BV4141

VT100 Mini-Term

Product specification July 2009 V0.a

Contents

Rev	Change
July 2009	Preliminary

1. Introduction

The BV4141 is a 132 x 132 pixel colour display that provides serial input. This then acts as a mini display terminal accepting text, graphics and pictures using a subset of VT100 commands and ordinary text.

It can be connected to a standard RS232 port (with conversion), Microcontroller UART or A BV103 USB to serial converter for direct connectivity to a PC.

The display is primarily intended for a replacement to the usual LCD type displays although the text size is much smaller. It can be used for Microcontroller applications or directly with a PC, displaying system information for example.

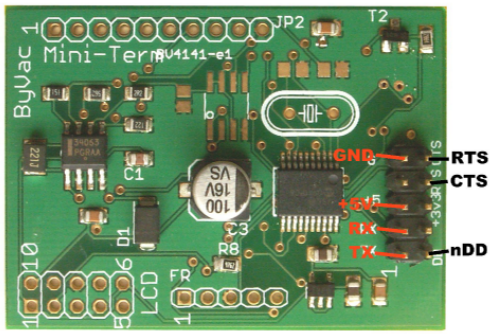
For programming examples see www.asi.byvac.com

2. Features

- Serial Input
- Automatic Baud rate from a selection of rates up to 112500
- Multiple displays on same Com port
- VT100 subset
- 8 bit colour
- Two Fonts normal and bold
- User definable sign on screen
- Supply voltage from 3.5* to 9V DC
- 3.3V logic
- Current: 35mA @ 5V, 45mA @ 3.3V, 13.5mA @5V with back light off.
- Size 40mm x 50mm x 22mm overall. Display size 27mm x 27mm

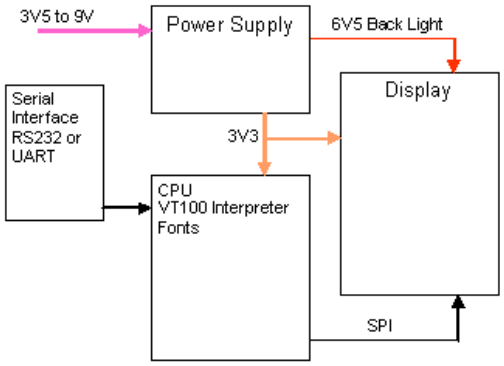
* see text (Power supply)

3. Physical Specification



NOTE: For this board only the CTS is not used and it is marked incorrectly on the PCB. The above illustration shows the correct position of RTS.

The BV4141 is a PCB that provides power and a serial interface to a Nokia type N61 display. It can be supplied in various formats, see www.byvac.com for details. The following text describes the full system including the display.



Block Diagram

Display

The display on its own is quite complex to drive and requires at least 6.6V for the back light. This is all provided by the subsystem. It is configured for 8 bit colour that will give 256 different colours, this is adequate for pictures as well as graphics and text.

Power Supply

The power supply provides the voltage for the back light and also for the CPU. It has a step down regulator so that any input (within limits) is converted to 3.3V this being the standard logic signals that the display uses. From here is stepped up to 6.5V to drive the back light. Using this arrangement any DC supply voltage from 3.5V to 9V can be used for correct operation.

NOTE: The system will work best with a supply voltage of 4V or more. This depends on the display tolerances. If the voltage is too low the contrast will vary producing a display that changes in brightness form time to time.

CPU

The CPU provides the VT100 interpretation, Serial Peripheral Interface output for the display, Font information and Baud rate detection. The Flash memory holds two complete Fonts, one normal text and the other bold text. There is also 240 bytes of EEPROM that can be programmed by the user to provide a splash screen at start up.

3.1. Serial Interface

The serial interface requires 3.3V logic levels although 5V logic will work satisfactorily. A direct connection to RS232 signals is not available. A suitable device for translation such as a MAX232 for RS232 signals or the BV101 range for a USB interface.

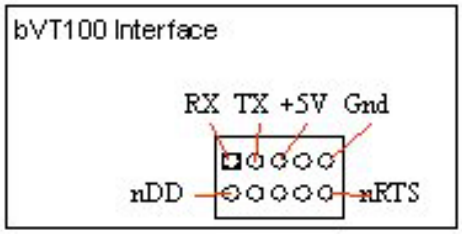
The BV4141 is an input only device and does not provide any serial (TX) output. It does however provide an RTS signal for connecting to the host CTS line for handshaking purposes.

Serial Connector

The serial connector is a 2x5 pin at the back of the device. Pin 1 is marked with 1, odd pins are down one side and even pins down the other.

Pin	Function	Function	Pin
1	RX	nDD	2
3	TX		4
5	+5V		6
7			8
9	Gnd	nRTS	10

This bVT100 serial connector will mate directly with the BV103. For this device the TX line is not used as the display does not output any information, it is a read only device.



This interface relies on hardware handshaking, without it the device will work but it will not be able to perform at its maximum speed. See below in the nRTS section.

RX This is the serial receive and expects signals 0 to 5V. The idle state is high (5V).

This shows a typical serial signal at 115,200 Baud. The byte sent is 1 and so the first signal is the start bit followed by a pulse high that represents 1. The scale is 2V per square. The latter part is 2 stop bits which are low and so cannot be differentiated form the 0's that are being transmitted. It follows a standard RS232 type protocol with 1 start bit and 1 or 2 stop bits so connection to just about any asynchronous serial interface will do provided the signalling is 0-5V

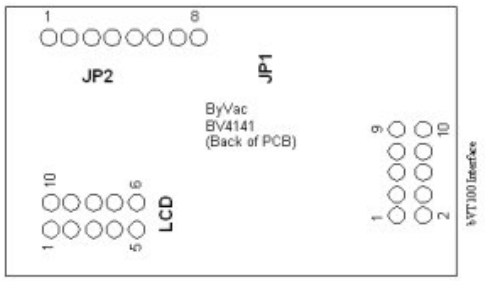
TX This signal is not used for this device as it is read a read only device.

+5V Is the main power supply for the display which should be capable of providing 40mA if the back light is going to be used.

nRTS (output) Hardware handshaking is an essential part of this interface. Without it the display will work but delays must be included after each character otherwise information will be lost. This line is held low (0V) by the display when it is ready to accept data, it will put this line high when it is busy, the transmitter should monitor this and only send characters when the line is low. This signal is normally connected to the CTS line of the transmitting device.

nDD This is called the Display Disable and if left disconnected the line is pulled high by an internal pull up resistor, thus enabling the device. If the line is taken low then the display becomes disconnected from the serial interface. This is so that more than one device can be connected to the same serial bus.

If the line is pulled low, the displays UART will be disabled and the RTS line will also be disconnected, effectively removing the display from the circuitry. This will take a finite time to accomplish (approximately 10uS) and so this must be taken into account when switching between displays.



PCB Connections

There are three user connections on the PCB, the diagram shows these looking from the back.

LCD Connector

Pin	Description
1	VCC [1]
2	#Reset
3	Data
4	Clock
5	#CS

6	VCC [1]
7	No Connection
8	Ground [2]
9	LED – [2]
10	LED +

Notes:
[1] These pins are connected to each other.
[2] These pins are connected to each other.

This connector provides a one to one connection with the display itself and for this application should not be used. It is provided for a non-populated PCB (a PCB without components on) to allow a manageable connection from the display to other equipment.

bVT100 Connector

See the Serial Interface section

JP2

Pin	Description
1	nDD Pin 2 of bVT100 connector
2	Reserved
3	Reserved
4	Reserved
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Power Pin 5 of bVT100 connector
10	Ground

This connector is mostly reserved for future use.

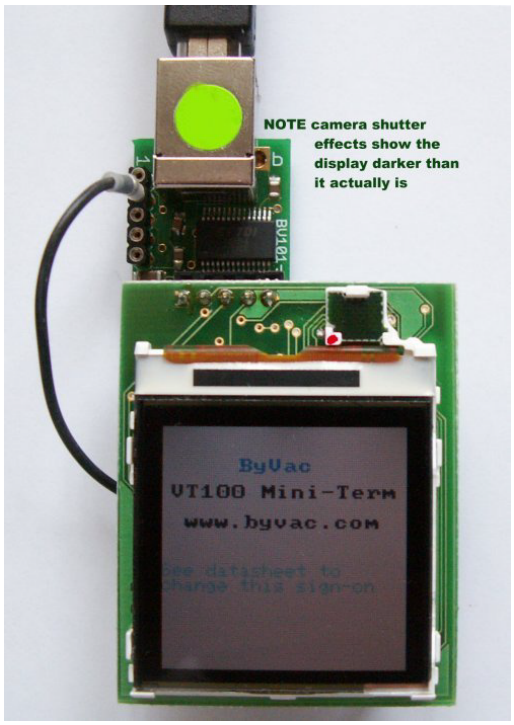
4. Connecting

As previously mentioned there are at least three ways to connect the Mini-Term to a serial interface.

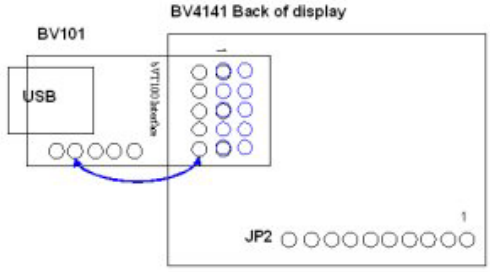
4.1. USB to TTL

This will allow direct communication with a PC, it not only provides the power required but also offers handshaking that will allow fast downloads of pictures and the like.

BV101 (old type)



The BV101 is a USB to TTL convertor that will (almost) plug straight into the BV4141. It has a 5V output and 5V logic levels.



The diagram shows the best way of connecting to a BV101. This is looking at the back of the display. The row sockets nearest the USB connector should go the pins furthest away from the BV4141 edge. This leaves a row of pins free at the edge of the BV4141 PCB and a row of sockets free at the edge of the BV101. A connection is then required from pin 10 of the BV4141 to the auxiliary connector pin 2 on the BV101 as shown. This connection provides the CTS<>RTS handshaking.

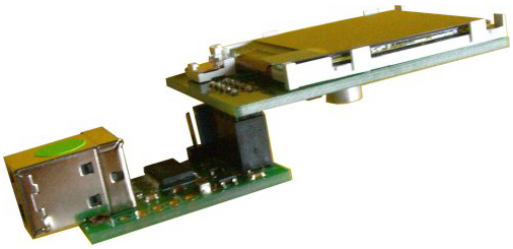
BV102

The BV102 is a USB to serial converter similar to the BV101 and has 3V3 logic levels, however it will only output 3V3 volts and this voltage slightly is too low for this device so It is not recommended.

BV101 (new type) BV103

The BV103 will plug directly into the BV4141. If using the BV103 then 3V3 is ideal position

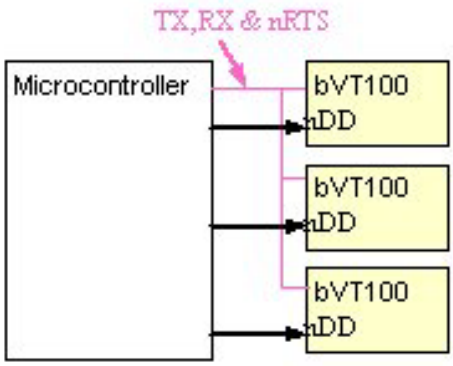
for the logic voltage jumper, 5V will still be supplied to pin 5.



BV103 + BV4141

4.2. Microcontroller & Multiple Displays

The BV4141 can be directly connected to a Microcontroller UART provided arrangements have been made for a CTS line for handshaking.

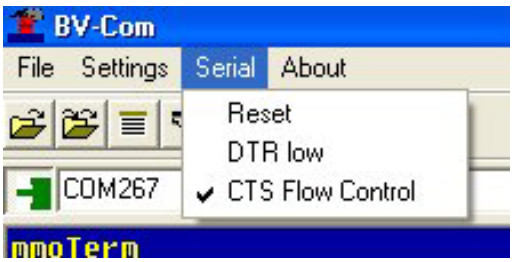


It is possible to control more than one display by using the nDD signal line. This is similar to Chip Select except it does the opposite, when asserted (taken low) it deselects the device.

In this example 3 GPIO lines are used from a microcontroller, at any time two of them will be low and one high to select one display.

5. Using

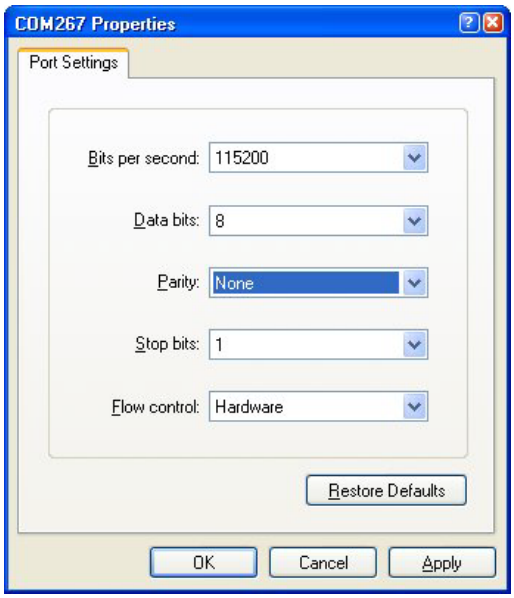
A standard terminal emulator is required such as BV-Com or HyperTerminal. Alternatively a microcontroller can be used. BV-Com can be obtained from www.asi.byvac.com and is a standard terminal emulator. Providing the nRTS line has been connected and a suitable method of connection as above has been established then the following settings are required:



1) make sure the CTS Flow control is checked or leave the nRTS line disconnected. Leaving the nRTS disconnected will seriously degrade the performance.

2) not essential but having the echo ticked will enable you to see what is typed.

HyperTerminal could also be used:



The above settings should work with hyperterminal.

6. Automatic Baud Rate

When the display is first switched on or has been reset with the appropriate VT100 command, the FIRST byte that MUST be sent is a Carriage Return (enter on the keyboard) this is byte value 13 or 0x0d in hex. The device will be able to establish the Baud rate from this single byte. Until this happens the device cannot be used.

The Baud rate is determined form the following standard rates and select one of these based on that first byte:

9600, 14400, 19200, 38400, 57600, 115200

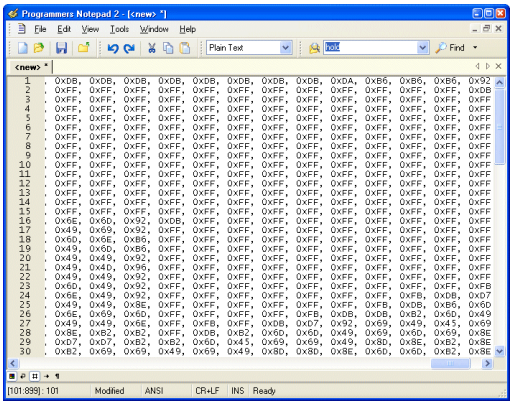
The transmitting terminal must be set at one of those rates. In some situations the transmitter may not be accurate and give unreliable results on the higher baud rates. If this is the case the rates can be fine tuned. See the appropriate commands in the VT100 section

7. Images

For this example the image components.bmp' will be used.

Open the file converter 'rgb2bmp.exe' and load the image. This will create a C header file; for this application we just need the data, without the '0x' and commas.

Copy and paste the data into a text editor (programmers notepad is ideal for this):



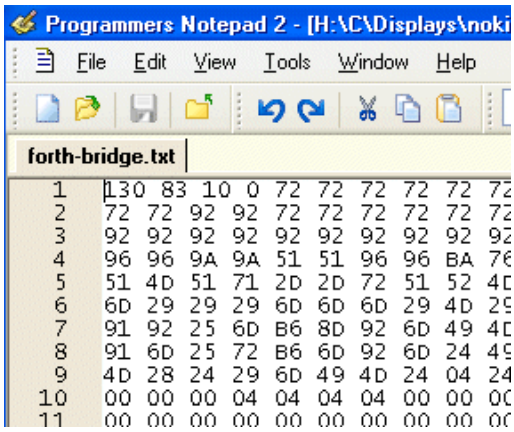
Use search and replace to remove the '0x'. Replace the '0x' with nothing, use search and replace to replace all the commas with nothing.

You will then end up with a text file consisting of hex values separated by spaces. At the beginning of the file place 4 decimal numbers representing width, height, row and column. The width and height must match the picture and this can be obtained form the created header:

#define image_width (130)

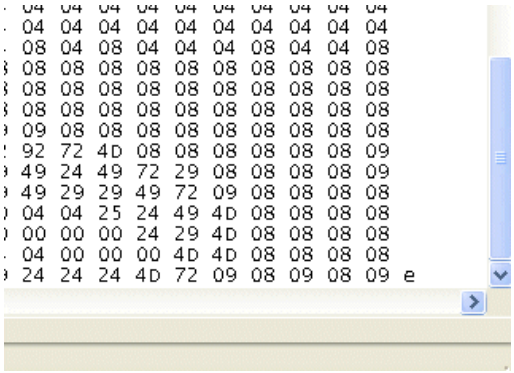
#define image_height (83)

It is up to you where to place the image on the screen, I have chosen 10 0



The start of the file is width, height, row and column.

At the end of the data place a lower case 'e'



Save this text file and load it using escb (escape followed by lower case b NO SPACES)

This command could always go at the front of the file of course.

8. VT100

Once connected the display simply accepts text and duplicates it to the screen. Connecting a terminal emulator such as BV_Comm or HyperTerminal will enable anything typed into it to be duplicated on the display.

To control the display, for example clearing the screen or placing letters in a certain location escape codes are used. An escape code is the escape key (byte 27) followed by various symbols, numbers and letters.

Escape codes have been used for many years and were a feature of the early terminals. The most popular set of codes is called VT100 used by old DEC terminals. It was so popular that it became a standard way to control terminal screens before the days of the PC.

This display wherever possible uses the most common of these codes.

8.1. Implemented Escape Codes

In the table the left hand column indicates:

S is the VT100 standard

P is partially standard

N is non-standard

S	Code	Bytes decimal	Description
Y	LF	10	Line feed (10)
Y	CR	13	Carriage return (13)
Y	BS	8	Back space (8)
P	esc(1	27,40, 49	Normal font
P	esc(2	27,40, 50	Bold font
Y	esc[?25I	27,91, 63,50,5 3,73	Hide cursor
Y	esc[?25h	27,91, 63,50,5 3,104	Show cursor
N	esc[?26I	27,91, 63,50, 54,73	Back light off
N	esc[?26h	27,91, 63,50, 54,104	Back light on
N	esc[?29 numa	see text Paragra ph 9.10	Writes to EEPROM Baud rate fine tune: 38,400
N	esc[?29 numb	see text Paragra ph 9.10	Writes to EEPROM Baud rate fine tune: 57,600
N	esc[?29 numc	see text Paragra ph 9.10	Writes to EEPROM Baud rate fine tune: 115,200
N	esc[?30a	27,91, 63,51,4 8,97	Displays current status of display
N	esc[8h	27,91, 56,104	Clear screen on bottom scroll (default)
N	esc[8l	27,91, 56,73	Don't clear screen [1]
N	esc[?27D	see text Paragra ph 9.11	Write defaults to EEPROM
N	esc[?27M	see text Paragra ph	Write sign on message to EEPROM
Y	esc[2J	27,63, 50,74	Clear screen using default background colour
N	esc[3J	27,63, 51,74	Clear screen and reset all defaults
Y	esc[<line> ;<col>H	see text Paragra ph 9.1	Move cursor to line and column [2]
Y	esc[<num >A	see text Paragra ph 9.2	Move cursor up one or more lines, <num> is optional
Y	esc[<num >B	see text Paragra ph 9.2	Move cursor down one or more lines, <num> is optional
Y	esc[<num >C	see text Paragra ph 9.2	Move cursor right one or more lines, <num> is optional
Y	esc[<num >D	see text Paragra ph 9.2	Move cursor left one or more lines, <num> is optional
Y	escD	27,68	Move cursor down one line
Y	escM	27,77	Move cursor up one line
Y	esc[H	27,63, 72	Cursor home
P	escc	27,99	Reset device
N	escb	see text	Load bitmap
N	esc[numf	see text Paragra ph 9.3	Sets foreground colour
N	esc[numb	see Text Paragra ph 9.3	Sets background colour
N	esc[numV		Sets contrast
Y	esc[K	27,63, 75	Clear line from cursor right
Y	esc[1K	27,63, 49,75	Clear line from cursor left
Y	esc[2K	27,63, 50,75	Clear entire line
N	esc{<row >;<col>P	see text Paragra ph 9.4	Places single pixel in current foreground colour

N	esc{<row>;<col>p	see text Paragraph 9.5	Places pixel without showing it.
N	esc{y0 x0 y1 x1L	see text Paragraph 9.6	Draws a line in the current foreground colour
N	esc{y0 x0 y1 x1B	see text Paragraph 9.7	Draws a box in the current foreground colour
N	esc{y0 x0 y1 x1F	see text Paragraph **Error! Reference source not found.**	Draws a filled box in the current foreground colour
N	esc{ y0 x0 <dia>C	see text Paragraph 9.8	Draws a circle in the current forground colour

Notes

[1] the letter following number in the escape code is lower case L

9. VT100 Commands

All VT100 commands begin with escape, this is byte decimal 27 or hex 0x1b. The commands are interpreted immediately as they are typed in, there is no need to follow with CR or LF. Most of the time the command is determined by the final character, for example turning the back light on and off is exactly the same command except for the final character.

Some commands are obvious and so will not be covered in detail. The less obvious commands are below.

9.1. esc[<line>;<col>H (Move Cursor)

This command is used to position the cursor on the display. The cursor whether visible or not determines where the next character is placed. Line is a number beginning at 0 and ending with one less then the number of lines available on the display. On an 8 line display the first line would be 0 and the last line would be 7. Column is the number of characters across the display, also starting at 0. It is calculated using the current font so some unexpected results could occur if the font has been changed within the line. Either the line or column values can be left out.

For the BV4141 the line and column values are:

Line 0-15

Column 0-24 (font 1) 0-15 (font 2)

esc[0;5H Moves the cursor to the first line 6th character along

esc[4;H Moves the cursor to line 5 retaining the current horizontal (column) position

esc[;3H Moves the cursor to the 4th character position on the current line.

See also esc{<row>;<col>p

9.2. esc[<num>A Move Cursor Up

Also commands:

esc[<num>B (down)
esc[<num>C (right)
esc[<num>D (left)

Moves the cursor by one or more character spaces or lines. The character space (horizontal) is calculated form the current font.

The number specifies how many character spaces to move and is optional, if left out a default of 1 is taken so:

esc[1B is the same as esc[B

9.3. esc[numf Set Foreground Colour

Also **esc[numb** (background colour)

The foreground and background colours are represented by 1 byte, that is decimal 0 to 255 where 0 is black and 255 is white. The colours are represented in binary as follows:

rrrgggbb

The following table gives some standard colours:

Binary	Hex	Decimal	Colour
00000000	0	0	Black
11111111	ff	255	White
11100000	e0	224	Red
00011100	1c	28	Green
00000011	3	3	Blue
11111100	fc	252	Yellow
11100011	e3	227	Magenta
00011111	1f	31	Cyan

9.4. esc{<row>;<col>P (Place Pixel)

Places a single pixel drawn in the foreground colour at the position given. The row is the vertical height and the column is the horizontal position, accepted values are:

row 1-130

column 1-130

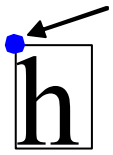
Where 1,1 is the top left corner of the screen.

To erase a pixel the change the foreground colour and use this command.

9.5. esc{<row>;<col>p (Move to Pixel)

The functionality of this command is similar to that of 9.4 except that it does not draw a pixel. It is mainly used for positioning and can be used to position text in the horizontal direction more accurately than the 'H' command.

This command with this device is able to define exactly the top left hand corner of the character box



The blue dot illustrates the point where this command specifies, the character is presented to the right and down from it.

This can be very useful for centering text titles or producing special effects by overlapping characters.

9.6. esc{y0 x0 y1 x1L Draw Line

This will draw a line in any direction, the starting co-ordinates are given by y0,x0 and the ending co-ordinates by y1,x1.

Y is the vertical co-ordinate and can have a value between 0 and 63. Zero is the top of the screen and 63 is the bottom.

X is the horizontal direction and can have values between 0 and 127. O being the left most position and 127 being the right most.

9.7. esc{y0 x0 y1 x1B Draws a Filled Box

Draws a box, y0,x0 specify the top left corner of the box and y1,x1 specify the bottom right corner of the box. y1 must be greater then y0 and x1 must be greater then x0 otherwise unexpected results may occur.

Y is the vertical co-ordinate and can have a value between 1 and 130. Zero is the top of the screen and 130 is the bottom.

X is the horizontal direction and can have values between 1 and 130.

The box will be filled in the current foreground colour.

9.8. esc{ y0 x0 <dia>C Draws a Circle

Draws a circle, the centre given by y0,x0.

9.9. Graphics Considerations

It is up to the user **not** to specify co-ordinates that are outside the available range. Some checking is done and can be tolerated, trial an error will give the best results. Be prepared to cycle the power as some incorrect commands will crash the display and stop it working until the power is cycled.

The circle drawing is not perfect as an attempt is made to fit into the pixels available. A very small diameter will produce a diamond rather then a circle.

9.10. esc[?29 numa Fine Tune Baud Rates

At higher Baud rates some hosts are not particularly accurate in the Baud rate timing. To prevent any mismatch between the host and this device the Baud rates can be fine tuned using this and its sister commands:

esc[?29 numa Fine tune 38,400 (51)
esc[?29 numb Fine tune 57,600 (34)
esc[?29 numc Fine tune 115,200 (16)

The number in the brackets gives an accurate timing for the respective Baud rates, the lower the number the faster the baud rate. The command writes a number to the EEPROM for use with the Autobaud detect. As detection only occurs at reset, the effect on any of these commands will not be seen until reset.

As an example suppose that the Baud rate of 115,200 gave unreliable results, the first try may be to slow it down and so these commands could be used:

esc[?29 17c (writes 17 to EEPROM)

escc (resets device)

Set the host to 115,200 and press enter (CR), see if the results are any better.

9.11. esc[?27D Write EEPROM Defaults

Some settings can be stored to EEPROM so that these become the default at start up. This command will write the settings to the EEPROM which will be used when the device is reset. The following shows which values are saved.

Default	Name
63	Contrast
255	Background colour
0	Foreground colour
0	Cursor colour
1	Font
on	Cursor visible
on	Clear screen when scrolling

	from bottom to top
51	Fine tune for 28,400 Baud rate
34	Fine tune for 57,600 Baud rate
16	Fine tune for 115,200 baud rate

The values are programmed using the 'Write defaults to EEPROM command (esc[?27D).

9.12. esc[?27D Sign On

The sign on message is stored in EEPROM and there are approximately 220 bytes available for a message that will be displayed at start up or reset. This message is displayed BEFORE the autobaud detect.

The message is made up of exactly the same escape codes and text that would form a normal display. The only difference is that it must end with CTRL z (^z or byte value 26)

esc[?27D

… message body and formatting codes

^z

The above is what the message should look like in order to be written to EEPROM. The final ^z is important. This command if not properly used may cause the display to stop working, if this happens then a Hardware reset is required.

Because the message is made up of normal text and escape codes it is probably wise to test it before committing to EEPROM.

10. Hardware Reset

The purpose of the hardware reset is to reset the EEPROM back to a known position so that the display will work correctly. The procedure is:

1) Disconnect the display from power
2) Connect the reset pins together (see below)
3) Apply power
4) Disconnect power

The display will now work correctly. The hardware reset can be found by looking for 5 holes in a line, one of the holes will have a square copper pad and marked with either 1 or FR or both. Connect this copper pad to the one next to it with a bit of wire.

11. Quirks

11.1. Baud rate

When using the highest Baud rate of 115200 it has been found that the transmitting device has to be highly accurate. This has been found to be not always the case with some microcontrollers. The symptoms are intermittent garbled text. The cure is to reduce the Baud rate or fine tune the receiving rate using the EEPROM settings.

11.2. Circles

Drawing of circles is not perfect. A radius of 4 or smaller circles will become a square. Should the circle go out of the drawing area then pixels may appear on non related parts of the screen.

11.3. Error Checking

There is very little error checking and as a consequence it is possible to 'crash' the display if an incorrect or impossible command is entered. Recovery from this will need a power recycle.