INTRODUCTION TO MODEL RAILROAD LAYOUT AUTOMATION

LCC 101

Layout Command Control for model railroad electronics

July 30, 2026 11:00 AM

By Dick Bronson

OpenOffice Impress deck

What this covers

- What LCC is
- A little history
- How nodes and events work
- How LCC works
- Specific LCC advantages
- CDI and configuration tools
- Software aids
- How LCC differs from DC/DCC
- Train Control Protocol
- Layout sizes and fit
- Example manufacturers
- Command and control principles
- More example scenarios
- How to plan, test, and avoid mistakes

LCC separates layout automation from train power

It gives layout devices a common way to report, request, and coordinate actions.

A layout control standard

LCC is the NMRA Layout Command Control standard for connecting accessories, detectors, panels, software, and controllers on a model Railroad layout.

A message network

Devices exchange events and data over a layout network or WiFi instead of relying only on direct point-to-point wiring.

A partner to DCC

DCC normally drives locomotives over the rails. LCC handles the broader layout: detection, turnouts, signals, sensors, lighting, panels, and automation.

Typical separation

DCC / train power
- Speed
- Direction
- Sound
- Functions

LCC / layout control
- Turnouts
- Signals
- Sensors
- Panels
- Automation
- Lighting

LCC grew out of a need for shared layout control

The standard reflects an open, collaborative path rather than a single-vendor specific bus.

1

More layout electronics

Model railroaders wanted better control of accessories, staging yards, lighting, and dispatcher panels without being tied to a single manufacturer’s offerings.

2

Early LCB work

The NMRA explored a new Layout Control Bus.

Givens:

Use Industry Standards, not proprietary designs. Use peer-peer networking, not central control. No patent limits for MRR use.

3

OpenLCB proposal

The NMRA chose to proceed with the OpenLCB Group’s proposals and collaborate with them on protocol documents and standards.

4

NMRA adoption

Adopted LCC document sets include: 2021-07-02 and 2025-01-24 standards releases.

The practical takeaway: LCC is designed for interoperability, growth, and shared layout control across NMRA standardized electronic hardware and software. As with all NMRA standards, they are available to all on an equal basis.

The core idea is event-driven layout control

Device changes announce what happened; interested devices respond to the change. i.e. a node to node message exchange replaces individual wires on the layout.

Node

A networked device, such as an input board, push button, signal driver, or software interface.

→

Event

Creates a unique message representing a condition, request, or state change on the layout.

→

Configuration

Is the saved setup that tells each node which events it produces, consumes, or exposes.

→

Node

A networked device, such as an output board, turnout driver, signal driver, or software interface.

Example event path

Sensor
Occupancy detected

→

Event
Block A occupied

→

Signal node
Logic shows red

→

Panel
Updates icon

LCC works by matching event producers to event consumers

The network carries shared messages; each configured node decides what those messages mean for its own inputs or outputs, or else ignores them.

Input

A button, detector, clock, or panel icon changes state.

→

Producer node

Announces a unique event that represents the input on the bus.

→

LCC bus

All devices hear the same event at the same time.

→

Consumer node

Only configured consumers respond.

→

Action

Just the appropriate output changes.

Configuration is the key step: With it you teach each node which events it should produce, consume, display, or ignore.

Specific LCC advantages are built into the protocol

Bus speed, startup behavior, identifiers, and event routing are standardized so devices can share layout state predictably.

125,000 baud CAN

The LCC CAN bus runs at 125,000 baud; LCC messages are compact for accessory events and feedback.

Self initialization

Nodes all have unique addresses and identify themselves and negotiate network use at startup, instead of relying on user-defined local addresses.

Global identifiers

Unique node and event IDs eliminate conflicts when equipment, modules, or club sections are combined.

Event fan-out

One producer event can be consumed by many panels, signals, turnouts, or software tools at the same time.

Distributed logic

Control decisions can live in the nodes, so a central computer is helpful, but not required.

Growth paths

CAN repeaters, gateways, and TCP/IP links are designed to allow larger layouts to expand without reassigning every address.

CDI lets a node describe its own setup format

Configuration **D**escription **I**nformation tells node configuration software what settings the node exposes and how to present them.

What CDI means

CDI stands for Configuration Description Information: the description of a node's user-configurable settings.

Where the data comes from

A CDI-aware tool reads the description and stored values from the node itself, so the user does not need to store or hunt for a separate decoder file.

Why it matters

The CDI turns arcane event IDs, ports, names, and options into a practical form a model railroader can edit using his own assigned user names.

What CDI editors show

Identity
- Maker/model
- Version

User Name
and description

Settings
- Ports
- Event IDs
- Options/logic

Refresh or Write
Backup or Restore

CDI utilities help configure, test, and recover nodes

Most beginner work happens in the CDI editor; diagnostic tools help when an event path does not behave, or a more specialized or sophisticated tool is desired.

Configure Nodes

Open a node's CDI, edit setup fields, and write changes back.

Refresh / Write

Reload current values or store any settings that may have changed.

Backup / Restore

Save a working node configuration and reload it after an issue or replacement.

Event Table

See which events are produced or consumed and assign user readable names.

Traffic Monitor / Send Frame

Watch messages during testing or send a controlled test message.

ID Tool / Memory Tool

Advanced checks: flash a physical node or inspect memory when troubleshooting.

Supporting tools can make LCC planning less abstract

These tools are not part of the LCC standard itself; they help plan, document, operate, or inspect an LCC layout.

LccTools

Mac app support for LCC operation, node configuration, events, firmware updates, and traffic monitoring.

LCCPro

JMRI tool to track event IDs, names, producers, consumers, node locations, and test results in one list.

MRRSystem / panels

Other developers may supply software that can provide panels, dispatcher views, database tools, and OpenLCB/LCC support.

Bow-tie event maps

Planning diagrams that visually show one producer event branching to consumers, feedback, and follow-up actions.

Layout diagrams

Mark node locations, cable routes, districts, power feeds, and network boundaries before wiring.

Compatibility checks

Confirm that new 3rd party tools can see the nodes, read their CDI, monitor traffic, and back up the nodes that you use.

LCC complements other control systems

The difference is where messages travel, what they control, and how much layout state is shared.

System	Primary job	Traffic direction	How LCC is different
DC block control	Varies rail voltage and polarity by block.	Controller → block → train	Layout device control is separate from rail power.
DCC	Sends power and digital train commands through the rails.	Command station → rails → decoders	LCC usually handles accessories, sensors, panels, and feedback.
Throttle buses	Connect throttles, panels, and command-station devices.	Throttle ↔ command station	LCC shares layout events across vendors.
Computer I/O	Software drives sensors and outputs. Master-slave design.	Computer → I/O boards	Any LCC node can communicate with any other. LCC can distribute logic in field nodes.
LCC	Coordinates layout devices and feedback using network events.	Producer → network → consumers, with feedback	The shared event traffic is the layout-control layer.

Simple design: use DCC to run trains; use LCC to make the layout around the trains communicate.

Train Control Protocol points LCC toward support of direct locomotive control in the future

Where hardware supports it, the same train-control messages can target LCC command stations or wireless trains.

One throttle control language

Train Control Protocol defines throttle-to-train messages for speed, functions, emergency stop, configuration changes, and feedback.

Command station path

A throttle can send LCC train-control messages to a command station, which acts as a proxy to drive the rails. Typically no feedback from the train.

Direct wireless path

The same protocol can also target a wireless train or loco directly, where the onboard hardware supports it. This mode has two way communication between the throttle and the train.

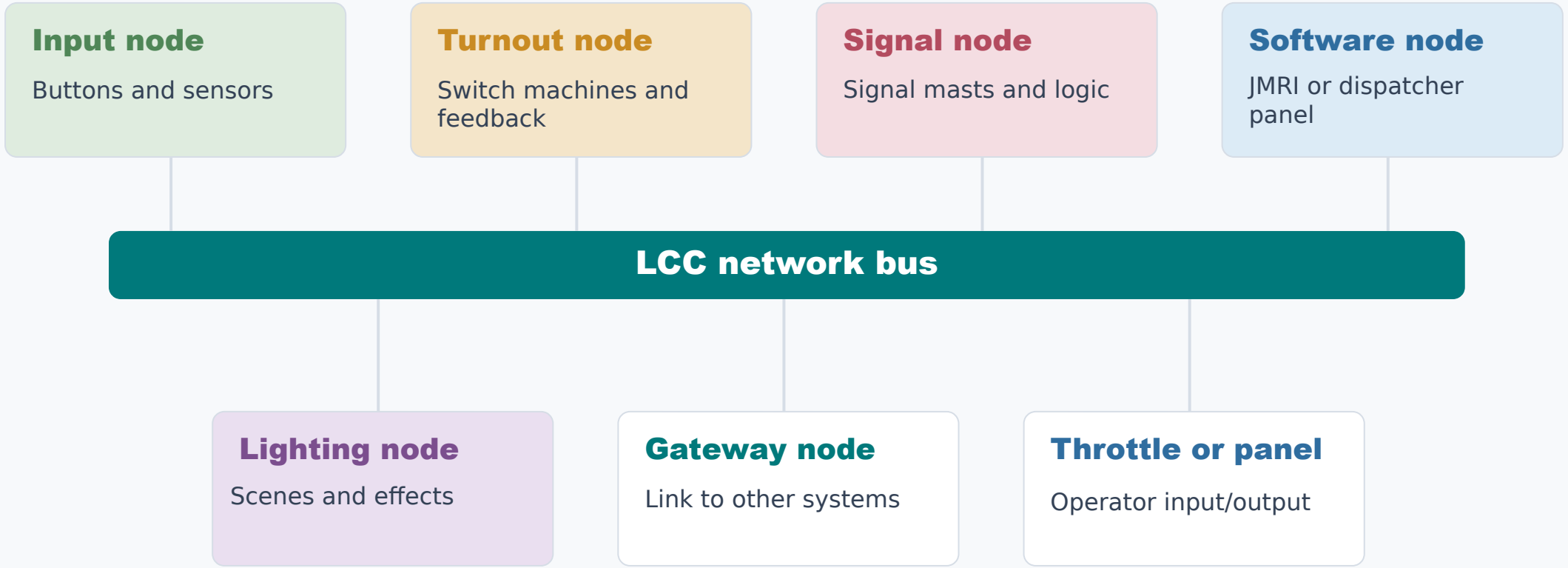
Train control paths

Throttle (wired or wireless)
↔
LCC aware Command station
→
DCC over Rails to train

Wireless Throttle
↔
Wireless train

Layout components become networked nodes

The same network can connect hardware on the layout and software at the operator desk.



LCC fits better as layout complexity grows

Size matters less than how many devices need to share state, feedback, and control.

Layout size or type	LCC fit	Why it helps	Good starting scope
Small shelf or loop	Optional	Direct wiring may be simpler unless you want panels, sensors, or animation - scenes.	One node for a small panel, animated lights, diorama, or a test area.
Medium home layout	Good	Turnouts, sensors, and fascia panels start to multiply. Direct wiring starts to grow complex.	Turnouts plus occupancy feedback.
Large home or club	Strong	Distributed nodes reduce long wire runs and can share information among nodes and operators. Local panels use minimal wiring.	District nodes and panels.
Modular or portable	Strong	Standard messages help sections reconnect predictably. Never any LCC conflicts. But we need new easy linking tools to assist here.	Per-module nodes and labeled events.
Dispatcher or signaled	Very strong	Routes and signals need reliable shared state and feedback.	Detectors, signals, and route logic.

Rule of thumb: LCC becomes more valuable when one action needs wide spread devices to interact, and physical wiring complexity is a problem.

Examples of LCC manufacturers

NMRA lists these LCC manufacturer links; verify current products before choosing hardware. Start small with an unknown supplier. Don’t worry about operating conflicts or address collisions between manufacturers.

- **RR-CirKits**
- **Train Control Systems (TCS)**
- **Logic Rail Technologies**
- **Country Robot**
- **SPROG DCC Ltd (UK)**
- **LCC Signals**
- **Model Rectifier Corp. (MRC)**
- **cmOS Engineering**
- **Snowball Creek Electronics**
- **SPROG (US)**

Commands describe the intended action

In LCC planning, commands should be simple, named, and tied to a real layout outcome. Always use unique names within a layout or group of interconnected modules.

Command principles

- State the desired result, not every electrical detail.
- Keep each command scoped to one meaningful action.
- Use clear names: *Turnout 12 norm* beats *T12N*. Be concise but not cryptic.
- Prefer route-level commands only when the route is well defined.
- Include an expected feedback event when the action should be verified.

Command examples

Turnout 12 norm
Turnout 12 rev
LO yard ladder 3
Hwy 200 flashers on
AF Engine house lights dim

Good command names are understandable when printed on a wiring diagram, but concise enough to fit on a wire.

Control confirms state and protects operations

Feedback and rules turn raw commands into dependable layout behavior.

Feedback

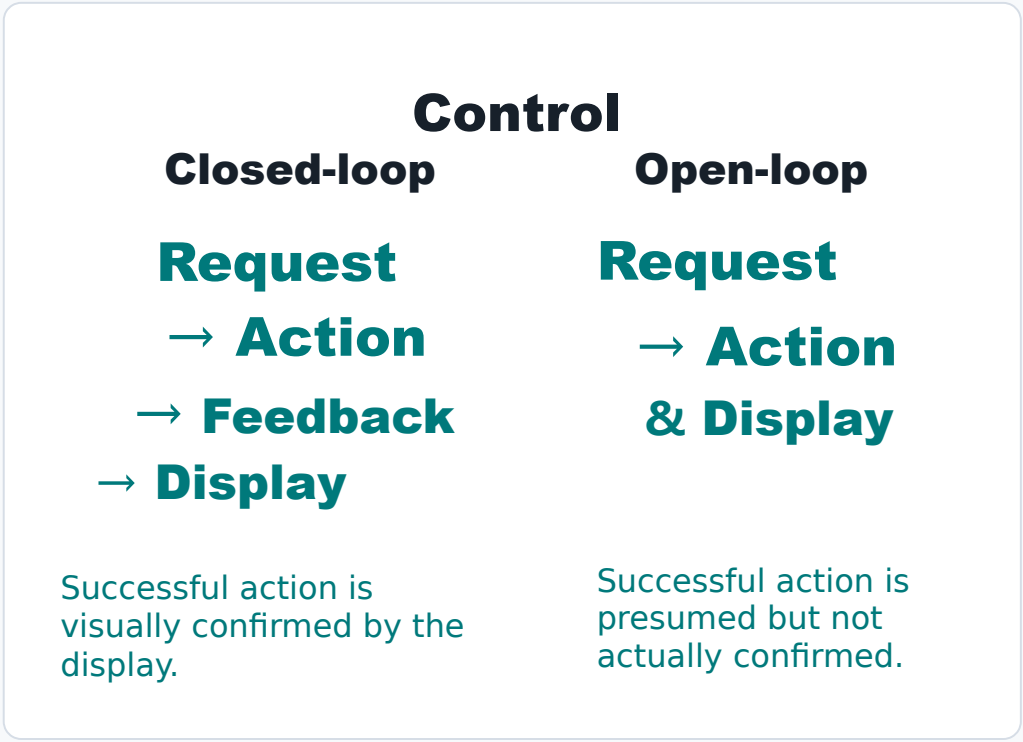
Use sensors, contacts, or node status to confirm that the layout has reached the intended state. Optional but very useful.

Interlocks

Prevent conflicting movement, such as throwing a turnout under an occupied train or clearing a signal into a blocked route.

Local resilience

Keep essential control close to the device when practical so that a node or wiring problem does not stop operation in other areas.



A typical workflow moves from plan to tested events

The best LCC installations are documented before wires and events multiply.

1. **Inventory**

   List turnouts, detectors, signals, panels, lighting, and software needs.

2. **Group**

   Place nodes near field devices to reduce long wire runs and simplify troubleshooting.

3. **Name**

   Create a naming plan for nodes, events, routes, and physical locations.

4. **Map events**

   Define who produces each event and who consumes it.

5. **Configure**

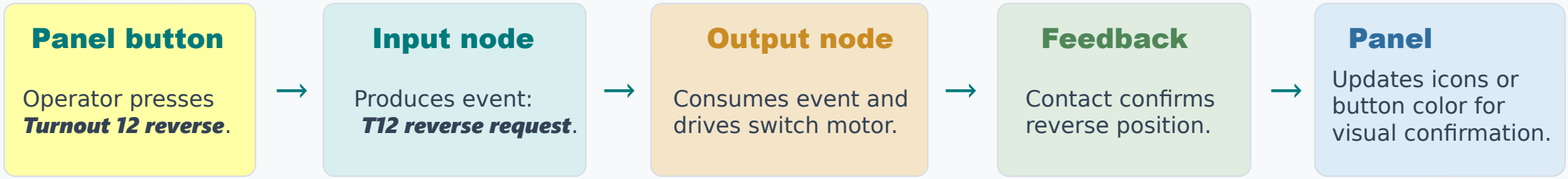
   Program nodes in small batches and back up configuration files.

6. **Test**

   Verify each event path, then test complete routes and failure cases.

Scenario: a turnout uses one clear event path

A button press, a turnout motor, and a status light can all coordinate without a dedicated wire for every relationship.



Why it matters: A sequence of event messages, controlling each button, motor driver, and feedback contact does not need direct wiring to every other device. Adding LCC messages for a more complete control setup does not create larger cables between nodes.

Scenario: signal aspects combine several layout states

Signal behavior usually depends on more than one input, so clean event design matters.

Inputs to the rule

- Block occupancy
- Turnout position
- Next signal aspect
- Route selection
- Direction of travel
- Manual override or dispatcher lock

Signal output

The signal node logic or external control software consumes the relevant events, applies the appropriate rules, and produces a signal-aspect event such as Stop, Approach, or Clear.

Block occupied → signal displays Stop

Scenario: a route command coordinates many outputs

One route event can line turnouts, update indicators, and set inputs to signal logic.

Dispatcher panel

Operator selects Yard ladder route 3.

→

Route event

Yard route 3 requested.

→

Turnout nodes

Several switch machines move to normal or reverse.

→

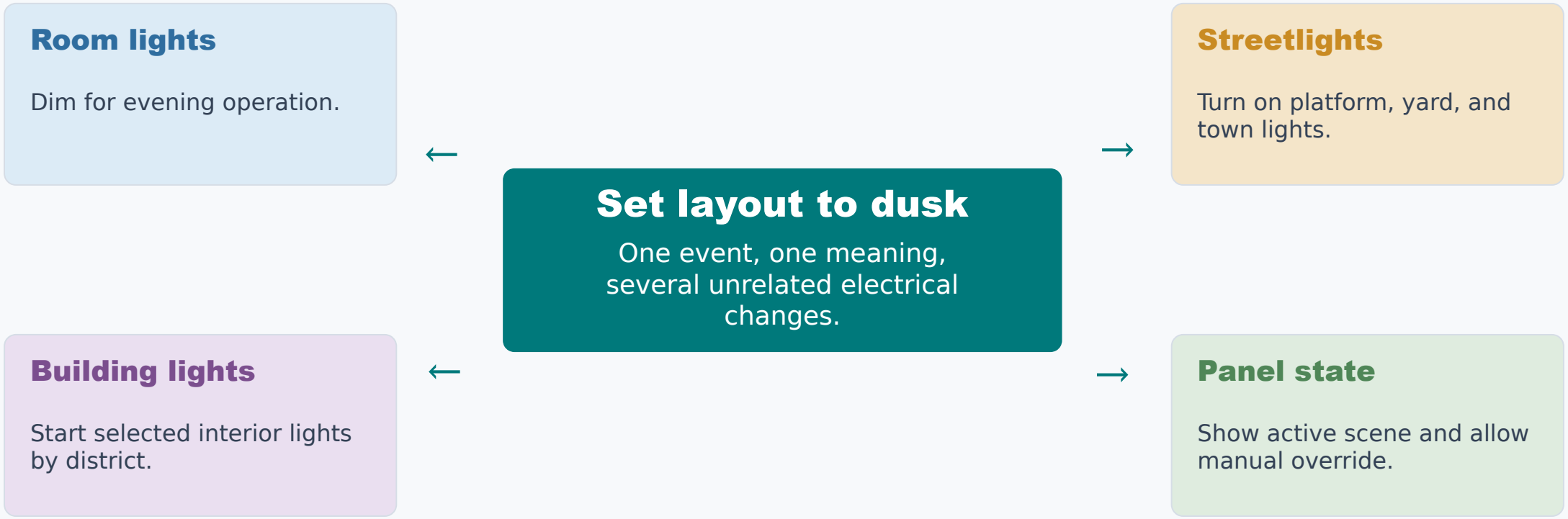
Feedback

Panel indicators and signal rules should update from confirmed states.

Design rule: a single route command should still be testable as individual turnout feedback events. The route should simplify operation, not ignore a broken device.

Scenario: lighting scenes use abstract events

Events do not have to represent hardware pins; they can represent operating states.



Scenario: growing layouts add segments without starting over

Gateways and repeaters let the network expand while preserving event meanings.

Town module

Local buttons, lighting, and block detectors.

→

Gateway or repeater

Moves needed messages between bus segments: A repeater moves LCC messages from one segment to another. A Gateway converts from LCC to some different protocol such as LocoNet.

→

Yard module

Turnout controls, route panel, and occupancy.

Why it helps

Keep event meanings when a module moves. No issues, with duplicate LCC EventIDs.

What to plan

Power, cable length, consider spare drops for future nodes, and segment boundaries.

What to test

Cross-segment events, especially for Gateways. Proper initialization of all nodes.

Best practices keep the network maintainable

A disciplined setup makes future expansion and troubleshooting much easier.

Name everything

Use location-based names for nodes, events, and routes. You will forget cryptic names.

Document event maps

Track producer, consumer, meanings and expected feedback.

Test in layers

Confirm hardware first, then test events using tables, then routes, and finally signals.

Back up configs

Save node configuration files after every stable milestone.

Leave capacity

Plan spare ports, power budget, and accessible network drops for future expansion.

Label the layout

Match physical labels to the names used in software. Things look different from underneath.

Common mistakes are usually planning mistakes

Most problems come from unclear names, missing feedback, or too much logic too soon.

Mistake	Why it hurts	Better habit
Vague or cryptic event names	No one remembers what the event means.	Use location and action names.
No feedback path	The panel says done even if the device did not move.	Add confirmation events. Upfront effort makes for better outcomes.
Assuming LCC replaces DCC	Train control and layout control get confused.	Treat them as partners.
Building everything at once	Failures become hard to isolate/correct.	Build and test in steps and layers.
No backup	A working setup is hard to recreate.	Save configs after stable changes.

Glossary: the words beginners meet first

These terms show up repeatedly in LCC product manuals, configuration tools, and discussions.

LCC	**L**ayout **C**ommand **C**ontrol, the NMRA layout-control standard.
OpenLCB	The open development effort behind the technology used for LCC.
Node	A device or software participant on the LCC network.
Event	A unique message that represents a request, condition, or state.
Producer	A node that announces an event.
Consumer	A node that responds to an event. Note: a consumer event may also produce an new event.
EventID	The actual numerical value that represents an event on the bus.
CDI	**C**onfiguration **D**escription **I**nformation, the node-supplied description tools used for setup options.
CAN	A common physical/data-link approach used by many LCC devices. (first used in modern vehicles)
DCC	**D**igital **C**ommand **C**ontrol, commonly used for locomotive control over the rails or DCC accessories.

LCC makes layout devices work as a coordinated system

The beginner model is simple:

1. Nodes connect to the LCC network.
2. Producers announce meaningful events.
3. Consumers respond to the events they understand.
4. Feedback (optionally) confirms the layout reached the intended state.

Next practical step

Draw one event map for one feature, then build and test only that feature first.

A basic LCC CAN physical network could be:

Terminator ↔ Power ↔ Node ↔ Computer Interface ↔ Terminator

Sources consulted: NMRA/OpenLCB/JMRI pages plus selected LCC-support software and manufacturer pages.

Time for questions.