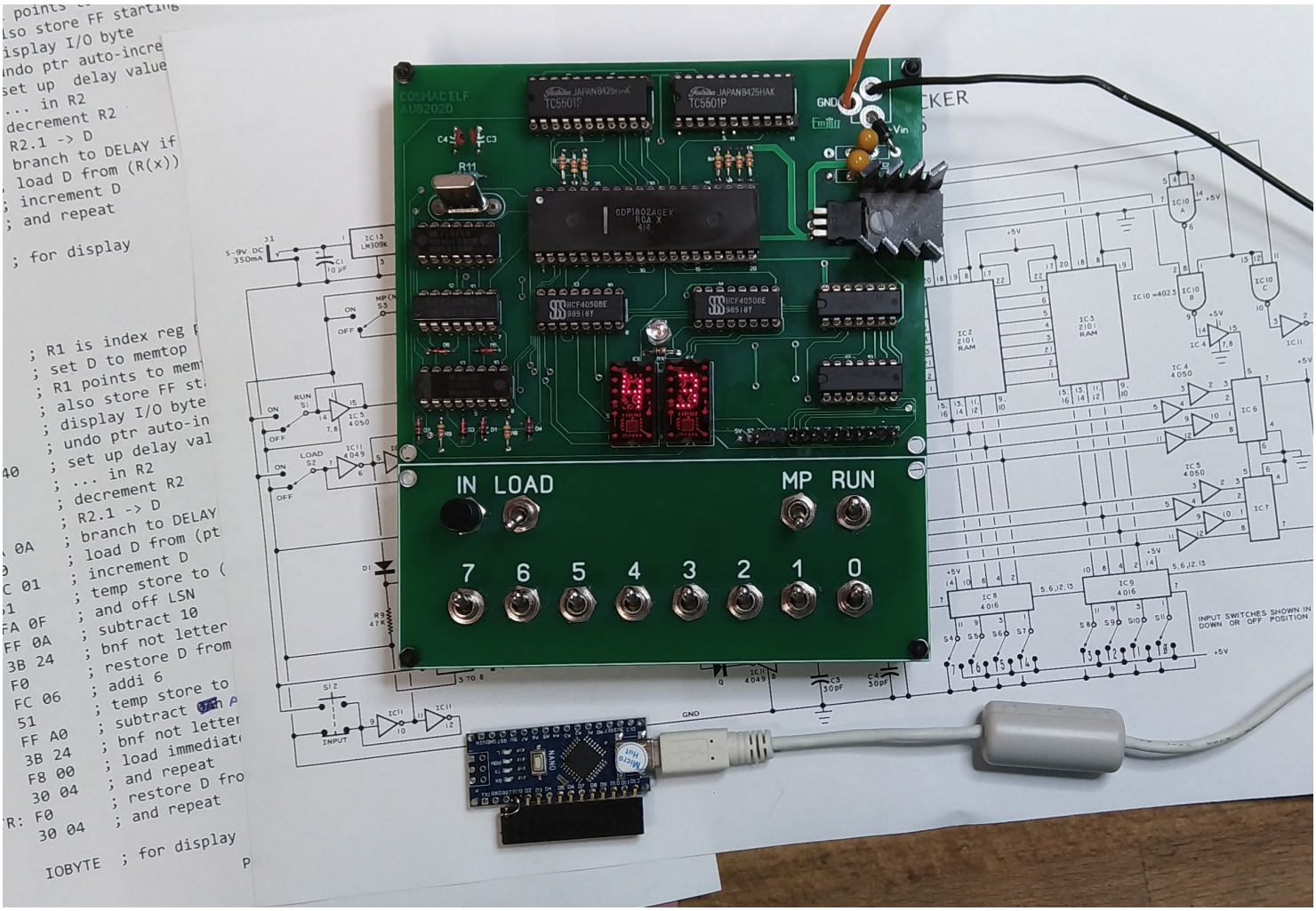
A combined Auto-Keyer and Hex Keypad for the Cosmac 'Elf' **Phil_G**



For a short while, loading programs with key-switches and LEDs is an enjoyable nostalgic experience but also a stark reminder of the bad old days :-)
As a Z80 and PIC assembler user for decades, the CDP1802 processor is new to me (45 years late...) so its slow progress but I now have an assembler (A18) which produces object code (ie machine code) files and this makes programs much easier to create, but of course the resulting code still has to be keyed in. The novelty of keying in programs in binary does fade, and some magic is needed :-)

So, in a nutshell....

This is a development of the previous Auto-Loader thingy which directly keyed in the object code produced by the assembler, making the whole process not only quicker but also a lot more pleasant :-)

This version however also sports a hex keypad which can be used to manually enter programs to memory much quicker than by toggle switches :-)

The widget connects to the Elf via the existing toggle-switch header and simply emulates all the switch throwing and button pressing - the Elf is unaware, it simply thinks your keying skills have improved! There is no 'keypad/serial' switch – both functions are alive simultaneously and have equal precedence. The unit 'disconnects' after each byte transferred and transferred bytes can be seen in hex on the Teraterm screen and on the Elf LED display.

Teraterm is used to send the object file - remember to tick 'binary' on the file send menu! The baudrate and delays are deliberately slow to make it all 'visible', of course it could key as fast as you like, but I much prefer to see whats happening, its great to watch all the hard work being done for you :-)

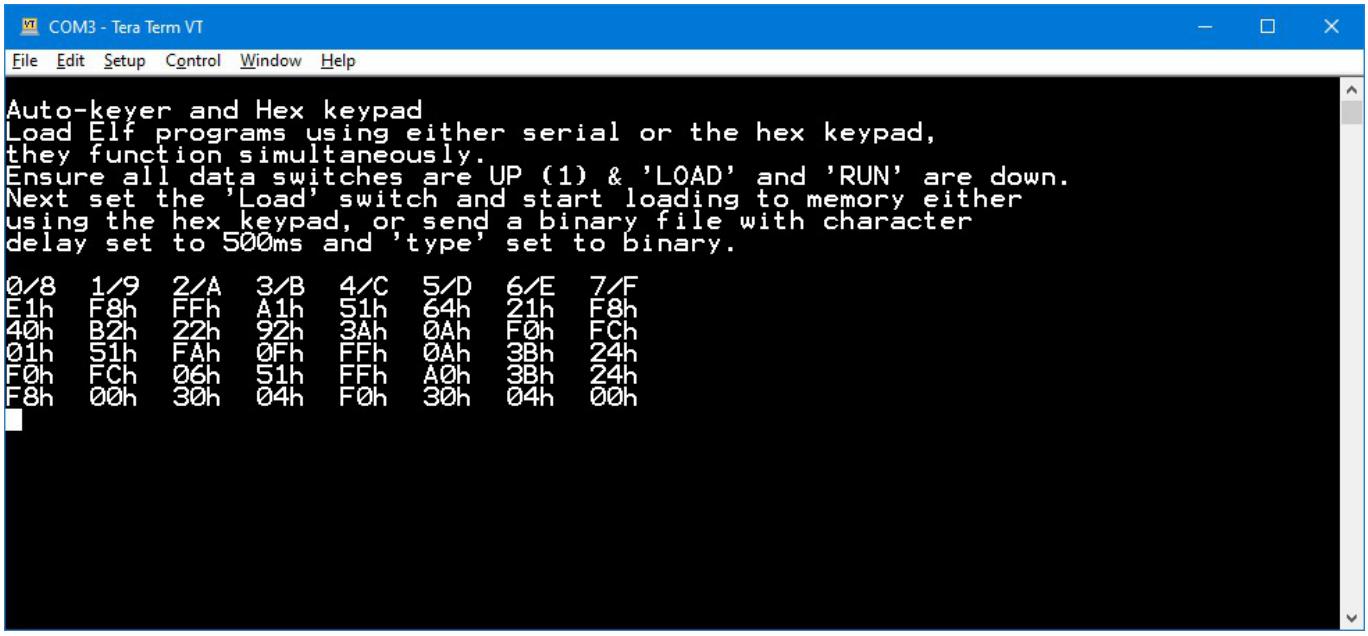
During the load process, the bytes can be seen both on the Elf display and in the Teraterm window.

Once its loaded up, we come out of 'load' mode, flick 'run' and away we go...
More on the Cosmac Elf forum: https://groups.io/g/cosmacelf/topics
and the updated Nano sketch is in the Cosmac Elf section of http://philg.uk/

I realise all this is old hat, its all been done before by various methods but I wanted it to load code immediately from reset without the need to key in bootstraps, and its just a tool for the job made using what I had to hand. It works a treat. Heres a video of the previous serial-only widget:
https://www.youtube.com/watch?v=mKJ1FKlVyFM

...and heres the new Auto-Keyer-with-Hex-Keypad:
https://www.youtube.com/watch?v=amUEFpjozSU

On powering up the adapter, the following brief instructions are displayed on the Teraterm window, and after loading for example the 'countd.bin' decimal counter file, the screen looks like this:



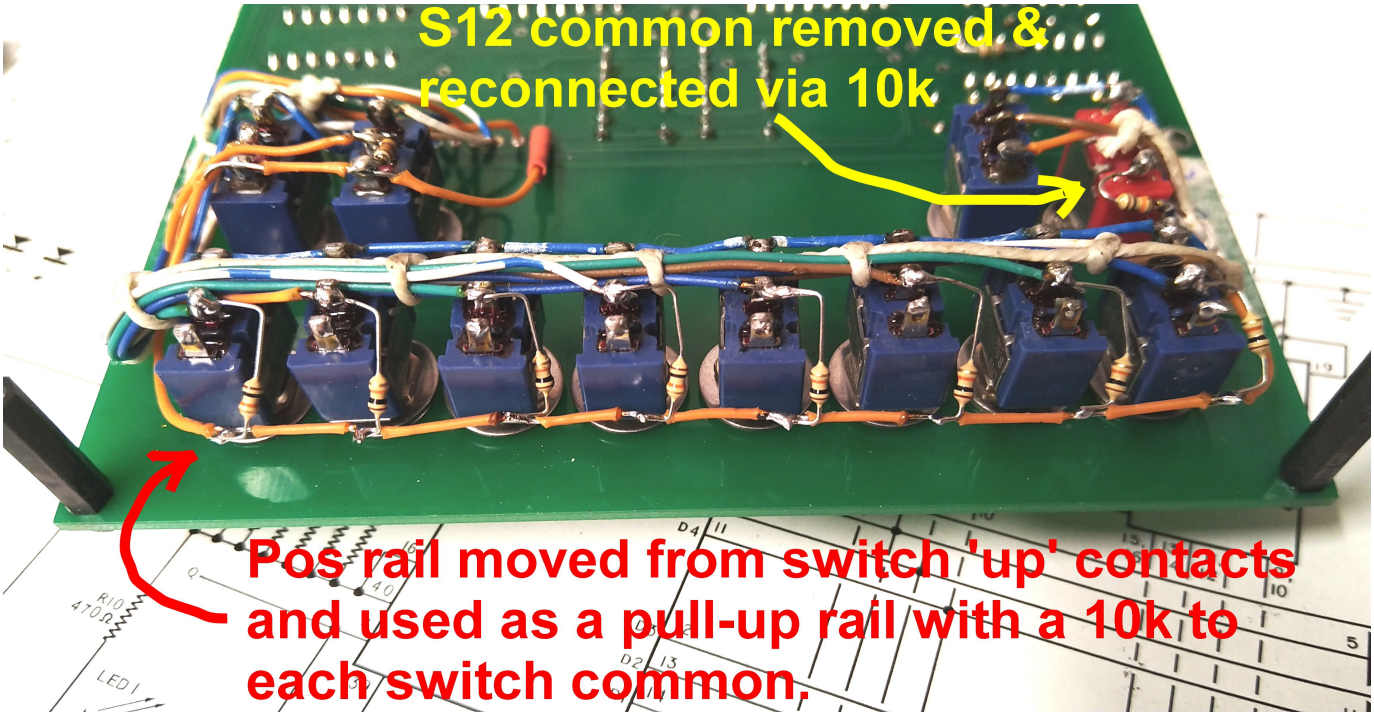
Of course the unit doesnt need to be connected to a terminal at all if you only intend using the hex keypad – these messages will still be sent but go nowhere. If you're not plugged into a PC USB port, then the device can be powered by a USB power-bank or from the Elf 5v rail.
Bear in mind that it sends its welcome message whether or not a terminal is connected - so there will be a brief delay before its ready, as indicated when the serial activity LED stays off.

A couple of little harmless changes to the Elf are necessary – these were detailed in the previous version but we'll repeat them here...

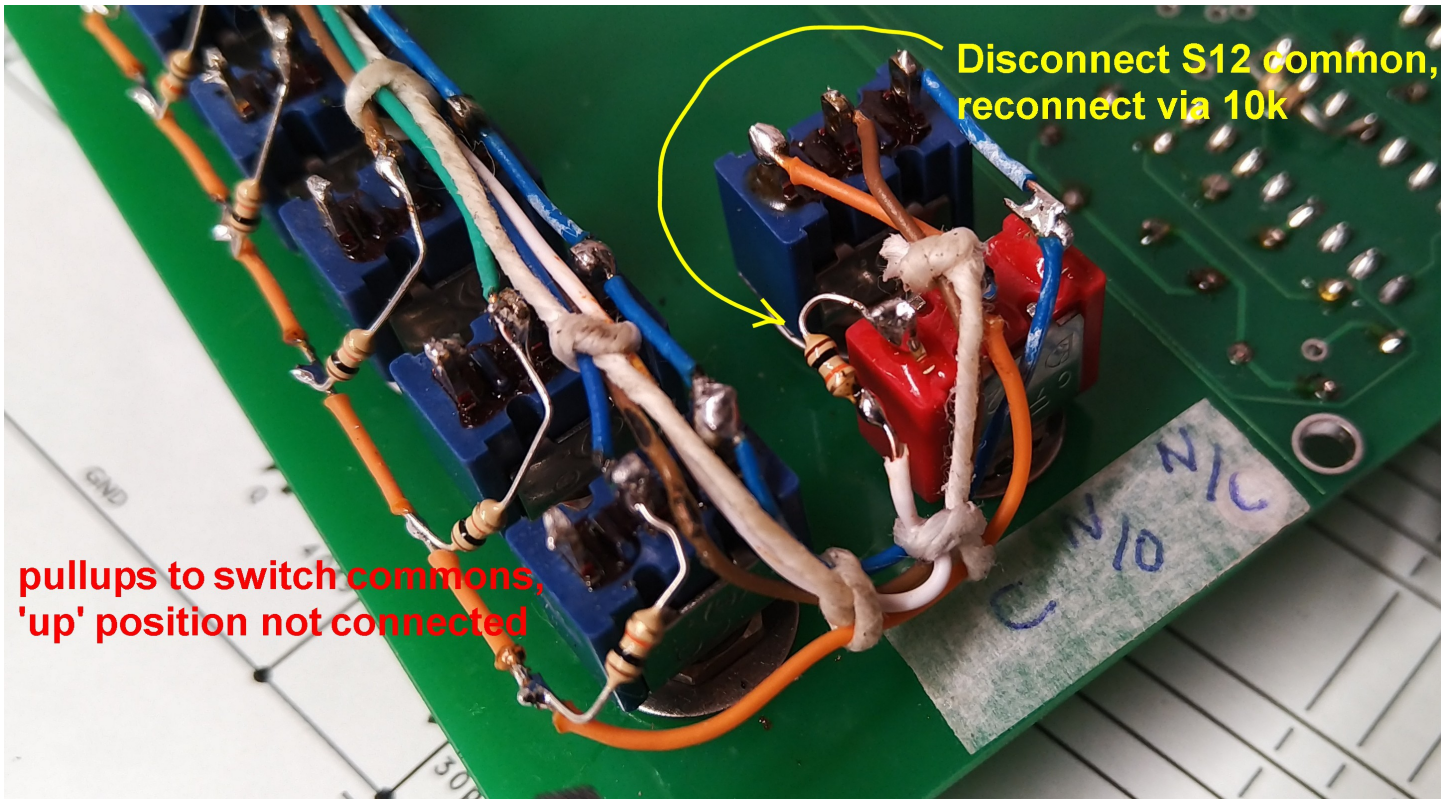
The data switches & the 'IN' button need to be converted to pull-ups, and the 'hard' feedback of the 'IN' bistable needs softening with a 1M resistor.
These mods dont affect the normal 'switch & LED' operation of the 'Elf' at all, it still works normally.

Here are a few pics of the Elf mods, they're very easy to do, just a few resistors and a header strip.

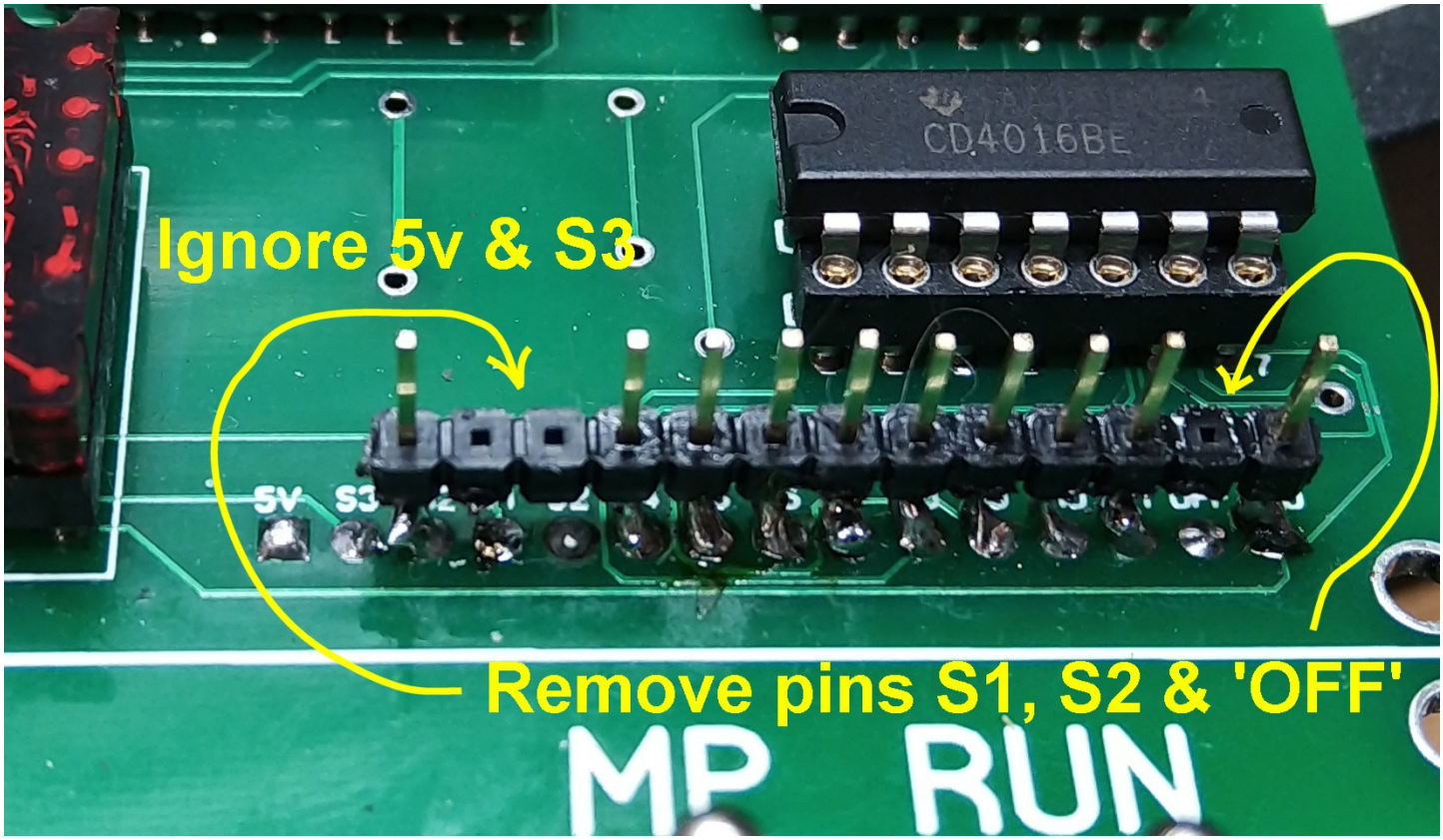
First of all, the positive rail along the 'up' contacts of all 8 data switches is desoldered and relocated across the bottom of the switches. A 10k resistor is soldered from each switch 'common' to the positive rail. The 'up' contacts nearest the camera are no longer used:



Similarly, the 'common' of the 'IN' switch (S12) is desoldered then reconnected via a 10k resistor:

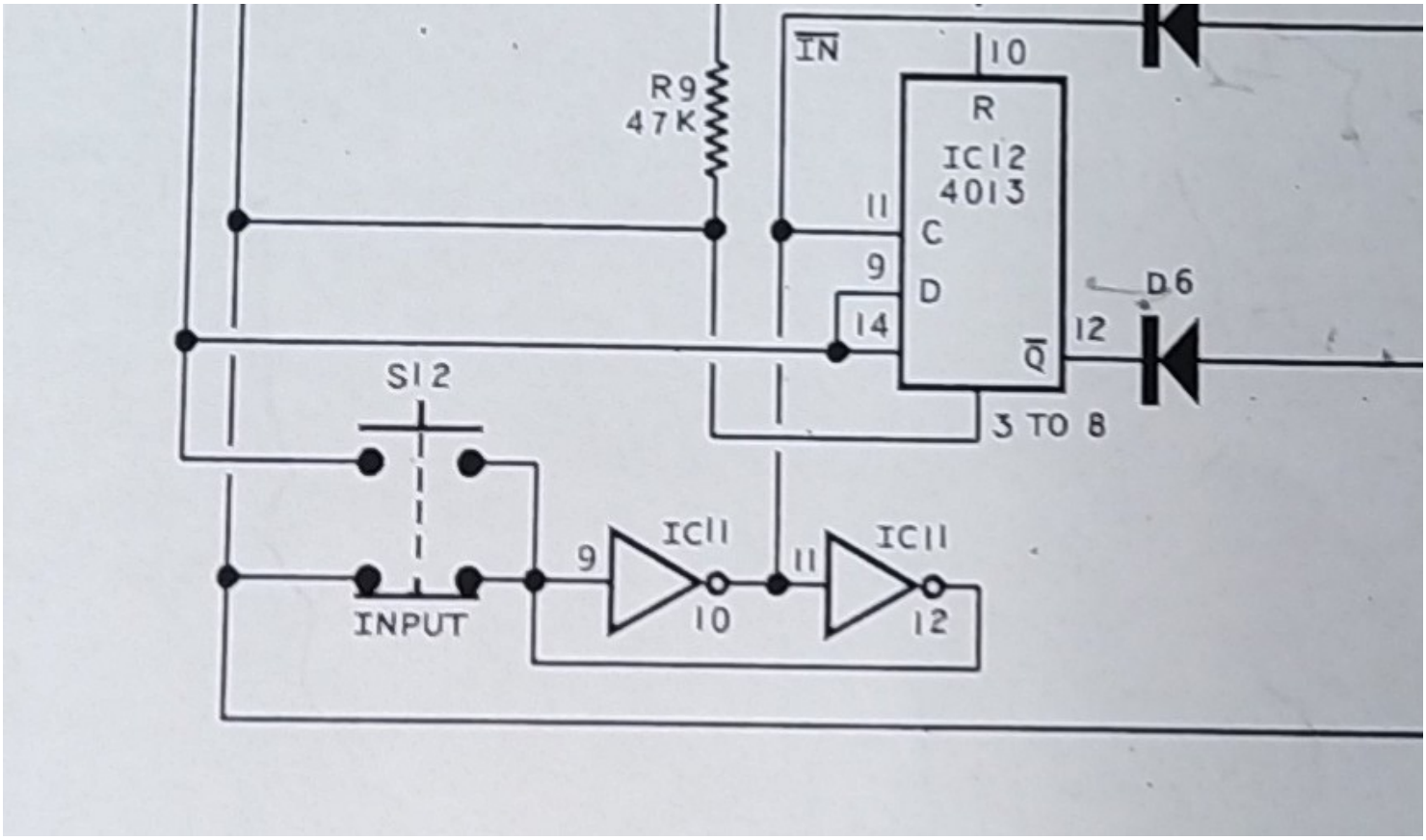


Next take a 13-way pin header and remove the 2nd, 11th & 12th pins.
This is carefully soldered to the row of switch connections on the top of the PCB. Be careful not to release the switch wire in each hole!

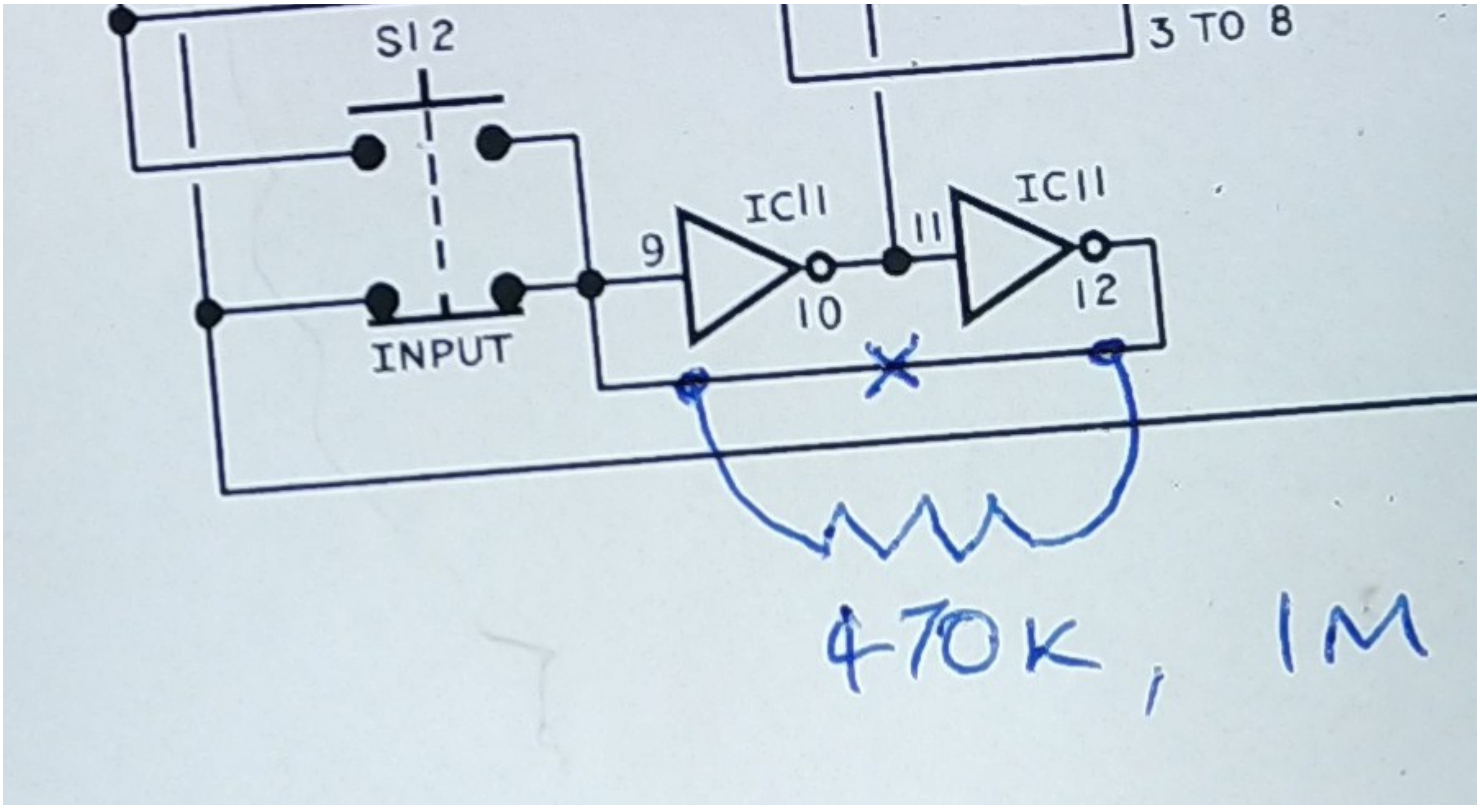


This isnt as easy as you might think, please ignore my crappy soldering!
A 90° header strip might be easier but I used what I had.
This is the header plug onto which the Nano will connect for loading. Ideally, on a new build, I would mount the header directly into the PCB. You could permanently mount the keyer under the PCB, it doesnt *have* to be on top.

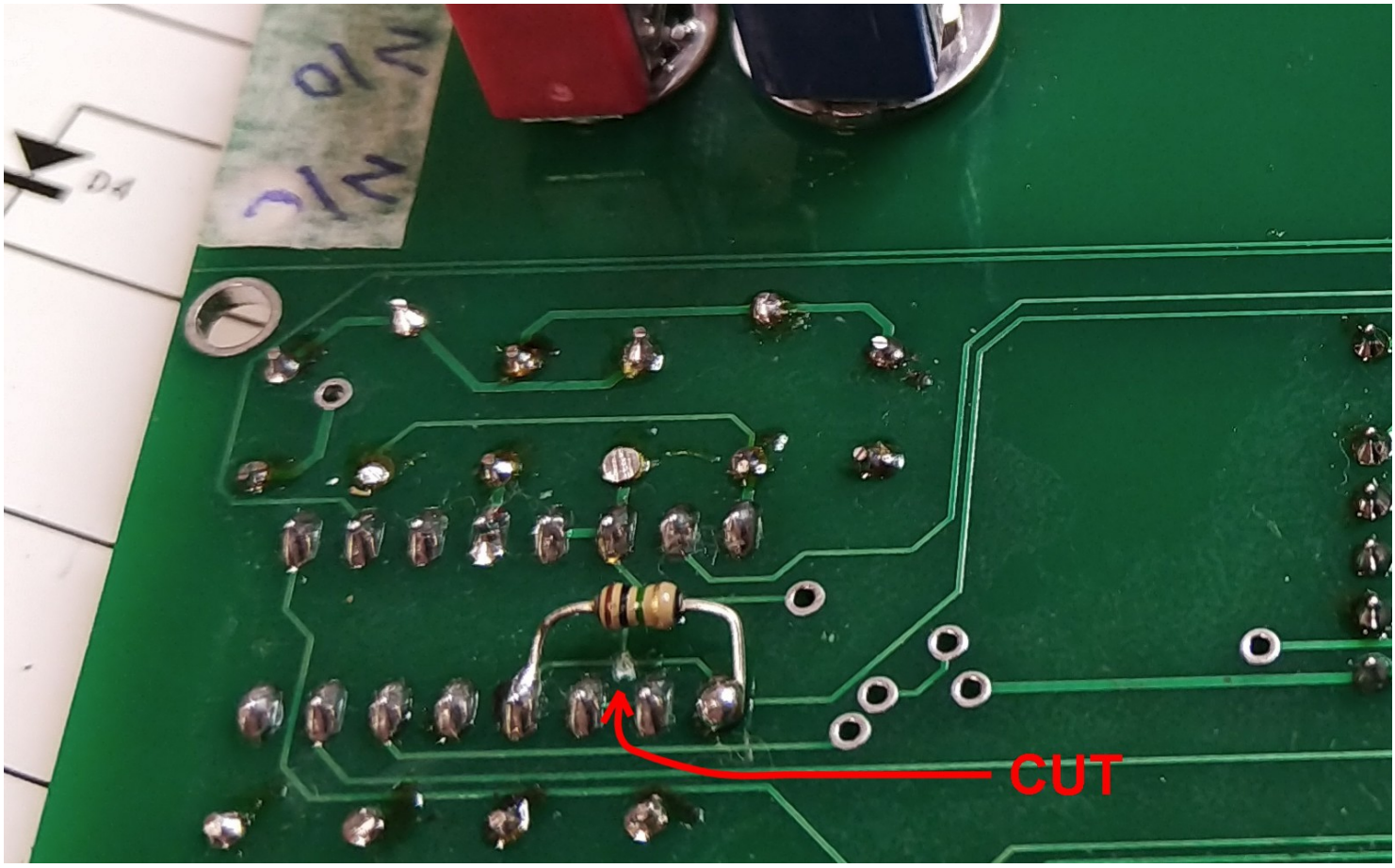
Next we need to soften the feedback of the 'IN' switch debouncer.
Here's the original, its IC11 pin 12 to pin 9:



...and heres the mod. The actual value isnt critical as long as its 'much more than 10k' :-)



...and here's how it looks. The track bridging IC11 pins 9 & 12 is cut with a scalpel and a high-value resistor soldered across those same pads:



This doesnt change the operation of the circuit but allows the Nano to easily override the 'IN' button state, which is necessary for the serial-keyer and the keypad.

To summarize what we've done so far:

Each of the data switches is now a 'make' contact to ground (low), with a pull-up when that contact is broken (high).

The 'IN' switch (S12) now has a 10k in series at the switch common.

We've done the resistor mod to IC11 ('IN')

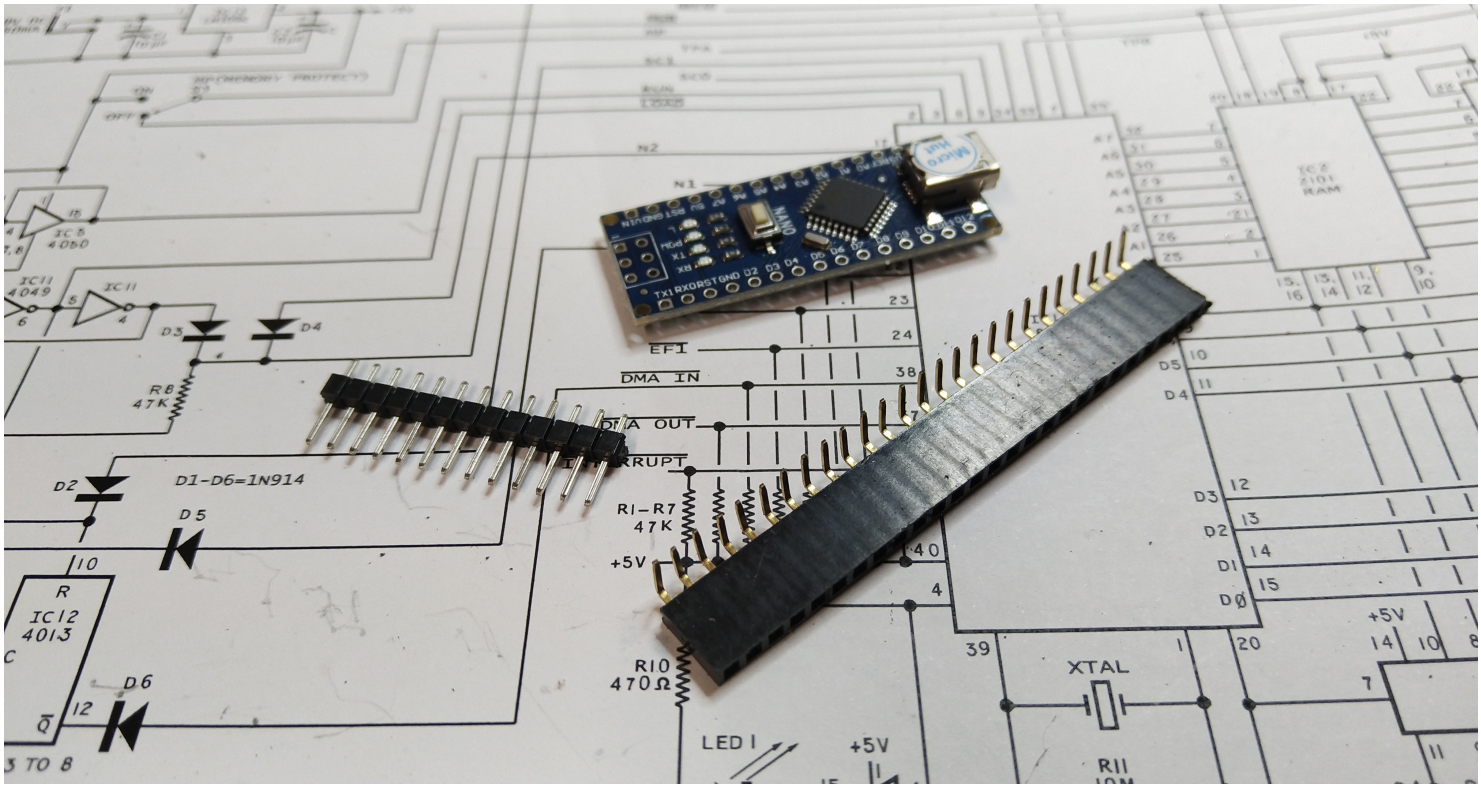
We have a header carrying from right to left:

Position 1	ground
Position 2	missing pin ('off')
Position 3	S11 (data bit 0 toggle switch)
Position 4	S10 (data bit 1 toggle switch)
Position 5	S9 (data bit 2 toggle switch)
Position 6	S8 (data bit 3 toggle switch)
Position 7	S7 (data bit 4 toggle switch)
Position 8	S6 (data bit 5 toggle switch)
Position 9	S5 (data bit 6 toggle switch)
Position 10	S4 (data bit 7 toggle switch)
Position 11	missing pin (S2)
Position 12	missing pin (S1)
Position 13	S12 ('in' pushbutton)
Position 14	missing pin (loader doesnt use S3)
Position 15	missing pin (5v, can optionally be used to power the Nano)

That completes the Elf mods, you might want to check at this stage that it still works as standard by keying in and checking a few bytes.

The loader

For the loader itself, take an ordinary Nano clone, my personal preference is for the CH340G versions rather than the FTDI ones, but either will do.



Fir a removeable adapter, take an 13 way header-socket row (you'll probably have to trim a longer one down) and using fine-nosed pliers, pull out pins 10 and 11 - just remove them completely.

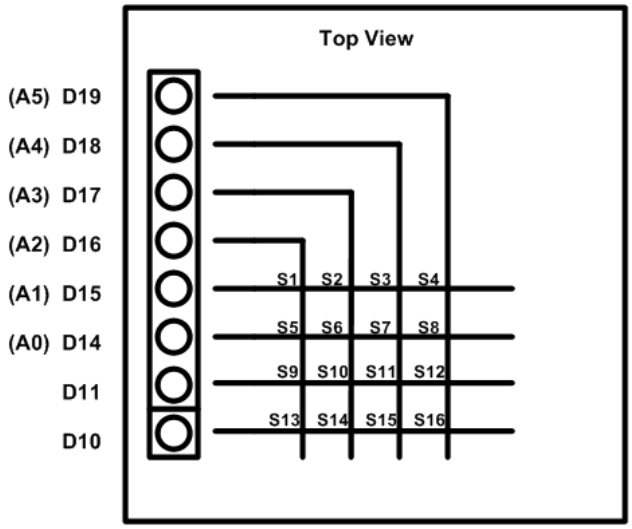
Bend the rightmost pin back on itself. This will plug onto the ground pin on the PCB, but doesnt align with the Nano ground, so bend it clear whilst the remaining pins are soldered.

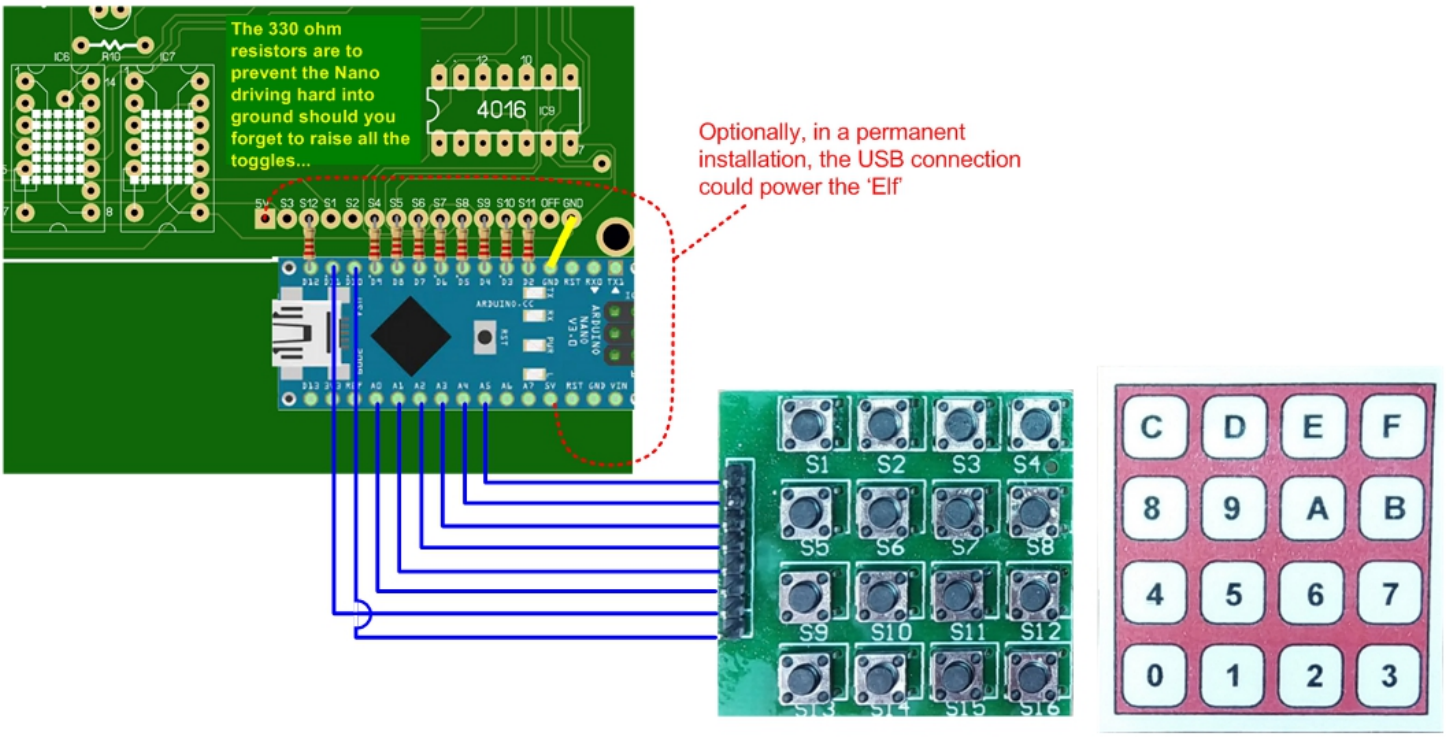
One for one, connect D2 to D12 to the header socket, either directly as I did, or preferably via a 1/8w 330Ω resistor for each (see next page)

You'll see from the D outputs used that all the required nano outputs align conveniently with the PCB, that is 'IN' and input switches 0-7, with the exception of the ground pin, which is one pin out-of-step with the Nano, hence the bridge on the socket (see photos below).

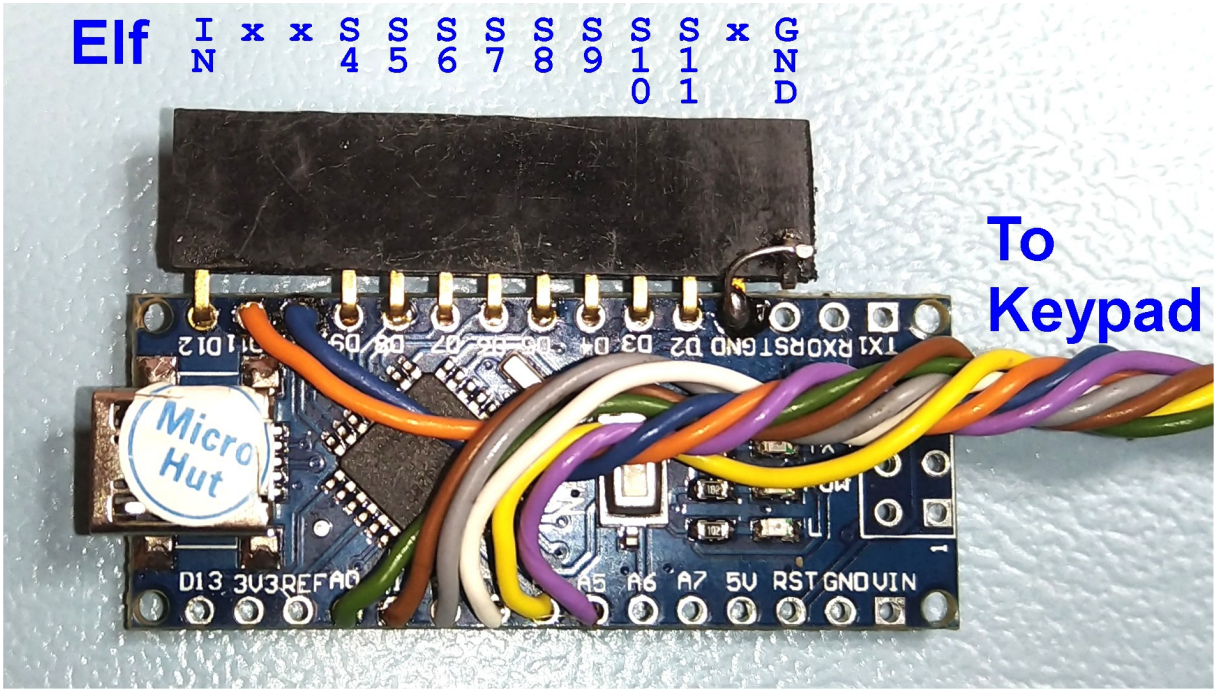
Hex Keypad:

The pad is a simple 4x4 array, connected as follows:

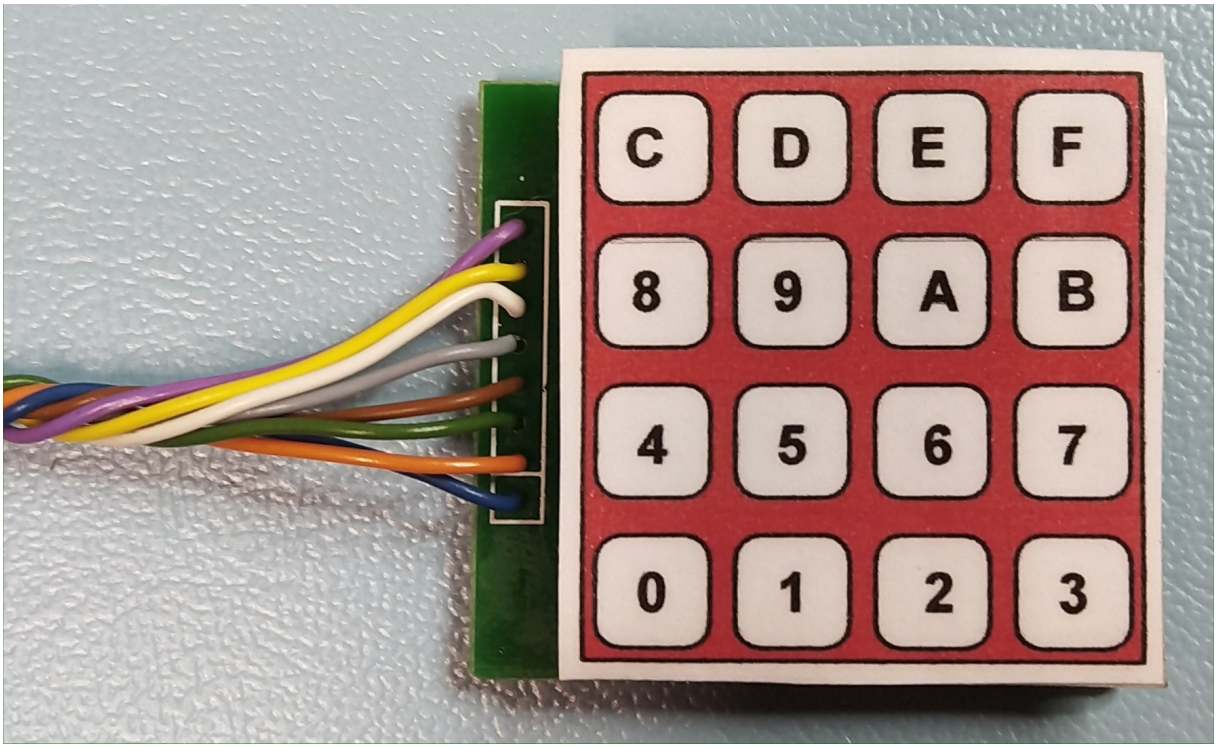








At the very right-hand end of the socket, take a short length of snipped -off resistor and solder the bent-back 'ground' pin to the Nano ground as in the pic above. The point is it should not connect to 'RST' :-)



The Nano can remain in place whilst the Elf is in use, or could even be permanently wired, possibly under the PCB out of sight. Previously I'd not bothered with the resistors on the grounds that a Nano costs about a pound, but on a permanent installation, its possible or even likely that you forget to keep all the data toggles 0-7 upwards - when down, they represent a hard-ground into which the Nano could be trying to assert a logic one. The resistors protect the Nano outputs - whilst the data toggles still have to be raised, with these resistors in place, the Nano wont see a short to ground if any of the toggles are left down to ground. To cut a long story short, on a new build I'd use the resistors.
In a permanent installation I'd also probably increase the baudrate to minimise the 'Welcome' delay.

Thats it!
I covered the Nano with clear heat-shrink then trimmed around the socket.

Operation:

Flick all the Elf data switches up to '1' and keep 'load' and 'run' off (down).to reset the Elf.
MP of course should be off (down) as we're writing to memory.

Serial loading:
Plug the USB port into the Nano and connect it to the toggles header, ensuring the pins are all aligned to the socket, and not 'Bobby's Kelly' (*Bobby's Kelly refers to buttoning up a jacket then realising you missed the first button-hole and all the rest are staggered*) (*maybe its a Yorkshire thing!*)

Connect the USB to the computer and a COM: port should appear. If you dont have the CH340G drivers, here's a tutorial: https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers/all
Flick the 'Load' switch up to put the Elf into program loading mode.

Fire up Teraterm. Choose the appropriate COM: port assigned to the Nano's CH340G or FTDI and on the 'Setup', 'Serial Port' menu, set it to 300 baud with a character delay of 500ms, then click 'New Setting' to apply these changes. If you wish the 300bd/500ms profile can be saved so Teraterm starts with that configuration.

If the Nano doesnt automagically reset, briefly press its reset button.
On the Teraterm screen you'll see a welcome message with brief instructions, reminding us to use say 500ms character spacing and binary transfers, it then prompts us to send the file.

From the Teraterm 'File' menu, choose 'Send File'.
Choose the object code (machine code) file you want and tick the 'Binary' box below.
Note that this is 8-bit raw data, NOT hex. Its generally a .OBJ or .BIN file.

You should see the Elf burst into life and if you have a program listing and you're quick, you can follow the bytes being entered on the display – and also displayed on the Teraterm screen.

For every byte entered the D13 LED on the Nano flashes in sympathy with the 'IN' presses and when this stops, the transfer process is complete.

Hex Keypad loading:
Power the Nano USB either by plugging into your PC USB port, or to a battery power bank, or via the Elf 5v rail. It sends its welcome message regardless, so there will be a brief delay before its ready.
Flick the 'Load' switch up to put the Elf into program loading mode, and start keying your pairs of hex characters, two keypresses to each byte of course. After keying a byte, it will appear on the Elf LED display exactly as it would when using the toggles.

The hex pad automagically pulses 'IN' every 2nd keystroke - ie every byte so theres no need to use the 'IN' key at all.

Bear in mind the standard Elf is a ROM-less system, to advance through memory it relies on the 1802's DMA increment so unfortunately if you do make a keying mistake, the recovery is exactly the same as the standard toggle-switch Elf – that is to reset, press 'IN' to step through memory with memory protected to the byte immediately before the bad one, unprotect memory, key in the new value and continue from there. Its a convoluted process that teaches you not to make mistakes :-)

After the load is complete, via serial or hexpad:
After each transfer the Nano ports are set to input mode to effectively disconnect them from the Elf bus.

Switch out of 'Load' (switch down), flick the 'Run' switch and your program should run.

Heres a video of the whole process: https://www.youtube.com/watch?v=amUEFpjozSU

Everything related to this project is in the 'Elf' section on http://philg.uk

I think the combination of the superb Rizal/Emilio 'Elf Recreation' PCB, the Auto-Keyer and the Hex Keypad makes a nice cheap development system for learning about the 1802, its certainly easier than toggling in a lengthy program - just look at the LED blink and compare the hex-keypad load time with the toggle switch load time!

The other really nice thing about the keypad is you can enter programs using just one finger :-)

I realise that modern components in a retro environment are not to everyone's taste - although the Atmega is 25 years old - so I appreciate the preference for authenticity in such a classic design as the Elf. But we've been in lockdown and basically I'm just making use of what I already have, to help me up the 1802 learning curve, and fundamentally the operation of the bare Elf is unaffected - this is just an additional tool to save time :-)

I hope all this makes some sense, at least enough for someone to have a go at making one.
Let us know how you get on via the Cosmac Elf forum: https://groups.io/g/cosmacelf/topics
or email me with any specific queries.

Links:

Here's where the Rizal/Emilio PCB was first introduced:
https://groups.io/g/cosmacelf/topic/new_cosmac_elf_recreation_on/75960108

Here are the gerbers, free for group members to use:
https://groups.io/g/cosmacelf/files/Cosmac%201802%20Recreation%20PCB

Autoloader and hexpad forum discussion here:
https://groups.io/g/cosmacelf/topic/84366975#35164

All the docs, files, photos, etc are in the ***Cosmac Elf*** section of
http://philg.uk
& there are a few simple demo progs for the bare Elf too. The Rizal/Emilio board is an easy build, and as you'll see from the photos, gives the impression of a true PE Elf from '76. I would do the pullup and switch-softening mods from day 1 rather than later.

Cheers, have fun :-)
Phil Green, Phil_G on all the forums
philg@talk21.com
Another lockdown 2021 project

[This doc is obviously an edit of the previous serial-only version so please let me know if I've missed anything out or if theres any thing here that shouldnt be... ta!]