Building a custom ST-Link V2.1

embedblog.eu/

For programming my STM32s, I've been using almost solely those 3-dollar USB dongles from China, which are basically ST-LINK/V2 clones. They do their basic task – programming and debugging STM32s – just fine. But they are missing two very handy features:

1. they do not have the SWO pins routed out (though it is possible to modify those dongles and replace one of the pins with the SWO)
2. they do not feature the VCP (Virtual Com Port), which basically enables your ST-Link to act as an USB-serial converter; ST introduced this on the ST-Link V2.1 (and then retrospectively on the V2A and B)
3. (also, the 2.1 has the MSD, or Mass Storage Device, functionality, which enables the ST-Link to act as an USB thumb drive and any bin files you copy into this "storage" get uploaded to any connected MCUs).

Theoretically, I could modify the hardware of my existing dongles, but since it would involve a lot of tedious work and I needed to get more programmers, I decided to instead roll a batch of my own ST-Link V2.1s.

Alternatives

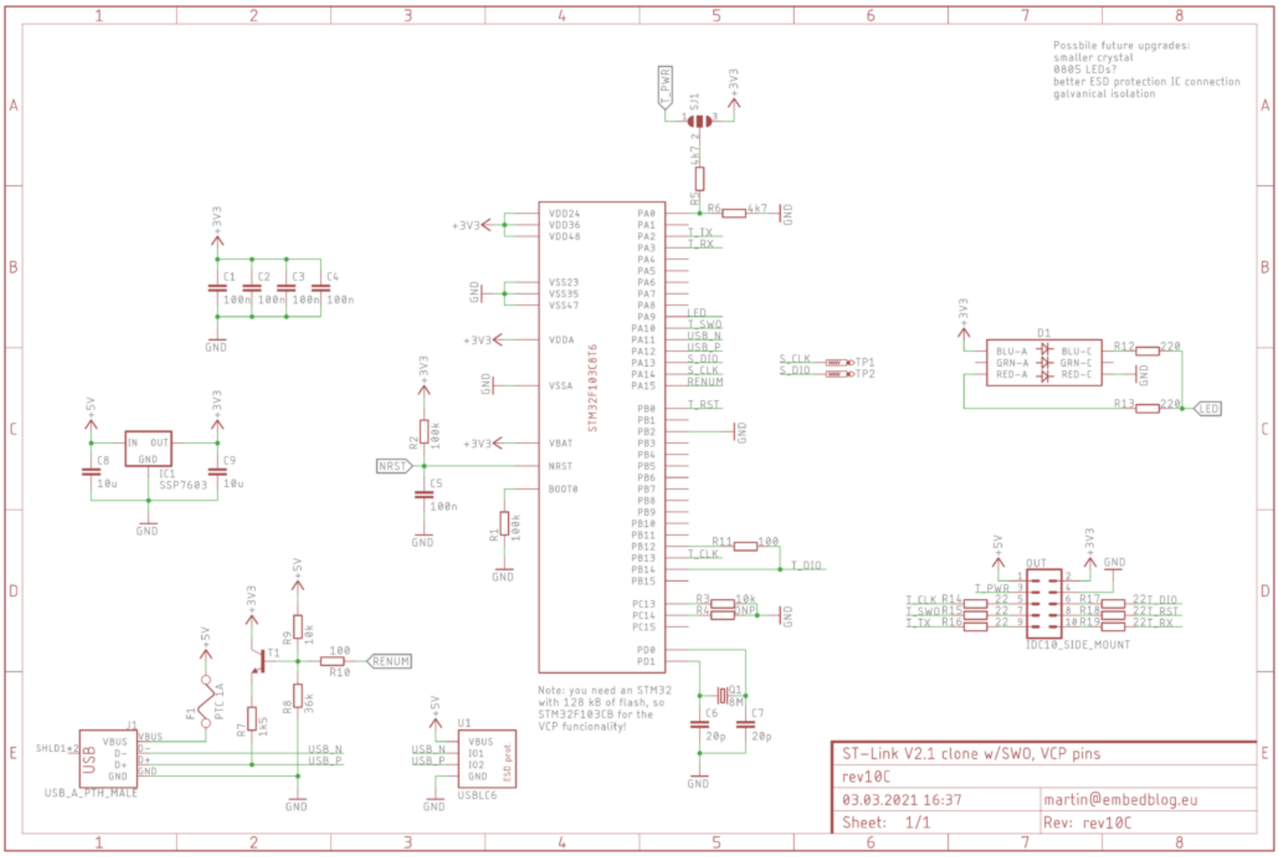
Before spending a lot of time designing something new, I had a look at existing alternatives:

1. The relatively new, official **ST-Link V3 MINI**, which is sold for 10-15 USD at places like Mouser. This was a tempting choice, but I don't like the form factor and the connector they are using. The only extra functionality this has (as far as I know) is that it can act as an USB-I2C/UART/SPI/etc. bridge.
2. **Black Magic Probe** – this is an open-source ST-Link firmware alternative, which does work even on the cheap Chinese USB dongles. The advantage is that it works with any JTAG/SWD chip (not just ST's), the disadvantages is that it's not out-of-the-box compatible with ST's tools (like the CubeProgrammer and IDE) – though there are relatively simple workarounds.

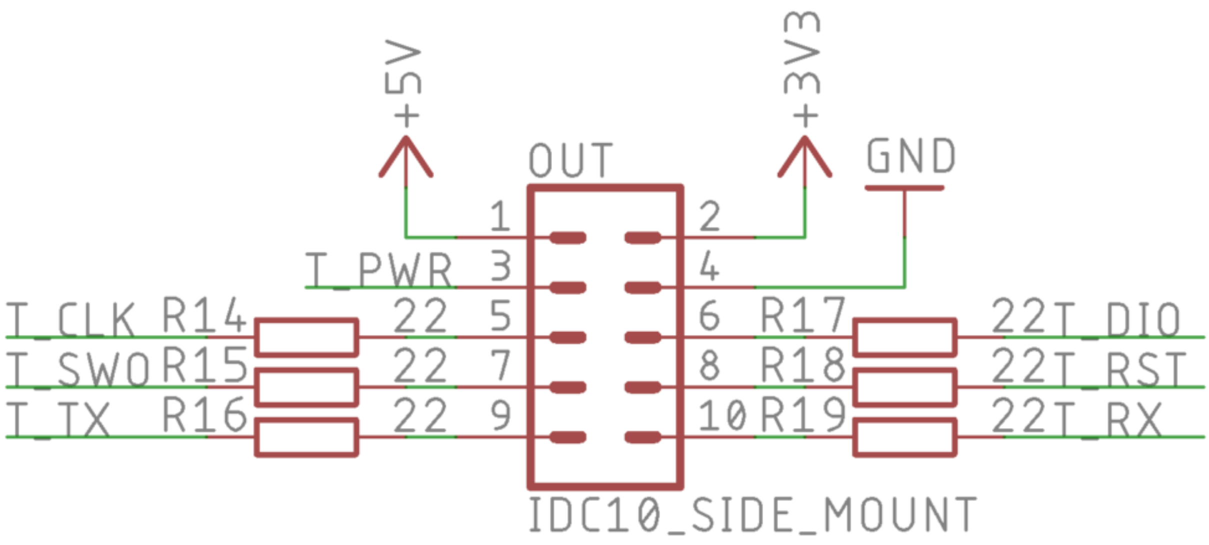
Anyways, in the end I decided to go the DIY route, mostly for the sake of just seeing if it will work.

Hardware

The hardware itself wasn't hard to figure out – there are plenty of schematics available online, just make sure you have the 2.1 schematic (an easy way to tell is that it has the the renumeration transistor on the D+ USB line). The only question is which MCU to use, since it is possible to use either the STM32F103C8 with 64 kB of flash or the CB version with 128 kB. To complicate matters further, some of the C8s on the market are counterfeits with 128 kB of flash. Anyways, the difference is that if you want the VCP functionality (ie your ST-link will also work as a USB-serial adapter), you need an IC with 128 kB of flash. This is the final schematics, inspired mostly from this picture:


Possbile future upgrades:
smaller crystal
0805 LEDs?
better ESD protection IC connection
galvanical isolation
STM32F103C8T6
Note: you need an STM32 with 128 kB of flash, so STM32F103CB for the VCP funcionality!
SSP7603
USBLC6
IDC10_SIDE_MOUNT
USB_A_PTH_MALE
ST-Link V2.1 clone w/SWO, VCP pins
rev10C
03.03.2021 16:37
martin@embedblog.eu
Sheet: 1/1
Rev: rev10C

As you can see, I also created my own pinout for the IDC 10 connector:



This 10 pin connector has all the functionality I need – 5 V and 3.3 V for power, TPWR to measure the DUT voltage (or in case I ever use a level translator or isolator), both SWD pins, SWO pin, RST pin, and two UART pins. All the pins have 22R resistors as protection, I've also seen some people use 33R PTCs – that might be an interesting upgrade.

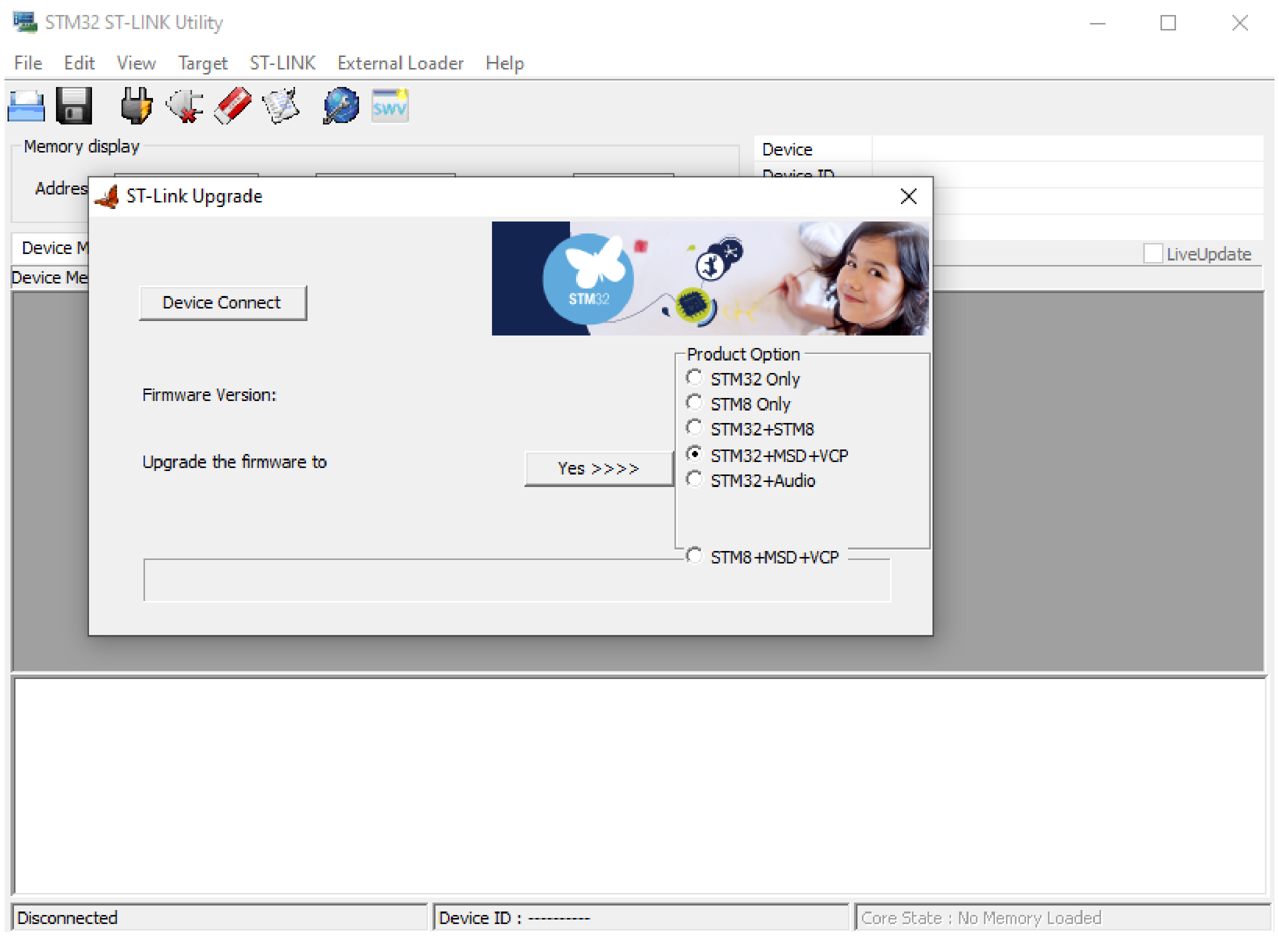
I also added a USB protection IC and a jumper, which enables to you select between measuring the TPWR pin or just the 3.3 V rail. I wanted to use a PLCC4 RGB LED, but I screwed up, so I ended up using two 1206 LEDs instead – but it works just fine. This mistake is fixed in the Github repo.

I laid out everything on a 4 layer PCB with a size of 15 x 35 mm, which is rougly the size of the clone ST-Link. Fitting everything on such a small PCB wasn't easy, so there was no remaining space for a screw hole. Also, if you are building this, consider using a smaller crystal than the HC-49 I used – it is relatively tall and makes the casing unnecessary thick.

Firmware

This was the trickier part – ST of course does not provide the firmware, since they consider it proprietary. Basically an ST-Link has two firmware components – the bootloader and the firmware itself. One approach is to somehow copy the entire firmware from another ST-Link and upload it to the new one, but I used a different approach – I wanted to download the bootloader and then use ST's own update utility to upload the firmware to the chip.

So after downloading and testing countless .bin files from weird Russian sites, I finally found a repo on Github with the bootloader for the 2.1. I uploaded the *Unprotected-2-1-Bootloader.bin* file to address 0x08000000 using another ST-Link, reconnected it from the USB and then used the old STM32 ST-LINK utility v4.3. In the top bar I selected *ST-LINK* and then *firmware update* and this is what I got:



Firmware Version:

Upgrade the firmware to

Product Option
( ) STM32 Only
( ) STM8 Only
( ) STM32+STM8
[✓ STM32+MSD+VCP]
( ) STM32+Audio
( ) STM8+MSD+VCP

Disconnected | Device ID : ---------- | Core State : No Memory Loaded

The first three options are straight-forward and do fit on a 64 kB chip, while the "STM32+MSD+VCP" requires 128 kB of flash (and this is what I wanted). I still haven't found out what the "STM32+Audio" option is for. After clicking *Device connect* and then *Yes*, the upload started and after a short while it was finished. However the uploaded firmware was not the newest one.

I then reconnected the ST-Link again and used the firmware update utility in the STM32Prog, which did update the firmware to the newest version (it also had an option to change type to only *Debug + VCP*). So all that was left was to print a simple 3D printed case and voilà, a fully working ST-Link clone.

Note on ST-Link firmware names: some of those FW update tools will show you the current firmware revision in the form V2JxMySz, where Jx is the STM32 programming revision (I believe it stands for JTAG), My is for the MSD+VCP revision and Sz is for the STM8 programming… and if some of those letters are missing, you do not have that this feature programmed.

Conclusion

I do consider this as a success, since it works flawlessly. All the required materials are on my Github. Having a method to create my own ST-Links either enables me to embed them into some other project or develop them further – ie I'll probably make another revision with an optical isolation.