NMRA 2019
Salt Lake City
Signaling with LCC - Overview
(Layout Command & Control)

Compiled by: Dick Bronson
RR-CirKits, Inc.

Signaling with LCC.

Part 1 (overview)
www.rr-cirkits.com/clinics/NMRA-2019-Signaling with LCC-A.pdf
Part 2 (details)
www.rr-cirkits.com/clinics/NMRA-2019-Signaling with LCC-B.pdf

Layout Command Control

What is LCC?

LCC is an information highway for your model railroad layout

What is LCC?

- **LCC is a common language for layout elements to talk to each other**

- Signals
- Turnouts
- Detectors
- Lights
- Panels
- PCs / Smart Phones
- Boosters
- Command Stations
- Throttles
- Power Managers
- Trains
- etc…

What is LCC *NOT*?

LCC does NOT replace DCC.

On the track – DCC

Beside the track – LCC

LCC is not dependent on DCC,
can run on DC or Märklin layouts
not locked to the DCC manufacturer

Why LCC?

- The critical role of software and its importance to our hobby, both now and for years to come, cannot be overstated. Our new digital world and its eventual successors will underpin much of our future innovations across a wide range of technological advancements.
- The ability of LCC nodes to be quickly and easily upgraded over the bus without the need to return them to the manufacturer, purchase replacement chips, or to even access them physically is a key new benefit.

Why LCC?

- I once heard it said that LCC was a solution looking for a problem, because we already had many ways to control our layouts.
- That is true, and it points out the problem. We have LocoNet, CMRI, XpressNet, MERG, plus other proprietary methods to connect our devices.
- Not one of these options can connect to another. (except via JMRI and a central computer)

Why LCC?

- Many of us simply use the DCC itself to control devices. That has its own problems.
- First and most restricting, DCC is a one way street. Has anyone here ever seen a DCC connected block detector, fascia control button, turnout position feedback contact, or any other input?
- Second, DCC does have a fixed, limited, bandwidth. Control traffic competes with the repetitive locomotive control information. This is probably not an issue on a 4x8 class layout, but not a good basis for expansion of a large sized layout over the next 20-30 years.

Why LCC?

- DCC is a Master–Slave system. This means that there is really no practical way for more than a single command station to control any given layout.
- DCC has a fixed address space. This was seen to be sufficient in the 20th century when it was designed, but today folks run into the limits, and tomorrow requires looking for new tools.
- Any single LCC node has a larger reserved address space than many DCC systems can muster.

Other Options

- What about CMRI, XpressNet, MERG, plus the LocoNet and other proprietary methods used to connect our devices?
- Most of these solutions originated due to the difficulty in using the DCC bus for any input information.
- Some are Master–Slave and others have limited accessibility due to licensing.

A solution is proposed

- The NMRA decided a number of years ago (2007) to sponsor an open (license free) method to interface to your layout. The intent was that, like the NMRA DCC standards, many manufacturers would be able to build layout accessory products that will interchange as freely as is now true for DCC mobile decoders.

A solution is proposed

- The bus must use license free commercial standards for its communications as much as is possible.
- It should be robust and viable even into the next generation of electronic products.
- It should be a peer-peer design with no requirements for any central control station.
- Any two devices from any manufacturers must be able to exchange data.
- The Open LCB group was chosen to develop this.

- The result was a set of protocols that can be sent over any media. For example, EtherNet, Wi-Fi, CAN (Control Area Network), and others. (some have said tin cans and string, but don't believe it)
- The NMRA calls this Open LCB standard LCC. Layout Command and Control. LCC is NOT a replacement for DCC. (unless you consider it replacing DCC accessory decoders)
- LCC can run along side of DCC, AC, DC, DCS, TMCC, RailPro, Battery power, etc. It is not a way to power your trains, it is a way to control your entire layout.

LCC Basic Concepts

- One of the early assumptions made by the OpenLCB developers was that the nodes should be peer-peer, globally unique, and self describing.
- Without these features LCC has little to differentiate itself from other legacy solutions. Granted it can be faster, and more reliable, but that is not in itself any reason to make a fundamental change in how we do things.

LCC Basic Concepts - Peer-Peer

- In general networks come in two varieties. They are **Master–Slave** and **Peer–Peer**
- As implied, **Master–Slave** has a master device that controls all communications with the remote nodes. (slaves) Usually this is done in a round robin method where the master unit polls each slave in turn to send/receive data.
- **Peer–Peer** on the other hand allows each node to communicate directly with every other node on an equal footing.

LCC Basic Concepts - Peer-Peer

- The advantage of a **Master–Slave** network is that the master controls the network timing and there are no collisions to detect and/or resolve. The big disadvantage is that nodes can not know the status of any other nodes. The (single) master device decides what will be done with all information.
- The advantage of a **Peer–Peer** network is that any/every node can watch and/or act on information from any other node. The disadvantage is that each node needs to monitor for any message conflicts and resolve them.

LCC Basic Concepts - Globally Unique

- In addition, every legacy model railroad control network that I am aware of is plagued by an identity crisis. To communicate on a network the first order of business is to give each device an ”address”. This is required in order to create a sense of identity and give order to network communications. Usually this is the first step in configuration. For example, you are not allowed to simply plop down a new locomotive on a layout and expect it to operate in an independent manner without first assigning it a unique (to that layout) address.

LCC Basic Concepts - Globally Unique

- The OpenLCB design folks recognized this flaw, and established an addressing convention large enough to allow factory preassigned globally unique addresses to each potential LCC node, world wide. Each NMRA DCC manufacturer could build over 16,000,000 nodes before they will run out of their own unique addresses and need to apply to the NMRA for a new address range. I think that the OpenLCB guys have this requirement covered for long enough that I don’t need to be worried about it.

LCC Basic Concepts - Globally Unique

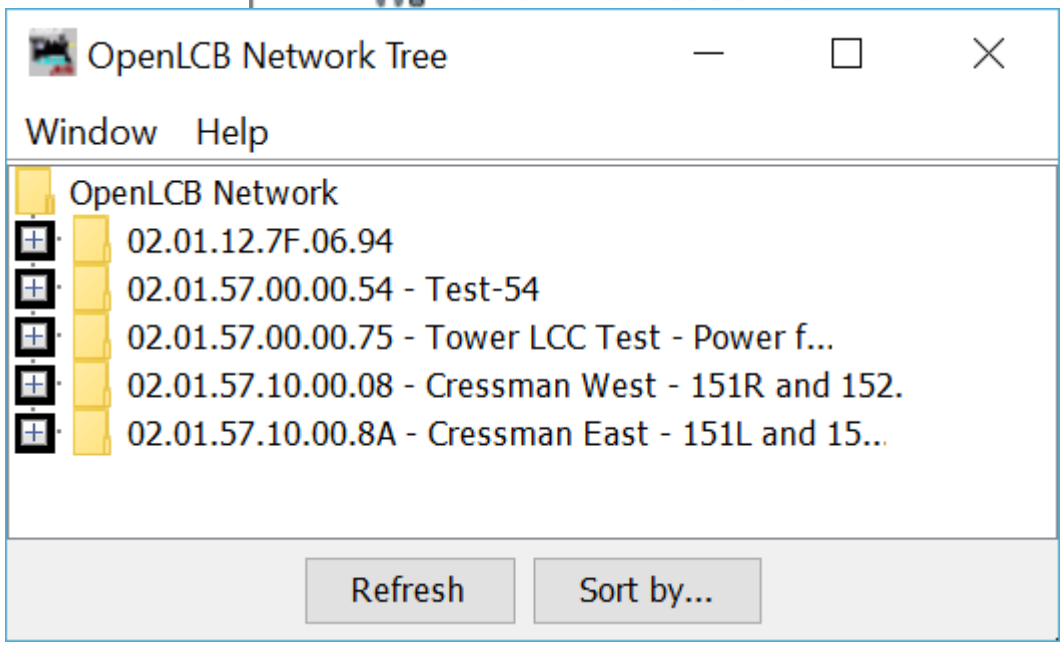
- Some of you may be old enough to remember when adding a serial port or printer port to your computer required you to set address switches and IRQ option jumpers on the I/O board.

- Is there anyone here today that did that with the latest computer that they purchased?
- Is there anyone here today that had to assign the address of the latest piece of model railroad electronics that they purchased? Enough said...

LCC Basic Concepts - Self Describing

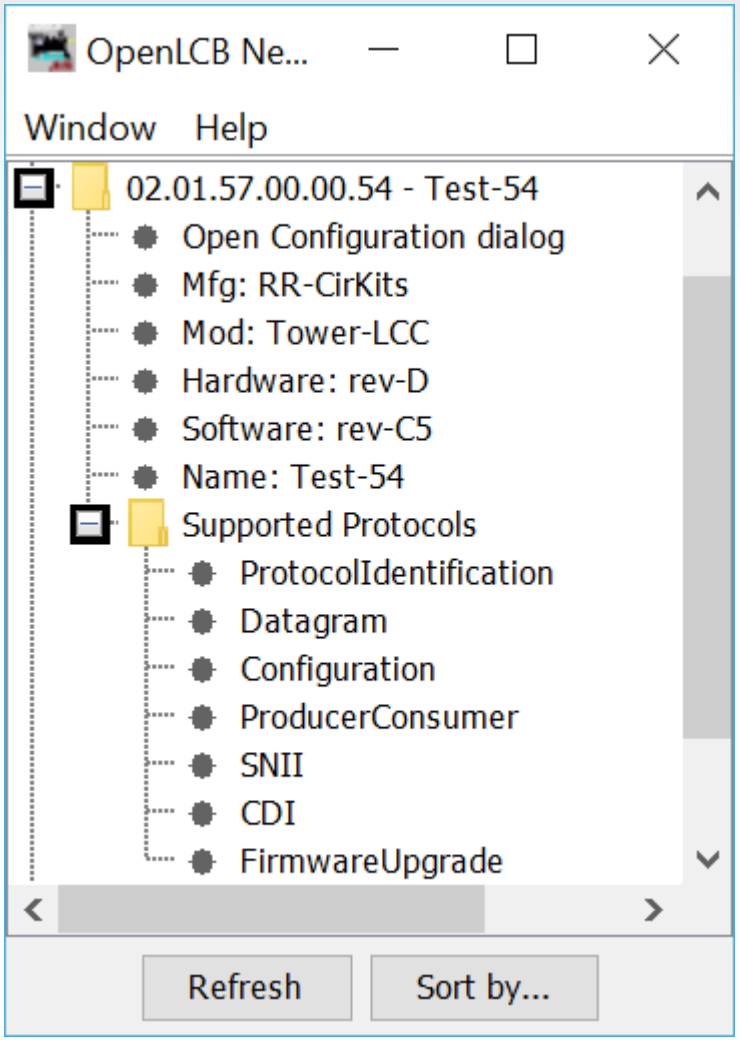
- When I add a new device to my modern computer, I expect it to automatically show up in the device manager as seen here.
- Why shouldn't we expect the same thing when we add a new node to our layout?

  Now, with LCC, that is a 21st century reality.



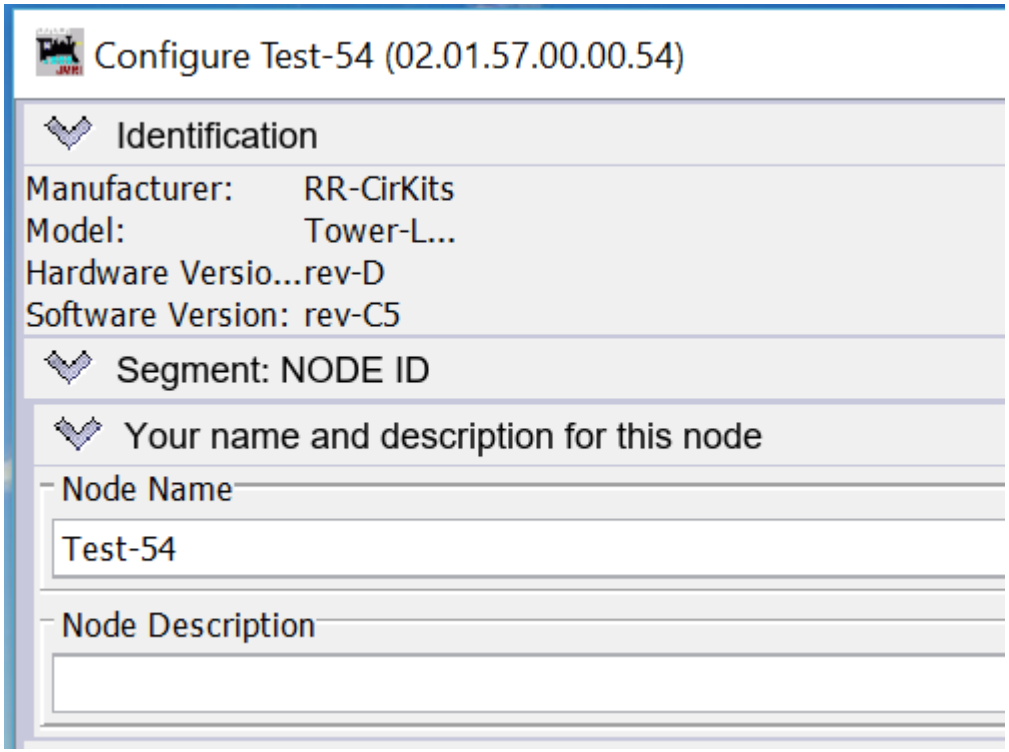
LCC Basic Concepts - Self Describing

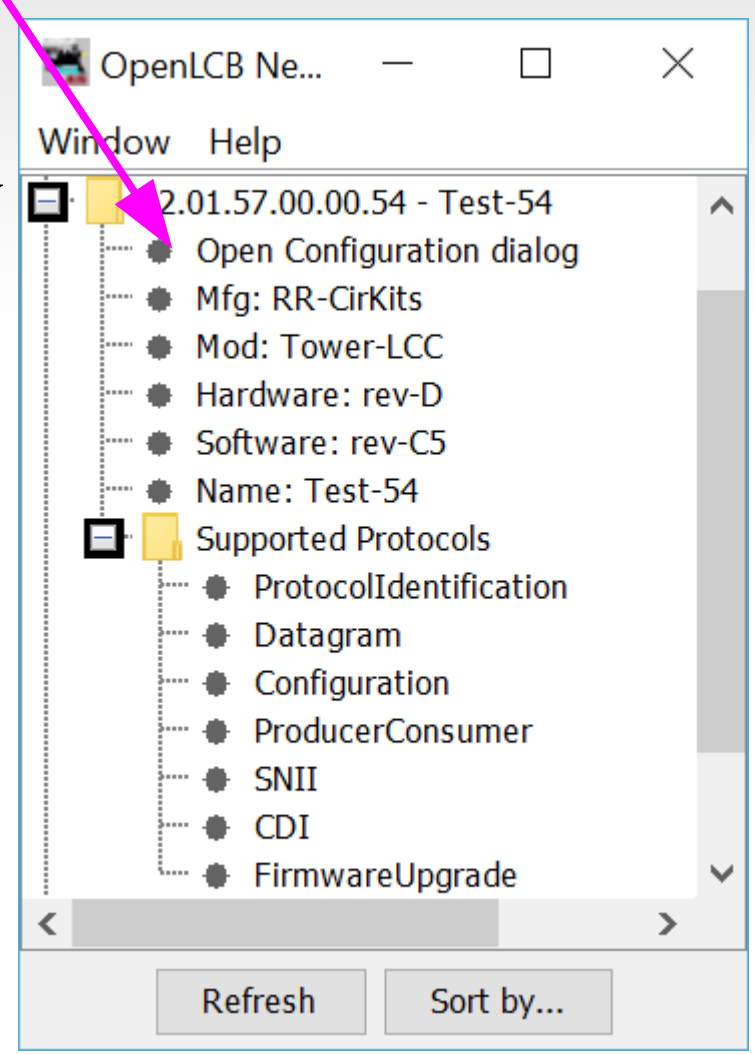
- With LCC you don't tell the system what hardware you have added. The system tells you what hardware you have added, what you have named it, and what capabilities it has.
- With DecoderPro if you want to program a decoder you need to first find the correct "decoder file" and then place the decoder in programming mode, maybe with a jumper or putting it on a dedicated 'programming' track.



LCC Basic Concepts - Self Describing

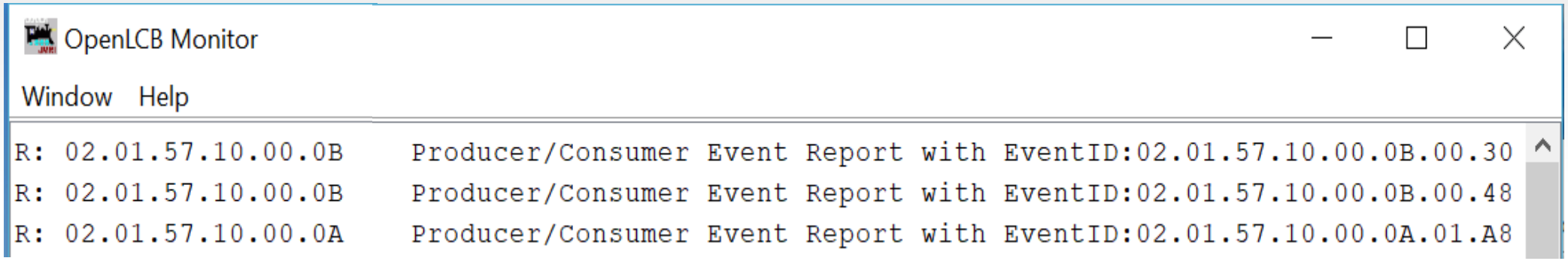
- With LCC you simply select ”Open Configuration dialog” and all required information is read from the node itself.
- This opens a DecoderPro like window that allows you to make changes.





LCC Basic Concepts - Event driven

- ”Event driven” means that the nodes communicate with one another by sending messages whenever something happens.



The event messages look like this, but you didn’t really need to know that, anymore than you need to know the format or content of the messages that your car’s O2 sensor uses when it sends messages to the ECU. If there is anything to remember, it is that 02.01.57.10.00.0B.00.30 (144,492,389,284,380,720) is a really large number that isn’t in any immediate danger of causing conflicts with other events from other devices on your layout.

LCC Basic Concepts - Event driven

- I will repeat this bottom line again here. EventIDs are simply magic numbers that represent your information on the bus, or over the air. There is no reason that you should ever need to type one out manually. There is little if any reason (other than curiosity) that you would ever need to know any details of what they mean. (which isn't a whole lot anyway)

LCC Basic Concepts - Event driven

- Its the Event ma'am, just the Event.
- In previous control systems that use a bus and events, (e.g. LocoNet and in a lesser sense CMRI) the events or messages sent on the bus have two parts, first an identifier number (address), and second the message type. This follows the original code line concept where each event was a hard coded station number plus one or more commands. For example: *turnout #23 set normal*.

LCC Basic Concepts - Event driven

- This is:

  1. a Turnout command
  2. for station #23
  3. set to normal

- A matching command with a predefined one bit different would mean *turnout #23 set to reverse.* Another one bit change would create *turnout #24 set to normal* etc.
- Sometimes the size of the DCC command space and the protocol design limits the number of possible options to a predefined set. (e.g. 2048 turnouts, 4096 sensors, etc.)

LCC Basic Concepts - Event driven

- For example turnouts only have two options, normal and reverse. If you have a three way turnout, (very rare on the prototype) sorry, you need to think of it as 2 two position turnouts. Have a three color signal, sorry, you need to think of that as either three different on, off, messages, (CMRI) or else combine two 2 position messages. (LocoNet) What about a more typical eastern US speed signal with 5, 6, or even more aspects?

LCC Basic Concepts - Event driven

- In the LCC world an event has no predefined meanings. None, Keiner, Nada! An LCC event simply says; ’something has happened’, or ’something should happen.’ How it is defined is 100% up to you, the user. In our previous example it could still mean *turnout #23 set normal.* However with LCC 'turnout #23' is just what you call it on your layout, not that it was pin 23 on some brand of hardware controller. *Set normal* just means that the event moves the turnout to normal. Undoubtedly you will want another event to move the turnout back, however that will be a completely different event with a different meaning. (e.g. *turnout #23 set reverse)*

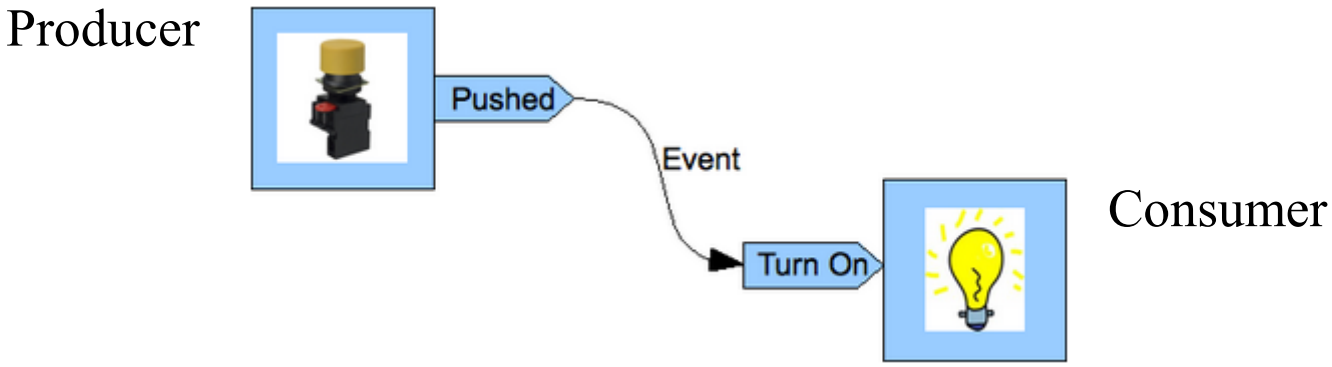
LCC Basic Concepts - Producer Consumer

- **Producer - Consumer** You will probably hear LCC folks throwing around terms like Producer and Consumer. They aren't talking about a big business takeover. They are just trying to sound educated. <G> The Producer-Consumer control concept is used for process controls and software queuing.
  - ***Producer*** simply means that some device can create (produce) an Event. Some examples might be a push button or block detector.
  - ***Consumer*** just means that some device can respond to (consume) an Event. It could be a lamp, a turnout driver, or anything else that you can control.
  - ***Events*** can have from 1 to many *Producers*. Events can have from 0 to many *Consumers*. Events are simply messages on the bus that say that something has happened or should happen.

http://openlcb.org/trunk/documents/notes/ProducerConsumerModel.html

LCC Basic Concepts - Producer Consumer

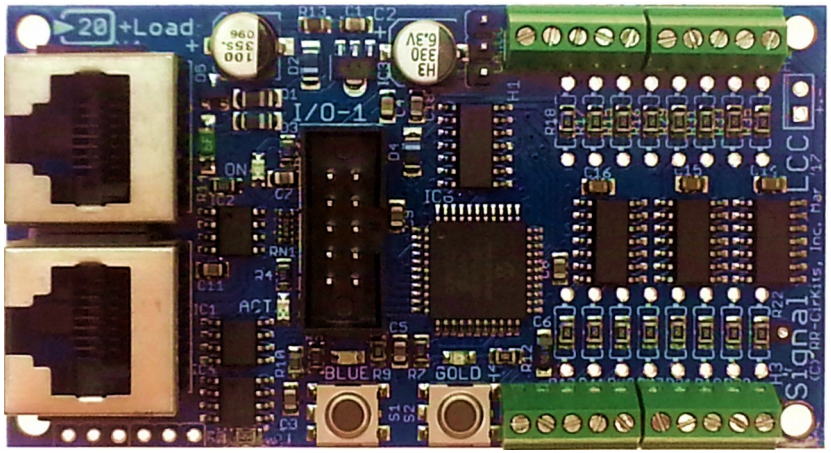
To elaborate a little bit. For an event to happen something must have sent it. Therefor there has to be at least one *producer*. In the LCC world it is possible for many different *Producers* to create the same event. For example you might want to have turnout control buttons track side and on a remote panel. Thus the statement that every Event has one or more *producers*.



LCC driving Signals

- Enough about LCC Basics.

  Today you are here to see how LCC can be used for driving signals on your layout.



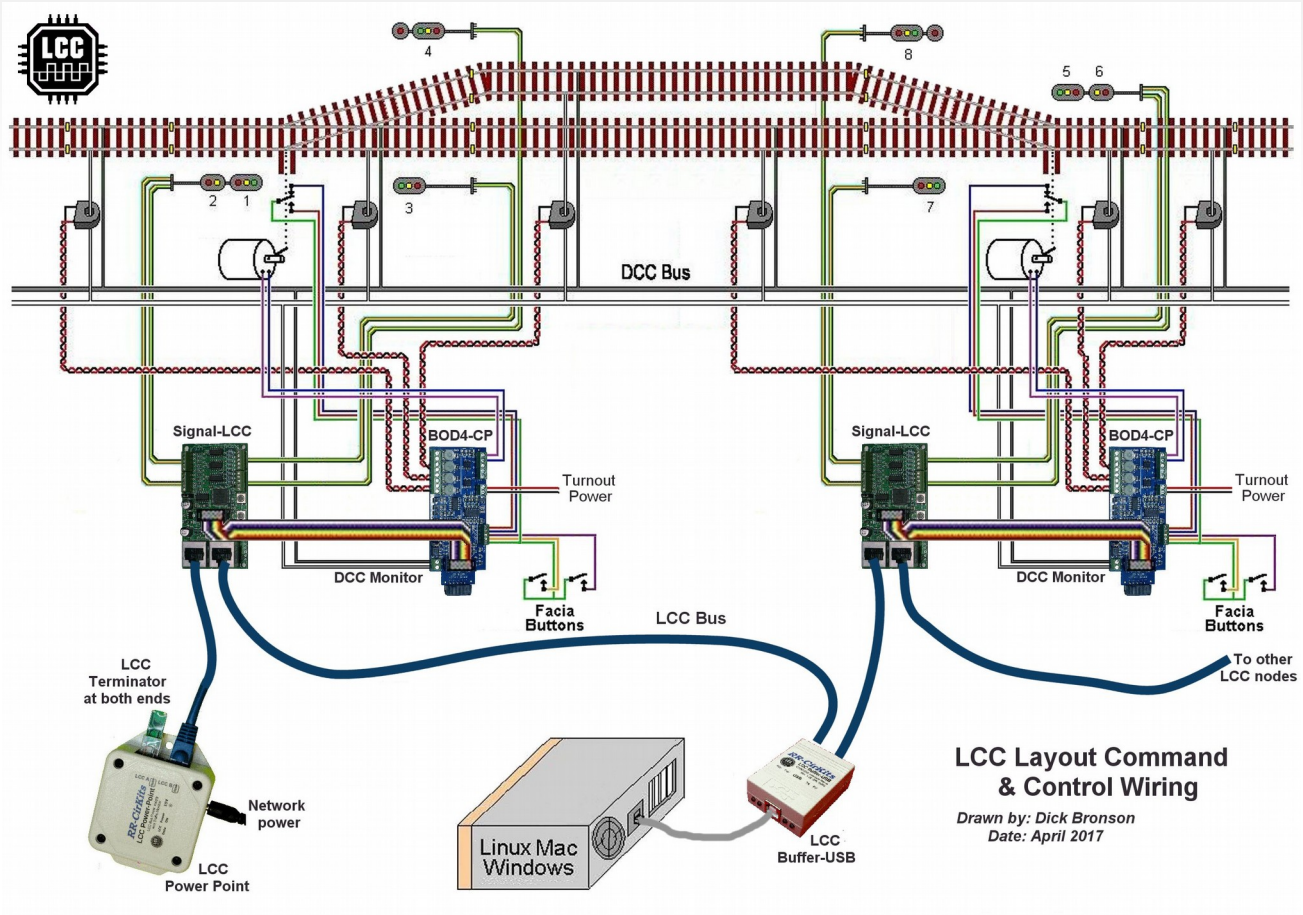
+

=

LCC driving Signals

- Well, actually its more like this.

Nobody said it wouldn't take wiring, but note that its all localized.



LCC driving Signals

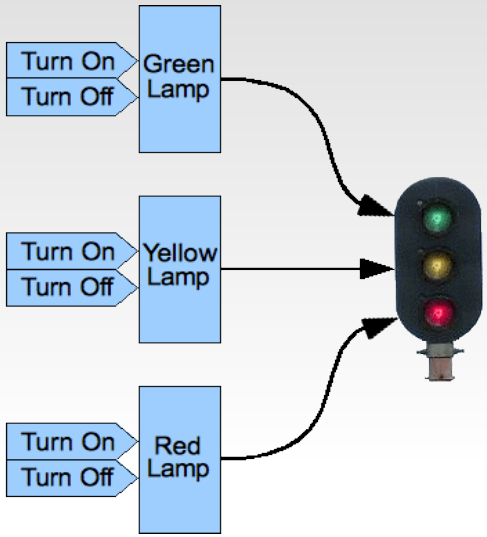
- Application to Signals

  Signaling usually requires more logic than can be handled via simple Events, e.g. occupancy, turnout position, look ahead to the next signals, etc. However a signal controller could be designed to listen to all of the appropriate Events and fully control the signal aspects. Note that it's also useful for a signal system to emit (produce) Events for each aspect change so that e.g. a control panel can mirror the appearance of the on-layout signals, or so that the next signal can know its aspect.

LCC driving Signals

- In the following examples we will compare different methods of controlling signals. This varies from individual LEDs to a full blown track side control point.

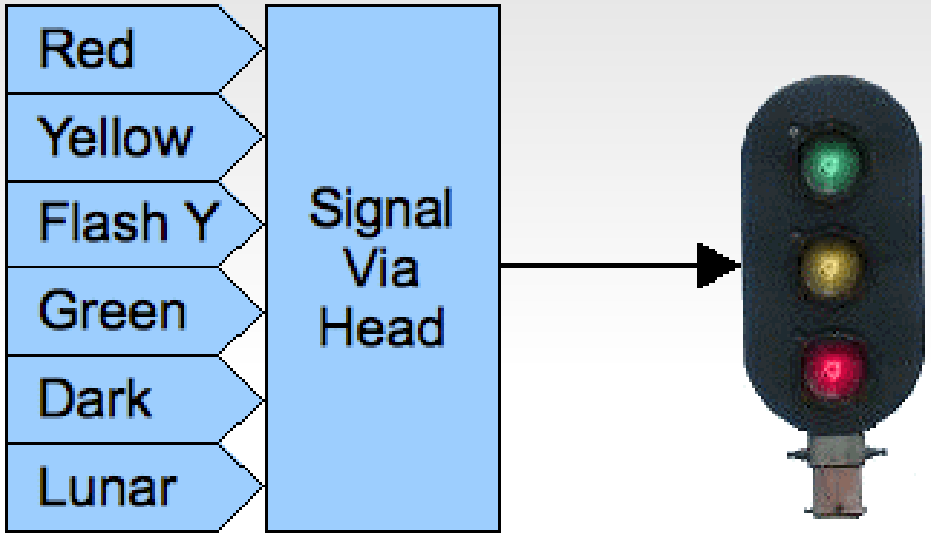
- # Signals via individual lamp drivers
- You can connect the lamps of a signal head to individual Consumers:



- This is a powerful but complicated approach. It requires that the controller individually turn each lamp on or off. This can cause excessive control traffic and that latency causes poor timing of flashing signals. There is also a cost/complexity trade off where lower cost drivers (more outputs per board) requires more wiring. Typically simple drivers lack special effects like fading and flashing.
- This is the method used by CMRI.

- # Signals via individual head drivers

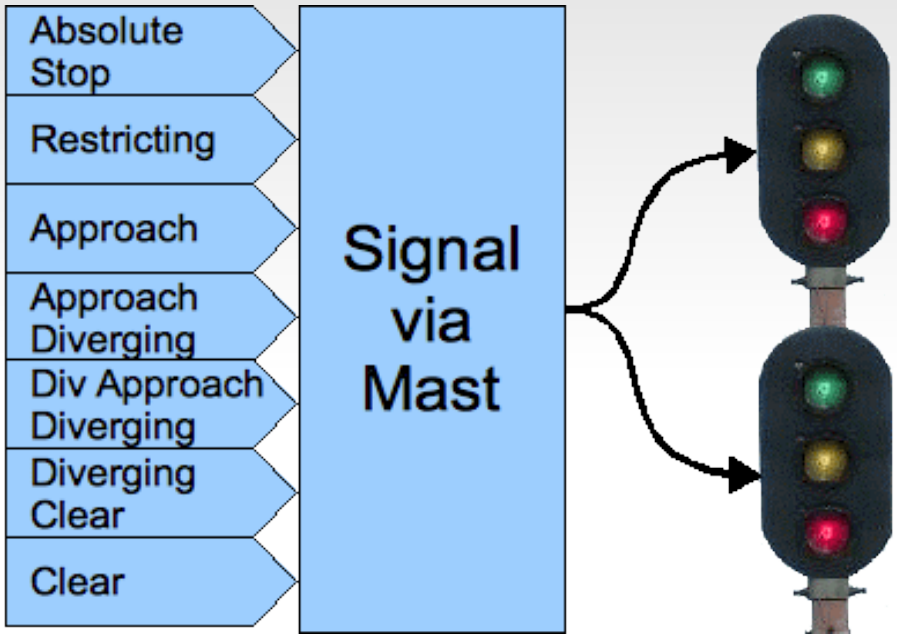
  You can also control signals with Events for the specific colors or functions of a single head.



- This method requires less command traffic than the previous one. However, if the controller does not know how to flash the signals, it may still result in constant streams of messages to be able to show flashing aspects. The Digitrax SE8c falls into this category. It normally only displays Green, Yellow, Red, and Dark. To show 'Flash Y' you need to alternate between sending Yellow and sending Dark. Got Lunar?

- # Signals via aspect drivers

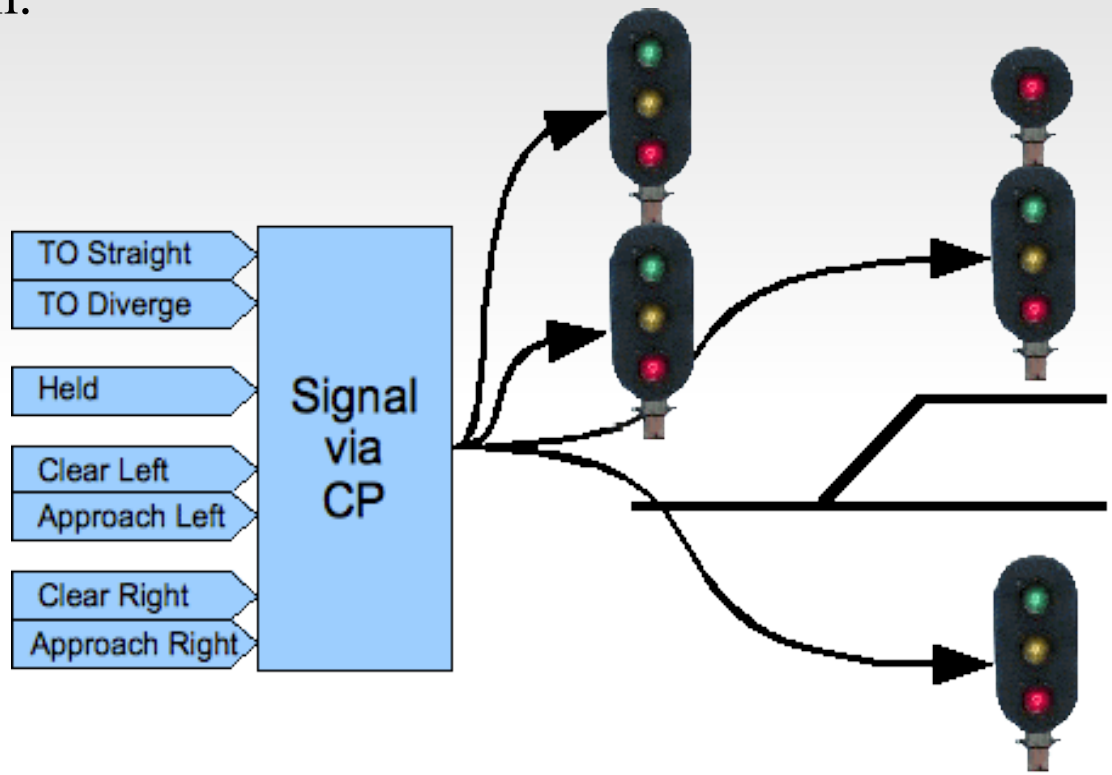
  You can control an entire signal mast with just one Event for each high-level aspect of the signaling system.



- This method requires the minimum amount of command traffic to control the signals themselves. However it still requires an external controller or a program such as JMRI to monitor the layout and calculate the proper aspects. The Team Digital SHD2, Signalist SC1, and RR-CirKits SignalMan in NMRA Signal Aspect mode fall into this category.

- # Signals via control point drivers

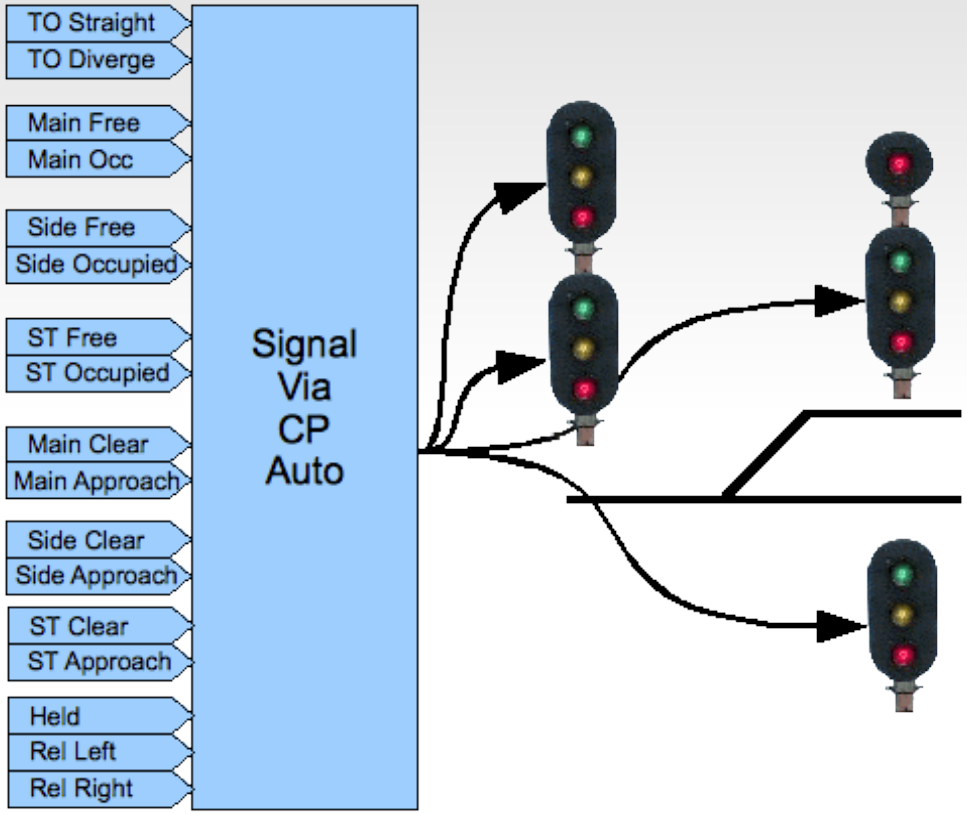
  You could also control an entire interlocking with just single Events for each high-level condition of the signaling system including turnout position.



- This method is similar to the signal aspect driver, but includes turnout control and possibly even occupancy detection on the same node. However it still requires an external controller or program such as JMRI to calculate the proper aspects. The original RR-CirKits LNCP is similar to this option.

Integrated Signals

In each of the examples above, the signal controller uses (consumes) Events that directly control the appearances of the signals.



- It's also possible to build a signal controller that watches all related status Events from the railroad and CTC panel and makes independent decisions about the proper signal states and appearances. This type of controller would be able to control its signals without any external computer involvement.

Configuration of LCC nodes

- One of the key new concepts in the LCC protocol is that, not only the configuration, but the ’decoder file’ (in JMRI terms) itself should reside in the LCC node. This is an important change from the status quo.
- Originally hardware had a fixed purpose. Each required its own dedicated connections. Lionel crossing gates flashed with contacts triggered by the passing wheels. (blink-blink....blink-blink....)

Configuration of LCC nodes

- Then some devices were connected to a bus. (or track) This required assigning addresses or channels. The usual solution for addressing was to include a set of jumpers or switches for the selection. In some cases it was a plug with different component values.
- As electronics improved the selection of addresses was moved into the device code itself. An example that we are all familiar with is modern DCC mobile decoders.

Configuration of LCC nodes

- One of downsides of this new method is that our decoders now need to be configured with a new (non default) address. That itself was automated by some manufacturers, but it soon became evident that something more was needed than simple interactions through a hand held throttle. Some of today's new decoders have 1000 or more values (Cvs) to configure.

Configuration of LCC nodes

- JMRI and other programs have come to the rescue, but the decoders are now so complex that a ’decoder file’ is required for each locomotive and stored on a computer to help keep track of changes. The DCC specification does not include an easy way to read information out from a decoder except very laboriously and slowly over a special connection. (called a programming track)

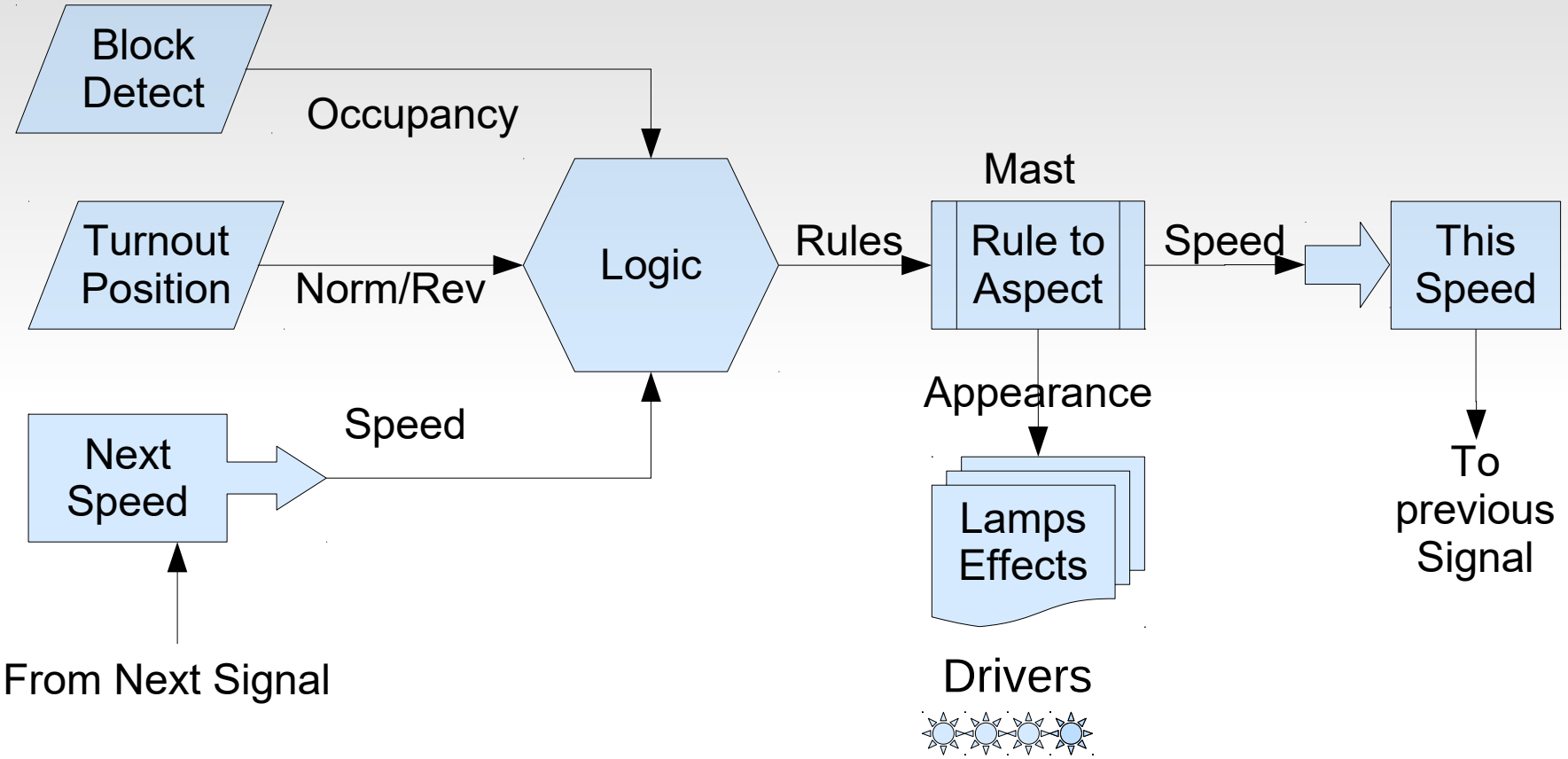
Configuration of LCC nodes

- This manual address assignment was deemed to be too error prone and inflexible for the new LCC equipment. Two key changes were required.
- The first was that any LCC node could be configured in place on the layout at any time with no need to access it for jumper changes or button presses.
- The second was that any information required to configure a node should reside in the node itself, and be available to any configuration tool connected to the network. Now any node could be configured in one place and moved to another with all the information moving with the node itself. This means not only configuration values but user names and comments as well.

Configuration of LCC nodes

- As previously mentioned, a key design choice of LCC was that the manufacturer would assign a node ID during manufacturing in a manner that prevents any duplication of addresses…. Ever, …. anywhere! (similar to how Ethernet MAC addresses work)
- This manufacturer based address assignment has another unforeseen benefit. Any automatic or user linking of two LCC nodes no longer needs to know anything at all about the rest of the network in order to prevent unintended conflicts We will take advantage of this for signaling.
- Adding a new LCC node to the layout will never conflict with any already installed devices.

Signal Logic Example



The basic signal logic overview.

- Rule logic is calculated using layout status information and next speed.
- The resulting ‘Rules’ are converted to lighted lamps, effects, and speeds.

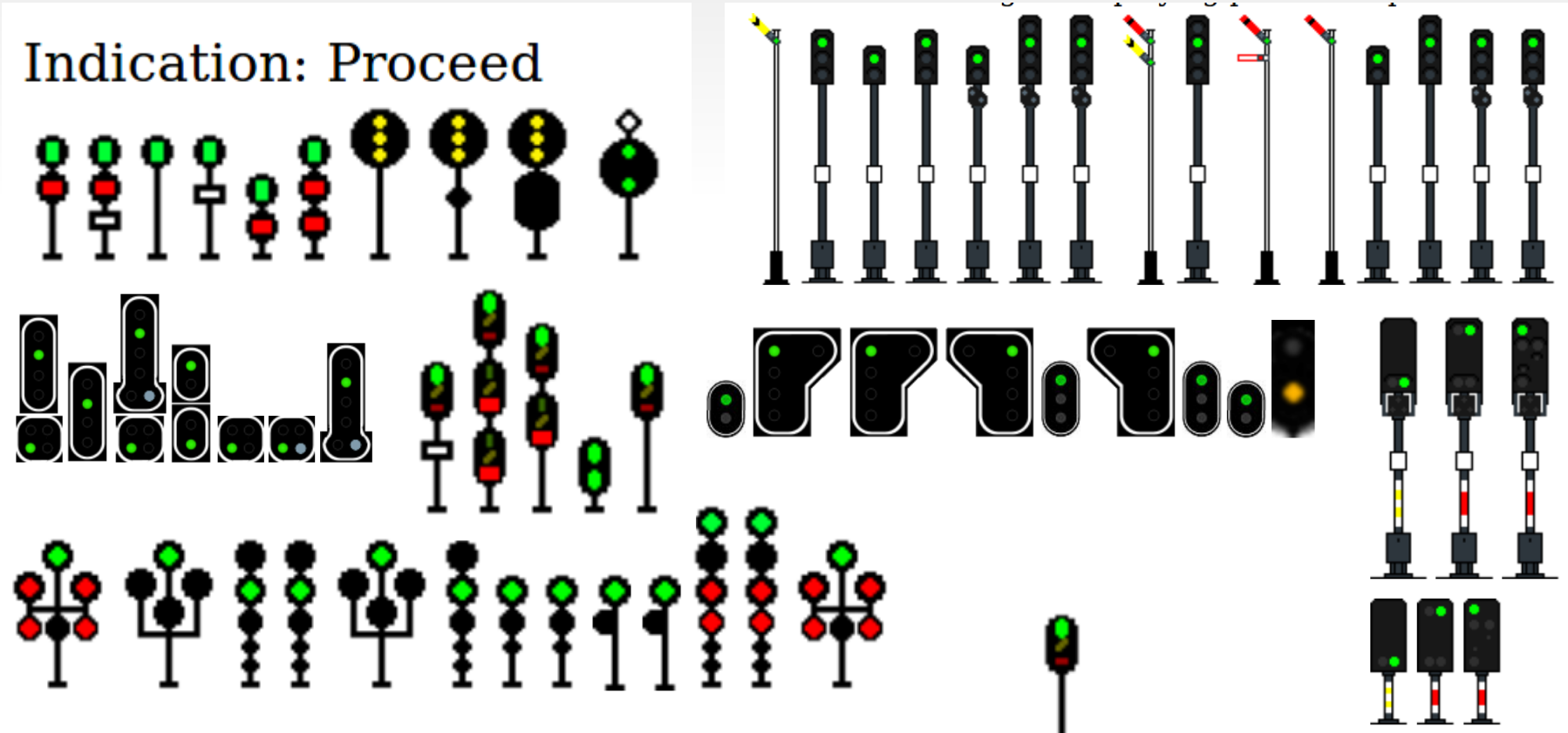
Signaling Requirements

- Signal Logic
- The heart of a signal controller is that it watches all related status Events from the railroad and CTC panel, (if used) and makes independent decisions about the proper signal states and appearances. It is the logic brain for the signal.
- The signal controller may be built into each signal board. It may be an external computer program like JMRI. It may be a dedicated logic board similar to the Team Digital CSC. (Central Signal Controller) It may be created from general purpose logic built into various nodes on the control bus. This is how our RR-CirKits LCC products work.

Signaling Requirements

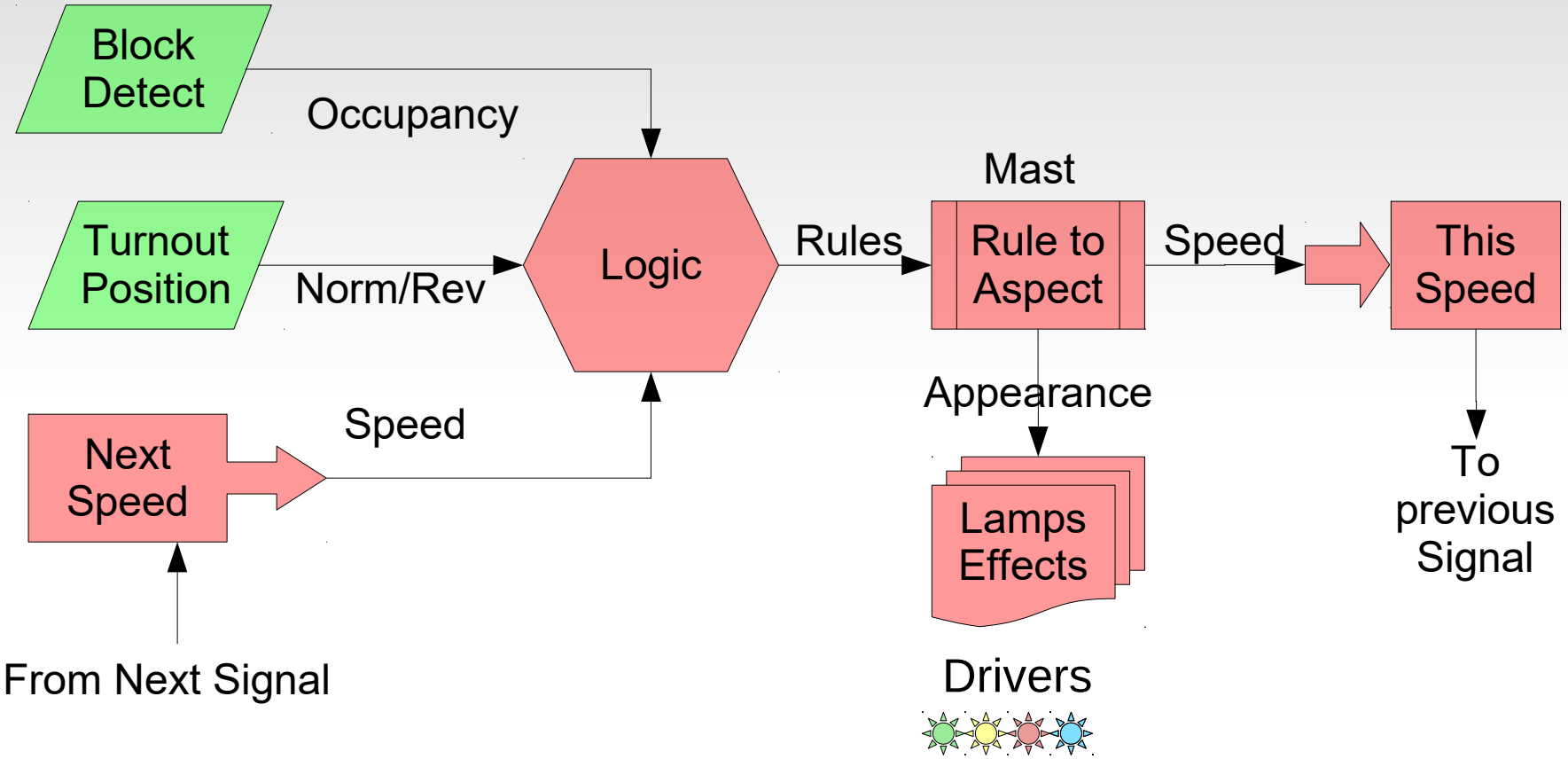
- Rule to Aspect conversion

Signal rules such as ’Stop’, ‘Approach’, ’Clear’, Etc. are displayed differently on different types of signals. A simple way to make these conversions is needed.



- My point of course is that a signal board designed for may not work for you.

Signal Logic Example



In typical existing systems the green items are part of the layout hardware, and the red items are taken care of by an attached computer.

Items such as Lamp Effects are difficult or impossible for the computer to accomplish well, due to interface latency and driver restrictions.

Signaling Requirements

- Track Circuits

  In order to properly calculate signal rules the signal logic must know the allowed speed upon approaching the next signal along each route. Prototype speed information is often sent from one mast to another over track circuits.

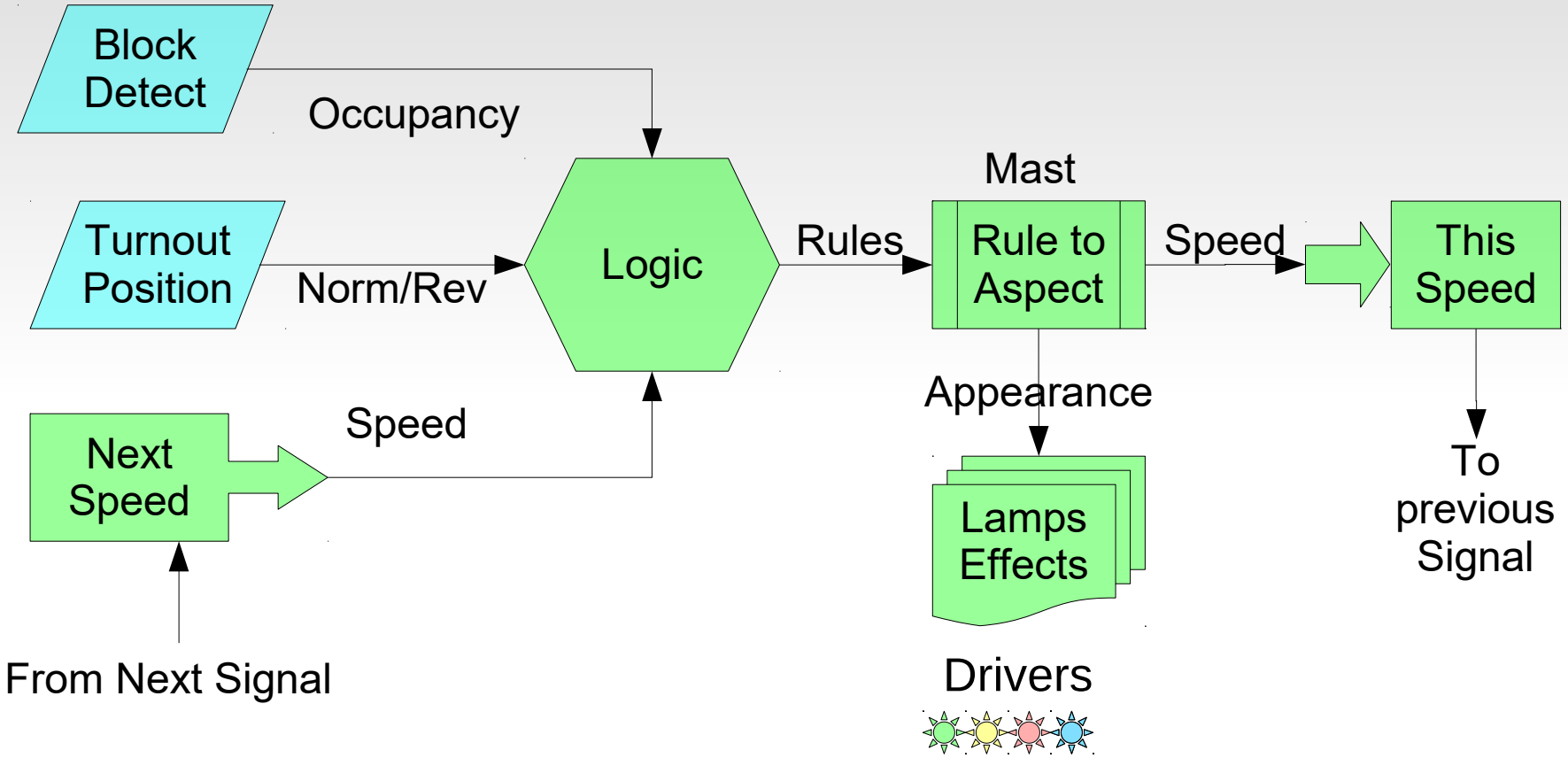
- Effects

  Signal lamps usually do not simply blink on or off as they change. Effects simulating incandescent lamp fade and other visual artifacts can increase the realism of our model signals.

- Brightness

  If we can fade our signals, then we should also be able to adjust their brightness to make them visually match between different colors.

Signal Logic Example



With the Signal LCC all of the control functions required for signaling exist locally. Light blue items are on a daughter card or different nodes.

If you want to off load (or monitor) any function with a computer you may do so by intercepting the LCC EventIDs that link sections with each other.

Signal Logic

- Signal Logic
- In order to build a signal controller that watches all related status Events from the railroad and the CTC panel, and makes independent decisions about the proper signal states and appearances, it must contain internal logic. This logic must either be user configured or else it must understand all possible signaling rules.
- Triggering the evaluation of a conditional is done when any monitored event is seen. There are two trigger options. In the first option evaluation of a conditional is only done if the monitored event actually changes the state of the variable. In the second case the evaluation is done when ever the event is seen, even if there is no resulting change to a variable. This allows repeated single events to trigger a conditional multiple times.

Signal Logic

- We will cover more details of signal logic in the next session.

Currently Available LCC Hardware

- With the recent addition of an option to place the DCC rail sync information on an otherwise unused pair, CAN can now support smart boosters.
- I have referred to the CAN version of LCC. Remember that the LCC protocol is also capable of being used over different systems, Ethernet, and Wi-Fi also being developed for use by other LCC developers.

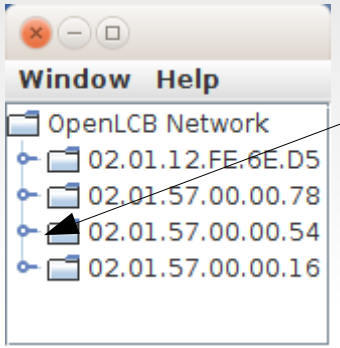
The Future of LCC

- *Smart Detector, Railcom, Circuit Breaker, Reversers
- Simple Detector, CT coil based.
- Stall Motor Driver (Support for ganged Tortoises, MP1, etc.)
- Dual Coil Solenoid Driver.
- *Servo controllers.
- *LocoNet to LCC Gateway. (LCC support for existing products)
- *Ethernet LCC Links.
- *Wireless LCC Links.
- *Throttles
- Smart Boosters, *Command Stations.

* denotes LCC nodes currently under development in 2019

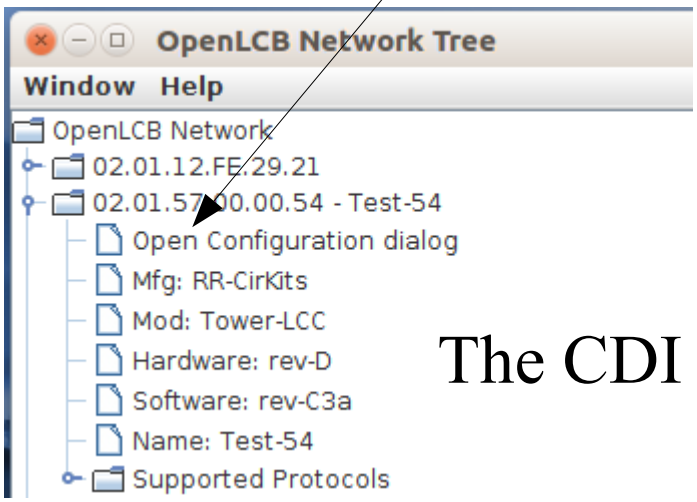
- The original CDI configuration tool was created as a part of JMRI. www.jmri.org

Select OpenLCB and choose ’Configure Nodes’



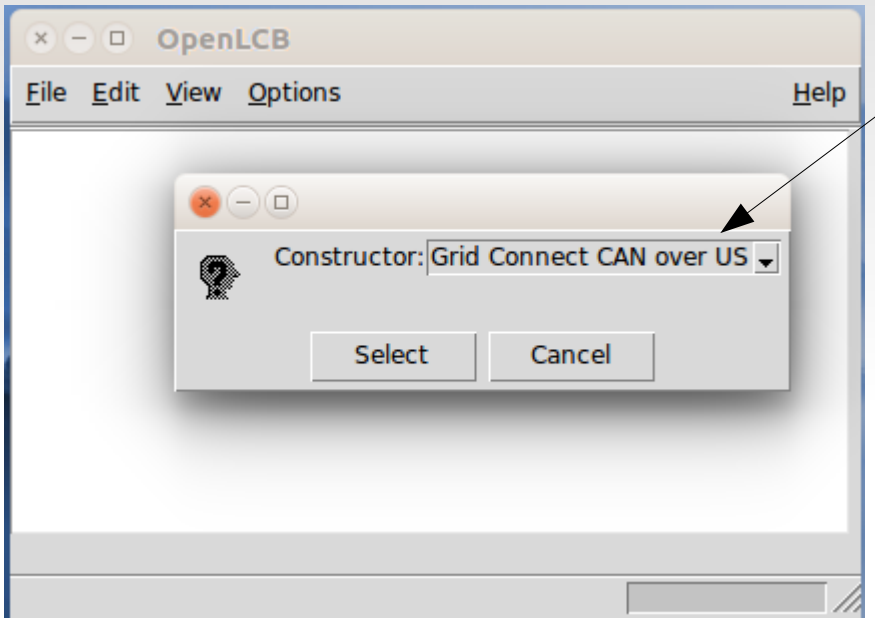
Next open the node you need to configure.

Open ’Configuration dialog’.

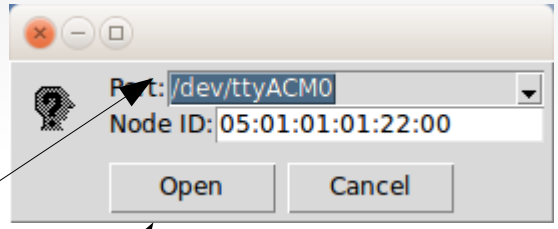


The CDI window will then open.

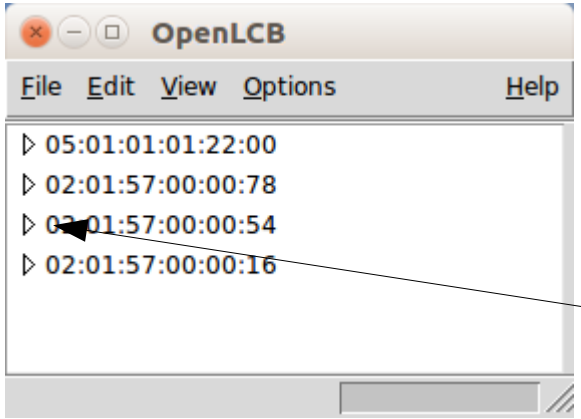
Because LCC is an open standard anyone can develop tools for it. One such developer is Robert Heller of Deepwoods Software. This is part of his model railroad software package.
http://www.deepsoft.com/home/products/modelrailroadsystem/downloadmr/
Run the OpenLCB tool.



If you are using the LCC Buffer-USB as your interface device, then select 'Grid Connect CAN over USB' .



Next select the proper COM port. (this example is on Linux)



Once you click on 'Open' a similar window to the one you saw in JMRI will open. The first entry is the program connection itself. The other entries are a list of the attached nodes.

As in JMRI, open the node you choose to configure by expanding its tree view.

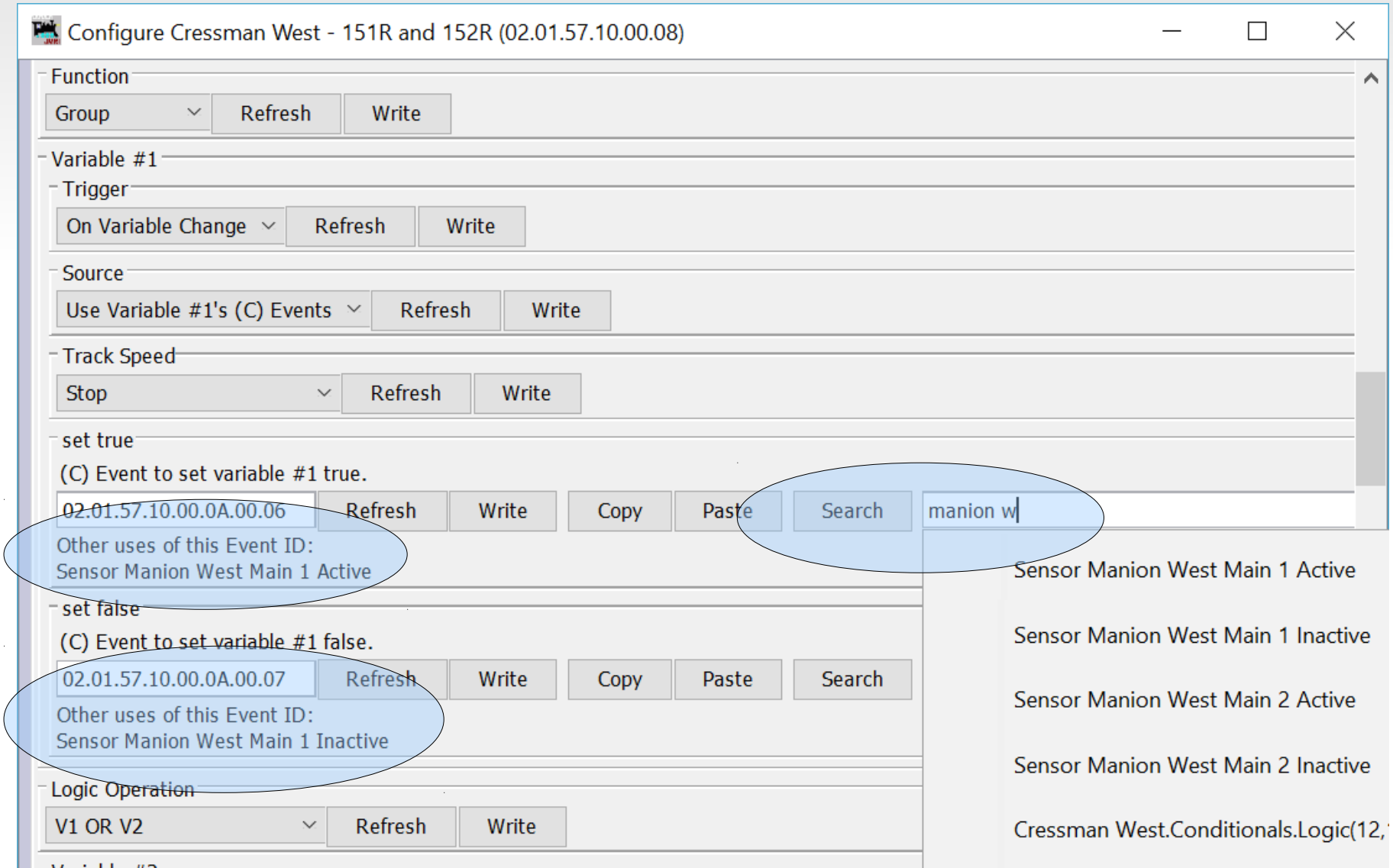
The Future of LCC

- I created the next slide back in 2017. I include it here for a little bit of perspective.

- Current configuration tools are still under development. One design target is to eliminate any reference to the actual EventID numbers, and simply use the users own names for items.
- I am not optimistic about seeing that in my lifetime, but once a line is configured you really can ignore the details of each EventID because you will not need to worry about any duplication, and you do not need to know them ahead of time to properly select the hardware like you do on existing networks. In LCC the hardware either offers you a new unused Event, or you may configure it to respond to your own already defined Events. (just copy your EventID to it)

The Future of LCC

- Look carefully at a current configuration presentation. Fortunately I have apparently outlived my pessimistic prediction.



Acknowledgements

Key OpenLCB Contributors: Bob Jacobsen, Alex Shepherd, David Harris, Stuart Baker, Balazs Racz, Jim Kueneman, Don Goodman-Wilson, John Plocher

Developer Group

10 to 15 actively working on code at any time
25 to 50 regular contributors and supporters
Many of the same people as supporting JMRI

OpenLCB User Group

Started November 2009
July 2019 we had over 290 members

NMRA liaison: Stephen Priest
NMRA w.g. chairman: Karl Kobel

Info

Users Groups:

https://groups.io/g/openlcb
https://groups.io/g/layoutcommandcontrol

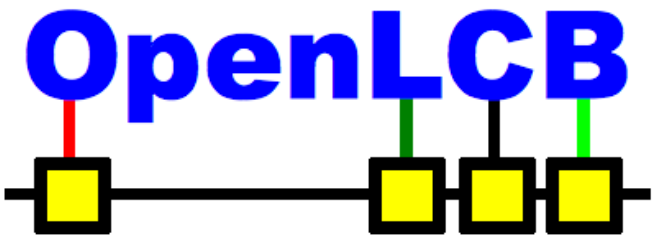
To Join: openlcb+subscribe@groups.io
layoutcommandcontrol+subscribe@groups.io

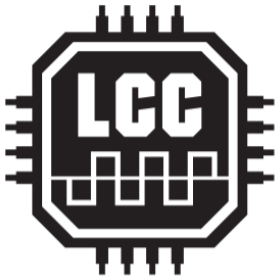
Useful Links:

http://openlcb.org or http://openlcb.com

http://nmra.org, choose S&RP scroll to 9.7

Book: Introduction to Layout Command Control
by Dana Zimmerli PhD





Questions

- ?