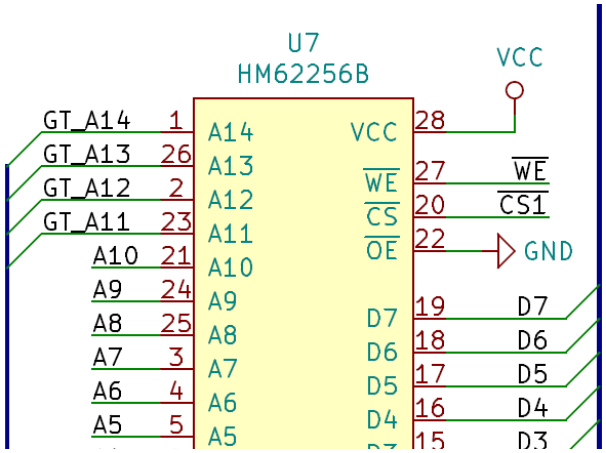
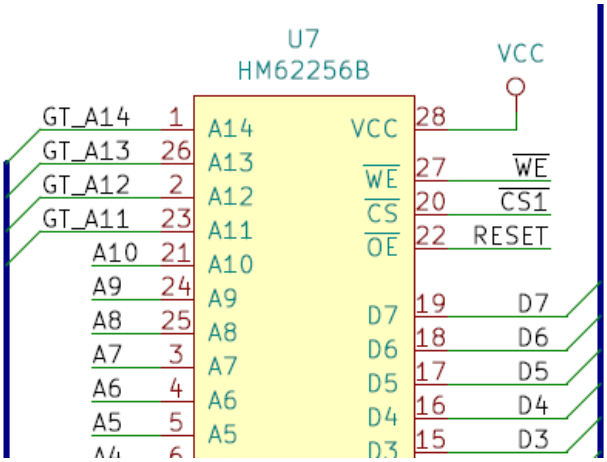
Errata 1 to EPROM Emulator NG, PCB versions 1.6 and below.

Dear users of the EPROM Emulator NG, I recently discovered a potential flaw in the design of the emulator. Refer to the diagram of the emulator posted to GitHub dated (2020-05-05), parts shown here:



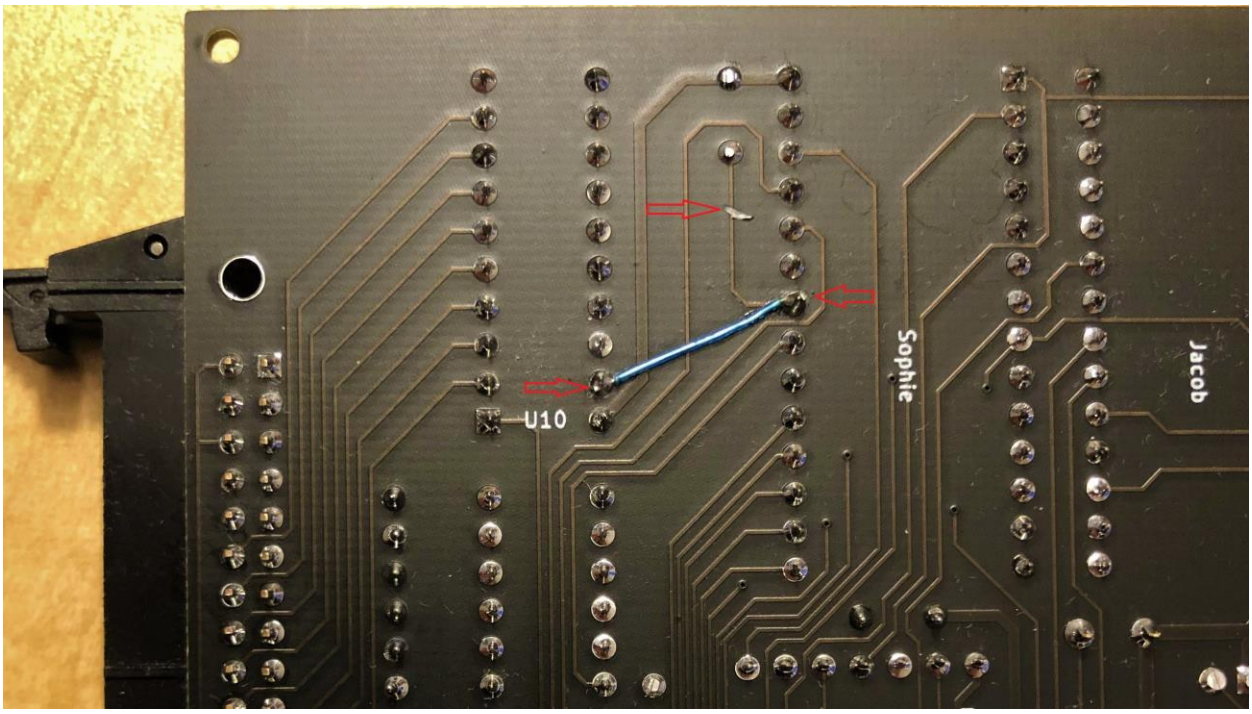
As you can see, the /OE line (pin 22) is connected to ground (GND), permanently enabling the output of the one of the SRAMs at any point of time (the other control line /CS is driven by the state of address line A15). This creates a potential issue, when the emulator is waiting for data from the computer and the shift registers are enabled, driving at the same time the data bus. This situation continues until data from computer or the SPI EEPROM is loaded into SRAM and the “emulation” starts (shift registers get disabled and external buffers - 74HC541 enabled). In my initial design and prototype of the emulator I had those lines correctly connected to the “Reset” signal, this way it guarantees only one of those can be active on the data line at any point, but while I was troubleshooting a completely unrelated issue I disconnected the OE lines from Reset and connected them to GND, I carried over the flawed design to later releases of the PCBs and only just noticed the mistake.

Interestingly this has not caused any functional issue for the many emulators that are currently in use, they are working and serving many of you very well, never the less I will publish updated diagram and new PCB design (1.7), the correct way to connect the SRAM is as follows:



New builds of the emulator should follow the corrected design and the new PCB layout. For current users I recommend doing **ONE** of the following:

Option A: Hardware modification of the PCB: very simple, cut one track to separate the /CS line from GND, and run a short link of wire to connect the /CS to RESET:



Option B: Use Arduino IDE and upload firmware 2.0rc1 or newer to the emulator. The new firmware disables the output buffers (and this way enables the shift registers) only once transfer from the computer starts (or the restore from SPI EEPROM is initiated), this way minimizing the potential for data bus "clash".

I have tested both options and they work as expected, but "**Option A"** would be **preferred** as it guarantees in hardware that data bus "clash" doesn't happen.