

Apple][VGA – User's manual

Apple][VGA

VGA adapter board for the Apple][,][+ and //e

User's manual

This card allows you to enjoy an extremely sharp picture while working with your Apple 2, which is achieved by connecting it to a VGA Computer Monitor instead of a standard Television.

I. Hardware installation

- | | |
|---------------------|---|
| 1. Apple][and][+ | 1 |
| 2. Apple //e | 1 |
| 3. First run | 2 |

II. Screen mode selection

- | | |
|--------------------------------------|---|
| 1. Manual configuration | 3 |
| 2. Using the Softswitches from BASIC | 4 |
| 3. Finding the card's slot | 5 |

III. Troubleshooting

IV. Technical reference

1. Specifications
2. Adjustment
3. Circuit description
4. Schematic diagram

I. Hardware installation

1.1 Apple][and][+

The Apple][VGA emulates a PAL color card, so the same restriction applies: The card needs to be installed in slot 7.

Step-by-step instructions:

1. Set all the “][mode” jumpers to select Apple][mode.
2. Set HSYNC to – and VSYNC to +.
3. Connect the VGA output cable.
4. Connect the video input wire to the pin labeled “Video IN”.
5. Connect the other end to the modulator connector (the 4-pin molex connector near the composite video output jack) on the Apple][motherboard.
6. Insert the card into slot 7.
7. Route the VGA output cable through one of the openings in the back of the Apple][.

1.2 Apple //e

The Apple //e does not support the PAL color card anymore, therefore the installation gets more difficult. The missing signals need to be grabbed from the AUX slot and the card can be installed in any slot, including slot 3, because the ROM is not needed.

1. Remove J1 and J2 to set Apple //e mode.
2. Set HSYNC to – and VSYNC to +.
3. Remove the card in the AUX slot.
4. Insert the AUX slot adapter into the AUX slot.
5. Insert the 80-col card.
6. Connect the ribbon cable from the AUX slot adapter to the header labeled “//e AUX”. Align the red wire with the “pin 1” mark on the PCB.
7. Remove the plastic cover from opening 5 or 6 and install the VGA plug there.
8. Connect the other end of the VGA cable to the card.
9. Insert the card into the desired slot.

1.3 First run

After installation of the card, start a first test run. If possible, leave the lid open, because you might need to tweak some settings on the card.

Power up the apple and your monitor. You should get a crisp text display. Start a game and adjust brightness and contrast of the monitor until it fits your needs.

If possible, adjust the picture size and shape. If not, tweak the “Sync polarity” jumpers. There is no general rule. There are four possible combinations and each may be better or worse than any other. Try all of them to find the best settings.

Reboot the Apple and get a screen full of text. Use a screwdriver to adjust the pot labeled “TEXT CONT”, to the desired brightness of the text. This will not affect graphics. Do not touch the capacitor trimmers labeled 7M and 14M. These are pre-aligned and shall not need any tweaking, they are not active in text mode anyway.

Now close the lid, put your monitor on the top and you are done!

II. Screen mode selection

2.1 Manual screen mode selection

The VGA adapter board has a variety of display modes:

1. Monochrome
2. Block coloring
3. Stripe coloring with color blurring
4. Stripe coloring without color blurring
5. NTSC TV emulation with color blurring
6. NTSC TV emulation without color blurring

Sometimes modes are switched automatically:

In TEXT mode, Monochrome displaying is forced.

In LORES, Block coloring is selected automatically.

In HIRES and Double HIRES, mode 5 will be selected by default. This behavior can be changed using the video mode soft switches.

Type `CALL-151` to enter the monitor.

`C0S0` sets the default mode.

Here is a listing how to set all the screen modes:

Monitor	BASIC	Mode	sharpness	Color generation
<code>C0S0</code>	<code>49280+s*16</code>	5	like NTSC TV - blurry	low bandwidth color
<code>C0S1</code>	<code>49281+s*16</code>	6	like NTSC TV - blurry	high bandwidth color
<code>C0S2</code>	<code>49282+s*16</code>	3	high	low bandwidth color
<code>C0S3</code>	<code>49283+s*16</code>	4	high	high bandwidth color
<code>C0S8</code>	<code>49288+s*16</code>	1	very high	no color

Note: replace "S" with slot number+8 and "s" with the slot number

Try all of these modes on your favorite graphic programs (i.e. games)

Find out what you do or do not like.

2.2 Using the softswitches from BASIC

Screen modes can also be selected from BASIC.

`POKE 49288+SLOT*16,0` selects monochrome mode.

This enables you to write a simple screen mode selection program:

```
NEW
100 SLOT=7 : REM Insert the correct slot
110 DIM MODES(7)
120 MODES(1)=49288+SLOT*16
130 MODES(3)=49282+SLOT*16
140 MODES(4)=49283+SLOT*16
150 MODES(5)=49280+SLOT*16
160 MODES(6)=49281+SLOT*16
170 PRINT CHR$(4);"PR#3"
200 HOME
210 PRINT "Apple][VGA HIRES SCREEN MODE SELECTION PROGRAM"
220 PRINT "(c) Ferdinand Meyer-Hermann 2008"
230 PRINT: PRINT "Apple][VGA in Slot",SLOT;
240 PRINT "Mode 1: Monochrome"
250 PRINT "Mode 3: Stripy display with low-bandwidth color"
260 PRINT "Mode 4: Stripy display with high-bandwidth color"
270 PRINT "Mode 5: NTSC TV emulation with low-bandwidth color (default)"
280 PRINT "Mode 6: NTSC TV emulation with high-bandwidth color"
290 PRINT:PRINT:PRINT
300 PRINT "Your choice:";
310 INPUT C%
320 IF C%=2 THEN 200
330 IF C%<1 THEN 200
340 IF C%>6 THEN 200
350 POKE MODES(C%),0
360 END
```

You can also add a `POKE` command to the beginning of your BASIC programs, to automatically select the correct screen mode.

But note while playing around with screen modes that a `RESET` does not set the settings back to default. You will need to power cycle the Apple for this purpose. It may be a good idea to set the default mode in your `HELLO` program.

2.3 Finding the card's slot

There is a good way to avoid that you have to enter the slot number into the program's source code or typing it every time the program is run:

If the Apple][VGA's ROM space is selected, it forces the high bit to zero. If the screen is cleared, (in text mode), a PEEK command would read the last data from the video scanner, which is \$A0 if the screen is cleared. But the card forces the high bit to zero, so this becomes \$20. This signature is unique to the Apple][VGA and can thus be used to identify it.

Note that another card could have the high bit reset in various locations, but it is very unlikely that a card's firmware never has the high bit set. So testing the whole ROM should be safe.

If the card is in slot 3 and the 80-col firmware is enabled, the probe would fail, so it is a good idea to switch to 40-col mode during the test:

```
10 PRINT CHR$(4);PR#0:HOME
20 FOR SLOT=1 TO 7
30 FOR A=0 TO 255 : REM Scan every byte of the ROM
40 IF PEEK(49152+SLOT*256+A)>127 THEN 80:REM high bit set?
50 NEXT A
60 GOTO 110:REM The whole ROM has the high bit reset
70 REM Found the Apple][VGA
80 NEXT SLOT
90 PRINT "Didn't find the Apple][VGA card. Sorry..."
100 END
```

This program can be added to the one above, so that it will automatically detect the card and display the selection menu then.

III. Technical reference

4.1 Specifications

Video

Parameter	Mode	VGA	NTSC equivalent	Unit
Video sampling clock	all,NTSC	28.63638	14.31818	MHz
Color carrier	all,NTSC	7.15909	3.5794545	MHz
Video sampling clock	all,PAL	28.5	14.25	MHz
Color carrier	all,PAL	7.125	3.5625	MHz
Luma bandwidth -3dB	monochrome, stripy color	>20	>10	MHz
	LORES	3.5	1.75	MHz
	TV emulation	6.5	3.25	MHz
Chroma bandwidth -3dB	LORES	14	7	MHz
	HIRES, high bandwidth	14	7	MHz
	HIRES, low bandwidth	2	1	MHz

Timing

Parameter	NTSC Apple	PAL Apple	Unit
Number of lines	524	624	
H sync	31.4	31.25	kHz
H total	31.85	32	us
H sync width	4.47	4.49	us
H active video	19.56	19.65	us
H blanking	38.6	38.6	%
V sync	59.92	50.08	Hz
V total	16.69	19.96	ms
V sync with (except][, Rev 7)	0.255	0.256	ms
V active video	12.23	12.29	ms
V blanking	26.7	38.5	%
total blanking	55	62.2	%

4.2 Alignment

The TV emulation circuitry contains two traps for 7 and 14 MHz, which need to be tuned exactly to the color carrier frequency. Although these are pre-aligned, they may need to be adjusted to completely get rid of the vertical stripes.

The first thing to do is to force the card to TV emulation mode in LORES, so that a simple LORES test pattern can be used. This is done by shorting Pins 6 and 7 of U8A at C9. If you boot your Apple and enter LORES graphics, it will be interpreting LORES as DHIRES. Now set up a test screen with one half in LORES color 3 (magenta) and the other half in LORES color 5 (gray). Let it warm up. Now adjust the 7MHz trap for minimum stripes in the magenta area and the 14MHz trap for minimum stripes in the gray area. Both settings interact to some extent, so it might be required to repeat the alignment.

4.3 Circuit description

The circuit can be split in many parts, which will be described in order of importance.

1. Timing, scanning and sync generation
2. Video shifting
3. RAM access
4. Color decoding and video output
5. Control logic

4.3.1 Timing, scanning and sync generation

THE MAIN TIMING GENERATOR

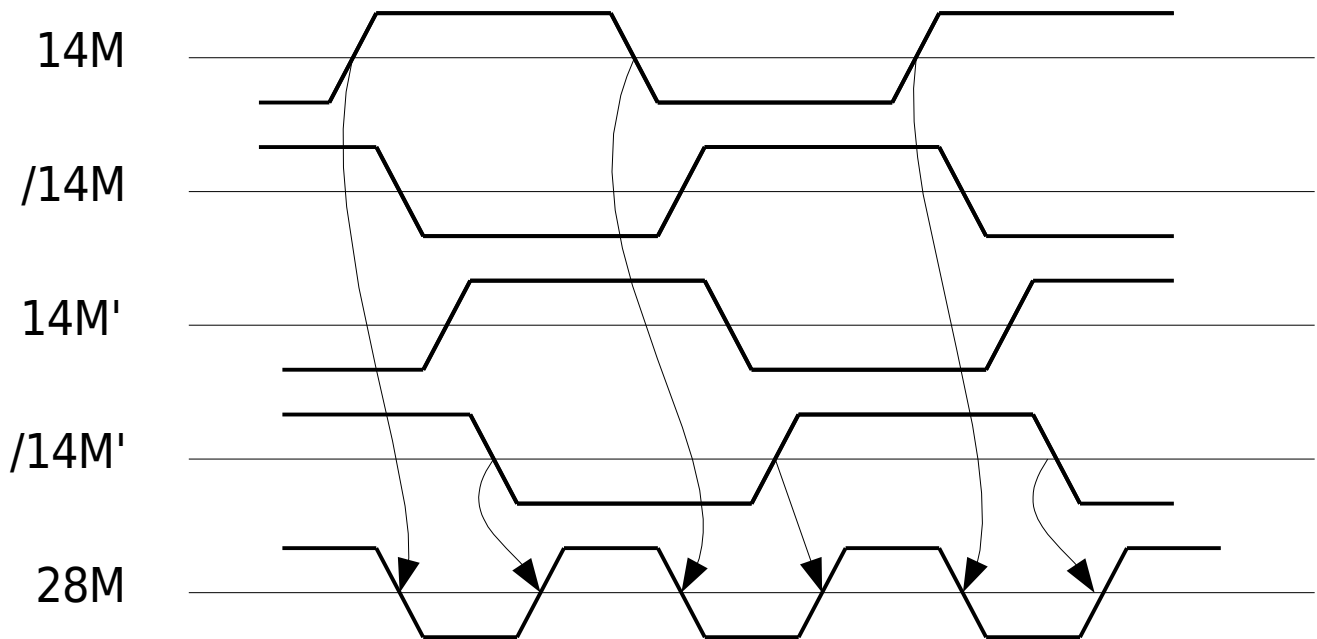


Fig 4.1: Generating 28M

U2C makes 14M symmetrical.

The 14M signal has a fairly high rise and fall times. If the signal was left as it is, it only be above the threshold for a short time.

If this happened the inverted 14M would high for a long time and low for a short time. assume high is +5V and low 0V. The average would be above 2.5V, maybe 3.5V. This is fed back via R19 to the input charging C1. If C1 is charged, the threshold is moved to lower voltage levels. So 14M is moved to the level required to get a symmetrical /14M and 14M'. But why is this done?

14M is inverted three times, and then XORed with itself. XORing a signal with its complement would cause a continuous high level. But because the triple inversion causes delay, the inputs of the XOR gates cycle through the states 01,11,10,00 and so on (see diagram). The both 11 and 00 cause the output to be low. The output cycles twice for every 14M cycle. The frequency has been doubled! This is why 14M needs to be symmetrical.

This signal is used to clock the video output shift registers and in color decoder timing.

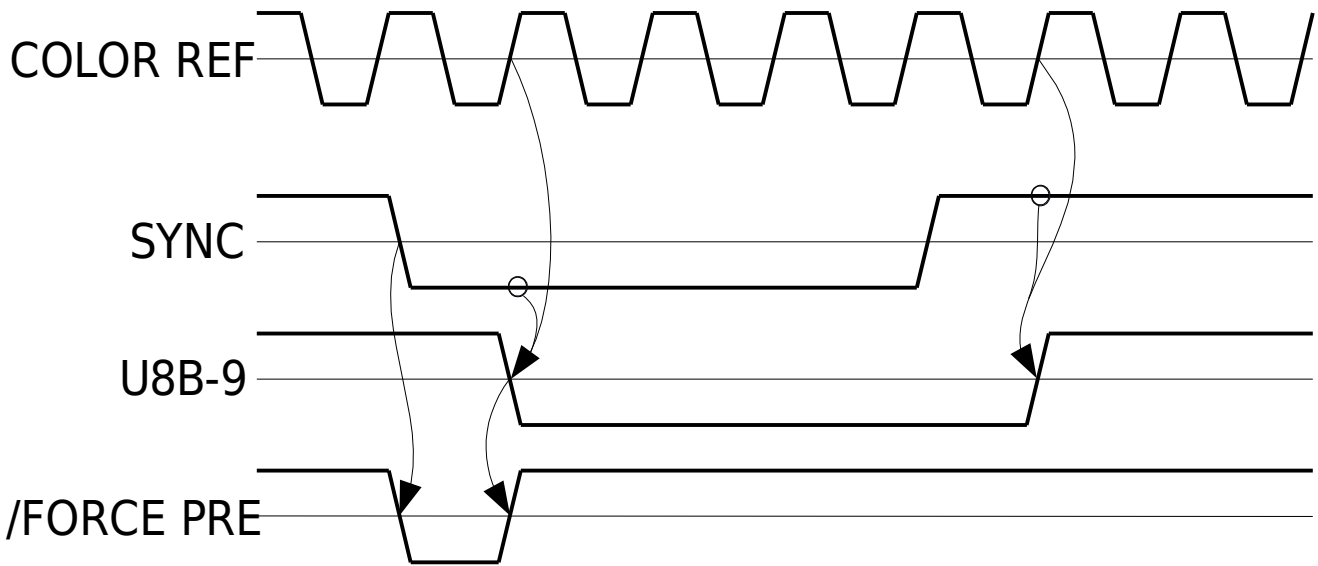


Fig 4.2: Generating /FORCE PRESET

Every time SYNC falls, FORCE PRESET will be asserted until the delayed from the U8B falls, i.e. /FORCE PRESET is asserted for less than one COLOR REFERENCE period. This signal presets the video scanning counters on the rising edge of COLOR REFERENCE.

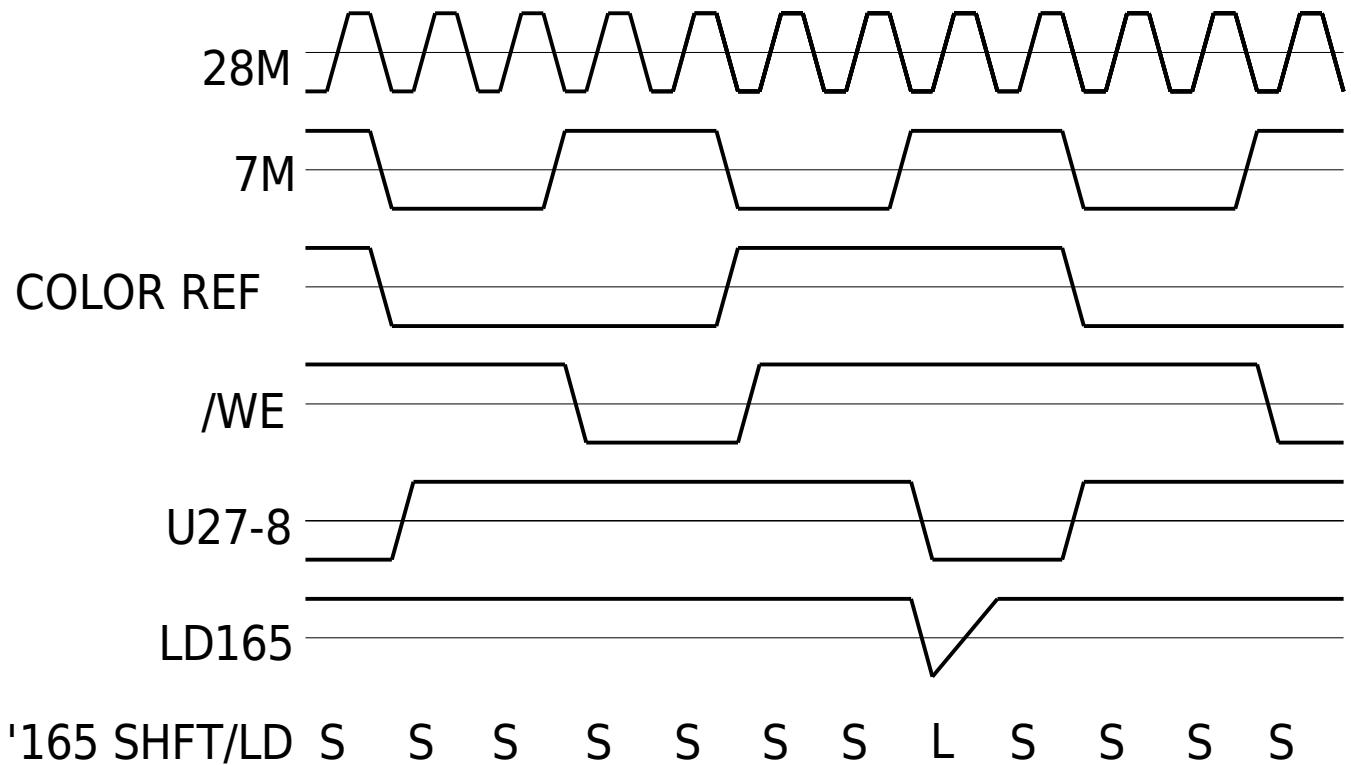


Fig 4.3: Generating LD165 and /WE

/WE is goes low when COLOR REFERENCE is low and 7M is high. This occurs late enough so that the address setup time of the RAM is met.

LD165 needs to be a short low-going pulse that immediately loads the 74HC165 shift register. If the pulse was too long, the register would not be shifting for too long losing some pixels. Generating this pulse with gates from the clock signals would have needed an extra chip, so another method was chosen:

U27C creates a signal (U27-8) whose high-to-low transition is exactly in the right place. This signal is the fed to a R-C (R58-C17) differentiator circuit which transforms the transition into a short pulse. It will also generate a positive-going pulse when U27-8 rises, but this pulse is cut off by the input protection of the 74HC165.

THE VIDEO SCANNING COUNTERS

The basic scanning mechanism:

A video scan line contains 560 pixels in DOUBLE HIRES and 80-col TEXT, 280 pixels in HIRES and 40-col TEXT and 40 pixels in LORES. To avoid switching logic, 560 pixels are always stored, storing each HIRES/40-col TEXT pixel twice and each LORES pixel seven times. Undisplayed pixels are also stored in memory to simplify the design. The saving of many chips outweighs the waste of RAM(88 bytes).

The 912 pixels are grouped into groups of 8 bits that form a byte in the RAM. Because every pixel lasts 70ns, every group is 560ns long, or 2 periods of COLOR REFERENCE.

When outputting the pixels to the VGA monitor, each pixel is 35ns long, so each group is 280ns long, or one period of COLOR REFERENCE. This is a nice coincidence that allows using timing signals that can be easily derived from COLOR REFERENCE.

There are only 114 bytes to be stored, so a 7-bit counter would be sufficient. The 8th bit can be used for other purposes.

The storage scanner:

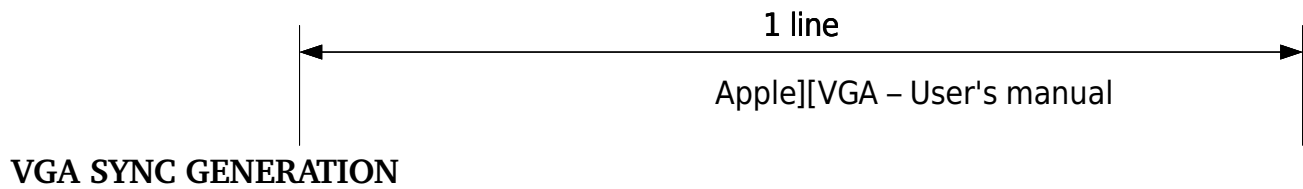
U12 and U13 form the storage scanner. Bit 0 is used as a frequency divider so that the remaining counter (bits 1-7) will increment every other COLOR REFERENCE cycle.

Bit 0 delivers another timing signal that transfers data from the shift register to the write data latch (see the section about RAM access). The storage scanner will be preset to \$0E every time /FORCE PRESET is asserted.

So counter runs in sync with the Apple's horizontal scanner.

The read scanner:

U10 and U11 form the read scanner, which is a little more complicated. It will preset to \$8C, count up to \$FF, overflow to \$00 and then be preset again. This cycle repeats forever and occurs twice every period of SYNC. Every time /FORCE PRESET is asserted the counter will be forced to preset (hence the name). So this counter is synced with the Apple's horizontal scanner, but runs at double speed.



VGA SYNC GENERATION

VSYNC:

The VSYNC signal is decoded from the Apple's SYNC signal. VSYNC is embedded in the video sync signal as follows:

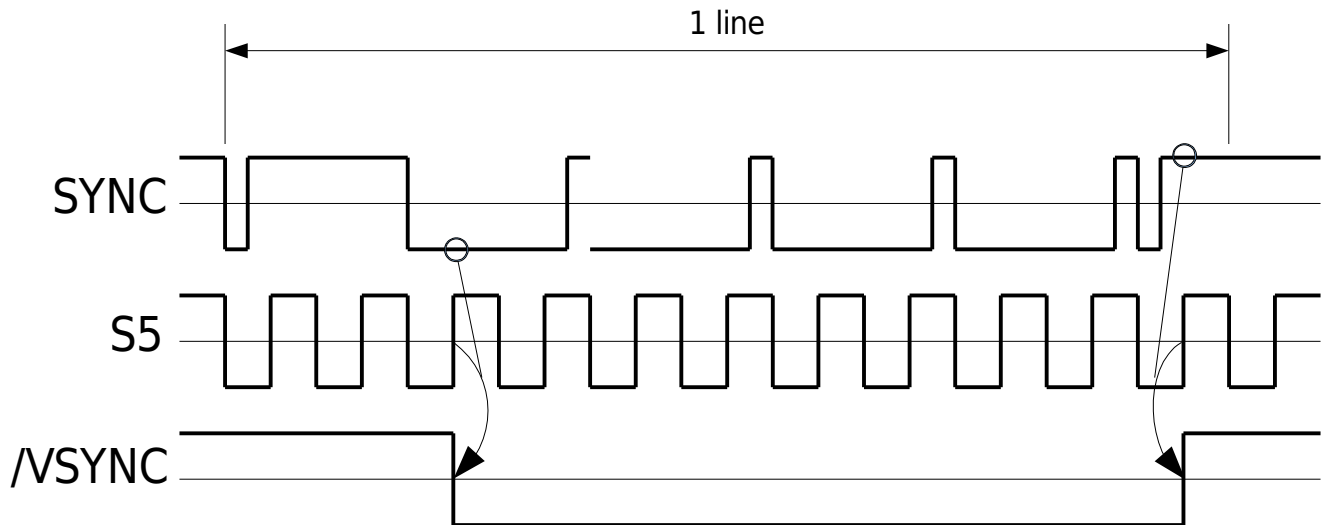


Fig 4.4: Generating VSYNC (Apple //e)

As you can see, the falling edge of SYNC always starts the next line. If SYNC stays low for a longer than it is high, a vertical retrace is started. I found out that, during vertical scanning, SYNC is always high when S5 (the 6th bit of the storage video scanner) rises. And, during vertical sync, it is always low, when S5 rises. U20B samples the SYNC signal every time S5 rises. This scheme also works with the doubly serrated sync of the Revision 7 Apple][as well as with the non-serrated sync of the Revision 0 Apple][.

HSYNC:

HSYNC is derived from the read scanner with a flip-flop (U20A). HSYNC is set when the read scanner presets and is reset, when the read scanner reaches \$94. The flip-flop will be reset some more times, but this does not have any effect.

The sync polarity jumpers:

The VGA standard transmits information about the number of displayed lines in the polarity of the sync signals. Only very few (very old) monitors examine this. Most monitors ignore the polarity and will work in any setting. But if you happen to own a very old VGA monitor that needs the correct polarity (probably an old original IBM without controls) the sync polarity can be changed using these jumpers. They simply select inverted or non-inverted output from U20A/U20B.

4.3.2 Video shifting

Video input shift:

The video data is always sampled at 14MHz and shifted into the 74HC595 shift register. The shift register is followed by a latch (also in the '595) that is loaded every two COLOR REFERENCE periods, i.e. just when a new video byte is ready, controlled by LD595. This happens every 560ns.

The '595 has three-state outputs to gate the video data to the data bus when needed.

Video output shift:

The video output shift register is a 74HC165 which always runs at 28MHz. The shift register is loaded every COLOR REFERENCE period, controlled by LD165. The load operation is interleaved with seven shift operations, so that an nearly evenly spaced bit stream is created. Because the signal is shifted through the video delay shift register (see section 4.3.4) correct bit spacing is not extremely important, as long as the color decode circuitry works reliably.

4.3.3 RAM access

Access timing:

As stated above, the video input shifter outputs one video byte every two COLOR REFERENCE cycles, and the video output shifter requires one video byte every COLOR REFERENCE cycle. To allow for one RAM to be used, the accesses are interleaved. Read access happens while COLOR REFERENCE is high. Write access happens when COLOR REFERENCE is low. Because there is video data only every other cycle, a “garbage write” occurs: Old video data is written to the address that will be overwritten in the next COLOR REFERENCE cycle.

Ram addressing:

Only 256 bytes of the 32KB RAM are actually used.

These 256 bytes are split into two pages of 128 bytes, with the first 12 bytes per page unused. Page 1 stores the video data, while page 2 outputs the data at double speed. After one line has been stored (and the last line has been output twice), both pages swap their function as U15C toggles. Now the video data is stored in page 2 and the video data that is present in page one will be output. If the line is finished, U15C toggles again and the process repeats.

U17 and U18 select the addresses for reading and writing.

4.3.4 Color generation and video output

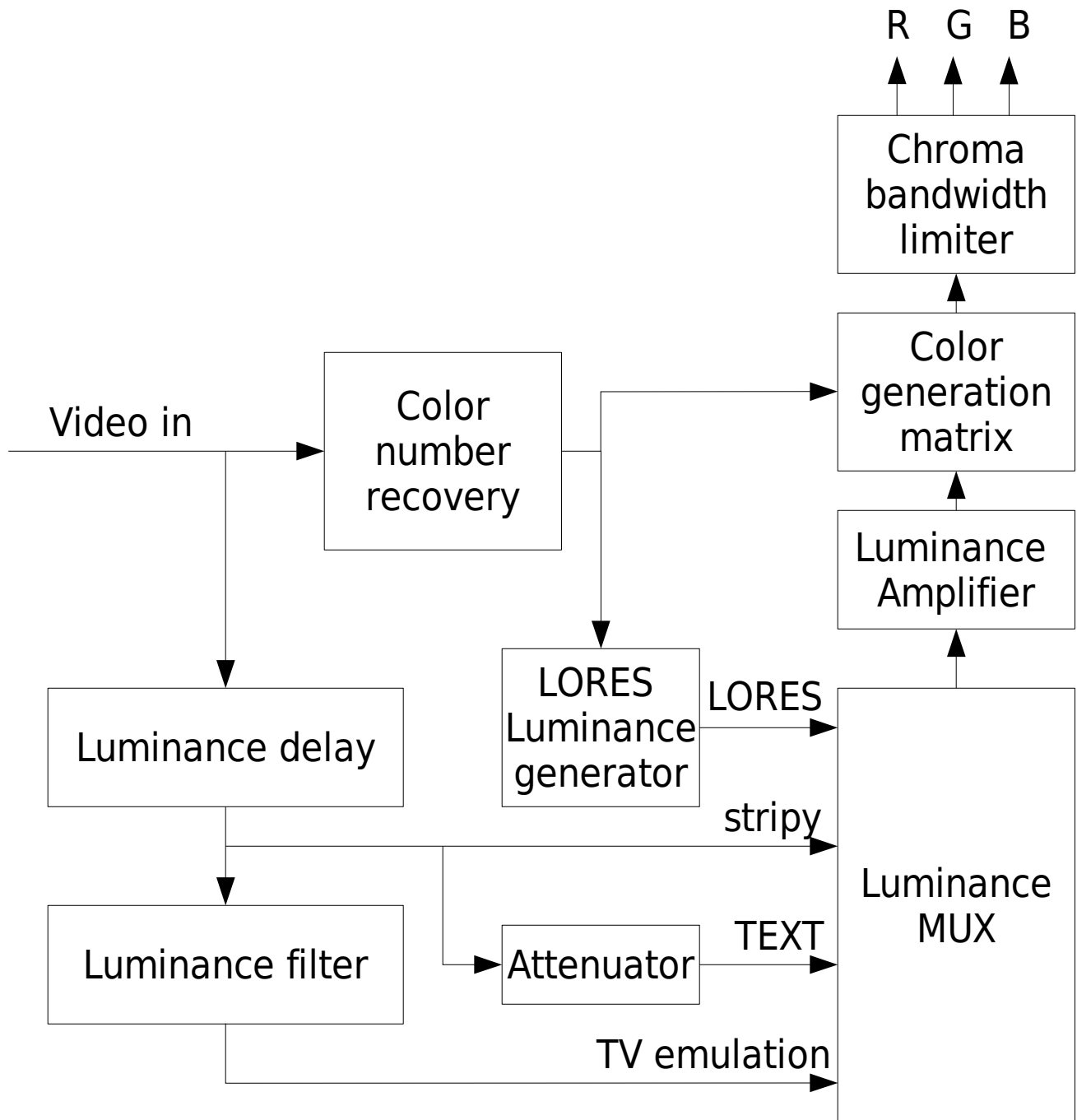


Fig 4.5: Video signal processing overview

LUMINANCE PROCESSING

The video signal consists of two parts that need to be separated.

There is the brightness of the pixel being displayed (“Luminance”), and the color information (“Chroma”). This part is about the luminance component.

The apple can only set a pixel to black or white. Gray is created by the monitor blurring adjacent black and white pixels to gray. And colors are created by other patterns of 4-pixel groups. Now if we have a very sharp monitor, these alternating black and white pixels will appear as vertical stripes. To avoid this, the picture could be blurred sacrificing the sharpness. Finding a compromise that is also suitable for text is impossible. But even then a compromise between sharpness, lack of stripes and lack of ringing (vertical bars appearing right of a colored area) needs to be found. To simplify this, another mode has been added, where the luminance is taken from the chroma decoder circuitry. This mode perfectly suited for LORES graphics.

Eye-stress can be greatly reduced by reducing the brightness of text.

So now we have:

1. Feed through
2. Attenuated feed through
3. Filtered
4. LORES

Because chroma processing causes some delay, the luminance needs to be delayed too, so that both arrive at the output at the same time. This is done in 5 bits of a '574(U25) wired as a 5-bit shift register. Only 4 bits delay are actually used. The 5th bit is also connected to allow modifying the card for an even lower chroma bandwidth, which would introduce more chroma delay. The LORES luminance is taken from the chroma circuitry, so that is is already delayed.

The filtering is done by two traps:

The chroma trap (7MHz) eliminates the signal component that is interpreted as color.

The gray trap (14MHz) eliminates the fast toggling which is done to display gray. As a side-effect, it eliminates the 1st harmonic of the chroma signal.

The chroma trap will also be found in a color TV set, but the gray trap is missing. Because “real” television is not a 1-bit video stream, it is not needed, and the cheap picture tube of a TV set blurs the picture anyway.

The LORES luminance signal generation will be described in the next section.

The luminance MUX (U26) selects the correct signal, which is then amplified by the luminance amplifier and fed to the color matrix.

CHROMA PROCESSING

A TV treats the signal frequencies around COLOR REFERENCE as chroma. In normal television these signals are created by additional circuitry. Not so in the Apple: The frequencies are “accidentally” created with the luminance signal by outputting 4 pixel sequences. There are 16 different sequences that form the 15 LORES colors (1010 and 0101 are both gray and they are counted as a single color).

The color saturation and brightness depends on the actual bit pattern, while the hue depends on the position of the pattern relative to COLOR REFERENCE.

0011001100110011, for example, could be seen as violet, orange, green or blue.

In the color decoder, all frequencies are doubled. For that reason, the reference is no longer COLOR REFERENCE, but the 7 MHz clock.

The first block called “color number recovery” samples 4 bits of the video signal. It consists of 2 '74 dual flip-flops. Each flip-flop samples one bit of the 4-bit group.

U23A samples the first bit after 7MHz rises, U23B samples the second, and so on.

This is actually the LORES color generation process reversed, the output of the 4 flip-flops is the LORES color number.

When in monochrome mode, the four flip-flops are all set to high state, which is equivalent to LORES color 15: white.

4.3.5 Control logic

This is one of the most straightforward circuits on this card. It can be split in

- LORES/HIRES soft switch
- mode soft switch
- glue logic
- detection aid

LORES/HIRES SOFT SWITCH

The address from the data bus is decoded by U16 and U9. If the LORES/HIRES soft switch is selected, U8B is also selected, so that its state will follow the LORES/HIRES switch. No connection to the motherboard is required.

MODE SOFT SWITCH

The mode soft switch has 16 positions, of which 5 are actually used. The other combinations are reserved for future use. The switch is connected as usual, using the address decoding present on the main board.

GLUE LOGIC

To ensure that the card can not be configured to clobber the display, the glue logic ensures the following:

- TEXT is always monochrome at high bandwidth
- LORES is always colored blocks
- LORES always has high chroma bandwidth

The glue logic is straightforward, examining its function is left to the reader. Only the relay driver is uncommon: Q2 and Q4 form an open-collector NAND gate with high voltage and high current capability.

DETECTION AID

Detecting a card is usually done by examining the ROM of the card. But what, if the card has no ROM and even doesn't respond to commands at all?

Because board space is limited, a simple solution needs to be found.

Because reading the command space of other cards is not safe (a disk controller would actually go crazy), the signature needs to be put into ROM space, which can be safely read. To save an extra driver chip a “bug” of in the Apple][is used: The video data will stay valid on the bus in all CPU read cycles, if no device responds to the current address, because the bus' capacity remains charged. In the Apple][VGA a diode (D3) discharges the high bit of the data bus if the ROM space of the card is selected. As a nice coincidence, all normal characters have the high bit set in the screen memory, so an informational message can be shown while detecting the card. The program searching the card needs to check all ROM locations to avoid that it just checked a ROM of another card which happens to have the high bit reset. It is impossible to find a ROM which has all high bits reset, because some very frequently required 6502 instructions have the high bit set.