The Serial Port
&
P.C.Arnold Technical Services
Present

The PCATS Graphics Enhancer

Release 1 00

The Serial Port
Burcott Manor, Wells, Somerset BA5 1NH
Tel: 0243 531194 Fax: 0243 531196

The Serial Port Copyright Policy

At The Serial Port we believe that whenever possible our software should NOT be copy protected, thus allowing you to easily make back-up copies of the software or run it from a hard disk. We do sometimes have to make exceptions to this rule but the decision to protect *a* program is not taken without a great deal of thought going into finding alternative methods to protect our copyright.

This means we are placing a great deal of trust, and our future as a software supplier, in your hands. We ask you not to abuse your position - by all means recommend and demonstrate our software to friends, but **please** do not give them a copy. If you do you will be doing harm to everybody, including yourself, as we will no longer be able to supply improved versions of our software at low cost to present users such as yourself or invest in new projects which will help increase the Archimedes user base - you wouldn't want to force us to go and program IBM PC's, would you?

We have every faith in our users and believe that we can safely supply our software in this unprotected form. Help us to help you - Spread the word, not the disk!

PLEASE NOTE: The ColourTrans V0.70 module supplied with this product Is based upon original code licensed from Acorn Computers Ltd. [t has been tested and found to work with a large majority of the commercial software available at the time of printing of this manual, but cannot be absolutely guaranteed to work with all software which will be written for RISC OS. [f and when new versions of RISC OS are released, the version of ColourTrans supplied with them should be used in place of the version supplied on the Graphics Enhancer utilities disk.

The Graphics Enhancer is a complex product which is under continu0us devel0pment and, while every eff0rt is made t0 ensure it functions as detailed, neither The Serial P0rt n0r Acom can accept any liability f0r any l0ss 0r damage resulting from the use The Graphics Enhancer 0r the inf0rmati0n in this manual.

Contents

Introduction

The PCATS Graphics Enhancer, marketed by The Serial Port, is an expansion card for the Acorn Archimedes range of microcomputers. It is designed as a low-cost, high performance upgrade for significantly improving the graphics performance of these machines. Coupled with the high speed and versatile architecture of the Archimedes, it can produce results *as* good as, if not better than, some professional CAD type workstations, at a far lower cost.

The Graphics Enhancer will work with any machine in the Archimedes range, from an A3000 to a 540. On the 310, 400 & 400/1 series, and the 540, it plugs into the podule expansion backplane, inside the machine. On the A3000, it fits externally onto the rear expansion connector, and requires a small adaptor board which fits over the VIDC chip inside the computer case.

The enhancements provided by the board can be grouped into five main areas, as follows:

(1) A 24-bit palette, giving 256 colours on screen simultaneously, selectable from 16,777,216 available colours. This feature allows modes with resolutions of up to 768 x 288 on a standard monitor, and resolutions of 832 x 328, 512 x 512, 640 x 480, & others on a multisync monitor. These modes all work correctly with the Desktop, and the new version of the ColourTrans module supplied will allow any properly written Desktop application to work in them.

(2) 12 bpp direct DAC modes, giving 4096 simultaneous colours, at resolutions of up to 512 x 288 on a standard monitor, and up to 554 x 328 on a multisync monitor. These modes are not Desktop compatible.

(3) 16 bpp direct DAC modes, giving 65536 simultaneous colours, at resolutions of up to 384 x 288 on a standard monitor, and up to 416 x 328 on a multisync monitor. These modes are not Desktop compatible.

(4) A set of up to seven VIDC clocks, software selectable by * commands or SWI calls. The clocks installed as standard are 24 MHz, 25.175 MHz, 32 MHz, & 36 MHz. There are 3 empty sockets on the Graphics Enhancer board, which can have user-supplied oscillators plugged into them. The Serial Port can supply oscillator modules for any frequency required. VLSI specifications give the maximum operating speed of the VIDC as 36 MHz, but we have found that most will work to 40 MHz, and some to as high as 48 MHz. However, we cannot guarantee that any particular VIDC will work above 36 MHz. We can supply oscillators of any frequency required, and we also produce a Mode Booster upgrade board, which is necessary for the Graphics Enhancer itself to run faster than 36 MHz. Contact The Serial Port for prices.

(5) The ability to define new modes as text files, using the Mode Description Language compiler supplied with the Graphics Enhancer. This will allow the user to make new modes at will, utilising all the extra video facilities that are provided by the hardware.

The Graphics Enhancer also adds many other new modes, including all those provided by the Computer Concepts mode module (supplied with Impression), all those provided by the Atomwide VIDC enhancer, and the new Super VGA modes as used on the 540. See Appendix G for a full list of defined modes.

Software support for the various features of the Graphics Enhancer is quite extensive, with many SWI calls and * commands. Several major software houses are now, or will soon be, adding features to their graphics-based programs to take full advantage of the new modes added to the standard Archimedes.

A Professional version of the Graphics Enhancer will soon be available, which will add full genlock capability to the other features supported by the standard version, allowing any of the 256 Desktop colours to have up to 24 bits of overlay keying, independently of the rest. Please contact The Serial Port for pricing and availability.

Why you need a Graphics Enhancer

The Acorn Archimedes computers are probably the fastest, easiest to use, and most programmable microcomputers available in their price range, or anywhere close to it. However, in one key area they lag behind other machines, such as the Apple Macintosh. That area *is* the quality of onscreen pictures. The graphics of the Archimedes are very fast, and much better than many machines can produce at two or three times the price. but have some fairly severe limitations, notably in the number and range of colours that are available.

In 256 colour modes, the Archimedes is quoted as having a palette of 4096 colours, implying that any 256 of these can be used at any one time. In fact, due to the somewhat peculiar way the video hardware is designed, colours can only be chosen in blocks of 16, which almost inevitably gives one colour that is very nearly, but not quite, the one required, and 15 others that are nowhere near it. Even given these limitations, the results that can be obtained are amazing. However, it would be nice to have a true palette, with any of the 256 colours independently seleetable from a larger range than in normally available.

This is precisely what the Graphics Enhancer does.

It allows 256 totally independent colours to be selected from a range of over 16.7 million, thus ensuring that whatever colour is wanted can be obtained. In most applications, 256 colours are sufficient, given that there is a large enough selection to pick from. Even so, there are times when more colours are needed onscreen simultaneously. In these cases, the Graphics Enhancer can also help, as it has 12 and 16 bpp modes, giving up to 65536 colours at once. Unfortunately, the current version of RISC OS cannot handle more than 8 bits per pixel, so these modes can only be used by programs written specifically for them.

The extended palette modes are fully supported by the operating system. Simple BASIC programs can easily take advantage of the extended range of available colours, thus allowing anyone with a modicum of programming knowledge to obtain spectacular results. For non-programmers, the Graphics Enhancer is being supported by several software houses, who are adding support for it to their RISC OS applications.

In addition to adding the facility of more and better colours, the Graphics Enhancer also allows the VIDC to be run at different frequencies than the default 24 MHz, thus allowing higher resolution modes, although only in 16 colours. It is fully compatible with the Atomwide VIDC enhancer, which means that people who have written programs using the various Atomwide modes can continue to use them when the Graphics Enhancer is installed. It also adds all the modes that are supported by the Computer Concepts Impression mode module, and those that are new to the A540. A full list of the modes that are supported as standard by the Graphics Enhancer can be found in Appendix G. In addition, new modes can be defined as text files, which are then compiled using the supplied program !MDLcomp to a 200 byte mode descriptor data block. Up to 36 of these data blocks can be loaded at once, taking up a mere 7400 bytes of RMA.

The Graphics Enhancer is totally transparent in operation, needing no special commands to turn it on or off. Any program that uses its extended graphics facilities will use it, and any program that does not use it will simply ignore it. As a result, there is very little in the way of user instructions needed, except for programming information for those who wish to write programs for the extended palette modes, etc. The manual that follows lists all the new * commands and SWI's that the Graphics Enhancer software provides, along with information on using the 12 & 16 bpp modes, on the CLEAR & TIFF file formats that are used for saving extended palette screens, and technical information on the VIDC chip itself. If the user does not wish to write programs using the new graphics facilities, much of this manual can simply be ignored. The section on configuration commands, at the end of the * command list, should be read by all users, as it can help if there are problems in the operation of the Graphics Enhancer. Such problems are almost inevitably due to one or more of the *Configure options being set wrongly.

Any comments or suggestions for changes to this manual or the Graphics Enhancer software are welcomed, and should be sent to The Serial Port. If you have any special project that requires technical information not given in this manual, please let us know and we will try to help.

* Commands

`*VIDC`	Programs VIDC registers directly.
`*Clock`	Sets or reads the VIDC clock rate.
`*Dac`	Enables 12 or 16 bpp direct DAC mode.
`*ExtPal`	Enables 24-bit extended palette mode.
`*Default`	Restores default Graphics Enhancer settings.
`*NormalVideo`	Restores default Archimedes video output.
`*Pmask`	Programs pixel mask.
`*Palette`	Programs extended palette.
`*Mode`	Sets screen mode.
`*PalSet`	Sets red scale/restores default palette.
`*DeskPal`	Sets Desktop equivalent palette in extended palette modes.
`*PalSave`	Saves the current extended palette.
`*PalLoad`	Loads an extended palette file.
`*ClearSave`	Saves the current extended palette mode screen as a CLEAR file.
`*ClearLoad`	Loads a CLEAR file to the screen.
`*GreyScale`	Sets the extended palette to a linear greyscale.
`*Gcol`	Sets graphics colour in extended palette modes.
`*Colour`	Sets text colour in extended palette modes.
`*LinkMode`	Loads a mode definition file and links it in as a screen mode.
`*LinkModeClear`	Clears all linked modes.
`*DeLinkMode`	Delinks an individual linked mode.

`*Configure CrystalSlots`	Configures occupied VIDC oscillator sockets.
`*Configure DefaultCrystal`	Configures default VIDC clock.
`*Configure ProcessorType`	Configures installed processor type.
`*Configure MonitorGroup`	Configures group of attached monitor.
`*Configure IREThreshold`	Configures blanking level of monitor.
`*Configure TurboModes`	Configures optional mode booster enable/disable.

*VIDC

Programs VIDC registers directly

Syntax *VIDC <register> <value>

Parameters <register> is a valid VIDC register
<value> is a number within the range allowed for the VIDC register

Use *VIDC allows direct programming of VIDC registers from the command line. This can be useful for experimenting with new values for creating new modes, without having to create a mode definition module. However, Risc-OS will not know about the new values, so the VDU drivers will not correctly handle the changes. See Appendix H for more details of register numbers and allowed values.

Warning! Use this command with caution, as programming the VIDC with the wrong values could damage your monitor.

Example *VIDC &A0 312 Sets the VCR to 312 rasters

Related commands None

Related SWIs Enhancer_VIDC (SWI &42A55)

Related vectors None

*Clock

Sets or reads the VIDC clock rate

Syntax *Clock [<n>]

Parameters [<n>] is a number between 0 and 7

Use *Clock connects the oscillator in socket <n> to the VIDC. The default clock frequencies and the socket numbers are as follows:

Socket 0: 24 MHz
1: 25.175 MHz
2: 32 MHz
3: 36 MHz
4: empty (for user-supplied oscillator)
5: empty (for user-supplied oscillator)
6: empty (for user-supplied oscillator)
7: 24 MHz

Only sockets that have been marked as occupied by *Configure CrystalSockets can be selected. *Clock with no parameter will return the current setting.

Example

*Clock 2 Sets the VIDC clock rate to 32 MHz

Related commands

*Configure CrystalSockets

Related SWIs

Enhancer_Clock (SWI &42A4C)

Related vectors

None

*Dac

Enables 12 or 16 bpp direct DAC mode

Syntax *Dac <bpp>

Parameters <bpp> is either 12 or 16

Use *Dac is used to force the Graphics Enhancer to either 12 or 16 bpp direct DAC mode. The Graphics Enhancer software transparently does this when displaying 12 or 16 bpp modes.

Example

*Dac 12	Sets 12 bpp mode

Related commands *ExtPal

Related SWIs Enhancer_Dac (SWI &42A4B)
Enhancer_ExtPalette (SWI &42A4A)

Related vectors None

*ExtPal

Enables 24-bit extended palette mode

Syntax *ExtPal

Parameters None

Use *ExtPal is used to force the Graphics Enhancer to 24-bit extended palette mode. The Graphics Enhancer software transparently does this when in extended palette modes.

Example *ExtPal Enables extended palette

Related commands *Dac

Related SWIs Enhancer_Dac (SWI &42A4B)
Enhancer_ExtPalette (SWI &42A4A)

Related vectors None

*Default

Restores default Graphics Enhancer settings

Syntax *Default

Parameters None

Use *Default is used to reset the Graphics Enhancer to normal Archimedes video output and 24 MHz VIDC clock rate.

Example *Default Reset Graphics Enhancer

Related commands *Normal Video

Related SWIs Enhancer_Default (SWI &42A46)
Enhancer_NormalVideo (SWI &42A47)

Related vectors None

*NormalVideo

Restores default Archimedes video output

Syntax *NormalVideo

Parameters None

Use *NormalVideo is used to reset the Graphics Enhancer to normal Archimedes video output, without affecting the VIDC clock rate.

Example *NormalVideo Restore normal Archimedes video output

Related commands *Default

Related SWIs Enhancer_Default (SWI &42A46)
Enhancer_NormalVideo (SWI &42A47)

Related vectors None

*Pmask

Programs pixel mask

Syntax *Pmask [<n>]

Parameters [<n>] is a number between 0 and 255

Use *Pmask is used to write the pixel mask. *Pmask with no parameter prints the current pixel mask.

Example

*Pmask 255 Set the pixel mask off

Related commands None

Related SWIs

Enhancer_PixelMaskWrite (SWI &42A44)
Enhancer_PixelMaskRead (SWI &42A45)

Related vectors None

*Palette

Programs extended palette

Syntax *Palette <C> <R> <G> <B>

Parameters
<C> is an 8-bit colour number
<R> is an 8-bit red component value
<G> is an 8-bit green component value
<B> is an 8-bit blue component value

Use *Palette is used to program individual extended palette entries. Please note that colour 0 cannot be programmed to other than black, for various hardware-related reasons.

Example

*Palette 100 255 0 0	Sets colour 100 to full red, no green, and no blue

Related commands None

Related SWIs
Enhancer_PaletteBlockWrite (SWI &42A40)
Enhancer_PaletteBlockRead (SWI &42A41)
Enhancer_PaletteWrite (SWI &42A42)
Enhancer_PaletteRead (SWI &42A43)

Related vectors None

*Mode

Sets screen mode

Syntax *Mode [<n>]

Parameters [<n>] is any valid mode number

Use *Mode is an exact equivalent of the BASIC MODE command. *Mode with no parameter prints the current mode.

Example *Mode 12 Sets mode 12

Related commands None

Related SWIs None

Related vectors None

*PalSet

Sets red scale/restores default palette

Syntax *PalSet <0|1>

Parameters <0|1> is 0 for default Archimedes palette, or 1 for red scale

Use *PalSet is used to set a linear red scale in 4 bpp modes, or to restore the default palette. If <n> is 1, the palette is set to a red scale, and if it is 0 the default palette is restored.

Example *PalSet 1 Sets linear red scale

Related commands None

Related SWIs Enhancer_PalSet (SWI &42A49)

Related vectors None

*DeskPal

Sets Desktop equivalent palette in extended palette modes

Syntax *DeskPal

Parameters None

Use *DeskPal is used to set the palette in extended palette modes to an equivalent of the standard desktop one.

Example

*DeskPal	Sets standard desktop palette

Related commands None

Related SWIs Enhancer_SetDesktopPalette (SWI &42A4E)

Related vectors None

*PalSave

Saves the current extended palette

Syntax *PalSave <filename>

Parameters <filename> is any valid filename

Use *PalSave is used to save the current extended palette as a palette file with the filetype of &C8A.

Example *PalSave palfile Saves current palette as "palfile"

Related commands *PalLoad

Related SWIs Enhancer_PaletteSave (SWI &42A52)
Enhancer_PaletteLoad (SW! &42A53)

Related vectors None

*PalLoad

Loads an extended palette file

Syntax *PalLoad <filename>

Parameters <filename> is any valid filename

Use *PalLoad is used to load an extended palette file and make it the current extended palette.

Example *PalLoad palfile Loads extended palette file "palfile"

Related commands *PalSave

Related SWIs Enhancer_PaletteSave (SWI &42A52)
Enhancer_PaletteLoad (SWI &42A53)

Related vectors None

*ClearSave

Saves current extended palette mode screen as a CLEAR file

Syntax *ClearSave <filename>

Parameters <filename> is any valid filename

Use *ClearSave is used to save the current extended palette mode screen as a CLEAR file, using the file format used by the graphics utility ! Translator, by John Kortink, public domain versions of which are available from a number of sources. The filetype of CLEAR files is &690. See Appendix A for details of the file format

Example *ClearSave screenfile Saves the current screen as "screenfile"

Related commands *ClearLoad

Related SWIs Enhancer_ClearSave (SWI &42A54)
Enhancer_ClearLoad (SWI &42A5C)

Related vectors None

*ClearLoad

Loads CLEAR file to the screen

Syntax *ClearLoad <filename>

Parameters <filename> is any valid filename

Use *Clearload is used to load a CLEAR file to the screen. It will change to the correct mode, program the extended palette, and load the image into screen memory. If the X & Y resolution of the CLEAR file image does not match that of any known mode, or the file does not use a palette, an error will be given.

Example *ClearLoad screenfile Loads "screenfile" as image

Related commands *ClearS ave

Related SWIs Enhancer_ClearSave (SWI &42A54)
Enhancer_ClearLoad (SWI &42A5C)

Related vectors None

*GreyScale

Sets extended palette to linear greyscale

Syntax *GreyScale <0|1>

Parameters <0|1> is 0 for normal greyscale, 1 for reversed greyscale

Use *GreyScale is used to set the extended palette to a linear greyscale, either from black to white (normal,<n>=0), or from white to black (reversed,<n>=1).

Example *GreyScale 0 Sets extended palette to normal greyscale

Related commands None

Related SWIs Enhancer_GreyScale (SWI &42A59)

Related vectors None

*Gcol

Sets graphics colour in extended palette modes.

Syntax *Gcol <colour> [<0|1>] [<action]

Parameters <colour> is an 8-bit number
[<0|1>] is 0 for foreground or 1 for background
[<action>] is the standard GCOL action (see Archimedes BASIC manual for details)

Use *Gcol is used to set the foreground or background graphics colour in extended palette modes. It is the equivalent of the BASIC GCOL command.

Example *Gcol 100 0 Sets foreground graphics colour to 100

Related commands None

Related SWIs Enhancer_Colour (SWI &42A56)
Enhancer_Gcol (SWI &42A57)

Related vectors None

*Colour

Sets text colour in extended palette modes.

Syntax *Colour <colour> [<0|1>]

Parameters <colour> is an 8-bit number
[<0|l>] is 0 for foreground or I for background

Use *Colour is used to set the foreground or background text colour in extended palette modes. It is the equivalent of the BASIC COLOUR command.

Example *Colour 200 0 Sets background text colour to 200

Related commands None

Related SWIs Enhancer_Colour (SWI &42A56)
Enhancer_Gcol (SWI &42A57)

Related vectors None

*LinkMode

Loads mode description file and links it in as a screen mode.

Syntax *LinkMode [<filename>]

Parameters [<filename>] is any valid filename.

Use *LinkMode is used to load a compiled MDL file, and to link it in so that it can be used as a normal screen mode. If the MDL file does not have a valid header, is the wrong size, or is the wrong filetype, an error will be given. *LinkMode with no parameter will list the currently linked modes, giving info on each. See Appendix D for details of the MDL mode block format.

Example *LinkMode mode22 Links file "mode22" as a screen mode.

Related commands *LinkModeClear
*DeLinkMode

Related SWIs Enhancer LinkMode (SWI &42A5D)
Enhancer_LinkModeClear (SWI &42A5E)
Enhancer_DeLinkMode (SWI &42A5F)

Related vectors None

*LinkModeClear

Deletes all currently linked modes

Syntax *LinkModeClear

Parameters None

Use *LinkModeClear is used to delete all currently linked MDL modes.

Example *LinkModeClear Clears all linked modes from memory.

Related commands *LinkMode
*DeLinkMode

Related SWIs Enhancer_LinkMode (SWI &42A5D)
Enhancer_LinkModeClear (SWI &42A5E)
Enhancer_DeLinkMode (SWI &42A5F)

Related vectors None

*DeLinkMode

Deletes individual linked mode

Syntax *DeLinkMode <val>

Parameters <val> is the number of a linked mode.

Use *DeLinkMode is used to delink a named linked MDL mode.

Example *DeLinkMode 40 Removes mode 40 from memory.

Related commands *LinkMode
*LinkModeClear

Related SWIs Enhancer_LinkMode (SWI &42A5D)
Enhancer_LinkModeClear (SWI &42A5E)
Enhancer DeLinkMode (SWI &42A5F)

Related vectors None

*TiffSave

Saves current extended palette mode screen as a class P TIFF file

Syntax *TiffSave <filename>

Parameters <filename> is any valid filename

Use *TiffSave is used to save the current extended palette mode screen as a Class P TIFF file. The filetype of TIFF files is &FF0. See Appendix B, and the Aldus/Microsoft technical documentation on TIFF files, for details of the file format.

Example *TiffSave screenfile Saves the current screen as "screenfile"

Related commands None

Related SWIs Enhancer_TiffSave (SWI &42A61)

Related vectors None

*Configure CrystalSlots

Configures occupied oscillator sockets

Syntax *Configure CrystalSlots <n>

Parameters <n> is an 8-bit number

Use *Configure CrystalSlots is used to tell the system which of the available oscillator sockets have oscillator modules in them. The 8-bit parameter is treated in a bitwise fashion, ie. each bit represents a socket. If a bit is set to 1, it has an oscillator. The sockets which are always occupied, 0-3 & 7, are masked out in software and cannot be changed. *Status shows the configured sockets as a bit pattern. A socket which does not have an oscillator in it should not be configured as occupied, as selecting it will remove all clock signals from the VIDC, which will usually crash the machine.

Example		
	*Configure CrystalSockets 2_10011111	Marks socket 4 as occupied

Related command: *Clock
*Configure DefaultCrystal

Related SWIs Enhancer Clock (SWI &42A4C)

Related vectors None

*Configure DefaultCrystal

Configures default VIDC clock

Syntax *Configure DefaultCrystal <n>

Parameters <n> is a number between 0 and 7

Use *Configure DefaultCrystal is used to tell the system which of the available oscillator sockets is to be used as the default one. The default crystal will be used for modes which run at 24 MHz on a standard machine, hence a default crystal of 1 will run all 24 MHz modes at 25.175 MHz. This setting is ignored if the monitor type is 0. A socket which is not marked as occupied cannot be set as the default.

Example *Configure DefaultCrystal 1 Sets socket 1 as the default one

Related commands *Clock
*Configure CrystalSlots

Related SWIs Enhancer_Clock (SWI &42A4C)

Related vectors None

*Configure ProcessorType

Configures installed processor type

Syntax *Configure ProcessorType <n>

Parameters <n> is a number between 0 and 3

Use *Configure ProcessorType is used to tell the system which of the various processor options is installed in the system. This is to keep internal timing loops running at the correct speed. The available processor types are as follows:

0	An ARM2 running at 8 MHz, as in standard 300, 400, & . 400/1 series machines, and the A3000.
1	An ARM3 running at 8 MHz with a 20 MHz cache.
2	An ARM3 running at 8 MHz with a 30 MHz cache.
3	An ARM3 running at 12 MHz with a 30MHz cache. This is currently only the 540.

Example

*Configure ProcessorType 2 ARM3 upgrade at 30 MHz.

Related commands None

Related SWIs None

Related vectors None

*Configure MonitorGroup

Configures group of attached monitor

Syntax *Configure MonitorGroup <n>

Parameters <n> is a number between 0 and 7

Use *Configure MonitorGroup is used to tell the system which of the various monitor groups is attached. This is so that the VIDC can be programmed with the correct parameters to keep the picture centrally on the screen. The currently available monitor groups are:

0	Philips type monitors.
1	Taxan type monitors.
2	Eizo type monitors.

Example *Configure MonitorGroup 1 Taxan monitor attached.

Related commands None

Related SWIs None

Related vectors None

*Configure IREThreshold

Configures blanking level of monitor

Syntax *Configure IREThreshold <0|1>

Parameters <0|1> is 0 for a blanking level of 0 IRE, & 1 for a level of 7.5 IRE

Use *Configure IREThreshold is used to set the blanking level of the Graphics Enhancer video signal, when in non-Archimedes video output modes. The two levels available are 0 IRE and 7.5 IRE. On most makes of monitor, the difference is not noticable, but on some it can make a large difference to the picture.

Example *Configure IREThreshold 1 Set blanking level of 7.5 IRE.

Related commands None

Related SWIs None

Related vectors None

*Configure TurboModes

Enables/disables optional mode booster board

Syntax *Configure TurboModes <0|1>

Parameters <0|1> is 0 for off, 1 for on.

Use *Configure TurboModes is used to turn on or off the optional mode booster board. When it is on, the VGA modes 103 & 126 are run at 40 MHz, giving a refresh rate of about 52 Hz. In addition, mode 127 (640 x 512 extended palette) is enabled. This requires a mode booster board to be installed, and a 40 MHz oscillator to be plugged into socket OSC4. This upgrade is not absolutely guaranteed, as VLSI specifcations give the maximun operating speed of the VIDC as 36 MHz, but in practice we have found that most VIDCs will run at 40 MHz correctly.

Example		
	*Configure TurboModes 0	Disable mode booster (default).

Related commands None

Related SWIs None

Related vectors None

Graphics Enhancer SWI calls

&42A40 `Enhancer_PaletteBlockWrite`
Writes block of 24-bit palette entries.

&42A41 `Enhancer_PaletteBlockRead`
Reads current 24-bit palette entry table.

&42A42 `Enhancer_PaletteWrite`
Writes individual palette entries.

&42A43 `Enhancer_PaletteRead`
Reads individual palette entries.

&42A44 `Enhancer_PixelMaskWrite`
Writes pixel mask.

&42A45 `Enhancer_PixelMaskRead`
Reads pixel mask.

&42A46 `Enhancer_Default`
Resets Graphics Enhancer board.

&42A47 `Enhancer_NormalVideo`
Restores normal Archimedes video output.

&42A48 `Enhancer_BorderOff`
Turns screen border off.

&42A49 `Enhancer_PalSet`
Sets Archimedes palette to red scale.

&42A4A `Enhancer_ExtPalette`
Enables 24-bit palette mode.

&42A4B `Enhancer_Dac`
Enables direct DAC mode.

&42A4C `Enhancer_Clock`
Sets or reads VIDC clock rate.

&42A4D `Enhancer_HardwarePresent`
Checks for presence of Graphics Enhancer.

&42A4E `Enhancer_SetDesktopPalette`
Sets standard desktop palette in extended palette modes.

&42A4F `Enhancer_Pointer`
Enables/disables mouse pointer X-coordinate correction factor

&42A50 `Enhancer_VsyncUpdatePalette`
Enables/disables extended palette vsync handler.

&42A51 `Enhancer_ModeValid` Checks for a valid extended palette mode.

&42A52 `Enhancer_PaletteSave`
Saves current extended palette.

&42A53 `Enhancer_PaletteLoad`
Loads extended palette file.

&42A54 `Enhancer_ClearSave`
Saves screen as CLEAR file.

&42A55 `Enhancer_VIDC`
Programs VIDC registers.

&42A56 `Enhancer_Colour`
Sets text colour in extended palette modes.

&42A57 `Enhancer_Gcol`
Sets graphics colour in extended palette modes.

&42A58 `Enhancer_CurrentModeValid`
Checks if current mode is extended palette mode.

&42A59 `Enhancer_GreyScale` Sets 256 level linear greyscale.

&42A5A `Enhancer_PaletteReadPointer` Reads pointer to current 24-bit palette table.

&42A5B `Enhancer_SpriteOp`
Does various sprite operations in extended palette modes.

&42A5C `Enhancer_ClearLoad` Loads CLEAR file to the screen.

&42A5D `Enhancer_LinkMode`
Loads mode definition file and links in as screen mode.

&42A5E `Enhancer_LinkModeClear`
Deletes all linked modes.

&42A5F `Enhancer_DeLinkMode`
Delinks individual linked modes.

&42A60 `Enhancer_HardwareBaseAddress`
Returns the base address of the Graphics Enhancer MEMC interface.

&42A61 `Enhancer_TiffSave`
Saves extended palette screen as a TIFF class P file.

Enhancer_PaletteBlockWrite (SWI &42A40)

Write block of 24-bit palette entries

On entry

R0 = pointer to block of palette entry words
R1 = number of words to process

On exit

R0 = pointer to word after last one processed
R1 = number of words processed

Interrupts

Processor mode

Processor is in SVC mode

Re-entrancy

Use

This call allows updating of more than one palette entry at a time. Up to the full 256 entries can be written at once. Note that colour 0 cannot be programmed to other than black, for various hardware-related reasons.

The format of the palette table is as follows:

```
Base +0:    BBGGRRCC
Base +4:    BBGGRRCC
  .             .
  .             .
  .             .
Base +n:    BBGGRRCC

Where:      BB = 8-bit blue value
            GG = 8-bit green value
            RR = 8-bit red value
            CC = 8-bit colour value
```

Related SWIs

Enhancer_PaletteWrite (SWI &42A42)
Enhancer_PaletteBlockRead (SWI &42A41)
Enhancer_PaletteRead (SWI &42A43)
Enhancer_PaletteReadPointer (SWI &42A5A)

Related vectors

None

Enhancer_PaletteBlockRead (SWI &42A41)

Read current 24-bit palette entry table

On entry R0 = pointer to 1024 byte buffer for table copy

On exit R0 preserved

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call copies the current palette table to the buffer given in R0. See Enhancer_PaletteBlockWrite description for table format.

Related SWIs Enhancer_PaletteBlockWrite (SWI &42A40)
Enhancer_PaletteWrite (SWI &42A42)
Enhancer_PaletteRead (SWI &42A43)
Enhancer_PaletteReadPointer (SWI &42A5A)

Related vectors None

Enhancer_PaletteWrite (SWI &42A42)

Write individual palette entry

On entry

R0 = Colour number
R1 = 8-bit red component
R2 = 8-bit green component
R.3 = 8-bit blue component

On exit

R0 - R3 preserved

Interrupts

Processor mode

Processor is in SVC mode

Re-entrancy

Use

This call writes a single 24-bit palette entry.Note that for various hardware-related reasons colour 0 cannot be programmed, and is always black.

Related SWIs

Enhancer_PaletteBlockWrite (SWI &42A40)
Enhancer_PaletteBlockRead (SWI &42A41)
Enhancer_PaletteRead (SWI &42A43)
Enhancer_PaletteReadPointer (SWI &42A5A)

Related vectors

None

Enhancer_PaletteRead (SWI &42A43)

Read individual palette entry

On entry R0 = Colour number

On exit R0 preserved
R1 = 8-bit red component
R2 = 8-bit green component
R3 = 8-bit blue component

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call reads a single 24-bit palette entry.

Related SWIs Enhancer_PaletteBlockWrite (SWI &42A40)
Enhancer_PaletteBlockRead (SWI &42A41)
Enhancer_PaletteRead (SWI &42A43)
Enhancer_PaletteReadPointer (SWI &42A5A)

Related vectors None

Enhancer_PixelMaskWrite (SWI &42A44)

Write pixel mask

On entry R0 = 8-bit pixel mask value

On exit R0 preserved

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call allows writing of the pixel mask. The pixel mask is used to mask the colour address before it is send to the the palette. A '1' in a location in the pixel mask leaves the corresponding bit in the colour address intact, while a '0' will mask out the bit. For example, a pixel mask of 126 (%01111110) and colour 149 (%10010101) would actually send colour 20 (%00010100) to the palette.

Related SWIs Enhancer_PixelMaskRead (SWI &42A45)

Related vectors None

Enhancer_PixelMaskRead (SWI &42A45)

Read pixel mask

On entry

On exit R0 = current pixel mask

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call reads the current pixel mask.

Related SWIs Enhancer_PixelMaskWrite (SWI &42A44)

Related vectors None

Enhancer_Default (SWI &42A46)

Restore default Graphics Enhancer settings

On entry

On exit

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call resets the Graphics Enhancer to give standard Archimedes video output and 24 MHz VIDC clock.

Related SWIs Enhancer_NormalVideo (SWI &42A47)

Related vectors None

Enhancer_NormalVideo (SWI &42A47)

Restore normal Archimedes video

On entry

On exit

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call resets the Graphics Enhancer to give standard Archimedes video output, but does not affect the currently set VIDC clock rate.

Related SWIs Enhancer Default (SWI &42A46)

Related vectors None

Enhancer_BorderOff (SWI &42A48)

Turn border off

On entry

On exit

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call turns the screen border off by setting it to black. It is used internally by the Graphics Enhancer software.

Related SWIs None

Related vectors None

Enhancer_PalSet (SWI &42A49)

Set Archimedes palette

On entry

On exit

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call sets the Archimedes palette to a linear red scale, in 4 bpp modes. It is used internally by the Graphics Enhancer software.

Related SWIs None

Related vectors None

Enhancer_ExtPalette (SWI &42A4A)

Enable 24-bit palette mode

On entry

On exit

Interrupts

Processor mode

Processor is in SVC mode

Re-entrancy

Use

This call enables the 24-bit extended palette mode. It is transparently used whenever an extended palette mode is selected, and would not normally be used otherwise. To get correct output when forcing the Graphics Enhancer to extended palette mode, a screen mode that is synchronous with the VIDC clock must be used (ie. modes 16 & 24, or modes based upon them).

Related SWIs

Enhancer_Dac (SWI &42A4B)

Related vectors

None

Enhancer_Dac (SWI &42A4B)

Enable direct 8, 12 or 16 bit DAC mode

On entry

R0 = 2 for 8-bit DAC mode
R0 = 3 for 12-bit DAC mode
R0 = 4 for 16-bit DAC mode

On exit

R0 preserved

Interrupts

Processor mode

Processor is in SVC mode

Re-entrancy

Use

This call enables the direct DAC mode. It is transparently used whenever a 12 or 16 bpp mode is selected, and would not normally be used otherwise. To get correct output when forcing the Graphics Enhancer to DAC mode, a screen mode that is synchronous with the VIDC clock must be used (ie. modes 16 & 24, or modes based upon them). 8-bit DAC mode is available in hardware, but not used.

Related SWIs

Enhancer_ExtPalette (SWI &42A4A)

Related vectors

None

Enhancer_Clock (SWI &42A4C)

Set or read VIDC clock rate

On entry

R0 = socket number of desired oscillator (0-7)
R0 = 8 for information on currently selected clock

On exit

If R0 on entry is 0-7, R0 corrupted
If R0 on entry is 8, R0 returns currently selected clock

Interrupts

Processor mode

Processor is in SVC mode

Re-entrancy

Use

This call will connect the oscillator in the requested socket to the VIDC, or return the currently selected oscillator socket. The default clock frequencies and their socket numbers are as follows:

Socket 0: 24 MHz
1: 25.175 MHz
2: 32 MHz
3: 36 MHz
4: empty (for user-supplied oscillator)
5: empty (for user-supplied oscillator)
6: empty (for user-supplied oscillator)
7: 24 MHz

This call is transparently used whenever a mode requiring a non-standard VIDC clock rate is selected. Only sockets that have been marked as occupied by *Configure CrystalSockets can be selected. If an empty socket is asked for, the SWI call will not change the current VIDC rate. See the documentation on the Graphics Enhancer *Configure commands for further details.

Related SWIs

None

Related vectors

None

Enhancer_HardwarePresent (SWI &42A4D)

Check for presence of Graphics Enhancer

On entry

On exit

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call is used internally to check for the presence of the Graphics Enhancer hardware. The 'X' form of the SWI will return with the V flag clear if the hardware is present, and set if it is not.

Related SWIs None

Related vectors None

Enhancer_SetDesktopPalette (SWI &42A4E)

Set standard desktop palette in extended palette modes

On entry

On exit

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call sets the extended palette colours used for the desktop to give the standard desktop look.

Related SWIs None

Related vectors None

Enhancer_Pointer (SWI &42A4F)

Enable/disable mouse pointer X-coordinate correction factor

On entry

R0 = 0 for normal pointer X-coordinates
R0 = 1 for corrected pointer X-coordinates

On exit

R0 preserved

Interrupts

Processor mode

Re-entrancy

Processor is in SVC mode

Use

This call is used internally to correct mouse pointer X-coordinate errors in extended palette modes. It should not be used otherwise, as it can have some very odd effects on the desktop.

Related SWIs

None

Related vectors

None

Enhancer_VsyncUpdatePalette (SWI &42A50)

Enable/disable extended palette vsync handler

On entry

R0 = 0 for vsync handler disable
R0 = 1 for vsync handler enable

On exit

R0 preserved

Interrupts

Processor mode

Re-entrancy

Processor is in SVC mode

Use

This call is used internally to enable or disable the extended palette vsync handler. If the vsync handler is turned off, the extended palette cannot be updated.

Related SWIs

None

Related vectors

None

Enhancer_ModeValid (SWI &42A51)

Check for valid extended palette mode

On entry R0 = mode to check

On exit R0 = 0 if the mode is not an extended palette one R0 = 1 if the mode is an extended palette one

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call is used to check if a mode is one that uses the extended palette.

Related SWIs None

Related vectors None

Enhancer_PaletteSave (SWI &42A52)

Save current extended palette

On entry R0 = pointer to filename

On exit R0 corrupted

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use

This call is used to save the current extended palette. The filetype of the palette file is &C8A. The format of the extended palette file is as follows:

Word 0:	&314C4150	header="PAL1"
Word 1:	Current pixel mask	
Word 2:	BBGGRRCC	
.	.	
.	.	
.	.	
Word 1028:	BBGGRRCC	

The palette file uses the Enhancer_PaletteBlockWrite SWI format, with the addition of a file identification header, and the current pixel mask.

Related SWIs Enhancer_PaletteBlockWrite (SWI &42A40)
Enhancer_PaletteBlockRead (SWI &42A41)
Enhancer_PaletteLoad (SWI &42A53)

Related vectors None

Enhancer_PaletteLoad (SWI &42A53)

Load extended palette file

On entry R0 = pointer to filename

On exit R0 corrupted

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call loads an extended palette file, and programs the palette and the pixel mask from it.

Related SWIs Enhancer_PaletteBlockWrite (SWI &42A40)
Enhancer_PaletteBlockRead (SWI &42A41)
Enhancer_PaletteSave (SWI &42A52)

Related vectors None

Enhancer_ClearSave (SWI &42A54)

Save screen as CLEAR file

On entry R0 = pointer to filename

On exit R0 corrupted

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call saves a Graphics Enhancer extended palette mode screen as a CLEAR file, using the file format used by the graphics utility ! Translator, by John Kortink, public domain versions of which are available from a number of sources. The filetype of CLEAR files is &690. See Appendix A for details of the file format.

Related SWIs Enhancer_ClearLoad (SWI &42A5C)

Related vectors None

Enhancer_VIDC (SWI &42A55)

Program VIDC registers

On entry R0 = VIDC register number

On exit R1 = value to program register with

Interrupts R0, R1 preserved

Processor mode

Re-entrancy Processor is in SVC mode

Use This call allows direct programming of the VIDC registers connected with screen modes, without having to write a screen definition module. Risc-OS will not know about the new values, so the VDU drivers will not correctly handle the changes, but it can be useful for getting a rough idea of what a new screen mode will look like, etc. See Appendix H for more details of register numbers and allowed values.

Warning! Use this call with caution, as programming the VIDC with the wrong values could damage your monitor.

Related SWIs None

Related vectors None

Enhancer_Colour (SWI&42A56)

Set text colour in extended palette modes

On entry

R0 = 8-bit colour number
R1 = 0 for foreground colour, 1 for background colour

On exit

R0, R1 preserved

Interrupts

Processor mode

Processor is in SVC mode

Re-entrancy

Use

This call sets the foreground or background text colour in extended palette modes. It is equivalent to the BASIC COLOUR command, but works from any language.

Related SWIs

Enhancer_Gcol (SWI &42A57)

Related vectors

None

Enhancer_Gcol (SWI &42A57)

Set graphics colour in extended palette modes

On entry

R0 = 8-bit colour number
R1 = 0 for foreground colour, 1 for background colour
R2 = GCOL action (See Archimedes manual)

On exit

R0-R2 preserved

Interrupts

Processor mode

Processor is in SVC mode

Re-entrancy

Use

This call sets the foreground or background graphics colour in extended palette modes. It is equivalent to the BASIC GCOL command, but works from any language. See the Archimedes BASIC manual for details of the GCOL parameters.

Related SWIs

Enhancer_Colour (SWI &42A56)

Related vectors

None

Enhancer_CurrentModeValid (SWI &42A58)

Check if current mode is extended palette mode

On entry

On exit

R0 = 0 if the current mode is not an extended palette one
R0 = 1 if the current mode is an extended palette one

Interrupts

Processor mode

Processor is in SVC mode

Re-entrancy

Use

This call checks if the current mode is one that uses the extended palette.

Related SWIs

Enhancer_ModeValid (SWI &42A51)

Related vectors

None

Enhancer_GreyScale (SWI &42A59)

Set 256 level linear greyscale

On entry

R0 = 0 for black to white greyscale
R0 = 1 for white to black greyscale

On exit

R0 preserved

Interrupts

Processor mode

Processor is in SVC mode

Re-entrancy

Use

This call sets the extended palette to a 256 level linear greyscale, either from black (colour 0) to white (colour 255), or from white (colour 0) to black (colour 255).

Related SWIs

Enhancer_ SetDesktopPalette (SWI &42A4E)

Related vectors

None

Enhancer_PaletteReadPointer (SWI &42A5A)

Read current 24-bit palette entry table

On entry

On exit R0 = pointer to start of palette table

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call returns the base address of the current palette table. See Enhancer_PaletteBlockWrite description for table format.

Related SWIs Enhancer_PaletteBlockWrite (SWI &42A40)
Enhancer PaletteBlockRead (SWI &42A41)
Enhancer PaletteWrite (SWI &42A42)
Enhancer_PaletteRead (SWI &42A43)

Related vectors None

Enhancer_SpriteOp (SWI &42A5B)

Do various sprite operations

On entry

R0 = reason code
Other registers are parameters

On exit

R0 preserved
Other registers depend on reason code

Interrupts

Processor mode

Processor is in SVC mode

Re-entrancy

Use

This call provides equivalents to some of the OS_SpriteOp calls, for use in the extended palette modes.

R0	Meaning
41	Read pixel colour
42	Write pixel colour

Related SWIs

None

Related vectors

None

Enhancer_SpriteOp 41 (SWI &42A5B)

Read pixel colour

On entry

R0 = 41 (&29)
R1 = pointer to control block of sprite area
R2 = sprite pointer
R3 = x coordinate (in pixels)
R4 = y coordinate (in pixels)

On exit

R0 - R4 preserved
RS = 8-bit colour number

Interrupts

Processor mode

Processor is in SVC mode

Re-entrancy

Use

This call does the equivalent of the OS_SpriteOp 41 call, but returns an 8-bit colour number for use in the extended palette modes.

Related SWIs

Enhancer_SpriteOp 42 (SWI &42A5B)

Related vectors

None

Enhancer_SpriteOp 42 (SWI &42A5B)

Write pixel colour

On entry

R0 = 42 (&2A)
R1 = pointer to control block of sprite area
R2 = sprite pointer
R3 = x coordinate (in pixels)
R4 = y coordinate (in pixels)
R5 = 8-bit colour number

On exit

R0 - R5 preserved

Interrupts

Processor mode

Processor is in SVC mode

Re-entrancy

Use

This call does the equivalent of the OS_SpriteOp 42 call, but takes an 8-bit colour number for use in the extended palette modes.

Relted SWIs

Enhancer_SpriteOp 41 (SWI &42A5B)

Related vectors

None

Enhancer_ClearLoad (SWI &42A5C)

Load CLEAR file to screen

On entry R0 = pointer to filename

On exit R0 corrupted

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call loads a CLEAR file to the screen and programs the palette accordingly. If the X & Y resolution of the CLEAR file image does not match that of any known mode, or the image uses more than 8 bpp, an error is given, and the call aborts.

Related SWIs Enhancer_ClearSave (SWI &42A54)

Related vectors None

Enhancer_LinkMode (SWI &42A5D)

Load & link compiled MDL file

On entry R0 = pointer to filename

On exit R0 = number of modes now linked

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call loads a compiled MDL file and links it in as a screen mode. If the file has an invalid header, is the wrong size, or is the wrong filetype, an error is given and the call aborts. It will also abort with an error if the maximum number of modes (36) are already linked. See Appendix D for details of the MDL mode block format.

Related SWIs Enhancer_LinkModeClear (SWI &42A5E)
Enhancer_DeLinkMode (SWI &42A5F)

Related vectors None

Enhancer_LinkModeClear (SWI &42A5E)

Delete all linked modes

On entry

On exit

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call removes all linked modes from memory.

Related SWIs Enhancer_LinkMode (SWI &42A5D)
Enhancer_DeLinkMode (SWI &42A5F)

Related vectors None

Enhancer_DeLinkMode (SWI &42A5F)

Delete individual linked mode

On entry R0 = linked mode number

On exit R0 = if mode requested exists, the new number of modes now linked
R0 = -1 if no modes linked

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call removes a specified linked mode from memory, and returns with the new number of linked modes in R0. If there are no linked modes, it returns with -1 in R0. If the linked mode specified does not exist, the call aborts with an error.

Related SWIs Enhancer_LinkMode (SWI &42A5D)
Enhancer_LinkModeClear (SWI &42A5E)

Related vectors None

Enhancer_HardwareBaseAddress (SWI &42A60)

Return MEMC base address of Graphics Enhancer hardware

On entry

On exit R0 = hardware MEMC base address + &2000

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call returns in R0 the base address of the Graphics Enhancer MEMC interface + &2000.

Related SWIs None

Related vectors None

Enhancer_TiffSave (SWI &42A61)

Saves the current extended palette screen as class P TIFF file

On entry R0 = pointer to filename

On exit R0 corrupted

Interrupts

Processor mode Processor is in SVC mode

Re-entrancy

Use This call saves a Graphics Enhancer extended palette mode screen as a class P TIFF file. The filetype of TIFF files is &FF0. See Appendix B, and the Aldus/Microsoft technical documentation on TIFF files for more details of the file format.

Related SWIs None

Related vectors None

APPENDIX A: CLEAR file format

The CLEAR file format that the Graphics Enhancer uses was designed by John Kortink for his graphics utility !Translator, as a very simple but versatile format for storing high-resolution images, with or without a palette. Basically, the CLEAR format is an uncompressed image file, which has a palette definition in it for images that have up to 8 bpp, and is simply raw RGB data for images above 8 bpp. The format will handle images up to the full 24 bpp that is the maximum usually used. A full definition of the CLEAR file format is as follows:

File start	+0:	A string containing the name of the program that created the file, n bytes long.
	+n:	A null byte as the string terminator.
	+(n+1):	A word containing the version number of the creating program. The lower 2 digits of this number are to the right of an implied decimal point, the rest are to the left. (ie. A version number of 1.45 would be stored as 145)
	+(n+5):	A word containing the X resolution of the image in pixels.
	+(n+9):	A word containing the Y resolution of the image in pixels.
	+(n+13):	A word containing the number of bits per pixel of the image. If the bpp is between 1 and 8, then a 24-bit wide palette table for all the available colours follows this entry. If the image has more than 8 bpp, the raw image data follows, stored as 1 byte each of R, G, & B for each pixel in the image.
	+(n+17):	If the image has 8 bpp or less, the next ((2^bpp)*3) bytes contain a 24-bit palette entry for each colour, stored as the 8-bit colour definitions for R, G, & B. If the image has more than 8 bpp, the next ((X*Y)*3) bytes contain the raw image data, stored as 1 byte per R, G, & B for each pixel. If the raw data is for less than full 24 bpp, the data is shifted to the top of the byte, and the lower bits are set to 0. For example, 12 bpp image data would be stored as the top 4 bits of each byte.
+(n+17+((2^bpp)*3)):		This is only if the image has 8 or less bpp. In this case, the next (X*Y) bytes contain 1-byte colour numbers, which are used as pointers to the appropriate palette entry in the table preceding this section.

APPENDIX B: Class P TIFF file format

A full description of the TIFF file format goes on for some 38 pages, and is very technical. For those people wanting full details on the TIFF specification, the Aldus/Microsoft TIFF 5.0 technical documentation is recommended. The following is a brief description of the subset of the full TIFF specification that is used to store 8-bit palette images, such as those made by the Graphics Enhancer. These are primarily intended for exporting images to other computers.

A TIFF (Tagged Image File Format) file consists of a number of fields, some mandatory and some optional, each of which has a unique identification number, or tag. The TIFF file starts with an 8 byte file header that points to one or more image file directories, of IFDs. Each IFD contains information about the image it is attached to, and also has pointers to the image itself.

The first 2 bytes of the file contain a number specifying the order of the bytes that make up the rest of the file. There are two options, &4949 (least significant to most significant), and &4D4D (most significant to least significant). The Graphics Enhancer uses &4949. The next 2 bytes contain &002A, which is used as a TIFF file identifier, in conjunction with the byte order specifier. The next 4 bytes contain a 32-bit pointer to the start of the first IFD. The IFD must be aligned to 2-byte boundaries, but may be anywhere in the file. All pointers used are referenced to the start of the file, which is offset 0. The Graphics Enhancer puts 8 in this field, signifying that the first IFD immediately follows the file header.

An IFD consists of a number of IFD entries, with a 32-bit pointer to the next IFD, which is 0 if this IFD is the last in the file. The first 2 bytes of an IFD contains the number of entries in it. Each IFD entry is 12 bytes long, and has a format as follows;

Bytes 0-1	Tag
Bytes 2-3	Field type
Bytes 4-7	Field length
Bytes 8-11	Value offset

The tags must be in numerically ascending order. The field types are as follows;

1 Byte	8-bit unsigned integer
2 ASCI1	0 terminated ASCII string
3 Short	16-bit unsigned integer
4 Long	32-bit unsigned integer
5 Rational	2 32-bit integers, the first being the numerator of a fraction, the second being the denominator

The length of a field is given in units of the field type, ie. a single Long is 1, five Bytes is 5, etc. The length of a string includes the terminating 0, but not any pad bytes.

If a field value is 4 bytes or less, the value offset is the value itself, otherwise it is a pointer to the actual value.

A list of the tags used by the Graphics Enhancer, and the values used, is given below.

ImageWidth, Tag &100, Short, Length 1
X resolution of screen

ImageLength, Tag &101, Short, Length 1
Y resolution of screen

BitsPerSample, Tag &102, Short, Length 1
8

Compression, Tag &103, Short, Length 1
1 (no compression)

PhotometricInterpretation, Tag &106, Short, Length 1
3 (palette used)

FillOrder, Tag &10A, Short, Length 1
1 (normal order)

Make, Tag &10F, ASCII, optional
Pointer to information string 1

Model, Tag &110, ASCII, optional
Pointer to information string 2

StripOffsets, Tag &111, Long, Length Y resolution/16
Pointer to table of 16-line image strip pointers

SamplesPerPixel, Tag &115, Short, Length 1
1 (1 byte per pixel)

RowsPerStrip, Tag &116, Long, Length 1
16 (16 lines per strip)

StripByteCounts, Tag &117, Long,Length 1
X resolution*16 (bytes per strip)

XResolution, Tag &11A, Rational, Length 1
Pointer to 64-bit value (&0000004000000000, 40 pixels/cm)

YResolution, Tag &11B, Rational, Length 1
Pointer to 64-bit value (&0000004000000000, 40 pixels/cm)

ResolutionUnit, Tag &128, Short, Length 1
3 (measure in cm)

Software, Tag &131, ASCII, optional
Pointer to information string 3

HostComputer, Tag &13C, ASCII, optional
Pointer to information string 4

ColourMap, &140, Short, Length 768
Pointer to palette data

The actual image is referenced by the StripOffsets field, which contains a pointer to the beginning of a table of pointers which point to the beginning of a number of strips of image data. Each strip is 16 lines deep, and (X resolution) pixels wide. The strips can be anywhere in the file, but in the case of the Graphics Enhancer files, are in a contiguous block. Strictly speaking, the StripByteCount field should have one entry for each image strip, but as they are all the same length, only one value is actually needed.

The palette data, referenced by the ColourMap field, is laid out in a rather strange manner. Each palette entry is a 16-bit value, with 0 as black and 65535 as white. The 256 red values are in a single block, followed by 256 green values and 256 blue values. The Graphics Enhancer software scales the 8-bit palette values to 16 bits by shifting them 8 bits left. This method does not quite give peak values, but is very fast.

The various information strings are optional, but used to distinguish Graphics Enhancer TIFF files from those created by other means.

The tags that must be used, according to the above mentioned Aldus/Microsoft for class P TIFF files documentation, are:

&100 ImageWidth	&115 SamplesPerPixel
&101 ImageLength	&117 StripByteCounts
&102 BitsPerSample	&11A XResolution
&103 Compression	&11B YResolution
& 106PhotometricInterpretation	& 128 ResolutionUnit
&111S tripOffsets	&140 ColourMap

APPENDIX C: Using the 12 BPP & 16 BPP modes

The use of the 12 & 16 bpp modes can be quite complex, and warrants a detailed explanation. The main difficulty is that Risc-OS does not handle more than 8 bpp, so you have to go around it. The Graphics Enhancer works by adding several 4 bpp pixels together to make larger pixels, and reduces the X resolution of the screen correspondingly. In 12 bpp modes, 3 pixels are added together to get one 12 bpp pixel , which is 3 time as wide as the individual sub-pixels that make it up. In 16 bpp modes, 4 pixels are added together, etc.

This means that the VDU drivers think that they are plotting 4 bpp pixels, and the colour range for each sub-pixel is 0 to 15. In the 12 bpp modes, the first 2 sub-pixels on each line must be skipped, as the Graphics Enhancer hardware, for various reasons, does not correctly use them. This effectively reduces the X resolution of that 12 bpp modes by one 12 bpp pixel.

To drive the 12 bpp modes from BASIC, you must write to each sub-pixel in turn, for each larger pixel. The screen is arranged as follows:

Start of line: NNRGBRGB etc. to the end of each line.

The sub-pixels marked N are not used and must be skipped. Each of the R, G, & B pixels is one 4 bpp sub-pixel. For example, to set the lower left pixel in a 12 bpp mode, such as mode 119, to white, you must do the following:

```
GCOL 15
POINT 4,0
POINT 6,0
POINT 8,0
```

This sets each of the individual sub-pixels to full intensity, giving the overall effect of a full intensity 12 bpp pixel.

In some respects, writing to the screen in assembler is easier. The bit pattern of a 12 bpp pixel is this:

$$B^3B^2B^1B^0G^3G^2G^1G^0R^3R^2R^1R^0$$

As in BASIC, the first byte on each line must be skipped. It is usually easier to caleulate the 12 bpp values for the first 2 pixels, giving a 3 byte value, and to write this to the screen. This prevents having to write 2 bytes, read the second one back, add 4 bits of the next 12 bit value to it, and write it to the screen again.

The 16 bpp modes are similar, but do not need to have any pixels skipped at the beginning of the line. The extra 4 bits are arranged as $IB^4G^4R^4$. The I bit is connected to the fifth bit of each of R, G, & B, and acts as an intensity bit. Therefore, the bit pattern of a 16 bpp pixel is as follows:

$$IB^4G^4R^4B^3B^2B^1B^0G^3G^2G^2G^0B^3B^2B^1R^0$$

To set the lower left pixel in a 16 bpp mode to white, ignoring the I bit, from BASIC, would require the following code:

```
GCOL 7
POINT 0,0          This sets bit 4 of RGB without setting the I bit.
GCOL 15
POINT 2,0
POINT 4,0
POINT 6,0
```

In assembler, the lower 4 bits of each of the RGB components can be assembled into a 12 bit value, and the upper bit of each component, plus the I bit if needed, can be shifted 12 bits left and added to the calculated 12 bit value. This complete 16 bit pixel value can now be written to the screen as 2 bytes. Again, it can often be quicker to calculate 2 successive 16 bit pixel values and combine them together into one 32 bit word, which can be written to the screen using STR.

There are some example programs showing the above techniques on the utilities disk supplied with the Graphics Enhancer.

APPENDIX D: Mode Description Language

The Graphics Enhancer software has a powerful and versatile way of creating new modes, which uses much less memory than the usual way of creating a relocatable module for the mode definition. It is the Mode Description Language, or MDL. This is a simple language, which uses standard text files, created with !Edit, which are then compiled to a standard mode definition block by the Graphics Enhancer utility !Mdlcomp. The compiled MDL files are only 200 bytes long, and take up very little disk space, or memory. The source code for the mode block can be altered and recompiled at any time, without having to start again from scratch. There are 3 SWI's, with their associated *commands, which are used to load and delete the MDL modes to and from memory. They are;

Enhancer_LinkMode & *LinkMode
Enhancer_LinkModeClear & *LinkModeClear
Enhancer_DeLinkMode & *DeLinkMode

See the appropriate sections of this manual for details of these commands.

The !Mdlcomp utility is used in the following manner:

Create a MDL source file using !Edit, and save it as a text file. Run the !Mdlcomp program by double-clicking on it. Drag the text file you have created to the !Mdlcomp icon, on the icon bar. A save window will appear, with a default save filename. If you wish to change this filename, select it by clicking on it, and type in a new filename. Now you can drag the icon for the compiled MDL file to the directory viewer where you want to save it. The compiler will now load the source file, if it is a text file, and attempt to compile it. It first scans the source file looking for syntax errors. If it finds any, a standard error window will be opened, and the line number of the offending statement will be given, along with the text of the line itself. At this point, any syntax errors can be corrected. If the compiler can find no syntax errors, it next checks for the presence of the few mandatory statements that form the file header. If these are present, it will then compile the source file, and save the compiled MDL file.

It is possible for a MDL file to eompile correctly, and still not apparently work as a new mode. This is invariably because the source file gave silly parameters for the VIDC registers, or the RISC-OS variables. See Appendix H for details of the VIDC registers, and the entry for the OS_ReadModeVariable SWI (&35) on page 350 of the Archimedes Programmers Reference Manual for full information on allowed values.

Some example MDL souree files are included on the Graphics Enhancer utility disk.

A RISC OS compliant mode generator that produces MDL text files will be available shortly.

A list of the MDL keywords is given below, along with their descriptions, ranges, and whether they are optional or mandatory.

hcr	**horizontal cycle register**	**0-1023**	**optional**
hswr	**horizontal sync width register**	**0-1023**	**optional**
hbsr	**horizontal border start register**	**0-1023**	**optional**
hdsr	**horizontal display start register**	**0-1023**	**optional**
hder	**horizontal display end register**	**0-1023**	**optional**
hber	**horizontal border end register**	**0-1023**	**optional**
hcsn	**horizontal cursor start register**	**0-1023**	**optional**
hir	**horizontal interlace register**	**0-1023**	**optional**
vcr	**vertical cycle register**	**0-1023**	**optional**
vswr	**vertical sync width register**	**0-1023**	**optional**
vbsr	**vertical border start register**	**0-1023**	**optional**
vdsr	**vertical display start register**	**0-1023**	**optional**
vder	**vertical display end register**	**0-1023**	**optional**
vber	**vertical border end register**	**0-1023**	**optional**
vcsr	**vertical cursor start register**	**0-1023**	**optional**
vcer	**vertical cursor end register**	**0-1023**	**optional**
cr	**control register**	**9 bits (bitwise)**	**optional**
maxcol	**maximum column number -1**	**127**	**optional**
maxrow	**maximum row number -1**	**127**	**optional**
logcol	**maximum logical colour (either 1,3,15, or 63)**	**63**	**optional**
xeig	**x-coordinate bit shift value**	**31**	**optional**
yeig	**y-coordinate bit shift value**	**31**	**optional**
linelen	**bytes per pixel row (chars • bpp • pixels per char)/8**	**2048**	**optional**
screen	**bytes per screen (must be multiple of 256 bytes)**	**491520 (480K)**	**optional**
log2bpp	**LOG base 2 bpp**	**3**	**optional**
log2bpc	**LOG base 2 bytes per char**	**3**	**optional**
xmax	**maximum x pixel -1**	**2047**	**optional**
ymax	**maximum y pixel -1**	**2047**	**optional**
montype	**monitor type of new mode**	**7**	**mandatory**
mode	**mode number of new mode**	**127**	**mandatory**
vbasemode	**vide list base mode**	**127**	**mandatory**
wbasemode	**workspace list base mode**	**127**	**mandatory**
vformat	**vide list format (0 for RISCOS 2)**	**1**	**mandatory**
wformat	**workspace format (0 for RISCOS 2)**	**1**	**mandatory**
crystal	**slot of oscillator for new mode**	**7**	**mandatory**
video	**video type (0=normal. 1=extended palate, 2=12 bpp DAC. 3=16 bpp DAC)**	**3**	**mandatory**
	comment line (ignored)	**-**	**optional**

Notes

(1) The syntax of an MDL source code line is <keyword>;<value>.
(2) There must only be one statement per line.
(3) A line beginning with \ is ignored, and is used for comments on the source file.
(4) Each mode defined must be in a separate file.
(5) Any optional parameter which is left out will use the default value of the base mode.
(6) In RISC-OS 2, both the vidc list format and the workspace format have the same number, which is 0. RISC-OS 2.01, as in the A540, can also have the format number 1. This format is currently not fully supported by the Graphics Enhancer software, for hardware reasons.
(7) With multisync monitors, the setting of the DefaultCrystal *Configure command is used for all modes that normally run at 24 MHz, ie. crystal 0. To force a mode to run at 24 MHz, regardless of the DefaultCrystal setting, use crystal 7 instead.

A definition of the MDL mode block is given below;

word offset	description
word 0	header (&45444F4D • "MODE")
word 1	definition mode number
word 2	crystal slot for definition
word 3	monitor type of definition
word 4	video type
word 5	vidc list format
word 6	vidc list base mode
word 7	vidc parameter 1
word 8	vidc parameter 2
word 9	vidc parameter 3
word 10	vidc parameter 4
word 11	vidc parameter 5
word 12	vidc parameter 6
word 13	vidc parameter 7
word 14	vidc parameter 8
word 15	vidc parameter 9
word 16	vidc parameter 10
word 17	vidc parameter 11
word 18	vidc parameter 12
word 19	vidc parameter 13
word 20	vidc parameter 14
word 21	vidc parameter 15
word 22	vidc parameter 16
word 23	vidc parameter 17
word 24	&FFFFFFFF
word 25	workspace list format
word 26	workspace base mode
word 27	mode variable index 1
word 28	mode variable value 1
word 29	mode variable index 2
word 30	mode variable value 2
word 31	mode variable index 3
word 32	mode variable value 3
word 33	mode variable index 4
word 34	mode variable value 4
word 35	mode variable index 5
word 36	mode variable value 5
word 37	mode variable index 6
word 38	mode variable value 6
word 39	mode variable index 7
word 40	mode variable value 7
word 41	mode variable index 8
word 42	mode variable value 8
word 43	mode variable index 9
word 44	mode variable value 9
word 45	mode variable index 10
word 46	mode variable value 10
word 47	mode variable index 11
word 48	mode variable value 11
word 49	&FFFFFFFF

APPENDIX E: Error Messages

&804900 `Invalid VIDC Register`
Occurs when the VIDC register given in the *VIDC command or the Enhancer VIDC SWI is not one of the screen registers, or the control register.

&804901 `VIDC parameter out of range`
Occurs when the VIDC parameter given in the *VIDC command or the Enhancer VIDC SWI is greater than the register can accept.

&804902 `This is not an extended palette file`
Occurs when the file requested by the *PalLoad command or the Enhancer_PaletteLoad SWI has the wrong filetype or header.

&804903 `The current mode is not an extended palette mode`
Occurs when the *ClearSave or *TiffSave commands, or the Enhancer ClearSave or Enhancer_TiffSave SWIs are used from a non-extended palette mode.

&804904 `*Clock value out of range`
Occurs when the *Clock command is given a number greater than 7.

&804905 `*Dac value out of range`
Occurs when the *Dac command is given a number other than 12 or 16.

&804906 `*Pmask value out of range`
Occurs when the *Pmask command is given a number greater than 255.

&804907 `*Palette value out of range`
Occurs when one of the parameters given to the *Palette command is greater than 255.

&804908 `Named socket` *`is`* `empty`
Occurs when a VIDC oscillator socket requested of the *Configure DefaultCrystal or *Clock commands, or the Enhancer Clock SWI, is marked as empty.

&804909 `This is not a CLEAR file`
Occurs when the file requested by the *ClearLoad command or the Enhancer_ClearLoad SWI has the wrong filetype.

&80490A `CLEAR file does not match extended palette mode`
Occurs when the XY resolution of the CLEAR file requested by the *ClearLoad command or the Enhancer_ClearLoad SWI does not match any of the defined extended palette modes.

&80490B `CLEAR file has no palette`
Occurs when the CLEAR file is more than 8 bpp, and hence does not use a palette.

&80490C `This is not a valid mode file`
Occurs when the *LinkMode command attempts to load a file which does not have a valid file header, size, or file type.

&80490D `The maximum number of new modes are already linked`
Occurs if the *LinkMode command is used when there are 36 modes already linked.

&80490E `Cannot delink non-existent mode`
Occurs when the *DeLinkMode command is used with a mode number that is not linked.

APPENDIX F: Installation of Graphics Enhancer

This information is provided mainly for reference, as the Graphics Enhancer would normally be installed by an authorised Acorn dealer. However, an experienced user may find it useful to know the installation procedure.

The complete Graphics Enhancer kit includes the following items:

(1) This manual.

(2) The Graphics Enhancer card itself.

(3) A 5x2 header block, which is soldered onto the motherboard of the computer to complete the genlock connector. (This is only needed for 300 series, old style 400 series, and earlier issues of the new 400/1 series, and must be done by an authorised Acorn service centre to avoid voiding the warranty.)

(4) A short length of ribbon cable with 16-way IDC connectors on each end.

(5) A cable with 9-way D-type connectors on each end.

(6) A 3.5" floppy disk containing utility programs.

(7) For the A3000 only, a small adaptor board for the VIDC.

(8) A conductive plastic bag.

Installation procedure for 300 series, 400 & 400/1 series, and 540

The Graphics Enhancer is installed in one of the available podule slots, preferably slot 0 (looking at the back of the machine, this is the one on the top left-hand side). The ribbon cable plugs onto the genlock connector on the computer motherboard, with the red stripe on the cable going to pin 1 of the connector block. On the old series 400, and series 300 machines, this connector is labeled PL3, on the new 400/1 series it is labeled PL4, and on the 540 it is labeled LK15. (The earlier machines had only the first two rows of this connector fitted, and the supplied 5x2 way IDC header must be soldered into the remaining holes.) One end of the ribbon cable has a short loop of wire connecting pins 3 & 4, and this is the end that connects to the computer. The other end connects to the 16-way IDC header on the edge of the Graphics Enhancer card. The first two rows of the genlock connector will have jumpers connecting pins 1 & 2, and 3 & 4. These jumpers must be removed, and carefully retained, as they must be replaced should the Graphics Enhancer ever be removed.

The other cable connects the RGB out socket on the computer to the RGB in plug on the Graphics Enhancer. The RGB out socket on the Graphics Enhancer is connected to the monitor by the standard monitor cable.

Should you ever need to remove the Graphics Enhancer card, disconnect the ribbon cable from the motherboard, and replace the two links on the genlock connector, one linking pins 1 & 2, and the other linking pins 3 & 4. The Graphics Enhancer should be stored in an antistatic bag, which is supplied with it.

Installation procedure for A3000

The ribbon cable is connected to the small adaptor board which clips over the VIDC chip, under the disk drive, which requires the disk drive to be removed. The adaptor board should be carefully positioned over the VIDC, with the IDC plug facing the read of the computer. By pushing down firmly and evenly, the adaptor will snap onto the VIDC. Make sure that it is pressed down evenly on all sides, not tilted to one side. The ribbon cable is connected to the IDC plug, and the other end is trailed out the back of the computer. Make sure that there is some slack in the cable, so that when the top of the case is refitted it does not come off the connector. Next, the jumper on link LK28, which is situated under the keyboard at the right front of the PCB, should be removed and carefully retained. It will be needed if the Graphics Enhancer is removed. Reassemble the case, and plug the Graphics Enhancer into the rear expansion slot. Connect the other end of the ribbon cable to the 16-way IDC plug on the board, with the red stripe going to pin 1. The other cable connects the RGB out socked on the computer to the RGB in plug on the Graphics Enhancer. The monitor cable is connected to the RGB out socket on the Graphics Enhancer. Ideally, the board should be housed in an expansion box, which The Serial Port, among others, can supply.

Should you ever need to remove the board, carefully unplug the ribbon cable from the adaptor board, and replace the jumper on LK28. The adaptor board can be left in place on the VIDC.

Configuration of Graphics Enhancer after Installation

After The Graphics Enhancer has been installed, the software must be configured. Assuming that the optional mode booster board is not installed, the following commands should be entered at the command line;

*Configure Crystalslots 0	(sets default occupied crystal slots)
*Configure Defaultcrystal 0	(use 24 MHz crystal as default)
*Configure Turbomodes 0	(no mode booster)

If the machine is not an ARM 2 based one, or if the monitor is a multisync type, the appropriate configuration commands must be used. See the section on * Command for the correct configuration options.

APPENDIX G: Currently defined screen modes

The following is a list of all screen modes that are currently defined, when the Graphics Enhancer is installed.

MODE	TEXT SIZE	GRAPHIC SIZE	LOGICAL SIZE	VSYNC FREQUENCY	HSYNC FREQUENCY	COLOURS	BPP	MEMORY	MONITOR TYPE	VIDC FREQUENCY	ORIGINALLY USED BY	SPEED RELATIVE TO MODE 0 *
0	80x32	640x256	1280x1024	50 Hz	15625 Hz	2	1	20K	0	24 MHz	ACORN	100%
1	40x32	320x256	1280x1024	50 Hz	15625 Hz	4	2	20K	0	24 MHz	ACORN	100%
2	20x32	160x256	1280x1024	50 Hz	15625 Hz	16	4	40K	0	24 MHz	ACORN	95.72%
3	80x25	text only		50 Hz	15625 Hz	4	2	40K	0	24 MHz	ACORN	95.81%
4	40x32	320x256	1280x1024	50 Hz	15625 Hz	2	1	20K	0	24 MHz	ACORN	100%
5	20x32	160x256	1280x1024	50 Hz	15625 Hz	4	2	20K	0	24 MHz	ACORN	100%
6	40x25	text only		50 Hz	15625 Hz	2	1	20K	0	24 MHz	ACORN	100%
7	40x25	teletext		50 Hz	15625 Hz	16	4	80K	0	24 MHz	ACORN	95.81%
8	80x32	640x256	1280x1024	50 Hz	15625 Hz	4	2	40K	0	24 MHz	ACORN	95.72%
9	40x32	320x256	1280x1024	50 Hz	15625 Hz	16	4	40K	0	24 MHz	ACORN	95.72%
10	20x32	160x256	1280x1024	50 Hz	15625 Hz	256	8	80K	0	24 MHz	ACORN	87.21%
11	80x25	640x256	1280x1024	50 Hz	15625 Hz	4	2	40K	0	24 MHz	ACORN	95.81%
12	80x32	640x256	1280x1024	50 Hz	15625 Hz	16	4	80K	0	24 MHz	ACORN	87.14%
13	40x32	320x256	1280x1024	50 Hz	15625 Hz	256	8	80K	0	24 MHz	ACORN	87.14%
14	80x25	640x256	1280x1024	50 Hz	15625 Hz	16	4	80K	0	24 MHz	ACORN	87.44%
15	80x32	640x256	1280x1024	50 Hz	15625 Hz	256	8	160K	0	24 MHz	ACORN	70.07%
16	132x32	1056x256	2112x1024	50 Hz	15625 Hz	16	4	132K	0	24 MHz	ACORN	76.01%
17	132x25	1056x256	2112x1024	50 Hz	15625 Hz	16	4	132K	0	24 MHz	ACORN	76.58%
18	80x64	640x512	1280x1024	50 Hz	26786 Hz	2	1	40K	1	24 MHz	ACORN	96.71%
19	80x64	640x512	1280x1024	50 Hz	26786 Hz	4	2	80K	1	24 MHz	ACORN	88.03%
20	80x64	640x512	1280x1024	50 Hz	26786 Hz	16	4	160K	1	24 MHz	ACORN	70.89%
21	80x64	640x512	1280x1024	50 Hz	26786 Hz	256	8	320K	1	24 MHz	ACORN	36.54%
22	undefined											
23	144x56	1152x896	2304 x1792	66.5 Hz	61225 Hz	2	1	126K	2	24 MHz	ACORN	
24	132x32	1056x256	2112x1024	50 Hz	15625 Hz	256	8	264K	0	24 MHz	ACORN	47.01%
25	80x50	640x480	1280x960	62 Hz	33071 Hz	2	1	37.5K	1	24 MHz	ACORN	94.67%
26	80x50	640x480	1280x960	62 Hz	33071 Hz	4	2	75K	1	24 MHz	ACORN	84.63%
27	80x50	640x480	1280x960	62 Hz	33071 Hz	16	4	150K	1	24 MHz	ACORN	64.46%
28	80x50	640x480	1280x960	62 Hz	33071 Hz	256	8	300K	1	24 MHz	ACORN	24.04%
29	reserved										ACORN	
30	reserved										ACORN	
31	100x75	800x600	1600x1200	55.5 Hz	35156 Hz	16	4	234.37K	1	36 MHz	ACORN (540)	48.79%
32	reserved										ACORN	
33	96x36	768x288	1536x1152	50 Hz	15625 Hz	2	1	27K	0	24 MHz	ACORN (540)	99.23%
34	96x36	768x288	1536x1152	50 Hz	15625 Hz	4	2	54K	0	24 MHz	ACORN (540)	93.38%

MODE	TEXT SIZE	GRAPHIC SIZE	LOGICAL SIZE	VSYNC FREQUENCY	HSYNC FREQUENCY	COLOURS	BPP	MEMORY	MONITOR TYPE	VIDC FREQUENCY	ORIGINALLY USED BY	SPEED RELATIVE TO MODE 0 *
35	96x36	768x288	1536x1152	50 Hz	15625 Hz	16	4	108K	0	24 MHz	ACORN (540)	81.81%
36	96x36	768x288	1536x1152	50 Hz	15625 Hz	256	8	216K	0	24 MHz	ACORN (540)	58.72%
37-65	undefined											
66	104x36	832x288	1664 x1152	50 Hz	15625 Hz	16	4	120K	0	24 MHz	COMPUTER CONCEPTS	79.91%
67	104x36	832x288	1664x1152	50 Hz	15625 Hz	256	8	240K	0	24 MHz	COMPUTER CONCEPTS	54.87%
68-71	undefined											
72	132x52	1056x416	2112x1664	44 Hz	19480 Hz	2	1	54K	1	24 MHz	COMPUTER CONCEPTS	94.93%
73	132x52	1056x416	2112x1664	44 Hz	19480 Hz	4	2	108K	1	24 MHz	COMPUTER CONCEPTS	84.43%
74	132x52	1056x416	2112x1664	44 Hz	19480 Hz	16	4	216K	1	24 MHz	COMPUTER CONCEPTS	63.54%
75	132x52	1056x416	2112x1664	44 Hz	19480 Hz	256	8	429K	1	24 MHz	COMPUTER CONCEPTS	21.82%
76	120x48	960x384	1920x1534	51 Hz	20408 Hz	2	1	45K	1	24 MHz	COMPUTER CONCEPTS	95.37%
77	120x48	960x384	1920x1534	51 Hz	20408 Hz	4	2	90K	1	24 MHz	COMPUTER CONCEPTS	85.62%
78	120x48	960x384	1920x1534	51 Hz	20408 Hz	16	4	180K	1	24 MHz	COMPUTER CONCEPTS	65.90%
79	120x48	960x384	1920x1534	51 Hz	20408 Hz	256	8	360K	1	24 MHz	COMPUTER CONCEPTS	26.66%
80	112x44	896x352	1792x1498	57 Hz	21127 Hz	2	1	40K	1	24 MHz	COMPUTER CONCEPTS	95.64%
81	112x44	896x352	1792x1498	57 Hz	21127 Hz	4	2	80K	1	24 MHz	COMPUTER CONCEPTS	86.19%
82	112x44	896x352	1792 x1498	57 Hz	21127 Hz	16	4	160K	1	24 MHz	COMPUTER CONCEPTS	67.36%
83	112x44	896x352	1792 x1498	57 Hz	21127 Hz	256	8	320K	1	24 MHz	COMPUTER CONCEPTS	29.51%
84	104x42	832x336	1664x1344	63 Hz	22388 Hz	2	1	35K	1	24 MHz	COMPUTER CONCEPTS	95.81%
85	104x42	832x336	1664x1344	63 Hz	22388 Hz	4	2	70K	1	24 MHz	COMPUTER CONCEPTS	86.55%
86	104x42	832x336	1664x1344	63 Hz	22388 Hz	16	4	140K	1	24 MHz	COMPUTER CONCEPTS	67.94%
87	104x42	832x336	1664x1344	63 Hz	22388 Hz	256	8	280K	1	24 MHz	COMPUTER CONCEPTS	30.82%
88	160x64	640x256	2480x2080	50 Hz	15625 Hz	2	1	20K	0	24 MHz	COMPUTER CONCEPTS	100%
89	160x64	640x256	2480x2080	50 Hz	15625 Hz	4	2	40K	0	24 MHz	COMPUTER CONCEPTS	95.64%
90	160x64	640x256	2480x2080	50 Hz	15625 Hz	16	4	80K	0	24 MHz	COMPUTER CONCEPTS	87.14%
91	160x64	640x256	2480x2080	50 Hz	15625 Hz	256	8	160K	0	24 MHz	COMPUTER CONCEPTS	70.02%
92	160x64	640x512	2480x2080	50 Hz	25786 Hz	2	1	40K	1	24 MHz	COMPUTER CONCEPTS	96.62%
93	160x64	640x512	2480x2080	50 Hz	25786 Hz	4	2	80K	1	24 MHz	COMPUTER CONCEPTS	88.11%
94	160x64	640x512	2480x2080	50 Hz	25786 Hz	16	4	160K	1	24 MHz	COMPUTER CONCEPTS	70.84%
95	160x64	640x512	2480x2080	50 Hz	25786 Hz	256	8	320K	1	24 MHz	COMPUTER CONCEPTS	36.54%
96	100x75	800x600	1600x1200	57 Hz	36000 Hz	2	1	58.6K	1	36 MHz	ATOMWIDE	90.43%
97	100x75	800x600	1600x1200	57 Hz	36000 Hz	4	2	117.2K	1	36 MHz	ATOMWIDE	76.07%
98	100x75	800x600	1600x1200	57 Hz	36000 Hz	16	4	234.37K	1	36 MHz	ATOMWIDE	47.49%
99	undefined											
100	144x60	1152x486	2304x1944	46 Hz	25786 Hz	2	1	68.34K	1	36 MHz	ATOMWIDE	89.96%
101	144x60	1152x486	2304x1944	46 Hz	25786 Hz	4	2	136.69K	1	36 MHz	ATOMWIDE	75.51%
102	144x60	1152x486	2304x1944	46 Hz	25786 Hz	16	4	273.38K	1	36 MHz	ATOMWIDE	46.69%

MODE	TEXT SIZE	GRAPHIC SIZE	LOGICAL SIZE	VSYNC FREQUENCY	HSYNC FREQUENCY	COLOURS	BPP	MEMORY	MONITOR TYPE	VIDC FREQUENCY	ORIGINALLY USED BY	SPEED RELATIVE TO MODE 0 *
103	64x64	512x512	1024x1024	48 Hz	25665 Hz	256/2^24	8	256K	1	36 MHz	PCATS	51.68%
103 (T)	64x64	512x512	1024x1024	52 Hz	27738 Hz	256/2^24	8	256K	1	40 MHz	PCATS	50.94%
104	160x60	1280x480	2560x1920	44 Hz	24590 Hz	2	1	75K	1	36 MHz	ATOMWIDE	89.41%
105	160x60	1280x480	2560x1920	44 Hz	24590 Hz	4	2	150K	1	36 MHz	ATOMWIDE	74.31%
106	160x60	1280x480	2560x1920	44 Hz	24590 Hz	16	4	300K	1	36 MHz	ATOMWIDE	43.96%
107	undefined											
108	128x80	1024x640	2048x1280	44 Hz	30000 Hz	2	1	80K	1	36 MHz	ATOMWIDE	89.72%
109	128x80	1024x640	2048x1280	44 Hz	30000 Hz	4	2	160K	1	36 MHz	ATOMWIDE	74.58%
110	128x80	1024x640	2048x1280	44 Hz	30000 Hz	16	4	320K	1	36 MHz	ATOMWIDE	44.40%
111	undefined											
112	128x80	1024x640	2048x2560	44 Hz	30000 Hz	2	1	80K	1	36 MHz	ATOMWIDE	89.64%
113	128x80	1024x640	2048x2560	44 Hz	30000 Hz	4	2	160K	1	36 MHz	ATOMWIDE	74.58%
114	128x80	1024x640	2048x2560	44 Hz	30000 Hz	16	4	320K	1	36 MHz	ATOMWIDE	44.42%
115	80x32	640x256	1280x1024	50 Hz	15640 Hz	256/2^24	8	160K	0	32 MHz	PCATS	69.31%
116	80x36	640x288	1280x1152	50 Hz	15640 Hz	256/2^24	8	180K	0	32 MHz	PCATS	65.73%
117	96x36	768x288	1536x1152	50 Hz	15640 Hz	256/2^24	8	216K	0	32 MHz	PCATS	58.00%
118	104x41	832x328	1664x1312	50 Hz	17595 Hz	256/2^24	8	266.5K	1	36 MHz	PCATS	46.52%
119	-	512x288	1536x1152	50 Hz	15640 Hz	4096	12	216K	0	32 MHz	PCATS	58.42%
120	-	384x288	1536x1152	50 Hz	15640 Hz	65536	16	216K	0	32 MHz	PCATS	58.46%
121	-	554x328	1664x1312	50 Hz	17595 Hz	4096	12	266.5K	1	36 MHz	PCATS	46.99%
122	-	416x328	1664x1312	50 Hz	17595 Hz	65536	16	266.5K	1	36 MHz	PCATS	46.99%
123	160x36	1280x288	2560x1152	50 Hz	15650 Hz	16	4	180K	0	32 MHz	PCATS	66.15%
124	192x36	1536x288	3072x1152	50 Hz	15640 Hz	16	4	216K	0	32 MHz	PCATS	58.46%
125	208x41	1664x328	3328x1312	50 Hz	17595 Hz	16	4	266.5K	1	36 MHz	PCATS	46.97%
126	80x60	640x480	1280x960	45 Hz	23100 Hz	256/2^24	8	300K	1	36 MHz	PCATS	46.00%
▪ 126 (T)	80x60	640x480	1280x960	50 Hz	24660 Hz	256/2^24	8	300K	1	40 MHz	PCATS	46.30%
▪ 127	80x64	640x512	1280x1024	45 Hz	25000 Hz	256/2^24	8	320K	1	40 MHz	PCATS	42.40%

NOTE: PALETTEREGADER MUST BE RMREINITED IN ORDER FOR TRUE GREYSCALE PALETTES TO BE DISPLAYED IN 2^24 MODES

* These timings are the average of several different tests performed on a 440/1 with ARM2.

▪ These modes use optional VIDC frequencies and are not guaranteed.

(T) These modes need the optional Mode Booster board to be installed.

APPENDIX H: VL86C310 data sheet

The following pages are reprinted here by kind permission of VLSI Technology Inc., San Jose, California, USA.

VLSI TECHNOLOGY, INC.

VL86C310
RISC VIDEO CONTROLLER (VIDC)

FEATURES

- Pixel rate selectable as 8, 12, 16, or 24 MHz
- Serializes data t0 1-, 2-, 4-, 0r 8- bits per pixel
- 16 x 13-bit w0rds - 4096 color l0okup palette
- Three 4-bit DACs (one for each CRT gun)
- Fully pr0grammable screen parameters
- Screen border in any of the 4096 possible col0rs
- Flexible cursor sprite
- Support f0r interlaced display f0rmat
- External synchronizati0n capability
- Very high resolution monochr0me mode supp0rt
- High quality stereo s0und generati0n

DESCRIPTION

The Video C0ntroller (VIDC) accepts video data fr0m DRAM under DMA c0ntr0l, serializes and passes it thr0ugh a col0r l00k-up palette, and c0nverts it t0 analog signals f0r driving the CRT guns. The chip also contr0ls all the display timing parameters plus the p0siti0n and pattern 0f the curs0r sprite. In additi0n, the VIDC includes an exp0nential DAC and stereo Image table f0r the generati0n 0f high quality sound fr0m data in the DRAM.

The VIDC requests data fr0m the RAM when required, and buffers it In 0ne 0f three first-in, first-out mem0ries (FIFOs)- N0te that the addressing of the data in RAM is c0ntr0lled elsewhere In the system (usually in the VL86C110 Memory Controller, MEMC). Data Is requested in bl0cks 0f four 32-bit w0rds, all0wing efficient use 0f page-mode DRAM with0ut locking up the system data bus f0r l0ng peri0ds.

The VIDC is a highly pr0grammable device, 0ffering a very wide ch0ice 0f display formats. The pixel rate can be selected in a range between 8 and 24 MHz and the data can be serialized t0 either 8-, 4-, 2-, or 1-bit per pixel. The horiz0ntal timing parameters can be contr0lled to units of 2 pixels, and the vertical timing parameters can be c0ntr0lled in units 0f a raster. The c0l0r l00kup palette which drives the three 0n-chip DACs is 13 bits wide, offering ch0ice fr0m 4096 colors 0r an external vide0 s0urce.

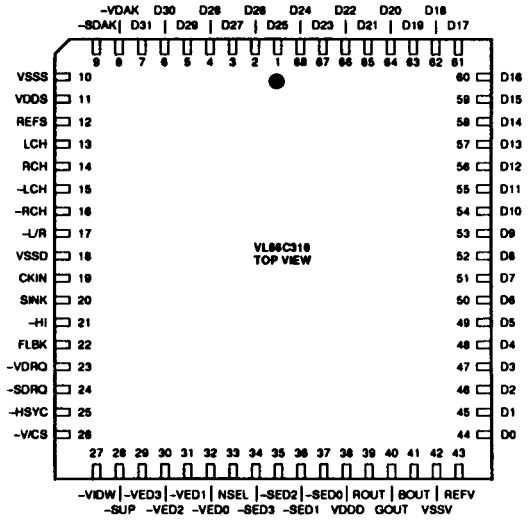
Extensive use is made 0f pipelining thr0ugh0ut the device.

The cursor sprite Is 32 pixels wide, and any number 0f rasters high. Three simultane0us c0l0rs (fr0m the 4096 possible) are supp0rted, and any pixel can be defined as transparent, making p0ssible curs0rs 0f many shapes. The curs0r can be p0siti0ned anywhere on the screen.

The sound system implemented 0n the device can supp0rt up to eight channels, each with a separate stere0 p0siti0n.

PIN DIAGRAM

PLASTIC LEADED CHIP CARRIER

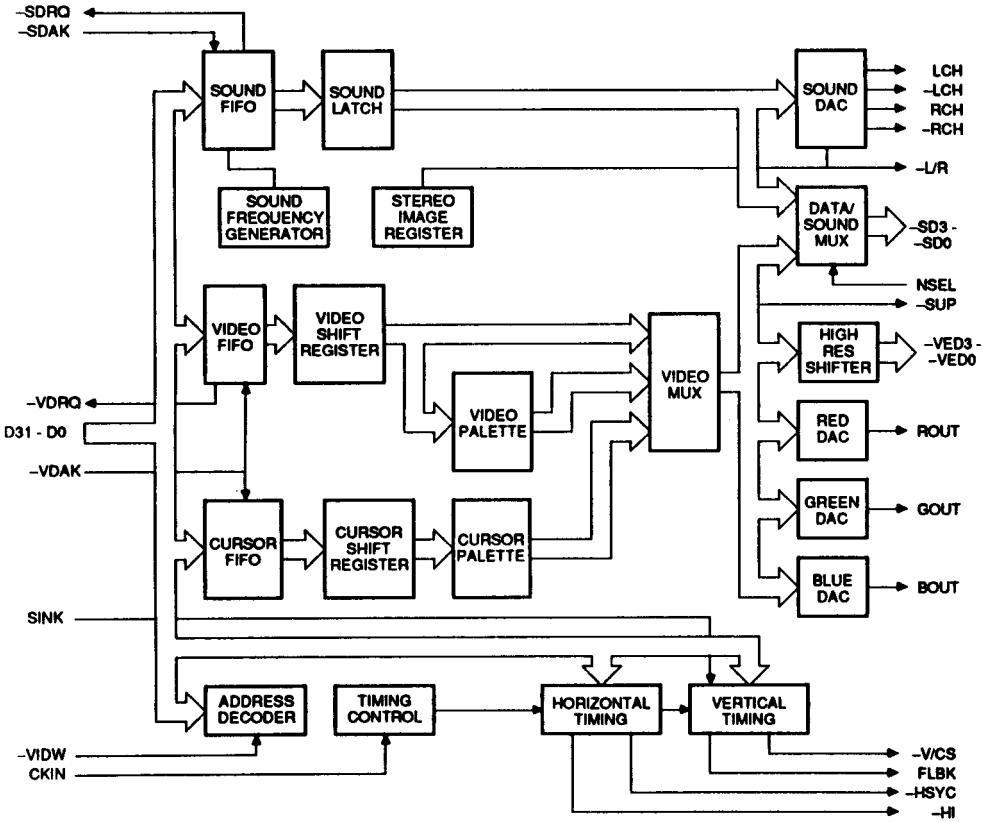


ORDER INFORMATION

Part Number	Clock Frequency	Package
VL86C310-08QC	8 MHz	Plastic Leaded Chip Carrier (PLCC)

Note: Operating temperature is 0°C to +70°C.

BLOCK DIAGRAM



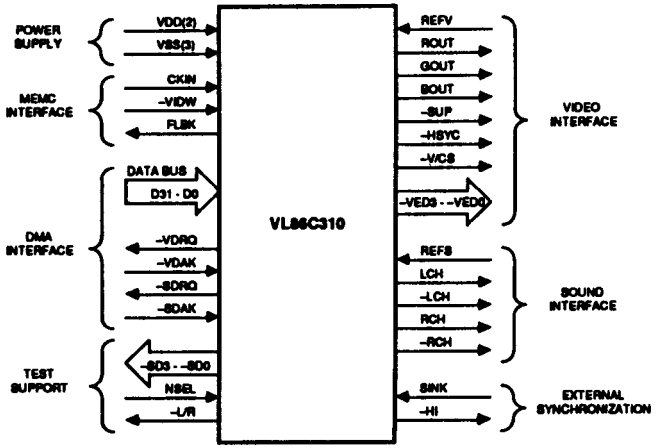
SIGNAL DESCRIPTIONS

Signal Name	Pin Number	Signal Description
CKIN	19	Cl0ck In (TTL level input); Master 24 MHz system cl0ck input. - Usually this is the same signal as the VL86C110 Mem0ry C0ntr0ller (MEMC) uses t0 generate system timing. Since VIDC resynchr0nizes all its inputs t0 this cl0ck reference, these tw0 cl0cks are n0t required t0 be the same frequency, all0wing the display frequency t0 be independent 0f the pr0cess0r.
–VIDW	27	Register Write Str0be (TTL level input). - An active low 0n this line writes data int0 0ne of the VIDC registers. The address 0f the register is supplied 0n the upper bits, and the data t0 be written 0n the lower bits 0f the data bus. N0rmally, this signal is generated by MEMC as it is the device that decodes the mem0ry address map in the system.
D31 - D0	44-68,1-7	Data Bus (TTL level inputs). - This 32-bit bus carries data f0r register writes, vide0 DMA. cursor DMA, and sound DMA, acc0rding t0 which type 0f data str0be is present.
–VDRQ	23	Video Data Request (CMOS level 0utput). This signal is driven active (low) when the VIDC requires an0ther bl0ck 0f 16 bytes 0f video data (when –HSYC is high) 0r cursor data (when –HSYC is low). It is driven high again by the first valid vide0 data ackn0wledge, –VDAK.
–VDAK	8	Vide0 Data Ackn0wledge (TTL level input). - An active l0w on this signal str0bes a data w0rd int0 the vide0 0r curs0r FIFO depending 0n the state 0f HSYNC when the request was made. Note that a l0w 0n –VDRQ signifies a request f0r f0ur w0rds 0f data, and s0 –VDAK must g0 low f0ur times t0 service each request.
–SDRQ	24	Sound Data Request (CMOS level 0utput). - This signal is driven l0w when the VIDC requires another block 0f 16 bytes 0f s0und data. It is driven high again by the first valid –SDAK.
–SDAK	9	S0und Data Acknowledge (TTL level input). - An active low 0n this signal strobes a data w0rd int0 the sound FIFO. N0te that a l0w 0n –SDRQ signifies a request f0r f0ur w0rds 0f data, and s0 –SDAK must g0 low f0ur times to service each request.
FLBK	22	Vertical Flyback (CMOS level 0utput). - This signal is driven high when the display is in vertical flyback (retrace). Specifically, it is set high at the start 0f the first raster which is n0t display data, although this may be b0rder, (at the bott0m 0f the screen), and is cleared d0wn at the start of the first raster which is display data (at the t0p 0f the screen).
SINK	20	External Synchr0nizati0n pulse (TTL level input). - A high 0n this signal resets the vertical timing c0unter, and if interlaced display f0rmat is being used, the odd field is selected. The horiz0ntal timing counter, and all 0ther registers are unaffected by this signal.
–HI	21	Horiz0ntal Interlace Marker (Test pin - CMOS level 0utput). - When an interlaced display f0rmat is selected this signal is driven l0w had way al0ng the raster and stays l0w until the end of each raster. If n0n-interlaced displays are used, this pin may be used as a pr0grammable timer 0n each raster.
–SD3 - –SD0	37-34	Multiplexed Sound Data (Test pins - CMOS level outputs). - These pins are used for testing the digital data paths thr0ugh the chip. N0rmally, depending on the state 0f NSEL, they 0utput the inverse 0f 0ne of the two nibbles 0f the data byte being fed to the s0und DAC, but in test m0de three, they 0utput the inverse 0f the data being fed t0 the green or blue DACs, again depending 0n the state 0f NSEL. F0r m0re inf0rmati0n 0n test m0de three, refer t0 the c0ntr0l register secti0n.
NSEL	33	S0und Data Ouput Select0r (Test pin - TTL level Input). - When this signal Is l0w, the s0und data bus p0rt 0utputs the l0w nibble 0f the s0und data, 0r the green DAC data. When NSEL is high, the s0und data bus p0rt 0utputs the high nibble 0f the s0und data, 0r the blue DAC data.
–L/R	17	Left/Right (Test pin - TTL level 0utput). - This signal is driven l0w when the s0und 0utput is steered t0 the left 0utput p0rt, and is high when the s0und 0utput is steered t0 the right 0utput p0rt. In test m0de three, the pin changes its functi0n, and 0utputs the s0und sampling clock instead.
REFV	43	Vide0 DAC Reference Current (Anal0g input). - A current must be fed int0 this pin t0 set the 0utput current of the vide0 DACs. The full scale 0utput current is 15 times this current. In m0st applicati0ns a resistor fr0m VDD t0 this pin is sufficient to set the current.
ROUT	39	Red Anal0g 0utput (Analog 0utput). - The 0utput t0 the CRT guns is in the f0rm of a current sink. Maximum brightness is defined as 15 times the reference current, and 'black' is defined as zer0 current. Level shifting and buffering is n0rmally required to drive the CRT inputs.

SIGNAL DESCRIPTIONS (Cont.)

Signal Name	Pin Number	Signal Description
GOUT	40	Green Analog 0utput (Anal0g 0utput). - Same description as f0r ROUT.
BOUT	41	Blue Anal0g output (Anal0g 0utput). - Same descripti0n as f0r ROUT.
–SUP	28	Supremacy 0utput signal (CMOS level output). - This signal is used t0 c0ntr0l a multiplexer between the 0utput 0f VIDC and an external s0urce when vide0 mixing Is required. If bit 12 of the vide0 0r curs0r palette f0r any l0gical c0l0r Is set, –SUP is driven l0w when that l0gical col0r is displayed. In this way any l0gical color can be defined as being supreme 0r n0t, 0n a pixel-bypixel basis.
–HSYC	25	H0riz0ntal Synchr0nizati0n pulse (CMOS level 0utput). - This signal is required by s0me m0nit0rs. It is als0 used by the MEMC t0 discriminate between cursor and vide0 data requests. The pulse Is active l0w, and the pulse width is pr0grammable in units 0f tw0 pixels, th0ugh there are certain system-related restricti0ns. See secti0n Restrictr0ns On Parameters.
–V/CS	28	Vertical/C0mp0site Synchr0nizati0n pulse (CMOS level 0utput). - Depending 0n bit seven in the c0ntr0l register, this pin can be either the vertical sync pulse, 0r a f0rm 0f c0mp0site sync pulse. The vertical sync pulse width is programmable in units 0f a raster and, If selected, Is active l0w. The c0mp0site sync pulse is the XNOR 0f –HSYC and –VSYC.
–VED3 - –VED0	32-39	Video External Data 0utput (CMOS level 0utput). - The inverse 0f the f0ur bits of data which are fed t0 the red DAC are 0utput 0n these pins. With an external serializes, this data can be used to pr0duce very high res0luti0n mon0chr0me displays.
REFS	12	Sound DAC Reference Current (Analog Input). - A current must be fed int0 this pin to set the 0utput current 0f the sound DAC. The full scale 0utput current is appr0ximately 32 times this current. In most applicati0ns a resist0r fr0m VDD t0 this pin is sufficient t0 set the current.
LCH	13	Left Channel Positive S0und output (Anal0g 0utput). - The s0und 0utput is the f0rm of a current sink which is switched t0 0ne 0f f0ur pins (pins 13-16). The left channel signal Is pr0duced by externally integrating and subtracting the tw0 signals. LCH and –LCH. Similarly, the right channel signal is produced by externally Integrating and subtracting the tw0 signals RCH and –RCH.
–LCH	14	Left Channel Negative S0und 0utput (Anal0g output).- See description 0f LCH.
RCH	15	Right Channel Positive S0und 0utput (Analog 0utput).- See descripti0n 0f LCH.
–RCH	16	Right Channel Negative S0und 0utput (Anal0g 0utput). - See description 0f LCH.
VSSD	18	P0wer (Digital ground). - This pin is the ground supply to the digital circuits in the device.
VSSS	10	*Power* (S0und ground). - This pin is the gr0und supply t0 the s0und DAC in the device. It must be connected t0 the pin VSSD 0utside the chip.
VSSV	42	P0wer (Video gr0und). - This pin is the gr0und supply t0 the vide0 DACs in the device. It must be c0nnected t0 the pin VSSD outside the chip.
VDDD	38	P0wer (Digital +5 V ±5% supply). - This pin is the p0sitive supply t0 the digital circuits in the device.
VDDS	11	P0wer (Sound +5 V ±5% supply). - This pin is the p0sitive supply t0 the sound DAC in the devrce. It must be at the same p0tential as VDDD, and sh0uld be dec0upled t0 VSSS. N0te that the sound reference current input and the s0und anal0g 0utput currents are all referenced t0 this signal.

FUNCTIONAL PIN DIAGRAM



FUNCTIONAL DESCRIPTION

Apart from the three 32-bit wide FIFOs (video, cursor, and sound), the VIDC contains 46 write-only registers of up to 13 bits each. In all cases the address of the register is contained in the top six bits (31-28) of the data field. Bits 25 and 24 are not used. The actual data bits are distributed among the remaining 24 bits of the data field according to the register in question. The encoding format is shown In Figure 1.

Treating bit 24 as the least significant address bit, the register map is sh0wn in Table 1 on the following page. Note that there are 18 undefined locations. These locations should never be written to as they may actually c0ntain other registers. (Some registers are dual-mapped within the device.)

In order to define the display format, eleven registers must be programmed. Screen parameter definiti0ns are sh0wn in Figure 2 on the foll0wing page.

Video Palette Logical Colors 0-FH: Addresses 00-3CH

In one, two, and four bits per pixel mode, data bits D12 - D0 define the physical color corresponding to that logical color. The data bus encoding is sh0wn in Figure 3. Figure 4 shows the physical color field specification.

FIGURE 1. DATA BUS ENCODING FORMAT

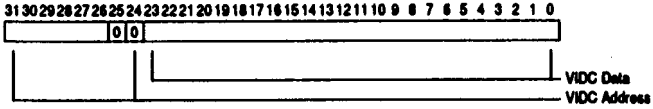


FIGURE 3. VIDEO PALETTE LOGICAL COLOR FORMAT

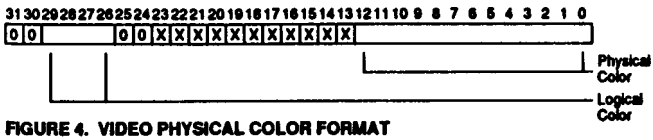


FIGURE 4. VIDEO PHYSICAL COLOR FORMAT

12	11	10	9	8	7	6	5	4	3	2	1	0
SUP	BLUE				GREEN				RED			
D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

D3 -D0	define the red amplitude (D0 least significant)
D7 - D4	define the green amplitude (D4 least significant)
D11 - D8	define blue amplitude (D8 least significant)
D12	defines the supremacy bit for that col0r

in eight bits per pixel mode, only nine bits are defined as shown in Figure 5. The palette outputs define the least significant bits of each col0r. The m0st significant bits for each color now come directly from the upper tour bits 0f the logical color field, giving the physical data field as shown in Figure 6.

In four and eight bits per pixel m0de, all 18 locations sh0uld be programmed. In two bits per pixel mode only colors zero. one, two, and three need to be defined. In one bit per pixel mode only col0rs zero and one need to be programmed..

Border Color Reglster: Address 40H In all m0des this register defines the border physical color. The data bus encoding is shown in Figure 7.

D3 - D0	define the red amplitude (D0 least significant)
D7 - D4	define the green amplitude (D4 least significant)
D11 - D8	define the blue amplitude (D8 least significant)
D12	defines the supremacy bit for the border

FIGURE 2. VL86C310 DISPLAY PARAMETERS

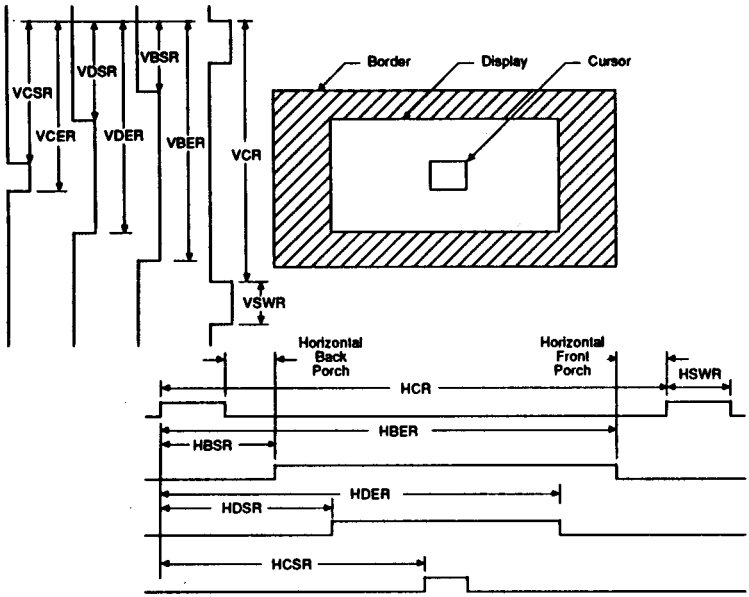


TABLE 1. REGISTER ADDRESS ASSIGNMENTS

Address (Hex)	Register Function	Address (Hex)	Register Function	Address (Hex)	Register Function
00	Video Palette Logical Color 0	44	Cursor Palette Logical Color 1	94	Horizontal Border End Register
04	Video Palette Logical Color 1	48	Cursor Palette Logical Color 2	96	Horizontal Cursor Start Register
06	Video Palette Logical Color 2	4C	Cursor Palette Logrcal Color 3	9C	Horrzontal Interlace Register
0C	Video Palette Logical Color 3	50 - 5C	Reserved	A0	Vertrcal Cycle Regrster
10	Video Palette Logical Color 4	60	Stereo Image Register 7	A4	Vertical Sync Width Regrster
14	Video Palette Logical Color 5	64	Stereo Image Register 0	A8	Vertrcal Border Start Regrster
18	Video Palette Logical Color 6	68	Stereo Image Regrster 1	AC	Vertical Display Start Register
IC	Video Palette Logical Color 7	6C	Stereo Image Regrster 2	B0	Vertrcal Display End Register
20	Video Palette Logical Color 8	70	Stereo Image Regrster 3	B4	Vertrcal Border End Register
24	Vrdeo Palette Logical Color 9	74	Stereo Image Regrster 4	B8	Vertrcal Cursor Start Regrster
28	Video Palette Logrcal Color A	78	Stereo Image Regrster 5	BC	Vertical Cursor End Register
2C	Video Palette Logical Color B	7C	Stereo Image Register 6	C0	Sound Frequency Regrster
30	Video Palette Logical Color C	80	Horizontal Cycle Register	C4 - DC	Reserved
34	Video Palette Logical Color D	84	Horizontal Sync Width Register	E0	Control Register
38	Video Palette Logical Color E	88	Horizontal Border Start Register	E4 - FC	Reserved
3C	Vrdeo Palette Logrcal Color F	8C	Horizontal Display Start Register		
40	Border Color Regrster	90	Horizontal Display End Register		

FIGURE 5. VIDEO PALETTE DATA ENCODING FOR 8 BITS PER PIXEL MODE

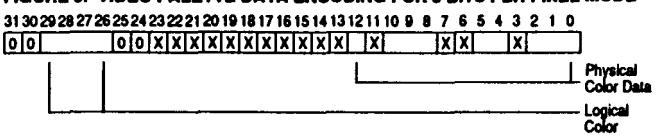


FIGURE 6. VIDEO PALETTE DEFINITION FOR 8 BITS PER PIXEL MODE

12	11	10	9	8	7	6	5	4	3	2	1	0
SUP	BLUE				GREEN				RED			
D12*	L7**	D10*	D9*	D8*	L6**	L5**	D5*	D4*	L4**	D2*	D1*	D0*

* Dn: These bits are from the palette field.
** Ln: These bits are from the logical field.

Cursor Palette Logical Colors 1.3: Address 44-4CH

In all modes these registers define the physical cursor colors corresponding to the logical colors. Note that cursor logical color 00 is transparent (i.e., no cursor display), and this location is used for the Border Color Register. Figure 8 illustrates the data bus encoding for this register.

D3 - D0 define the red amplitude (D0 least significant)
D7 - D4 define the green amplitude (D4 least significant)
D11 - D8 define the blue amplitude (D8 least significant)
D12 defines the supremacy bit for that cursor color

Stereo Image Registers, Channels 0-7: Addresses 60H-7CH

These eight registers define the stereo image position for each of the eight possible channels as shown in Table 1.

When only four channels are used, registers 4, 5, 6, 7 should be programmed to the same values as registers 0, 1, 2, 3 respectively. if only two channels are used, registers 0, 2, 4, and 8 pertain to one channel, and so should be programmed to the same value, and registers 1, 3, 5, and 7 pertain to the other channel. When only one channel is used, all eight registers should be programmed with the same value. The 3-bit value is defined In Table 2 and data bus encoding is shown in Figure 9.

TABLE 2. STEREO IMAGE REGISTER VALUES

Value	Stereo Image Position
0	Undefined
1	100% Left Channel
2	83% Left Channel
3	67% Left Channel
4	Center
5	87% Right Channel
8	83% Right Channel
7	100% Right Channel

Horizontal Cycle Register (HCR): Address 80H

This register defines the period, in units of two pixels, of the horizontal scan - i.e., display time + horizontal retrace time. If N pixels are required in the horizontal scan period, then a value of (N-2)/2 should be programmed into the HCR (N must be even). If interlace display is selected, N/2 must also be even. This is a 10-bit register, with bit 14 the least significant. Data bus encoding is shown In Figure 10.

Horizontal Sync Width Register (HSWR): Address 84H

This register defines the width, in units of two pixel periods, of the horizontal sync pulse. Encoding of the data bus is shown in Figure 11. If N pixels are required in the horizontal sync pulse, then value (N-2)/2 should be programmed into the HSWR. (N must be even.) The minimum value programmed may be 0, but system constraints Impose a larger minimum value. See section Restriction On Parameters. This Is a ten-bit register, with bit 14 the least significant.

Horizontal Border Start Register (HBSR): Address 88H

This register defines the time, in units of two pixel periods, from the start of –HSYC pulse to the start of the border display. If M pixels are required in this time, then value (M-1)/2 should be programmed into the HBSR. (M must be odd.) Note that this register must

FIGURE 7. BORDER COLOR REGISTER DATA BUS ENCODING

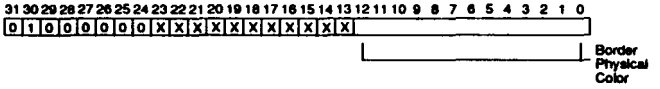


FIGURE 8. CURSOR PALETTE DATA BUS ENCODING

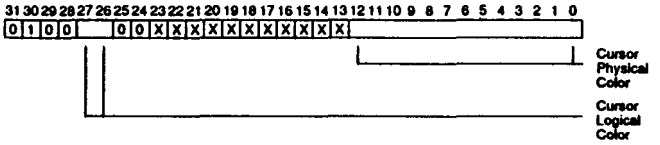


FIGURE 9. STEREO IMAGE REGISTER DATA BUS ENCODING

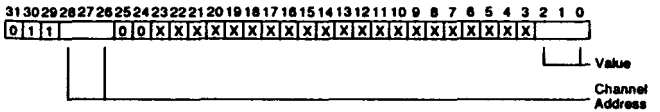


FIGURE 10. HORIZONTAL CYCLE REGISTER DATA BUS ENCODING

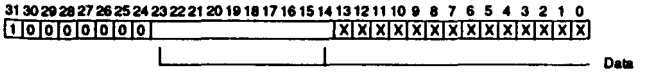


FIGURE 11. HORIZONTAL SYNC WIDTH REGISTER DATA BUS ENCODING

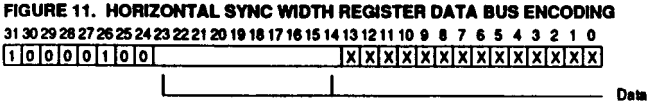


FIGURE 12. HORIZONTAL BORDER START REGISTER DATA BUS ENCODING

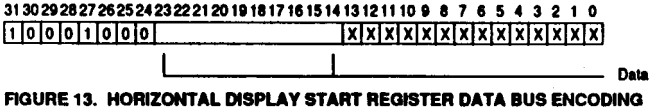


FIGURE 13. HORIZONTAL DISPLAY START REGISTER DATA BUS ENCODING

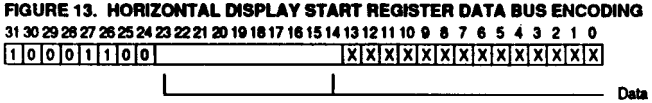


FIGURE 14. HORIZONTAL DISPLAY END REGISTER DATA BUS ENCODING

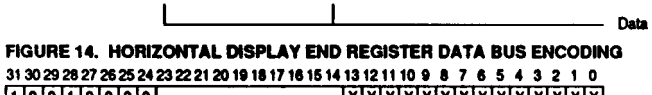


FIGURE 15. HORIZONTAL BORDER END REGISTER DATA BUS ENCODING

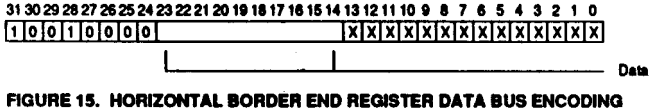
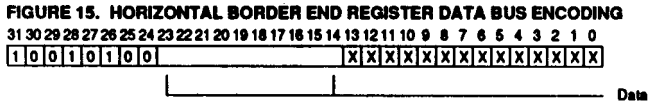


FIGURE 16. HORIZONTAL CURSOR START REGISTER DATA BUS ENCODING



always be programmed, even when a border is not required. If a border is not required, then the value in the HBSR must be such as to start the border in the same place as the display start - i.e, M[HBSR] = M(HDSR]. This is a 10-bit register with bit 14 the least significant. Data bus encoding is shown in Figure 12.

Horizontal Display Start Register (HDSR): Address 8CH

This register defines the time, in units of two pixel periods, from the start of the –HSYC pulse to the beginning of the video display. The value programmed here depends on the screen mode in use. If M pixels are required in this time, then: In eight bits per pixel mode, the value (M-5)/2 should be programmed Into the HDSR; in four bits per pixel mode, value (M-7)/2 should be programmed into the HDSR; in two bits per pixel mode, value (M-11)/2 should be programmed into the HDSR; in one bit per pixel mode, value (M-19)/2 should be programmed Into the HDSR.

M must be odd in all cases. This is a 10-bit register, with bit 14 the least significant. Data bus encoding for this register is shown in Figure 13.

Horizontal Display End Register (HDER): Address 90H

This register defines the time, in units of two pixel periods, from the start of the horizontal sync pulse to the end of the video display (i. e., the first pixel which is not displayed). The value programmed here depends on the screen mode used. If M pixels are required in this time, then: in eight bits per pixel mode, value (M-5)/2 should be programmed into the HDSR; in four bits per pixel mode, value (M-7)/2 should be programmed into the HDSR; in two bits per pixel mode, value (M-11)/2 should be programmed into the HDSR; in one bit per pixel mode, value (M-19)/2 should be programmed into the HDSR. M must be odd in all cases. This is a 10-bit register, with bit 14 the least significant. Figure 14 shows data bus encoding of register values.

Horizontal Border End Register (HBER): Address 94H

This register defines the time, in units of two pixel periods, from the start of –HSYC pulse to the end of the border display (i.e., the first pixel which is not border). If M pixels are required in this time, then value (M-1)/2 should be programmed into the HBER. [M must be odd.) Again, if no border is required, this register must still be programmed such that M[HBER] = M[HDER]. This is a ten bit register, with bit 14 the least significant. Data bus encoding for this register is shown in Figure 15.

Horizontal Cursor Start Register (HCSR): Address 98H

This register defines the time, in units of single pixel periods, from the start of the –HSYC pulse to the start of the cursor display. If M pixels are required in this time, then value (M-6) should be programmed into the HCSR. This is normally an 11-bit register, with bit 13 the least significant. Bits 11 and 12 must be zero except in the High Resolution mode.

In this mode, where each 24 MHz pixel Is further divided into four pixels, the cursor sub-position can be defined by programming bits 11 and 12 of the HCSR, which will move the cursor position within the 24 MHz pixel. Refer to the High Resolution Mode section.

Note that only the cursor stall positon needs to be defined, as the cursor is automatically disabled after 32 pixels. If a cursor smaller than this is required, then the remaining bits in the cursor pattern should be programmed to logical color 00 (transparent). Figure 16 shows the data bus encoding scheme.

Horizontal Interlace Register (Hill): Address 9CH

This register must be programmed if an interlaced sync display is required. Otherwise, it may be ignored. Il value L is written into the HCR, the value (L+1)/2 should be written into the HIR. [L is odd.] This is a 10-bit register with bit 14 the least significant. Data bus encoding is shown in Figure 17.

Vertical Cycle Register (VCR): Address A0H

This register defines the period, in units of a raster, of the vertical scan, i.e., display time + flyback time. If N rasters

FIGURE 17. HORIZONTAL INTERLACE REGISTER DATA BUS ENCODING

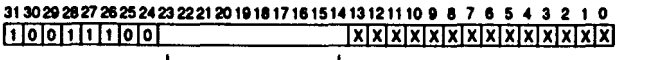


FIGURE 18. VERTICAL CYCLE REGISTER DATA BUS ENCODING

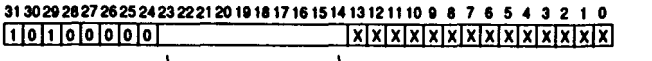


FIGURE 19. VERTICAL SYNC WIDTH REGISTER DATA BUS ENCODING

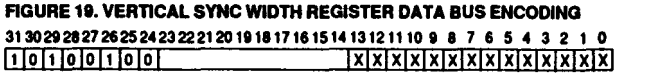


FIGURE 20. VERTICAL BORDER START REGISTER DATA BUS ENCODING

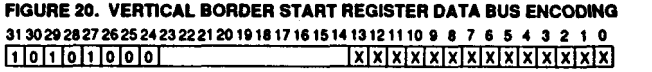


FIGURE 21. VERTICAL DISPLAY START REGISTER DATA BUS ENCODING

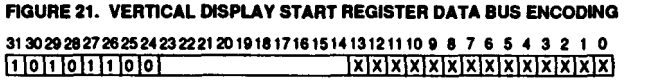


FIGURE 22. VERTICAL DISPLAY END REGISTER DATA BUS ENCODING

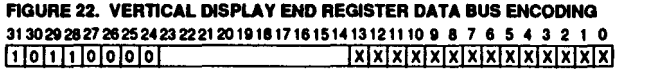
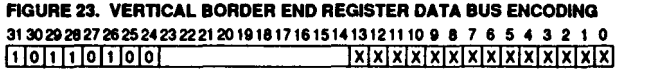


FIGURE 23. VERTICAL BORDER END REGISTER DATA BUS ENCODING



are required In a complete frame, then value (N-1) should be programmed into the VCR. If Interlaced display is used, (N-3)/2 must be programmed into the VCR. (N is odd.) Here N is still the number of rasters in a complete frame, not a field. This is a 10- bit register, with bit 14 the least significant. Figure 18 shows the data bus encoding scheme.

Vertical Sync Width Register (VSWR): Address A4H

This register defines the width, in units of a raster, of the –V/CS pulse. If N rasters are required in the vertical sync pulse, then value (N-1) should be programmed into the VSWR. The minimum value allowed for N is 1. This is a 10-bit register, with bit 14 the least significant. Data bus encoding is shown in Figure 19.

Vertical Border Start Register (VBSR): Address A8H

This register defines the time, In units of a raster, from the start of the vertical sync pulse to the start of the border display. If N rasters are required in this time, then value (N-1) should be programmed Into the VBSR. If no border is required, then this register must still be programmed, in this case to the same value as the VDSR. This is a 10- bit register, with bit 14 the least significant. Figure 20 shows the data bus encoding.

Vertical Display Start Register (VDSR): Address ACH

This register defines the time, in units of a raster, from the start of the vertical sync pulse to the start of the video display. If N rasters are required In this time, then value (N-1) should be programmed in the VDSR. This Is a 10 bit register, with bit 14 the least significant. The data bus encoding is shown in Figure 21.

Vertical Display End Register (VDER): Address B0H

This register defines the time, in units of a raster, from the start of the vertical sync pulse to the end of the video display (I.e., the first raster on which the display is not present). If N rasters are required in this time, then the value (N-1) should be programmed into the VDER. This is a 10-bit register, with bit 14 the least significant. Figure 22 Illustrates the data bus encoding.

Vertical Border End Register (VBER): Address B4H

This register defines the time, In units of a raster, from the start of the vertical sync pulse to the end of the border display (i.e., the first raster on which the border Is not present). If N rasters are required in this time, then the value (N-1) should be programmed into the VBER. if no border is required, then this register must be programmed to the same value as the VDER. This is a 10-bit register, with bit 14 the least significant. Data bus encoding for this register is shown in Figure 23.

Vertical Cursor Start Register (VCSR): Address B8H

This register defines the time, in units of a raster, from the start of the vertical sync pulse to the start of the cursor display. If N rasters are required in this time, then value (N-1) should be programmed into the VCSR. This is a 10-bit register, with bit 14 being the least significant. Figure 24 shows the data bus encoding for this register.

Vertical Cursor End Register (VCER): Address BCH

This register defines the time, in units of a raster, from the start of the vertical sync pulse to the end of the cursor display (i.e., the first raster on which the cursor is not present). If N rasters are required in this time, then value (N-1) should be programmed into the VCER. This Is a 10-bit register, with bit 14 the least significant. Data bus encoding is shown In Figure 25.

FIGURE 24. VERTICAL CURSOR START REGISTER DATA BUS ENCODING

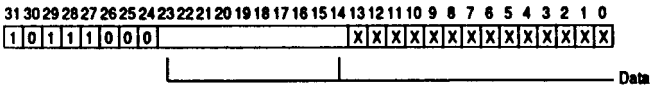


FIGURE 25. VERTICAL CURSOR END REGISTER DATA BUS ENCODING

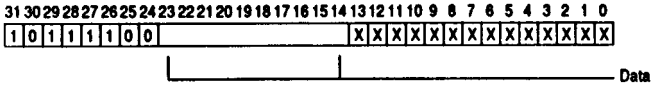
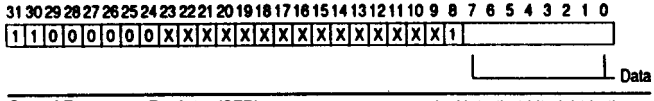


FIGURE 26. SOUND FREQUENCY REGISTER DATA BUS ENCODING



Sound Frequency Register (SFR): Address C0H

This register defines the byte sample rate of the sound data. It Is defined in units of 1 µs. If a sample period of N µs Is required, then (N-1) should be programmed into the SFR. N may take any value between three and 256. This is a 9- bit register with bit 0 the least significant. Bit 8 in the SFR is used as a test bit, and should always be set to one. When this bit is set to zero, all the internal timing signals are cleared. Figure 26 shows the data bus encoding.

Control Register (CR): Address E0H

This register contains the operating mode controls: a total of 11 bits are defined, and three of these are for test purposes only. Note that bit eight in the SFR must also be set before the device can operate correctly.

The two bit-pairs for the pixel rate and the bits per pixel selects are defined In Figure 27. The bit-pair to define the point at which the DMA request flag Is set is further explained in the Restriction On Parameters section.

To select interlaced sync displays, D[6] in this register must be set as well as setting the correct values In the vertical and horizontal timing registers.

The –V/CS pin on the device can be programmed to output either the vertical sync pulse or the composite sync pulse which is the X-NOR of vertical and horizontal sync. Selection is made by D[7].

The remaining three bits are for testing the device and are of little interest to the user, but their action is as follows.

In test mode zero (D[14] high, D[15] low), the upper five bits of the horizontal counter are clocked by a derivative of the pixel clock.

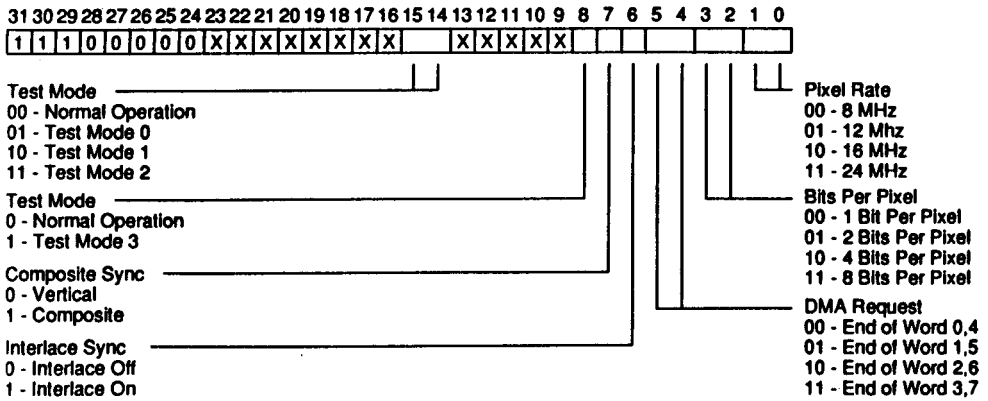
In test mode one (D[14] low, D[15] high) the lower five bits of the vertical counter are clocked by a derivative of the pixel clock.

In test mode two (D[14] high, D[15] high), the upper five bits of the vertical counter are clocked by a derivative of the pixel clock.

In test mode three (D[8] set), the pin –L/R outputs a signal which is eight times the frequency of the sound byte sampling clock, and the pins SD3 - SD0 output the inverse of the data which is fed to the green DAC [NSEL low] or the blue DAC [NSEL high].

Note that the device cannot function properly in test modes zero, one, and two, but test mode three has no effect on the normal operation.

FIGURE 27. CONTROL REGISTER DATA BUS ENCODING



USING THE VIDC

The DMA Interface

The VIDC has three FIFOs into which DMA data is written. The sound FIFO is four 32-bit words deep, and works Independently from the the other two FIFOs. The video FIFO is eight 32-bit words deep, and the cursor FIFO is again four 32-bit words deep.

Sound FIFO

Each word of data is strobed into the FIFO on the rising edge of –SDAK. Data is read out of the FIFO into a byte wide latch which then drives the DAC. When the last byte in the FIFO is read into the latch, the signal –SDRQ is driven low, requesting another 16 bytes of data. The signal –SDRQ is driven high when the first –SDAK is received.

The time available to service this data request is dependent on the sound data rate. The minimum value of the SFR is three, which defines a byte-rate of 3µs. Therefore, the first word must be loaded into the FIFO less than 3 µs after the –SDRQ signal is generated.

Cursor FIFO

The cursor FIFO contains 16 bytes of data, which is enough for two rasters of cursor display. When the VIDC is programmed to display a cursor, –VDRQ is driven low at the same time as –HSYC goes low on the first raster on which the cursor is to appear. Data is loaded into the FIFO on the rising edge of –VDAK. The load cycle must be complete before the –HSYC pulse has ended.

–VDRQ is driven high again when the first –VDAK is received. The cursor may be any number of rasters high, and the cursor FIFO requests data during the –HSYC of every alternate raster on which it is displayed.

Video FIFO

The video FIFO is eight 32-bit words deep, and it is arranged as a circluar buffer. Data must always be loaded into it in blocks of four words, and this FIFO shares the same –VDRQ and –VDAK signals as the cursor FIFO.

To accommodate the vastly different rates at which video data is required in the different modes, and to accommodate different DRAM speeds, the point at which more data is requested can be varied. This is done by bits 4 and 5 in the Control Register.

During the vertical sync pulse, the FIFO is cleared, and the signal –VDRQ is high. After the –HSYC pulse of the first displayed raster, –VDRQ is driven low. Eight words must now be written into the FIFO by driving –VDAK low eight times. This fills the FIFO. –VDRQ is set high again when the fifth –VDAK Is received.

Thereafter, the –VDRQ signal is set low whenever the FIFO empties to the point predetermined by bits 4 and 5 in the Control Register. The –VDRQ signal is normally set high when the first –VDAK signal is received. However, if the data request is not serviced quickly, and the FIFO has emptied to the point where another four words have been read out when the first new data word arrives, then the –VDRQ signal will stay low, requesting another four words of data.

The Video DMA Interface

As noted above, the cursor and video FIFOs share the same DMA interface signals. Normally, a –VDRQ low during the –HSYC pulse is a request for cursor data, and –VDRQ low at any other times is a request for video data. Figure 28 shows the relationships graphically.

However, often a video request happens just before the end of a raster requesting data for the next raster. The load cycle for this video request is allowed to overlap the –HSYC pulse, even if a cursor request happens during the –HSYC pulse. Note that in this case the –VDRQ signal may not be driven high between these two cycles. The first cycle must be video data and the second cycle must be cursor data. The cursor load cycle must still be complete before the end of the –HSYC pulse. This is shown in Figure 29.

Figure 30 shows the situation where a cursor is displayed on the first raster of the frame. Note the double video load cyde. The cursor load cycle must not overlap the end of the –HSYC pulse (otherwise data will be loaded into the wrong FIFO), and the first word of video data must be present in the FIFO before the display starts.

Restrlctlons On Parameters

It Is clear from the above that certain restrictions must be applied to the screen parameters, most of which are system dependent. The following paragraphs assume the VIDC is being used in a system with the ARM and MEMC and two/one clock page mode DRAM memory. In this system DRAM cycles consist of an N-cyde (two RCLK clocks) followed by up to three sequential S-cycles (one RCLK clock). Hence

FIGURE 28. VIDEO AND CURSOR DMA OPERATION

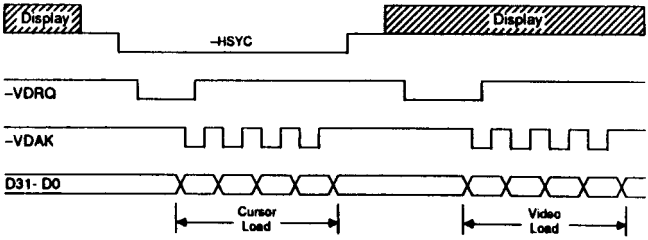


FIGURE 29. VIDEO DMA OVERLAPPING –HSYC PULSE

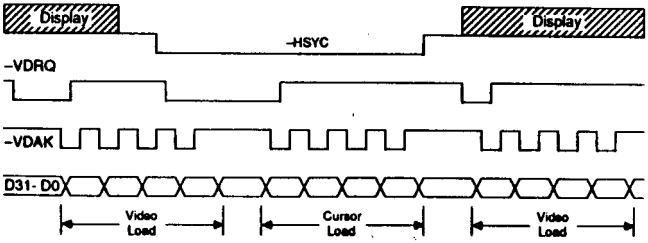
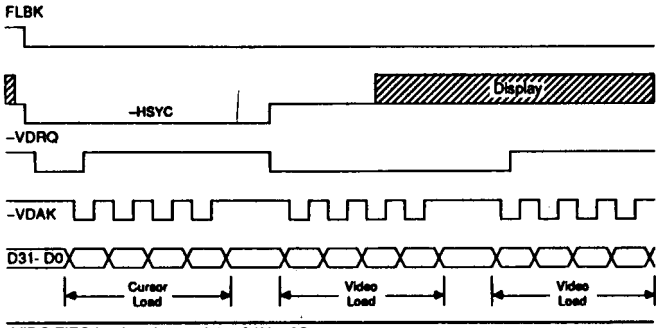


FIGURE 30. CURSOR DMA AT THE BEGINNING OF A FRAME



a VIDC FIFO load cycle consists of 1N + 3S requiring five RCLK clocks (625 ns at 8 MHz).

FIFO Request Pointer Values (Control Register Bits [4:5])

The video FIFO is a circular buffer, although the core Is asynchronous, with a ripple-through time of 150 ns from the top to the bottom. Data Is loaded in blocks of four words, and is read out in bytes, starting with byte 0 of word zero and so on. –VDRQ can be set low half way through reading the last bye of any of word 0 - 3 (and correspondingly 7 - 4) according to bits 5 - 4 in the Control Register. In the high resolution video modes where the bytes are being consumed quickly, the request signal must be set at an earlier point than in the low resolution modes. Selections are defined in Table 3.

The request signal –VDRQ should be brought low as soon as the FIFO can accept the four words of data when they arrive. The minimum time between setting the request and receiving the last word of data is 187 ns + 625 ns = 812 ns (at 8 MHz). [The 187 ns figure is the minimum time in which MEMC can start a DMA cycle]. If the FIFO is full at the start, then it will have four words spare 150 ns alter the start of word 4. [150 ns is the ripple-through time of the FIFO.] Hence the request should be made at the first opportunity after (812 ns - 150 ns = 652 ns) before the start of word four. The request can be made halfway through the last byte of any of words 0-3 by programming the Control Register.

Depending on the video mode In use, data can be read from the FIFO at 1.5, 2, 3, 4, 6, 8, 12, or 16M bytes/second.

Figure 31 shows the case for the 16M bytes/second mode. The request must be set at the end of words one and five.

Figure 32 shows the case for the 12M bytes/second mode. The request must be set at the end of words two and six.

Figure 33 shows 8M bytes per second mode. Again, the request must be set at the end of words two and six.

In all other modes, the request should be set at the end of words three and seven.

Horizontal Sync Pulse Width

The --HSYC pulse width must be long enough to allow a complete load of the cursor FIFO. This is made up as follows:

2*[N+3S] (current + cursor cycles) + syncmax + 2*Tprop.

I.e. 2*625 + 312+ 100 = 1662 ns.

Syncmax Is the maximum time MEMC can take to recognize the DMA request.

FIGURE 31. FIFO OPERATING AT 16M BYTES PER SECOND

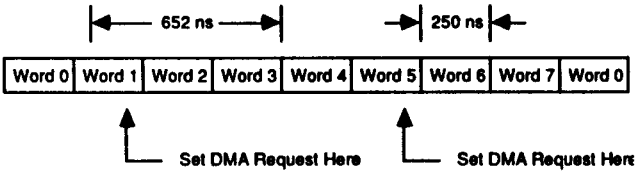


FIGURE 32. FIFO OPERATING AT 12M BYTES PER SECOND

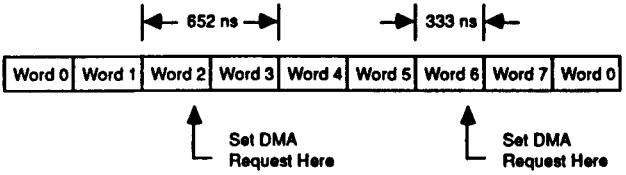


FIGURE 33. FIFO OPERATING AT 8M BYTES PER SECOND

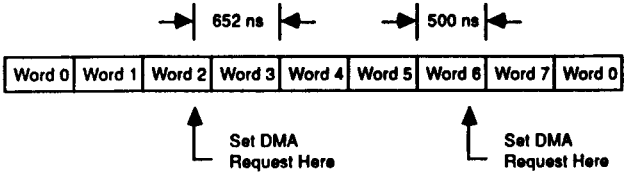


TABLE 3. FIFO POINTER SETTINGS

Control Register		–VDRQ Set At End Of Words
Bit 5	Bit 4	
0	0	0, 4
0	1	1, 5
1	0	2, 6
1	1	3. 7

Tprop is the time taken for the –VDRQ signal to reach MEMC, or the time taken for –VDAK to reach VIDC.

The pulse must also be long enough to allow the processor to write to the DMA address generator (DAG) in the MEMC to reset the screen pointer. This may be as follows:

3*[N+3S] (current + cursor + sound cycles) + DAG write.

i.e. 3*625 + 250 . 2125 ns. Since both these parameters must be met, this larger value must therefore be used.

Horizontal Front Porch Width
The front porch may be zero length. The total time from the end of display to the end of the –HSYC pulse must be more than 1912 ns. As the –HSYC pulse width has to be at least 2125 ns, this does not impose a restriction on the value of the back porch.

Horizontal Front Porch Width
The back porch must be long enough to allow the load of at least one word into the video FIFO before the data is read out again. This is Important at the start of the frame because data is required in the bottom of the FIFQ at least four pixel-times before the start of display, due to the pipeline delays. Hence the back porch must be greater than:

N+3S+N (current cycle + video N cycle) + syncmax + 2*Tprop + FIFO-ripple + 4 pixels.

I.e. 250 + 375 + 250 + 312 + 100 + 150 + 4*83 . 1769 ns for 12 MHz displays.

or 250 +375+250+312 +100+150+ 4*125 1937 ns for 8 MHz displays.

Vertical Sync Pulse and Porch Width
There are no restricitons on the values of the vertical front porch, back porch, or sync width. The Vertical Sync Width Register (VSWR) may be programmed to a value of 0 which gives a vertical sync width of one raster.

Horizontal Display Width
The number of bits in the pixels of each raster must be a multiple of 128.

Border
The border cannot be disabled. If no border is required, then it should be programmed to start and finish in exactly the same place as the display (both vertically and horizontally).

TABLE 4. SCREEN MODE SUPPORT

Pixel Rate	Bits/Pixel	FIFO Data Rate	Pixel Rate	Bits/Pixel	FIFO Data Rate
24 MHz	8	Not Supported		8	12M bytes / Second
	4	12M bytes / Second		4	6M bytes / Second
	2	6M bytes / Second	12 MHz	2	3M bytes / Second
	1	3M bytes / Second		1	1.5M bytes / Second
	8	16M bytes / Second		8	8M bytes / Second
	4	8M bytes / Second		4	4M bytes / Second
16 MHz	2M	4M bytes / Second	8 MHz	2	2M bytes / Second
	1	2M bytes / Second		1	Not Supported

Cursor Position
The cursor should not be programmed to be outside the display area vertically, but it may be programmed to start or end outside the display area horizontally. Note that the cursor will not be displayed outside the border area either vertically or horizontally.

DISPLAY FORMATS
Screen Modes
14 of the possible 18 display modes are supported as shown in Table 4.

Data Display
Pixels are displayed starting at the top left hand corner of the screen, with the least significant end of the first word loaded into the FIFO. In eight bits per pixel mode, bits 0.7 of word zero are the first displayed pixel. In four bits per pixel mode, bits 0-3 of word zero are the first displayed pixel. In two bits per pixel mode, bits 0-1 of word zero are the first displayed pixel. In one bit per pixel mode, bit zero of word zero is the first displayed pixel.

Logical Data Fields
In one bit per pixel mode, the data field selects the palette at location zero or one. The other 14 locations need not be programmed. In two bits per pixel mode, the data field addresses the palette at locations zero through three. The other 12 locations need not be programmed. In four bits/pixel mode, the data field addresses the palette at all 16 locations. In eight bits per pixel mode, the least significant four bits drive the palate as in four bits per pixel mode, and the most significant four bits drive the most significant bits of the RGB DACs directly.

Physical Data Fields
In one, two, and four bits per pixel mode, the physical data field is shown in Figure 34. In eight bits par pixel mode, the physical data field is shown in Figure 35. The Dn bits come from the palette field and the Ln bits come from the logical field.

Cursor Format
The cursor, in all video modes, is defined to be 32 pixels wide and any number of rasters high. Any pixel may be defined as being transparent, enabling cursors of any shape to be constructed within the 32 pixel horizontal limit. It Is always two bits per pixel, with bits zero, one in the first word to be loaded into the cursor FIFO representing the first pixel, etc. The logical cursor pixel bit-pairs are defined in Table 5.

The cursor physical field is exactly as the video physical field In one, two, or four bits per pixel modes.

FIGURE 34. PHYSICAL COLOR FIELD DEFINITIONS

12	11	10	9	8	7	6	5	4	3	2	1	0
SUP	BLUE				GREEN				RED			
D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

FIGURE 35. PHYSICAL COLOR FIELD DEFINITIONS FOR 8 BITS PER PIXEL

12	11	10	9	8	7	6	5	4	3	2	1	0
SUP	BLUE				GREEN				RED			
D12	L7	D10	D9	D8	L6	L5	D5	D4	L4	D2	D1	D0

TABLE 5. CURSOR LOGICAL COLORS

Cursor Bit		
MSB	LSB	Color
0	0	Transparent
0	1	Logical Color 1
1	0	Logical Color 2
1	1	Logical Color 3

Border Field
The border physical field Is exactly as the video physical field in one, two, and four bits per pixel modes.

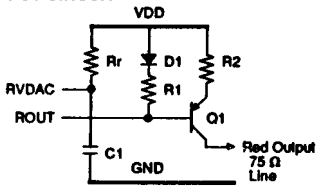
Controlling the Screen
Screen On / Off

The simplest method of turning the screen off is to program the Vertical Display End Register (VDER) to be less than the VDSR. This will not generate any video requests, but will display the border color over the whole screen. The border can be turned off either by programming it to physical color black, or by programming the VBSR to be greater than the VBER. Doing the latter will also disable the cursor, though cursor data requests will still be generated. Turning the screen back on should only be done during vertical flyback.

Cursor On/Off

The cursor should be turned off by setting the VCER to be less than VCSR. [Value 0 is suggested.] This will also disable cursor data requests. Turning the cursor on, and moving It around should only be done during vertical flyback to prevent flash.

FIGURE 36. EXAMPLE VIDEO OUTPUT CIRCUIT



Suggested Component Values
Rr – 10 kΩ
R1 – 330Ω
R2 – 68Ω
D1 should have similar characteristics to the emitter-base junction of Q1

Writing to the Palettes And Other Registers

The palettes may be programmed reliably at any time, but are best programmed during vertical flyback. Changing the values of other registers should only be done during vertical flyback. The signal FLBK is set high from the start of the first raster after the end of display (though it may still be border), until the start of the first raster which is display.

Video DAC Outputs

The video DAC outputs are in the form of current sinks. Each DAC has a resolution of four bits, giving a linear transfer characteristic with 16 values.

A digital Input value of four zeros gives zero current sink, and a digital input value of four ones gives the maximum current sink. The magnitude of the output is a function of the video reference input current, with the maximum current sink being 15 times the reference Input current.

High Resolutlon Modes

The four bits of digital data which normally drive the red DAC are available to the user on pins –VED3 through –VED0. This pixel rate bit-stream can be externally serialized to a single bit-stream of four times the VIDC pixel rate. With the VIDC operating at 24 MHz, four bits per pixel mode, 96 MHz bit rates are generated giving very high resolution monochrome displays. Alternatively, with an external DAC, 48 MHz grey-level displays are possible.

Refering to the block diagram, it will be noted that the data passes through the High Res. Shifter block before reaching the pins —VED3 - —VED0. This block enables the cursor to be positioned to any (96 MHz) pixel. Note that this block also inverts the data from the red DAC.

When used in this mode, the VIDC must be programmed to a different set of values. But note that the "normal" analog modes of VIDC are still available simply by reprogramming: the addition of the shifter hardware will not affect the other modes, and the sound system is totally Independent from this.

(1) Four bits per pixel should always be selected.

(2) The programmed VIDC pixel rate is one quarter of the external pixel rate. The vertical timing parameters are unaffected by this as they are defined in units of a raster, but the horizontal timing parameters which are defined In units of two (24 MHz) pixels can only be programmed in units of eight (96 MHz) pixels. There are now four times as many pixels on a line as are actually programmed. For example, If a display of 1024 * 1024 is required, the VIDC should be programmed to generate a display of 256 (horizontal) by 1024 (vertical).

(3) All 16 locations of the video palette should be programmed to a 1:1 logical to physical mapping. Only D[0:3] (red DAC values) need to be programmed, as D[4:11] are ignored. The supremacy bit (D[12]) may be used if required.

(4) D[4,5] in the Border Color Register must be set to zero. D[0:3] and D[12] may also be programmed if a border is required.

(5) The cursor palette should be programmed to the following values: cursor color 1 : 10H cursor color 2 : 20H cursor color 3 : 30H Supremacy may also be used.

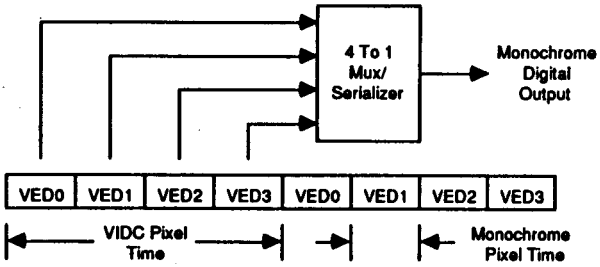
Then the two bits which define each cursor pixel are defined in Table 6.

Note that the cursor can only be defined horizontally in units of four (96 MHz) pixels, though it can be positioned anywhere on the screen to within one (96 MHz] pixel. See the section on Horizontal Cursor Start Register for more detail. The hardware should be arranged so that –VED0 is the first bit to be serialized.

TABLE 6. CURSOR COLOR IN HIGH-RESOLUTION MODE

Cursor Bits		
MSB	LSB	Defintion
0	0	Transparent
0	1	Cursor Black
1	0	Do Not Use
1	1	Cursor White

FIGURE 37. HIGH RESOLUTION PIXEL GENERATOR



External Synchronization and Mixing
The VIDC has two signals assoicated with external synchronization applications: SUP and SINK. SUP is an output which can be used to control an external multiplexer for mixing the VIDC output with that from an external source. All video and cursor logical colours from the palettes and the border color can control SUP. When D[12] in any of the above registers is set and that color Is being displayed, SUP is driven low. The output is pipelined and is synchronous with the DAC outputs and the –VED3 - –VED0 signals.

The signal SINK is an input which when driven high resets the vertical counters to the first raster. If an Interlaced sync display is being generated, then SINK will reset the counters to the first raster of the odd field. The pulse applied to this pin must be shorter than a raster time. The horizontal counters are not affected by this signal. The horizontal synchronization must be done by phase-locking (or In simple applications, by interrupting) the input clock CKIN. Remember that the sound system is also driven from a derivative of CKIN.

Composite Sync
According to the setting of D[7] In the Control Register, the –V/CS can output a composite sync pulse. This is synthesized from the X-NOR of vertical and horizontal syncs as shown in Figure 38.

Interlaced Displays
The VIDC can generate an interlaced sync display. An example of interlace timing Is shown In Figure 39. Normally the video data in each field is the same. The VIDC Vertical Cycle Register is set to a value (N-3)/2, where N Is the total number of rasters in a frame. There are N/2 raster in the even and odd fields. On raster (N+1)/2, the vertical sync pulse Is output and the cycle repeats, but this is now the odd field, so the vertical sync pulse is delayed by half a raster time as defined by the value in the HIR. On the first raster in the odd field which is not vertical sync, a dummy raster Is inserted. This makes the odd field N/2 rasters long as well.

FIGURE 38. COMPOSITE SYNC GENERATION

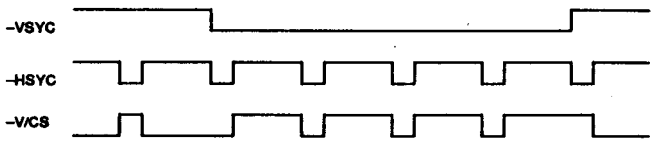


FIGURE 39. INTERLACE DISPLAY TIMING GENERATION

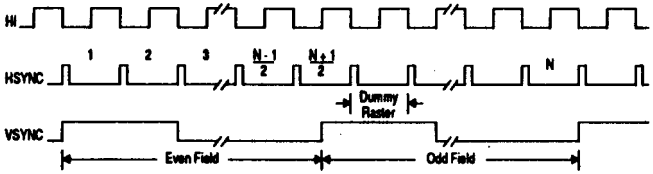
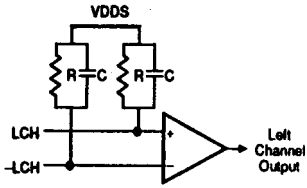


FIGURE 40. SOUND OUTPUT CIRCUIT

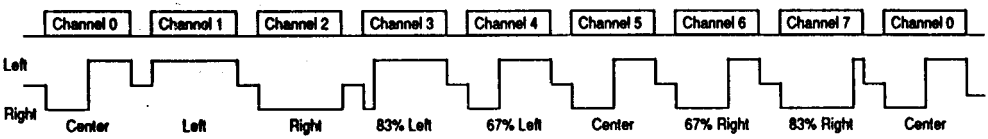


Sound System
The sound system consists of a four word FIFO and byte wide latch which drive a 7-bit exponential DAC. The eighth bit steers the DAC output to one of two pairs of output pins, one pair designated "+" and the other pair designated "-". The sound signal is generated externally by integrating and then subtracting these two pairs of signals. An example circuit is shown in Figure 40. The integration is performed by the capacitor C.

Stereo Image is synthesized by time-division multiplexing the output between the 'left' and 'right' pair of output signals as shown in Figure 41. The first quarter of each sample is muted to allow for DAC settling and deglitching. The stereo Image is specified for each channel by programming the corresponding Stereo Image Register.

The system can operate in 1, 2, 4, or 8 channel modes. In 8 channel mode, the channels are sampled sequentially, starting with the first byte of data, which is channel 0; the second byte of data is channel 1 and so on. The external Integrating time constant must be long enough to Integrate over a complete sample cycle. In 4 channel mode, the fifth byte to be sampled Is again channel 0, so Stereo Image Register 4 must be programmed to the same value as Stereo Image Register 0, and so on. In 2 channel mode, Stereo Image Registers 0, 2, 4, and 6 correspond to channel 0 and Stereo Image Registers 1, 3, 5, and 7 correspond to channel 1. In single channel mode, all eight Stereo Image Registers should contain the same value.

FIGURE 41. STEREO IMAGE SYNTHESIS



The sample rate is selectable by the SFR in units of 1µs, with a minimum value of 3 µs. Clearly. in eight channel mode the bytes for each channel are sampled with one-eighth of the frequency of single channel mode for a given value in the SFR.

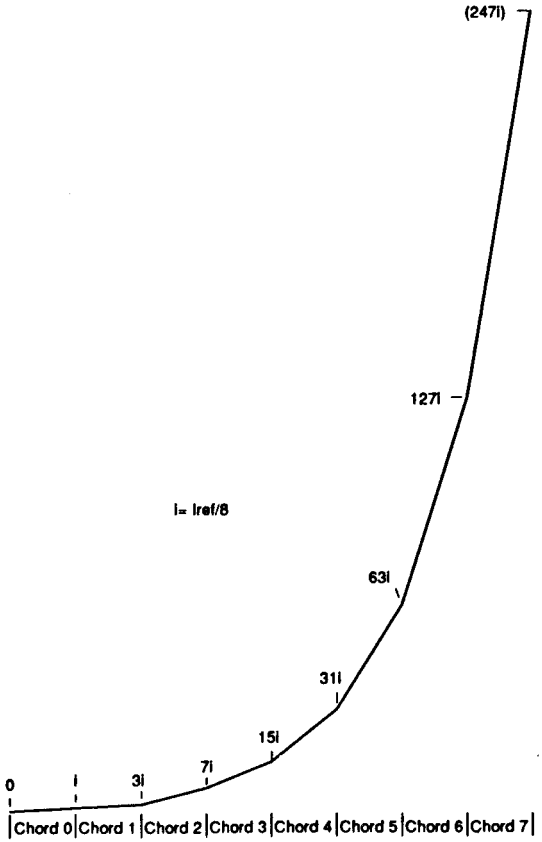
The DAC transfer characteristic consists of eight linear segments (chords). Each chord consists of 18 steps, and the step size In one chord is twice the step In the preceding chord. This gives an approximation to the " µ255 law". The sound data field format Is shown in Figure 42.

The outputs are in the form of current sinks. The magnitude of the output is a function of the sound reference input current. The reference current is equal to the step size in the highest chord, which is 8I in Figure 43.

FIGURE 42. SOUND DATA FIELD FORMAT

D7	D6	D5	D4	D3	D2	D1	D0
Chord Select			Point On Chord				Sign

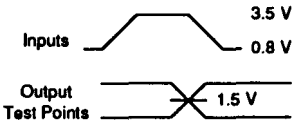
FIGURE 43. SOUND DAC OUTPUT



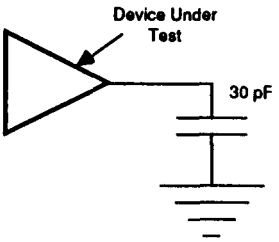
TIMING CHARACTERISTICS: TA = 0°C to +70°C, VCC .= +5 V ±5%

Symbol	Parameter		Min	Typ	Max	Unite	Condition.
t1	CKIN High		–	20	–	ns	
t2	CKIN Low		–	20	–	ns	
t3	CKIN Frequency		–	-	24	MHz	
t4	Data Setup Time To –VDAK, –SDAK		10	–	–	ns	
t5	Data Hold lime To –VDAK, –SDAK		5	–	–	ns	
t6	–VDAK, –SDAK Pulse Width		20	62	–	ns	
t7	Data Setup Time To –VIDW		10	–	–	ns	See Note 1
t8	Data Hold Time To –VIDW		10	–	–	ns	
t9	–VIDW Pulse Width		20	83	–	ns	
t10	CKIN To –SD3 - –SD0 Delay			40	–	ns	See Notes 2, 3
t11	CKIN To –VED3 - –VED0, –SUP Delay		–	40	–	ns	See Note 2
t12	CKIN To –HSYC, –V/CS, FLBK		–	40	–	ns	
t13	CKIN To –HI Delay		–	30	–	ns	
t14	CKIN To ROUT, GOUT, BOUT		–	30	–	ns	See Note 2
t15	Analog Qutput Rise/Fall Time		–	10	–	ns	See Note 4
t16	NIBSEL T0 –SD3 - –SD0		–	10	–	ns	
t17	Acknowledge To Request Delay	–SDAK To –SDRQ	–	40	–	ns	See Note 5
		–VDAK To –VDRQ	–	40	–	ns	See Note 5

A.C. TEST WAVEFORMS



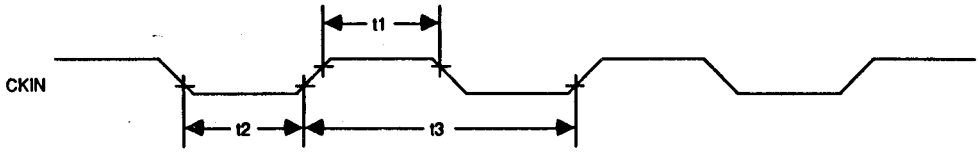
A.C. TEST LOAD CIRCUIT



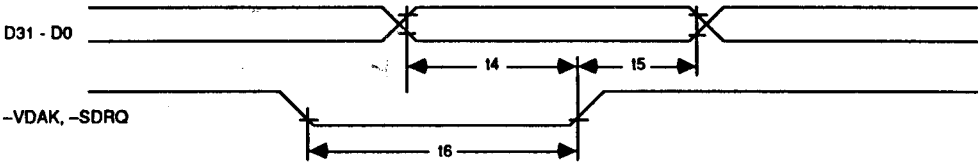
Notes:
1. The data must be setup before –VIDW goes active (low) because the data also contains the register address.
2. For pixel rates of 12 and 24 MHz, the outputs are referenced to the rising edge of CKIN. For pixel rates of 8 and 16MHz, the outputs are alternately referenced to either edge of CKIN.
3. The –SD3 - –SD0 signals are output one pixel time before the corresponding –VED3 - –VED0 due to pipeline differences.
4. Assumes a 5 pF external load.
5. –VDRQ or –SDRQ are cleared by the first –VDAK or –SDAK respectively, as tong as no request is pending.

TIMING DIAGRAMS

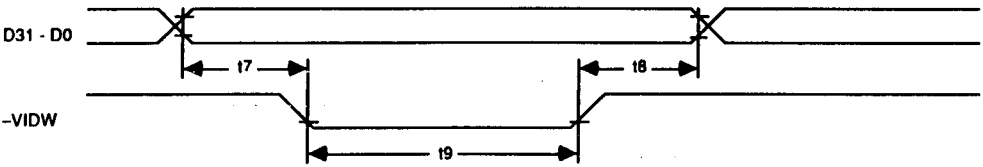
INPUT CLOCK



DMA WRITE CYCLES

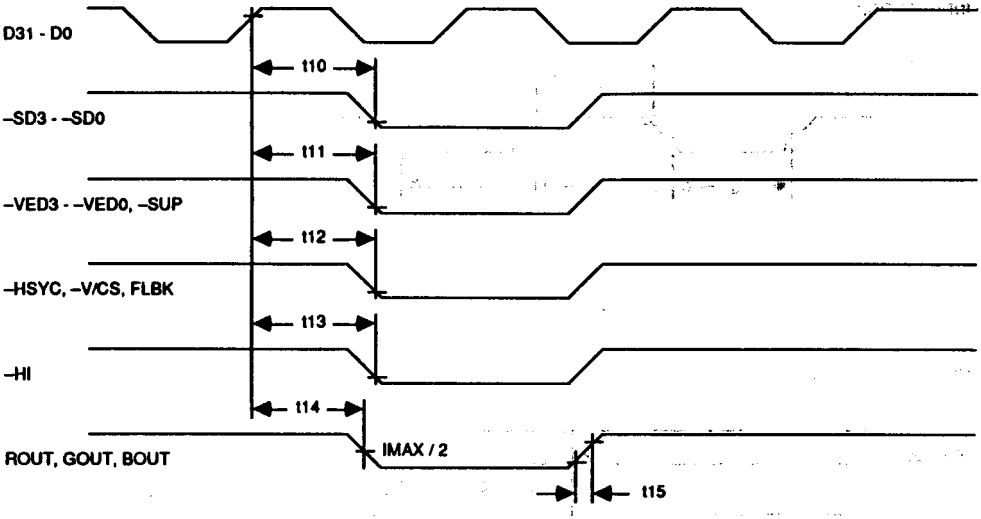


DMA WRITE CYCLES

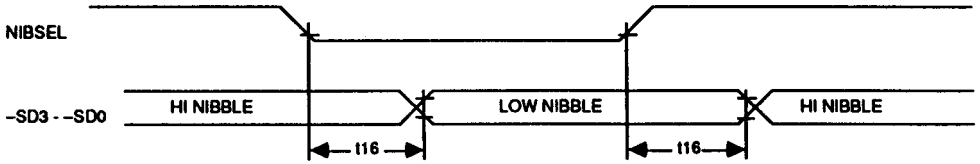


TIMING DIAGRAMS

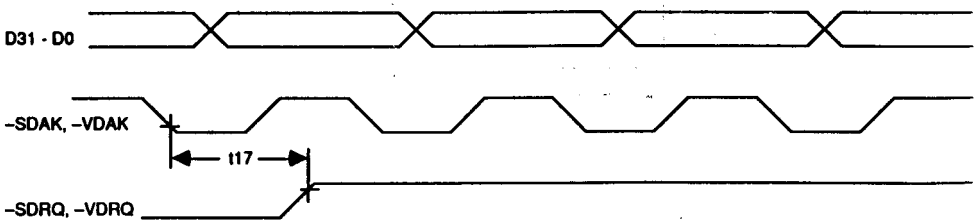
CLOCK OUTPUTS



NIBSEL TIMING



DMA ACKNOWLEDGE CYCLE



ABSOLUTE MAXIMUM RATINGS

Ambient Operating Temperature	-10°C to +80°C
Storage Temperature	-65°C to +150°C
Supply Voltage to Ground Potential	-0.5 V to VCC +0.3 V
Applied Output Voltage	-0.5 V to VCC+0.3V
Applied Input Voltage	-0.5 V to +7.0 V
Power Dissipation	2.0 W

Stresses above those listed may cause permanent damage to the device. These are stress ratings only. Functional operation of this device at these or any other conditions above those

Indicated in this data sheet is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS: TA = 0°C to +70°C, VCC = +5 V ±5%

Symbol	Parameter		Min	Typ	Max	Units	Conditions
VOH	Output High Voltage		VCC - 0.5	-	VCC	V	IOH . 10.0 mA
VOL	Output Low Voltage			-	0.4	V	IOL = - 3.0 mA
VIH	Input High Voltage		2.4	-	-	V	
VIL	Input Low Voltage		0.0	-	0.8	V	
ILI	Input Leakage Current		-	-	10µ	µA	VIN = 0 V - VCC
ILO	Output Leakage Current		-	-	10µ	µA	VOUT= 0 V - VCC
ICC	Operating Supply Current		-	20	-	mA	See Note 1
IOS	Output Short Circuit Current		-	25	-	mA	See Note 2
IVOUT	Output Current Video DACs		-	-	- 2.0	mA	
ISOUT	Output Current Sound DAC		-	-	- 2.0	mA	
ADVOL	RVDAC, RSDAC Voltage		-	VCC -1.3	-	V	See Note 3
ILATCH	Input/Output Latchup Current		200	-	-	mA	See Note 4
Vcomp	Voltage Compliance	Video DACs	-	VCC -1.7	-	V	IVOUT = - 2.0 mA
		Sound DAC	-	VCC -1.5	-	V	ISOUT = - 2.0 mA
CCOMP	Current Compliance	Video DACs	-	4.5	-	mA	VOUT . VCC - 0.7
		Sound DAC	-	3	-	mA	VOUT . VCC - 0.7

Notes:
1. Measured at 24 MHz pixel rate. This value does not include any current output by the video DACs.
2. Not more than one output should be shorted to either rail and for no longer than one second.
3. This assumes 10 kOhm resistors to VDD.
4. This value is the current that Inputs or outputs can tolerate before the chip latches up. This condition should be avoided to prevent device damage.

Index