NMRA 2018
Kansas City
Signaling with LCC
(Layout Command & Control)

Compiled by: Dick Bronson
RR-CirKits, Inc.

Signaling with LCC.

Part 1 (overview)
www.rr-cirkits.com/clinics/NMRA-2018-Signaling with LCC-A.pdf
Part 2 (details)
www.rr-cirkits.com/clinics/NMRA-2018-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,
could run on DC or Märklin layouts
not locked to the DCC manufacturer

Why LCC?

- I have heard some say that LCC is a solution looking for a problem, because we already have to many ways to control our layouts.
- That is true, and it is part of the problem. We have LocoNet, CMRI, XpressNet, MERG, plus other proprietary methods to connect our devices.

Why LCC?

- Many of us simply use the DCC itself to control devices. That has two problems.
- First it is a one way street. Have you ever seen a DCC connected detector? (yes, Railcom could possibly do it)
- Second, DCC is limited in bandwidth, and competing with the repetitive locomotive control information.

Other Options

- What about LocoNet, CMRI, XpressNet, MERG, plus the many other proprietary methods to connect our devices.
- Most of these solutions originated due to the difficulty in using the DCC bus for any input information.

LCC Basic Concepts

- Its the Event ma'am, just the Event.
- In previous control systems using 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 station number plus one or more commands. For example: *turnout #23 set normal*.

LCC Basic Concepts

- 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

- 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?

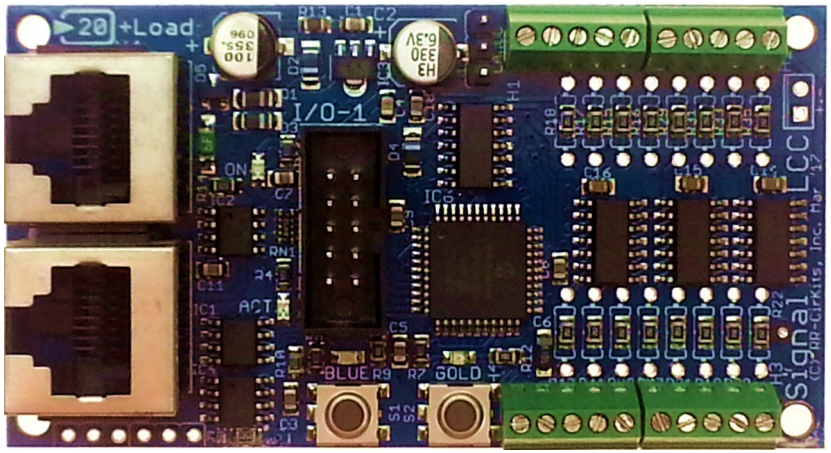
LCC Basic Concepts

- 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 driving Signals

- Enough about LCC Basics.

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



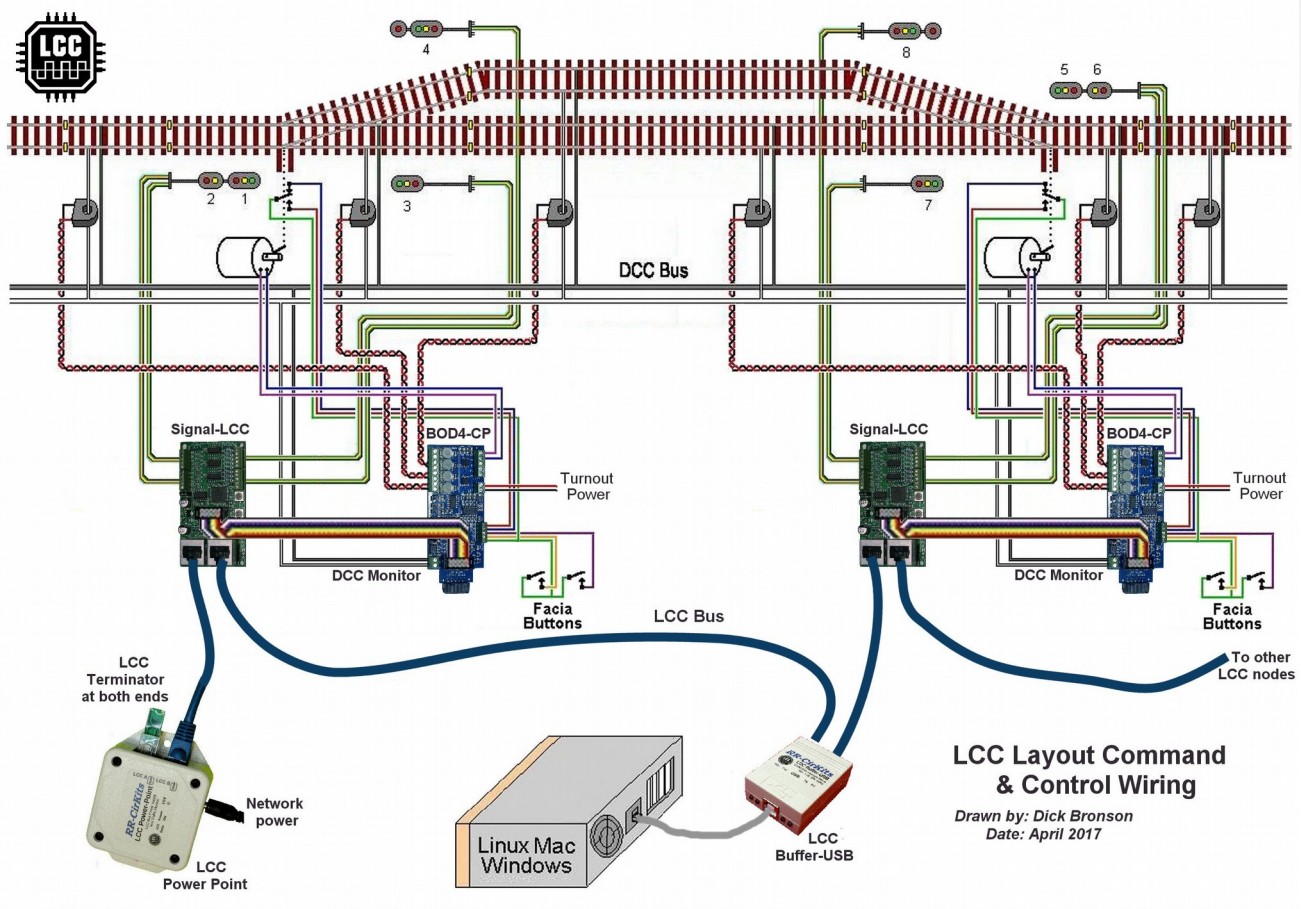
+

=

LCC driving Signals

- Well, actually its more like this.

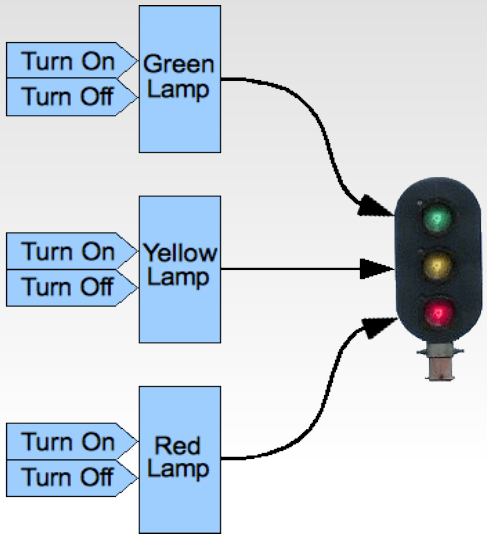
Nobody said it wouldn't take wiring, but note that its all localized.



LCC driving Signals

- 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:

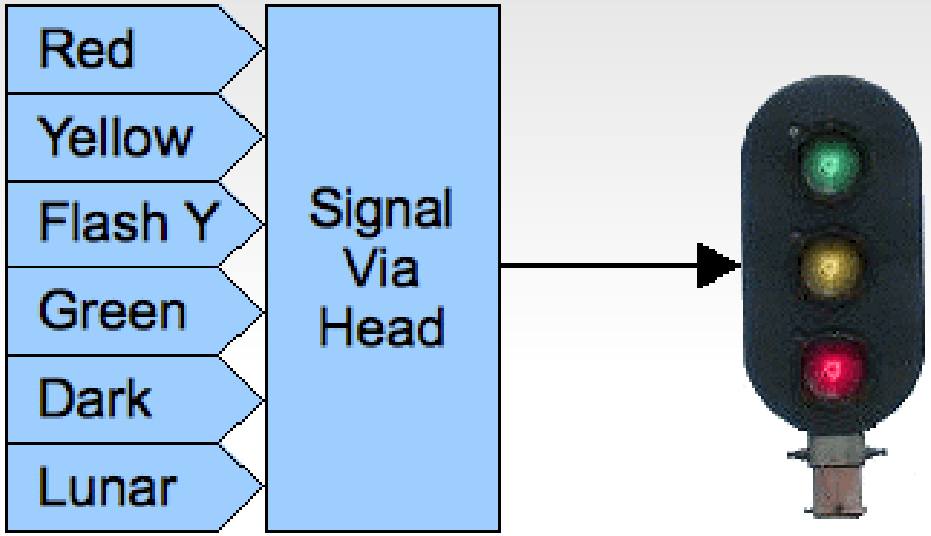


  This is a powerful but complicated approach. It requires that the controller individually turn each lamp on or off. This can cause excessive control traffic and latency causes poor timing of flashing signals. There is also a cost/complexity trade off where lower cost drivers (more outputs per board) 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

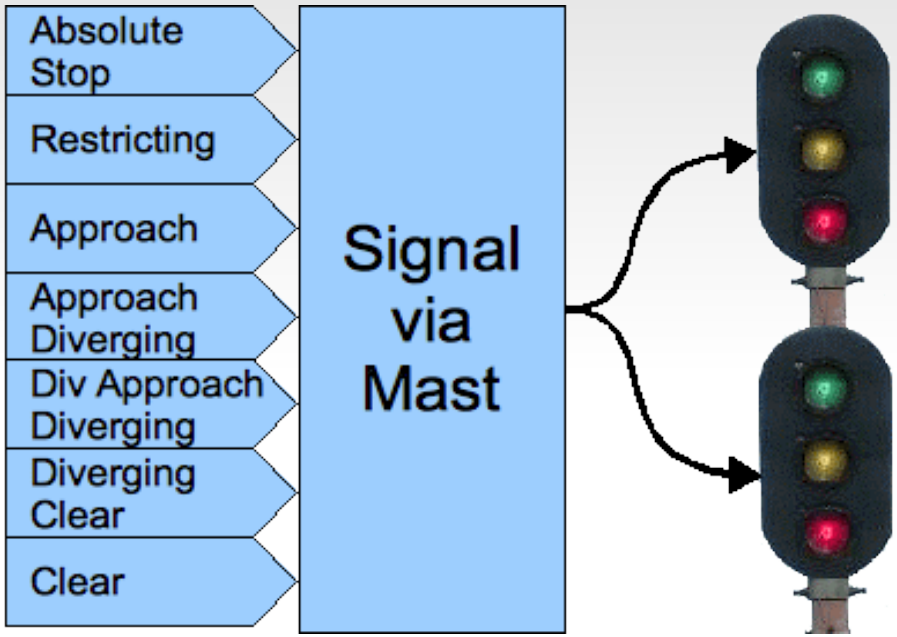
You can also control signals with Events for the specific colors or functions of a single head.



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? Nope!

- # Signals via aspect drivers

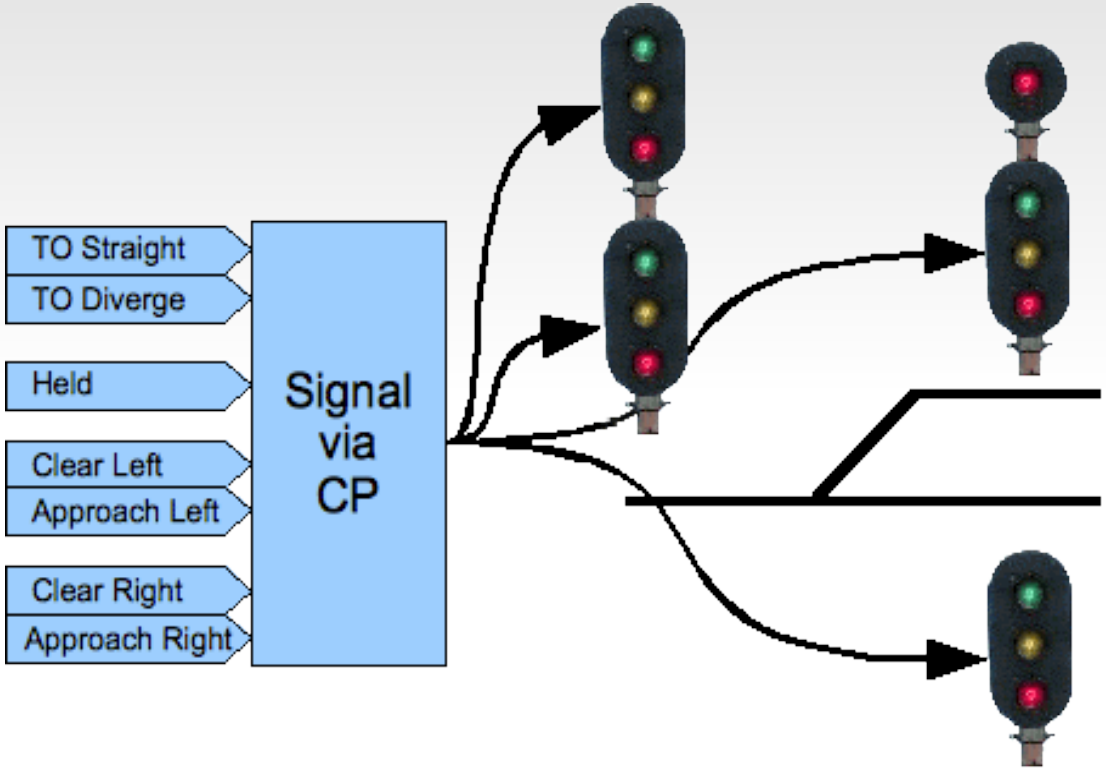
You can control an entire signal mast with just one Event for each high-level aspect of the signaling system.



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 our RR-CirKits SignalMan in NMRA Signal Aspect mode fall into this category.

- # Signals via control point drivers

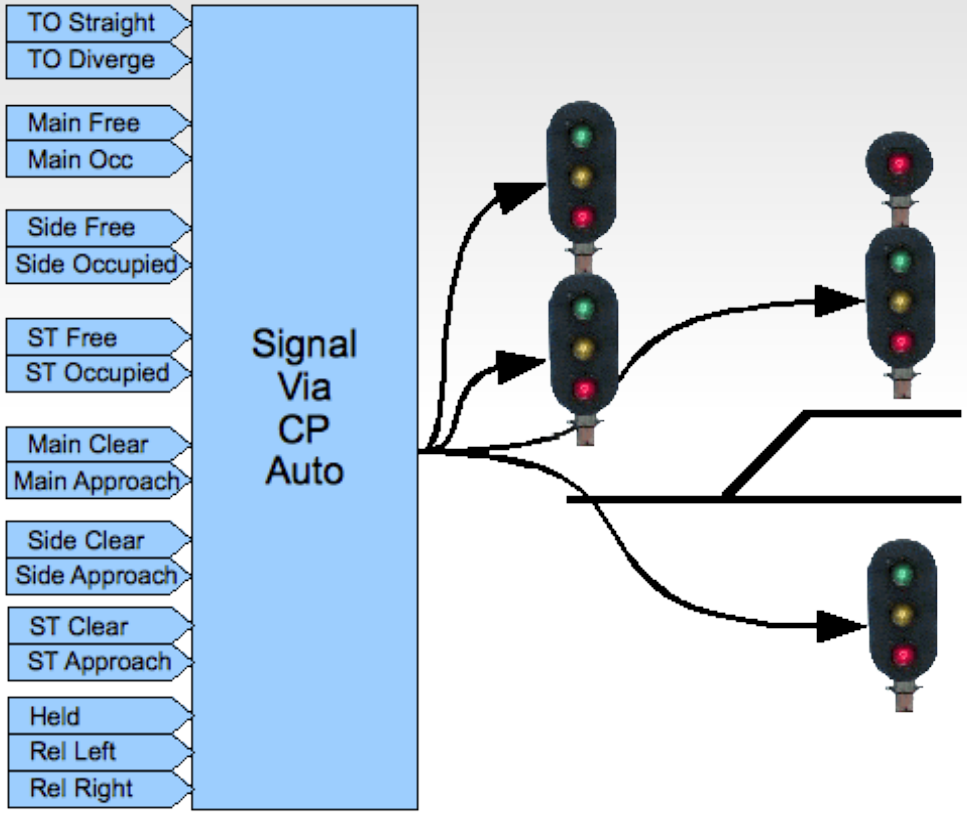
You could also control an entire interlocking with just single Events for each high-level aspect of the signaling system including turnout position.



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 new decoders currently have 1000 or more values 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 from a decoder except very laboriously and slowly over a special connection. (called a programming track)

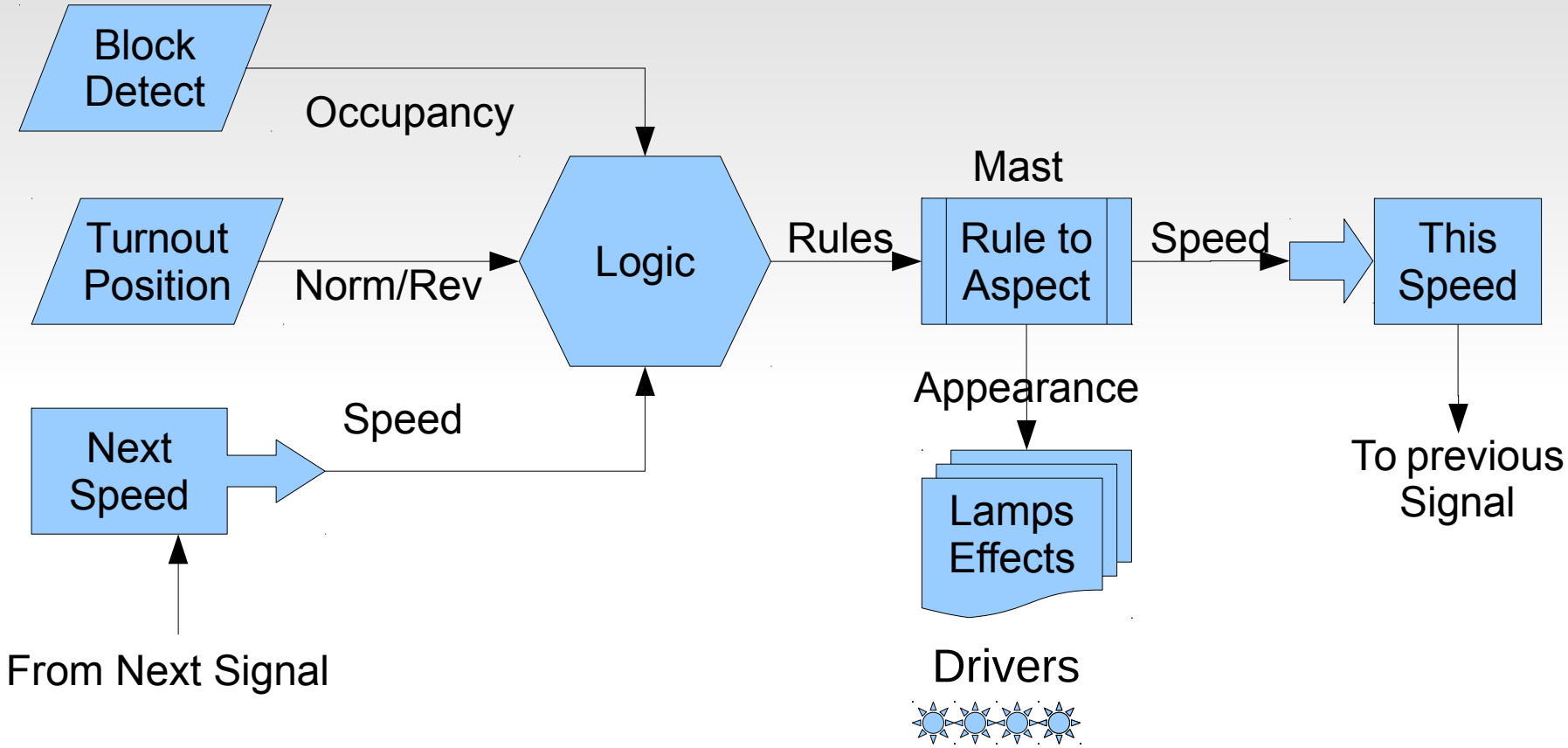
Configuration of LCC nodes

- This manual address assignment was deemed to be too slow 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

- Another 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 Ethernet MAC addresses)
- 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 layout in order to prevent unintended conflicts We will take advantage of this for signaling.
- Adding a new node to the layout will never conflict with any already installed devices.

Signal Logic Example



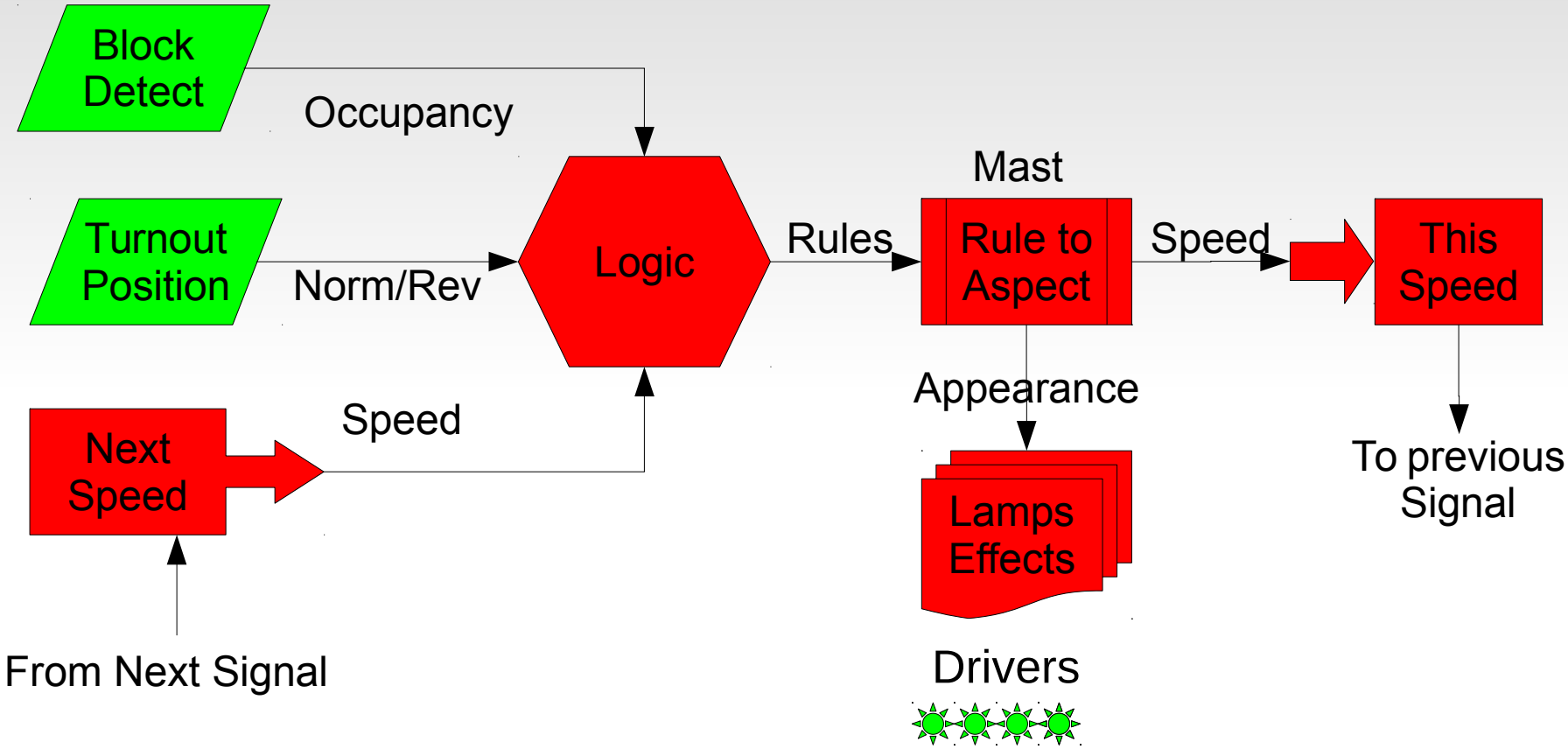
The basic signal logic overview.

- 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, 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.

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.

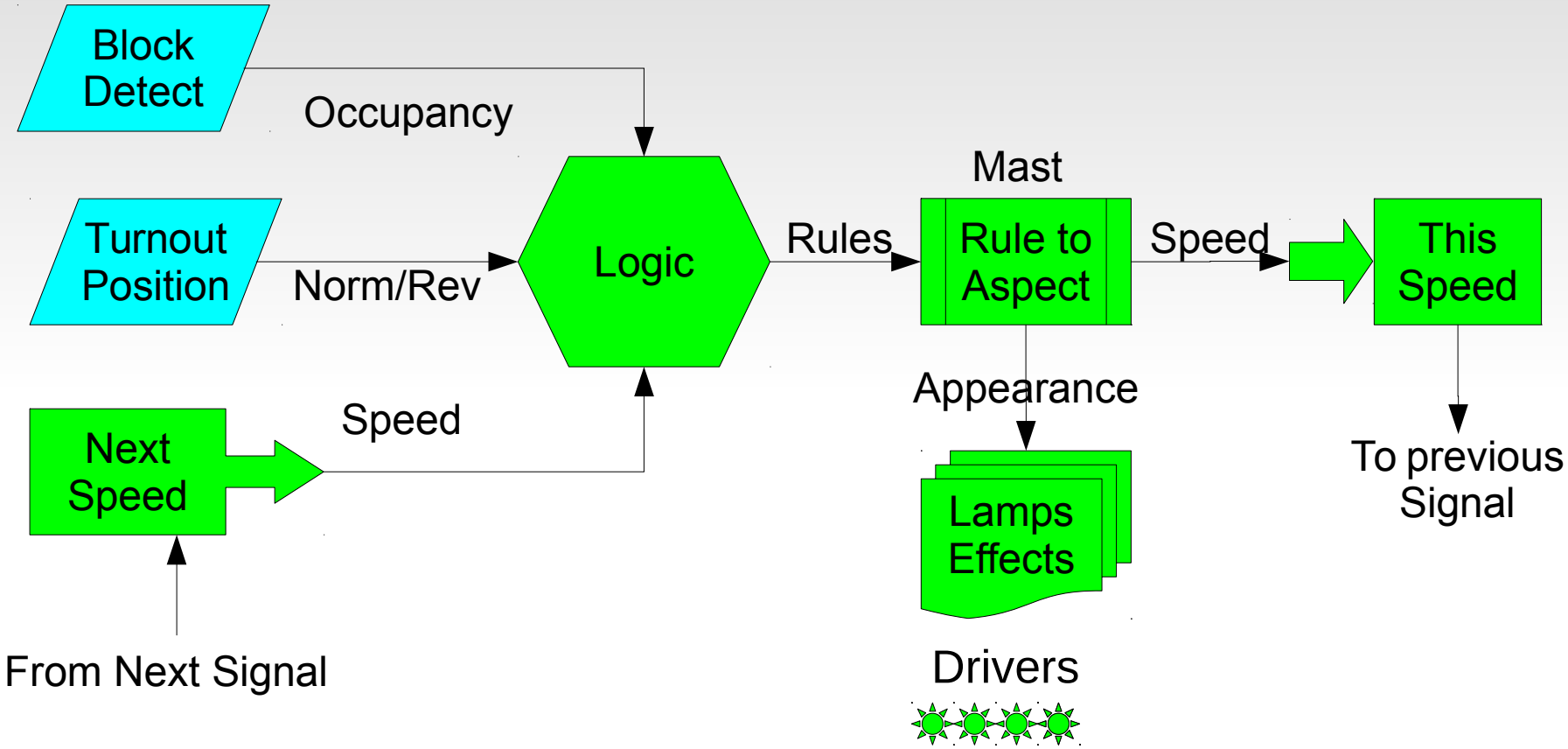
- Effects

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 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 a daughter card or different nodes.

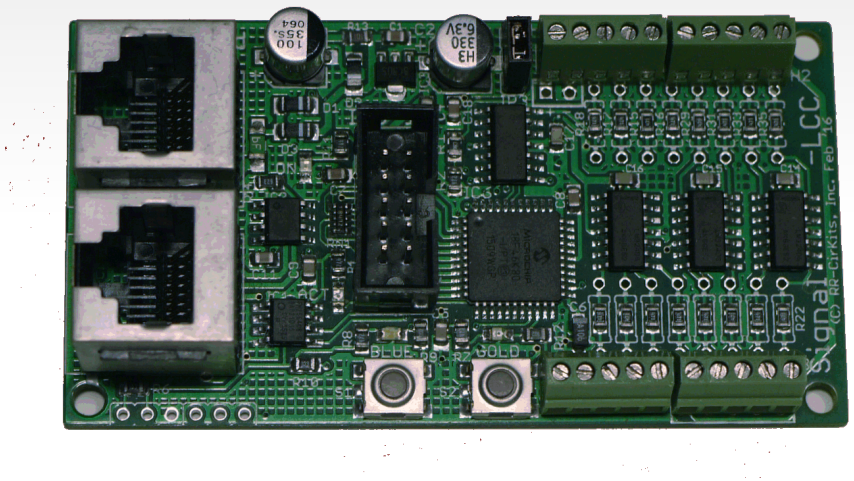
If you want to off load (or monitor) any function with a computer you may do so by intercepting the LCC EventIDs that link sections with each other.

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.

Latest LCC Hardware

- Last year we designed a Signal Driver.



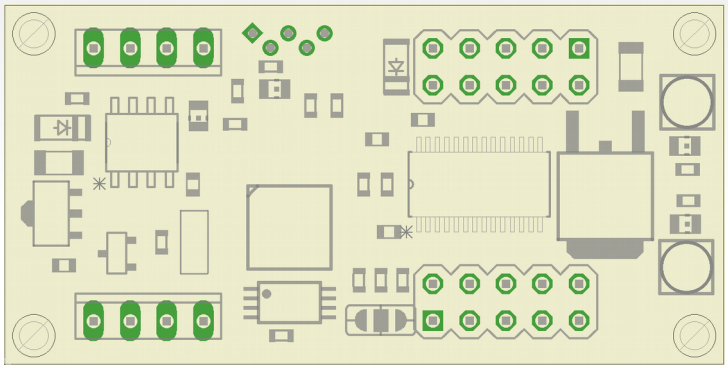
Signal-LCC
- True aspect-based signaling
- Easy to configure logic
- Max. 8 signal masts, 16 LEDs
- Up to 32 aspects

Plus 8x I/O lines (like the Tower-LCC)

Latest LCC Hardware

- This year's newest node is also a Signal Driver.

- Signal-Mini
  - True aspect-based signaling
  - Easy to configure logic
  - Max. 16 signal masts, 16 LEDs
  - Up to 32 aspects



- Lower cost and smaller board. (25mm x 50mm)
  - Uses just two pairs of the standard CAT-5 LCC cable
  - Miniature Tyco MTA-100 IDC connectors for CAN bus
  - Common Anode only (no Atlas, nor Tomar searchlight)
  - Constant current LED drive, no resistors required

LCC Configuration Tools

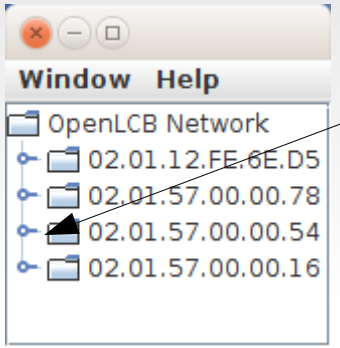
- This node information is stored in the node as a CDI file. (Configuration Description Information) The CDI is in .xml format, but because it references internal register locations it is not advisable to attempt making any changes manually.

Example CDI info as stored in a node:

```
<?xml version='1.0'?>
<cdi xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:noNamespaceSchemaLocation=
        'http://openlcb.org/schema/cdi/1/1/cdi.xsd'>
<identification>
<manufacturer>RR-CirKits</manufacturer>
<model>Signal-LCC</model>
<hardwareVersion>rev-A</hardwareVersion>
<softwareVersion>A-4</softwareVersion>
</identification>
<segment space='253'>
```

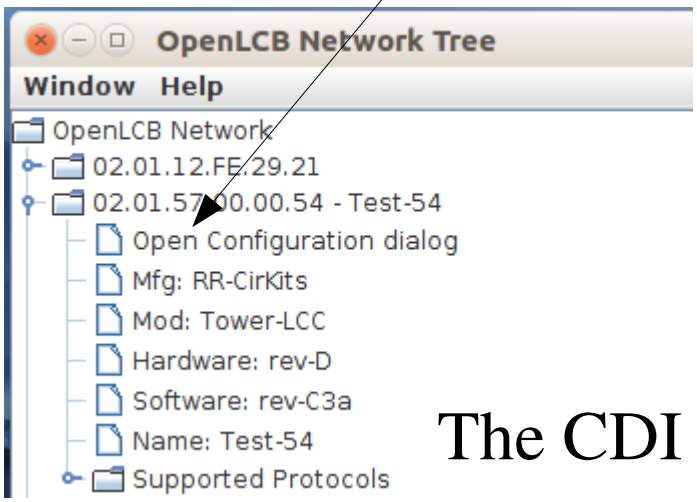
- The original CDI configuration tool was created as a part of JMRI. www.jmri.org

Select OpenLCB and choose ’Configure Nodes’



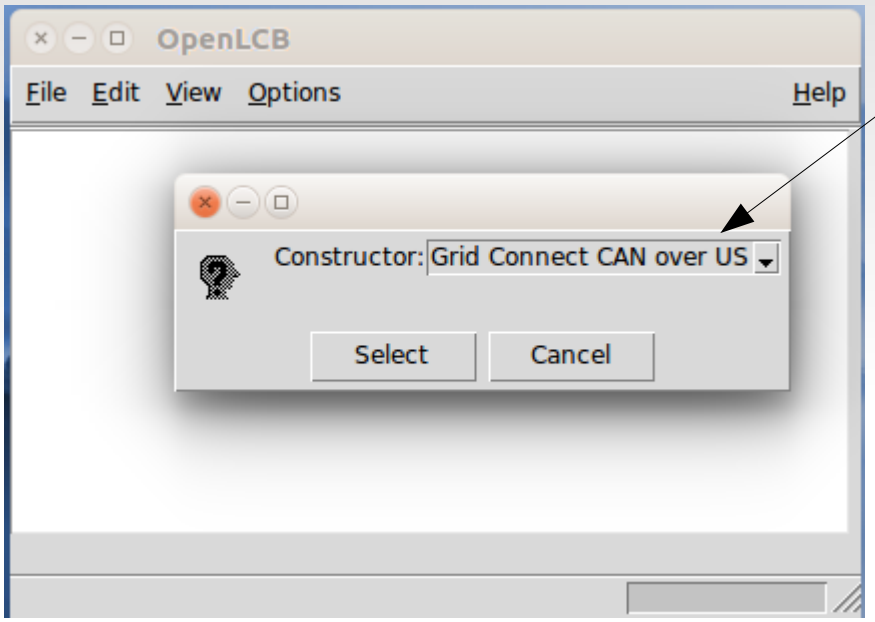
Next open the node you need to configure.

Open ’Configuration dialog’.

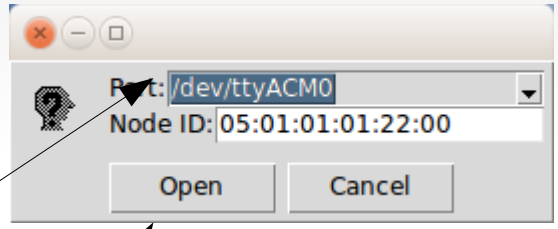


The CDI window will then open.

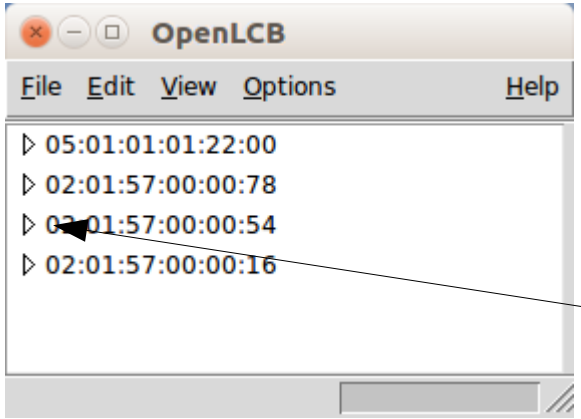
Because LCC is an open standard anyone can develop tools for it. One such developer is Robert Heller of Deepwoods Software. This is part of his model railroad software package.
http://www.deepsoft.com/home/products/modelrailroadsystem/downloadmr/
Run the OpenLCB tool.



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.

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 2018 we had over 250 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

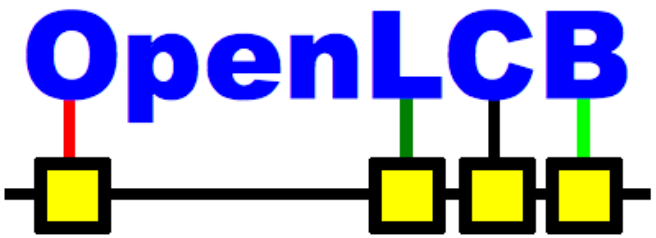
Useful Links:

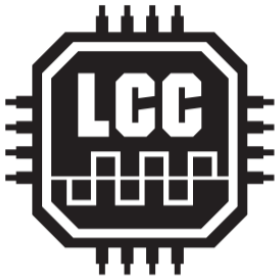
http://openlcb.org or http://openlcb.com

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

Not so Useful Link:

http://www.layoutcommandcontrol.com/





Questions

- ?