

NMRA MER 2017
The Susquehannock
Signaling with LCC
(Layout Command & Control)

Compiled by: Dick Bronson
RR-CirKits, Inc.

Signaling with LCC.

[www.rr-cirkits.com/clinics/MER-2017-Signaling with LCC-A.pdf](http://www.rr-cirkits.com/clinics/MER-2017-Signaling%20with%20LCC-A.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 it said that LCC is a solution looking for a problem, because we already have 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.

Other Options - CMRI

- CMRI was actually the first system to allow for two way communication with the layout. In my opinion its primary drawback is its Master/Slave nature.
- A CMRI system always requires a single master computer to control everything. Until recently it was also proprietary.
- No CMRI node can communicate directly to another CMRI node.

Other Options - LocoNet

- The LocoNet was the first Peer-Peer model railroad network. That means that any device may talk to any other device without any master unit being in charge. (except during programming)
- Unfortunately the LocoNet is proprietary and requires licensing for commercial use.
- The LocoNet operates only slightly faster than DCC itself.
- The LocoNet does not overload gracefully. 

Other Options - Etc.

- The XpressNet is a Master/Slave network with the same limitations as CMRI, and has little US presence or support.
- The UK based MERG group have many excellent designs, but they require an annual membership fee to access many of them. Their addressing is not globally unique, so you must still keep track of node address usage.

A solution is proposed

- The NMRA decided a decade ago to call for the creation of 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 into the next generation of electronic products.
- It should be a peer-peer design with no requirements for any central control.
- 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 from the Open LCB group that can be sent over any media. For example, EtherNet, Wi-Fi, CAN (Control Area Network), and others. (some say tin cans and string, but don't believe it)
- The NMRA calls this Open LCB protocol ***LCC***. ***Layout Command and Control***. LCC is NOT a replacement for DCC. (unless you consider it replacing DCC accessory decoders)
- LCC can be 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 whole layout.

Why CAN?

- It is important to remember that LCC can be transported over many network technologies.
- When we (RR-CirKits, Inc.) decided to build LCC devices we had to make a choice of which transport to use. Wired Ethernet was one option, but designing a peer-peer network for Ethernet was way above our pay grade. The other problem is that it would require multi port Ethernet Switches with direct cable connections to each device. This would require more wiring, not less, than current options. Wireless sounded nice, but it has issues with many nodes, and putting a radio in every node seems like a complex and costly solution as well.

Why CAN?

- CAN was initially developed as a solution for automotive networking. This means that it is noise tolerant, an industry standard, and designed for the 12-24V world. CAN can be operated over a wide speed range, with a linear trade off between bus speed and bus length.
- The OpenLCB engineers picked a 125Kb rate and 1000' length as a good compromise for model railroad use. This is still an order of magnitude faster than DCC.
- Also, unlike the other popular Peer-Peer system, CAN can operate continuously at 100% data throughput with error free collision resolution.

CAN Disadvantages?

- The relatively high CAN bus speed does not allow for free form network designs. A CAN network segment requires a linear bus with a termination at each end to operate properly.
- Due to timing and other electrical limitations a single CAN segment is limited to 40 or fewer physical nodes. There are fairly simple ways to expand a CAN network into multiple segments, so this is not a serious concern.

Why CAN?

- CAN has several different cabling and connector standards. Some of these use large and costly connectors. Often CAN uses the same DB-9 connectors used by RS-232 serial cables. These are still relatively large and no longer very easy to locate, especially in longer lengths.
- Another CAN connector option uses the same RJ45 connectors and cables as wired Ethernet does. The OpenLCB engineers opted for RJ45 connectors because of the relatively low cost and their common availability world wide. The 4 pairs of a standard Ethernet cable additionally allow for optional power and DCC booster drive signals in addition to the CAN data pair itself.

LCC Basic Concepts

- How many here understood Brian Pickering's August 2017 NMRA Magazine technical article on Events in LCC?
- I will repeat his 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 enter one manually. There is little if any reason that you would ever need to know the details of what they mean. (which isn't a whole lot anyway)

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 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 Basic Concepts

- Maybe you want all turnouts to move to normal when you first start up. With our conventional control bus you need some way to send the proper commands to each turnout. (sometimes called Routes) With the LCC system you could simply define a new Event that says 'set *all turnouts normal*' and then configure each turnout to also respond to that command (by moving in the appropriate direction)

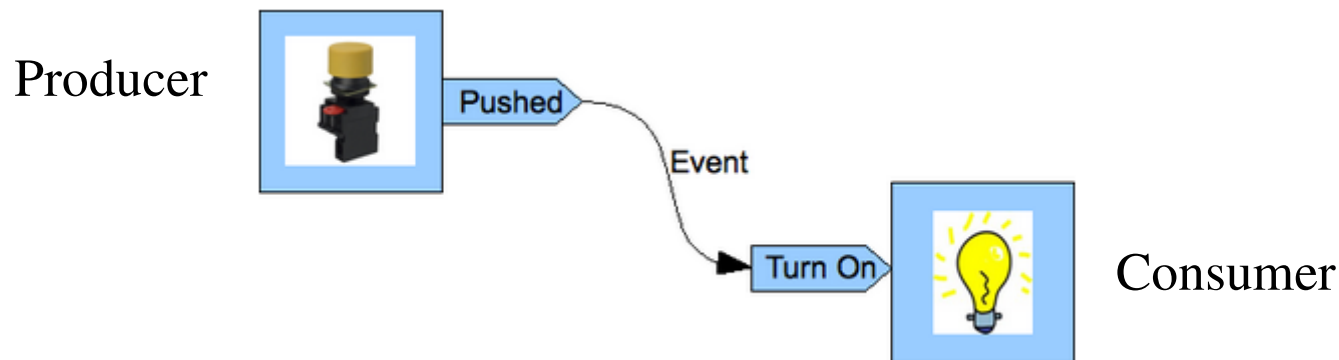
LCC Basic Concepts

- **Producer - Consumer** You will probably hear LCC folks throwing around terms like Producer and Consumer. They aren't talking about a big business takeover. The Producer/Consumer concept is an industry standard terminology to separate requests from execution.
 - *Producer* simply means that some device can create (produce) an Event. Some examples might be a push button or block detector.
 - *Consumer* simply 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*.

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

LCC Basic Concepts

To elaborate a little bit. For an event to happen something must have requested it. Therefore there has to be at least one *producer*. In the LCC world it is possible for many different *Producers* to create the same event or request. 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 Basic Concepts

For *consumers* the picture is a bit different. There is nothing in the specification that says any device has to respond to an Event. You may have built a panel for a passing siding that doesn't yet exist. If you press its turnout control button an Event or request message gets sent out. (*producer*) However there is nothing to respond. (*consumer*) Later you might add a turnout controller and a computer based CTC machine and have several *consumers* that can respond to that Event. Thus the statement that every Event has zero or more *consumers*.

LCC driving Signals

- Application to Signals

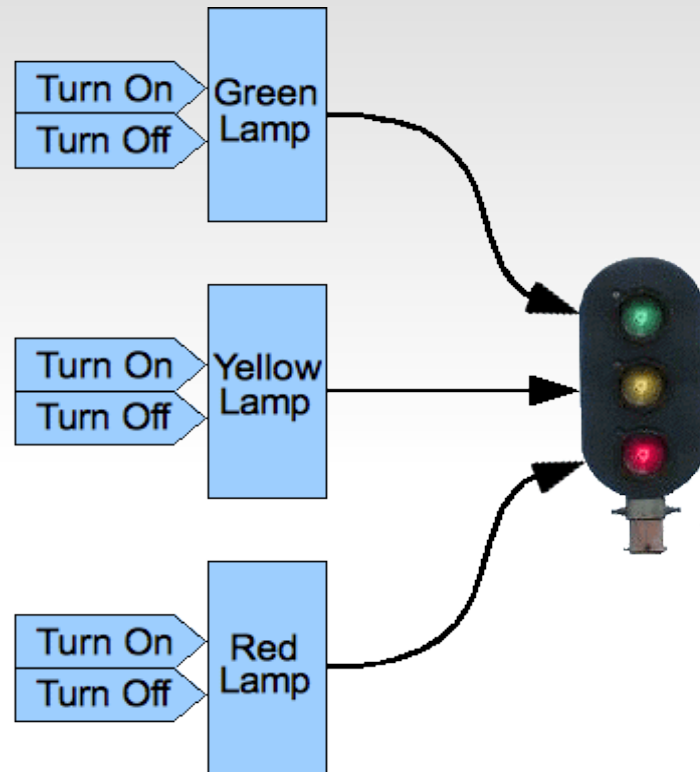
Signaling usually requires more logic than can be handled via simple Events, e.g. occupancy, turnout position, look ahead to the next signals, etc. However a signal controller could be designed to listen to all of the appropriate Events and fully control the signal aspects. Note that it's still useful 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

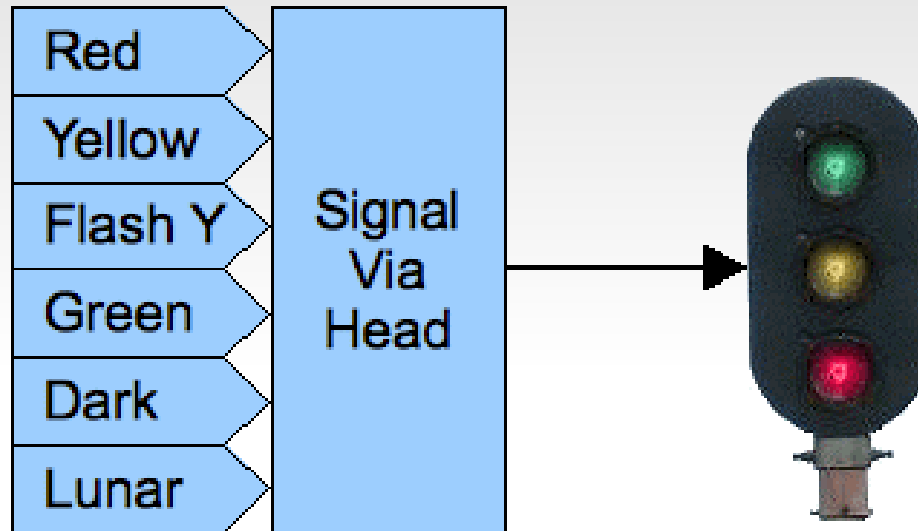
You can connect the lamps of a signal head to individual Consumers:



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 latency causes poor timing of flashing signals. This is the method used by CMRI.

- Signals via individual head drivers

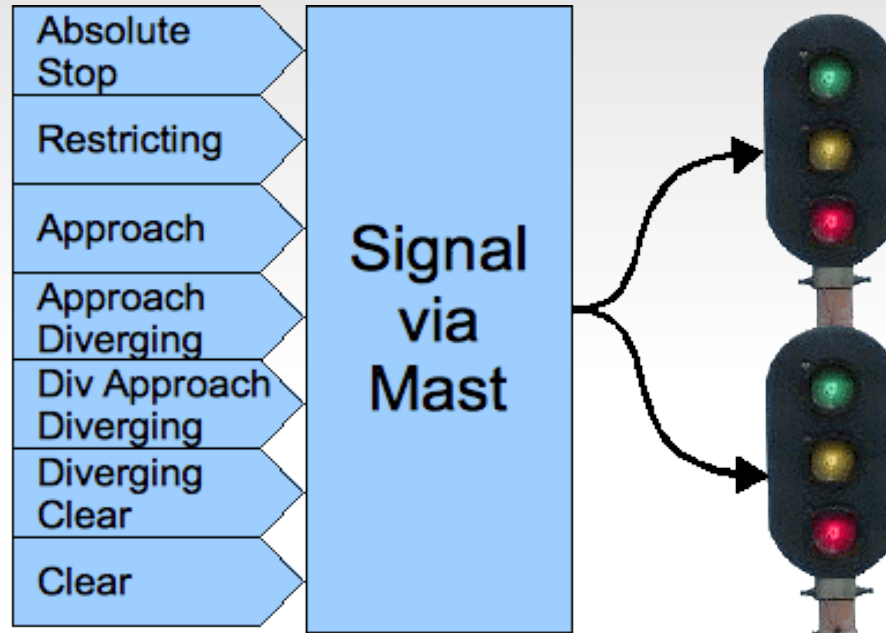
You can also control signals with Events for the specific colors or functions of a single head.



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

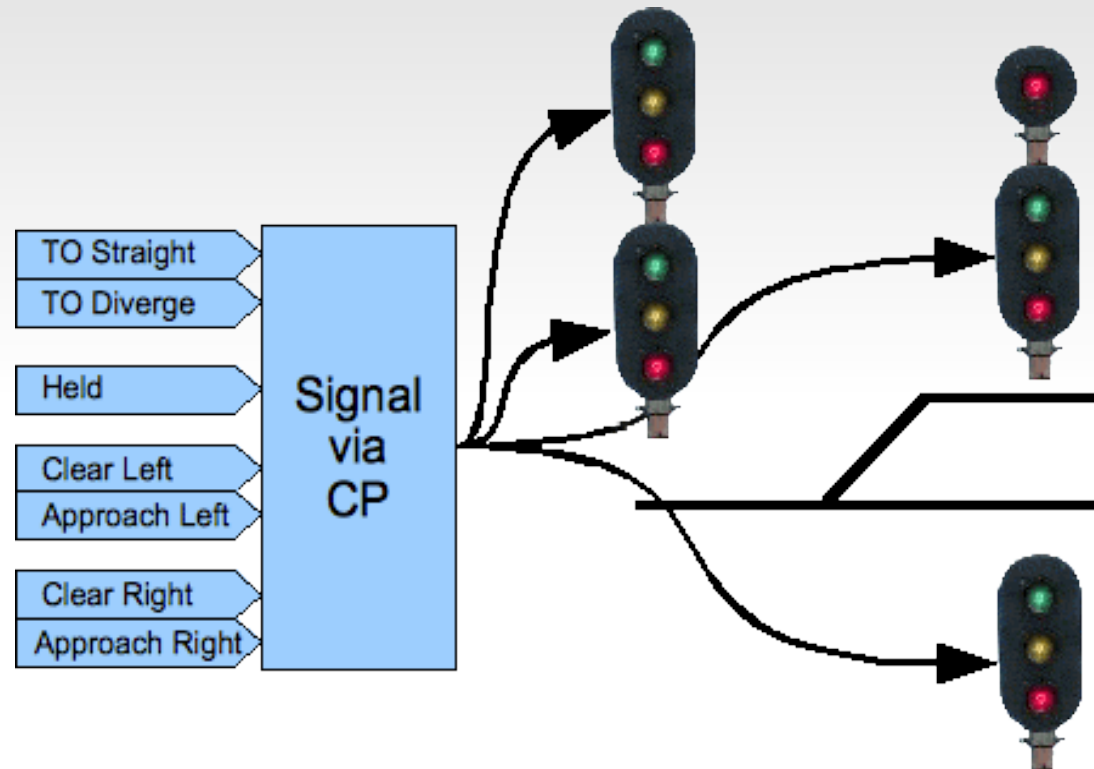
You can control an entire signal mast with just one Event for each high-level aspect of the signaling system.



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

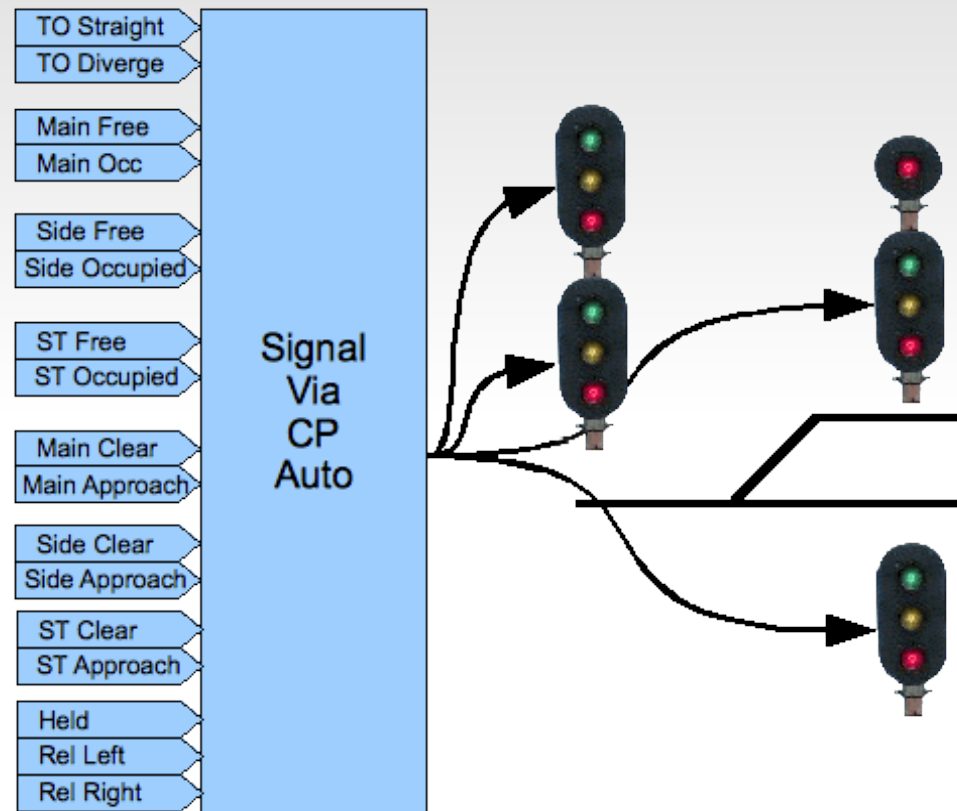
You could also control an entire interlocking with just single Events for each high-level aspect of the signaling system including turnout position.



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 old 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.

■ LCC Background

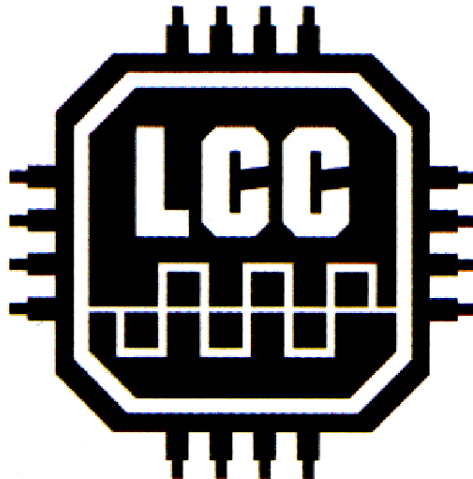
At the Detroit NMRA National back in 2007 the NMRA was seeking a network standard to be known as NMRAnet for layout control. They proposed that the Manufacturers Working Group create a standard in a 6 month time frame. (if my memory serves me correctly)

LCC grew out of a concept first presented by John Socha-Leialoha during a lunch meeting in the food court following that meeting. John proposed that the PC (Producer Consumer) model be used by this new standard, and proceeded to try to explain to some of us gathered around the table just what he meant by that. Time and politics passed, and the NMRA tentatively accepted one early proposal. However, other folks didn't agree and formed an independent project known as...



More time and politics passed, and the NMRA finally decided to get out of the specification writing morass, and turned that job over to the original OpenLCB group.

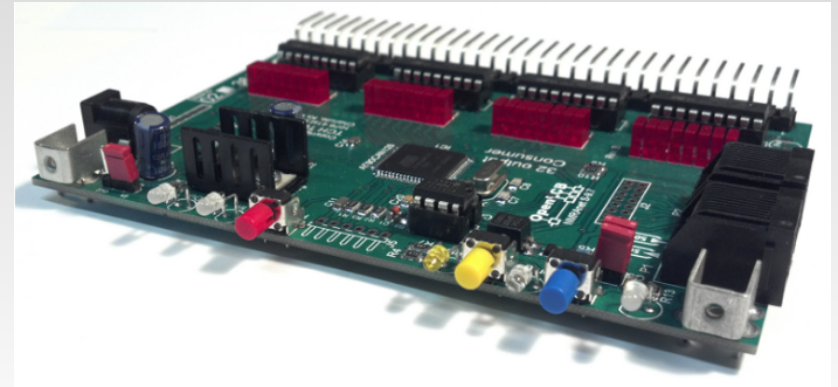
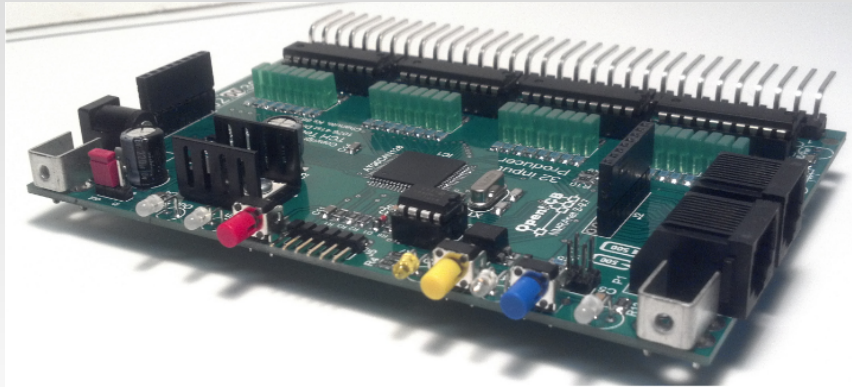
- Fast forward to just prior to the Cleveland NMRA National in 2014. The NMRA went back to the OpenLCB group and gave them an ultimatum. Present a proposal to the NMRA or they would declare the project as dead. This created a new sense of urgency and the basic specifications were presented to the NMRA in time for the early 2015 board meetings. Unfortunately in the rush to publish something, some key features were omitted, so it was not until late 2015 before we (RR-CirKits, Inc.) felt that the specifications were mature enough to actually start delivering hardware. Specifically we wanted our users to have an approved method for upgrading their products.
- With the NMRA approval came their new branded version of the OpenLCB specifications. They call it LCC. (Layout Command and Control)



Early Hardware

- One of the first manufacturers of CAN based layout control nodes was the MERG group. They proposed that the NMRA accept their protocol. In fact some of the early development work was done using their hardware.
- Another early proposal came from Don Voss. (brother of Di Voss) It was Don's proposal that was originally entertained by the NMRA as the NMRAnet.
- As I previously mentioned, the OpenLCB group felt that these CAN only protocols were too restrictive to be chosen as the next generation standard, so they pushed forward with their own ideas and protocol proposals.
- Fortunately there were some in the NMRA that were taking notice.

- One of the first manufacturers of OpenLCB nodes was Tim Hatch of TCH Technologies. A couple of Tims products are shown here.

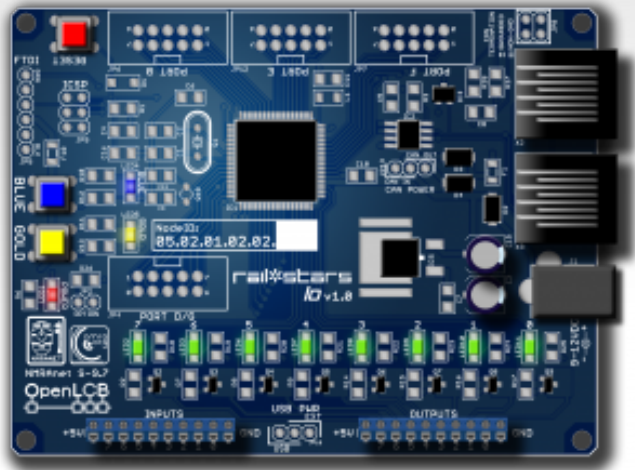


These early boards were essentially OpenLCB replacements for the 32 line Bruce Chubb input and output boards. They were developed as a way for the developers to run real hardware to prove out the specifications. Unfortunately they have the same limitations as their CMRI equivalents in that they are strictly input or output slaves to a computer program. These boards are no longer available nor supported by recent JMRI versions.

TCH also manufactured the first CAN bus to USB interface available for the OpenLCB.

- Another early hardware developer was Don Goodman of Railstars. His OpenLCB board includes both inputs and outputs. It is called Io. (named for the Jovian moon)

<http://railstars.com/hardware/io/io/>



As far as I know the Railstars Io board is no longer available. However, like the TCH boards, the Io supports just two producers or consumers per line, so it is essentially an I/O board tied to a computer program.

Currently Available LCC Hardware

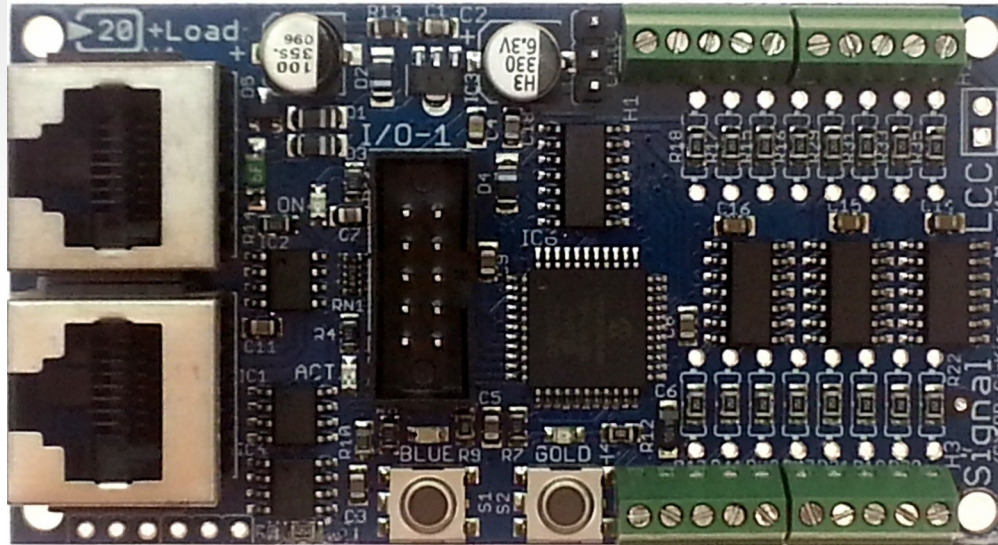
- We at RR-CirKits watched all the above history unfold, and when we figured that the smoke was mostly cleared, we started the design cycle on a family of CAN based OpenLCB boards. (now NMRA branded as LCC) This initial development process was delayed due to a missing firmware upload protocol, but we finally started shipping hardware to our first customers in January 2016. (a year after the NMRA accepted the current protocol)

Currently Available LCC Hardware

- Our (RR-CirKits, Inc.) current product line includes the basic items required to start investigating the new LCC bus. We chose our first I/O node to be compatible with our existing product line of daughter cards. This allows the user to do basic train detection, turnout control, and similar functions using available hardware options. Just released is also a signal mast driver.

Latest LCC Hardware

- The newest node we have designed is 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)

Currently Available LCC Hardware

- Power – The LCC CAN bus has two basic options for power supply to the nodes. The first is to supply power to each node. The second is to power the nodes from the CAN bus cable. Of course a node could also do both. (some early hardware did that) Because one of the desirable features of the LCC is to eliminate as much layout wiring as is practical, we chose the second option. We suggest that the user supply power to the bus as required by using our Power-Point module and/or our LCC Repeater module.

Currently Available LCC Hardware

- Termination – The LCC CAN bus is much faster than any existing layout control buses, therefore it requires termination at both ends for proper response. Again, the standard allows for this termination to be part of each board, using jumpers or switch selection, or to be separately provided.
- We chose the latter option because we feel that it is less likely to be configured incorrectly. The only places that the terminators may be easily connected are at each end of a bus segment, just exactly where they are required, and no place else.

Currently Available LCC Hardware

- Cables – The CAN version of LCC was specified to use the commonly available CAT5, CAT5E, and CAT6 cables with RJ45 connectors. The industry standard basic pin out for CAN over CAT5 was chosen. This was done so that the user could easily purchase or construct his own cables. The 4 and 6 conductor silver satin cables used by some other manufacturers are no longer as easy to find as they were 20 years ago. The two different systems (CAN and Ethernet) that use these cables supposedly will not suffer damage if the cables are cross connected accidentally between networks. Of course neither network will work in that case.

Currently Available LCC Hardware

- Cables – The CAN version of LCC was specified to use the commonly available CAT5, CAT5E, and CAT6 cables with RJ45 connectors. The industry standard pin out for CAN over CAT5 was chosen. This was done so that the user could easily purchase or construct his own cables. The 4 and 6 conductor silver satin cables used by some other manufacturers are no longer as easy to find as they were 20 years ago. The two different systems (CAN and Ethernet) that use these cables supposedly will not suffer damage if the cables are cross connected accidentally between networks. Of course neither network will work in that case.

Currently Available LCC Hardware

- With the recent addition of an option to place the DCC rail sync information on an otherwise unused pair, LCC over 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 many different systems, Ethernet, and Wi-Fi are also being developed for use by other LCC developers/manufacturers.

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 was 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

- When devices were first connected to a bus, (or track) we required assigning addresses or channels. The original 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 DCC decoders now each 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 along with their addresses.

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 using a special connection. (called a programming track)

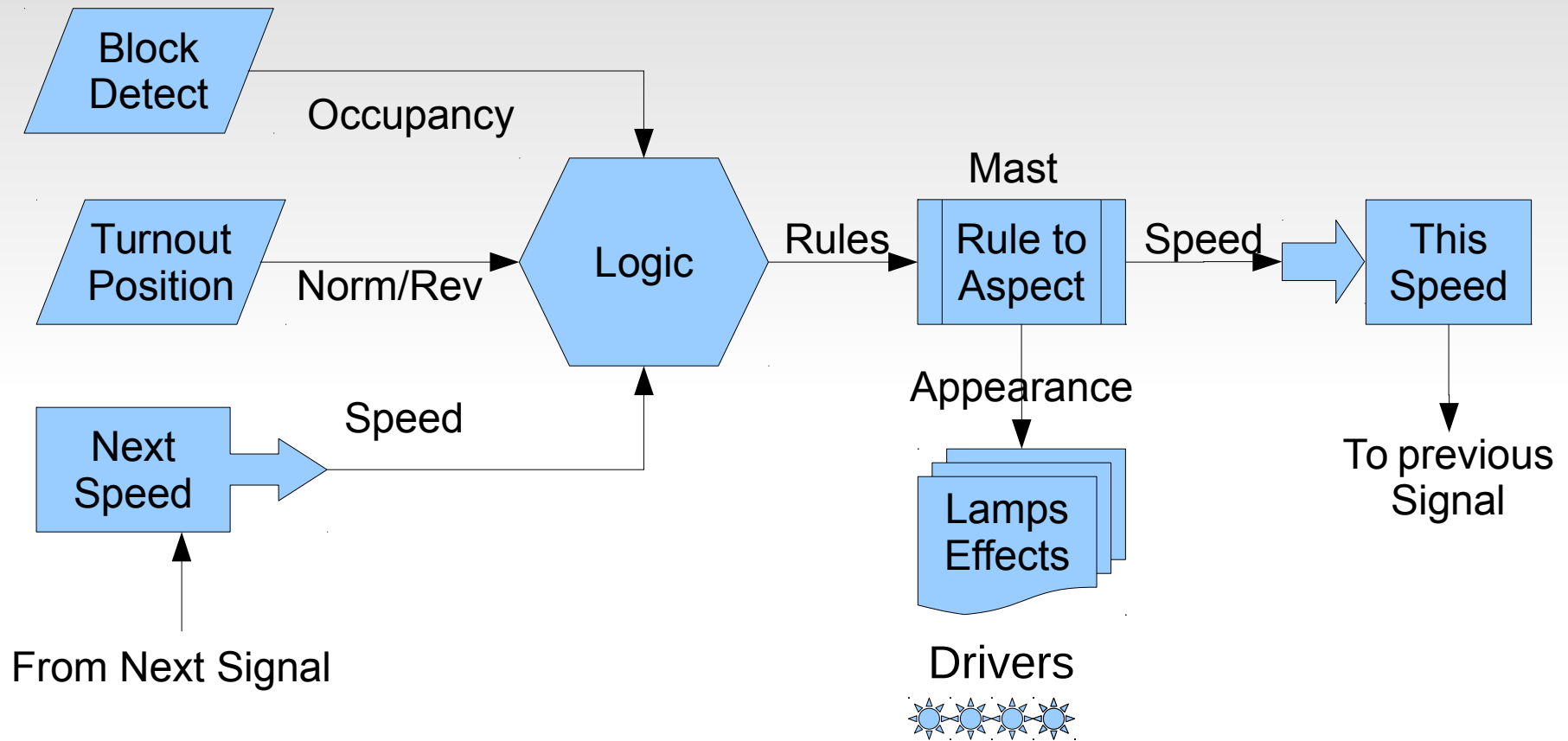
Configuration of LCC nodes

- 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, or to use a special programming circuit.
- 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 any 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.

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 vital logic for the signal.

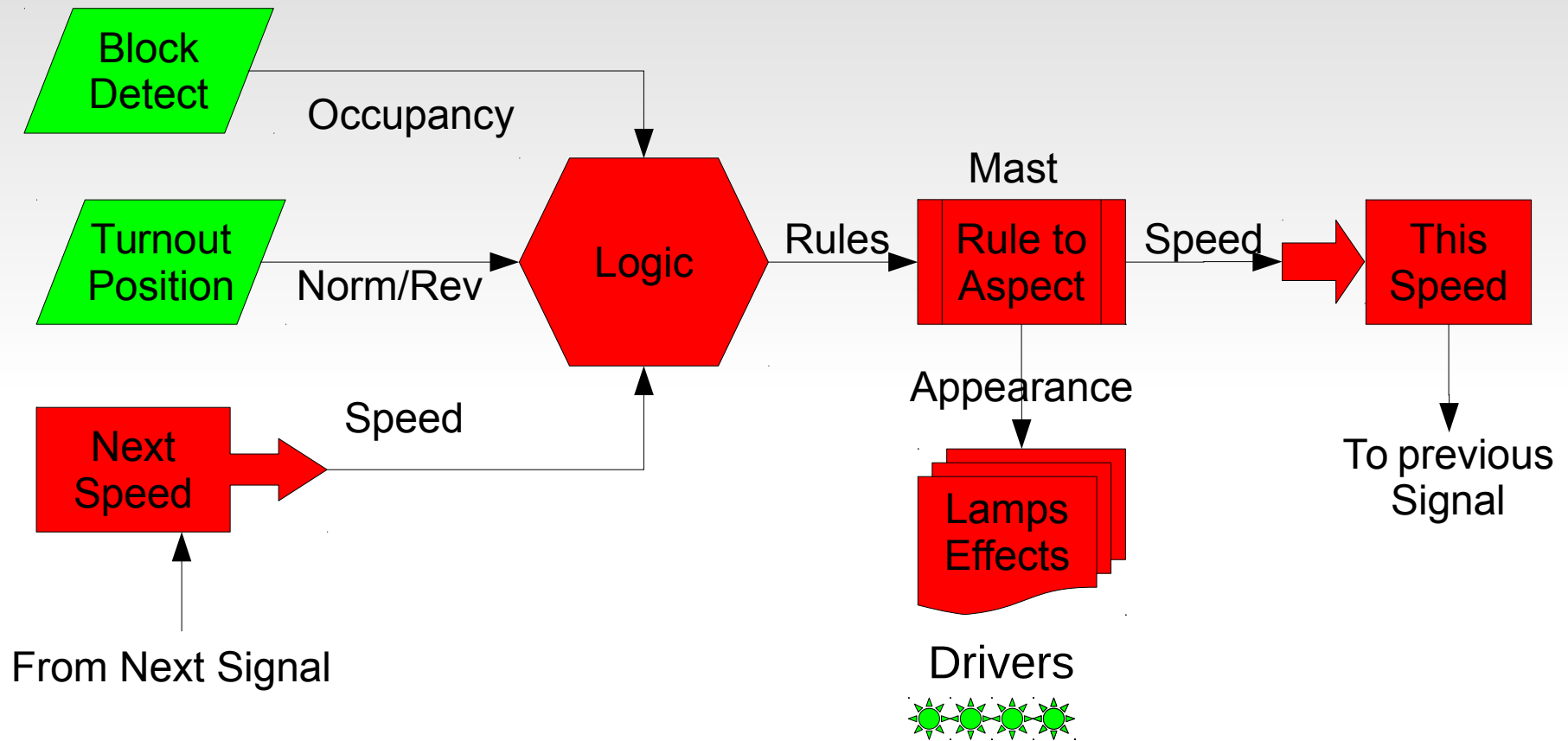
- **Rule to Aspect conversion**

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.

- **Signal Drivers**

Signal LEDs may be driven from 5V logic levels, but there are good reasons to use higher voltages like 12V for the primary drive. Signals are sometimes wired common anode, and sometimes wired common cathode. Drivers may also be responsible for special effects such as lamp fading. (both up and down)

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.

Features 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 the previous over track circuits.

- **Effects**

Prototype signal lamps do not always 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 lamps and colors.

Signaling Continued

- To be continued with:

MER-2017-Signaling with LCC-B