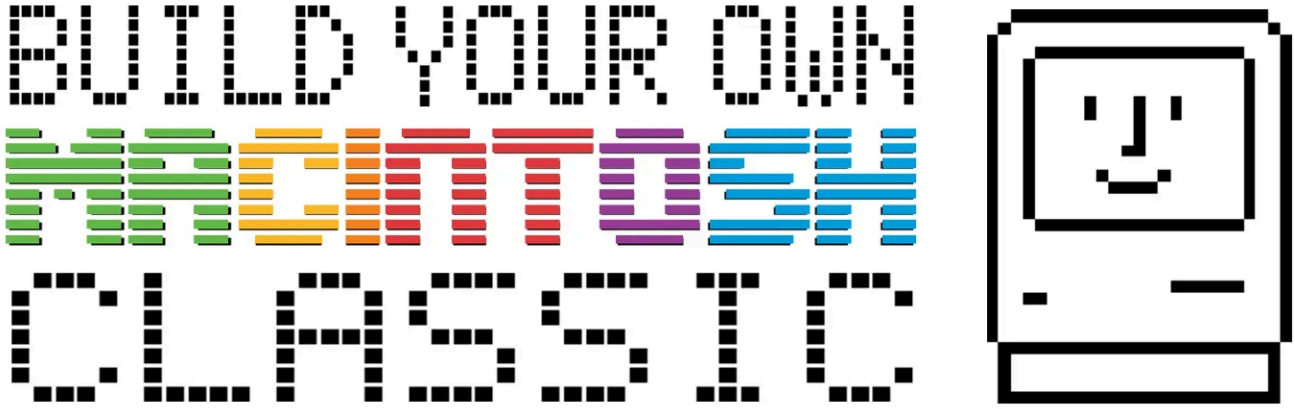
Control a Macintosh Classic CRT with a BeagleBone Black - Part 1

June 26, 2016 | Written by: nerdhutblog
Posted in: Electronics

🥀 Some of the information in this article may be outdated due to service shutdowns, API changes, or software updates. As a result, the instructions might no longer work as described.



Introduction

This article will explain how the timing of the Macintosh Classic CRT works and how I tried (and failed) to interface it with the Raspberry Pi, and how I successfully interfaced it with the BeagleBone Black's PRU.

How does the CRT display an Image?

I will not explain how the CRT works in theory, other people have done that before. But I want to write about the signals needed to compose an image on the CRT-display.

The screen basically needs three signals: HSYNC, VSYNC and DATA (It might remind you of VGA). HSYNC stands for horizontal synchronization, VSYNC is for vertical sync, DATA sends the video pixels to the display. Since the Macintosh Classic display is monochrome (black/white), there is only one DATA line. If it had more colors, there could be more separate lines (like one for red, one for blue and one line for green).

HSYNC triggers a new line on the display. At every falling edge, the CRT's circuit positions the screen's electron beam to the beginning of the next line. The signal has a frequency of roughly 22.25kHz and it looks like this:

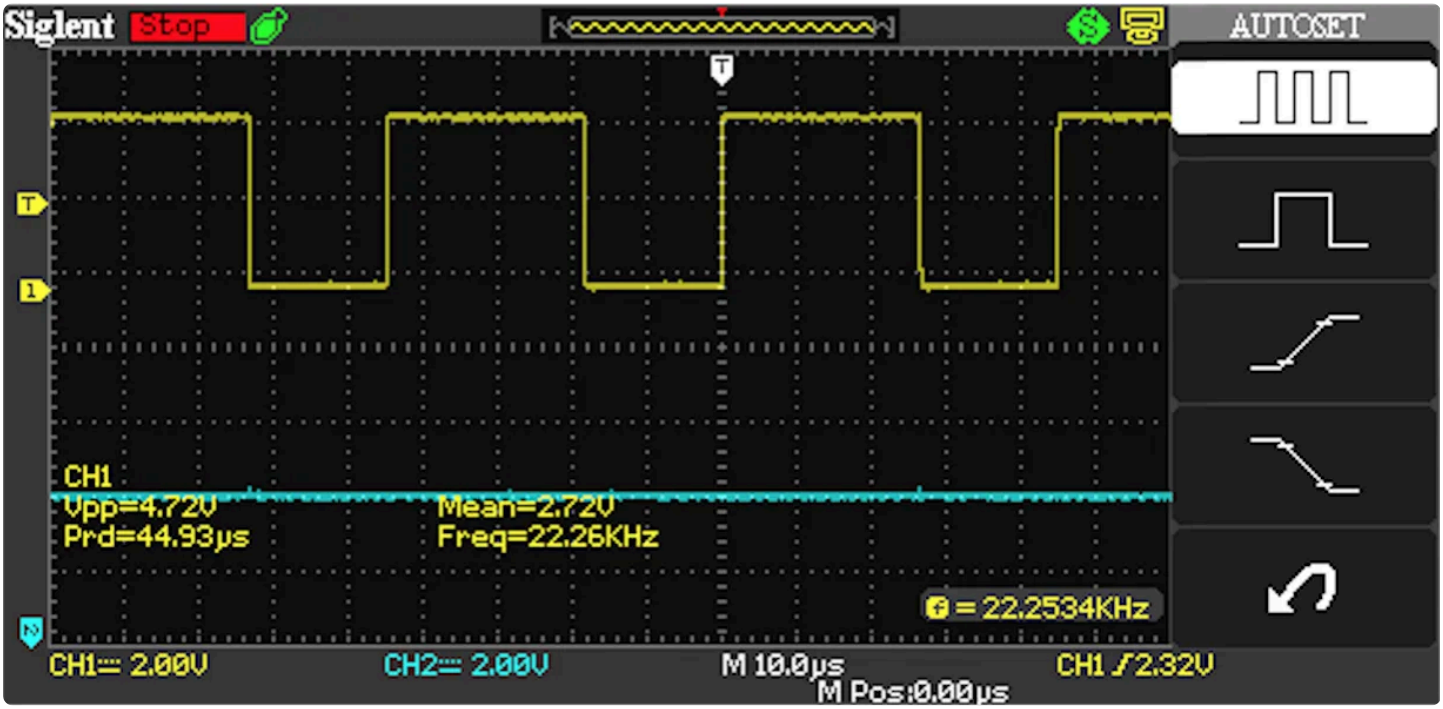


Fig. 1: The HSYNC signal for the CRT

Each time the HSYNC signal gets low, the screen gets ready to display a new line of data. You can see that the period is about 45µs for one complete frame. But you can also see, that the signal is HIGH 59% of the time and LOW for 41%.

The VSYNC signal triggers a new frame on the display. It has a frequency of about 60.15Hz, which is also referred to the refresh rate of the screen. This signal was a bit harder to create because it had to be synchronized with the HSYNC signal. In my application, the VSYNC signal occurs after 342 HSYNC edges. The HSYNC pulses continue during the VSYNC, but the video data doesn't:

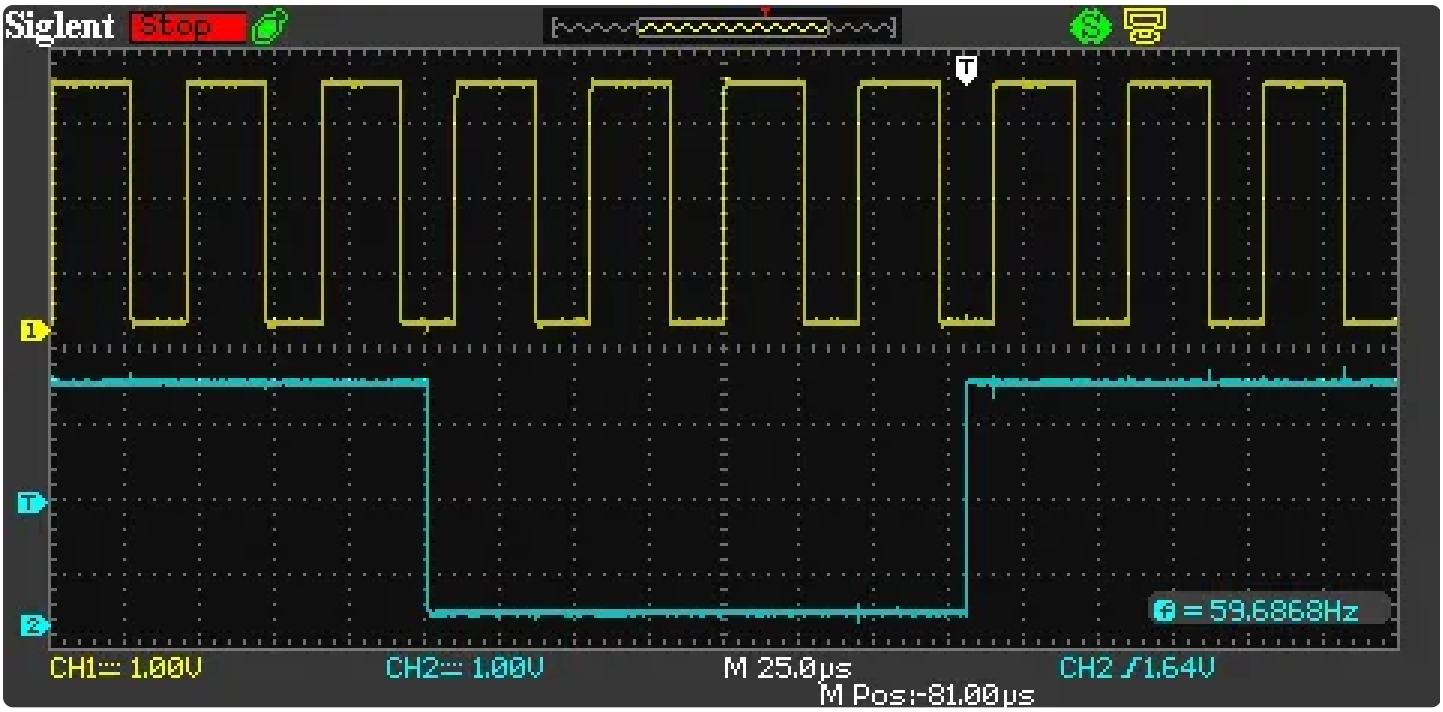


Fig. 2: HSYNC pulses (ch1, yellow) and VSYNC pulse (ch2, cyan)

You can clearly see that the HSYNC pulses continue while the VSYNC line get's low. VSYNC is also triggered on the falling edge.

The last signal is the DATA line. This was the trickiest bit and it was not possible to produce it with the Raspberry Pi. The problem is, that this signal is time critical. It always has to take exactly the same time to send this data to the CRT (and this is not possible to make with the raspberry pi by just writing code. However, you can construct slower frequencies exactly with the Pi).

The data is sent by bit banging the pixels to the CRT with a dot clock frequency of around 16MHz (15.6672 MHz to be precise). Bit banging means that you send out the state of a pixel (HIGH = white, LOW = black pixel) via the DATA line in an exact timing, one bit after another:

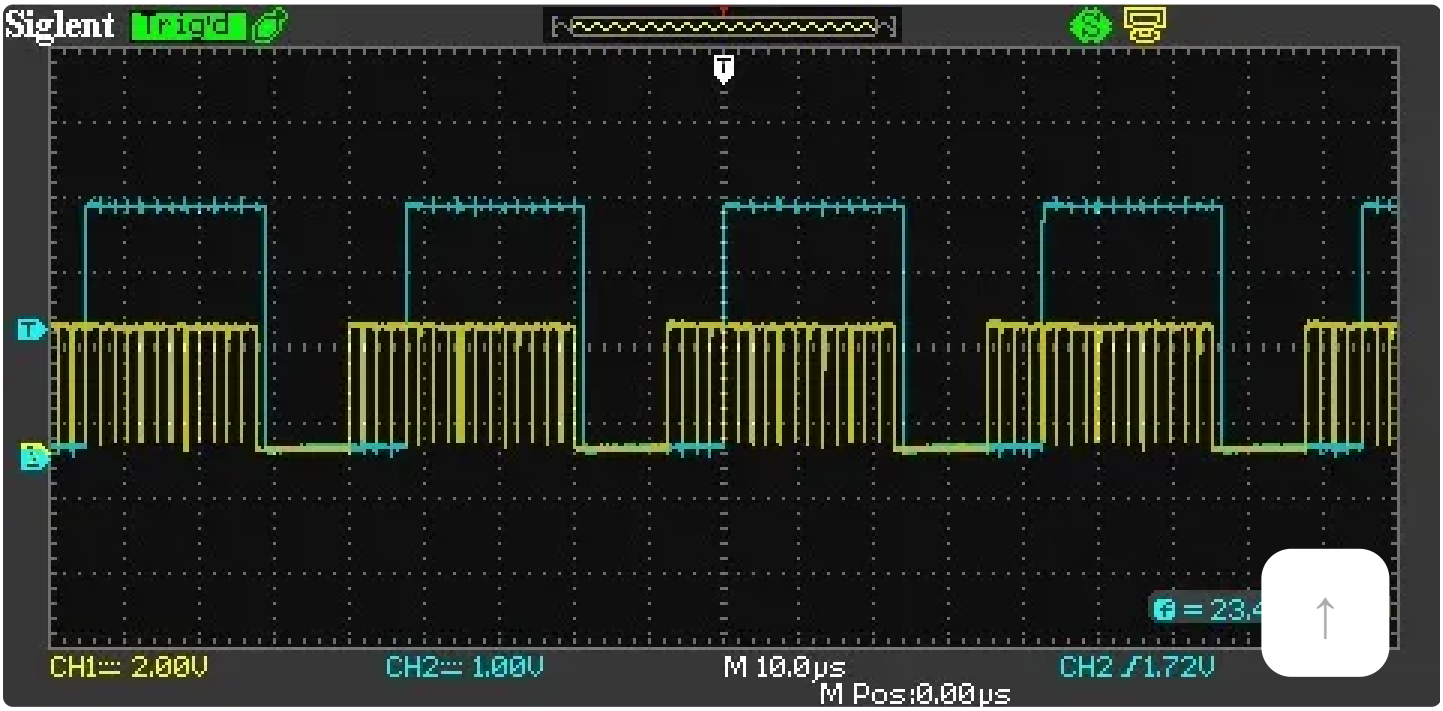


Fig. 3: You can see the HSYNC signal (ch2, cyan) start a new frame and the DATA (ch1, yellow)

As you can see in fig. 3, the DATA does not start exactly at the falling edge of the HSYNC signal, it starts (roughly) 10µs after the falling edge. This time is needed by the CRT to position the electron beam at the start of the next line. There is also a time needed after a VSYNC to reposition it to point at the first pixel's position on the screen. You can get a better understanding of the timings of these two signals by looking at fig. 4:

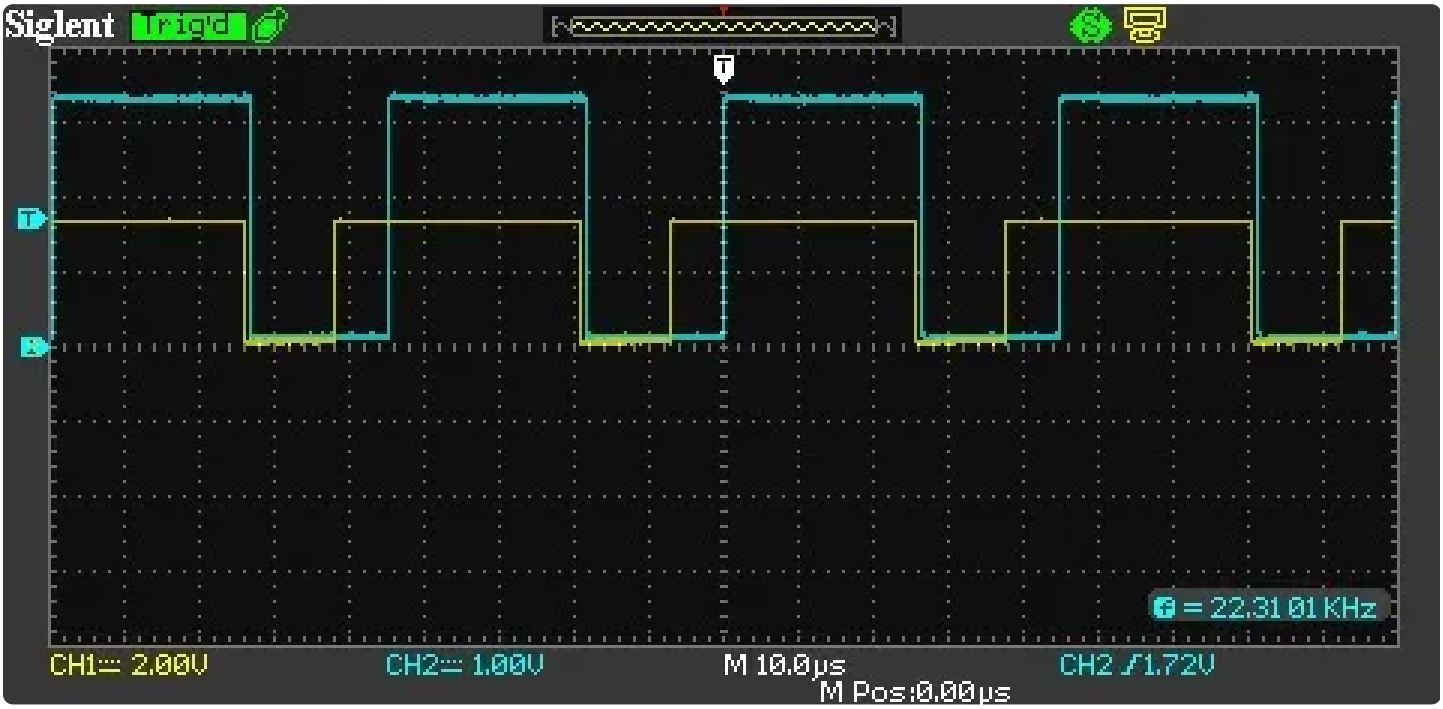


Fig 4: Sending completely white lines.

Note: The DATA line should be high while no pixels are sent to the CRT, fig. 3 and 4 show the DATA line getting low after the pixels were transmitted, to make the pulse duration visible.

This was basically everything that is needed to display an image on the CRT. The signals have to run synchronized and with exactly the frequencies they need to have. Otherwise, the image might get distorted.

Important numbers

If you want to rebuild this or a similar project, here are some important numbers for this particular CRT monitor:

```
HSYNC-Frequency
22.25kHz, 45µs period, 18.45µs low 59% PWM duty cycle

VSYNC-Frequency (Refresh rate)
60.15Hz, 16700µs period, 180µs low 99% PWM duty cycle
```

```
DATA
15.6672MHz, 512 pixels in roughly 32.8µs Signal has to be high while no pixels are sent
```

If you can find a datasheet or a table with the exact timings for your display, you can get the information from that. Otherwise, you can try to measure the signals produced by the original hardware's video interface. Luckily, most older computers had such documents and you can easily find them on the interweb. Macintosh Classic Developer Note, Macintosh Classic II Developer Note.

Create the signals with the Beaglebone's PRU

Creating exactly timed signals is possible with the BeagleBone Black. You can read about the process here. This was not a big deal, the harder part was to create the DATA line. I'll describe the method in the next part of the series!

Table of contents

An alternative version of this project using a Raspberry Pi 4.

Tags: Black classic crt hardware pwm macintosh timings vga