

**NMRA 2017**  
**Orange Blossom Special**  
**Signaling with LCC**  
**(Layout Command & Control)**

Compiled by: Dick Bronson  
RR-CirKits, Inc.

**Signaling with LCC.**

[www.rr-cirkits.com/clinics/NMRA-2017-Signaling with LCC.pdf](http://www.rr-cirkits.com/clinics/NMRA-2017-Signaling%20with%20LCC.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

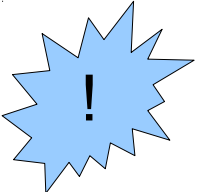
- What about LocoNet, CMRI, XpressNet, MERG, plus the many other proprietary methods to connect our devices.
- Most of these solutions originated due to the difficulty in using the DCC bus for any input information.



# 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.

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

# LCC Basic Concepts

- How many here read Brian Pickering's August 2017 NMRA Magazine article on Events in LCC?
- How many understood it?
- 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*)

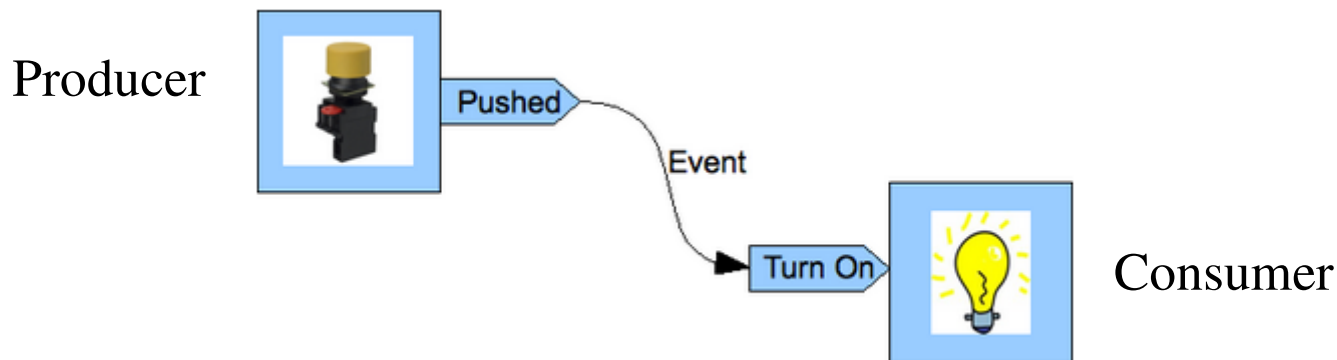
# LCC Basic Concepts

- **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>
  - *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*.

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

# LCC Basic Concepts

To elaborate a little bit. For an event to happen something must have sent 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. 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 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* for that Event. Thus the statement that every Event has zero or more *consumers*.

# LCC driving Signals

- Application to Signals

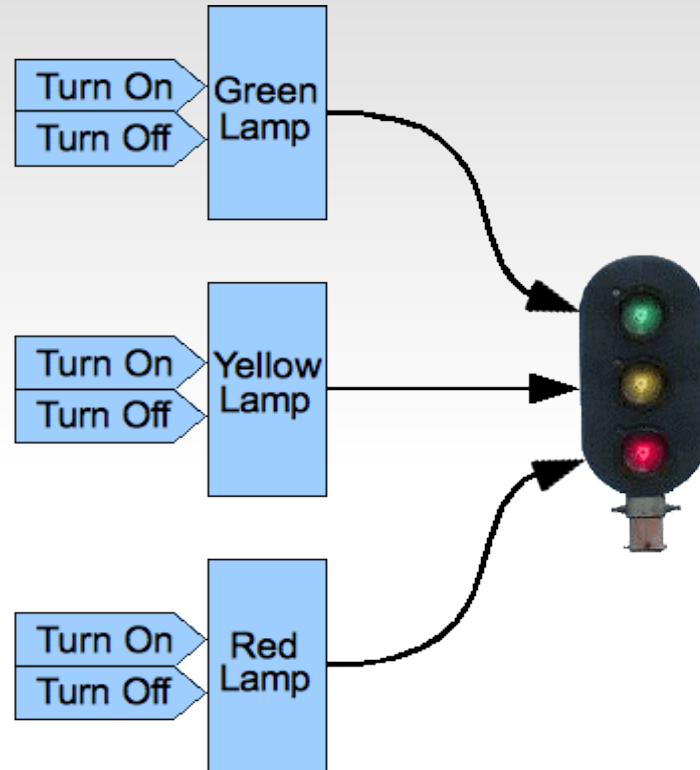
Signaling usually requires more logic than can be handled via simple Events, e.g. occupancy, turnout position, look ahead to the next signals, etc. However a signal controller could be designed to listen to all of the appropriate Events and fully control the signal aspects. Note that it's still useful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

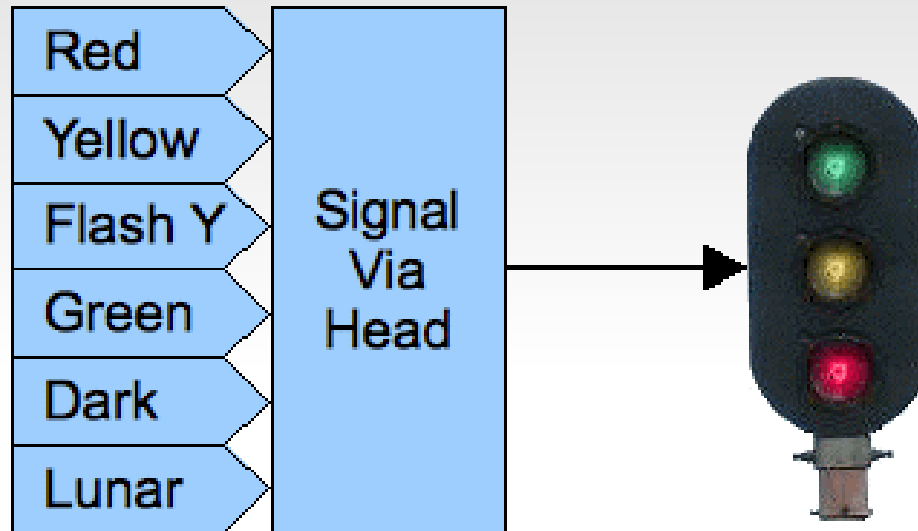
You can connect the lamps of a signal head to individual Consumers:



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.

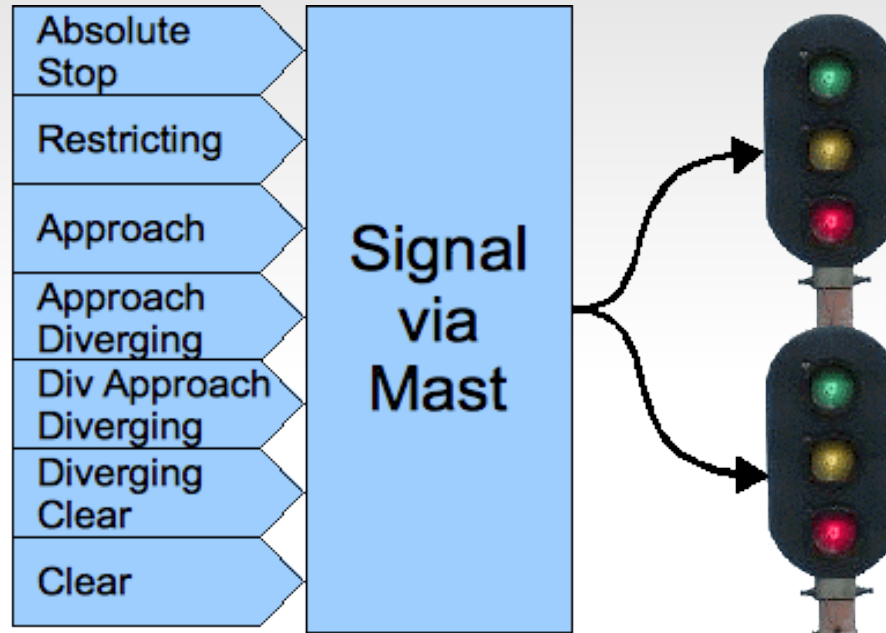


This method requires less command traffic than the previous one. However, if the controller does not know how to flash the signals, it may still result in constant streams of messages to be able to show flashing aspects. The Digitrax SE8c falls into this category. It normally only displays Green, Yellow,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



## ■ Signals via aspect drivers

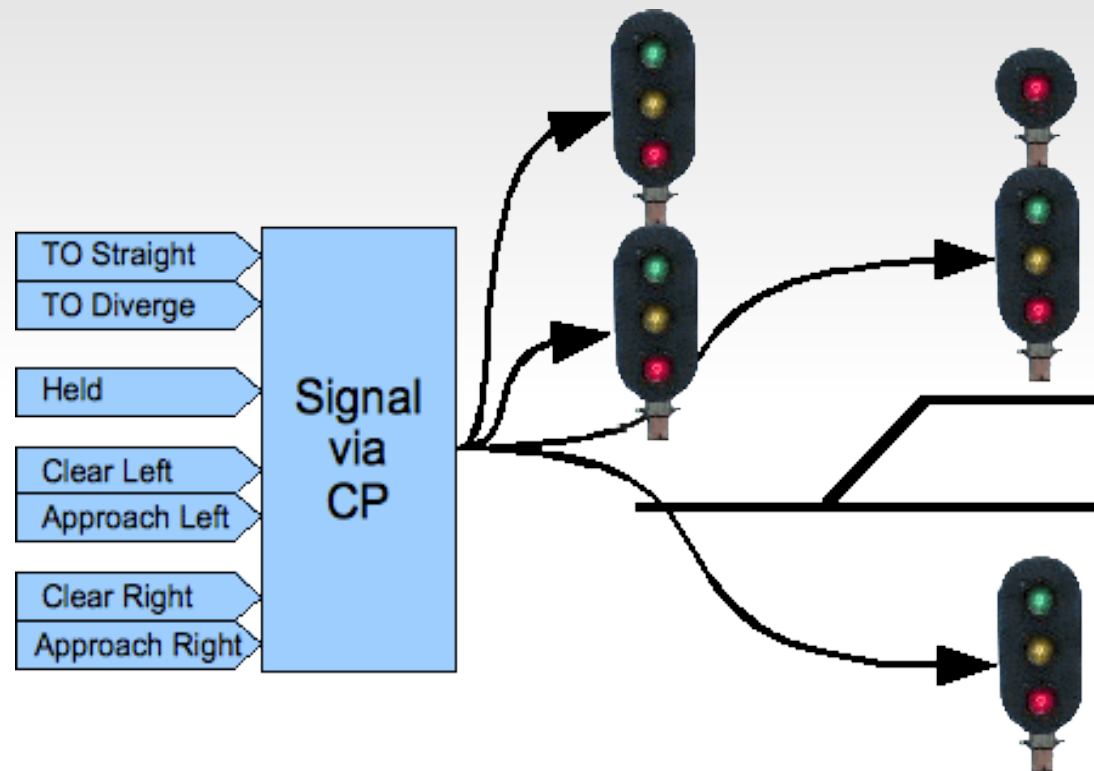
You can control an entire signal mast with just one Event for each high-level aspect of the signaling system.



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

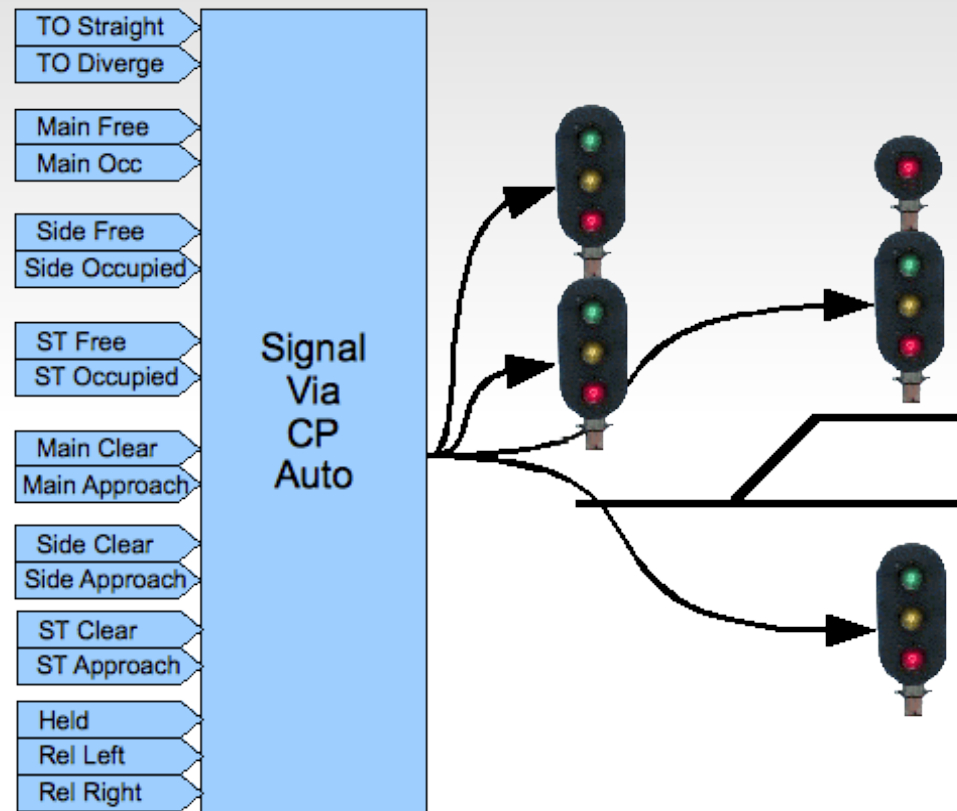
You could also control an entire interlocking with just single Events for each high-level aspect of the signaling system including turnout position.



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.

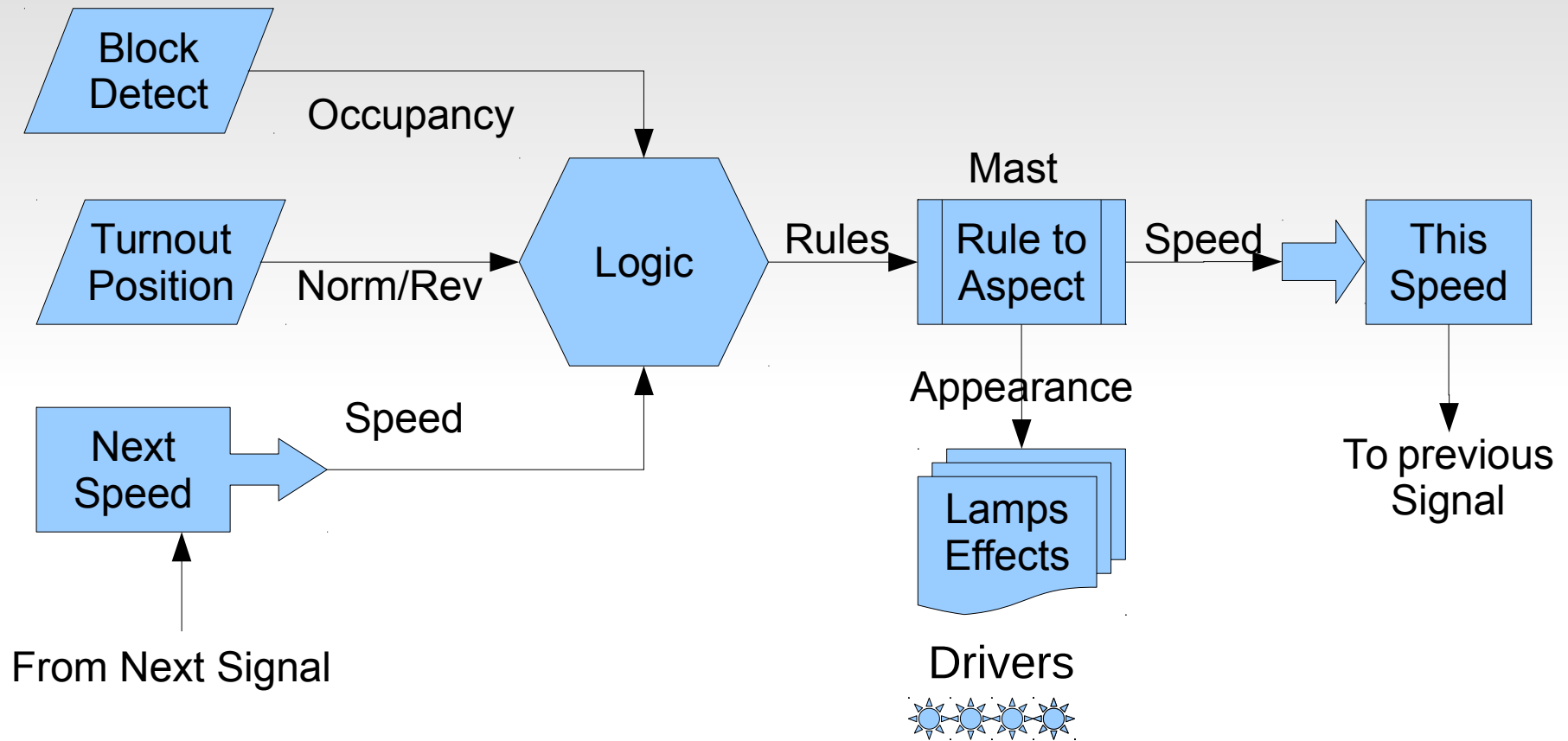


It's also possible to build a signal controller that watches all related status Events from the railroad and CTC panel and makes independent decisions about the proper signal states and appearances. This type of controller would be able to control its signals without any external computer involvement.

# 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 logic for the signal.

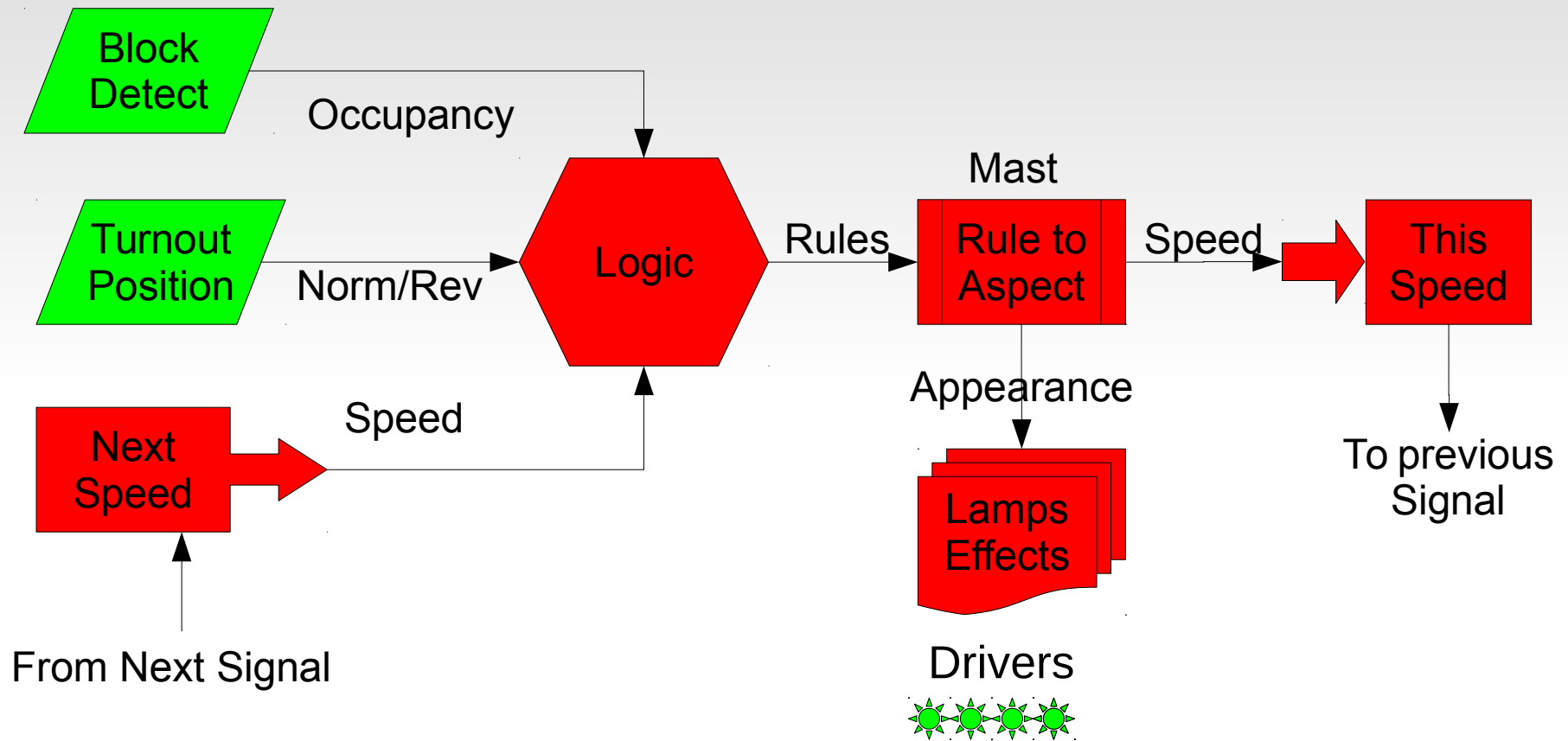
- **Rule to Aspect conversion**

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.

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 do not always simply blink on or off as they change. Effects simulating 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.