 OPEN #1:"PIO"
130 PRINT #1,USING 110:TCOST
```

Running this sample program will effectively show how the PRINT USING statement will work with an open file. Of course, there are many other variations of IMAGE that can be used, so experiment with them and watch how it performs when line 130 dumps it to the printer. Shown below are a few more examples for use with an open file.

```
110 IMAGE "##.##      ##.##"
130 PRINT #1,USING 110:COST1,COST2
```

This IMAGE statement will allow you to print two (or more) variables at a pre-determined spot on the same line. The length of the string expression in the IMAGE statement can be as long as you wish, up to the limit of an Extended BASIC line.

```
110 IMAGE "##########      ##.##"
130 PRINT #1,USING 110:"TOTAL COST",
TCOST
```

This version shows how you can format the printed line for string data as well as numerical data. A string variable could be used in place of the string constant, as below.

```
105 A$="TOTAL COST"
110 IMAGE "##########      ##.##"
130 PRINT #1,USING 110:A$,TCOST
```

It is also possible to place the IMAGE statement inside the PRINT USING statement, as shown below. First, delete line 110.

```
130 PRINT #1,USING "##.##":TCOST
     or
130 PRINT #1,USING "##########     ##.##
"
:A$,TCOST
```

A few other points to remember include the fact that IMAGE and PRINT USING can be used to round off calculated variables. A single string expression such as " ######.##" will round off and decimal align numbers as small as .01 up to 999999.99, and print the number at any designated location. This function could save many hours of algorithm development for accomplishing the same thing. So, in the long run, the PRINT USING statement is one that any programmer should be very familiar with, and use as much as possible.

Just a short note of congratulations to Nick and his wife Debbie on their new son, Michael Nicholas.