VideoNuLA

User Guide

Version 1.3
February 2018

Contents

Introduction

When first released in 1981, the Acorn BBC Microcomputer boasted a range of advanced facilities that were either the equal of, or superior to, those found in other home micros.

As time moved on, other micros caught up, and even surpassed, the capabilities of the "Beeb". Third-party suppliers responded with a wealth of upgrades and Acorn produced newer models, typically with more memory and even more advanced features.

However, the video system was one area in which Acorn did not introduce any improvements and every model in the BBC range uses a similar custom-designed integrated circuit (IC), called the Video ULA, to produce graphic displays. Although capable of producing a range of screen modes and high resolutions that surpassed its rivals, it only provides a palette of eight solid colours and a further eight flashing colours.

This limits the choices of colours for users and software writers and, as newer machines from other manufacturers entered the market, the BBC Micro's palette could look garish in comparison.

VideoNuLA is designed to address this (albeit belatedly!) and give access to a wide range of colours. It significantly increases the BBC Micro's palette well beyond the capabilities of its 8-bit contemporaries and into the realm of the later 16-bit and 32-bit machines. By allowing existing colours to be redefined in a way that is completely invisible to software, VideoNuLA can transform existing programs and open up exciting possibilities for new ones.

It also provides attribute-based graphic and text modes that increase the number of available colours on screen in medium and high-resolution modes.

Finally, it adds hardware support for single-pixel horizontal scrolling in all graphic modes unlocking effects that previously required prodigious coding skills.

Whether you simply want to revitalise old games or develop new software to make use of its advanced features, VideoNuLA opens up a world of new graphic possibilities for your Beeb!

Acknowledgements

I would like to thank members of the Stardot forum (www.stardot.org.uk) for their support and encouragement during the development of this project. In particular, I am very grateful to Dominic Beesley and Jason Flynn for their help and advice when designing the VideoNuLA PCB, to Richard Broadhurst and Rich Talbot-Watkins for suggesting enhancements, to Jonathan Hartson for his help in reviewing this manual and developing the support ROM and to Simon Hooper and Tom Williamson for encouraging me to develop the prototype design.

1 Before installation

Before installing VideoNuLA, please check that you have received the following:

- VideoNuLA board
- 28-pin IC socket
- Flying-lead with spring clip probe
- VideoNuLA support ROM

VideoNuLA provides an analogue colour palette so you will also need a display capable of accepting an analogue signal and a suitable cable. The cable must connect the computer's RGB socket to the display's analogue input (e.g. an RGB to SCART lead). **VideoNuLA is not designed to work with the computer's composite or UHF outputs.**

The VideoNuLA board replaces the original Video ULA integrated circuit used in the BBC Models A, B, B+, Master and Master Compact computers. This is fitted into a socket on some models but is soldered to the motherboard on others.

To identify whether your video ULA is socketed or soldered, you must first remove the computer's lid. Instructions on how to do this can be found in the relevant Welcome guides – it usually involves removing four screws located at the underside and/or rear of the computer. **Please ensure that the machine is not plugged into the mains electricity supply before proceeding.**

Once you have removed the computer's lid, locate the Video ULA IC. This is usually marked as follows:

- IC6 on the BBC Models A and B
- IC53 on the BBC Model B+
- IC42 on the BBC Master
- IC16 on the BBC Master Compact

On some models, typically earlier model A and B machines, the Video ULA will be fitted with a heatsink and should therefore be easily identified.

It may be necessary to remove internal upgrades (such as ROM boards, co-processors etc.) before you can identify the Video ULA IC and/or fit the VideoNuLA board. Where possible, you should consult the relevant manufacturer's instructions on how to do this.

Removal of a socketed Video ULA IC

If the original Video ULA IC is fitted in a socket, ensure that the machine is not powered and remove the Video ULA IC from its socket. *[One technique for this is to slowly lift alternate sides of the IC using a small, flat-bladed screwdriver.]*

The Video ULA IC should now be placed into some of the conductive foam provided for safe keeping.

You can now move on to the Installation section.

Removal of a soldered Video ULA IC

If the existing Video ULA IC is soldered to the PCB, it will need to be desoldered and the supplied 28-pin IC socket installed in its place. **Only attempt this if you are confident that you have the required skills as an error could cause damage to your computer. I cannot be held responsible for any damage that occurs so if you have any doubts about your ability to carry out this procedure please contact me.**

To desolder the existing Video ULA IC, first remove the motherboard from the case - instructions on how to do this can be found in the relevant service manual.

The 128K variant of the model B+ may have a wire soldered to a pin of the Video ULA IC. Note the position of this wire and remove the connection to the Video ULA IC before attempting to remove the IC itself. Following installation, this wire should be soldered to the equivalent position on the VideoNuLA board.

Once the motherboard has been removed, locate the 28-pin Video ULA IC connections on the underside of the motherboard and desolder each of the pins in turn. *[I have found that using a good quality solder sucker, desoldering braid and the odd application of new solder works well.]*

Once you are confident that all of the pins are free, the Video ULA IC can be lifted from the board and the supplied 28-pin socket soldered in its place.

Check that the removal procedure and fitting of the socket has been successful by fitting the original Video ULA IC into the socket and testing the machine. After reassembling the machine, remember to test it in a variety of screen modes to ensure that it is operating correctly. **If the machine doesn't boot successfully, do not leave it powered-on for too long as certain faults can cause damage to the machine's RAM chips.** Remove the motherboard and test for continuity and shorts.

Assuming that the process has been successful, follow the procedure in the section above and remove the Video ULA IC. You can then move onto the Installation section.

2 Installation

The VideoNuLA board can now be fitted in place of the original video ULA IC. **<u>Again, ensure that the machine is not connected to the mains electricity supply before removing the lid and fitting the VideoNuLA board.</u>** Figure 1 below shows the correct orientation of the board for each model. Ensure that you have the board in the correct orientation and that it is lined up with the 28-pin socket before inserting it. When inserting, alternately apply gentle and even pressure to both sides of the board as this will prevent the pins from bending.

[The model B+ has a crystal and header either side of the video ULA IC. These prevent the VideoNuLA board from being inserted directly into the socket. This can be overcome by either stacking a couple of 28-pin sockets or by using a socket adapter so that the VideoNuLA board sits above the crystal and header.]

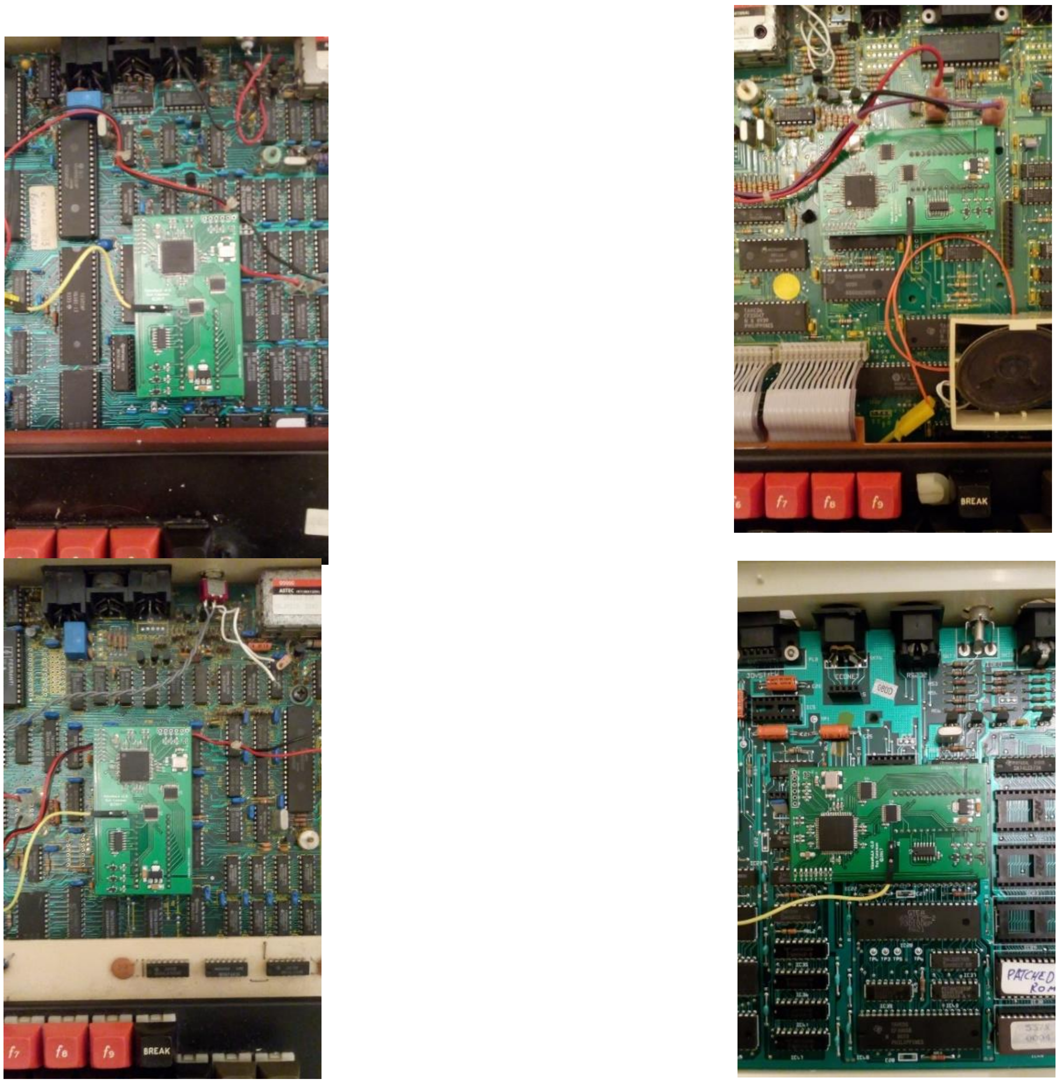


Figure 1: VideoNuLA installed in a model B (top-left), Master 128 (top-right), B+ (bottom-left) and Master Compact (bottom-right)

Once the board is in place, attach the fly-lead to the connector marked 'A1' as shown in the pictures above. Then connect the spring clip to a source of the A1 address line. Suitable locations include:

- Pin 37 of IC3 (6522/System VIA) on models A and B (40-pin IC)
- Pin 9 of IC71 (or any populated ROM socket) on the model B+ (28-pin IC)
- Pin 37 of IC6 (6522/User VIA) on the BBC Master (40-pin IC)
- Pin 37 of IC25 (6522/System VIA) on the BBC Master Compact (40-pin IC)

Figures 2 and 3 below show how to identify the relevant pins. If the suggested locations are not suitable for your configuration either consult the relevant service manual to identify other sources or contact me to discuss.

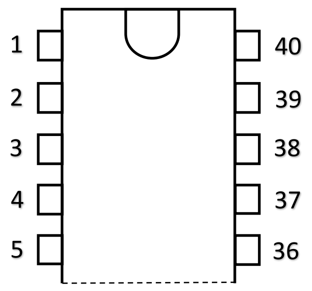


Figure 2: Identifying the pins on a 40-pin IC

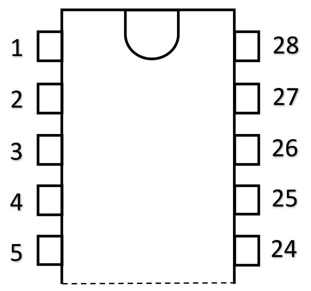


Figure 3: Identifying the pins on a 28-pin IC

The lid can now be refitted and the machine powered on. It should boot successfully but, if it fails to boot, **turn off the power as soon as possible.** As mentioned above, the machine's RAM chips can be damaged if they are powered on for too long without the refresh signals they require. In the event of a problem, disconnect the power, remove the lid and check that you have fitted the VideoNuLA board correctly, including the fly-lead connection to a source of the A1 address line. If you are unable to resolve the problem, please contact me.

If the machine has booted successfully, test the board in a variety of screen modes to ensure that it is operating correctly. Once you are satisfied that the machine is working correctly, any internal upgrades that were removed earlier can be reinstalled.

You can now move on to installing the VideoNuLA support ROM which greatly simplifies the use of VideoNuLA and provides many extra features.

However, VideoNuLA will function without the ROM if you are short of free ROM sockets or prefer not to install it. The VideoNuLA support disk image contains a copy of the support ROM image if you wish to use sideways RAM and other utilities. See Appendix F for a complete list of the disk image contents.

Installation of the VideoNuLA support ROM

<u>If you are installing the support ROM inside the computer, ensure that the machine is not connected to the mains electricity supply before removing the lid.</u>

<u>The support ROM has to be installed in a free paged ROM socket. The instructions below assume that it is to be installed within one of the machine's original sockets. If you intend to install the ROM in a third-party add-on (such as a ROM board or cartridge), please consult the manufacturer's instructions instead.</u>

ROM installation in a BBC Model B

The paged ROM sockets are located at the bottom-right corner of the PCB under the keyboard and are marked as IC51, IC52, IC88, IC100 and IC101.

The keyboard is removed by unscrewing the two bolts that hold it in place and detaching the keyboard connector and speaker cable from PL13 and PL15 respectively. (If you are in any doubt, full details can be found in the BBC Microcomputer Service Manual.)

Identify a free ROM socket - IC51 and IC52 usually contain the operating system and BASIC ROMs respectively and should be left in place. Remove the VideoNuLA support ROM from its protective foam and inspect the ROM's pins and ensure that none are bent or out of alignment. Ensure the ROM is orientated correctly by identifying the notch and making sure that it is pointing toward the back of the PCB.

Gently place the ROM in the chosen socket, checking that the ROM's pins are correctly aligned with the socket receptacles. Slowly apply even downward pressure until the ROM is securely in place.

You can now refit the keyboard, connectors and lid.

ROM installation in a BBC Model B+

The paged ROM sockets are located at the top-left corner of the PCB and are marked as IC35, IC44, IC57, IC62, IC68 and IC71. **<u>Note: IC29 and IC37 are used for the speech upgrade and are not paged ROM sockets.</u>**

Identify a free ROM socket – IC71 usually contains the combined operating system/BASIC ROM and should be left in place. Remove the VideoNuLA support ROM from its protective foam and inspect the ROM's pins and ensure that none are bent or out of alignment. Ensure the ROM is orientated correctly by identifying the notch and making sure that it is pointing toward the back of the PCB.

Gently place the ROM in the chosen socket, checking that the ROM's pins are correctly aligned with the socket receptacles. Slowly apply even downward pressure until the ROM is securely in place.

You can now refit the lid.

ROM installation in a BBC Master

The paged ROM sockets are located at the right of the PCB and are marked as IC24, IC27, IC37 and IC41. You will need to move the plastic speaker/cartridge port support to access all of the sockets.

Identify a free ROM socket – IC24 usually contains the 1Mb operating system/application ROM and should be left in place. If free, IC27 is the most suitable socket as fitting a ROM into either IC37 or IC41 causes the loss of two 16KB sideways RAM banks.

You must ensure that link LK19 is set to the East position if you decide to fit the ROM into IC37. Similarly, you must ensure that link LK18 is set to the East position if you decide to fit the ROM into IC41.

Remove the VideoNuLA support ROM from its protective foam and inspect the ROM's pins and ensure that none are bent or out of alignment. Ensure the ROM is orientated correctly by identifying the notch and making sure that it is pointing toward the left of the PCB.

Gently place the ROM in the chosen socket, checking that the ROM's pins are correctly aligned with the socket receptacles. Slowly apply even downward pressure until the ROM is securely in place.

You can now refit the lid.

ROM installation in a BBC Master Compact

The paged ROM sockets are located at the right of the PCB and are marked as IC17, IC23, IC29, IC38 and IC49.

Identify a free ROM socket – IC49 usually contains the system ROM and should be left in place. You must ensure that PL11 is set to the South position if you decide to fit the ROM into IC38.

Remove the VideoNuLA support ROM from its protective foam. Inspect the ROM's pins and ensure that none are bent or out of alignment. Ensure the ROM is orientated correctly by identifying the notch and making sure that it is pointing toward the left of the PCB.

Gently place the ROM in the chosen socket, checking that the ROM's pins are correctly aligned with the socket receptacles. Slowly apply even downward pressure until the ROM is securely in place.

You can now refit the lid.

3 Using VideoNuLA

Overview

On power-up, VideoNuLA behaves exactly like the original Video ULA IC and so should be fully compatible with all existing software. However, VideoNuLA expands on the capabilities of the original IC in four areas:

- The original palette of 8 solid and 8 flashing colours is expanded to an analogue palette of 4096 colours (16 levels of red, green and blue). This can be used in any mode (including the teletext mode, mode 7).
- The BBC's 16-colour mode (mode 2) can now also be configured to display up to 16 different solid colours. Colours can be chosen from the 4096-colour palette.
- Hardware support for single pixel horizontal scrolling is available in modes 0-6. Previously, depending on the screen mode used, the BBC's hardware scrolling was limited to either 2, 4 or 8 pixels horizontally. VideoNuLA also supports a configurable blanking of the left hand side of the screen to hide scrolling artefacts.
- The provision of new attribute-based graphic and text modes allowing more simultaneous colours on screen in medium and high-resolution.

The following four sections explain how to make use of these features and cover:

- The interactive palette utility
- The support ROM commands
- The VDU driver extensions
- Technical details

The sections get increasingly technical with the later sections aimed at those who wish to make use of VideoNuLA's features in their own programs. However, even if you don't wish to write your own programs, you can still enjoy revamping existing software.

4 The Palette utility

The Palette utility allows each of the 16 physical colours to be redefined to any of the 4096 colours in VideoNuLA's analogue palette. To run it, type:

`*VNPALETTE` **RETURN**

Once loaded, the help page will be displayed. The main utility is entered by pressing the spacebar. You can return to the help page by pressing the 'H' key.

The left and right cursor keys ('←' and '→') select which physical colour is being redefined. The number of the current colour is shown along with the values of its red, green and blue components. Pressing the 'R', 'G' or 'B' keys will increase the red, green or blue component respectively. Holding down the 'SHIFT' key while pressing 'R', 'G' or 'B' will decrease the value of the respective component.

By default, each of colours 8 to 15 is set to flash between its own colour and that of its complement/partner. Pressing the 'F' key toggles this behaviour and will switch any of colours 8 to 15 between flashing and solid.

When one of these colours is set to flash, modifying its red, green and blue components will only affect the first (or primary) flashing colour. To modify the second flashing colour, you have to alter the partner's colour. For example, to alter both flashing colours for colour 9, it is necessary to modify colour 9 and colour 14.

The palette can be reset to its default values at any time by pressing the 'D' key.

Once you are happy with your selections, you can exit the program by pressing the 'Q' key. You can now load any existing software and it will make use of the redefined palette.

Alternatively, you can save the palette by pressing the 'S' key and entering a filename. Palettes can be subsequently re-loaded by pressing the 'L' key. You can change filing system or directory at any time by pressing the '*' button and entering the appropriate command.

The palette is retained until the machine is powered off or VideoNuLA is reset.

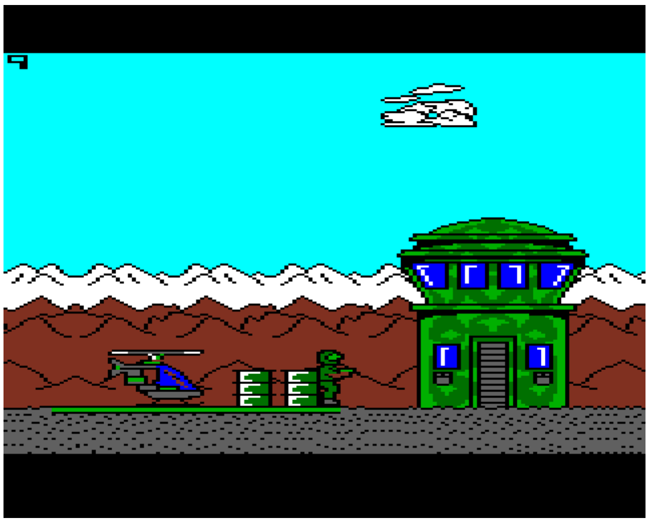
Note: If you modify colour 0 and subsequently use mode 7/teletext mode, all other colours will be affected (see the Technical Details section for more information on this).

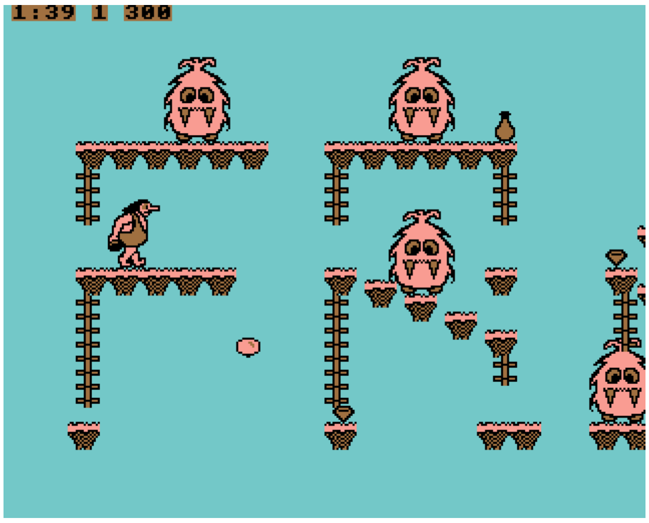
Example palette files

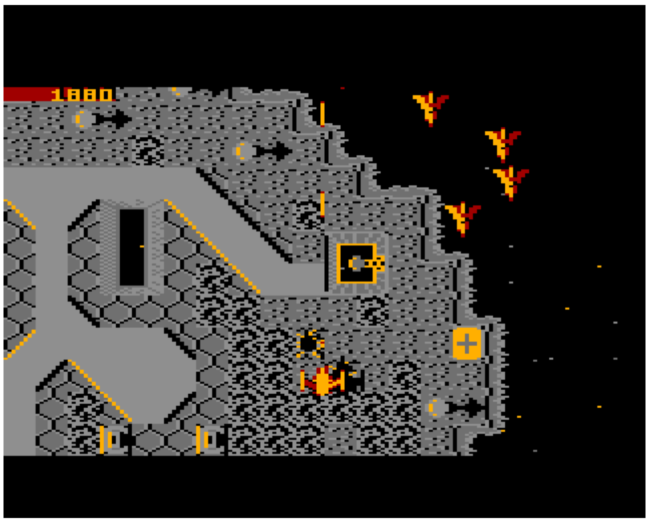
A number of example palette files for use with some popular games are included within the ROM. Details are included in an information file which can be viewed by typing:

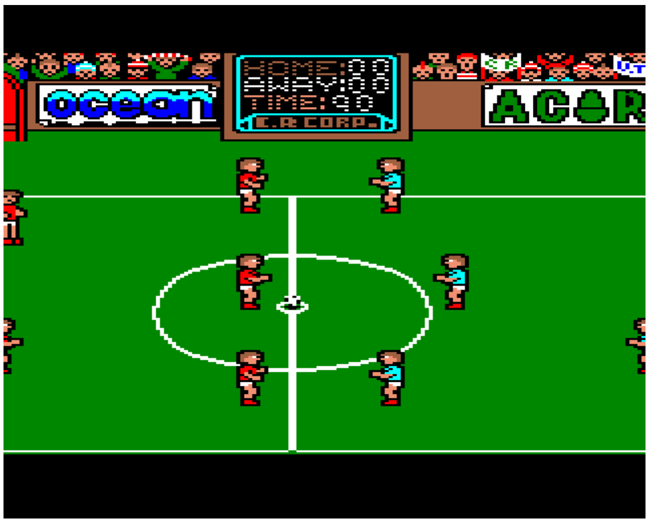
```
*ROM RETURN
*TYPE PALINFO RETURN
```

The screenshots below show some of these palettes in use:









5 The support ROM commands

The support ROM provides utilities, commands and functionality designed to help you enjoy the advanced features of VideoNuLA. Much of this functionality can be accessed via the following *commands:

```
*VNDISABLE
*VNGREY
*VNPALETTE
*VNRESET
*VNVDU
```

The function of each of these commands is described on the following pages along with example syntax and references to further information. You can list the available commands by typing:

```
*HELP VIDEONULA RETURN
```

*HELP VIDEONULA

Purpose

Displays the VideoNuLA support ROM version number and its commands.

Example

```
*HELP VIDEONULA RETURN


VideoNuLA 1.02
  VNDISABLE
  VNGREY
  VNPALETTE
  VNRESET
  VNVDU ON/OFF
```

*VNDISABLE

Purpose

The original Video ULA IC could be accessed at a range of addresses. Therefore, it is possible (though unlikely), that some existing software might inadvertently access VideoNuLA's additional control and palette registers and produce unexpected behaviour.

This command disables VideoNuLA's extra features and should resolve any compatibility issues. Once issued, VideoNuLA cannot be re-enabled in software and so this command remains in effect until the next power-on reset.

Example

```
*VNDISABLE RETURN
```

See also

***VNRESET, control code 5**

*VNGREY

Purpose

To produce a greyscale test card that can be used to calibrate the display. When run, 16 coloured bars ranging from pure black to pure white are displayed. The monitor's contrast and brightness controls should be adjusted to give the best possible display.

.

Example

```
*VNGREY RETURN
```

See also

***VNPALETTE**

*VNPALETTE

Purpose

To interactively redefine VideoNuLA's analogue RGB palette. Each of the BBC Micro's 16 physical colours can be mapped to one of VideoNuLA's 4096 RGB colours. Once defined, the palette is retained until VideoNuLA is reset or the next power-on reset.

Full details on how to use this utility are provided in the Palette utility section.

Example

```
*VNPALETTE RETURN
```

Or:

```
*ROM RETURN
PAGE=&1900 RETURN
CHAIN"PALETTE" RETURN
```

See also

VDU 19, *VNRESET

*VNRESET

Purpose

To revert VideoNuLA's additional features to their default settings. Following this command, the analogue palette reverts to its original state, the horizontal offset and blanking are set to zero and attribute modes are turned off.

This command can be used to put VideoNuLA into a known state before programming.

Example

```
*VNRESET RETURN
```

See also

***VNDISABLE, Control code 4**

*VNVDU ON/OFF

Purpose

The VideoNuLA support ROM augments the operating system's VDU drivers to extend the VDU 19 command and provide additional screen modes. The extended drivers are turned off by default to ensure maximum compatibility with existing software. Once turned on, they remain operational until turned off (including following **BREAK**) or the next power-on reset.

Examples

```
*VNVDU ON RETURN
```

This enables the extended VDU drivers.

```
*VNVDU OFF RETURN
```

This disables the extended VDU drivers.

See also

COLOUR, MODE, VDU 19

6 The VDU driver extensions

Overview

The BBC machine operating system (MOS) contains an extensive VDU driver that provides a range of functions to control output devices (including the display). These functions are available to any software running on the machine and are accessed by sending the appropriate VDU code. (For a complete list of VDU codes, see Appendix L of the BBC Micro User Guide.)

The VideoNuLA support ROM extends the VDU driver to provide convenient interfaces to VideoNuLA's additional functionality. These extensions cover three areas:

- Colour (VDU 17, VDU 18)
- Palette control (VDU 19, VDU 20)
- Screen modes (VDU 22)

These extensions are covered over the following pages along with examples of the corresponding BASIC commands.

VDU 17, COLOUR

Purpose

To set the foreground or background logical text colour. Values below 128 set the foreground colour, values of 128 or above set the background colour.

Certain logical colour combinations are not permitted within the new attribute-based modes due to the way that they hold colour information. The permitted logical colour combinations for each new mode can be found in the Technical details section. As a rule of thumb, the background colour should be set before setting the desired foreground colour.

The combinations of colours displayed on screen can be modified by changing the palette with the VDU 19 command.

For the new attribute-based modes, the default colours are as follows:

Mode numbers	Available colours	Default colours
96, 98, 99, 100	4 foreground, 1 background	0=black, 1=red, 2=green, 3=yellow, 4=white
97	12 foreground, 1 background	0=black, 1=red, 2=green, 3=yellow, 4=blue, 5=magenta, 6=cyan, 7=white, 8=flashing black/white, 9=flashing red/cyan, 10=flashing green/magenta, 11=flashing yellow/blue 12=flashing blue/yellow
101, 102, 103, 104	8 foreground, 1 background	0=black, 1=red, 2=green, 3=yellow, 4=blue, 5=magenta, 6=cyan, 7=white, 8=flashing black/white

Example

```
COLOUR 129 RETURN
COLOUR 0 RETURN
```

Sets the background text colour to logical colour 1 (default is red) and the foreground text colour to logical colour 0 (default is black).

See also

VDU 18, VDU 19, VDU 22

VDU 18, GCOL

Purpose

To set the foreground or background graphics colour and plot mode.

The VDU driver extensions will use the parameters of any VDU 18 (or GCOL command) to set the graphics colour or plot mode in the supported attribute modes (96, 97 and 99).

However, use of the various operating system plotting routines will produce unexpected results as they are unaware of the properties of attribute modes. For example, drawing a horizontal line is likely to modify attribute bits and therefore change the colour of the line.

Examples

```
GCOL 0,1 RETURN
```

Sets plot mode to 1 and foreground graphics colour to logical colour 1.

See also

VDU 17, VDU 22

VDU 19

Purpose

Controls the standard and extended palette. The ROM extensions support three forms of this command:

VDU 19, L, P, 0, 0, 0 maps logical colour L (0-15) to physical colour P (0-15). This is also supported in attribute modes.

VDU 19, 0, P, r, g, b maps physical colour P (0-15) to analogue colour r,g,b (0 < r, g, b < 255).

VDU 19, L, 16, r, g, b maps logical colour L (0-15) to analogue colour r,g,b (0 < r, g, b < 255).

Note: Only the top 4-bits of the red, green and blue components are significant when specifying analogue colours.

Examples

```
VDU 19, 0, 4, 0, 0, 0 RETURN
```

Maps logical colour 0 to physical colour 4.

```
VDU 19, 0, 4, 128, 128, 128 RETURN
```

Maps physical colour 4 to analogue colour 128, 128, 128 (mid-level grey).

```
VDU 19, 3, 16, 128, 48, 32 RETURN
```

Maps logical colour 3 to analogue colour 128, 48, 32 (brown).

See also

VDU 20

VDU 20

Purpose

To restore the standard palette to its default state after it has been modified with VDU 19.

Note: This command has no effect on VideoNuLA's extended palette. To reset the extended palette, see the *VNRESET command.

For the new attribute-based modes, the default colours are as follows:

Mode numbers	Available colours	Default colours
96, 98, 99, 100	4 foreground, 1 background	0=black, 1=red, 2=green, 3=yellow, 4=white
97	12 foreground, 1 background	0=black, 1=red, 2=green, 3=yellow, 4=blue, 5=magenta, 6=cyan, 7=white, 8=flashing black/white, 9=flashing red/cyan, 10=flashing green/magenta, 11=flashing yellow/blue 12=flashing blue/yellow
101, 102, 103, 104	8 foreground, 1 background	0=black, 1=red, 2=green, 3=yellow, 4=blue, 5=magenta, 6=cyan, 7=white, 8=flashing black/white

Example

```
VDU 19, 0, 4, 0, 0, 0 RETURN
```

Maps logical colour 0 to physical colour 4 (default is blue).

```
VDU 20 RETURN
```

Resets the standard palette so logical colour 0 is mapped to physical colour 0 (default is black).

See also

VDU 19

VDU 22, MODE

Purpose

Changes the screen mode. The ROM extensions support a range of new attribute-based modes that increase the number of simultaneous colours on screen at the expense of pixel and colour resolution. The new modes are numbered 96 to 104 with shadow RAM equivalents numbered 224 to 232 (where available).

Each of the new screen modes is based on one of the standard BBC screen modes (modes 0, 1, 3, 4 and 6) but the pixels are one third wider than in the standard modes. This saves two bits in every 8-bit byte. These bits hold a colour attribute this dictates the set of colours used in the adjacent 3 or 6 pixel block. The new modes use a narrow font to retain 40 or 80 characters per line.

Modes 101 to 104 make use of an extra attribute bit (giving three attribute bits in every 8-bit byte) by forcing the last bit displayed (every sixth pixel) to be set to zero/background. This increases the number of colours available in these modes and creates some very attractive text-only modes.

The complete list of new modes is:

Mode	Text resolution	Graphics resolutuion	Colours	Memory usage
96	80 x 32	480 x 256	4 foreground, 1 background	20KB
97	40 x 32	240 x 256	12 foreground, 1 background	20KB
98	80 x 25	Text-only	4 foreground, 1 background	16KB
99	40 x 32	240 x 256	4 foreground, 1 background	10KB
100	40 x 25	Text-only	4 foreground, 1 background	8KB
101	80 x 32	Text-only	8 foreground, 1 background	20KB
102	80 x 25	Text-only	8 foreground, 1 background	16KB
103	40 x 32	Text-only	8 foreground, 1 background	10KB
104	40 x 25	Text-only	8 foreground, 1 background	8KB

Note: Some features will be unavailable or are likely to produce unexpected results in these modes as they have a different structure to the standard modes.

Example

```
MODE 101 RETURN
```

Sets the screen mode to an 80x32 character, 8 foreground/1 background colour, text-only attribute mode.

7 Technical details

Overview

This section is designed for those who wish to gain an in-depth understanding of VideoNuLA. A degree of familiarity with the machine and hexadecimal numbers is assumed.

The original registers (0xFE20, 0xFE21)

The original video ULA IC is controlled by two registers located at 0xFE20 (control) and 0xFE21 (palette) in the I/O processor's memory. VideoNuLA fully implements these original registers to mimic the behaviour of the original video ULA IC. Details on how these registers function can be found in Chapter 19 of the Advanced User Guide for the BBC Microcomputer.

VideoNuLA's additional features are controlled by two more registers which are located at 0xFE22 and 0xFE23. The register at 0xFE22 is an auxiliary control register which governs the additional functions of the VideoNuLA. The register at 0xFE23 controls the mapping of colours to the new 4096-colour palette. As with the original video ULA IC, VideoNuLA's registers are write-only and so cannot be read by the CPU.

[The original video ULA IC is usually accessed at 0xFE20 and 0xFE21 but, as the addresses are not fully decoded, it can also be accessed at 0xFE22 and 0xFE23. VideoNuLA has an option to mirror this behaviour (disabling its additional features) until the next power-on reset. This is described in the section on the auxiliary control register below.]

Accessing the registers from code running on a co-processor

As the four registers are located in the I/O processor's memory they cannot be poked directly from a program running on a co-processor. For example, you cannot write to the auxiliary control register using BASIC's '?' operator from code running on a co-pro.

However, the OSBYTE 151 (&97) call performs a write to any address in the range &FE00-&FEFF on the I/O processor (SHEILA). This call can be conveniently accessed via the *FX 151 command. The command has the following syntax:

```
*FX 151, register, value
```

For example:

```
?&FE22 = &40 RETURN
```

Can simply be replaced with:

```
*FX 151, 34, 64 RETURN
```

The auxiliary palette register (0xFE23)

VideoNuLA provides a 4096-colour/12-bit palette giving 16 levels for each of the red, green and blue components. Each colour in the palette is set by two writes to 0xFE23: the first write specifies which colour to change plus the 4-bit red component, the second write specifies the 4-bit green component and the 4-bit blue component. **Changes to the palette do not take effect until the second write has completed.**

First write:	Colour index	Red component
Second write:	Green component	Blue component

The colour index is a 4-bit value from 0 to 15 (0 to F in hexadecimal). Similarly, the red, green and blue components are also 4-bit values.

You can try changing the palette by starting the machine and typing the following at the BASIC prompt:

`?&FE23 = &78` RETURN (or `*FX 151,35,120` RETURN if using a co-pro)

`?&FE23 = &88` RETURN (or `*FX 151,35,136` RETURN if using a co-pro)

This will change colour 7 (white) to a mid-level grey (red, green and blue components all set to 8). The following will change colour 7 to brown:

`?&FE23 = &7A` RETURN (or `*FX 151,35,122` RETURN if using a co-pro)

`?&FE23 = &42` RETURN (or `*FX 151,35,66` RETURN if using a co-pro)

Each of the BBC Micro's 16 physical colours can be mapped to VideoNuLA's auxiliary 4096-colour palette in this way. However, due to the way that the teletext mode/mode 7 works, changing colour 0 (black) will affect the rest of the colours and leave colour 0 as black. This only applies in teletext mode – colour 0 can be redefined in other modes.

When working in assembly language, it is possible to update VideoNuLA's auxiliary palette many times per scanline. This technique can be used to simultaneously display thousands of colours on screen.

Changes to the auxiliary palette are retained until power-on reset or a reset control code is received (described below).

The auxiliary control register (0xFE22)

In addition to the extended palette, VideoNuLA has a number of other features that are accessed by writing to the auxiliary control register. Each function is accessed by specifying the relevant 4-bit control code and, in some cases, a parameter. This is shown below:

Control code	Parameter (or zero)

Control codes summary

The table below summarises the functions of the various control codes:

Code	Function
1	Set palette mode
2	Set horizontal scroll offset
3	Set left blanking size
4	Reset extended features to defaults
5	Disable extended features
6	Enable/disable attribute modes
7	Enable/disable extended attribute modes
8	Set flashing flags for logical colours 8-11
9	Set flashing flags for logical colours 12-15

The use of each control code is described in detail over the following pages.

Control codes 0 and 10-15 are reserved for future use.

Control code 1: Set palette mode (1-bit parameter)

By default, VideoNuLA will mimic the Video ULA IC and map logical colours to the 16 physical colours using the original palette (accessed at 0xFE21). It then translates the physical colours into 12-bit RGB colours using the auxiliary palette (accessed at 0xFE23). In this mode, it is straightforward to change the colours used by existing software. If you wished to change the cyan sky in your favourite game to a shade of light grey, knowing that cyan is physical colour 6, you would simply type the following before running the game as normal:

`?&FE23 = &69` RETURN (or `*FX 151,35,105` RETURN if using a co-pro)

`?&FE23 = &99` RETURN (or `*FX 151,35,153` RETURN if using a co-pro)

This two-phased palette mapping works well with old/existing software and is fully compatible with the operating system (e.g. VDU 19). However, unlike RISC OS on later Acorn machines, it does not allow logical colours to be mapped *directly* to 12-bit RGB colours.

VideoNuLA can also implement logical colour mapping allowing logical colours to be mapped directly to 12-bit RGB colours using the auxiliary palette register at 0xFE23. This functionality is primarily designed for those who wish to develop new VDU drivers that mimic the behaviour of RISC OS.

To switch from physical-mode to logical-mode mapping, type:

`?&FE22 = &11` RETURN (or `*FX 151,34,17` RETURN if using a co-pro)

This will cause the original palette (accessed at 0xFE21) which maps logical colours to physical colours to be ignored (so commands like VDU 19, l, p, 0, 0, 0 have no effect). VideoNuLA can be switched back to the default physical-mapping mode by performing a power-on reset, sending reset control code 4 or by typing the command:

`?&FE22 = &10` RETURN (or `*FX 151,34,16` RETURN if using a co-pro)

Control code 2: Set horizontal scroll offset (3-bit parameter)

Using registers 12 and 13 of the 6845 CRTC chip, the BBC Micro can scroll the screen horizontally in single byte increments. This equates to: 8 pixels in modes 0, 3, 4 and 6; 4 pixels in modes 1 and 5 and 2 pixels in mode 2. (See chapter 18 of the BBC Micro Advanced User Guide for more information.)

VideoNuLA has an adjustable horizontal scroll offset that operates in single *bit* increments. This allows hardware horizontal scrolling in single pixels in modes 0, 3, 4 and 6; 1/2 pixels in modes 1 and 5 and 1/4 pixels in mode 2. This functionality does not apply in the teletext mode (mode 7) where the offset is ignored.

The offset works by delaying the screen in the increments described above (incrementing the offset moves the screen to the right, decrementing it moves the screen to the left). It takes values of 0 to 7 and can be used in conjunction with the 6845 registers to generate a smoothly scrolling screen.

As an example, the following BASIC commands will move the screen five pixels to the right:

`MODE 4` RETURN

`?&FE22 = &25` RETURN (or `*FX 151,34,37` RETURN if using a co-pro)

The offset can be returned to zero by performing a power-on reset, issuing a reset control code or by entering:

`?&FE22 = &20` RETURN (or `*FX 151,34,32` RETURN if using a co-pro)

Control code 3: Set left-hand side blanking size (4-bit parameter)

The left-hand side blanking can be used in conjunction with the horizontal scroll offset (see above) to give the impression that the screen is scrolling to the right. It can also be used to hide artefacts (e.g. tearing) so that scrolling appears to be completely smooth.

The parameter takes values of 0 to 15 and operates in byte sized units. This equates to units of 8 pixels in modes 0, 3, 4 and 6; 4 pixels in modes 1 and 5 and 2 pixels in mode 2.

As with the horizontal scroll offset, this functionality does not apply in the teletext mode (mode 7) where the blanking size is ignored.

The following table gives the allowed blanking sizes and scrolling units in the various modes:

Mode	Bits per pixel	Scrolling units	Blanking sizes
0, 3, 4, 6	1	1 pixel	0, 8, 16, …, 120 pixels
1, 5	2	½ pixel	0, 4, 8, …, 60 pixels
2	4	¼ pixel	0, 2, 4, …, 30 pixels
7	N/A	N/A	N/A

In addition to facilitating horizontal scrolling, the blanking could also be used to reduce the need for sprite clipping as, even though the visible screen decreases in size, its memory layout does not change.

As an example, the following blanks the first 8 pixels (one character in mode 4):

`MODE 4` **RETURN**

`?&FE22 = &31` **RETURN** (or `*FX 151,34,49` **RETURN** if using a co-pro)

To reset it, type:

`?&FE22 = &30` **RETURN** (or `*FX 151,34,48` **RETURN** if using a co-pro)

Control code 4: Reset VideoNuLA's additional features (no parameter)

By writing zero (or any value from 0-15) to 0xFE22, all of VideoNuLA's additional features can be reset to their default states. This means that the extended palette reverts to the usual BBC Micro colours, horizontal scrolling and blanking are set to zero and the attribute modes are turned off. However, the contents of the original ULA registers at 0xFE20 and 0xFE21 are retained.

This is useful if you wish to revert to normal BBC Micro behaviour or wish to place VideoNuLA in a known state. From BASIC, this can be done by typing:

```
?&FE22 = &40 RETURN
```

Or, if using a co-processor:

```
*FX 151,34,64 RETURN
```

Control code 5: Disable A1 address line (no parameter)

After receiving control code 3, writes to 0xFE22 and 0xFE23 are instead mapped to 0xFE20 and 0xFE21 respectively. This effectively disables VideoNuLA's additional features until the next power-on reset as, once activated, there is no way to turn off this behaviour in software.

Therefore, in the unlikely event that you encounter any compatibility issues, typing the following from the BASIC prompt should resolve them:

```
?&FE22 = &50 RETURN
```

Or, if using a co-processor:

```
*FX 151,34,80 RETURN
```

Control code 6: Enable/disable attribute modes (2-bit parameter)

Alongside the standard screen modes, VideoNuLA is able to generate a number of new modes that provide additional colours by making modest compromises to colour and pixel resolution. These are called “attribute modes” as blocks of pixels share common attributes (in this case, colour). Other 8-bit microcomputers, such as the Sinclair Spectrum and Oric, used similar arrangements for their graphics modes.

There are three different types of attribute modes:

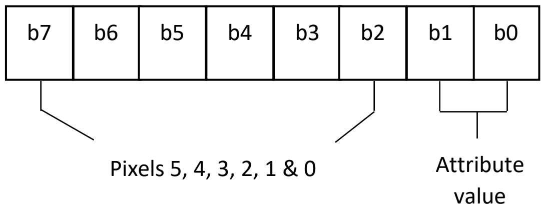
1) 6/12MHz modes that give extra colours at the expense of horizontal resolution
2) “Spectrum” mode that mimics the Sinclair ZX Spectrum’s display
3) “Thomson” mode that mimics the Thomson MO6’s display

Type 1 Attribute modes

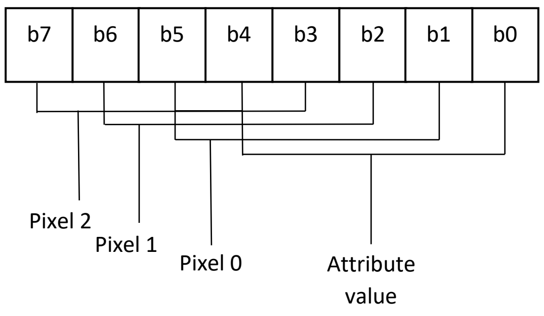
Unlike the standard graphic modes which operate at pixel data frequencies of 8 or 16MHz, the first set of new attribute modes have lower frequencies of 6 or 12MHz. As the computer’s memory is still running at its usual speed, the extra data supplied holds the attribute information.

The bottom two bits in every byte hold the attribute value for that byte, while the remaining six bits hold pixel data. This reduces the horizontal resolution by 25% (from 640 to 480 pixels or from 320 to 240 pixels) but increases the number of simultaneous colours available on screen.

In the attribute mode equivalents of modes 0, 3, 4 and 6, each byte of screen memory is interpreted as follows:



In the attribute mode equivalents of mode 1, each byte of screen memory is interpreted as follows:



The following table describes how the attribute value can be used in the various modes:

Attribute value	Logical colours displayed in modes 0, 3, 4 and 6	Logical colours displayed in mode 1
0	0, 1	0, 1, 2, 3
1	4, 5	4, 5, 6, 7
2	8, 9	8, 9, 10, 11
3	12, 13	12, 13, 14, 15

Attribute modes are turned off by default. To enable them from BASIC, type:

`?&FE22 = &61` RETURN (or `*FX 151,34,97` RETURN if using a co-pro)

They can be turned off by performing a power-on reset, issuing control code 4 or by typing the following command:

`?&FE22 = &60` RETURN (or `*FX 151,34,96` RETURN if using a co-pro)

The original palette at 0xFE21 should be programmed as for the BBC Micro's mode 2 for all attribute modes (see appendix D for details).

Obviously, the BBC Micro's operating system is unaware of the presence of attribute modes and so they require supporting software before they can be fully utilised. The VideoNuLA support ROM provides a number of new attribute-based screen modes that take advantage of these features – see the VDU driver extensions section for more details.

Type 2 Attribute mode (Sinclair ZX Spectrum mode)

This mode is designed to allow the BBC to reproduce the display capabilities of the ZX Spectrum. The mode is turned off by default. To enable it from BASIC, type:

`?&FE22 = &62` RETURN (or `*FX 151,34,98` RETURN if using a co-pro)

It can be turned off by performing a power-on reset, issuing control code 4 or by typing the command:

`?&FE22 = &60` RETURN (or `*FX 151,34,96` RETURN if using a co-pro)

The Spectrum uses one byte of data to define colour, brightness and flashing attributes for each 8x8 block of pixels in its display. Each attribute data byte is encoded as shown below:

b7	b6	b5	b4	b3	b2	b1	b0
Flash	Bright	Paper2	Paper1	Paper0	Ink2	Ink1	Ink0

The Spectrum uses "paper" to represent the background colour and "ink" to represent the foreground colour. If set, the flash attribute bit swaps the paper and ink colours (rather than inverting them as on the BBC). The bright attribute bit selects a brighter set of colours but there are only 15 to choose from so black is always "dark" black.

Colours are arranged differently to the BBC as bit 0 controls the blue gun, bit 1 controls the red gun and bit 2 controls the green gun.

VideoNuLA mimics this behaviour but each attribute byte only affects an 8x1 block of pixel data (this is equivalent to the HiColor mode on the Timex machines). In addition, each attribute byte must immediately precede the pixels it acts on in the BBC's memory.

VideoNuLA uses the BBC's mode 0 to mimic the Spectrum's display but alternate horizontal bytes represent attribute and pixel data. Pixel widths are then doubled to give an effective resolution of 320x256 which accommodates the Spectrum's resolution of 256x192 pixels and allows space for its coloured border.

The Spectrum mode uses logical palette mapping so that the BBC's original palette is ignored. The attribute colours are mapped to the equivalent logical colours on the BBC so bit 0 (blue) is mapped to bit 2, bit 1 (red) is mapped to bit 0 and bit 2 (green) is mapped to bit 1.

The Spectrum's dark colours are represented by logical colours 1-8 and the bright colours are represented by logical colours 9-15. VideoNuLA maps logical colour 8 to both "dark" and "light" black. To allow logical colour 0 to represent the border, it is only displayed following the special attribute value of 0x80 (which would ordinarily represent flashing dark black on dark black).

The Spectrum's flashing behaviour is accessed by setting the relevant attribute bits and setting the flash bit (bit 0) in the BBC's VideoULA control register at 0xFE20. By using OSBYTE/*FX 9 and 10, the colours can be made to flash every 16 frames as on the Spectrum.

An example of how to use this mode is given in appendix E. **For those with a Raspberry Pi co-processor, a disk image containing a ZX Spectrum emulator ("SpectROM") is included in the VideoNuLA support pack. This allows ZX Spectrum programs and games to run on the BBC.**

Type 3 Attribute mode (Thomson MO6 mode)

This mode is designed to allow the BBC to reproduce the display capabilities of the Thomson MO6. The mode is turned off by default. To enable it from BASIC, type:

`?&FE22 = &63` RETURN (or `*FX 151,34,99` RETURN if using a co-pro)

It can be turned off by performing a power-on reset, issuing control code 4 or by typing the command:

`?&FE22 = &60` RETURN (or `*FX 151,34,96` RETURN if using a co-pro)

This attribute mode is similar to that of the ZX Spectrum but it uses one byte of data to define foreground and background colour attributes for each 8x1 block of pixels in its display. Each attribute data byte is encoded as shown below:

b7	b6	b5	b4	b3	b2	b1	b0
Back3	Fore3	Back2	Back1	Back0	Fore2	Fore1	Fore0

This mode allows 16 colours to be chosen for foreground and background from the 4096 colour palette at a resolution of 320x256. As with the ZX Spectrum mode, logical palette mode is enforced so the BBC's original palette is ignored. Colours are also mapped in the same way as the ZX Spectrum mode.

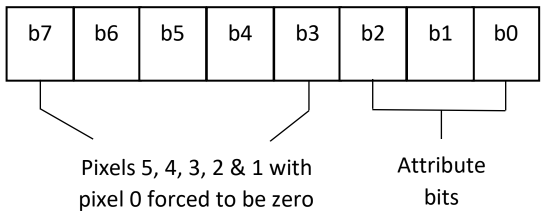
As an additional "feature", bit 7 doubles up as a flashing bit. If bit 7 is set and the flashing bit (bit 0) of the BBC's original ULA control register at 0xFE20 is set to 1, the foreground and background colours are swapped. Flashing can be turned off by using OSBYTE/*FX 9 & 10.

Control code 7: Enable/disable 8-colour attribute text modes (1-bit parameter)

As described above, the Type 1 attribute mode equivalents of modes 0, 3, 4 and 6 give four sets of two colours at horizontal resolutions of 480 or 240 pixels by using the bottom two bits in every byte as a palette index/colour attribute. However, in a text-only display, there is usually a single bit space between characters and these “spacer” bits are effectively wasted.

Control code 7 switches between the Type 1 attribute modes (as described above) and special text-only 3-bit attribute modes. These use the bottom three bits in every byte as a palette index so that eight sets of two colours can be shown on screen.

In these modes, bits 7 to 3 are used as pixel data. The rightmost pixel is always assumed to be zero/blank (the space between the characters). Each byte of screen memory is interpreted as follows:



The attribute values work with the pixel data to produce colours as follows:

Attribute value	Logical colours displayed
0	0, 1
1	2, 3
2	4, 5
3	6, 7
4	8, 9
5	10, 11
6	12, 13
7	14, 15

These modes are turned off by default. To enable them from BASIC, first enable type 1 attribute modes (with control code 6 as described above) and then type:

`?&FE22 = &71` RETURN (or `*FX 151,34,113` RETURN if using a co-pro)

They can be turned off by performing a power-on reset, issuing control code 4 or by typing the command:

`?&FE22 = &70` RETURN (or `*FX 151,34,112` RETURN if using a co-pro)

The original palette at 0xFE21 should be programmed as it would be for the BBC Micro’s mode 2 (see appendix D for details). By programming all of the even logical colours to one physical colour and each of the odd logical colours to a separate physical colour, it is possible to have 8 foreground colours on a single background colour. The physical colours can be chosen from any of the 4096 colours in VideoNuLA's extended palette.

The VideoNuLA support ROM uses this technique with a 5-pixel wide font to produce a variety of colourful text modes with character resolutions of 80x32, 80x25, 40x32 and 80x25. See the VDU driver extensions section for more details.

Control code 8: Set flashing flags for colours 8-11 (4-bit parameter)

This sets (or clears) flags that indicate which of logical colours 8 to 11 should flash. By default, all flags are set to 1 (i.e. flashing) until a colour is redefined via the auxiliary palette register (at 0xFE23) when it becomes solid.

Setting a flag to 1 indicates that it should flash, setting it to 0 indicates that the colour should be solid. The MSB controls colour 8, the LSB controls colour 11. The following BASIC statement sets colour 8 to flashing but colours 9 to 11 to solid:

```
?&FE22 = &88 RETURN
```

Or, if using a co-processor:

```
*FX 151,34,136 RETURN
```

Note: If colour N (8≤N≤15) is redefined and then set to flash, colours N and (N EOR 7) will be displayed alternately.

Control code 9: Set flashing flags for colours 12-15 (4-bit parameter)

In a similar way to control code 8 above, this sets (or clears) flags that indicate which of logical colours 12 to 15 should flash. The MSB controls colour 12, the LSB controls colour 15. The following BASIC statement sets colour 12 to flashing but colours 13 to 15 to solid:

```
?&FE22 = &98 RETURN
```

Or, if using a co-processor:

```
*FX 151,34,152 RETURN
```

Note: If colour N (8≤N≤15) is redefined and then set to flash, colours N and (N EOR 7) will be displayed alternately.

Appendix A – Default physical colours

0	Black
1	Red
2	Green
3	Yellow
4	Blue
5	Magenta
6	Cyan
7	White
8	Flashing black/white
9	Flashing red/cyan
10	Flashing green/magenta
11	Flashing yellow/blue
12	Flashing blue/yellow
13	Flashing magenta/green
14	Flashing cyan/red
15	Flashing white/black

Appendix B – Standard screen modes and Type 1 attribute mode equivalents

Mode	Resolution	Colours	Attribute mode resolution	Attribute mode colours (*)	Attribute mode colours (**)
0	640 x 256	2	480 x 256	4 sets of 2	8 sets of 2
1	320 x 256	4	240 x 256	4 sets of 4	---
2	160 x 256	16	---	---	---
3	640 x 200	2	480 x 200	4 sets of 2	8 sets of 2
4	320 x 256	2	240 x 256	4 sets of 2	8 sets of 2
5	160 x 256	4	---	---	---
6	320 x 200	2	240 x 200	4 sets of 2	8 sets of 2
7	Teletext	8	N/A	N/A	N/A

*** Obtained by sending control code 6**

**** Obtained by sending control codes 6 and 7 – only suitable for text displays**

Note: There are no attribute mode equivalents of modes 2 and 5. Also, there is no 3-bit attribute mode equivalent of mode 1. In each case, the screen will only display the background colour and the cursor.

Appendix C – Extended palette test program

The following BASIC program will setup the VideoNuLA's extended palette to display 16 shades of grey and then print text in each colour. (The first line will be printed in black and so will not be visible.)

Note: This program uses a 20K screen mode and so is not suitable for an unexpanded 16K model A machine. It uses *FX 151 to write to addresses 0xFE22 and 0xFE23 on the I/O processor to ensure that it runs correctly on co-processors.

[This program can also be found on the VideoNuLA disk image as "GRSCALE".]

```
 10 MODE 2
 20 PROCpoke(&22,&40):REM Reset VideoNuLA
 30 FOR I%=0 TO 15
 40 PROCpoke(&23, I%*17):REM Colour index=I%, red=I%
 50 PROCpoke(&23, I%*17):REM green=I%, blue=I%
 60 COLOUR I%
 70 PRINT"THIS IS GREY"
 80 NEXT I%
 90 END
100 DEF PROCpoke(reg%,byte%)
110 OSCLI("FX 151,"+STR$(reg%)+","+STR$(byte%))
120 ENDPROC
```

Appendix D – Programming the original palette for attribute modes

All attribute modes expect the original palette to be programmed in the same manner as mode 2. This means that the palette is programmed once for each logical colour (details can be found on pages 379-382 of the Advanced User Guide for the BBC Microcomputer).

As the operating system does not understand the new attribute modes, the palette has to be modified by writing directly to the register at 0xFE21. The example program below sets up the 480 x 256 pixel mode 0 equivalent with 4 sets of 2 colours. The palette is defined so that all background colours are black and the foreground colours are red, green and yellow and blue. *[This program can also be found on the VideoNuLA disk image as "ATTR1".]*

```
 10 DIM oswblock% 5
 20 MODE 0
 30 PROCwriteIO(&FE22, &40):REM Reset VideoNuLA
 40 FOR I%=0 TO 15 STEP 4
 50 REM Set logical colour I% to physical colour 0
 60 REM Value sent to FE21 is physical colour EOR 7
 70 PROCwriteIO(&FE21, (I%*16) EOR 7)
 80 REM Set logical colour I%+1
 90 PROCwriteIO(&FE21, ((I%+1)*16)+(((I% DIV 4)+1)
EOR 7))
100 NEXT I%
110 REM Set attribute mode
120 PROCwriteIO(&FE22, &61)
130 REM Display some colour
140 PROCwriteIO(&3000,&FF):PROCwriteIO(&3008,&FE):
PROCwriteIO(&3010,&FD):PROCwriteIO(&3018,&FC)
150 PRINT
160 END
170 DEF PROCwriteIO(addr%,val%)
180 A%=6:X%=oswblock%:Y%=oswblock% DIV 256
190 !X%=addr%:X%?4=val%:CALL &FFF1
200 ENDPROC
```

After running the program, you should see four coloured lines at the top left of the screen.

If you now try to list the program or work in this screen mode, the display will look very strange with characters appearing in various colours! You can revert back to the usual mode 0 by typing:

`?&FE22=&40` **RETURN** (or `*FX 151,34,64` **RETURN** if using a co-pro)

`MODE 0` **RETURN**

Programming the original palette for the mode 1 equivalent (4 sets of 4 colours) or for the 3-bit attribute text mode works in the same way. The palette entry for each logical colour is programmed once with the value of the desired physical colour exclusive-OR'd with 7.

The example below shows how to set up the palette for a 480 x 256 pixel text mode with 8 sets of 2 colours. The colours are defined so that all background colours are black and the foreground colours are red, green, yellow, blue, magenta, cyan, grey and white. . *[This program can also be found on the VideoNuLA disk image as "ATTR2".]*

```
 10 MODE 0
 20 PROCpoke(&22,&40):REM Reset VideoNuLA
 30 PROCpoke(&23,&76):PROCpoke(&23,&66):REM phys.col.
7=grey
 40 PROCpoke(&23,&8F):PROCpoke(&23,&FF):REM phys.col.
8=white
 50 FOR I%=0 TO 15 STEP 2
 60 PROCpoke(&21, (I%*16) EOR 7):REM background =
black
 70 PROCpoke(&21, ((I%+1)*16) + (((I% DIV 2)+1) EOR
7))
 80 NEXT I%
 90 PROCpoke(&22,&61):REM Enable attribute modes
100 PROCpoke(&22,&71):REM Enable 3-bit attribute mode
110 END
120 DEF PROCpoke(reg%, byte%)
130 OSCLI("FX 151,"+STR$(reg%)+","+STR$(byte%))
140 ENDPROC
```

You can revert back to the usual mode 0 by typing:

`?&FE22=&40` **RETURN** (or `*FX 151,34,64` **RETURN** if using a co-pro)

`MODE 0` **RETURN**

Appendix E – Programming the ZX Spectrum attribute mode

The following BASIC program shows how to use VideoNuLA's ZX Spectrum attribute mode. It uses VDU 23 to define a screen character that holds the bit pattern for the blue ink attribute. This is then printed before each text character.

Note: This program uses a 20K screen mode and so is not suitable for an unexpanded 16K model A machine. It uses *FX 151 to write to address 0xFE22 on the I/O processor to ensure that it runs correctly on co-processors.

[This program can also be found on the VideoNuLA disk image as "SPECCY".]

```
 10 MODE 0
 20 VDU 23,255,1,1,1,1,1,1,1,1:REM Blue ink attr.
 30 PROCpoke(&22, &62):REM Spectrum mode
 40 PRINTCHR$255;″H″;CHR$255;″e″;CHR$255;″l″;
CHR$255;″l″;CHR$255;″o″
 50 END
 60 DEF PROCpoke(reg%,byte%)
 70 OSCLI("FX 151,"+STR$(reg%)+","+STR$(byte%))
 80 ENDPROC
```

You can revert back to the usual mode 0 by typing:

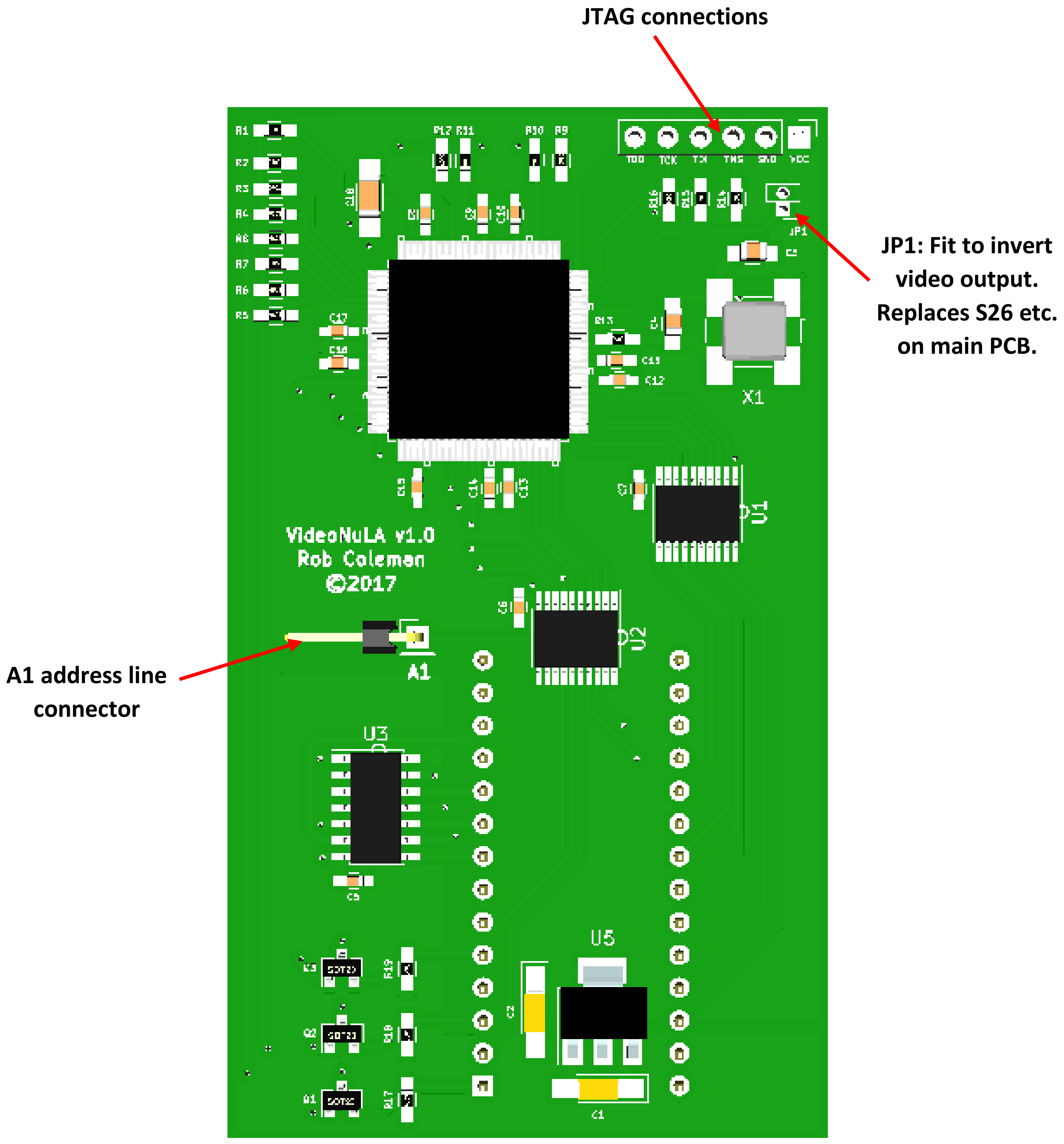
`?&FE22=&40` **RETURN** (or `*FX 151,34,64` **RETURN** if using a co-pro)

`MODE 0` **RETURN**

Appendix F – VideoNuLA support disk contents

!BOOT	- Boot file (loads Palette utility program)
ATTR1	- Attribute mode demonstration program 1
ATTR2	- Attribute mode demonstration program 2
GRSCALE	- Greyscale demonstration program
PALETTE	- Palette utility program
SPECCY	- ZX Spectrum attribute mode demonstration program
VNDIS	- *command equivalent to support ROM's *VNDISABLE
VNRESET	- *command equivalent to support ROMS's *VNRESET
VNULA	- VideoNuLA support ROM image
P.BARBARI	- Palette file for "Barbarian" by Superior Software
P.BOFFIN	- Palette file for "Boffin" by Addictive Games
P.COMMNDO	- Palette file for "Commando" by Elite
P.CRZYRID	- Palette file for "Crazee Rider" by Superior Software
P.FIRETRA	- Palette file for "Firetrack" by Electric Dreams
P.FORTRES	- Palette file for "Fortress" by Pace
P.FRAK	- Palette file for "Frak" by Aardvark
P.GREY	- Greyscale palette file
P.HUNCHBA	- Palette file for "Hunchback" by Ocean
P.LASTNIN	- Palette file for "The Last Ninja" by Superior Software
P.MATCHDY	- Palette file for "Match Day" by Ocean
P.REVS	- Palette file for "Revs" by Acornsoft
P.SNAPPER	- Palette file for "Snapper" by Acornsoft
P.SNOOKER	- Palette file for "Snooker" by Acornsoft
P.STRYKER	- Palette file for "Stryker's Run" by Superior Software
P.YIEARKF	- Palette file for "Yie Ar Kung Fu" by Imagine

Appendix G – v1.0 Board layout


JTAG connections
JP1: Fit to invert video output. Replaces S26 etc. on main PCB.
A1 address line connector
VideoNuLA v1.0
Rob Coleman
©2017
TDO
TCK
TDI
TMS
GND
VCC
JP1
X1
U1
U2
U3
U5
A1