A HP Terminal Simulator

Version 0.2

Martin Hepperle, Joachim Siebold, February 2025

Content

[Introduction 1](#__RefHeading___Toc2767_1505928153)

[Implementation 2](#__RefHeading___Toc2878_3514471831)

[Required Software 2](#__RefHeading___Toc2724_4266408828)

[Build the Terminal Emulator 2](#__RefHeading___Toc2896_3514471831)

[Run the Terminal Emulator 2](#__RefHeading___Toc2882_3514471831)

[Notes on Serial Connections 3](#__RefHeading___Toc2880_3514471831)

[Notes on Telnet Connections 4](#__RefHeading___Toc3016_3514471831)

[ANSI and HP Mode 4](#__RefHeading___Toc2884_3514471831)

[Keyboard Emulation 4](#__RefHeading___Toc2769_1505928153)

[Cursor Movement 4](#__RefHeading___Toc2875_1505928153)

[Other Keys 4](#__RefHeading___Toc2877_1505928153)

[Recognized Escape Sequences 5](#__RefHeading___Toc2886_3514471831)

[General Control Characters 5](#__RefHeading___Toc2888_3514471831)

[ANSI Sequences 5](#__RefHeading___Toc2890_3514471831)

[HP Single Byte Escape Sequences 5](#__RefHeading___Toc2892_3514471831)

[HP Multi-Byte Escape Sequences 6](#__RefHeading___Toc2894_3514471831)

[Graphics Settings Sequences (Esc \* d) 6](#__RefHeading___Toc2879_1505928153)

[Pen Select Sequences (Esc \* n) 7](#__RefHeading___Toc2881_1505928153)

[Cursor Positioning Sequences (Esc & a) 7](#__RefHeading___Toc2883_1505928153)

## Introduction

This program simulates a classical HP terminal of the 1980s, resembling a HP 2627A with some features of a HP 2648.

It offers text and graphics functions and in its current stage is useable for basic applications using programs like EDIT/1000. Besides the HP-specific Enq/Ack protocol it also supports DC1 handshaking for block transfers.

The goal was NOT to generate a faithful visual representation of a specific HP terminal. Instead the terminal should be useful to work with old real or emulated HP systems of the HP 1000 family. These terminals are quite complex and offer many settings, escape sequences and handshake methods and I implemented only those which I needed.

Therefore, only a subset of the real terminals features are implemented, mostly Forms and Page Mode processing are missing. Also, the definition and execution of the function keys f1 to f8 are not implemented yet.

The interface to the host system is currently via a RS-232C line or a Telnet connection. Peripheral support (tape drives, HP-IB, additional serial interfaces) has not been implemented.

## Implementation

In order to be (almost) platform independent, the emulator is written in Java. Version 0.2 was developped using Java version 1.17. The software has been used it with a USB–RS-232C converter cable (CH340 chip) at 9600 baud and with a Telnet connection to a HP21xx/HP1000 SIMH emulator (<https://simh.trailing-edge.com/hp/>) running RTE-6/VM.

## Required Software

The emulator uses additional libraries for serial communication and Telnet. The serial library (jSSC-2.9.6 Release version, see https://github.com/java-native/jssc ). The library contains modules for native Microsoft Windows and Mac OS X. This library can be distributed according to the “GNU Lesser General Public License” (see <http://www.gnu.org/licenses/>).

For Telnet, the Apache TelnetClient Library is used which is part of the Apache Commons Net Package (see <https://commons.apache.org/proper/commons-net/index.html>) which is released under the „Apache License“ (see <https://www.apache.org/licenses/LICENSE-2.0>).

## Build the Terminal Emulator

The project has been developed in Java using the Eclipse development environment. To build the software without Eclipse you must install a Java compiler, a Java Runtime and the Java build tool Ant.

Running:

ant -f makejar.xml clean

ant -f makejar.xml

builds the file “HPTerminal.jar” in the root directory of the software distribution. This jar file already contains the required serial and telnet libraries.

Directory structure:

|  |  |
| --- | --- |
| **Directory** | **Description** |
| *bin/mh* | compiled code, generated by the build process |
| *src/mh* | source code of classes (see below) |
| *lib* | location of the serial and Telnet library jar files |
| *doc* | generated JAVA documentation |

Class files

|  |  |
| --- | --- |
| **Class** | **Description** |
| *Beeper* | simulation of bell sound |
| *EscapeSequence* | escape sequence parsing |
| *GraphicsScreen* | graphics window |
| *HPTerminalApplication* | the main class |
| *SoftKeys* | soft keys |
| *TerminalScreen* | text window |
| *TerminalSettings* | for saving and restoring settings |
| *VectorFont* | general vector font |
| *VectorFontSimplex* | Simplex vector font for text on graphics screen, derived from HP 150 font |

The main escape sequence parser is located in the *HPTerminalApplication* class.

## Run the Terminal Emulator

You can simply start the emulator without any additional parameters:

java –jar HPTerminal.jar

This will start the program listening to the default serial port “COM1” set to a default speed of 9600 baud.

If you want to select a different serial port and its speed, or use a Telnet connection instead or change other parameters, you can use a few optional command line parameters:

java –jar HPTerminal.jar [-port PORTNAME] [-fontsize FONTSIZE] [-speed BAUDRATE] [-sound {0|1}] [-type {ANSI|HP2627A|HP2648A}] [-logging {0|1}] [-debug {0…}] (for serial connections)

java –jar HPTerminal.jar [-telnethost HOSTNAME] [-telnetport TELNETPORT] [-fontsize FONTSIZE] [-sound {0|1}] [-type {ANSI|HP2627A|HP2648A}] [-logging {0|1}] [-debug {0…}] (for Telnet connections)

|  |  |
| --- | --- |
| -port PORTNAME | Select the serial port for the connection. |
| -speed BAUDRATE | Select a baud rate of the serial port (default: 9600 baud). |
| -telnethost HOSTNAME | Select the hostname of the Telnet server |
| -telnetport TELNETPORT | Selct the port number of the Telnet server (default: 23) |
| -fontsize FONTSIZE | Select a font size, also affects the window size (default: 16). |
| -sound {0|1} | If set to 1 the Bell character will beep (default: 1). |
| -type {ANSI|HP2627A|HP2648A} | Selects the terminal emulation (default: HP2627A). |
| -logging {0|1} | If set to 1 received data is written to “HPTerminal.log” (default: 0). |
| -debug {0|1} | If set to 1 or higher debug output is written to the console (default: 0).  If debug > 1 keyboard input is output.  If debug > 2 the content of the receive buffer is output.  If debug > 50 plot commands are written to “HPTerminal.hpgl”. |

Remember that in Microsoft Windows operating systems higher port numbers have to be specified with a path name like \\.\COM45. In Unix-like system you specify a device like /dev/tty45.

Note: The parameters „-port“ and „-telnethost“ are mutually exclusive!

## Notes on Serial Connections

I observed that some HP systems seem to drive the serial lines with relatively high voltages, which are fine according to the RS-232C standard, but not all USB-serial cables can handle this. The result may be repeated characters and other spurious effects. I could remedy this usually be either adding a voltage divider into the output TX line of the host or simply by inserting a RS-232C indicator block with LED outputs for the signal lines. Obviously this caused a slight voltage drop or had some other beneficial filtering effect.

The same occurred when I connected a HP 2392 terminal to a modern CP/M system with a MAX232 converter and adding a voltage divider to the output line of the terminal solved the problem.

I also tried to run the simulator on a Raspberry Pi with Java 1.6. First with the same USB–RS232-C cable, and then with the built-in serial port and a 3.3 V USB-TTL cable. The USB-Serial cable was recognized as /dev/ttyUSB0 by Linux, but I was not able to configure it properly. I could set the baud rate, but all other settings were obviously ignored. It always produced something like 6-bit data frames intermixed with some erroneous bits. As I found out after much searching on the internet the Linux system did not have a proper driver for the CH-340 USB-cable and after trying for a whole afternoon I finally gave up.

With the serial port /dev/ttyAMA0 on the GPIO-pins communication via a 3.3 V TTL–RS-232C converter worked, but was extremely sluggish.

However, after I replaced the Linux distribution on this Raspberry Pi with a different one, I obtained a reasonable response time. Not perfect, but acceptable. Even the CH-340 USB cable was recognized without additional hacks. So: Linux ≠ Linux.

## Notes on Telnet Connections

A Telnet connection closed by the server is only recognized when the program tries to send information to the host.

The „Pause“ key (sending a break) is silently ignored in Telnet mode.

## ANSI and HP Mode

While the emulator is primarily intended to simulate a HP terminal, it can also be run in ANSI mode. However, it supports only a limited set of ANSI sequences. Even in HP Mode the ANSI sequences are recognized. This behavior might be switched off in future releases, if it would interfere with HP sequences.

## Keyboard Emulation

### Cursor Movement

In ANSI mode the four cursor movement keys send:

|  |  |
| --- | --- |
| Cursor up arrow | Esc [ A |
| Cursor down arrow | Esc [ B |
| Cursor right arrow | Esc [ C |
| Cursor left arrow | Esc [ D |

In HP mode they send:

|  |  |
| --- | --- |
| Cursor up arrow | Ctrl A |
| Cursor down arrow | Ctrl B |
| Cursor right arrow | Ctrl C |
| Cursor left arrow | Ctrl D |

### Other Keys

|  |  |
| --- | --- |
| Cursor up arrow + Control | scroll screen up by 1 line |
| Cursor down arrow + Control | scroll screen down by 1 line |
| Page up | scroll screen up by 1 page |
| Page down | scroll screen down by 1 page |
| Home | cursor to Home Up position (upper left) |
| End | cursor to Home Down position (lower right) |
| Insert | toggle insert mode |
| Insert + Control | insert one line |
| Delete | delete one character at cursor position |
| Delete + Control | delete one line |
| Pause | send a 200 ms Break |
| Scroll Lock | toggle keyboard locked stat |
| F1 to F8 | function keys (not fully implemented yet) |
| F9 | toggles function keys between System and User keys |
| F10 | toggle visibility of the graphics screen. Note: focus must be in text screen window |

## Recognized Escape Sequences

### General Control Characters

|  |  |
| --- | --- |
| ENQ | HP mode: prepare for sending ACK, ANSI mode: send terminal ID |
| DC1 | recognized as a trigger when information is requested from the terminal |
| Esc | Graphics-Text mode: end text and output string, else: start of Esc sequence |

### ANSI Sequences

|  |  |
| --- | --- |
| Esc [ Pn @ | insert characters in line |
| Esc [ Pn A | cursor left |
| Esc [ Pn B | cursor right |
| Esc [ Pn C | cursor down |
| Esc [ Pn D | cursor up |
| Esc [ row;col H | set cursor relative to screen (0-based) |
| Esc [ row;col f | set cursor relative to screen (0-based) |
| Esc [ J | erase from cursor to end of screen |
| Esc [ 0 J | erase from cursor to end of screen |
| Esc [ 1 J | erase from cursor to start of screen |
| Esc [ 2 J | erase complete screen |
| Esc [ K | erase from cursor to end of line |
| Esc [ 0 K | erase from cursor to end of line |
| Esc [ 1 K | erase from cursor to start of line |
| Esc [ 2 K | erase entire line |
| Esc [ 1 L | insert blank line, shift current and remaining lines down |
| Esc [ 1 M | delete current line, shift remaining lines up |
| Esc [ P | delete characters at cursor, shift trailing characters on line left |
| Esc [ X | clear 1... characters (at max up to end of line) |
| Esc [ m | normal character attribute |
| Esc [ 0 m | normal character attribute |
| Esc [ 1 m | highlight character attribute |
| Esc [ 5 m | underline character attribute |
| Esc [ 7 m | inverse character attribute |
| Esc [ 30...37 m | foreground color character attribute |
| Esc [ 40...47 m | background color character attribute |
| Esc [ 6 n | cursor position request, reply: Esc [ <row> ; <col> R (zero-based) |
| Esc [ ? 7 h | enable line wrap |
| Esc [ ? 7 l | disable line wrap |
| Esc [ > 1 s | home down |
| Esc [ > 0 s | home up |

### HP Single Byte Escape Sequences

|  |  |
| --- | --- |
| Esc A | cursor up |
| Esc B | cursor down |
| Esc C | cursor right |
| Esc D | cursor left |
| Esc E | hard reset |
| Esc F | cursor home down |
| Esc G | cursor to left margin |
| Esc H | cursor home up |
| Esc J | clear from cursor to end |
| Esc K | clear from cursor to end of line |
| Esc L | insert line |
| Esc M | delete line |
| Esc P | delete character |
| Esc Q | start insert character mode |
| Esc R | end insert character mode |
| Esc S | scroll text up (view moves down) |
| Esc T | scroll text down (view moves up) |
| Esc X | HP: format mode OFF |
| Esc Z | display functions OFF |
| Esc a | request cursor pos. in memory, reply: Esc & a <col> c <row> R (0-based) |
| Esc b | unlock keyboard |
| Esc c | lock keyboard |
| Esc d | request data, reply: one line + CR from cursor position (after DC1) |
| Esc e | send binary data without handshake, followed by two null bytes |
| Esc g | soft reset |
| Esc h | cursor home up |
| Esc i | backtab |
| Esc l | begin memory lock mode |
| Esc m | end memory lock mode |
| Esc ^ | primary status request, sends Esc \8000020[CR] after DC1 |
| Esc ~ | secondary status request, sends Esc |4506000[CR] after DC1 |
| Esc @ | one second delay |
| Esc 4 | set left margin |
| Esc 5 | set right margin |
| Esc 7 | save cursor and attributes |
| Esc 8 | restore cursor and attributes |

### HP Multi-Byte Escape Sequences

Note that HP-Multi-Byte sequences (a.k.a. “Parameterized Escape Sequences”) having the same prefix can be combined. However, the last, terminating character in a HP multi-byte sequence must be an uppercase character. In this document all sequences are shown with lower case characters.

#### Graphics Plotting Sequences (Esc \* p)

|  |  |
| --- | --- |
| Esc \* p a | lift pen |
| Esc \* p b | lower pen |
| Esc \* p c | recognized, but not implemented |
| Esc \* p d | plot point at current position |
| Esc \* p e | set origin for relocatable plotting |
| Esc \* p f | plot ASCII, absolute (default) |
| Esc \* p g | plot ASCII, incremental |
| Esc \* p h | plot ASCII, relocatable |
| Esc \* p i | plot binary, absolute |
| Esc \* p j | plot binary, short, incremental |
| Esc \* p k | plot binary, incremental |
| Esc \* p l | plot binary, relocatable |
| Esc \* p z | NOP, used as a terminator |

### Graphics Settings Sequences (Esc \* d)

|  |  |
| --- | --- |
| Esc \* d <pen#> a | graphics clear |
| Esc \* d <color#> b | set graphics memory |
| Esc \* d c | graphics screen ON |
| Esc \* d d | graphics screen OFF |
| Esc \* d e | alpha screen ON |
| Esc \* d f | alpha screen OFF |
| Esc \* d k | graphics cursor ON |
| Esc \* d l | graphics cursor OFF |
| Esc \* d q | alpha cursor ON |
| Esc \* d r | alpha cursor OFF |
| Esc \* d s | graphics text mode ON |
| Esc \* d t | graphics text mode OFF |
| Esc \* d <lox,loy,hix,hiy> y | graphics display size in pixel (inclusive) (e.g. Esc \*d0,0,511,389Y) |
| Esc \* d z | NOP |

#### Graphics Attribute Sequences (Esc \* m)

|  |  |
| --- | --- |
| Esc \* m 2 a | set draw mode |
| Esc \* m 3 a | set draw mode |
| Esc \* m <n> b | set line type <n>, <n> in [1…11] as per 2648A, no user defined style |
| Esc \* m <x1,y1,x2,y2> e | fill rectangle absolute (e.g. Esc \*m0,0,511,287E) |
| Esc \* m <n> m | set text size <n> in [1…8] |
| Esc \* m <n> n | set text orientation <n> in [1,2,3,4] |
| Esc \* m o | set text slant ON |
| Esc \* m p | set text slant OFF |
| Esc \* m r | set Graphics defaults |
| Esc \* m <pen> x | set primary pen [0...7] (e.g. Esc \*m0X) |
| Esc \* m z | NOP |

### Pen Select Sequences (Esc \* n)

|  |  |
| --- | --- |
| Esc \* n <pen> x | set graphics text pen to <pen> |
| Esc \* n x | set graphics text pen to track primary pen |

#### Information Request Sequences (Esc \* s)

|  |  |
| --- | --- |
| Esc \* s 1 ^ | read terminal ID, returns: 5 character string e.g. “2627A” + CR |
| Esc \* s 2 ^ | read graphics pen position and state, returns e.g. “+00639,+00399,0” + CR |
| Esc \* s 3 ^ | read graphics cursor position, returns a string like “+00000,+00000” + CR |
| Esc \* s 4 ^ | read graphics cursor position, waits for key press or mouse click, returns a string like “+00345,+00123,065” + CR |
| Esc \* s 5 ^ | read display size and resolution, returns a string in the form “+00000,+00000,+00511,+00389,00003.,00003.” + CR |
| Esc \* s 8 ^ | read zoom status, always returns: “001.,0” + CR for “unzoomed” |

### Cursor Positioning Sequences (Esc & a)

|  |  |
| --- | --- |
| Esc & a <row> r <col> c | row (relative to memory) and column |
| Esc & a <row> y <col> c | row (relative to screen) and column |
| Esc & a <col> c | column only |
| Esc & a <row> r | row only (relative to memory) |
| c | column |
| r | row relative to memory |
| y | row relative to screen |
| +/– | leading sign: relative to current position |

#### Display Enhancement Sequences (Esc & d)

|  |  |
| --- | --- |
| Esc & d @ | end enhancement |
| Esc & d B | inverse |
| Esc & d C | inverse + blink (blink not implemented) |
| Esc & d E | recognized, but not implemented |
| Esc & d F | recognized, but not implemented |
| Esc & d G | recognized, but not implemented |
| Esc & d J | recognized, but not implemented |

#### Keyboard Sequences (Esc & d)

|  |  |
| --- | --- |
| Esc & q <n> L | <n>=0: unlock, <n>=1: lock keyboard |