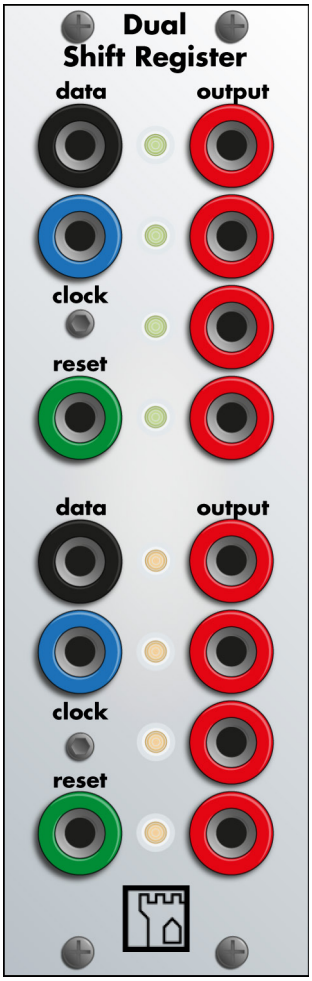
Castle Rocktronics

004 — Dual Shift Register

Two 4-bit shift registers with parralel outputs and reset

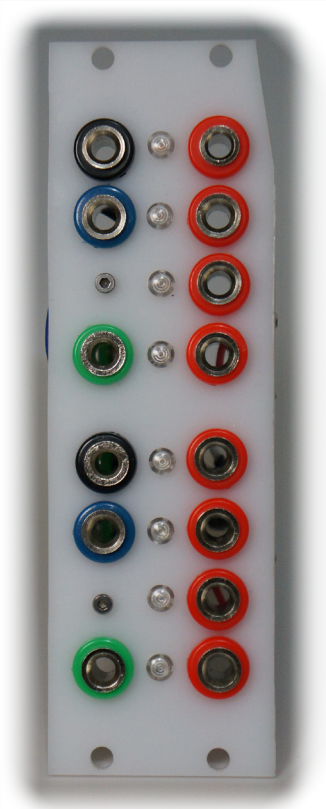


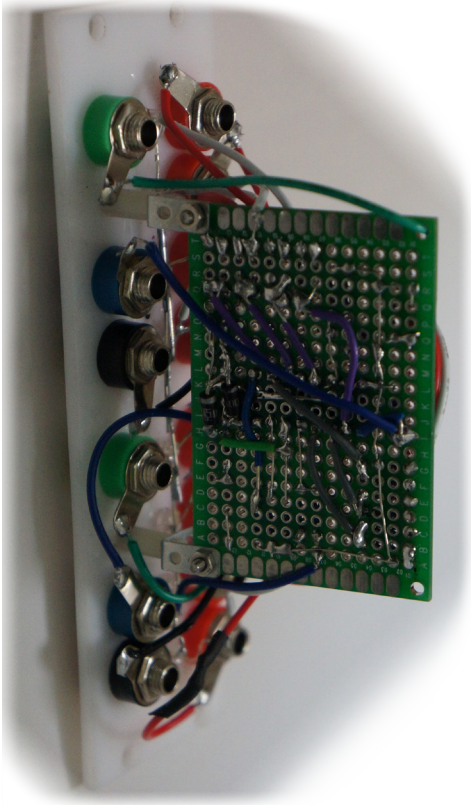
Comments, suggestions, questions and corrections are welcomed & encouraged:
contact@castlerocktronics.com

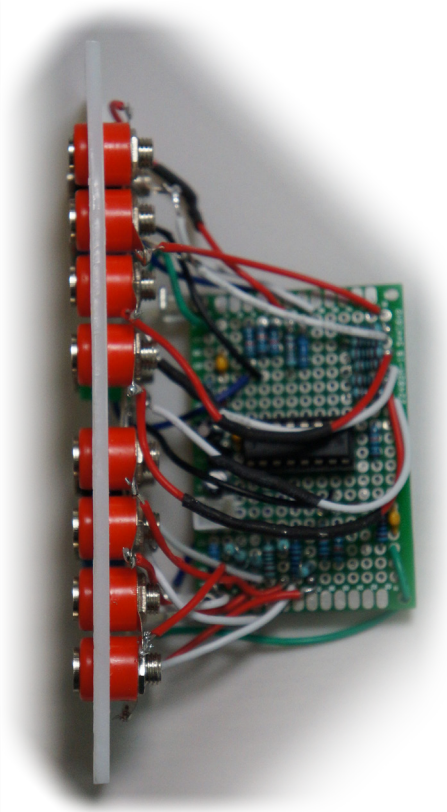
Table of Contents

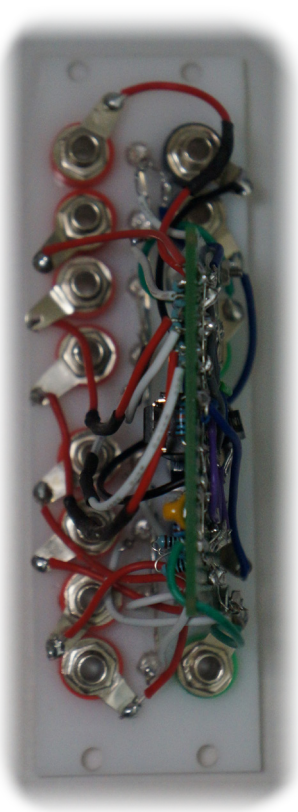
0. Build documents

0.1 Photos

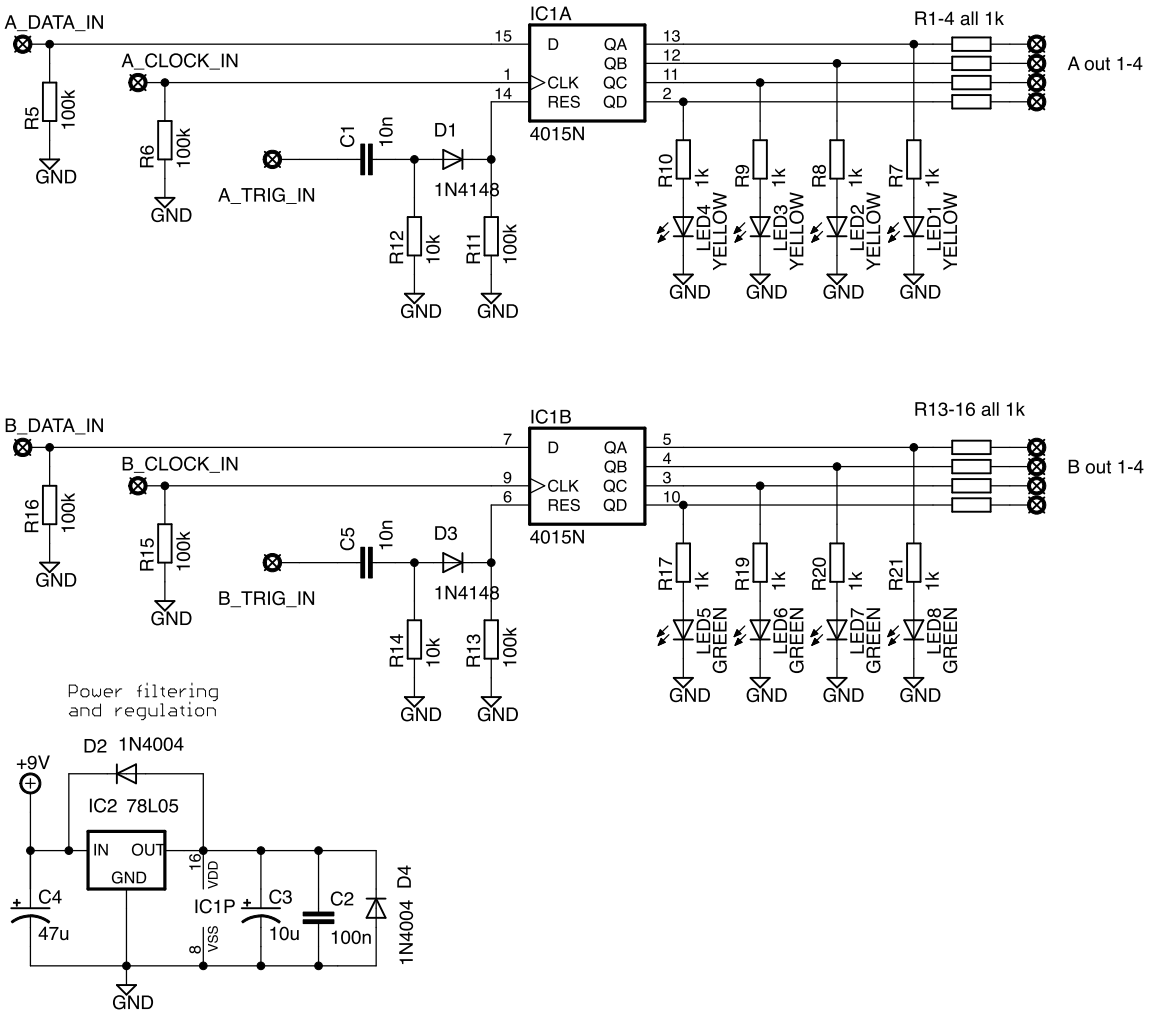








0.2 Schematic



0.3 Bill of Materials

ICs

78L05	*x1*
4015	*x1*

Resistors

1kΩ	*x16*
10kΩ	*x2*
100kΩ	*x4*

Capacitors

100nf	*x3*
10µF	*x1*
47µF	*x1*

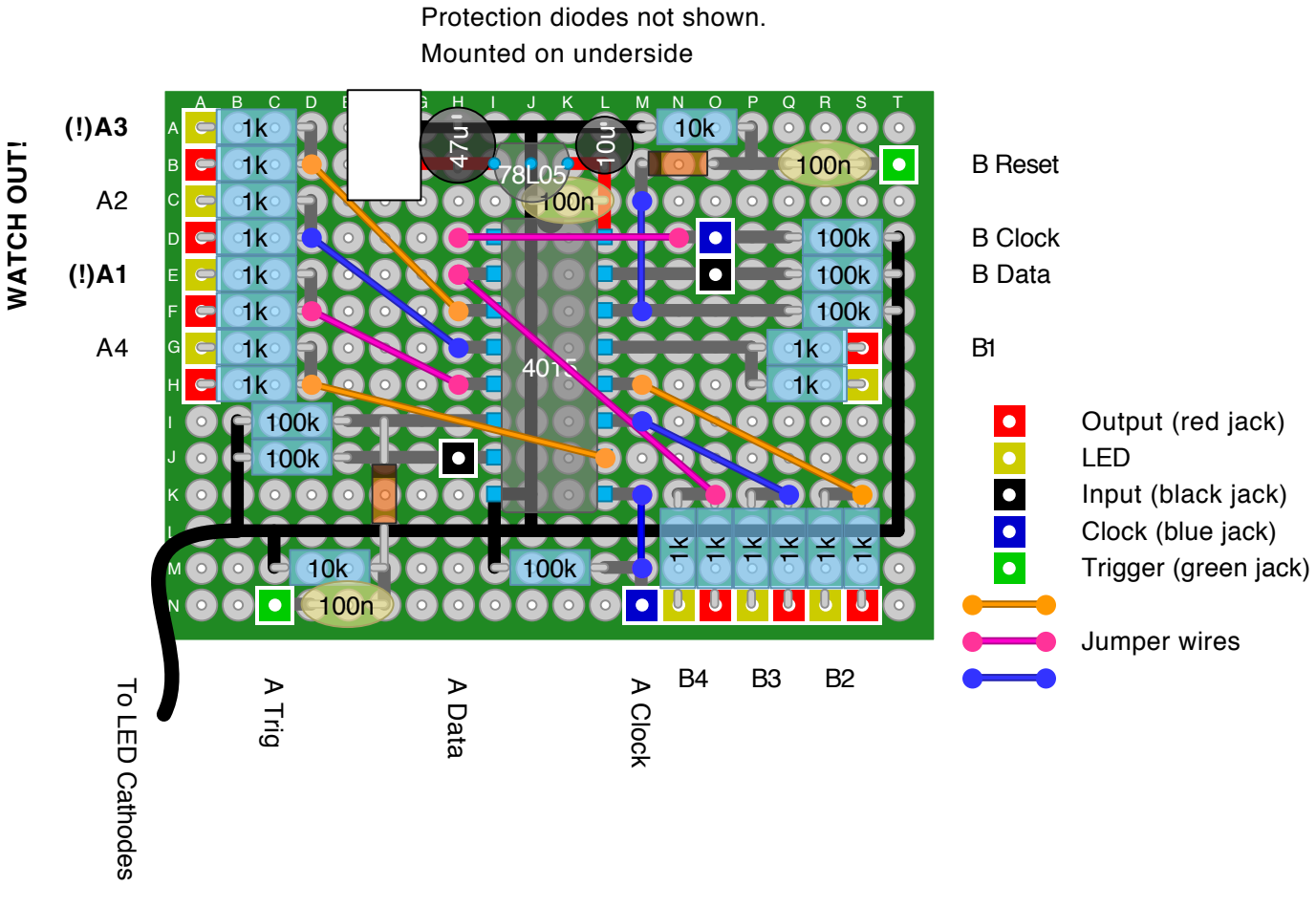
Diodes

1N4004	*x2*
1N4148	*x2*
LEDs	*x8*

Connectors

Banana Jacks	
Red	*x8*
Green	*x2*
Blue	*x2*
Black	*x2*
XH–2.54	*x1*

0.4 Layout



0.5 Construction notes

The board is attached to the front panel by M2 spacers, bolts and L-brackets.

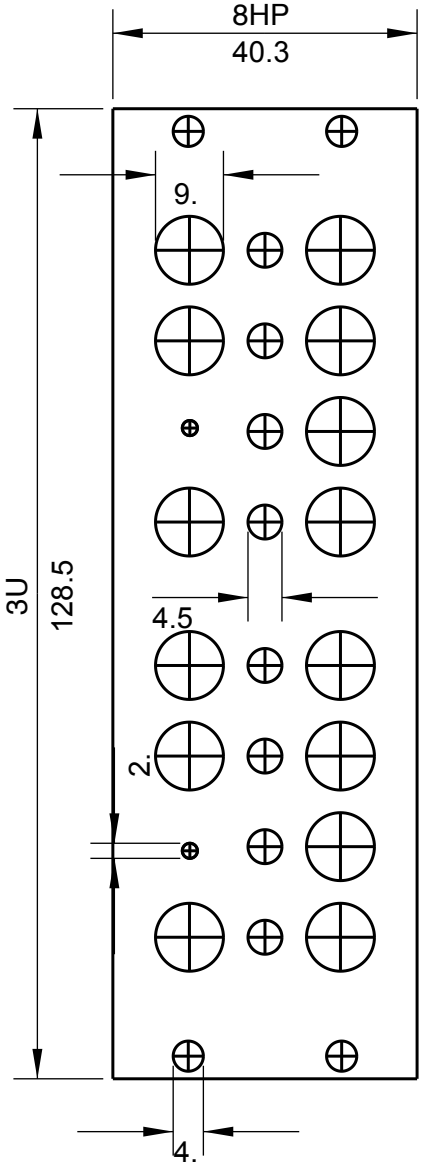
Be careful of the order of the outputs for shift register A!

Tie all of the LED annodes together behind the panel. It's easy to just bend each one 90° in till it touches the next one and solder them all together in a line. You might be able to see this in the photos, but it is obscured by the other parts a little.

Be careful that the banana jack lugs don't short to one another, the leds, or another module! Nothing will blow up but it will leave you scratching your head as to why things aren't working as expected.

The tigher arrangement of the power regulation and filtering doesn't space for the protection diodes on the top. Mount them on the bottom if you want them. Just don't trim some of the leads sticking out the bottom and solder them there.

0.6 Drill Template



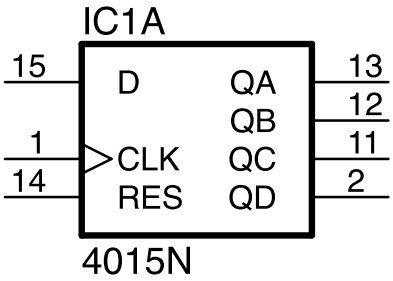
All measurements in millimetres

1:1 scale

Print out to use

1. Explanation

1.1 Shift register? I hardly know her!



(fig 1.1a)

The humble shift register *(fig 1.1a)* is an integrated circuit which might seem to have little relevance for synths at first glance, but is actually very versatile when you get used to it. To describe it in the simplest way possible, when shift-register's clock input goes from low to high it looks at its data input to see if it is low or high. Then, it changes output one to the same as the data input while also changing output two to whatever output one used to be and changing output three to what output two was and so on and so on, "passing" bits down its outputs one step at a time with every clock pulse. Ultimately, you can think of it as a sort of 1-bit delay line.

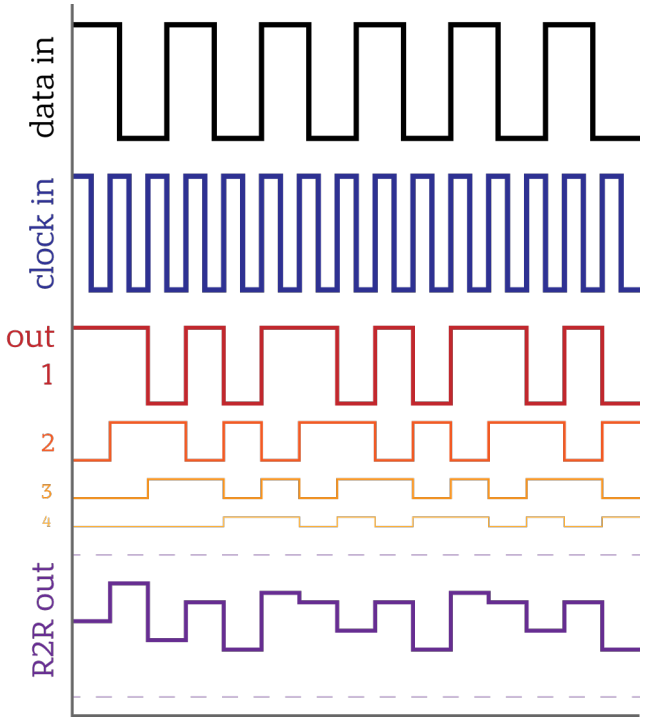
Shift-registers come in a few different flavours. The 4015 chip that our module is based on has two "4-bit serial-in, parallel-out" shift registers, which means that they have 4 stages with only one input but separate outputs for all four stages. Since we have two of them in one module, we can also chain them to make one 8-bit shift register if we want! There is also a master reset for each shift register, which forces all the inputs to 0, regardless of the clock or input or anything else.

This might seem not particularly useful for music. Who would want a delay that can only output 1s and 0s? However, as we'll show you, it's more useful than you might think.

1.2 Using the shift register

1.2.1 Modulating square-waves.

Feeding two different frequency square-waves into the clock and data inputs can result in some very awesome and interesting sounds that sound a bit like a combination of ring-modulation and bit-crushing and aggressive hard-syncing. Take



(fig 1.2a)

a look at *(fig 1.2a)* to get an idea of exactly what is going on.

Try keeping the clock input the same while changing the pitch on the data input. Especially fun is starting with the data input at a lower frequency than the clock then sweeping it higher and higher till it passes the clock frequency. The output sounds like it actually goes down in pitch before rising back up again. If you want to know more about what causes this effect, then you might want to read about the Nyquist rate and aliasing, which is what is going on in this example.

1.2.2 Pseudo-random sequences

Feeding different frequency square-waves into the clock and data inputs once more, we get a semi-random stream of bits out of each output which we can then add together to create a stepped voltage output which is perfect for driving a VCO. *(fig 1.2a)* The next module *(005 – Dual R2R Ladder)* is exactly for this purpose, so I recommend that is the next one you build. The two of them go together like carrots and peas. I've drew the diagram to show what the output from the R2R ladder would look like. You should also notice that although the sequence appears to start looping, it will actually change slightly every few repetitions as the two inputs fall in and out of sync.

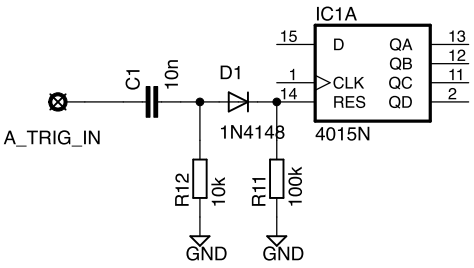
This method of generating a sequence is actually at the core of the Turing Machine module manufactured by Music Thing, though the Turing Machine uses more than 4 stages.

1.2.3 Frequency divisions

For this, you will need a digital inverter. All you need to do is feed the output from any stage through an inverter and then back into the input. Once you have done this, the outputs will be a division of the clock input. If you do it from the 3rd output, then the outputs of all stages will stay high for three clock pulses, then stay low for another three. If you did it from the 4th then the outputs would stay high for four, then low for four and so on. Since the outputs don't go low and high at the same time however, you can also use this to create weird, irregular tuplet rhythms where your sequences lag behind one another. Summing the outputs together equally (as opposed to with the R2R ladder example) can also be used to create stepped triangle wave outputs.

2. Analysis

2.1 The reset input



(fig 2.1a)

One of the things that has to be kept in mind with a lot of reset inputs for CMOS chips is that a many of them will also act as an inhibit. This means that so long as the reset input is held high, all other inputs will be ignored. This can be a useful feature, but one of the things I had decided on when setting out to make my modular was that I wanted to be able to ignore what form an output takes as much as possible. In this instance, it means that I didn't want to have to think about the duty cycle or anything of the signal I am using to drive resets. I just want it to do it on a rising edge. This means that any signal, so long as it is square enough, has to get the same response from the chip.

I achieved this by creating a simple "leading edge" detector out of a cap, a resistor and a diode. *(fig 2.1a)*

The capacitor and 10kΩ resistor form a high-pass filter/differentiator. The output between the cap and 10kΩ diode consists of a positive spike as the input goes from low to high and then a negative going spike as the input transitions from high to low. The negative spike could be potentially damaging to our CMOS chip however, as well as having no use. We solve this issue by adding a diode so that only positive going spikes can make it through. The cap and resistor value used are also far on the side of caution and much larger than needed just to be safe.

2.1.1 The 10kΩ resistor

The two resistors in this sub-circuit both perform very important functions even though they may not seem necessary at first glance. The 10kΩ resistor is needed not only to form the differentiator that does the edge detection, but without it the capacitor would not be able to discharge on

a negative edge. It'd have no path from which to draw current as the diode would be preventing it from doing so. This would result in the capacitor simply becoming more and more charged with every incoming pulse, possibly to the point of failure and being destroyed! This method of using diodes to stop capacitors from discharging is actually useful in other applications. Voltage doublers (and even triplers and quadruplers and higher!) use this trick to change an incoming AC voltage into a higher DC voltage.

2.1.2 Calculating the cap value

To know what cap value we need, we have to figure out how long we need the voltage spike created by a low to high to stay above a certain value. Thankfully, this is exactly what datasheets are for! Taking a sneaky peek at the TI datasheet for the CD4015B we are told that when the chip is being operated at 5V that the reset pulse width (t_WR) needs to be at least 200 nanoseconds. On the following page, it mentions that for an input to be considered high when the chip is powered by 5V that it needs to be at least 3.5V (V_{IH} Min). From here we can figure out the resistor and cap value since we now know we need to create a spike that spends at least 200 nanoseconds above 3.5V.

If you recall the article for module *001 – 4xSquare*, we talked about the RC time constant, also known as tau (τ). This simple equation, where we just multiply the value of our resistor by the value of our capacitor, holds the answer to our problem. There are a few problems however:

1) Our trigger input is not actually 5V because of the 1k protection resistors we have been using on our outputs. The cap will only charge to about $^{10}/_{11}{}^{ths}$ of 5V (4.54V) as the 1kΩ protection resistor and the 10k resistor in the differentiator form a voltage divider. It should also be noted that the diode and 100kΩ pull-down resistor make this even lower, but their effect is negligible compared to the 10kΩ resistor in the high-pass filter, only accounting for another 100mV or so.

2) R×C gives us the time to discharge to ~36.8% of the starting voltage. For 4.54V this is 1.67V, nearly 2V too low!

These issues are not difficult to deal with either, all we have to do is adjust our capacitor discharging calculation from article *001*.

$$V(t) = V_0\left(e^{-t/RC}\right)$$

Slightly different to what we did in article *001* however, we now know the time and voltage we want, but do not know the cap (though we already chose the resistor to be 1kΩ). Thus, all we need to do is patch in our values and solve for C to figure out what our smallest possible cap can be.

$$\begin{aligned}
(200nS) &= 4.54V\left(e^{-200nS/1000C}\right)\\
3.5V &= 4.54V\left(e^{-2\times10^{-7}/1000C}\right)\\
\frac{3.5}{4.54} &= e^{-2\times10^{-7}/1000C}\\
0.7709 &= e^{-2\times10^{-7}/1000C}\\
\ln(0.7709) &= \frac{-2\times10^{-7}}{1000C}\\
-0.2602 &= \frac{-2\times10^{-7}}{1000C}\\
260.2 &= \frac{2\times10^{-7}}{C}\\
C &= \frac{2\times10^{-7}}{260.2}\\
C &= 7.686\times10^{-12}\\
C &= 7.7pF
\end{aligned}$$

You'll notice that this is a pretty tiny value. Personally, I went for something much larger to err on the side of caution. 10nF also still allows reset speeds acceptable low audio range modulation up to a few hundred Hz.

3. Modifications

3.1 Reset and inhibit

Leaving out the differentiator and simply using a diode to the pull-down resistor at the reset pin allows for a reset and inhibit input that, when brought high, will bring all outputs low and then cause the register to ignore the clock and data input until the voltage at the reset pin goes low again. Use a white jack if you are sticking to the colour scheme!

3.2 Faster reset

Simply replace the cap on the trigger input with one of a smaller value. You should theoretically be able to get up with anything down to 10pF, though you may be better of experimenting a little. Anything lower than even 1nF may have negligible use unless you are driving all the shift register inputs with very high frequencies.

4. Lessons learned

4.1 Cutting acrylic

Turns out that narrow boards are tough to cut correctly by knife as they have a habit of not snapping neatly along the groove you cut. Will be using the circular saw attachment for my hobby drill next time, even though it scares me a little, but I am a big wuss.

4.2 Drilling acrylic properly

For once, this is a happy story where something I tried worked really well. First I bumped the speed back up on my drill which proved to work well to stop the acrylic vibrating. Then I used my stepped bit, which has the nice big flutes experts recommend for drilling plastics, and used it to widen my pilot holes to 4mm where all the LEDs went. Then I used my 4.5mm bit to widen that hole just a tad more and ended up having no problems with cracking! Finally!

4.3 Don't wire with the panel upside down

The first time I wired the panel up, I had done it with the board rotated round by 180°, so I ended up wiring the LEDs and output jacks upside down. The registered ran from bottom to top instead of top to bottom. I had to rewire every LED and output, but that wasn't the end of my woes...

4.4 Check the pin outs

You might notice that I have bolded output A1 and A3 on the layout, that's because I forgot they were swapped round and ended finding that after I had already rewired my panel, it was still wrong, because it went 3214 instead of 1234. Much heavy sighing ensued.

4.4 1kΩ, 10kΩ and 100kΩ resistors look very similar.

So then after I had rewired the front panel, I noticed one of the LEDs was strangely dim. Looking at the board, I realised I had accidentally put 3 10kΩ resistors in place of1ks on the outputs and a 100kΩ in another place. Somehow,

only one of the 10kΩs was wired to an LED and I would probably not have noticed if it wasn't for that. I would have just ended up being confused about why some of the outputs were behaving differently.

4.5 Modules can short one another out behind the panel

With all the gremlins in my module tracked down, gremlins appeared next door in my LFO. If you remember that I made the layout for the LFO a little tight and the board was right up at the edge of the panel, it turned out that one of the banana jacks was shorting a pot on the LFO causing one of the LFO outputs to be stuck as an inversion of the shift register output it was shorted too. That took a little while to figure out, I re-soldered the pot on the LFO twice before I realised what was going on.