

Appendix B - Screen and Character Codes

The Intuition Engine uses ASCII (Appendix C) throughout. There is no second character set, no "screen code" remapping, and no PETSCII layer between a program and the glyphs it writes. A byte written to a text-mode buffer is rendered as the glyph at that ASCII code point in the active font.

This appendix documents the small per-engine differences in how the ASCII code reaches the screen.

B.1 VGA text mode

The VGA text buffer at \$B8000 (Chapter 5) holds one character + attribute pair per cell, little-endian:

Byte	Meaning
0	ASCII code point.
1	Attribute byte: bits 0-3 foreground colour, bits 4-6 background colour, bit 7 blink (when blink mode is enabled).

The active font is the standard IBM 8x16 ROM font. Code points \$00-\$1F and \$7F are rendered as the line-draw / shape glyphs of code page 437 rather than as ASCII control characters. Code points \$80-\$FF are the page-437 high half (line-draw extensions, accented letters, mathematical glyphs).

B.2 ULA text mode

The ULA framebuffer (Chapter 8) is purely bitmap; characters are drawn by software rather than by a hardware text mode. The font shipped with the ULA boot ROM is an 8x8 ZX Spectrum-style font covering ASCII \$20-\$7F. Code points outside that range render as blank cells unless the program installs a font of its own.

B.3 TED text mode

The TED video block (Chapter 6) implements a 40 by 25 character grid with 121-colour cells. Each grid cell holds a single ASCII byte plus an 8-bit TED colour byte in colour RAM. Bits 0-3 select the hue, bits 4-6 select the luminance, and bit 7 is used by the text renderer as described in Chapter 6. The font is an 8x8 PETSCII-shape font indexed by ASCII code points, not by PETSCII screen codes. The reader sees the same letter A whether the source byte is \$41 in a PRINT statement, a POKE to the cell, or a machine-code MOVE . B.

B.4 ANTIC + GTIA text modes

ANTIC modes 2, 4, and 6 (Chapter 7) read character data from the character-base pointer in ANTIC_CHBASE. The shipped default points at an 8x8 ASCII-indexed font. A program that wants the Atari "internal code" convention (control characters at low code points, inverse video in the high half) installs its own font table at any 1-KB-aligned address and writes that address to ANTIC_CHBASE.

B.5 The terminal

The terminal (Chapter 38) ingests one byte at a time through `$F0700`. Bytes `$0A` (line feed) and `$0D` (carriage return) move the cursor; `$08` (backspace) erases the previous cell; all other printable bytes draw at the cursor position. Bytes outside the printable range and outside the small set of recognised control codes are dropped silently.

B.6 Summary

A single rule covers every text surface on the machine: the byte is the glyph, the glyph table is ASCII (Appendix C), and the per-engine differences are limited to which extension fonts (page 437, ZX, PETSCII shapes, or Atari internal) supply the bytes above `$7F`.