

To control the display format of the characters, routine COUT1 uses the value at location 50 as a logical mask to force the setting of the two high-order bits of each character byte it puts into the display page. It does this by performing the logical AND function on the data byte and the mask byte. The result byte contains a 0 in any bit that was 0 in the mask. BASICOUT, used when the 80-column firmware is active, changes only the high-order bit of the data.

Important

If the 80-column firmware is inactive and you store a mask value at location 50 with zeros in its low-order bits, COUT1 will mask out those bits in your text. As a result, some characters will be transformed into other characters. You should set the mask to the values given in Table 3-5 only.

Switching between character sets is described in the section "Display Mode Switching" in Chapter 2.

If you set the mask value at location 50 to 127 (hexadecimal \$7F), the high-order bit of each result byte will be 0, and the characters will be displayed either as lowercase or as flashing, depending on which character set you have selected. Refer to the tables of display character sets in Chapter 2. In the primary character set, the next-highest bit, bit 6, selects flashing format with uppercase characters. With the primary character set you can display lowercase characters in normal format and uppercase characters in normal, inverse, and flashing formats. In the alternate character set, bit 6 selects lowercase or special characters. With the alternate character set you can display uppercase and lowercase characters in normal and inverse formats.

Original Ile

On the original Apple Ile, the MouseText characters are replaced by uppercase inverse characters.

Standard input features

The Apple Ile's firmware includes two different subroutines for reading from the keyboard. One subroutine is named RDKEY, which stands for *read key*. It calls the standard character input subroutine KEYIN (or BASICIN when the 80-column firmware is active), which accepts one character at a time from the keyboard.

The other subroutine is named GETLN, which stands for *get line*. By making repeated calls to RDKEY, GETLN accepts a sequence of characters terminated with a carriage return. GETLN also provides on-screen editing features.

For more information on GETLN, see the section "Editing With GETLN" later in this chapter.