TEXT SHIFTER

A must for programs that will run on the Apple II Plus and older Apple II's

One of the many problems Apple programmers face when writing programs for the several varieties of Apple II computers is deciding how to handle upper- and lowercase text. There are millions of Apple II computers out there, and less than a third of them are Apple II's and II Pluses (the two types of Apple II's that can't display lowercase letters). In other words, the majority of Apple II's can display lowercase as well as uppercase. IT'S NO FUN TO READ UPPERCASE TEXT LIKE THIS ALL DAY. UPPERCASE-ONLY TEXT IS HARDER TO READ, AND IT ALWAYS SEEMS TO BE SHOUTING AT YOU! On the other hand, a program written in both upper- and lowercase text is useless to those who don't have a IIGS, IIe, IIc, or lowercase adapter.

Until now, there have been only two solutions to this problem. You could either write programs in uppercase only, or you could keep two versions of the same program on hand. The latter solution can cause more problems than it solves, so many people (as well as software publishers and magazines) have stuck with uppercase text only.

TEXT.SHIFTER (**Listing 1**) is a short machine language utility that will examine an Applesoft program in memory and shift all the lowercase program text to uppercase. It is 48 bytes long, relocatable, and works under DOS 3.3 and ProDOS.

USING THE PROGRAM

TEXT.SHIFTER can be used as a standalone module. You can LOAD an Applesoft program with upper- and lowercase letters in it, BRUN TEXT.SHIFTER, and then save the all-capitals version of the Applesoft program under a new name.

You can also incorporate an Applesoft version of Text Shifter into your Applesoft programs. Write your program using upper- and lowercase. Remember that any letters that will be displayed in inverse must be uppercase. Put the four lines from **Listing 3** at the beginning of your program, and you can give it to anybody you want. No matter which Apple II they have, they will be able to use your program. And you will never need to keep two versions of the same program again!

ENTERING THE PROGRAM

If you have an assembler, simply enter the source code shown in **Listing 1** and assemble. If you do not have an assembler, you can enter the code from **Listing 2** and save it with the command

```
BSAVE TEXT.SHIFTER,A$300,L$31
```

Enter the Applesoft program in **Listing 3** and save it to disk with the command

```
SAVE TEXT.SHIFTER.FP
```

For help with entering *Nibble* listings, see the Typing Tips section.

HOW THE PROGRAM WORKS

TEXT.SHIFTER is a simple program. When you first run TEXT.SHIFTER, it picks up a zero-page pointer from Applesoft that points to the beginning of your Applesoft program. Then, using the Y-register and indirect indexed addressing (better referred to as post-indexed addressing), TEXT.SHIFTER examines each line of the Applesoft program. Let's look at this a bit closer.

Each Applesoft program line starts with a four-byte header. The first two bytes contain the address of the next highest program line in the 6502's usual low-byte/high-byte format. The second two bytes contain the line number, which is not used by TEXT.SHIFTER. The end of the program is marked with a two-byte header, both of which are zeros. These final bytes are pointed to by the previous line, in that line's header.

The actual program line follows the header. This will consist of tokenized Applesoft commands and ASCII data, such as variable names, string literals, and so on. The tokens and the ASCII data are easy to tell apart because the tokens are all greater then 127, and the ASCII data is all less than 128. When TEXT.SHIFTER examines each Applesoft program line, it skips all tokens and then skips anything less than 96 ($60). What remains are lowercase letters and a few symbols that are not found in the Apple II Plus character set, which are converted to uppercase and put back into the Applesoft program line.

Eventually, TEXT.SHIFTER will find a zero in the high byte of the pointer to the next line. When this happens, program control returns to Applesoft, with the program text fully screened of lowercase characters. If your program was actually running at the time TEXT.SHIFTER was called, it will continue as if nothing happened!

Listing 3 shows TEXT.SHIFTER.FP, a short Applesoft program that tests the Apple II identification byte at $FBB3 (−1101). If this byte equals six, the machine is a IIGS, IIe, or IIc, and the program is left alone. Otherwise, TEXT.SHIFTER is POKEd into memory and executed, which shifts all the lowercase program text in memory to uppercase. Then, the modified Applesoft program continues as if nothing happened.

LISTING 1: TEXT.SHIFTER Source Code

```
*
* TEXT.SHIFTER Source Code
* By Steven Meuse
* Copyright(c) 1988
* MicroSPARC, Inc.
* Concord, MA 01742
* Merlin Assembler

T       EQU   0
        ORG   $300
*
        LDA   $67       ;Get pointers to start of
        STA   T         ;Applesoft program, and put
        LDA   $68       ;them into our temp pointer.
        STA   T+1
        LDY   #1        ;Is there a program in memory?
        LDA   (T),Y
        BEQ   DONE
CHECK   LDY   #4        ;4 is the offset to the first
GET     LDA   (T),Y     ;byte of this line.
        BEQ   NEXTLINE  ;If byte = 0, this line is done.
        BMI   NEXTBYTE  ;If hi bit is on, this is a token.
        CMP   #$60      ;Check if text is lowercase.
        BCC   NEXTBYTE  ;No, get another byte.
        AND   #$5F      ;Yes, shift it to uppercase...
        STA   (T),Y     ;...and put it back.
NEXTBYTE INY            ;Point to next byte.
        BNE   GET       ;Always taken.
NEXTLINE LDY  #0        ;Get the pointers to the next line
        LDA   (T),Y     ;and put them into temp pointer.
        TAX
        INY
        LDA   (T),Y     ;This conditions the Z flag for
        STA   T+1       ;the BNE test below.
        STX   T
        BNE   CHECK     ;A>0 if there are more lines
DONE    RTS
        CHK             ;Merlin CHK code = 90
```

END OF LISTING 1

LISTING 2: TEXT.SHIFTER

```
Start: 300              Length: 31

F4 0300:A5 67 85 00 A5 68 85 01
9A 0308:A0 01 B1 00 F0 21 A0 04
DE 0310:B1 00 F0 0D 30 08 C9 60
D9 0318:90 04 29 5F 91 00 C8 D0
48 0320:EF A0 00 B1 00 AA C8 B1
7C 0328:00 85 01 86 00 D0 DF 60
82 0330:9D

TOTAL: A66C
```

END OF LISTING 2

LISTING 3: TEXT.SHIFTER.FP

```
3E  1  IF  PEEK ( - 1101) < > 6 THEN  FOR I = 768
       TO 815: READ J: POKE I,J: NEXT : CALL 768
C2  2  DATA 165,103,133,0,165,104,133,1,160,1,177,0
       ,240,33,160,4
77  3  DATA 177,0,240,13,48,8,201,96,144,4,41,95,14
       5,0,200,208
C1  4  DATA 239,160,0,177,0,170,200,177,0,133,1,134
       ,0,208,223,96

TOTAL: 9BF9
```

END OF LISTING 3