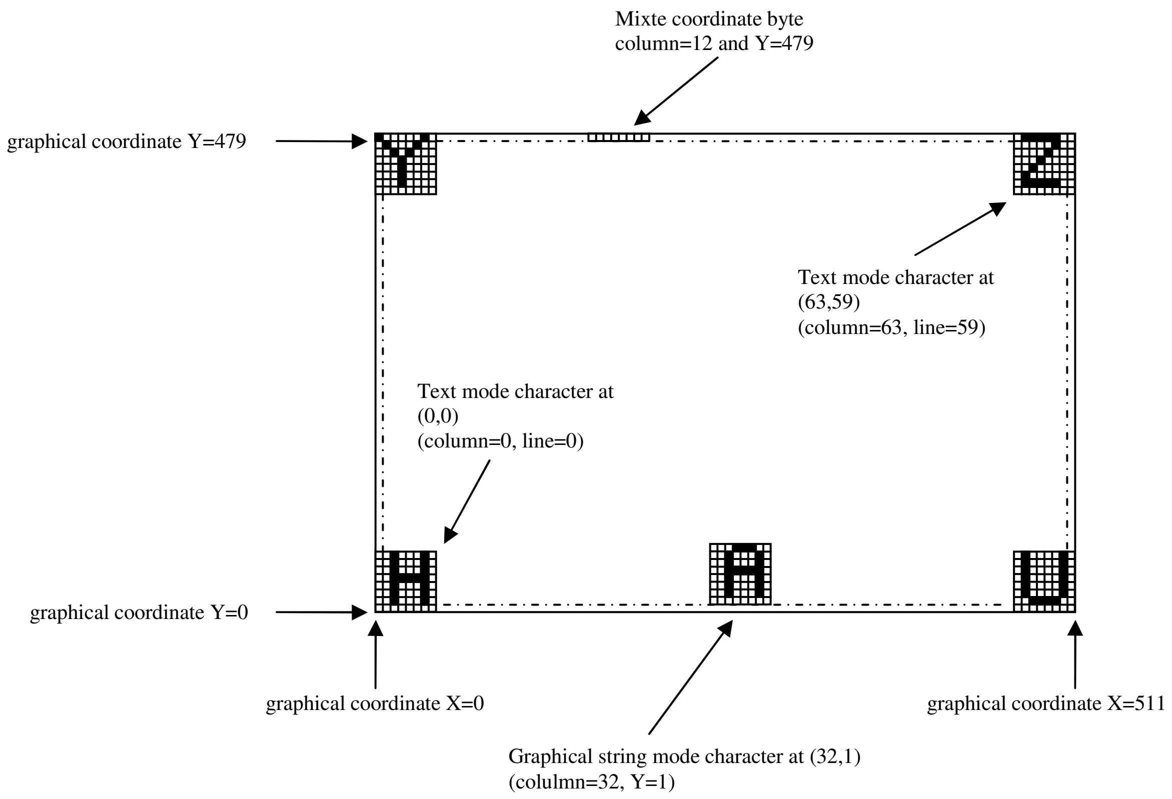
100147 RS232-VGA GRAPHICS CARD

1) Display coordinates

The RS232-VGA card handles text and graphics modes simultaneously:
Text mode: 64 columns by 60 rows, each character is encoded on 8 bits
Graphics Mode: 512 horizontal dots x 480 vertical dots

The coordinate 0,0 is bottom left of the screen in graphical and text mode.

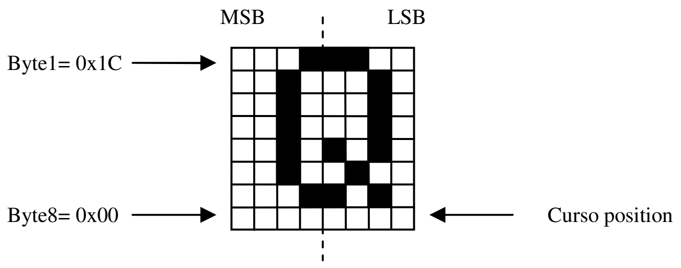
Coordinate limits in text mode: (0,0) to (63,59)
Coordinate limits in graphics mode: (0,0) to (511,479)
Some commands (in graphical text mode) use text type coordinates horizontally and graphics type coordinates vertically: (0,0) to (63,479)



2) Character set

The character set includes a set of standard ASCII characters from space (0x20) to tilde (~, 0x7e), and a set of semi-graphic characters from 0x80 to 0xff.

Each character is defined by a 8x8 matrix (8 bytes, LSB left) describing the character from the top down. For example the character Q is described by: 0x1c 0x22 0x22 0x22 0x2A 0x24 0x00 0x1a.



3) Display modes

Scroll mode
After reset, the card is in "scroll" mode, 9600 baud, the cursor is at position (0,0) (bottom left of the screen). In this mode, it can only receive text and control codes CR, LF, 0x04 or ESC-U to switch to "fixed" text and graphics mode. Code ESC-V selects a different baud rate. The text scrolls from bottom to top of the screen (text is always written on the bottom line), and characters scrolling of the top of the screen are lost.

Fixed mode
Upon receiption of code 0x04, the card switches to a fixed text and graphics mode. It can then receive text, all single byte terminal-type control codes between 0x01 and 0x20, and high-level commands for drawing graphic objects. Text and graphics can be fully mixed, with text having priortiy on transparency (graphic objects are drawn only where its pixels are, a text object overwrites an area of 8x8 pixels).

The text starts at the top left of the screen and scrolls up and down depending on the cursor. The cursor automatically moves to the right, and returns to the beginning of next line if it leaves the screen on the right. When text falls of the bottom of the screen, it will continue at the top left of the screen.

4) Cursor handling

The cursor position indicates the writing position of a character in text mode. Its coordinates range from (0,0) to (63,59). The cursor is not shown on the display, and only its position can be changed. However, there is a control code (0x10), which allows an inversion of the 8 bits that correspond to the position of the cursor (inverted video), which has the effect of 'showing' the cursor. This control allows your program to manage a blinking cursor. Caution: this inversion is purely graphical, and conducted by reading-inverting-writing. If you overwrite a portion of the cursor with a graphic object when it is 'visible', the following extinction of the cursor may surprise you!
Similarly, if you move the cursor before it was extinguished, it will not follow your movement. Think of the cursor as a merely graphic object.

The cursor is moved from a position in the 4 directions by the control codes 0x08 to 0x0B. It will not scroll off the screen. 0x0C to 0x0F codes also change the cursor position.

Codes 0x01 and 0x02 are used to (de)activate auto-increment of the cursor position when writing text characters.

In a character, the cursor is located at the lowermost byte in the 8x8 matrix.

5) Pixel display control

Two codes 0x05 and 0x06 (de)activate displaying of pixels on the screen. This function is useful for standby modes (screen saver). It also useful for creating animations or to speed up graphics calculations. CPU time allocated for graphics calculations is less than 10% when the display is on and more than 90% when the display is off.

This technique has the drawback that toggling the display on and off is perceptible.

6) High level commands

Access to high-level commands is through escape sequences ESC (0x1B). All commands begin with the character ESC (0x1B), followed by a letter that determines the command to execute, then a number of characters depends on the type of command. Be sure to send all the characters as defined in the section detailing the protocol, there are no optional parameters. The card will wait until it has received the exact number of characters as defined by the protocol.

Two command formats are available:

1. The **decimal format** allows you to send commands using the standard ASCII character set. In this format, the letter that determines the command must be upper case.
This format can be used to learn the commands without programming a single line of software. The commands can be sent via a terminal program (f.i. RealTerm) or directly from the keyboard, or through text files. The pixel coordinates are sent as decimal numbers '0' to '9'.
The few characters that are not available on the keyboard can be sent via small files containing one or more control codes. These files are very easy to make from any kind of hex editor (f.i. HexEditor).
For a quick try without using control code files, the ESC-U can switch the card to fixed mode, and most commands can be used immediately. However, commands that only exist in the form of a control code must be sent via a text file if you want to try them.
The author urges you to use this method extensively to understand how this graphics card works and discover what is possible.

2. The **binary format** allows you to send exactly the same commands, but is more suited for microcontrollers.
In this format, the letter that determines the order must be lower case. The pixel coordinates are sent as bytes (one byte for text-type coordinates 0 to 63, two bytes with MSB first for graphics-type coordinates 0 to 511).

7) Control codes

A control code consists of a single byte, in the range 0x01-0x1B. Some were diverted from their original meaning and are therefore not compliant:

0x01: Auto increment the display position after writing a character
0x02: Do not auto increment, the display position is unchanged after writing

0x03: Scroll mode, the text is written on the bottom line and scrolls up
0x04: Fixed text and graphics mode, access to high-level commands

0x05: Video ON, display memory is visible on the screen
0x06: Video OFF, no pixels are drawn, CPU time devoted to calculations

0x08: Move cursor one position to the left
0x09: Move cursor one position to the right
0x0A: Move cursor one position down
0x0B: Move cursor one position up

0x0C: Clear to end of line and move the cursor to the beginning of the line
0x0D: Move the cursor to the beginning of the line (CR)
0x0E: Clear screen and move the cursor top left
0x0F: Move the cursor top left (home)

0x10: bitwise inversion of the byte located at the cursor location

0x1B: Get access to high-level commands (ESC)

It is difficult to directly send certain control codes directly from the keyboard. However, they can be stored in files and sent with a terminal.

8) Decimal-format commands

The decimal-format commands contain only standard ASCII characters, except ESC (0x1B), CR (0x0D) and the characters contained in strings to be displayed (**StringASCII**).

- **Cursor : ESC C CC LL (6 characters)**
  CC is a text column number from ‘00’ à ‘63’
  LL is a text line number from ‘00’ to ‘59’

  Text type command: Move the cursor to (CC,LL).

- **Disk : ESC D XXX YYY RRR Show (12 characters)**

  XXX is a graphics column number from ‘000’ to ‘511’
  YYY is a graphics row number from ‘000’ to ‘479’
  RRR is the radius of the circle from ‘005’ to ‘255’
  Show=’1’ → pixels On, Show=’0’ → pixels Off

  Graphics type command: draw a circle with radius RRR with the centre at (XXX,YYY). The corresponding pixels are shown if Show=’1’.

- **Fill : ESC E Y1Y1Y1 Y2Y2Y2 PPP (11 characters)**
  YYY is a graphics row number from ‘000’ to ‘479’
  PPP is a pattern from ‘000’ to ‘255’

  Graphical text type command (byte mode): Fill the bytes on the lines from Y1Y1Y1 to and including Y2Y2Y2 with the pattern PPP. If PPP =’000’ all pixels will be off, if PPP =’255’ all pixels will be on.

- **Line : ESC L X1X1X1 Y1Y1Y1 X2X2X2 Y2Y2Y2 Show (15 characters)**
  XXX is a graphics column number from ‘000’ to ‘511’
  YYY is a graphics row number from ‘000’ to ‘479’
  Show=’1’ → pixels On, Show=’0’ → pixels Off

  Graphics type command: Draw a line from (X1X1X1,Y1Y1Y1) to (X2X2X2,Y2Y2Y2). The corresponding pixels are shown if Show=’1’.

- **Pixel : ESC P XXX YYY Show (9 characters)**
  XXX is a graphics column number from ‘000’ to ‘511’
  YYY is a graphics row number from ‘000’ to ‘479’
  Show=’1’ → pixels On, Show=’0’ → pixels Off

  Graphics type command: Put a pixel at (XXX,YYY). The corresponding pixel is shown if Show=’1’.

- **Rectangle : ESC R X1X1X1 Y1Y1Y1 X2X2X2 Y2Y2Y2 Show (15 characters)**
  XXX is a graphics column number from ‘000’ to ‘511’
  YYY is a graphics row number from ‘000’ to ‘479’
  Show=’1’ → pixels On, Show=’0’ → pixels Off

  Graphics type command: Draw a rectangle defined by two points (X1X1X1,Y1Y1Y1) and (X2X2X2,Y2Y2Y2). The corresponding pixels are shown if Show=’1’.

- **Graphical string: ESC S CC YYY Size StringASCII CR**
  CC is a text column number from ‘00’ à ‘63’
  YYY is a graphics row number from ‘000’ to ‘479’
  Size=’1’ → characters are 8x8, Size=’2’ → characters are 16x16
  StringASCII : string of displayable characters (ASCII 0x20 à 0xFF)
  CR : end of string delimiter 0x0D (Carriage Return)

  Graphical text type command : Write a string of characters starting at (CC,YYY). Double size characters expand to the right and up. This is a graphical object and display overflow is not checked.

- **Text : ESC T CC LL Direction StringASCII CR**
  CC is a text column number from ‘00’ à ‘63’
  LL is a text line number from ‘00’ to ‘59’
  Direction=’0’ → horizontal, Direction=’1’ → vertical
  StringASCII : string of displayable characters (ASCII 0x20 à 0xFF)
  CR : end of string delimiter 0x0D (Carriage Return)

  Text type command : Write a string of characters starting at (CC,LL), horizontally or vertically. This is a formatted text object occupying 8-byte memory blocks. Display overflow is handled. This command is not affected by the cursor auto-increment commands.

- **Unscroll : ESC U (2 characters)**
  Enter fixed mode (identical to control code 0x04).

- **Speed : ESC V Speed (3 characters)**
  Speed is the baudrate of the RS232 connection, value from ‘0’ to ‘9’

  ‘0’ → 300 baud, ‘1’→600 baud, ‘2’→1200 baud, ‘3’→2400 baud, ‘4’→4800 baud,
  ‘5’→9600 baud, ‘6’→ 19200 baud, ‘7’→38400 baud,
  ‘8’→57600 baud, ‘9’→115200 baud. Default is 9600 baud.

- **Zone : ESC Z C1C1 Y1Y1Y1 C2C2 Y2Y2Y2 PPP (15 characters)**
  CC is a text column number from ‘00’ à ‘63’
  YYY is a graphics row number from ‘000’ to ‘479’
  PPP is a pattern from ‘000’ to ‘255’

  Graphical text type command (byte mode) : Fill the rectangular zone defined by the points (C1C1,Y1Y1Y1) and (C2C2,Y2Y2Y2) with pattern PPP. If PPP =’000’ all pixels will be off, if PPP =’255’ all pixels will be on.

9) Binary format commands

A two byte value must have its MSB first. If you use a text editor hex to text batch files, you will most probably want to convert the decimal data in hex. This will generally not be the case for micro controller compilers, who can perfectly handle the decimal.

- **Cursor : ESC c C L (4 bytes)**
  C is a text column number from 0 à 63
  L is a text line number from 0 to 59

  Text type command: Move the cursor to (C,L).

- **Disk : ESC d XX YY R Show (8 bytes)**

  XX is a 2-byte (msb first) graphics column number from 0 to 511
  YY is a 2-byte (msb first) graphics row number from 0 to 479
  R is the radius of the circle from 5 to 255
  Show=1 → pixels On, Show=0 → pixels Off

  Graphics type command: draw a circle with radius R with the centre at (XX,YY). The corresponding pixels are shown if Show=1.

- **Fill : ESC e Y1Y1 Y2Y2 P (7 bytes)**
  YY is a 2-byte (msb first) graphics row number from 0 to 479
  P is a pattern from 0 to 255

  Graphical text type command (byte mode): Fill the bytes on the lines from Y1Y1 to and including Y2Y2 with the pattern P. If P =0 all pixels will be off, if P =255 all pixels will be on.

- **Line : ESC l X1X1 Y1Y1 X2X2 Y2Y2 Show (11 bytes)**
  XX is a 2-byte (msb first) graphics column number from 0 to 511
  YY is a 2-byte (msb first) graphics row number from 0 to 479
  Show=1 → pixels On, Show=0 → pixels Off

  Graphics type command: Draw a line from (X1X1,Y1Y1) to (X2X2,Y2Y2). The corresponding pixels are shown if Show=1.

- **Pixel : ESC p XX YY Show (7 bytes)**
  XX is a 2-byte (msb first) graphics column number from 0 to 511
  YY is a 2-byte (msb first) graphics row number from 0 to 479
  Show=1 → pixels On, Show=0 → pixels Off

  Graphics type command: Put a pixel at (XX,YY). The corresponding pixel is shown if Show=1.

- **Rectangle : ESC r X1X1 Y1Y1 X2X2 Y2Y2 Show (11 bytes)**
  XX is a 2-byte (msb first) graphics column number from 0 to 511
  YY is a 2-byte (msb first) graphics row number from 0 to 479
  Show=1 → pixels On, Show=0 → pixels Off

  Graphics type command: Draw a rectangle defined by two points (X1X1,Y1Y1) and (X2X2,Y2Y2). The corresponding pixels are shown if Show='1'.

- **String graphique : ESC s C YY Size StringASCII CR**
  C is a text column number from 0 à 63
  YY is a 2-byte (msb first) graphics row number from 0 to 479
  Size=1 → characters are 8x8, Size=2 → characters are 16x16
  StringASCII : string of displayable characters (ASCII 0x20 à 0xFF)
  CR : end of string delimiter 0x0D (Carriage Return)

  Graphical text type command : Write a string of characters starting at (CC,YYY). Double size characters expand to the right and up. This is a graphical object and display overflow is not checked.

- **Texte : ESC t C L Direction StringASCII CR**
  C is a text column number from 0 à 63
  L is a text line number from 0 to 59
  Direction=0 → horizontal, Direction=1 → vertical
  StringASCII : string of displayable characters (ASCII 0x20 à 0xFF)
  CR : end of string delimiter 0x0D (Carriage Return)

  Text type command : Write a string of characters starting at (CC,LL), horizontally or vertically. This is a formatted text object occupying 8-byte memory blocks. Display overflow is handled. This command is not affected by the cursor auto-increment commands.

- **Unscroll : ESC u (2 bytes)**
  Enter fixed mode (identical to control code 0x04).

- **Speed : ESC v Speed (3 bytes)**
  Speed is the baudrate of the RS232 connection, value from 0 to 9

  0 → 300 baud, 1→600 baud, 2→1200 baud, 3→2400 baud, 4→4800 baud, 5→9600 baud, 6→ 19200 baud, 7→38400 baud,
  8→57600 baud, 9→115200 baud. Default is 9600 baud.

- **Zone : ESC Z C1 Y1Y1 C2 Y2Y2 P (9 bytes)**
  C is a text column number from 0 à 63
  YY is a 2-byte (msb first) graphics row number from 0 to 479
  P is a pattern from 0 to 255

  Graphical text type command (byte mode): Fill the rectangular zone defined by the points (C1,Y1Y1) and (C2,Y2Y2) with pattern P. If P=0 all pixels will be off, if P=255 all pixels will be on.

10) Auto test and quick start

The card has a small demo which doubles as self-test. To enable this demo, the RxD pin must be kept low during power up, by either connecting pins 3 & 5 of J3 connector or by placing a jumper on the 3-pin connector J5. Exiting the demo is done by a reset, power cycle or removing the grounding of RxD.

To do a quick test, connect a PC to J3 (SUBD9) of the VGA card: only 2 wires are necessary, mass (pin 5) and the connection from TX of the PC to RX of the card (pin 3).
Launch a communication software of some kind (RealTerm is highly recommended, you can also use Terminal or Hyperterminal, but beware of bugs!). Configure the COM port of the PC, the speed at 9600 baud, 8 data bits, no parity, 1 stop bit, no flow control. Connect a 9VDC power supply to the VGA card. Type a letter on the keyboard, it must appear at the bottom left of the screen, because the card is in terminal mode (scroll mode) at power up. Press ESC then type U to switch to fixed mode. If the characters are now displayed in the upper left of the screen, your card is ready for operation.

If you are using Hyperterminal, typing of characters on the keyboard causes sometimes surprising results: a wrong character every two identical characters. To avoid this, switch to Minitel (file / properties / settings / Minitel emulation). If this problem persists, switch to auto detect (file / properties / settings / detect. auto).

11) Command files

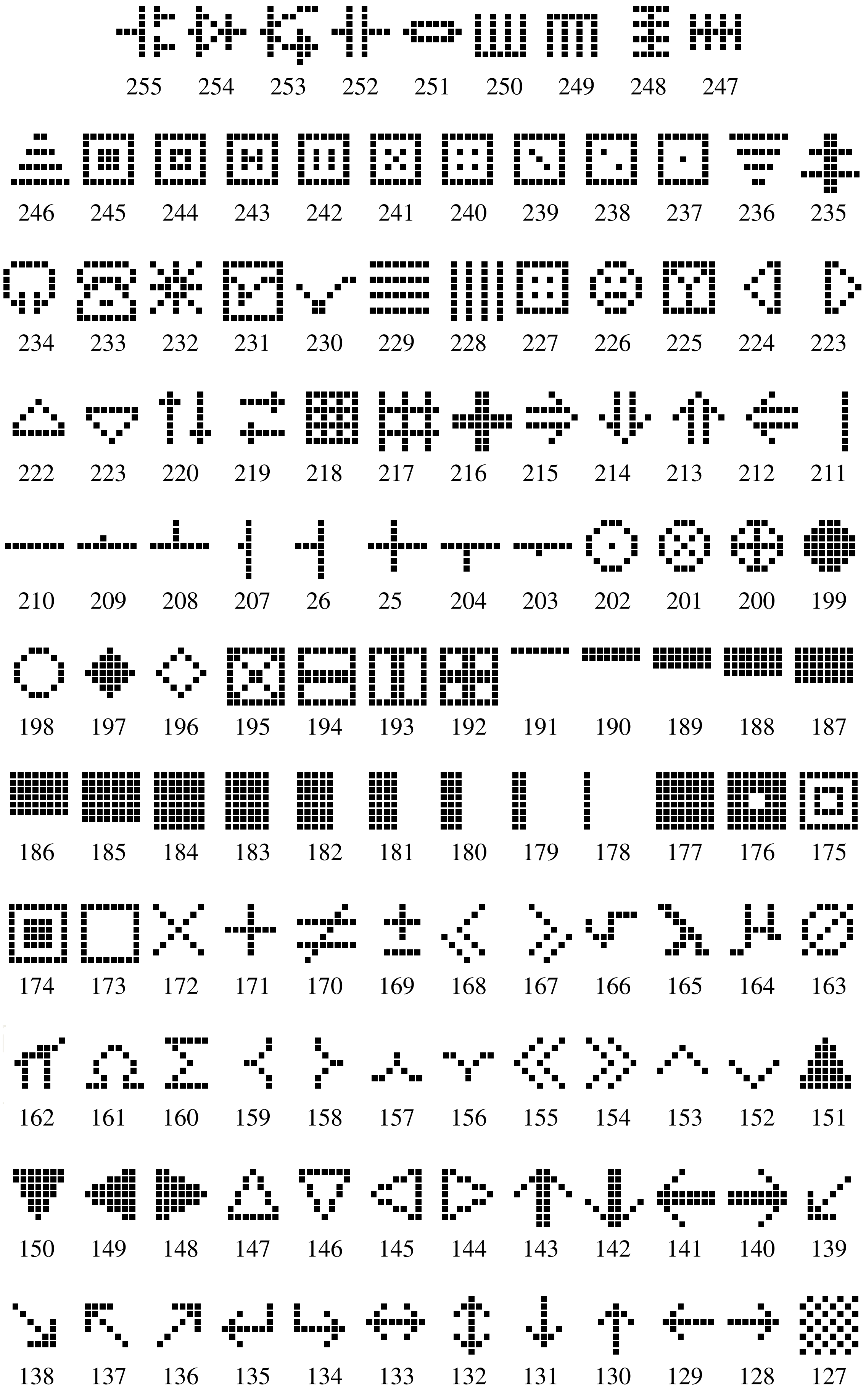
Command files are particularly useful for building, visualizing and refining the framework of a graphical project. Put the control codes, commands and displayable characters in a file, just as if you had sent through the keyboard. This text file is then transmitted to the graphics card using a terminal program.

Connect your card to a PC via the RS232 connector (J3), and power it by J1. Start a terminal configured to 9600 baud on the PC (in RealTerm, Port menu and then '9600' and 'Change'), send a letter with the keyboard, it must appear at the bottom of the screen. Press ESC then type U, the screen goes blank and the card goes into graphics mode.

Send the file to the card 'exemple1.txt' (on RealTerm, Menu 'send' and then select the file 'exemple1.txt' then 'send file'), and observe the result on screen. If you open the file 'exemple1.txt' with a text editor, or even better with a hex editor, you can easily make the connection between the screen and the file contents. Observe the order in which the commands are send, and the effect of priority when text is superimposed on a graphics object (bottom of the screen).

12) Character set

The characters 32 to 126 are standard ASCII. The characters 127 to 255 are specific to the RS232-VGA card:


255 254 253 252 251 250 249 248 247
246 245 244 243 242 241 240 239 238 237 236 235
234 233 232 231 230 229 228 227 226 225 224 223
222 223 220 219 218 217 216 215 214 213 212 211
210 209 208 207 26 25 204 203 202 201 200 199
198 197 196 195 194 193 192 191 190 189 188 187
186 185 184 183 182 181 180 179 178 177 176 175
174 173 172 171 170 169 168 167 166 165 164 163
162 161 160 159 158 157 156 155 154 153 152 151
150 149 148 147 146 145 144 143 142 141 140 139
138 137 136 135 134 133 132 131 130 129 128 127

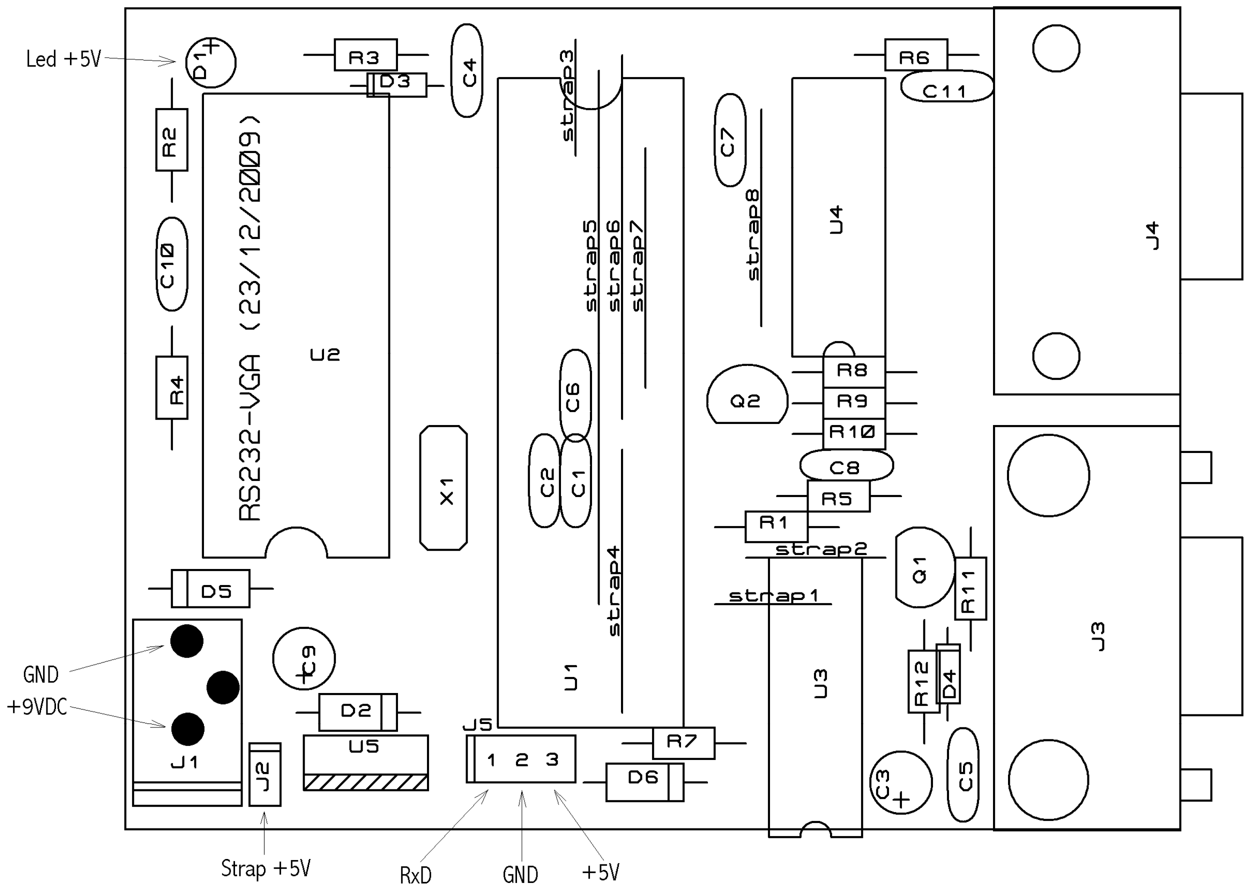
13) Technical support

The author may be contacted for technical questions regarding this project. (PCB, programming of dsPIC, etc.) by e-mail:
etiennes33@yahoo.fr

Terminal ‘RealTerm’: http://sourceforge.net/projects/realterm/files/

Hexadecimal editor ‘Hex Editor’: http://www.hhdsoftware.com/Family/hex-editor.html

Inputs & output of the VGA card



Led +5V
GND
+9VDC
Strap +5V
RxD
GND
+5V
RS232-VGA (23/12/2009)
U1
U2
U3
U4
U5
J1
J2
J3
J4
J5
1 2 3
D1
D2
D3
D4
D5
D6
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
Q1
Q2
X1
strap1
strap2
strap3
strap4
strap5
strap6
strap7
strap8