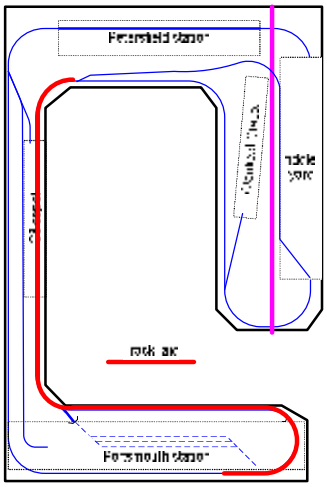
4: Track Wiring, Part 1



The track power is segmented into sections for block occupancy detection. I've used a Dremel tool with a cutting disc to make the necessary breaks in the "red" rail. Where these are on (or close to) curves, I've soldered the track to copper nails to make sure the cut sections continue to be aligned. So far I've done this in hidden sections (which I thought would be a good place to learn). It looks awful, and a more cosmetic approach could be needed.

To get the exact locations for track breaks, we needed to determine how quickly a train is stopped after a block becomes occupied. (Assume Railroad&Co's TrainController will stop a train instantly when it enters a stop zone). The BDL168 sensors seem to have a delay of approx 1 second before registering train presence: this makes the distance significantly speed dependent. I've settled on stop zones 140mm long, starting 40mm before the "expected stopping point".

Dropper wires are soldered to the track before laying where possible. Power connections to each section's dropper wires are via crimp connectors. This allows rapid connection – one operation of the ratchet tool adds a connector in a very permanent way – and the sections can still be isolated for maintenance. "Bullet" type connectors are used onto the track feeders; power feeds from the block detector boards will be connected via the "Scotchlock" type connectors.

To make the Scotchlock connectors work reliably, the wire needs to be relatively large gauge. I've ended up with 24/0.2mm ($0.75mm^2$) wire. This is good for current capacity (rating ~6A) but less good for wire bundle size!

SEEP point motors are being used. I asked around about different choices, with lots of different opinions returned. The SEEP motors were considered OK by "NTRACK" users. I've used a "standard" mounting, with the point motor screwed directly to the bottom of the baseboard. My initial impression is that there is too little adjustment available – the mounting holes have to be EXACTLY right. I'll probably put the next ones a further 12mm down, using a wooden spacer: this should extend the movement of the motor itself, and give some more mounting flexibility.

The SEEP motors have been connectorised, by addition of 2.8x0.8mm tags to which a blade type crimp connector attaches. This means no soldering upside down under the railway. Initially I soldered these terminals to the PCB traces, but found them prone to snapping off. To add these connectors reliably, drill a 1mm hole in each of the 6 PCB traces on the connector (a handheld Dremel tool is good for this). The connector pin pushes into the hole, and is then soldered. There is a significant degree of force required to insert the pin; I'm hoping that will lead to a similar amount of force to remove it (and therefore the soldered join being less stressed).

I have noticed that the SEEP motors, when they slide across, short momentarily. I haven't yet determined if that causes momentary power loss to the track zone in which they are located. If they do, then I have a problem!

Electronics

There is a fair amount of DCC and LocoNet electronics associated with the railway. For the track wiring, PM42 units split track power into a total of 16 power zones – each of which is short circuit protected individually. BDL168 units provide block detection: a total of 8 BDL168 units will be needed providing a total of 110 separately monitored occupancy sections.

The PM42 and BDL168 boards connect to the layout using a 44 way edge connector, to which the user is expected to attach wires. The exact method is left as an exercise for the modeller; my observation is that the holes in the tags are too small. For the PM42 boards, I've added fat wires then brought them out to a "chocolate block" arrangement. For the BDL168, I wanted the outputs to be connectorised (for fault location etc) and I've soldered 3.5mm pitch screw terminals onto the edge connector. NEVER AGAIN: I've got 2 boards of each type wired like this and it's a mess. I'm considering making a small PCB with decent screw terminals to improve matters. I may yet go back and retro-fit the bits I've already done, because I'm worried about long term reliability.

A PC is now installed for the railway. To configure and test the electronics, I'm now running Stefan Traschler's **LocoNetChecker** program. This reads the settings from most of the electronics that it finds, and can program some of the units "live" via LocoNet. PM42 and BDL168 settings can be read and written directly. There is also an interesting feature (that I haven't got to work yet) to make locomotives and block detectors have "human readable" names. So for example detector 15,7 can announce itself as "Petersfield station platform 3" occupied.

We have CAD design of complete layout & wiring, and computer monitoring of layout operation. If only someone would write a module to control a robot to lay the track and do the wiring!

Rolling Stock Update

Locomotive	**Decoder**
Kato Eurostar	DN163K0B
Farish class 94xx 0-6-0 PT	DZ123
Farish class 08 shunter	Not yet!
Farish Class 47	Not yet!
Farish class 66	DZ123
Farish class 159 DMU	Not yet!

The class 66 decoder installation was quick and painless: there are clearly marked solder pads for the decoder wires and plenty of room for the decoder itself.

Expect to see updates in this area. After initial track laying & wiring, the next activity will be testing reliability of train movements: so I need to catch up with decoder installs etc. So far the smoothest running loco (and therefore least useful for testing) is the Eurostar. The class 66 runs smoothly on plain track, but it has fine detail on its bogies that snag bits of the pointwork. I need to determine compatibility issues like this before going much further with track laying. I'm expecting Michael and Christopher will be involved in train driving for this.

I've adopted JMRI's **DecoderPro** for managing decoder settings. This was very quick to get started: I was able to put existing locos on the programming track and read back their settings in moments. It has configuration screens for a huge number of different decoder types, and saves them all into its local database. No more calculating and programming CVs by hand.