Tynemouth Blog

Tynemouth Software - Making new things for old computers - sellmyretro.com/store/tynemouth-software

Sunday 15 October 2023

How the ZX81 Generates Video

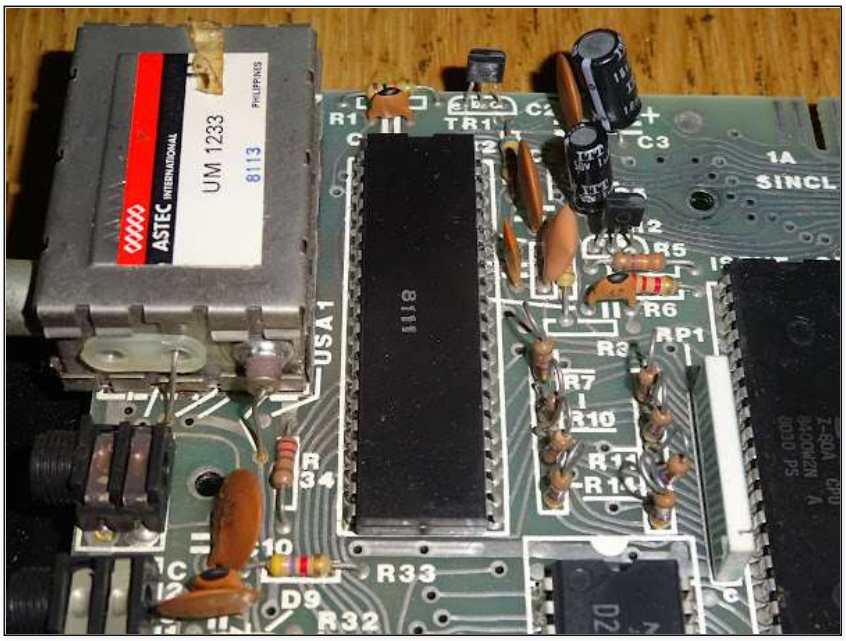
Now the Minstrel kits are back in stock, it's time for the second in this series of posts from my Patreon, this one covering the video output of the ZX81.

Following on from the previous post which covered the inner workings of the ZX80, today we look at the changes made for the ZX81.

Most of the video generation circuitry is the same as that in the ZX80, so please refer to the previous post if you have not already seen it.

http://blog.tynemouthsoftware.co.uk/2023/10/how-the-zx80-generates-video.html

Before discussing the actual changes, I should first address the biggest practical change. All of the TTL logic chips used in the ZX80, as well as all the logic required to implement the other changes have all been contained within a single IC. The ULA (Uncommitted Logic Array), an early form of CPLD, in this case a factory mask programmed array of logic gates. This is a very early one, with just a date code.



The new ZX81 board (right) was a lot smaller and simpler than the ZX80 (left) thanks to most of the chips being replaced by the ULA.

The exact implementation of the new features is not obvious from looking at the board or the schematic. It has been reverse engineered by various people over the years, most notably Grant Searle who's website is a mine of information for all things 8-bit single board computer and ZX80/81 related..

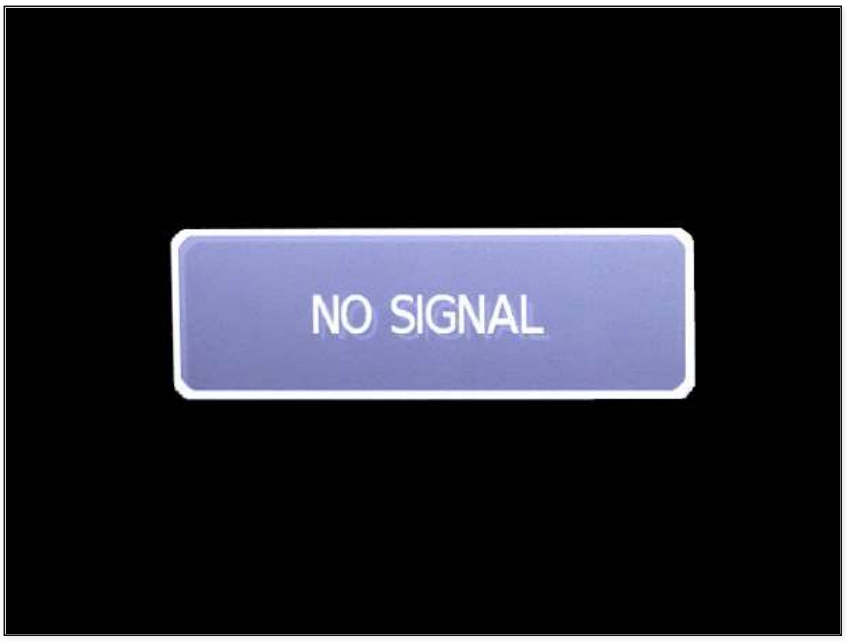
http://searle.x10host.com/zx80/zx80.html

Other than the cost reduction and simplification of the ZX81 into essentially a 4 chip computer, the main aim was to get rid of the flicker whenever a key was pressed or code was run.

The Z80 in the ZX80 was usually in one of two states, it was either drawing the screen, or running code.

Since the Z80 is so intricately involved in generating the screen, it cannot run code at the same time. When it has to run code, it can not generate a screen. This leads to anything from a slight flicker when you press a key to extended periods where no video signal is generated as user code is executing.

A modern TV will switch to "No signal" after a few seconds of that.



(you still get a black screen on an old TV, it would only be snow if the modulator was shut down, but it's difficult to take a picture of a black screen, so here is a picture of some snow in case you are feeling festive)

To get around that, the ZX81 added a new mode. The ZX80 style mode became known as "fast" mode. A mode where code is run at full speed, but cannot generate a screen.

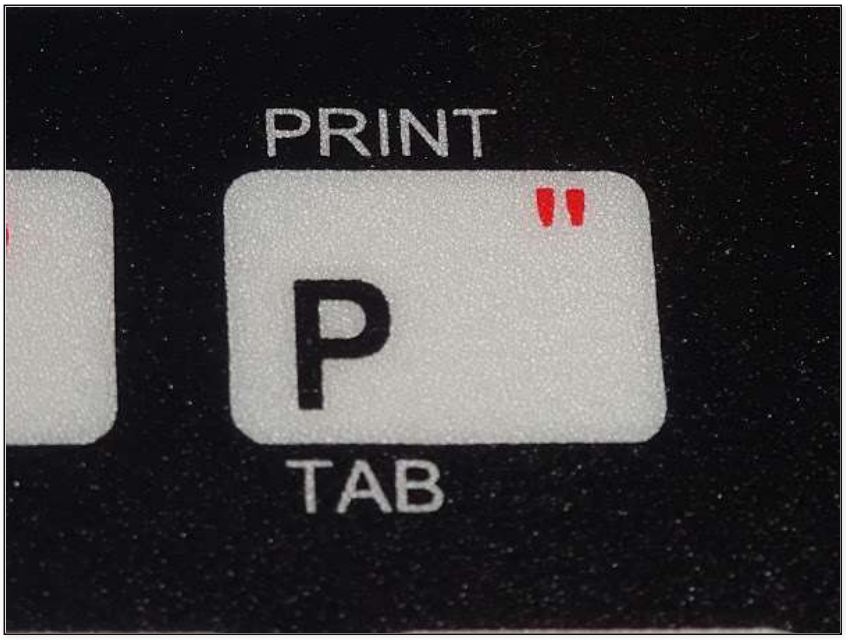
The ZX81's new mode was the corresponding "slow" mode. Here, code is run when the blank areas at the top and bottom of the screen are being generated. The rest of the time, the screen is generated as normal. Code is only run for a short amount of time, hence the "slow" mode.

The USA models had fewer blank lines each frame (64 vs 112), so slow mode on a TS1000 was even slower than on a ZX81.

The "fast" mode was still available if a lot of code needed to be run quickly, a notable example being the "mists of time" when the maze is generated in 3D Monster Maze (oddly, this is mainly due to that section being written in BASIC, unlike the rest of the game).

The new version of BASIC for the ZX81 was double the size, an 8K ROM, with trig functions and a floating point number system. It also had an updated font (see ZX80 post for examples of the change to the K character).

It added lots of functions, and moved many of the existing ones around, into more familiar places such as PRINT and " on the P key, and LOAD on J.



There were also keywords to enter the new SLOW mode and FAST mode.

So, how does it work?

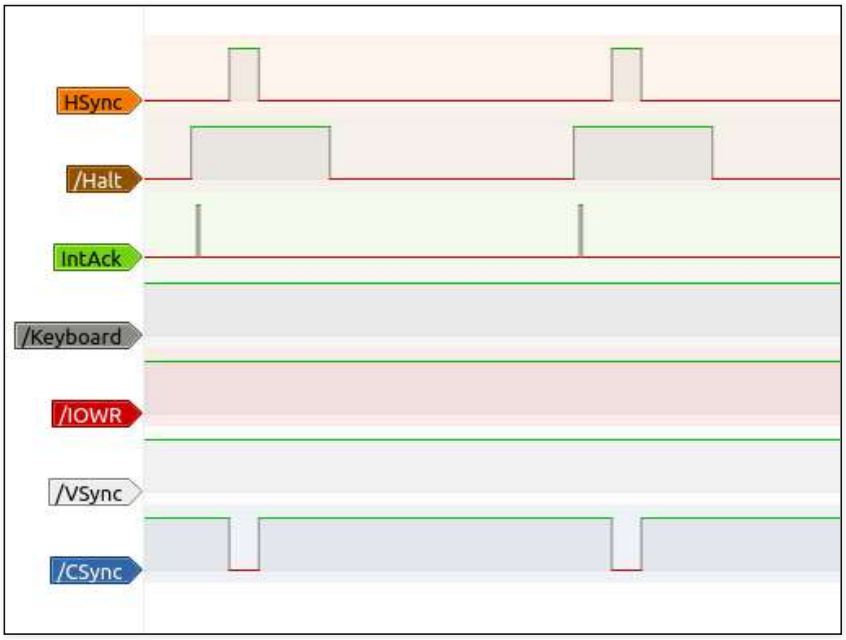
A quick recap of how the ZX80 generates normal lines:

The Z80 is setup to read a line of characters in the display file. It progresses through that line, incrementing the refresh counter for each character, until the new line character is reached. This halts the Z80 and it remains halted for a short time until woken by an interrupt at the end of the line.

HSync indicates the start of a line, /Halt goes low whilst halted, and the IntAck pulses show where the interrupt is acknowledged. (I am not showing /Int here as it is tied to A6 which does a lot of other things, so it does not help)

The blank lines at the top and bottom of the display are generated by pointing at the start of the DFILE which is a newline character, so it immediately halts and starts to run NOP instructions internally.

It remains halted for the rest of the line, until it is again woken by an interrupt.



That is a lot of time spent halted, and this wasted time is used on the ZX81 to execute user code.

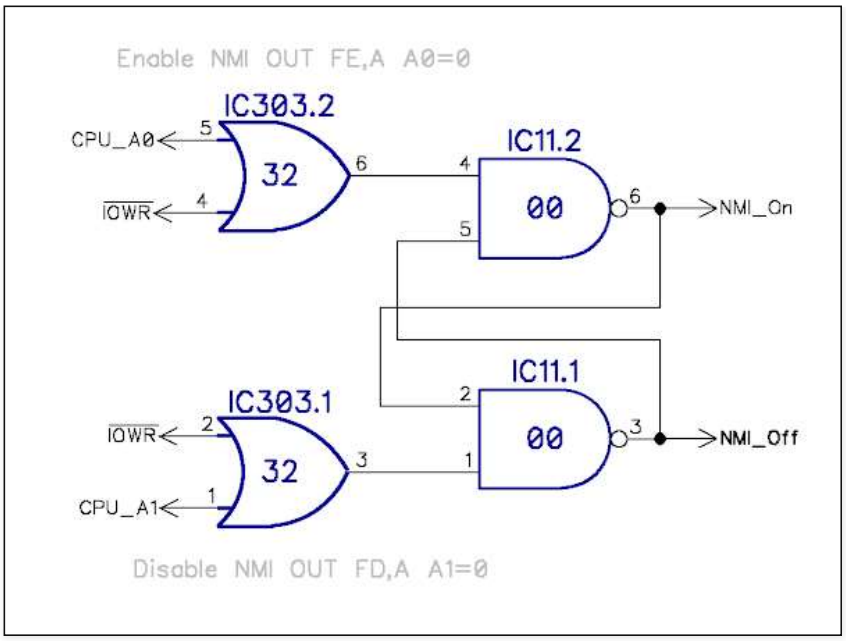
Previously the mechanism to generate the interrupt at the end of the line relied on the Z80 refresh counter counting up to a point where A6 was low and an interrupt was triggered.

Using the refresh counter relies on the instructions all being the same number of cycles (they are NOP instructions). User code will contain a variety of instructions and the number of cycles each instruction uses varies, so this will no longer be a constant time.

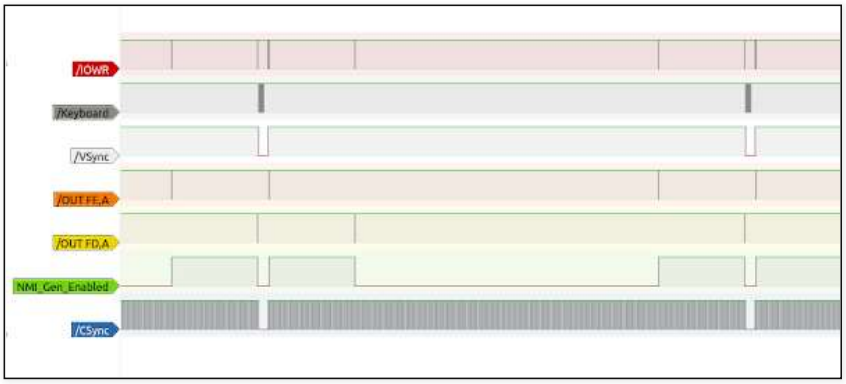
To get around this, the ZX81 added a counter which is reset at the start of each line. When it counts 208 cycles, it is reset and the next line begins. The start of each line generates the horizontal sync pulse, and when configured to do so, a Non Maskable Interrupt that will interrupt user code and increment an internal counter of how many lines have been drawn.

NMI generator enable

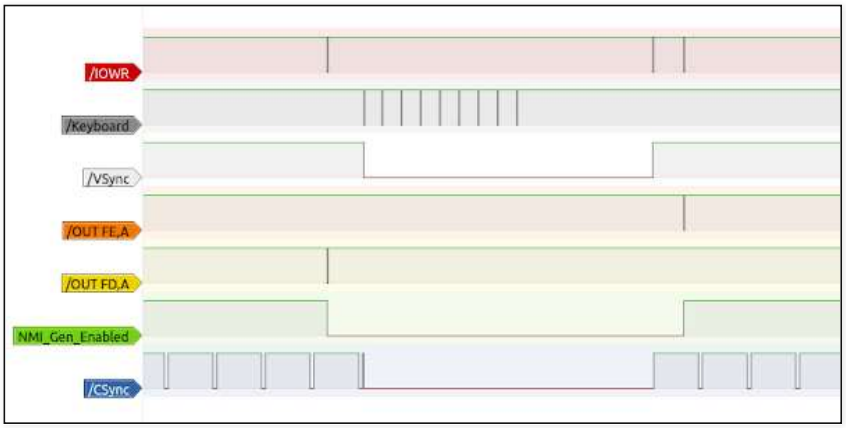
To control if the NMI signals should be generated, a latch is used. This is set by OUT FE,A, and cleared by OUT FD,A. These calls are generated by the OS as part of generating the display.



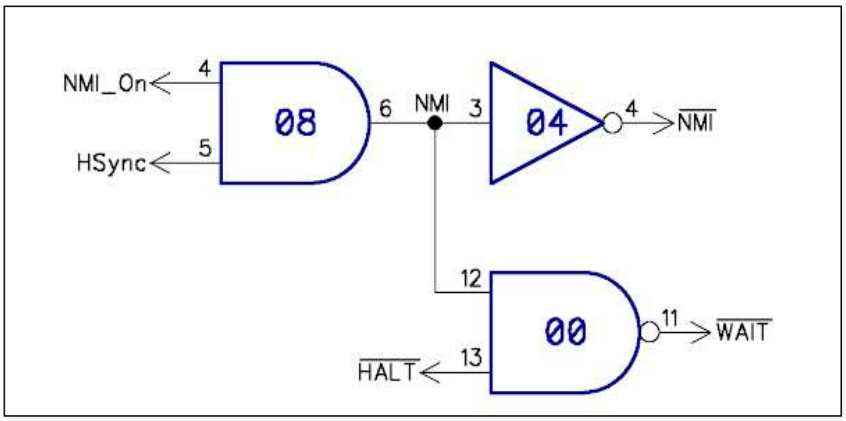
Looking at a whole frame, you can see the NMI generator is enabled in the sections at the top and bottom of the screen.



Why is the generator disabled during the VSync pulse? well that is when the keyboard is scanned, as it was in the ZX80. The first keyboard read is actually the trigger for the VSync pulse *(see the ZX80 post for further details)*.

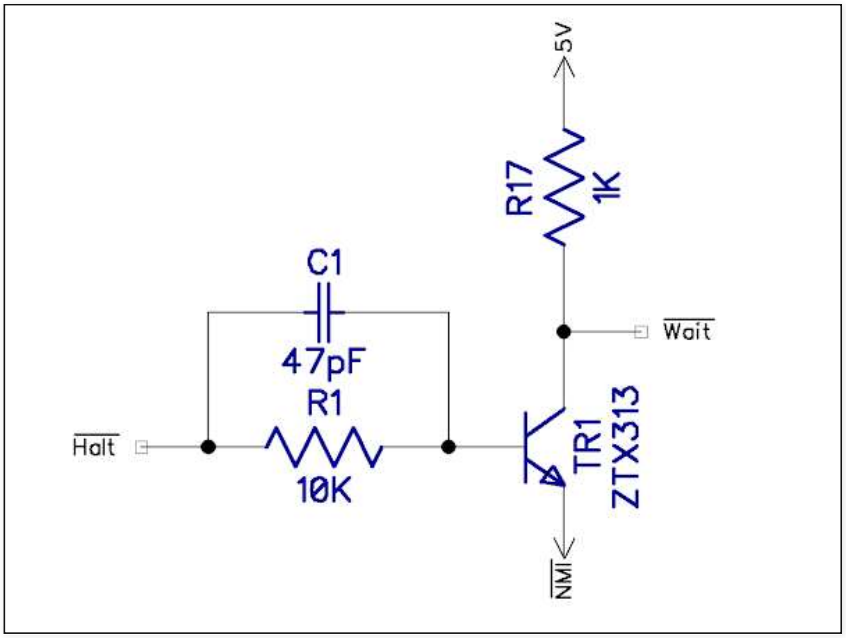


The NMI Generator enabled signal (here shown as NMI_On) is used to gate the HSync signal to create an /NMI pulse at the same time.



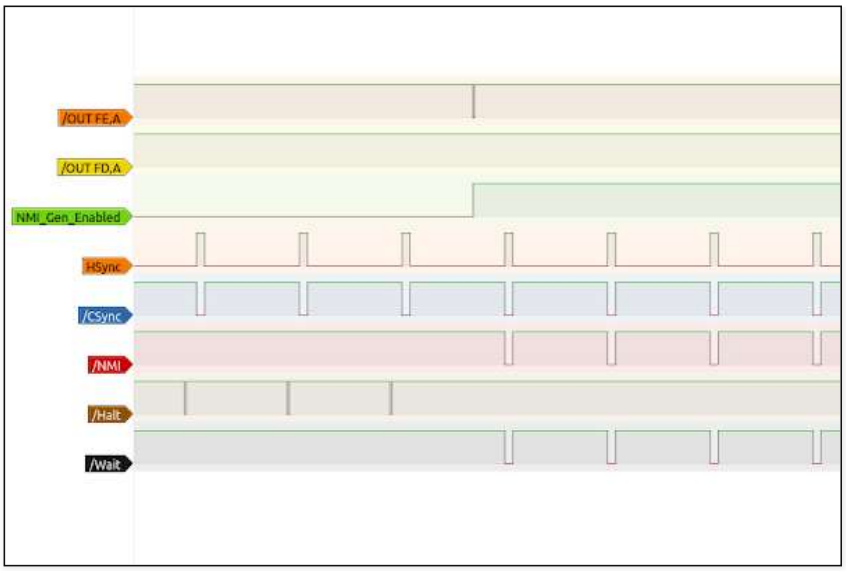
/Wait is also pulled low, to ensure the Z80 is in the correct T-State when the NMI is serviced. This is gated by the /Halt signal, which would always be high as that is normally triggered by processing the End of Line character, which will not happen on the non-visible lines.

I have drawn the /Wait signal generator using a NAND gate (as it is in the Minstrel 3), but that is not the way it was implemented on the ZX81. The ULA contains all the counters and logic to create the /NMI pulse, but the /Wait signal was implemented externally with a single transistor and some passive components.



It looks a little odd in the ZX81 schematic, so I have redrawn it differently to make it easier to see what is happening. For the moment, imagine the emitter was connected to 0V instead of /NMI. It is then a simple inverter. When /Halt is low, the transistor is off, so /Wait is pulled high by R17. When /Halt is high, the transistor switches on and pulls /Wait low. Now, add in the complication that the emitter is actually tied to /NMI. No current can flow in the transistor unless /Halt is high and /NMI is low, in which case, /Wait is also pulled low. *(what is the capacitor doing there? it speeds up the switching time of the transistor).*

Here you can see the last few visible lines, the NMI generator being enabled and then the first lines where user code is run.

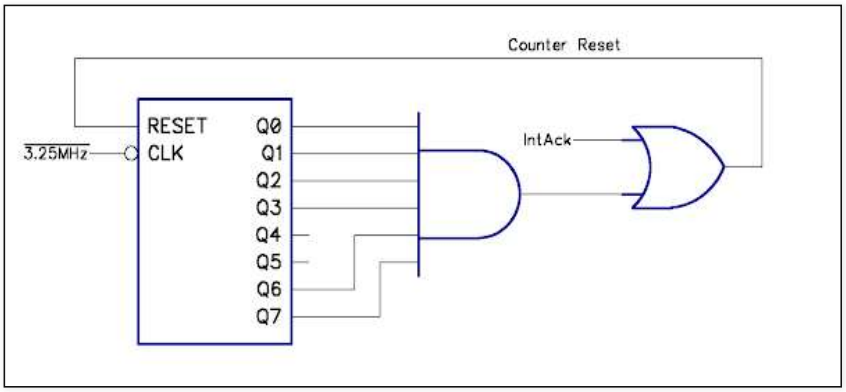


During the text lines, /Halt is used to halt the Z80 until it is at the end of the line. This stops when user code is running, and /NMI pulses take over.

HSync counter

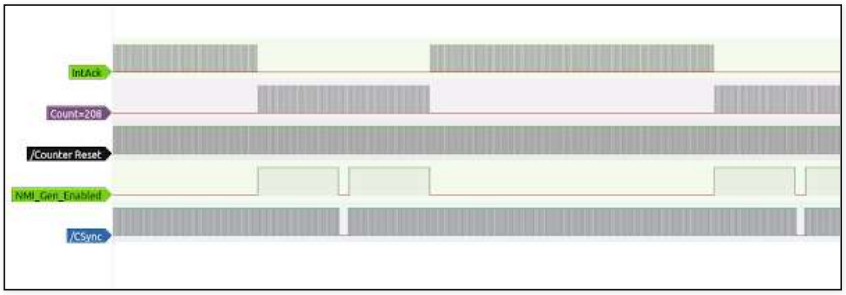
HSync pulses are generated based on a self resetting counter. The counter is reset when outputs Q0, Q1, Q2, Q3, Q6 and Q7 are high. The first time this is reached, Q4 and Q5 will be low, so the value will be 1100 1111, 0xCF, or 207. So it counts from 0 to 207 based on a 3,25MHz clock. 3,250,000 / 208 = 15,625. The gives a line frequency of 15.625KHz, a period of 64uS, exactly as required.

Q4 and Q5 are not tested, so the pulse would be generated at CF, DF, EF and FF, but the counter should be reset the first time the other outputs are all high at CF, so it is not necessary to test those.

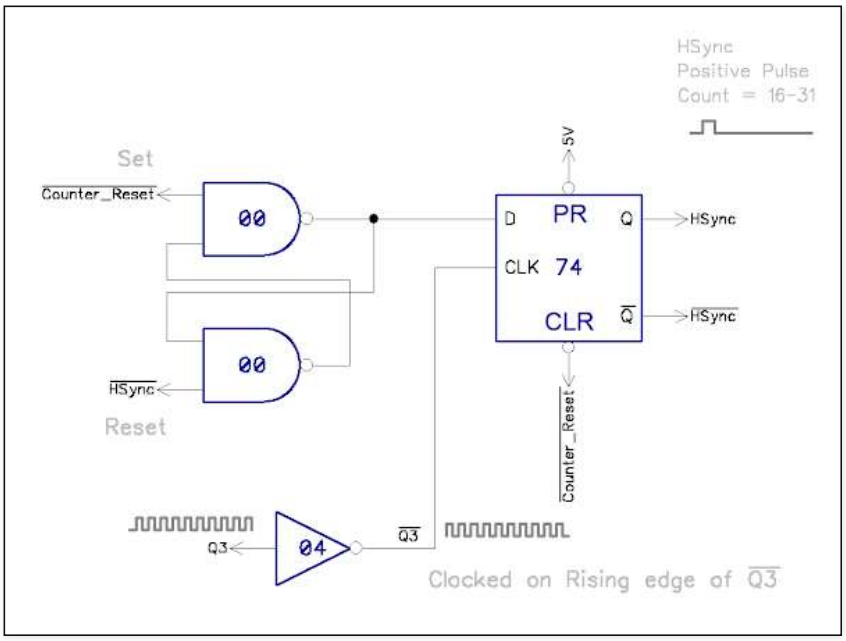


I have drawn the schematic using ideal parts, to make it easier to follow. For example, the counter is actually two 4 bit counters chained together, and you can't get a 6 input AND gate, so in practice a triple 3 input AND gate or dual 4 input AND gate would be used, but that makes it a little more difficult to follow.

For the text lines, the end of the line causes an interrupt to fire. Once this is processed, an Interrupt Acknowledge signal is generated. The IntAck appears to arrive just before the counter would reach 208, so the count = 208 pulse is not fired for text lines. Both the IntAck and counter=208 pulses should reset the counter, so they are ORed to create the counter reset pulse.



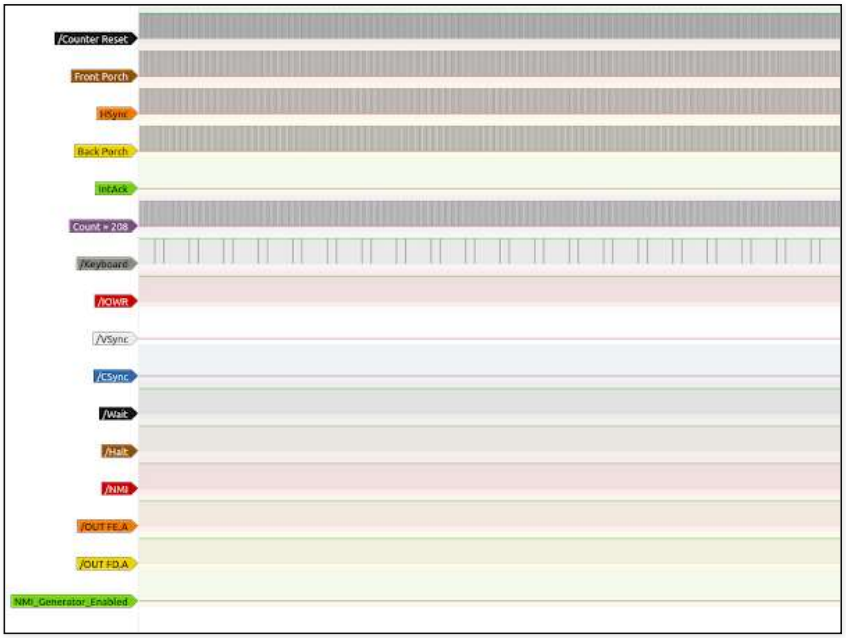
The counter is tapped at the Q3 point to get a 16 cycle clock, a pulse every 4.92uS. A series of flip flops is then used to generate some pulses based on this. HSync needs to start after 16 cycles, and last 16 cycles, so the following logic is used to create the appropriate 16 cycle / 4.92uS pulse.



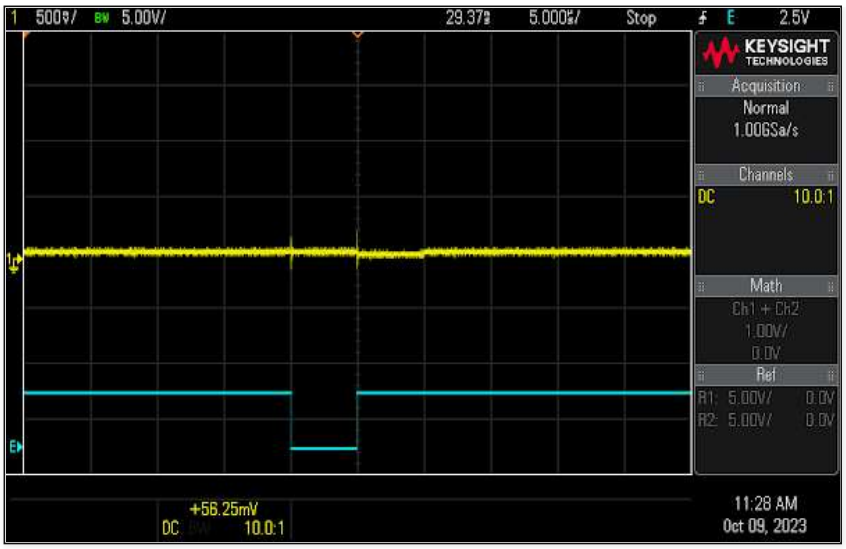
Fast Mode

In "fast" mode, (the only option in the ZX80) the Z80 is dedicated to running code, so the screen does not get updated.

The 208 cycle counter is still running and generating HSync pulses, but there is no activity on the other signals.

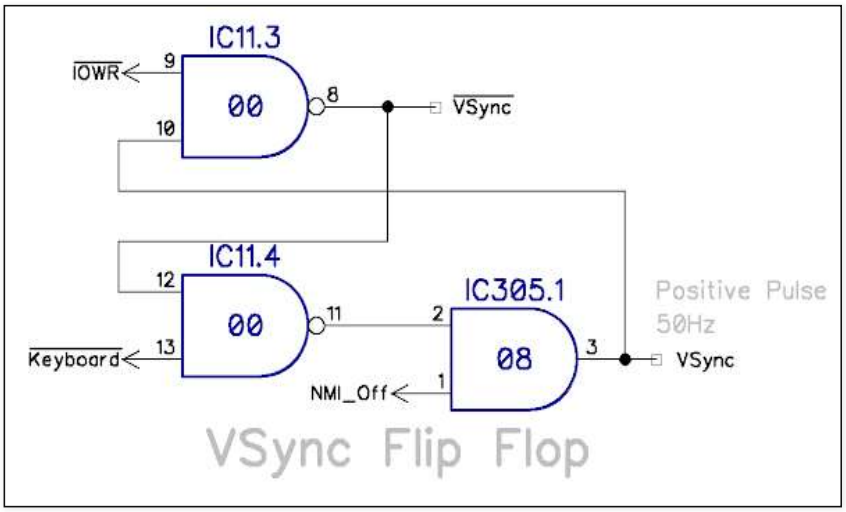


The NMI generator is not enabled, so no /NMI pulses are generated. The keyboard is being periodically checked, which activates the VSync, so that signal is permanently stuck low, and the composite video signal is flat. Yellow is composite video, blue shows the HSync signal.

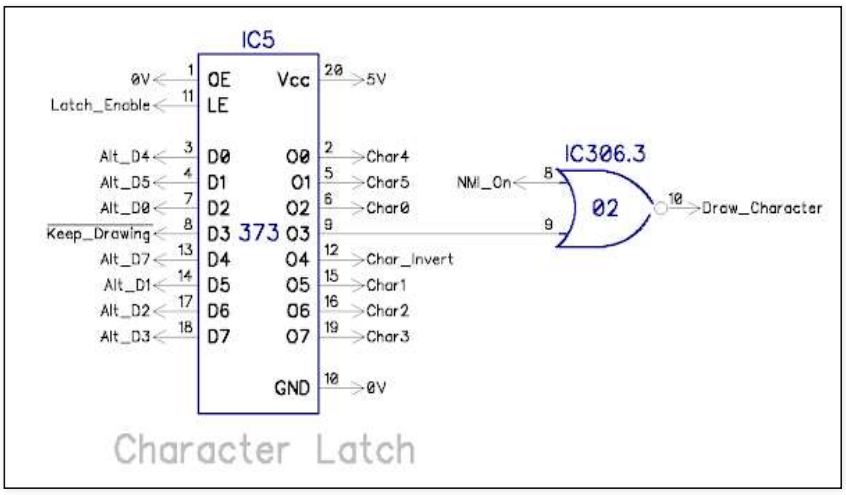


Disabling VSync Generation

When blank lines are being generated and user code being run, the VSync generation circuitry is disabled. An AND gate is added to stop a user keyboard read from starting a VSync pulse in the middle of a line.



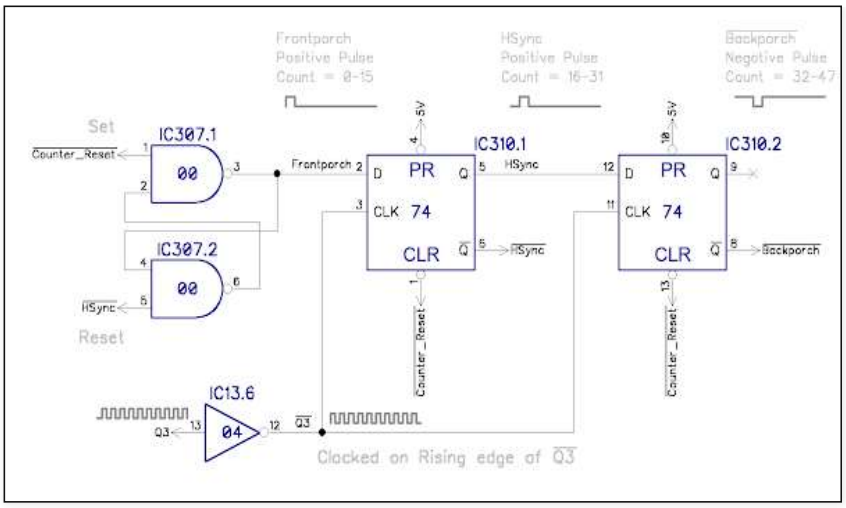
A NOR gate is used to interrupt the signal which loads the shift register, so the output remains low, which translates to a white line on the screen.



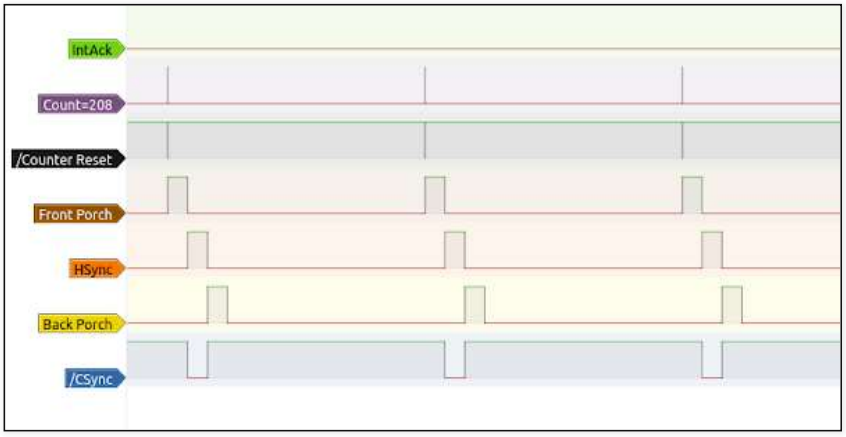
The HSync signal generated by the old hardware is no longer used, so in the Minstrel 3 I omitted the circuit. That means only VSync is used as the save signal now, with no glitches caused by the HSync circuitry.

Back Porch

On the Minstrel 3, I added an extra flip flop to the circuit that generates the HSync pulse. This is an additional 16 cycle pulse directly after the HSync, and is used to create the missing back porch of the video signal. On the Minstrel 2, this was generated using an RC circuit, but I prefer the this approach.



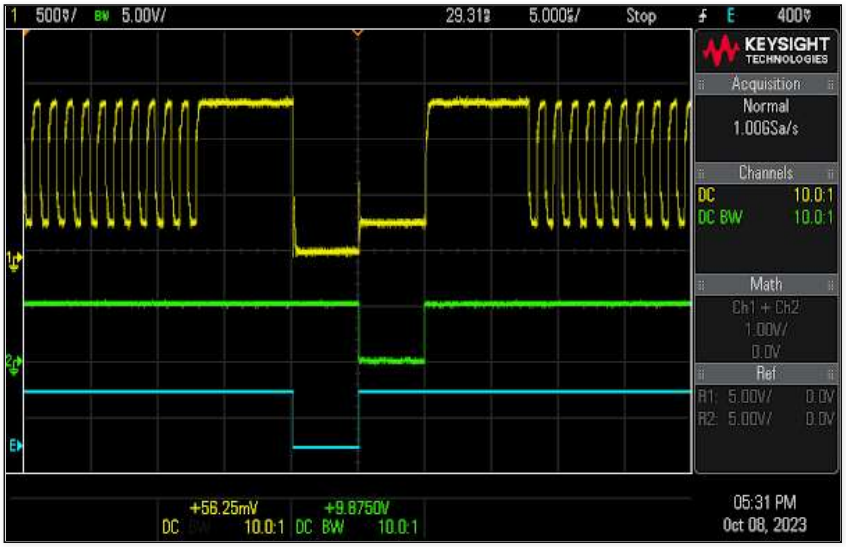
Here you can see the three pulses generated at the start of each line.



The timings are close enough to the requirements, so that works well.

Signal	Ideal Timing	Actual Timings	Cycles
Horizontal Sync	4.70µS	4.92µS	16
Back Porch	5.80µS	4.92µS	16

Here you can see the /Hsync in blue and the back porch in green and the resulting composite video signal in yellow at the end of a line.



The later ZX81 2C210E ULA also had a back porch signal, probably generated in a similar way.

The first pulse on the above schematic is marked as the front porch, although it is not used. That could have been generated and ANDed with the back porch signal to generate a proper front and back porch, but in practice, it does not seem to matter. The timing would also need adjusting. The front porch should be approximately 1.65µS, so that 16 cycle pulse would need to be gated with the lower bits of the times to make it only active for the last 4, 5 or 6 cycles of the 16 cycle pulse.

Signal	Ideal Timing	Actual Timings	Cycles
Front Porch	1.65µS	4.92µS	16
Alternate 1	1.65µS	1.23µS	4
Alternate 2	1.65µS	1.54µS	5
Alternate 3	1.65µS	1.85µS	6

Other Minstrel 3 changes

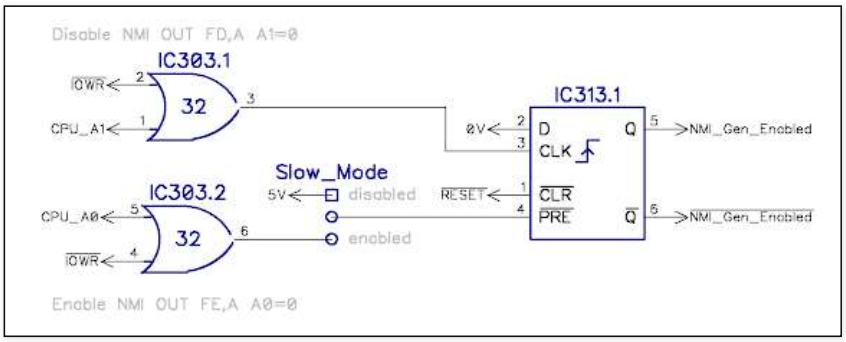
The Minstrel 2 followed the ZX80 schematic quite closely, other than the improved composite video circuitry tacked onto the end in place of the RF modulator, and the modern ROM and RAM chips.

The Minstrel 3 deviated quite considerably. I started with the Minstrel 2 version of the ZX80 schematic and added in all of the above changes.

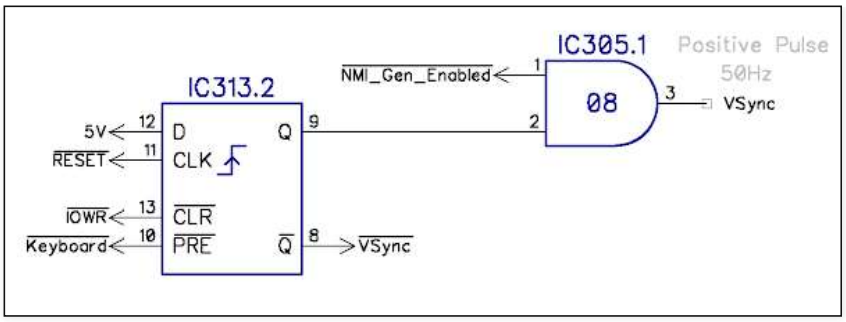
A big change was the switch from 74LS series TTL chips to the high speed CMOS 74HC series. This cleaned up all the signals, and reduced the current consumption by about a third.

There were also two new jumper options.

The first disables slow mode to make the circuitry act like the ZX80. It stops the NMI generator latch being triggered, so slow mode will never be enabled.

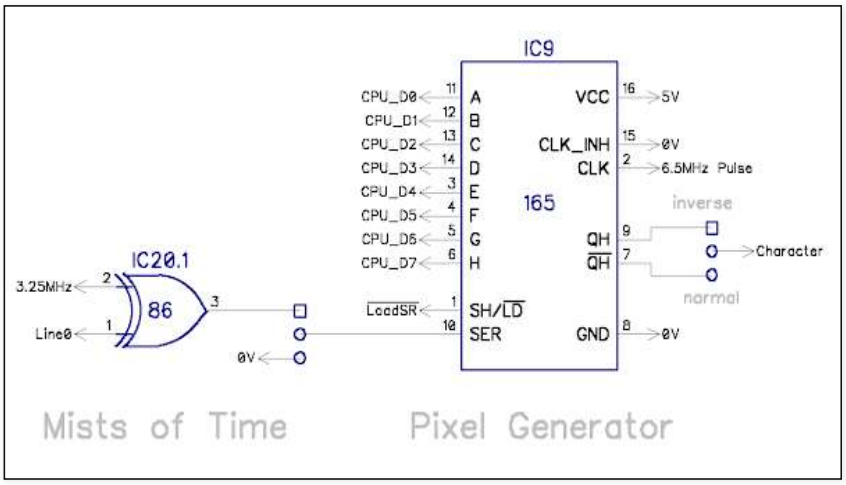


This has also been reimplemented with a 74HC74 flip flop, so the power on state can be controlled by the reset line. The same has happened with the VSync flip flop.



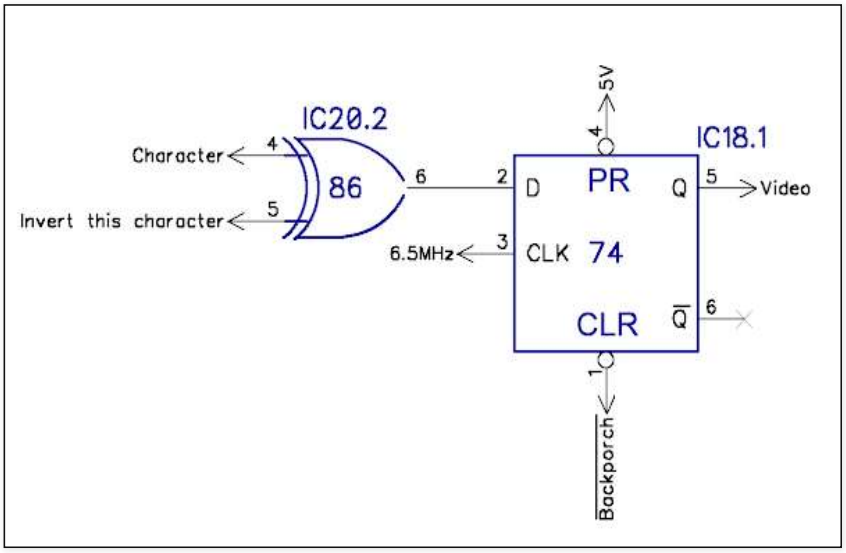
This helps when using the ZX80 4K ROM on the Minstrel 3 hardware. The 8K ROM starts with **OUT ($FD),A** which disabled the NMI generator, but the 4K ROM knows nothing of it. VSync starting on just means the composite video output with start low for a clean first frame.

The second creates a grey-scale border by feeding a chequerboard pulse into the shift register input which is used when no data is being clocked in, when no characters are being displayed. This is the clock pulse XORed with the lowest bit of the line counter so it alternates each line. As seen in the previous post, this gives an insight into the unexpanded display file.

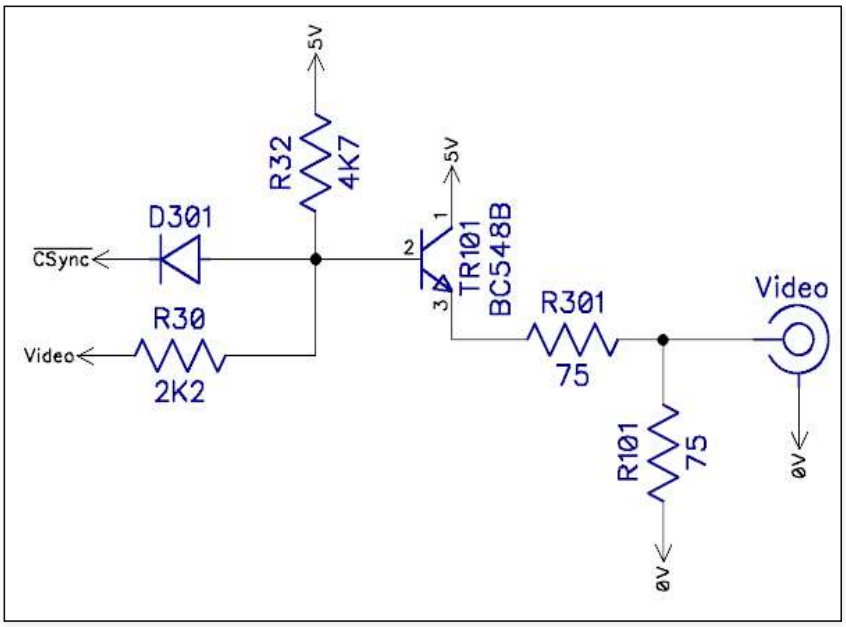


There are also jumpers to select normal / inverse video and 50Hz / 60Hz frame rate as in the Minstrel 2, and jumpers to select the upper address lines to allow multiple ROM images to be used.

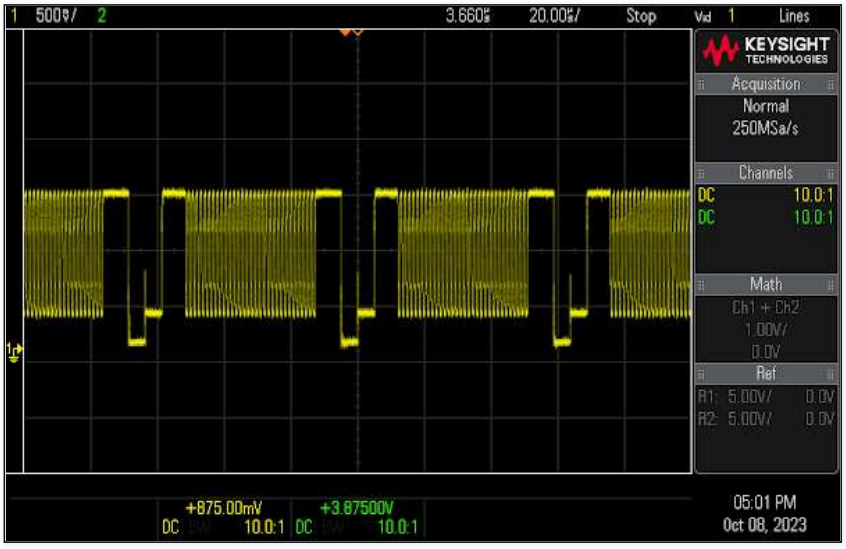
As was later added to the Minstrel 2, a character synchronisation flip flop is used to tidy up the video signal so it always generates equally sized pixels. The clear input of the flip flop is used to add the back porch signal.



A composite video mixer is used to provide a reasonably standard composite video signal.



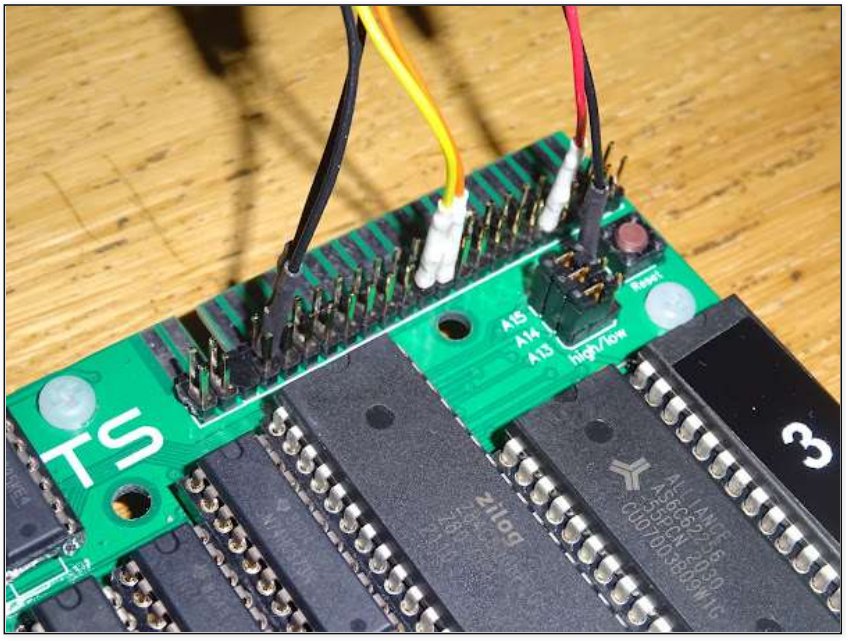
Here showing a screen full of chequerboard characters.



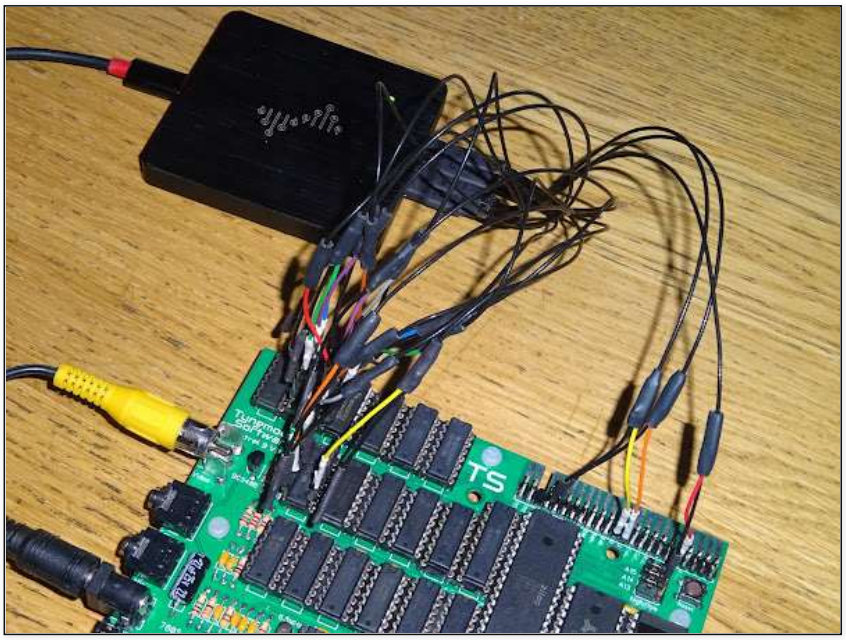
For details of other changes, including the mysterious "high resolution graphics enabling device", see a previous post:

http://blog.tynemouthsoftware.co.uk/2019/12/developing-the-minstrel-issue-3.html

The latest revision of the Minstrel 3 has a pin header which mirrors the signals at the edge connector. This is going to be useful for expansion cards, and also for attaching logic analysers for debugging.

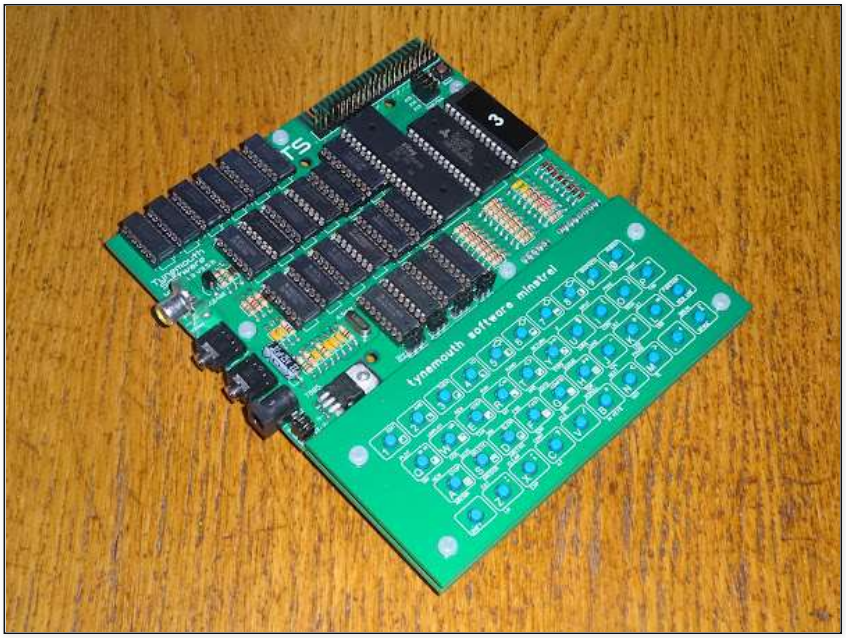


For the traces in this post, most of the signals I need are on the other side of the board, so there are a few more wires there.



Advertisements

Minstrel 3



The Minstrel 3 is a ZX81 compatible Z80 based computer with 32K of RAM and 8K floating point BASIC. It does support slow mode, and several high resolution mechanisms, so the majority of ZX81 games will run fine.

Minstrel 3 kits are available from Z80kits.com

- https://z80kits.com/shop/tynemouth-minstrel-3/

And also in built, kit and PCB only form from SellMyRetro

- Built and tested with keyboard - https://www.sellmyretro.com/offer/details/63941
- Built and tested for ZX81 case - https://www.sellmyretro.com/offer/details/63727
- Full kit with keyboard - https://www.sellmyretro.com/offer/details/63947
- Full kit with no keyboard - https://www.sellmyretro.com/offer/details/62147
- Keyboard kit with 8K keywords - https://www.sellmyretro.com/offer/details/63943
- PCB only - https://www.sellmyretro.com/offer/details/39921
- Keyboard PCB set - https://www.sellmyretro.com/offer/details/62033
- Keyboard PCB no overlay - https://www.sellmyretro.com/offer/details/61986

See also Minstrel 2 (ZX80 compatible) and Mini PET (Commodore PET compatible) kits, more info here

- http://blog.tynemouthsoftware.co.uk/2023/09/minstrel-and-mini-pet-kit-updates.html

And Minstrel 4D kits (Jupiter Ace compatible)

- https://tfw8b.com/product/minstrel-4d-turbo-jupiter-ace-compatible-computer-kit/

Patreon

You can support me via Patreon, and get access to advance previews of posts like this and behind the scenes updates. These are often in more detail than I can fit in here, and some of these posts contain bits from several Patreon posts. This also includes access to my Patreon only Discord server for even more regular updates.

https://www.patreon.com/tynemouthsoftware

Posted by Dave Curran at 11:19

Labels: HowDoesItWork, ZX81

Email ThisBlogThis!Share to TwitterShare to FacebookShare to Pinterest

Newer Post Older Post Home

Tynemouth Software Links

- Twitter
- Patreon
- TFW8b Store
- Sell My Retro Store
- Z80Kits.com
- Game Digital Downloads

Support This Blog

- Support via Patreon
- Donate via PayPal
- Order from Sell My Retro
- Order from the TFW8B Store
- Order from Z80kits.com
- Merch store
- Buy from itch.io

BECOME A PATRON

Search This Blog

Search

Blog Archive

- ▶ 2024 (2)
  - ▶ January (2)
- ▼ 2023 (52)
  - ▶ December (5)
  - ▶ November (4)
  - ▼ October (5)
    - Building a Mini PET B Kit
    - PET IEEE-488 Diagnostics (updated version)
    - How the ZX81 Generates Video
    - Testing some new PET games on the Mini PET
    - How the ZX80 Generates Video
  - ▶ September (4)
  - ▶ August (4)
  - ▶ July (5)
  - ▶ June (4)
  - ▶ May (4)

- ▶ May (1)
- ▶ April (2)
- ▶ March (2)
- ▶ February (3)
- ▶ January (3)

- ▶ 2013 (38)
  - ▶ December (2)
  - ▶ November (2)
  - ▶ October (4)
  - ▶ September (10)
  - ▶ August (6)
  - ▶ July (1)
  - ▶ June (2)
  - ▶ May (1)
  - ▶ April (2)
  - ▶ March (3)
  - ▶ February (2)
  - ▶ January (3)

- ▶ 2012 (24)
  - ▶ December (3)
  - ▶ October (1)
  - ▶ August (1)
  - ▶ July (1)
  - ▶ June (3)
  - ▶ April (1)
  - ▶ March (3)
  - ▶ February (9)
  - ▶ January (2)

- ▶ 2011 (2)
  - ▶ November (2)

- ▶ 2010 (6)
  - ▶ September (1)
  - ▶ August (4)
  - ▶ July (1)

Labels

Commodore (151) Repair (138) PET (96) VIC20 (53) Minstrel (49) Vintage (45) ZX81 (44) Hardware (38) USB Keyboards (36) ZX80 (36) PenultimateCartridge (34) ZX Spectrum (32) 4032 (30) Arduino (27) MiniPET (26) 8032 (25) C64 (25) Minstrel 4D (25) Sinclair (25) Review (19) Atari (18) Composite Video Mod (18) Keyboard (17) Game (16) Restoration (16) BBC (15) PET microSD (15) Raspberry Pi (15) IEEE-488 (14) 2001 (13) AdventCalendar (13) 4th (12) C128 (11) RC2014 (11) PET diagnostics (10) Spectrum (10) Cartridge (9) DivMMC (9) ZX81 Clone (9) 6502 (8) Diagnostics (8) Electric Vehicles (8) Electron (8) Nissan Leaf (8) Vinatge (8) 2600 (7) 4080 (7) Acorn (7) Forth (7) SD card (7) USB (7) 4080D (6) Ace (6) C16 (6) Car (6) Development (6) Disk Drive (6) LED Clock (6) Oric (6) Rant (6) TVOut (6) Upgrade (6) Z80 (6) 8032-SK (5) Data Recovery (5) HowDoesItWork (5) Library (5) Power Supply (5) ROMRAM (5) SD2IEC (5) SD2PET (5) USB Joystick (5) ZX Spectrum +2 (5) 8250 (4) Amiga (4) Custom PC (4) Datasette (4) HTPC (4) How The Mini PET Works (4) Joystick (4) Laptop (4) MiniVIC (4) Oric-1 (4) PSU (4) ST (4) Teardown (4) 1541 (3) 3032 (3) 5200 (3) 6502 diagnostics (3) A500 (3) A500+ (3) AY-3-8910 (3) AY-3-8912 (3) Amstrad (3) Binatone (3) C64C (3) EPROM programmer (3) Etsy (3) GPIB (3) IBM 5160 (3) Kit (3) LCD Clock (3) NTSC (3) PET LCD (3) Preview (3) SD2BBC (3) SID (3) SellMyRetro (3) TV Game (3) YM2149 (3) ZX Spectrum Plus (3) ZXpand (3) 128K (2) 2001N (2) 7800 (2) 800XL (2) BBC Model B (2) Build (2) CP/M (2) CPC464 (2) CPC6128 (2) Harlequin (2) I2C (2) ICL OPD (2) Ikea (2) Lambda 8300 (2) Logic Analyser (2) MSX (2) Microcontroller (2) Oric Atmos (2) Picade (2) Programmer (2) QL (2) SATA (2) TED (2) TI99/4A (2) TS1000 (2) TZXduino (2) Tapes (2) Toastrack (2) Toshiba (2) Updates (2) Virus Scan (2) 116 (1) 1571 (1) 2600Jr (1) 3D Printing (1) 400 (1) 64K (1) 6530 (1) 65XE (1) 8096 (1) 8250LP (1) A1200 (1) A3000 (1) A500Mini (1) A600 (1) AVR (1) Android (1) Atmel (1) BBC Model A (1) BBC Model B+ (1) BIOS (1) Blog (1) C128D (1) Camputers Lynx (1) Chromecast (1) Colecovision (1) Dead Test (1) Doctor Who (1) Epson (1) Fail (1) Google (1) HX-10 (1) IBM (1) IO (1) Jupiter Ace (1) LIRC (1) MITx (1) MPF (1) MPF-I (1) Marina64 (1) Merch (1) MicroProfessor (1) Microace (1) Microdrive (1) Model M (1) MythTV (1) Netflix (1) News (1) Nintendo (1) OUYA (1) PAL (1) PIC (1) PLA Replacement (1) PS2 (1) PSION (1) PartialSuccess (1) Phone (1) Pimoroni (1) Plus4 (1) Programming (1) RISC OS (1) RS232 (1) Refurbished PCs (1) Restoration. (1) RetroPie (1) SMS (1) SMS II (1) SNES (1) SX-64 (1) Screenshots (1) Sega (1) Spanish (1) Subversion (1) TRS80 (1) TS1000. TS1500 (1) Timex (1) Transfer (1) VP-390 (1) Value (1) Valve Amp (1) Vectrex (1) Views (1) Viglen (1) Wellon (1) Windows 10 (1) Windows 8 (1) XEGS (1) XPet (1) eMac (1) ebay (1)