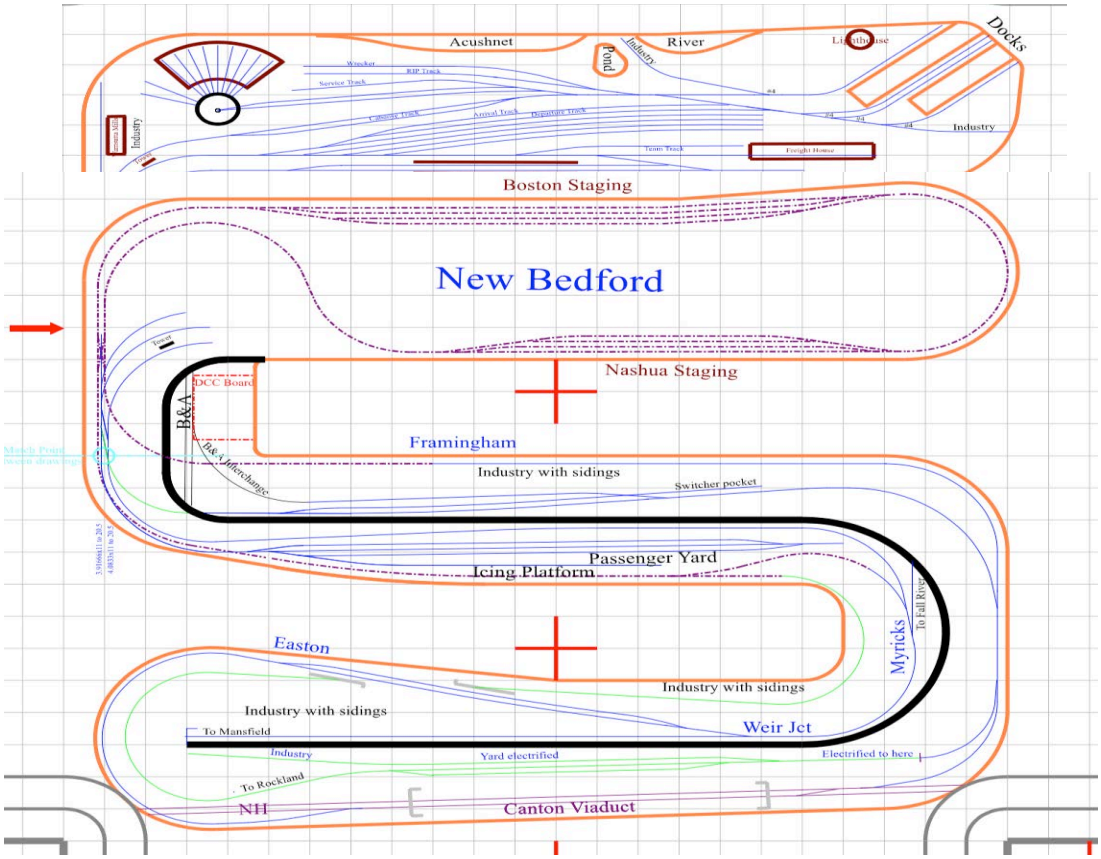
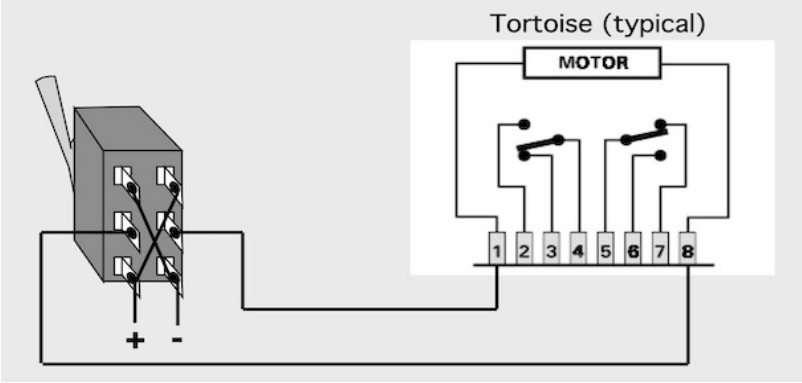
LCC and Tortoises



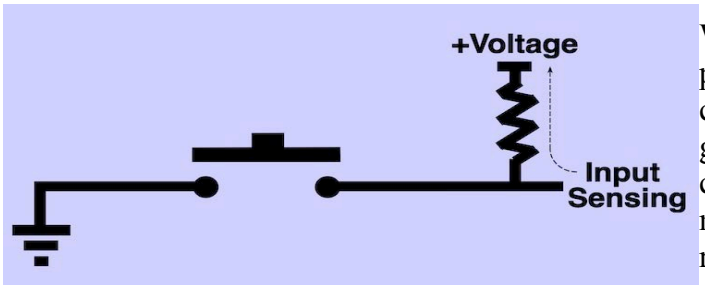
Jerry asked me to help with the electricity for his layout. He had finished the room and had the first 20+' long benchwork built. The next step was installing the first track, which will become, ultimately, "underground" staging. There were 4 parallel tracks on each side leading to single track connections at each end. That requires 3 turnouts at each end – 12 in all -- to have control over the 4 tracks on each side. Already 12 turnouts to control and we weren't even at ground level – there would be a lot more. Further Jerry wanted signalling (well, semaphores specifically), and the option for dispatcher control. And it would be big – this would be the first of three long peninsulas or some other evolving layout....

Before I discuss controlling them, I need to talk a bit about Tortoise by Circuitron switch machines – A wonderful invention, now challenged by several imitators; including, now, servo motors. I am not going to discuss these options – as, luckily for me, Jerry had already bought a lot of Tortoises [you know how we model railroaders accumulate what we need over years and hopefully have it when we actually need it....] - so that was what we were going to use.

Controlling them is not a simple pushbutton operation as the old twin coil machines were – push one button for one way and another button for the other. You could even create diode matrixes to control multiple turnouts – just what I wanted to do here – easily – and then – one button triggers the whole route. But twin coils have their own problems, not the least of which is the current required to operate them, noise, beating up the points, etc. And besides, Jerry already had Tortoises



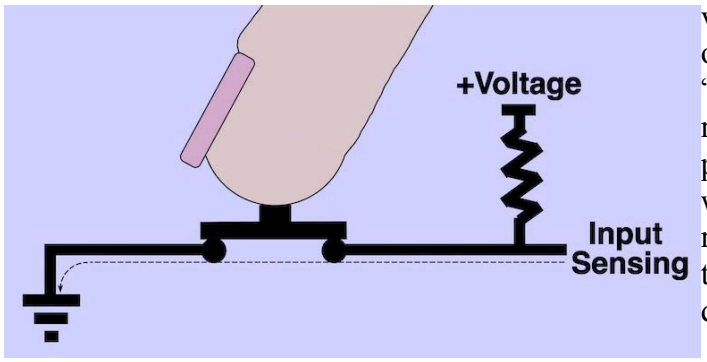
Tortoises are controlled by switching DC Voltage polarity on the two outside connections of the edge connector – Plus on one side pushes it one way, reversing the + & - and it goes back the other way. It is intended that the current stay connected when thrown to keep pressure on the points. So that means we can't just have a push button control it – at least not directly. To manually control it, we would need a toggle switch wired to reverse the polarity and keep the contacts together – or a similarly wired relay – which would then allow pushbutton or contact closure control.



Why say contact closure, rather than just pushbutton, since where we most often have a contact closure is when we push a button. We're going to see that there are more ways to make a contact closure than just a button – and a lot more things we can do with it. But it is good to remember that push button....

Generally a pushbutton connects to ground when pressed – or when the internal contacts are "closed". Connecting the wire to ground sends that ground "signal" back down the wire to the receiving device – whatever it might be.

How does the device "see" the ground? Inputs generally have a "pull-up" resistor connected to + voltage and monitored. If there's no connection on the input the input "see's" that + voltage, or a "high" with a small current through the pull up resistor, and waits. The moment someone pushes that button, the contact closes and the wire now connects to ground through much less resistance (the wire) than the pull up resistor – the input monitor "sees" the "low" and tells the device that that input has been "activated".

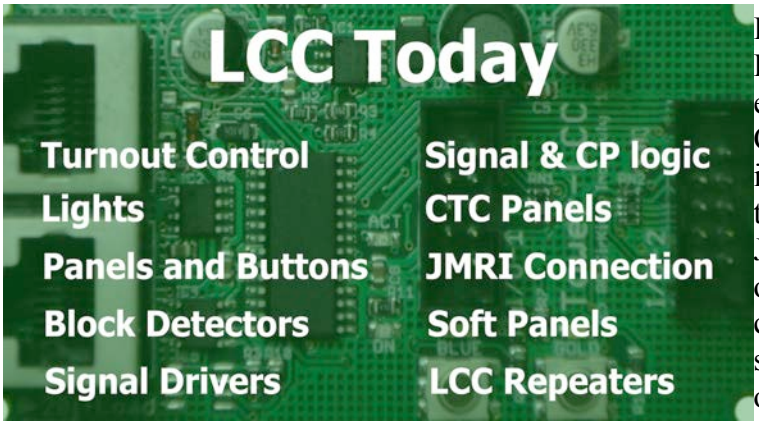


I did consider many options for controlling the Tortoises, especially here in the staging yard which would not necessarily be controlled by a dispatcher but would need some way of choosing the proper 4 tortoises to activate a track. Jerry wanted local pushbutton control panels on each side to control track choice – so whatever we used had to also allow local button input.

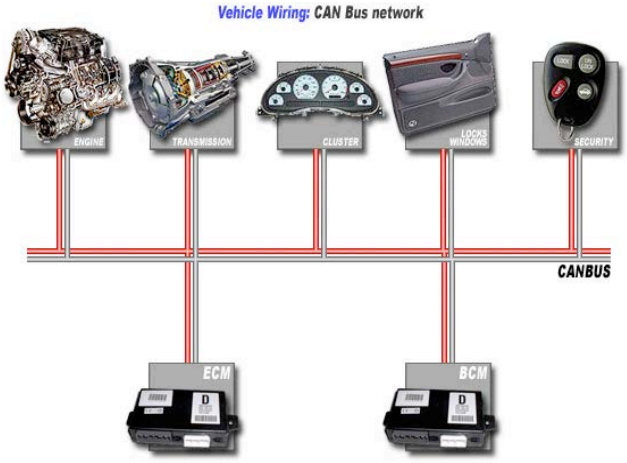
There are, of course, stationary DCC decoders which allow control from the DCC Cab. Some of them also include the option for adding local contact closures (buttons) for turnout and/or route control in addition to DCC control. However Jerry did not want to have to use the handheld to control turnouts – but he did want, at some point, to have dispatcher control of mainline turnouts.

I wired my layout a few years back – and used then state-of-the-art DCC stationary decoders, still

available. It was then a Team Digital SMD 82, of course now it's up to SMD 84. When I chose them, they were the least expensive stationary decoder per output. Each has 8 Tortoise or twin coil outputs – as well as those necessary local button inputs. We could have gone a similar route here, the SMD 84 and many other DCC stationary decoders are available....



I have been following the development of Layout Command Control for several years, especially at the NMRA National Conventions, so I had a pretty good idea what it could do – and the more I thought about it, the more it seemed like the best option for Jerry's layout. Since ultimately, some form of dispatcher control would be involved – connecting everything to a network of some sort is necessary. That would allow the use of JMRI or others for signalling and turnout control. But when a dispatcher isn't around, we'd still like to be able to control turnouts (those buttons again) and maybe have the signalling run itself – without having to attach and operate a laptop to make it go [and how many in the Module Group know how to set up the signalling system?]. And that's where LCC shines. Yes, it does require a computer to set up (contrary to the NMRA ad, sadly) - but once loaded, the components create and consume "events" completely on their own (we'll get more into that) – no attached PC running JMRI to make it go.



There is product available right now that is cost effective compared to DCC stationary decoders and offers the same and more control possibilities. LCC's network is based on the CAN bus that was invented by Bosch for car components to talk to each other. It is a very robust network – designed to function well in a moving vehicle with lots of interference. It is similar to Ethernet (and LCC can run on Ethernet and many other connection topologies as well); but is even more robust, including some very neat data hierarchy to protect from noise and erroneous data. And it's a network – a series of independent "nodes" connected with RJ45 - plain ole' - Ethernet cables, available almost everywhere these days.

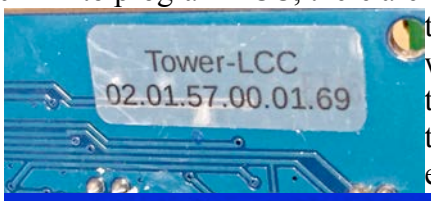
Yes, I will freely admit that RR Circuits is, right now, the only manufacturer making a full line of LCC product – and I'm not in any way in their employ – but their product does what we need it to and is available now. Later I will mention some of the products we have been teased with for future availability, not only from RR Circuits but some other major players as well. Dick Bronson of RR Circuits is not only a really nice guy and a major proponent of LCC, but is incredibly helpful to folks like me doing a first setup and trying to understand a new way of doing things – he really knows all about LCC and is trying to produce what we want. The first LCC products were variations on products he had already developed using LocoNet, so it wasn't like they were reinventing the whole thing. Some of the options, for example – the board that drives 8 Tortoise outputs, have been a product for quite a

while – as any contact closure will enable them. And he is adamant about how much better LCC is than LocoNet

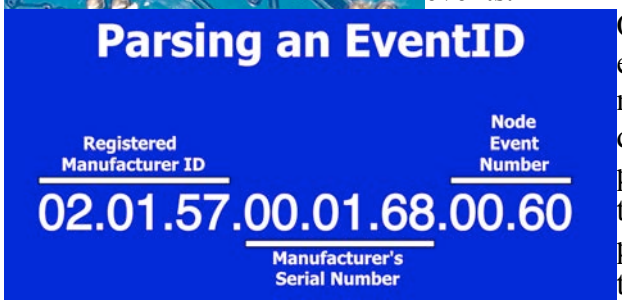
So why LCC – rather than just adding to the DCC cab bus. The LCC bus is a lot faster so there's "space" for more data to zip back and forth. LCC over the CAN Bus is capable of 125kbps (kilo bits per second) while DCC is but 8kbps. Once you get a lot of operators operating a lot of trains especially with sound commands and perhaps a dispatcher (or CATS) looking at an occupancy map and throwing turnouts, the (older) DCC cab bus starts to get noticeably less responsive. So if that's where you're going, you might want to separate out all the "unnecessary to train running" information and set up a

new bus – especially since the cab bus is now mostly wireless anyway and doesn't require as many fixed input panels as completely wired control does. And it's an easy network to setup – you can get pre-made CAT5e or CAT6 cables in almost any length. And if you want to make custom cables, good ratchet crimping tools, wire, and RJ45 connectors are readily available – from Home Depot and Lowes among many others.

One of the neat things about LCC is that it's a network of nodes, which share data and power. So what's a node anyway? A node pretty much boils down to - any circuit attached to the network electrically – that produces or consumes "events". What's an "event"? An "event" is – something that happens! Just like in the physical world. An event produced in one node is instantly shared with all the other nodes on the network. Events in LCC are actually 8 bytes of data. Fortunately when using JMRI to program LCC, there are "Copy" and "Paste" buttons on each window so you rarely have to



type the bytes. Each device on the network comes from the factory with a completely unique binary identity code and that code becomes the basis for events sourced from that node – that the node thoughtfully generates for us when we're setting up to "produce" events.



On an LCC network there will be nodes that create events – like from pushing a button. There will also be nodes that "consume" events – like when a Tortoise driver "sees" the button push event and changes polarity to reverse the Tortoise. The fun part is that these events could really be anything – from button pushes and straightforward occupancy detection, through signal logic and contact closures to control trackside signals, operate accessories, even room lights!

Since the nodes, for the most part, get their power from the network; you only have to wire a single CAT cable from one node to the next – up to 40 nodes on one "leg" - depending, of course, on the node's power requirements – like how many LED's it's powering.

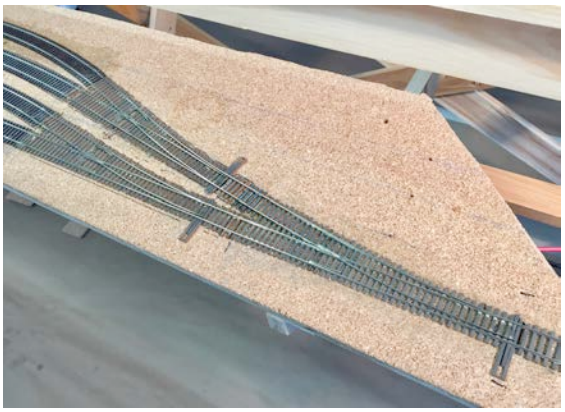
That power comes from one of the two necessary "non-nodes" that are connected to the network – a dual powering module with 1/2Amp output (on each jack). This power supply can power a network of up to 25 Tower LCC's on one leg – there are, of course, ways to link up networks – including over the internet, so really no limit on total nodes.

The second “non-node” is the USB interface to the LCC network. While the PC attached through this USB interface running JMRI (or other LCC programs) is a node, the interface itself is not (can't produce or consume events). One thing I will note, as it came up when I was programming Jerry's system, there must be a minimum of two nodes for LCC to work [LCC wouldn't run after I removed my PC until I added a second Tower LCC to the network]. My guess is that it needs something to talk to and it's programmed to not talk to itself .

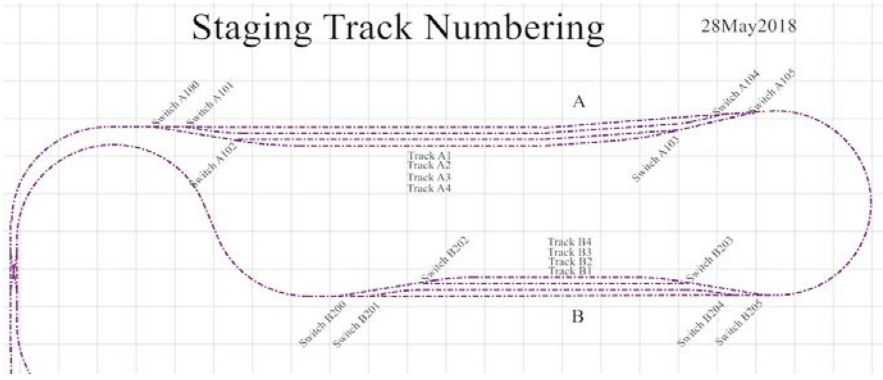
The final hardware **required** for the LCC network is a termination at each end of the network. These properly terminate the network and reduce noise in the data. The RR-Circuits ones have LEDs to show that power and data connections are active.

It's really not a big deal to add a node – unplug the termination, add a CAT cable to the new node position, plug in the new node and terminate the other jack – done.

At this point, in preparing to control Jerry's turnouts, the events we're dealing with will change a turnout from normal to thrown (and vice versa) - especially – in groups to make a track active at both ends. So here we'll have pushbuttons to create events and switch machine drivers to consume them.



Let's look at the intended layout and how each track is made active. At each end of the 4 track yard are three turnouts, the first chooses the front or rear two tracks and the next two switch each track. So to access any specific track requires controlling two turnouts – at each end – so 4 turnouts for each track. Fortunately we want each end to basically do the same thing – so we could use one Tortoise output to control a Tortoise at each end at the same time. I checked with RR Circuits to make sure I could attach two Tortoises to one output. Dick Bronson assured me that their Stall Motor Driver has adequate output (25mA@) to drive two Tortoises (~10mA@). Now that I knew how they were going to be wired, I needed to determine what events needed to be created to operate them from button pushes.



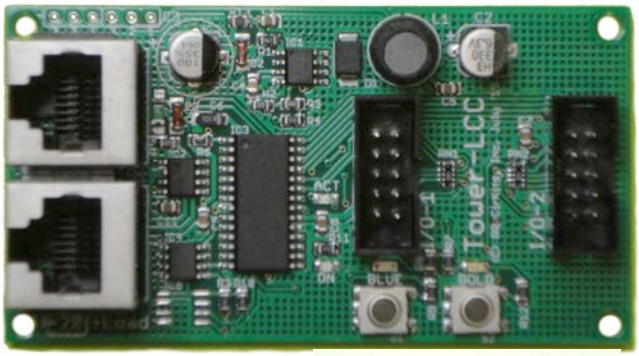
First, some definitions, so what I create will be consistent and easy to understand later when repair or alteration is required. So, looking at the upper entrance to staging, I named the turnouts and made this diagram: Switch A100, etc. to be able to define each turnout state. A turnout is “Normal” when the

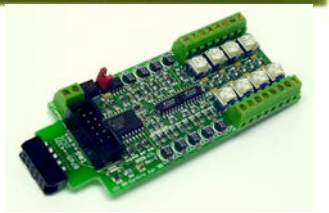
straightest route is chosen and “Thrown” when the diverging route is chosen. And, yes, which is actually “normal” can be (finally) determined when physically attaching the two power wires to the Tortoise [and if “Thrown” is actually “Normal” – reversing those wires “fixes” it].

I then created a spreadsheet with the states that I needed – For example: to access the outside track (#1) both turnouts A100 and A101 need to be “Normal” – as well as their parallel turnouts A105 and A104 at the other end. To access track #2, Turnout A101 (and A104) must be “Thrown”, etc. I can then create two “events” for each turnout output to “consume” – one for “Normal” and one for “Thrown”. Then add the appropriate events to each button's list of events produced when active (pressed). Once I created this spreadsheet, I knew how to program the node.

Producers									Events	Consumers	
Staging Pushbuttons										Tortoises	
Tower LCC ...68 [I/O #1]							Tower LCC ...69 [I/O #1]			Tower LCC ...68 [I/O #2]	
A1	A2	A3	A4	B1	B2	B3	B4	Thru	X	Event Number	Device	Action
X	X									02 04 57 00 01 68 00 60	Switch A100/105	Normal
		X	X							02 04 57 00 01 68 00 61	Switch A100/105	Thrown
X										02 04 57 00 01 68 00 6C	Switch A101/104	Normal
	X									02 04 57 00 01 68 00 6D	Switch A101/104	Thrown
			X							02 04 57 00 01 68 00 78	Switch A102/103	Normal
		X								02 04 57 00 01 68 00 79	Switch A102/103	Thrown
				X	X					02 04 57 00 01 68 00 84	Switch B200/205	Normal
						X	X			02 04 57 00 01 68 00 85	Switch B200/205	Thrown
				X						02 04 57 00 01 68 00 90	Switch B201/204	Normal
					X					02 04 57 00 01 68 00 91	Switch B201/204	Thrown
							X			02 04 57 00 01 68 00 9C	Switch B202/203	Normal
						X				02 04 57 00 01 68 00 9D	Switch B202/203	Thrown
								X		02 04 57 00 01 68 00 A8	Crossover A	Normal
									X	02 04 57 00 01 68 00 A9	Crossover A	Thrown
								X		02 04 57 00 01 68 00 B4	Crossover B	Normal
									X	02 04 57 00 01 68 00 B5	Crossover B	Thrown

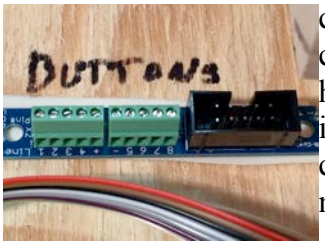
The node that's available now and makes this all happen is the RR-Circuits **Tower LCC**. This node's 16 “lines” can be set up as either input or output lines, on a line by line basis. There are two I/O jacks available on the Tower LCC – each a 10-pin jack allowing for 8 I/O connections and power and ground. So a single Tower LCC can process a total of 16 “Lines” made up of a combination of contact closure inputs and contact closure outputs



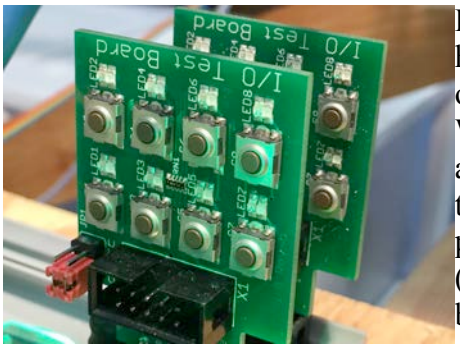


There are several accessory boards that can connect to the TowerLCC's I/O jacks: the servo switch machine controller we're using as well as: a solenoid switch machine driver, a relay board, several different signal driver boards (4-aspect and searchlight versions) and occupancy detectors capable of remote sensing. Each of these uses one of the I/O connectors and, therefore, 8 inputs or outputs. These I/O connectors can also be connected to ribbon cable through inexpensive, crimp-on connectors for “real world” connections like buttons or lamps. If you use rainbow ribbon cable you can easily tell which wire is which – and get the right one on the right button.

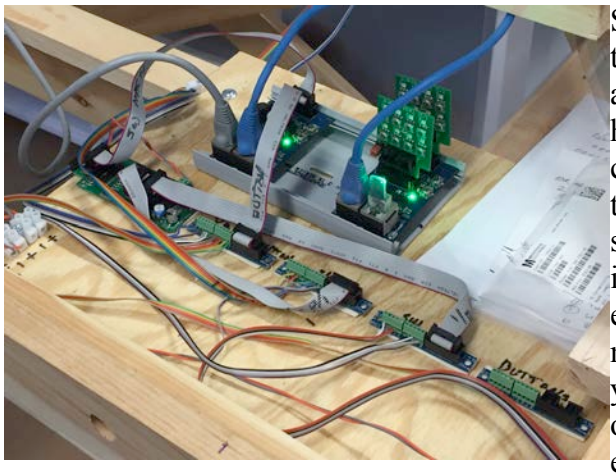
On Jerry's layout, a button will be used to choose each of the 8 tracks available, so 8 button inputs are needed on the TowerLCC. We will be using the first 10-pin I/O jack to connect buttons and the second one will be used to attach the stall motor driver board. The buttons are connected to ground from the ground



connection of the 10-pin jack on the Tower LCC. The other pushbutton connection is to an input on the Tower LCC. Again, these inputs are "held high" internally – looking for that push button connecting to ground to indicate "ON". Note the use of RR Circuits wonderful "Breakout Board" converting 10 pin header to screw terminals – best $6 investment you can make.

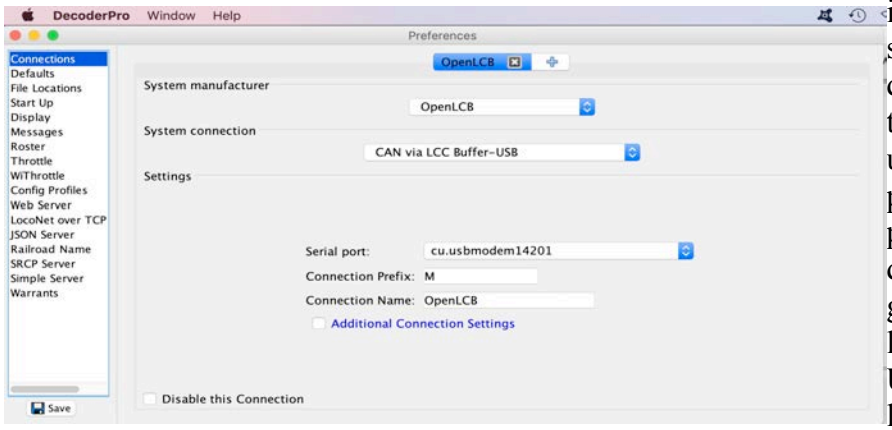


RR-Circuits also makes a wonderful **I/O Test Board** – a most helpful device. It has buttons and LEDs for each line, showing output and allowing input on each line – a most useful $8 device. With it you can test inputs by pressing a button (that contact closure again) and view outputs with LED's as well as an LED that flashes to show that an event has occurred. It allows you to test the programming – create an input (push button) and see an output (LED Lights) to see that your events are occurring properly – before you attach the Tortoises!

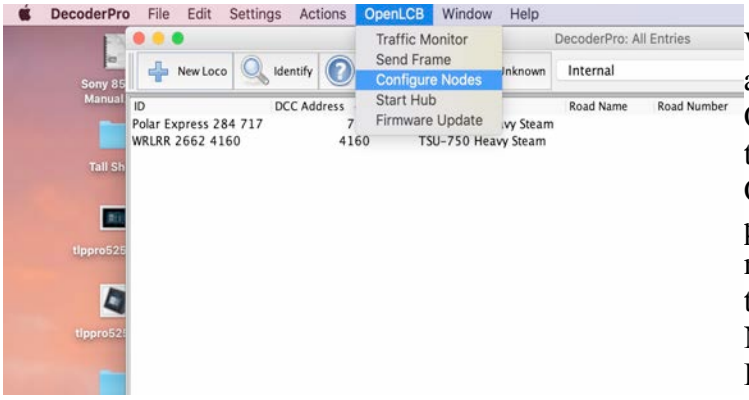


So, at this point we have a powered network with terminations at each end, a Tower LCC board (or two), and a PC connected through the USB interface. The hardware is in place, so now we can program it to create and consume events! I'm going to briefly show the commands that I created to automate a button and a switch driver – but this isn't how-to – but more a – this is what's involved. I will freely admit that this got a lot easier as I got into it – the first couple were difficult – mostly due to reading and dealing with all the variables you can control on each TowerLCC line – there is a lot of power in the software that runs this. One of the engineers writing the JMRI LCC code is Balazs Racz, a Google employee. He has incorporated Google-like search and dialog in filling out the tables – and "copy" and "Paste" are my friends. A little side note about the interoperability of the nodes, I have seen Balazs pull out a node that wasn't communicating properly and quickly program, in fact, a RR Circuits, replacement – literally in moments, during a clinic at a NMRA National.

So now for the rest of us, the first step in programming LCC nodes is running DecoderPro, part of the Java Model Railroad Interface (JMRI). There are help groups on line to assist so I'm not going to get involved here with getting it started. Yes, there are sometimes difficulties starting up the first time – mostly Windows or Mac unpleasantness, finding the right port, etc. – but with group and personal help, it is actually not difficult to get going and most get going with no problem at all. At least the LCC buffer is a direct USB connection and should at least show up [doesn't require the

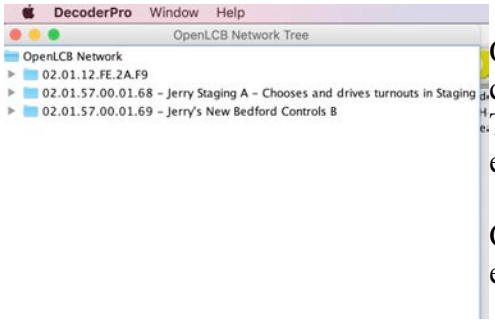


RS232 to USB converter that NCE PowerHouse does, for example]



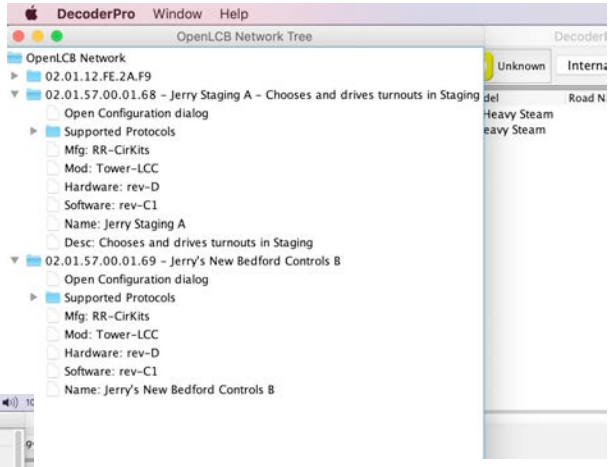
What's different here is that we will be accessing code that was created by the OpenLCB group (OpenLCB is a forerunner to NMRA's LCC). OpenLCB stands for Open Local Control Bus – LCC uses just part of the OpenLCB code, it is capable of much more. Click on the OpenLCB Tab on the DecoderPro page and choose "Configure Nodes". If you don't see an OpenLCB tab, DecoderPro will send you to Preferences to set this up when you start again.

We will be using the OpenLCB Connection utilizing "CAN via LCC Buffer-USB".

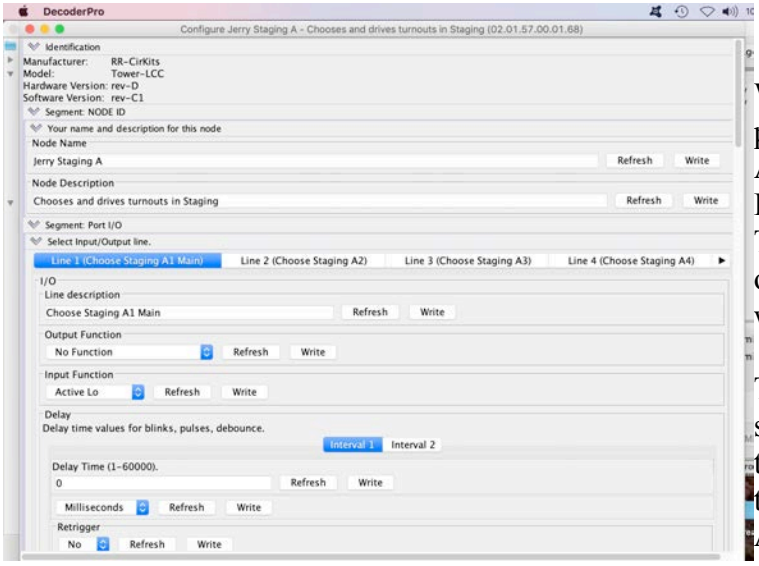


Once we are able to "OpenLCB Network" - by clicking – we can see the nodes connected – the slide shows the two TowerLCC's presently on Jerry's network. Open the dialog for each node.

Clicking on "Configuration Dialog" expands information about each node.

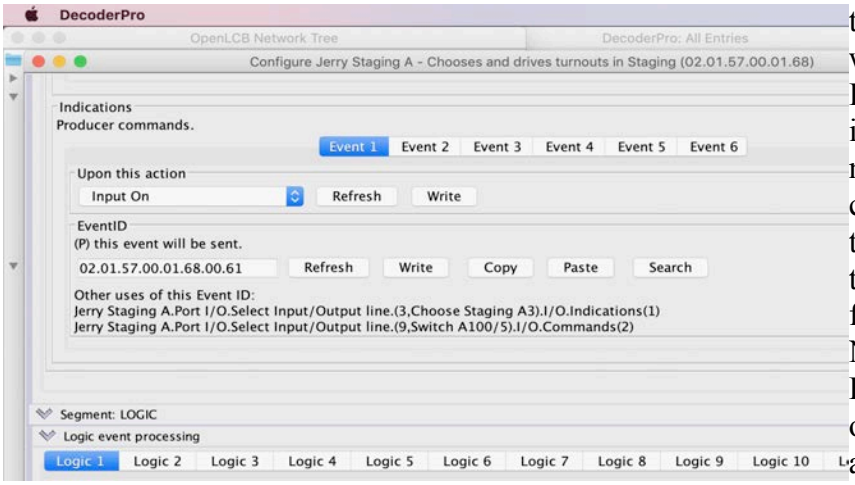


Clicking on the top Tower LCC (which is the one we're using to control this yard) –

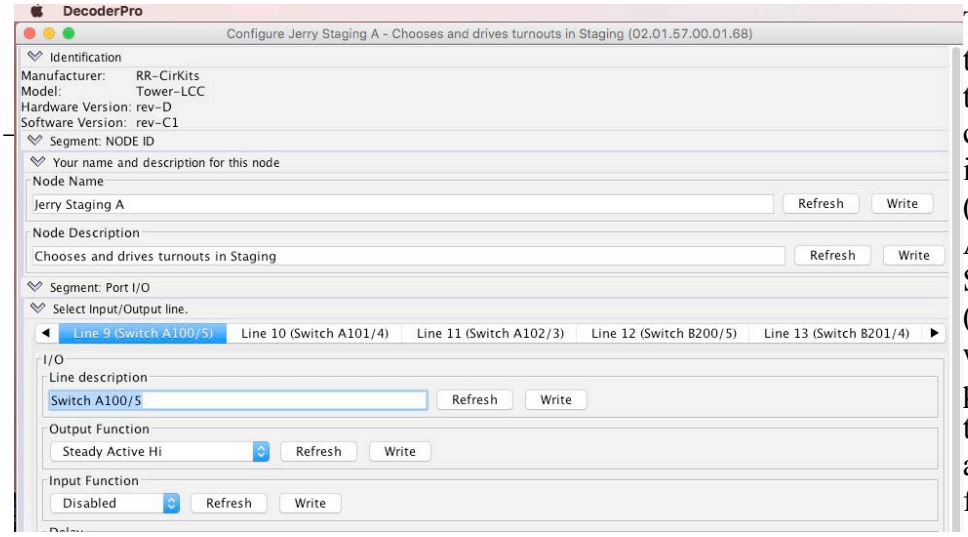


We see that Line 1 which represents pushbutton 1 is set up to "Choose Staging A1 Main" when Input Function is Active Low and that there is no output function. This title is something I added when I first opened the node – so I could know what it was supposed to do.

There's a lot of capability shown that I'm skipping over, as I don't need it to throw turnouts – but maybe later I'll find wonderful things that make stuff work better... Anyway, further down the page we can see



that Producer Commands Event 1 will send out <event #> when that Input goes "On". The Event Number is generated automatically by the node and the first 6 pairs of hex characters are specific to the node – they're even posted on the back of the circuit card. Unique to this card from this particular manufacturer. Note also that multiple (up to 6) Events can be triggered by an action on a Line. Of practical importance are the "Copy" and "Paste" buttons – allowing you to rarely enter the long event ID



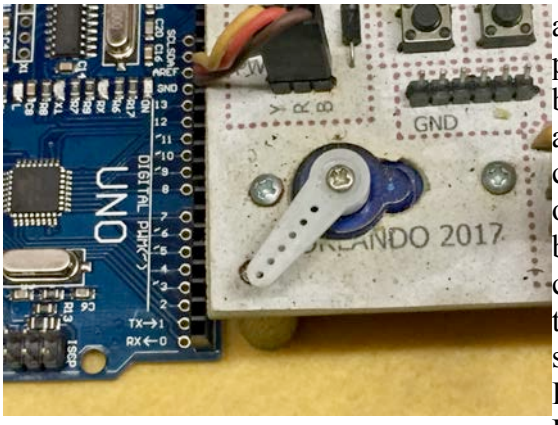
The other I/O connector of the Tower LCC is connected to a Switch Machine Driver clearly all outputs, so I filled in Line 9 as Switch A100/5 (as it controls Turnouts A100 and A105) with a Steady Active Hi (Remember that Tortoises want to be continuously powered to keep pressure on the points). This Line is not an input so that input function is disabled.

and you can see from the "Consumer Commands" window that you can have up to six events consumed with specific outcomes on the Output line – in this case Normalizing or Throwing the Tortoise controlled turnout. JMRI thankfully shows where else you've used this Event so you can be reminded where else it's going.

It is actually best to configure the Consumer lines first, and let the Tower LCC pick the <event #> for each state – normal and thrown. Then you know the event number to fill in when you wish to produce an event to throw or normalize the turnout.

Now that the commands are entered into the node, you can disconnect the laptop and just let the system run on it's own – generating and consuming events. Of course right now the generating is limited to 8 pushbuttons and the consuming is limited to 12 Tortoises – but it only gets bigger from here.

I did mention the future – and here that involves semaphore signals. Right now, the only factory method of controlling semaphores involves Tortoises and, of course, Jerry has a few of them on the shelf. Now controlling them for semaphores would be just an extension of turnout control – but wouldn't this be a perfect place to use servos? With servos you can control both speed and final angle – both really useful in replicating a semaphore's action. So I'm working on an interface to control semaphores with servos – which is mostly working out a mount for the servo so it can move the semaphore blade from under the sub-roadbed consistently. Right now control of the servo would involve an Arduino (or the like) triggered, most likely, from a TowerLCC. I'm hoping for a servo LCC driver interface before I get to the semaphores so I can avoid the Arduino step, but at least options do exist. Picture is my Uno test fixture from NMRA National in Orlando – where I learned how to program servos with Arduino.



Speaking of hoping for new function, there have been some exciting new products hinted – some have been announced and some just rumors. I suspect the most exciting new LCC product is the Train Control Systems (TCS) WiFi Command Station. This has been shown at the last two NMRA Nationals as well as Springfield in January, so it's really close to existing. This is a completely new 5 Amp Command Station that's WiFi based – around their new WiFi handheld throttle. Not only wireless WiFi – but there is a wired connection that, right now, offers connection to the NCE cab bus. Yes, you can plug in your existing NCE cabs and run your layout through TCS's new box. They have plans to port this to other manufacturers as well, but NCE is already working. So there's a really nice new WiFi throttle with a command station that also allows iPhone and Android WiFi connection, and an interface to existing NCE hardware – it's a pretty capable box/system. But there's more -- this modest white box is also an LCC node! You can re-name LCC consumers from their WiFi Throttle and directly control turnouts, etc. So you can get the switch and occupancy detection off the DCC bus yet still have direct control from your handheld. The future is fun!

Another product that RR Circuits was intending was a small 4 output stall machine driver with direct LCC interface. Being able to have switch machine control right at the Tortoises with only a CAT cable connection (and power for the Tortoises) seemed like a really good idea. Unfortunately, the prototype turned out to be too small and used non-standard connectors (not RJ45's) and thus won't be seen in the market – but a slightly larger version may... Those user groups can be vicious to new product that doesn't "make" it – in this case the non-RJ45 connectors doomed it

The good news is that many companies are looking into developing products for this new standard network, joining the few trailblazers that are making this happen. I am happy to make Jerry's layout one of the first to take advantage of this latest NMRA boon. I know as the layout gets larger we will have capacity to control whatever Jerry can imagine – just extend that LCC network node to node.....