

# ***RR-CirKits***

Specializing in Affordable Electronics for Model Railroads

***Installation Guide***

***Revision-b***

***July 2023***

# ***DRAFT***

## ***Signal LCC-32H***

***LCC (Layout Command and Control***

***8 line Input Output board***

***Plus***

***32 Head controller with special effects***

This PDF is designed to be read on screen, two pages at a time. If you want to print a copy, your PDF viewer should have an option for printing two pages on one sheet of paper, but you may need to start with page 2 to get it to print facing pages correctly. (Print this cover page separately.)

# Copyright

---

This document is Copyright © July 2023 by **RR-CirKits, Inc.**. You may distribute it under the terms of either the GNU General Public License, version 3 or later (<http://www.gnu.org/licenses/gpl.html>), or the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0/>), version 3.0 or later.

All trademarks within this guide belong to their legitimate owners.

## Authors

Dick Bronson

## Feedback

Please direct any comments or suggestions about this document to:

[dick@rr-cirkits.com](mailto:dick@rr-cirkits.com)

## Contact Information

RR-CirKits, Inc.  
7918 Royal Ct.  
Waxhaw, NC USA 28173

<http://www.rr-cirkits.com>  
[sales@rr-cirkits.com](mailto:sales@rr-cirkits.com)  
[support@rr-cirkits.com](mailto:support@rr-cirkits.com)  
1-704-843-3769  
Fax: 1-704-243-4310

## Publication date and software version

Published June 2023.

Firmware version rev-B6a

---

**WARNING:** This product contains a chemical known to the state of California to cause cancer, birth defects or other reproductive harm. Do not ingest.

---

*You can download an editable version of this document from  
<http://www.rr-cirkits.com/manuals/SignalLCC-32H-manual-a.odt>*

# Contents

Copyright.....	2
Overview.....	6
1 About LCC.....	7
1.1 Some Definitions.....	7
1.1.1 EventID.....	7
1.1.2 State.....	8
1.1.3 CAN.....	8
1.1.4 Node.....	9
1.1.5 Segment.....	9
1.1.6 Line.....	9
1.1.7 Consumer.....	10
1.1.8 Producer.....	10
1.1.9 Head Drivers.....	10
2 Signal LCC-32H Features.....	10
2.1 Electrical Specifications.....	11
3 Line Details.....	11
3.1 Consumer.....	11
3.2 Producer.....	13
3.3 Sample Mode.....	13
4 Connections and Indicators.....	14
4.1 CAN LCC® Compatible Connector.....	14
4.2 Power Connections.....	15
4.3 Status Indicators.....	15
4.4 Blue/Gold Buttons and LEDs.....	15
4.4.1 Error indicatons.....	16
4.4.2 Signal Test Mode.....	16
4.5 Signal LCC-32H I/O-1 Connector Wiring.....	16
4.6 Signal LED driver connections.....	16
4.6.1 Signal LED Polarity.....	16
5 Getting Started.....	17
5.1 CDI (Configuration Description Information).....	17
6 Input/Output Configuration.....	19
6.1 Identification.....	19
6.2 NODE ID.....	19
6.3 Lines (I/O Port).....	19
6.3.1 Lines.....	20
6.3.2 I/O Configuration.....	20
6.3.3 Delay.....	22
6.3.4 Commands.....	22
6.3.5 Indications.....	23
6.3.6 Signal LCC-32H Secondary Messages.....	24
7 Signal LCC-32H Logic.....	24
7.1 Logic Conditionals.....	24
7.2 Logic Description.....	25
7.3 Group function.....	25
7.3.1 Blocked -.....	25
7.3.2 Group -.....	25
7.3.3 Last (Single) -.....	26

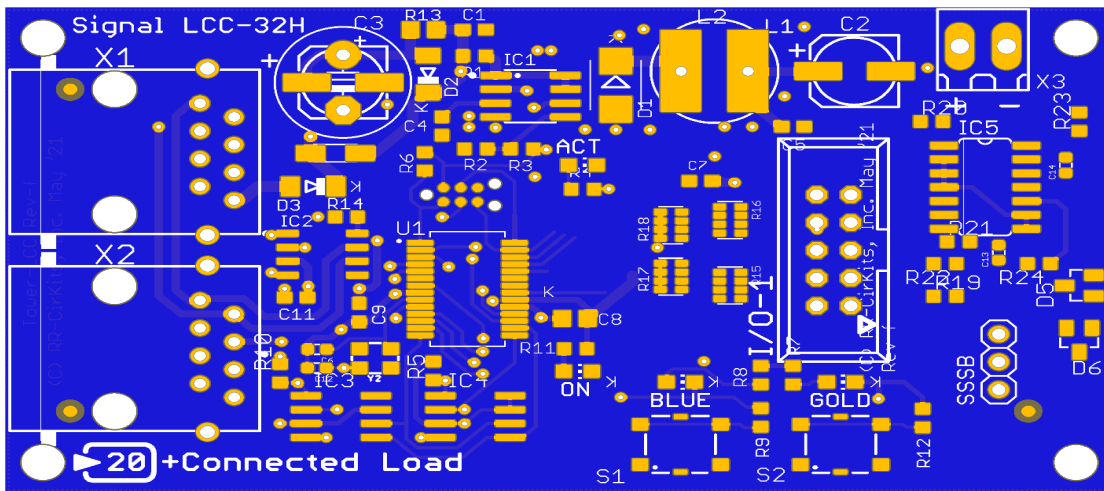
7.4	Variables.....	26
7.4.1	Variable Trigger.....	27
7.4.2	Variable Source.....	27
7.5	Variables and 'Not'.....	28
7.6	Logic function.....	28
7.7	Action when Conditional = True.....	28
7.7.1	Send then Exit Group.....	28
7.7.2	Send then Evaluate Next.....	29
7.7.3	Send then Send Next.....	29
7.7.4	Exit Group.....	29
7.7.5	Evaluate Next.....	29
7.8	Action when Conditional = False.....	29
7.8.1	Send then Exit Group.....	29
7.8.2	Send then Evaluate Next.....	29
7.8.3	Send then Send Next.....	29
7.8.4	Exit Group.....	29
7.8.5	Evaluate Next.....	30
7.8.6	Setting up Virtual Code Lines.....	30
7.9	Delay.....	30
7.10	Actions (Producers).....	30
8	Masts.....	31
8.1	Mast Processing.....	31
8.1	Mast Processing.....	35
8.1.1	Unused.....	35
8.1.2	Normal.....	35
8.1.3	Link to Previous.....	35
8.2	Mast ID.....	35
8.3	(P) Track Circuit Link Address.....	36
8.4	Lamp Fade.....	36
8.5	Rules.....	36
8.6	Aspect to Appearance.....	37
8.6.1	Individual Aspect Lamps.....	38
8.6.2	Lamp Selection.....	38
8.6.3	Lamp Phase (A-B) - Flash Rate.....	38
8.6.4	Appearance Effects.....	38
8.6.5	Effects Lamp.....	38
9	Track Circuits.....	38
9.1	Simulating a Code Line with Events.....	39
9.2	Linking Virtual Code Lines (Track Circuits).....	39
9.3	Prototype Code Line.....	40
9.4	Speed indications.....	40
9.5	ABS and APB Signal examples.....	42
9.6	Signal Lamp Wiring.....	43
9.6.1	Color Light Signal Connections.....	43
9.6.2	Color Position Light Signal Connections.....	44
9.6.3	Position Light Signal Connections.....	45
9.6.4	Searchlight Light Signal Connections (bi-polar).....	47
9.6.5	Searchlight Light Signal Connections (Dual Color).....	48
9.6.6	Stall Motor Machine Wiring.....	50
9.6.7	Switch Machine Connections.....	50

9.6.8 Semaphore Machine Connections.....	51
10 Rules.....	52
11 Brightness.....	52
12 Signal LCC-32H compatible Input/Output Cards.....	52
12.1 BOD4-CP (DCC 4 Block Occupancy Detector - Dual Turnout Driver).....	52
12.2 BOD4 (DCC 4 Block Occupancy Detector).....	53
12.3 BOD-8 (DCC Block Occupancy Detector - 8 block).....	54
12.3 OIB-8 (Opto Isolator Board - 8 input).....	54
12.4 SMD-8 (Stall Motor Driver - 8 line).....	54
12.5 RB-4 (Relay Board - 4 x SPDT).....	55
12.6 BOB-S (Break Out Board).....	55
13 Trouble shooting.....	55
13.1 Sanity Test.....	55
13.2 Activity Test.....	56
14 Boot Loader.....	56
14.1 Firmware Upgrade.....	56
15 Grounding and Isolation.....	57
16 Warranty Information.....	57
17 FCC Information.....	57
18 Signal LCC-32H Work Sheets.....	59
18.1 Sample Rule Sheet.....	59
18.2 Sample Logic Sheet.....	60

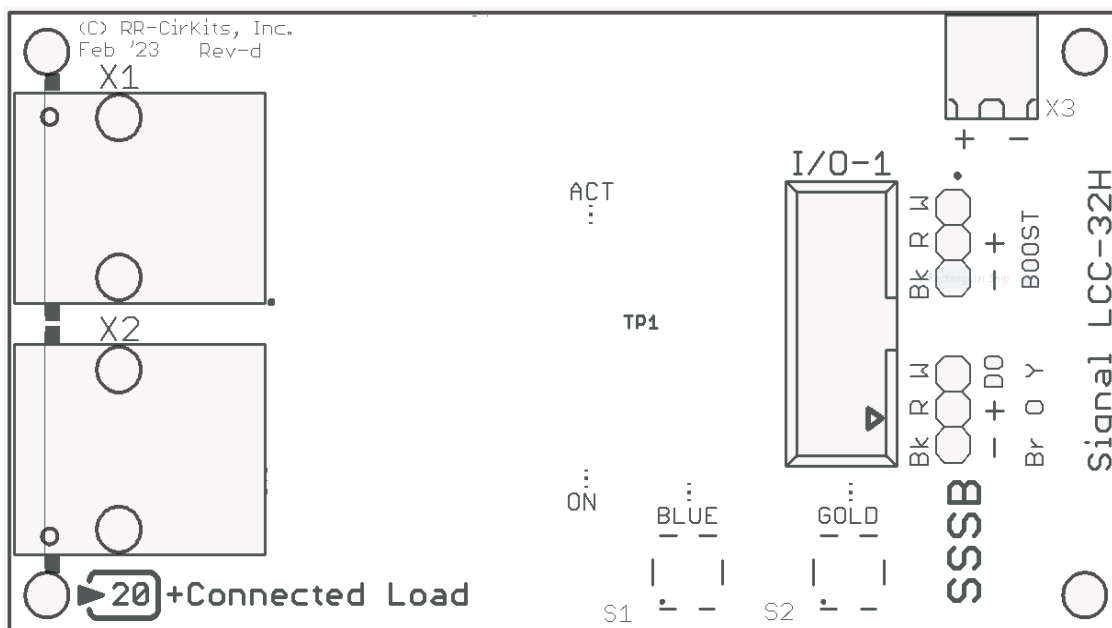
# Overview

The Signal LCC-32H (Layout Command & Control) interface provides an inexpensive and easy way to connect between the NMRA LCC® CAN bus and the signals on your layout. The Signal LCC-32H may be connected at any convenient point on the NMRA LCC® CAN bus.

The Signal LCC-32H is able to drive many heads because it moves the individual head brightness and driver circuits from the main board out to individual small head driver boards located close to, or at each head. These head driver boards are serially addressed by their position on a 3 wire SSSB (Simple Serial Signal Bus) consisting of “Futaba J” style servo extension cables.



Signal LCC-32H Image



Signal LCC-32H connectors

LCC® is a registered trademark of the NMRA. [www.nmra.org](http://www.nmra.org)

The Signal LCC-32H controller provides the tools to simulate virtually any railway signal used world wide. It includes some internal logic to calculate the proper signal rules based on local and remote events and creates matching EventIDs for each Rule.

It accepts these same (or other) EventIDs for each rule and provides a Rule (Name) to Aspect (Appearance) table that allows virtually any Aspect to be displayed on a Mast as required by your particular railroad's rule book.

However due to the large number of heads that it controls, you may need to rely on other nodes or a computer for full signal logic support.

---

*This capability requires many user options, but remember that most signaling situations will only require a few of the many options, and the rest may be ignored. Do not allow the many possible options to prevent you from using the board for simple signaling situations. For example, if you use an external program such as JMRI to control your signals the entire logic section, and the virtual track circuits may never be used.*

---

## 1 About LCC

---

The NMRA LCC® is a subset of the OpenLCB specifications created by the OpenLCB group for Layout Command and Control. <http://www.openlcb.org/>

### 1.1 Some Definitions

#### 1.1.1 EventID

Events consist of messages sent over some communications path. This can be over wires, over WiFi, over Ethernet, etc. There are LCC devices using each of these methods, and maybe even others as well.

NMRA LCC® devices produce and are controlled by events. (also sometimes called EventIDs) Each event has a unique value that will never be repeated by any other LCC® event in use anyplace on your system, nor even on anyone else's system. The only meaning given to any specific event is that which you give it. For example on the Signal LCC-32H node a particular EventID may indicate to display 'Stop' on a given signal head. A different head's 'Stop' will use a different EventID, and of course 'Clear' on the same head will also have its own unique EventID.

---

*This event uniqueness is a key difference between the LCC and legacy systems. You can always create a link between any two points in the LCC world, without needing to know anything else about the system. No more address conflicts, no more dedicating of messages to specific hardware, no more address space limitations.*

---

The LCC uses 64 bit numbers to represent events, (18,446,744,073,709,551,615 possibilities) so we are not planning to run out of unique event numbers anytime soon. Events are created by event 'Producers' and used by event 'Consumers'. The

same event may be created by one or more Producers, and may be used by any number of Consumers. (or none at all)

---

*Events are not like wires. They are messages that tell us things, not something that can be observed with a meter or lamp. They are as transient as a flash of light, and have no meaning other than what you give them.*

---

Events happen, they are not logic states nor the status of indicators nor the position of switches, they are simply messages that indicate changes to these things.

### **1.1.2 State**

The only memory of past events exists in hardware that has responded to them. An event can tell you to turn a light 'on'. A different event can tell you to turn a light 'off'. Different events can also tell you to turn the same light 'on'. However, there is no event to tell you that the light is 'on'. That is a **state**, and only resides in the hardware that controls the light. (or some other hardware that is shadowing it)

We use variables to remember the state of things in logic. We use variables to remember if an output is active or not. In other words the node remembers that a line is on or off. We call this memory variable the 'State' of the output. Note that the state of an output may be 'on' but the actual voltage on the output line may be blinking or a pulse, etc.

### **1.1.3 CAN**

From Wikipedia, the free encyclopedia

A **Controller Area Network (CAN bus)** is a robust [vehicle bus](#) standard designed to allow [microcontrollers](#) and devices to communicate with each other in applications without a [host computer](#). It is a [message-based protocol](#), designed originally for [multiplex](#) electrical wiring within automobiles to save on copper, but is also used in many other contexts.

We chose the CAN bus version of LCC for our first products because we feel that it is a good match for the requirements of a model railroad layout. The voltages are low for safety, but the bus itself is very noise immune which is important in the noisy, often inexpertly wired environment that we call a 'Layout'.

The CAN bus may be operated at any speed. However the CAN bus length varies inversely with how fast you move data over the wires. The NMRA LCC uses a speed/length combination of 125kb/s and 1000'. (industry rates 125kb at as much as 1500', so the NMRA is actually being conservative)

The CAN error detection/correction method is very robust, and average data rates do not drop at all in the presence of collisions. This data rate is sufficient to carry about 10 times the amount of data as is carried over DCC.

The CAN bus is an asynchronous peer-peer network which means that any two nodes may communicate with each other without any master node or computer required. Unlike DCC, this peer-peer nature of course means that data may flow just as easily in one direction as the other.



### **1.1.4 Node**

We (RR-CirKits) use the term 'Node' to indicate a single device or board that has both an electrical and logical connection to an LCC network. Some nodes may have multiple logical connections to the network, but only count as one node because there is only a single electrical connection. (transceiver) Some devices may have an electrical connection to the network, but not interact with the LCC logically in any way. An example might be a Repeater. It is electrically connected, but other nodes can not interact with it in any way. It does not count as an LCC node, but must be accounted for when counting the number of devices on a segment.

---

*Please note that this is not exactly the same usage of the term 'node' as is documented in the NMRA LCC specifications.*

---

### **1.1.5 Segment**

On a CAN based LCC network there are electrical limitations on the total number of devices connected to the same cable, or 'Segment'. These limitations take several forms. Electrical limits may be overcome by the use of a repeater.

- Electrical current limits. The CAT5 cable used has a limitation of 1A per conductor. The user is responsible to assure that sufficient power is supplied to the cable to supply all nodes drawing power from the bus within 20' of a power injection point without exceeding this amount. Each node is marked with the amount of current required or supplied to assist the user in this calculation. Be sure to count external loads such as signal lamps, etc.
- Propagation delay. The speed of any CAN network is inversely proportional to its total length. The LCC CAN network was chosen to run a maximum of 1000'/300m at 125K bits per second. This is a good match to most layouts.
- Further, the maximum cable length is reduced by 20'/6m for each physical node attached to the segment. This limits each segment to about 48 nodes, or even fewer for longer segments.
- Each CAN segment must form a single serial string of nodes with a single termination at each end. Short side branches are allowed, but they count as double their length when subtracted from the total segment length. This requirement assures reliable data transmission at high speeds. The terminators not only prevent electrical reflections, but also provide the load for the recessive data bits. The lack of long branching prevents introducing jitter into the signals. Both things are necessary to prevent issues inherent with long lines at high data rates.

### **1.1.6 Line**

Each Signal LCC-32H contains 8 general purpose I/O lines plus control for 32 head drivers boards. (3 LEDs per head) Each I/O line has the ability to watch for 6 events (consumers) and to send out 6 events. (producers) Each I/O line contains two registers. One register remembers the 'State' of the line. (on or off) The second register remembers the state of the 'Veto' option. The veto option may be used to allow or disallow some events used to control or respond to the line.

### **1.1.7 Consumer**

Consumer EventIDs are those that are used to control the node and its outputs. The term 'Consumer' simply means that the node consumes or uses the EventID for some purpose. Some example consumers would be events that turn on or off an output line, or that enable or disable a veto function. Normally Consumer events change the state or status of some part of the hardware. A consumer may also be used in logic, and depending on the options set, may be used to trigger some other action by in turn producing new events.

### **1.1.8 Producer**

Producer EventIDs are those created or 'produced' by the node as a result of something related to the node. This could be a change on an input line, a change of a function generator, or the result of some logic calculation done with on board logic circuits.

### **1.1.9 Head Drivers**

In addition to its 8 general purpose I/O lines, each Signal LCC-32H can control up to 32 head driver boards each with support for 3 signal LEDs with special effects on each driver. These driver boards are normally used in combinations of up to 4 LEDs per each aspect. This allows for the creation of most signal aspects found world wide. Masts are created by combining multiple head drivers together, but limited to a maximum of 16 aspects per mast by the Signal LCC-32H hardware design. This is covered in the 'Masts' section 7.

The firmware also allows each output LED to be individually controlled by two EventIDs. This makes it easy to use the LEDs for control panels, crossing gate flashers, or building lighting where each LED needs to respond to a single control pair. Each lamp must be selected to be controlled either by Individual Lamp Control, or by Signal Head control. This selection option is found in the Individual Lamp Control segment, and defaults to Signal Head Control.

For Individual Lamp Control there are only two consumer EventIDs available and they are normally controlled as a 'Turnout' by JMRI. Basic brightness and fade effects are supported

## **2 Signal LCC-32H Features**

---

- The Signal LCC-32H uses the CAN bus implementation of the NMRA LCC.
- Communicates over the LCC CAN bus at 125Kb.
- Support for a total of 8 general purpose Input/Output lines:
  - Up to 8 Input Lines. (internal pull-up termination on all lines)
  - Up to 8 Output Lines.
  - Any mix of inputs and outputs by using 'Sample Mode'.
- Control for 32 signal driver boards - (96 LEDs total). (internal current limiting for each LED)
- Rule (Aspect) based signal control for up to 512 rules. (16 total per mast group)

- Each rule can light any of the available LEDs in its mast group.
- Internal Logic Blocks with up to 32 conditional statements.
- Support for up to 8 virtual code lines per mast. (virtual speed tables linking masts)
- CDI (Configuration Description Information) controlled programming via a program connected to the LCC bus.
- I/O Lines may be configured as I/O ports or as individual lines.
- Automatically saves all input/output, signal aspect, and logic states during power down.
- Boot Loader allows touch-less user firmware upgrades over the LCC® (Layout Command & Control) bus connection.
- Power is supplied over the LCC® bus, or from an external power source. The Signal LCC-32H itself requires 20mA. plus whatever load may be imposed by the I/O module that you choose, plus the power supplied to the LED drivers that you have connected.

## 2.1 Electrical Specifications

I/O Port 1:

- Pin 1 (line 8) - 5 volt logic level at  $\pm 25\text{mA}$ .
- Pin 2 (line 7) - 5 volt logic level at  $\pm 25\text{mA}$ .
- Pin 3 (line 6) - 5 volt logic level at  $\pm 25\text{mA}$ .
- Pin 4 (line 5) - 5 volt logic level at  $\pm 25\text{mA}$ .
- Pin 7 (line 4) - 5 volt logic level at  $\pm 25\text{mA}$ .
- Pin 8 (line 3) - 5 volt logic level at  $\pm 25\text{mA}$ .
- Pin 9 (line 2) - 5 volt logic level at  $\pm 25\text{mA}$ .
- Pin 10 (line 1) - 5 volt logic level at  $\pm 25\text{mA}$ .

Note: These are absolute maximum ratings. Normally I/O loads should be limited to much less than these values.

## 3 Line Details

---

### 3.1 Consumer

Each consumer event may be configured to control the line's state or veto register state in one of several ways. These are;

- 'None'
- 'On (Line Active)'
- 'Off (Line Inactive)'
- 'Change (Toggle)'
- 'Veto On (Active)'
- 'Veto Off (Inactive)'
- 'Gated On (Non Veto Output)'
- 'Gated Off (Non Veto Output)'

The 'Output Function' may be configured for steady response, pulse response, blinking response, or various sample combinations. Note that this is for the actual output line voltage, not the state of the line as controlled by the events. The state of the line is 'on' or 'off' as controlled by the events. The output of the line may be None, Steady, Pulse, or Blinking (phase A or B) based on the state. It may be Active High or active Low. (default)

Output Function:

- None - No Output Function
- Steady - Means that the output line will be high or low respectively when the output is 'on', and the opposite when it is 'off'.
- Pulse - Means that the output line will pulse to either high or low level based on the timing delay intervals, then return to normal. Delay 'Interval 1' sets the time delay before the pulse occurs, and delay 'Interval 2' sets the pulse length itself.
- Blink A - Used to control devices such as crossing gate flashers directly from the output lines. 'A' and 'B' are the two phases of the flashing. Blink A or Blink B refers to which phase starts the action.
- Blink B - Delay 'Interval 1' sets the length of phase A, and delay 'Interval 2' sets the length of phase B. For a Signal LCC-32H board it will be more realistic to use the LED drivers that can also add in the proper fade effects.

Drive Polarity: This is where you determine if a line is high or low when it is 'on'. For example many devices use a positive common connection and ground the control lines when they turn 'on'. (sometimes called negative logic) This is the default for our devices.

- Hi (5V) - High voltage when 'on'
- Low (0V) - Low voltage when 'on'.

The hardware uses an internal function generator that may be configured to create different types of actual outputs. These are; steady (output line follows the output state), blink (output line alternates when output state is 'on'), and pulse. (output line alternates one time when the output state is first 'on')

When a line is configured as an 'output', then the timing intervals are used to control the function generator timing of pulses and blinks. This also includes delay settings for both 'on' and 'off' transitions. These delays can be used to control the blink rate and pulse length, or to simply delay the output action for some interval after the controlling event is seen. (e.g. to simulate "running time" on a CTC panel) Delay 'Interval 1' is the time before the output changes, and delay 'Interval 2' is the length of the pulse or blink.

The consumer events may also be used to control a Veto state. If the veto state is 'on', then the consumer 'Gated on' (activate) and 'Gated off' (inactivate) events are ignored unless in 'sample' mode. The consumer veto action is not active in 'Sample' mode because 'sample' mode implies that the output is always active.

The producer 'on' (Non Veto Output), and 'off' (Non Veto Output) are also ignored (blocked) when the veto state is 'on'. These rather awkward names mean that these 'on' or 'off' events are only produced when the output is not vetoed.

## 3.2 Producer

Input Function;

- None - No Input Function
- Normal - The input is direct acting, EventIDs are sent for each change.
- Alternating - The input is alternate action. EventIDs are sent alternately for each change.

Input Polarity: This is where you determine if a line is high or low when it is 'activated'. For example many devices use a ground for the control lines to turn 'on'. (sometimes called negative logic) This is the default for our devices.

- Hi (5V) - High voltage is 'on'.
- Low (0V) - Low voltage is 'on'. (default)

Each producer event can be configured to trigger in one of 9 ways:

- 'None'
- 'Output State On command' and 'Output State Off command', are trigger options that allow you to produce a new event based on receiving a command to activate or inactivate the output. This might be used to cascade a yard ladder. This option is used to produce an EventID based on other consumer events.
- 'Output On (function hi)' and 'Output Off (function lo)', trigger a new event when ever the actual output of the function generator changes. This might be used to build a realistic traffic light controller where each phase is triggered by the end of the previous phase. **Be very careful with this option because it can create a lot of traffic continuously**, especially if the function output is blinking rapidly.
- 'Input On' and 'Input Off', allow you to trigger events based on a change of the input line. This is the normal and default use of a producer.
- 'Gated On (Not Veto Input), and 'Gated Off (Not Veto Input)', allow events to respond to, or ignore, any input changes based on the veto state. For example this would allow you to enable/disable fascia buttons for local control of a turnout by using the output events from a panel switch to control the veto.

Note that the 'on'-'off' time delays are used as function output delays or input debounce delays depending on the line's status.

## 3.3 Sample Mode

Sample Mode: The I/O lines may be configured as both inputs and outputs on the same line with some restrictions. Note that this requires specially designed devices such as the Berrett Hill Touch Trigger, the RR-CirKits BOD4-CP board, or the Button Quik-Link, that can support both input and output data on the same wire.

The restrictions are:

- Any input must include a 1K series resistor to prevent shorting out any output that may be active at the same time.
- Any output must not load the line with more than a 10K load to prevent the load from giving a false input.

Also note that the output function must be tolerant of the brief sample times when the output may change state during the sample period.

The Signal LCC-32H will automatically enter Sample Mode if both Output and Input are enabled on the same line.

## 4 Connections and Indicators

The Signal LCC-32H (8 Line I/O Board plus 32 head signal controller) has six connectors and four status indicators. Two of these connectors are the standard RJ45 connections to the LCC bus network. One, the I/O port, is used as connections to the 8 I/O lines. The other three are used to power and connect to the signal drivers. This section covers the system connections consisting of the CAN bus port connectors, I/O port connections, Status indicators, Power connection, and LED driver connections. See images at page 6.

### 4.1 CAN LCC® Compatible Connector

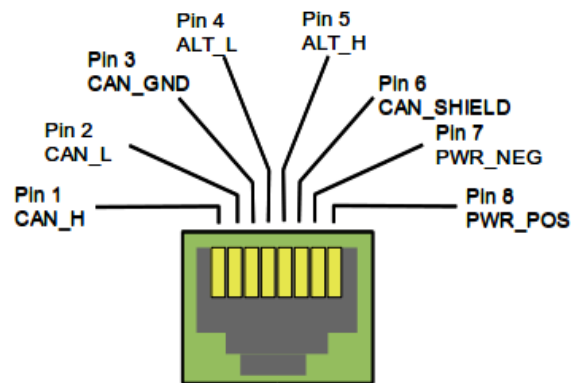
The data connection is made to the Signal LCC-32H via a standard RJ-45 CAT5 cable connected to either of the two RJ-45 connectors. (X1 and X2) The LCC wiring passes straight through between both connectors.

The LCC specification requires a minimum of 1' of cable length between connectors. However shorter cables should not significantly impact operation on smaller layouts or space constrained situations.

These cables are commonly sold for wired Ethernet use.

#### Pin outs for the CAN LCC RJ-45 data connector:

Pin	Description
1	CAN H
2	CAN L
3	CAN GND
4	Alt L (DCC negative)
5	Alt H (DCC positive)
6	GND
7	GND
8	+Power 12-27V



LCC power is supplied between Pin 7 and Pin 8. Power can be from +12VDC to +27VDC. The RR-CirKits LCC Power-Point delivers approximately 15VDC to the bus.

The LCC connectors accept standard Ethernet style CAT5 (or better) cables. 4 pair cables are required by the Signal LCC-32H. For any but the smallest networks it is recommended that you choose AWG 24 CAT-5 cables. The use of AWG 26 wiring reduces the maximum length of your network to approximately 40% of its specified length. Especially avoid using copper clad aluminum wire or AWG 28 low profile wiring as they have even higher resistance at the relatively low frequencies used by the LCC. This higher resistance shortens the maximum distance for reliable communications even more than using AWG 26 wiring does.

A note on connectors for hand built cables: RJ-45 crimp connectors are made with three blade styles. (the piece that crimps into/onto the wire) Single 'U', double 'UU', and triple 'VVV' points. Stranded cables may be made with any of the three blade styles because the points crimp into and between the individual wire strands. However if you are using solid wire, then you must only use the three point style of blade. It is designed to trap the solid wire between the three points, two on one side, and the center one on the other side, for a corrosion tight connection. The single or double pointed blades simply press against the side of the solid wire, and will fail in time. (usually the morning of your open house or operating session)

## 4.2 Power Connections

The Signal LCC-32H node requires an external power source of between 7.5 and 27 volts DC from the LCC cable and/or from the X3 connector. (Note the X3 connector does NOT supply any power to the LCC bus.)

The LCC Power-Point unit is a convenient way to supply the required power to the Signal LCC-32H and other LCC boards over standard RJ45 cables.

Each segment of LCC® cable requires a terminator at each end. Power can also be supplied by other powered LCC modules, with the RR-CirKits LCC Repeater, or even with properly wired adapter cables.

## 4.3 Status Indicators

The Signal LCC-32H has two status indicators located near to the LCC connectors. The green ON status indicator shows the power status of the Signal LCC-32H itself. The red ACT (activity) indicator normally shows all data activity on the bus, and also any activity/error status during a boot loader firmware upgrade. (see section 8.0)

## 4.4 Blue/Gold Buttons and LEDs

A limited amount of configuration may be accomplished by using the Blue and Gold push buttons and indicators.



*LCC Power-Point shown with Terminator*



### 4.4.1 Error indicators

The Gold LED can indicate two different error messages. If it is blinking (50% duty cycle) it indicates that it is idling in forced boot loader mode. If the Gold LED is flashing (10% duty cycle) it indicates that the board was unable to initialize itself on the network, most likely because it could not establish an alias with another node.

### 4.4.2 Signal Test Mode

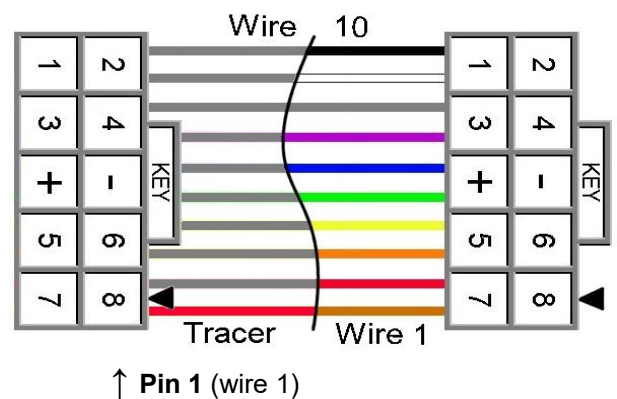
If the Blue and the Gold buttons are pressed simultaneously for several seconds the node will go into a signal test mode. When in signal test mode each decoder/driver unit will cycle between red, green, yellow, and off. (or whatever color it is showing normally)

## 4.5 Signal LCC-32H I/O-1 Connector Wiring

The I/O port connector's wiring is as follows. Note that the pin numbers and I/O line numbers are NOT the same, and actually run opposite to one another.

Pin (wire) number	Connection name
1	line 8
2	line 7
3	line 6
4	line 5
5	Ground
6	+5VDC
7	line 4
8	line 3
9	line 2
10	line 1

10 position IDC cable



## 4.6 Signal LED driver connections

All Signal Head Driver boards are connected in a daisy chain configuration using 'Futaba J' style extension cables connected from one driver board to the next.

### 4.6.1 Signal LED Polarity

Currently the only driver boards that we have available are for common anode wired signals. Older Atlas signals and Tomar searchlight signals are wired Common Cathode and not suitable for these drivers.



## 5 Getting Started

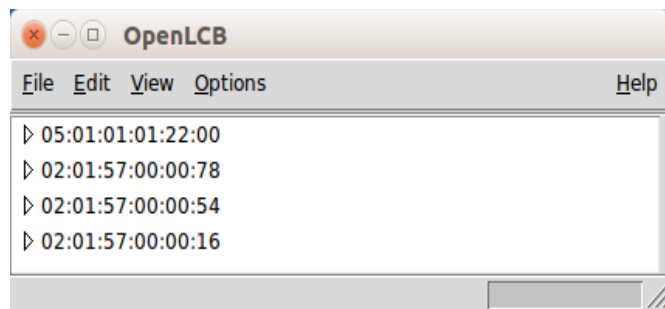
We suggest that you use a computer program such as the JMRI DecoderPro (4.8 or later) <<http://www.jmri.org/>> or Deepsoft LCC-CDI <<http://www.deepsoft.com/>> to configure the Signal LCC-32H. These "point and click" interfaces will save you much time and frustration while setting the many possible options that you will need to configure, and in fact are the only way that we offer for configuring the Signal LCC-32H node.

**Node Address:** Each Signal LCC-32H has a globally unique node address that is used for CDI programming on the layout. Each individual Signal LCC-32H has this node address imprinted on a label on the back side of the board.

### 5.1 CDI (Configuration Description Information)

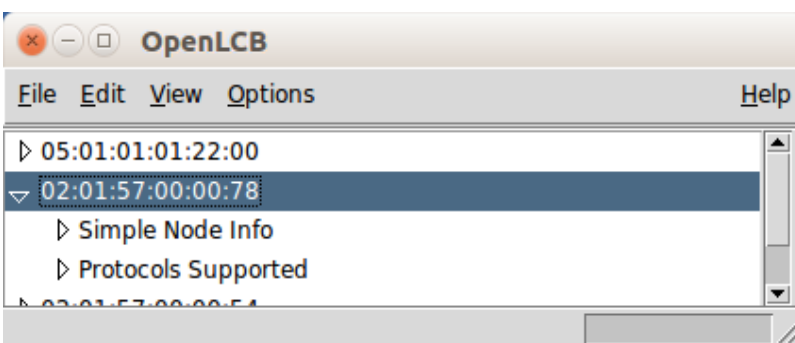
The CDI is the way that all LCC nodes store their configuration options internally. Instead of relying on printed manuals or volunteer created files to present the various decoder options, (like most DCC devices have for the past 20 years) the LCC specification expects the manufacturer of the LCC node itself to present its capabilities and options in a standardized manner from an internal file. This allows any LCC configuration tool to be used interchangeably, and not need to be updated to support the latest hardware or firmware upgrades.

**Start** up the CDI tool. Once the tool is monitoring the LCC network you will be presented with a list of nodes similar to this.



*CDI Window*

The list should include all the nodes that are currently visible on your LCC network. In this example the first entry in the list (05:01:01:01:22:00) is the configuration program itself as seen through the interface.



*Open Node*

**Select** the node that you desire to configure and click on its arrow to open it. This will open up further options for that node. In this example there is an option for 'Simple Node Information' and a list of 'Protocols Supported'.

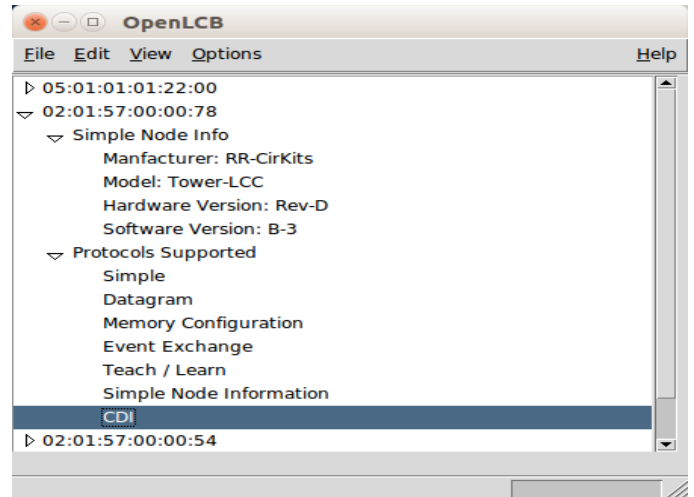
The Simple Node Information will help you to be sure that you have chosen the correct node.

With the Deepsoft OpenLCB program you simply highlight a Protocol item to open it in a new window.

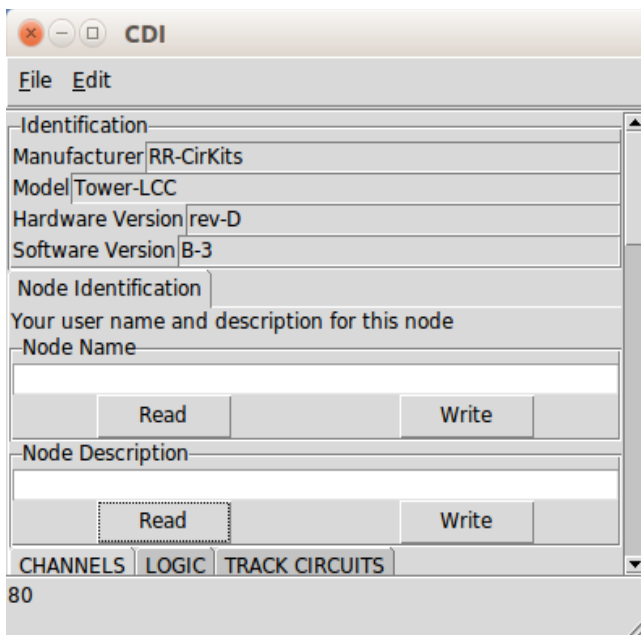
The JMRI CDI tool is similar.

**CDI** is the supported Protocol that you will need for configuration purposes.

Remember that with the LCC this display information is provided by the manufacturer and stored in the node itself rather than on some piece of paper, your memory, or some external file or program. This means that if you use JMRI instead of Deepsoft OpenLCB to open your node for configuration, the layout of the tool windows may be different, but the content will remain unchanged.



*Highlight 'CDI' to open it in a new window.*



*Deepsoft CDI Window*

value you must always then do a 'Write' to store it into the node.

The newest JMRI CDI tool will highlight the entry with orange until it has been written to the board. This is a helpful reminder that the change has not yet been stored into the board where it can take effect.

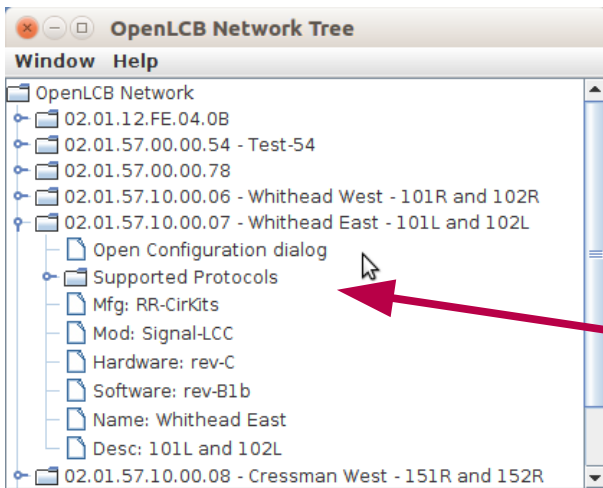
Once the CDI window opens up you can modify its contents by using the 'Read' and 'Write' buttons found near to each item.

**Read** button will fill in the field with any contents that were previously stored in the node. Using 'Read' for an EventID will present a new, unused, value if one has not been previously stored.

There is also a 'Read All' button located at the bottom of the window. Be forewarned that it can take a long time to fill in all of the options and values stored in the node, so you may want to use discretion with this option.

**Write** button will store the currently displayed value or selection into the node's memory. If you have changed any

# 6 Input/Output Configuration



*Open the JMRI CDI Window*

We suggest that the user take advantage of the JMRI CDI tool or a similar program to set the Signal LCC-32H configuration values.

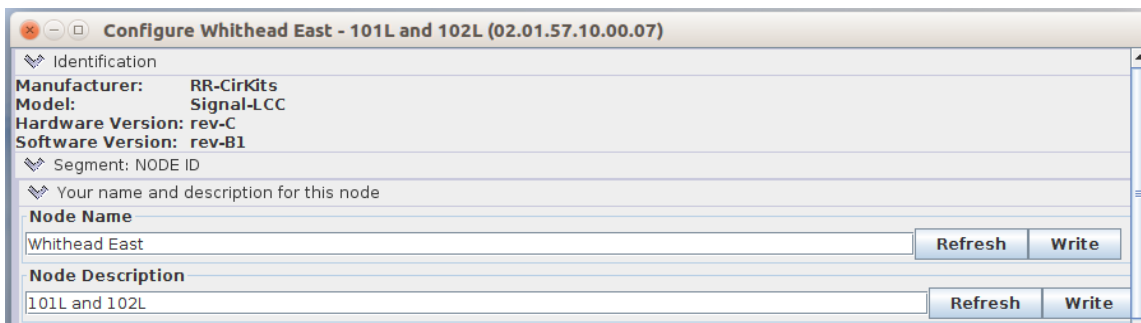
The following examples are using the JMRI CDI tool for the Signal LCC-32H. Select the 'OpenLCB' drop down list and click on 'Configure Nodes'. When the node selection window opens, choose the node to configure, and click on 'Open Configuration dialog'.

This will open the CDI tool and automatically read in the basic information for the node.

This information is presented in a tabular format to allow a reasonably compact display but still have easy access to the vast amount of configuration information.

## 6.1 Identification

The first section shown will be the Identification. It includes the manufacturers name and node model plus any version information.



*JMRI CDI Window*

## 6.2 NODE ID

The next item is the Node Identification. It contains the Name and Description that you give to the node. The name of this node is 'Whithead East' and it controls masts 101L and 102L. In the JMRI CDI tool this name and description will appear in the node selection window to make it easier to select the correct node for configuration. There is a 64 character limit to the name and description items, but clear and concise is best.

## 6.3 Lines (I/O Port)

The Signal LCC-32H has one 8 bit Input/Output port for a total of 8 lines. This port is named 'I/O-1' and is normally configured to be either all Inputs, or all Outputs, to be compatible with the various RR-CirKits I/O modules. However each line may

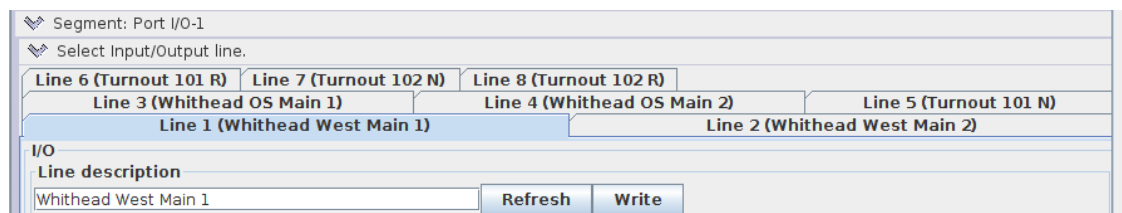
be individually set as either input or output for special purposes. One example of a mixed configuration is the BOD4-CP which has 4 input lines, and 4 sampled lines.

For the special case of one wire I/O a line may even be configured as both input and output at one time. (Sample Mode section 1.2) The Berrett Hill Touch Trigger is an example of a one line device. The Signal LCC-32H can both control the Touch Trigger's color using consumers and report its output using producers on the same line.

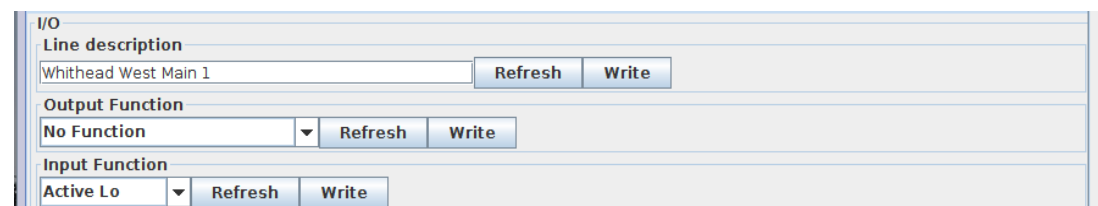
Any special effects may be applied differently for each line. E.g. one line may be held steady while another sends a pulse or is blinking.

### 6.3.1 Lines

Each I/O line is called a 'Line' and is selected by tabs and presented separately. Any previously entered 'Line Description' will automatically be shown in the tab by JMRI to make searching simpler.



### 6.3.2 I/O Configuration



For example in the above snapshot I have selected the Line #1 'Output Function' to be 'No Function' and its 'Input Function' to be 'Active Lo'.

The Output function options are:

- No Function - no output action
- Steady Active Hi - the output voltage follows the output state. (true = 5V)
- Steady Active Lo - the output voltage is the inverse of the output state. (true = 0V)
- Sample Steady Active Hi - the output voltage follows the output state while the line's input is sampled.
- Sample Steady Active Lo - the output voltage is the inverse of the output state while the line's input is sampled.
- Alt Sample Steady Active Hi - the output voltage follows the output state while the line's input is sampled to produce alternating events.

- Alt Sample Steady Active Lo - the output voltage is the inverse of the output state while the line's input is sampled to produce alternating events.
- Pulse Active Hi - the output voltage pulses to +5V when the output state is true.
- Pulse Active Lo - the output voltage pulses to 0V when the output state is true.
- Sample Pulse Active Hi - the output voltage pulses + when the output state is true, and the line's input is sampled.
- Sample Pulse Active Lo - the output voltage pulses - when the output state is true, and the line's input is sampled.
- Alt Sample Pulse Active Hi - the output voltage pulses + when the output state is true, and the line's input is sampled to produce alternating events.
- Alt Sample Pulse Active Lo - the output voltage pulses - when the output state is true, and the line's input is sampled to produce alternating events.
- Blink A Active Hi - the output voltage blinks phase A +5V when true.
- Blink A Active Lo - the output voltage blinks phase A to 0V when true.
- Blink B Active Hi - the output voltage blinks phase B +5V when true.
- Blink B Active Lo - the output voltage blinks phase B to 0V when true.

The input function options are:

- Disabled - no input action.
- Active Hi - the input is set true when it goes high.
- Active Lo - the input is set true when it goes low.
- Alt Action Hi - the input alternates true/false when it goes high.
- Alt Action Lo - the input alternates true/false when it goes low.
- Sample Hi - the input is set true when it samples high.
- Sample Lo - the input is set true when it samples low.
- Alt Sample Hi - the input alternates true/false when it samples high.
- Alt Sample Lo - the input alternates true/false when it samples low.

Normally a line should only have output functions or input functions enabled. The exception is when one of the 'Sample' options is enabled. Note that these configuration options relate to the true or false status of the line, not the actual output voltage levels on the lines or events consumed or produced. These are controlled in the individual event settings.

For safety, any input function will over ride any output function so you must be sure to set an input to 'Disabled' in order to use it as an output.

### 6.3.3 Delay

Each line includes two delay timers that are used to control blinks, pulses, and input debounce times.

Interval 1 controls the 'on' delay or time, and Interval 2 controls the 'off' delay or time. The count may be set from

0-60,000 and the base interval may be set to Milliseconds, Seconds, or Minutes. This allows for delays from 1ms. to over 41 days. Probably the extremes will never be required, but this gives you a good range to choose from. Actual accuracy is 1/2 % or better.

Retrigger allows the time interval to be reset if the line state is set to 'true' again prior to the end of the delay time.

If the line state is set to 'false' prior to the end of the delay time, then the output does not occur. This allows a timer to be reset prior to its action completing.

Delay  
Delay time values for blinks, pulses, debounce.

Interval 1 Interval 2

Delay Time (1-60000).  
0 Refresh Write

Milliseconds ▼ Refresh Write

Retrigger  
No ▼ Refresh Write

### 6.3.4 Commands

Commands are consumer events that control the line. A command can directly control the line, for example by turning it on or off. A command may also indirectly control the line, for example by controlling its veto status. Each line has 6 consumer events associated with it. Each Consumer event may control one of 8 possible actions.

For example event 1 could be used to turn an output 'on', and event 2 could be used to turn the output 'off'. Another example could be a single event that serves as a master reset for many turnouts, simply by adding it as an additional 'On' or 'Off' event to each turnout in the group.

Commands  
Consumer commands.

Event 1 Event 2 Event 3 Event 4 Event 5 Event 6

EventID  
(C) When this event occurs,  
02.01.57.10.00.08.00.00 Refresh Write Copy Paste Search

the line state will be changed to.  
None ▼ Refresh Write

The 8 possible consumer event actions are:

- None - Default no action
- On (Line Active) - Sets the output state to 'true'
- Off (Line Inactive) - Sets the output state to 'false'
- Change (Line Toggle) - Changes the output state to the opposite state

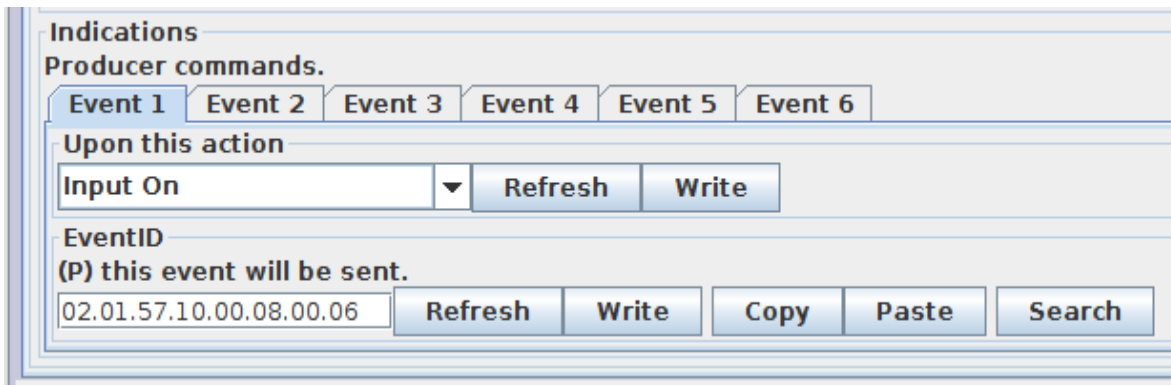
- Veto On (Active) – Sets the veto state to ‘true’
- Veto Off (Inactive) – Sets the veto state to ‘false’
- Gated On (Non Veto Output) – Sets the output state to ‘true’ only if veto is ‘false’
- Gated Off (Non Veto Output) – Sets the output state to ‘false’ only if veto is ‘false’

The last two items probably need some clarification. An event set to ‘On’ will always set the output state to ‘true’. However an event set to ‘Gated On’ will only set the output to ‘true’ if the veto state is ‘off’. If the veto state is ‘on’ then the ‘Gated On’ and ‘Gated Off’ events will be ignored and have no effect.

For example when a CTC operator controls a turnout he would send events configured as ‘On’ and ‘Off’. However a local operator button would send (different) events configured as ‘Gated On’ and ‘Gated Off’. The CTC operator could then activate a ‘Lock Lever’ that would send a ‘Veto On’ event that would block the local operator from controlling the turnout, but still allow normal operation from his own panel.

### 6.3.5 Indications

Indications are the producer events that are sent out to the bus. Each line has 6 producer events associated with it.



Each Producer event is associated with one of 8 possible actions. For example event 1 could be used to indicate that the input line is ‘on’, and event 2 could be used to indicate that the input line is ‘off’. Another possibility would be to send events based on the actual changes to an output line. A blinking output could send an event for each time it goes on and/or off. Be very careful with this option as it can easily flood the network with events.

The producer actions are:

- None
- Output State On command – Responds to an output state change to ‘true’.
- Output State Off command – Responds to an output state change to ‘false’.
- Output On (Function hi) – Responds to an output level change to ‘high’.



- Output Off (Function lo) - Responds to an output level change to 'low'.
- Input On - Responds to an input level change to 'high'.
- Input Off - Responds to an input level change to 'low'.
- Gated On (Not Veto Input) - Gated response to an input level change to 'high'.
- Gated Off (Not Veto Input) - Gated response to an input level change to 'low'.

### **6.3.6 Signal LCC-32H Secondary Messages**

In these examples there are no second or third messages being sent in the sense of our previous products. However additional messages may be sent or responded to if desired by utilizing unused events. For example in order to sequence a yard ladder, to activate a 'Next' turnout whenever the 'first' turnout event is sent, you could use the 'Output State On command' to send an event to the next turnout in the ladder. These additional messages, if enabled, are sent whenever the primary event occurs. Another simple example would be to allow two or even three different messages to be sent whenever a single button is pressed. For example a single button could easily sent events to throw both turnouts in a crossover.

## **7 Signal LCC-32H Logic**

---

In addition to its 8 I/O lines, signal LED drivers, and virtual track code lines, the Signal LCC-32H also includes 32 logic conditionals. This logic is event driven, not state driven. These logic conditionals may be joined into logic blocks that may be used to create signaling logic or other animations such as control of grade crossings. They may also be used to create NX (eNtry Exit) routing through an interlocking.

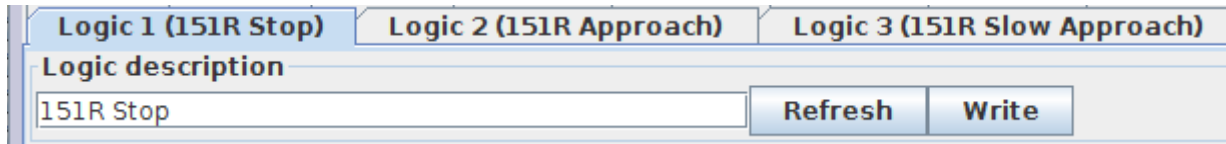
### **7.1 Logic Conditionals**

Each conditional statement in the logic table consists of an identifier and two memory locations (called variables see 7.4) that remember the last command given for any items that they are 'watching'. It also includes a logic operator or function, and a delay timer. Each conditional may generate (produce) up to four events when it evaluates to true, (or false) and optionally after any delay is complete. These 'Variables' are stored in non-volatile memory which allows decision making, even over power failures.

To calculate this logic, the Signal LCC-32H keeps a list of all events that can change the state of any variable. Any time one of these events is seen on the bus, a pass is made through the Logic table. Each variable is checked for a match to the new event, and changed if appropriate. Any change, or optionally just a match itself, will reevaluate each logic conditional containing the variable. If the conditional is found to be 'true' then up to 4 action (producer) events will be sent.

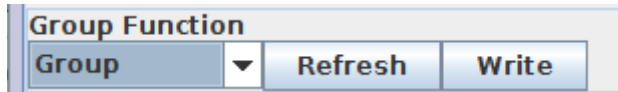


## 7.2 Logic Description



This is the name that you give to the logic conditional. It will appear in the selection tab for this logic conditional to make it easier to find in the future.

## 7.3 Group function



### 7.3.1 Blocked -

This conditional is not used, and will not be evaluated.

### 7.3.2 Group -

This conditional is part of a group (typically a mast) calculation. This means two things:

First, the entire Group (Mast) is checked to see if any conditional prior to the entry point into this group is already true. If so, then the group is skipped over. This means that some previous conditional (e.g. a more restrictive signal rule) is still in effect and must not be overridden.

Second, once any statement at, or following, the entry point generates its action, then the rest of the Group (Mast) may (optionally) be skipped over. For example, once a signal aspect is found to be true then only it should send an aspect (rule) event. No other less restrictive rules may be sent to the same mast.

'Group' logic statements are designed to make the calculation of signal aspects very easy to do. Groups of mast conditionals are always configured with the most restrictive aspects entered first, then in increasing speed order to the least restrictive aspect.

Evaluation effectively begins starting from the first to the last conditional in each group.

Actions (rule messages) only get sent when an evaluation causes a change in a mast's aspect, either to a more restrictive or to a less restrictive one.

Whenever any conditional in a mast group evaluates to 'true', but without any aspect change, then the processing of conditionals is skipped forward past the next 'End Item', and no aspect changes are sent for that mast.

If a 'Mast Group' conditional evaluates to 'true' and causes an aspect change, then its associated delay (if any) is triggered and the evaluation process is ended for that group by skipping forward past the next 'End Item'. Once any delay runs to completion then its associated events are sent. (produced) From one through four events may be sent. This allows for the creation of basic multiple lamp signal head control.

Normally the Signal LCC-32H logic will send a single 'Rule' ('Name') event to then be processed by the Rule to Aspect table to illuminate the required individual lamps.

If no conditional in a 'Group' evaluates as true, then the evaluation simply proceeds to the next logic conditional with no resulting event being produced.

A default rule message may be sent by placing a 'null' (true) conditional as the last item of a 'Mast Group'. It will always be triggered if no previous conditional in the group was found to be true.

Each mast's logic block is treated as if it were a group of statements independent of any other groups, but in fact each group is checked, but if no events match the new event it will simply be passed over.

### 7.3.3 Last (Single) -

This conditional stands alone or is the final entry of a Group.

This group function marks the end of a block of (mast) conditionals containing one or more items to be evaluated in order. Once a conditional in a group has been evaluated as 'true' then processing normally skips ahead past the next 'Last Item'. If any conditional is the last one in a block of Mast conditionals or a stand alone conditional, then it should be identified with this function to allow this to happen.

## 7.4 Variables

The screenshot displays a configuration window for 'Variable #1'. It is organized into several sections:

- Variable #1 Trigger:** A dropdown menu set to 'On Variable Change', with 'Refresh' and 'Write' buttons.
- Variable #1 Source:** A dropdown menu set to 'Use Variable #1's (C) Events', with 'Refresh' and 'Write' buttons.
- Variable #1 Track Speed:** A dropdown menu set to 'Stop', with 'Refresh' and 'Write' buttons.
- EventID (C) Event to set variable #1 true:** A text input field containing '02.01.57.10.00.0A.00.06', followed by 'Refresh', 'Write', 'Copy', 'Paste', and 'Search' buttons.
- Other uses of this Event ID:** A text area containing 'Sensor Manion West Main 1 Active'.
- EventID (C) Event to set variable #1 false:** A text input field containing '02.01.57.10.00.0A.00.07', followed by 'Refresh', 'Write', 'Copy', 'Paste', and 'Search' buttons.

Each variable normally has two events (consumers) associated with it. The first event enables the variable, (true) and the second disables it. (false)

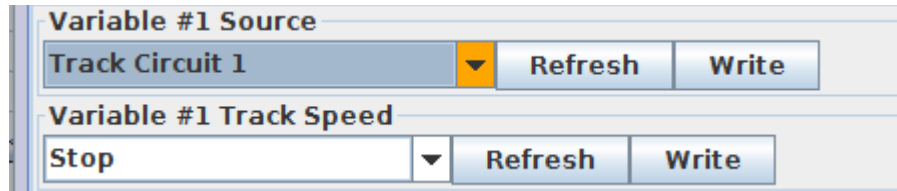
Any time a variable changes state, the evaluation of its conditional may result in conditional action events. Optionally the evaluation of any conditional may be disabled.

Each conditional statement contains one or two variables for identifying the required events in order to do simple signal interlocking or route generation. Any logic statement evaluates to true only if both its function and variables are true. A 'null' function entry is considered to be true if evaluated. (but will not itself trigger

any evaluation) Thus a 'null' entry placed at the end of a mast logic block will always cause a default rule message to be sent if no previous conditional statement within the same group has been evaluated to 'true'.

The state of all variables is remembered internally in order to compare them. An event or change in either variable's state may optionally trigger a new comparison. This dual variable capability makes it easy to calculate a signal aspect or train direction based on turnout position and occupancy.

Optionally a variable may point to a Track Circuit. (one of 8) These Track Circuits each follow the indicated speed sent by another mast. (normally



The screenshot shows a software interface with two sections. The top section is titled 'Variable #1 Source' and contains a dropdown menu with 'Track Circuit 1' selected, followed by 'Refresh' and 'Write' buttons. The bottom section is titled 'Variable #1 Track Speed' and contains a dropdown menu with 'Stop' selected, followed by 'Refresh' and 'Write' buttons.

this will be the next mast in each given route) The track circuit function allows the logic to easily trigger on the speed limit of the next mast.

If a Track Circuit is being used, then the user needs to keep track of which Track Circuit is paired with the next mast in each route in order to properly calculate the signal rules. The Logic Variable entry will simply say something like "Track Circuit 1 - Slow" or "Track Circuit 2 - Stop" with no indication of the remote mast's actual name.

## 7.4.1 Variable Trigger

### 7.4.1.1 On Variable Change -

This conditional is evaluated when an event actually changes the state of the variable. (to true, or to false)

### 7.4.1.2 On Matching Event -

This conditional will be evaluated (or reevaluated) if either event is matched, even if the event does not cause the variable itself to change state. This option is required if a conditional needs to be repeatedly triggered by the same event.

## 7.4.2 Variable Source

### 7.4.2.1 Events -

Normally each Variable is set to true or false by a pair of consumer EventIDs that follow this selection. (Event/s to set variable true/false)

### 7.4.2.2 Track Circuit (RX) -

Optionally a logic variable may be pointed to a 'Track Circuit' Receive (RX) table. To use the track circuit (RX) table you select the following items:

**RX Circuit** - The (RX) table that you are monitoring with this conditional. (Circuit 1 through Circuit 8)

**Track Speed** - The allowed speed approaching the next mast.

Note: In order to utilize a track circuit (RX) table a pointer must first be setup in the 'Track Circuits' Segment. (Section 8)

## 7.5 Variables and 'Not'

Each variable is controlled by a pair of events, one that sets it 'true' and the other which sets it 'false'. To invert the sense of any variable in a logic function, reverse the positions of the two controlling events for that variable. If you have 2 events, 'Crack of Dawn' and 'High Noon' that are used to control the logic 'Its morning', then use 'Crack of Dawn' to control 'true' and 'High Noon' to control 'false'. However if your logic needs to be 'Its **not** morning' then use 'High Noon' to control 'true' and 'Crack of Dawn' to control 'false'.

*Note:* Virtual speed table entries are simply true or false. Any change of 'speed' automatically sets all other 'speed' entries in the same track circuit to false. There are no 'set speed to false' events. Often a change of signal aspect will send the same speed event as was last sent. This repeated event will not cause any change to the virtual speed table setting, nor trigger any recalculation of the signal logic.

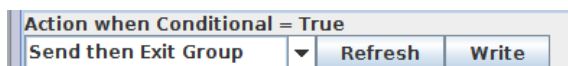
## 7.6 Logic function

The following are the available logic operators:

- V1 AND V2 = 'V1' AND 'V2' is true
- V1 OR V2 = 'V1' OR 'V2' is true
- V1 XOR V2 = 'V1' XOR 'V2' is true (either V1 or V2, but not both)
- V1 & V2 => change 'V1' AND 'V2' is true has changed state (any change triggers a single evaluation of the group)
- V1 OR V2 => change 'V1' OR 'V2' has changed state (any change triggers a single evaluation of the group)
- V1 AND Then V2 => First 'V1' is true AND THEN 'V2' becomes true. This special operator makes it easy to determine train direction. A V2 change to true triggers a single evaluation of the group only if V1 is already true. The V1 state is retained so that further changes to V2 true may re-trigger the evaluation.
- null => true This entry can be used to send a default event anytime the evaluation of a group is triggered, but no other entry evaluates as true. Of course it must always be the last entry in a group if it is used.

## 7.7 Action when Conditional = True

This entry describes the actions taken when a conditional is true.



### 7.7.1 Send then Exit Group

This is the normal function for a Mast Group. If the conditional is true, then any Actions are performed. (EventIDs are produced) The evaluation process then skips to after the next 'Last (Single)' item and continues from there.

## 7.7.2 Send then Evaluate Next

This is the normal function for individual conditionals. If the conditional is true, then any Actions are performed. (EventIDs are produced) The evaluation process then continues normally with the next triggered item. (if any)

## 7.7.3 Send then Send Next

This is the normal function for a Route Group. (yard ladder) If the conditional is true, then any of its Actions are performed. (EventIDs are produced) The next conditional is then triggered and treated as true. (without any evaluation) This process may repeat until a conditional is reached that can exit normally. This may also be used in cases where more than 4 actions are required as a result of a true evaluation. (e.g. eNtry eXit calculations that change many turnout positions)

## 7.7.4 Exit Group

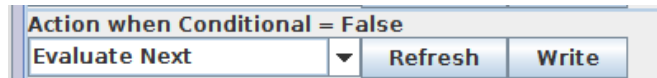
If the conditional is true, then the evaluation process skips to after the next 'Last (Single)' item.

## 7.7.5 Evaluate Next

If the conditional is true, the evaluation process continues with the next triggered item. (if any) No actions are performed.

## 7.8 Action when Conditional = False

This entry describes the actions taken when a conditional is false.



### 7.8.1 Send then Exit Group

If the conditional is false, then any Actions are performed. (EventIDs are produced) The evaluation process then skips to after the next 'Last (Single)' item.

### 7.8.2 Send then Evaluate Next

If the conditional is false, then any Actions are performed. (EventIDs are produced) The evaluation process then continues with the next triggered item. (if any)

### 7.8.3 Send then Send Next

If the conditional is false, then any of its Actions are performed. (EventIDs are produced) The next conditional is then triggered and treated as true. (without any evaluation) This process may repeat until a conditional is reached that can exit normally. This may also be used in cases where more than 4 actions are required as a result of a false evaluation.

### 7.8.4 Exit Group

If the conditional is false, then the evaluation process skips to after the next 'Last (Single)' item.

## 7.8.5 Evaluate Next

This is the normal function for a Mast Group. If the conditional is false, the evaluation process continues with the next triggered item. (if any) No actions are performed.

*Note: Unconditional actions and/or exits may be created by specifying the same actions for both true and false conditionals.*

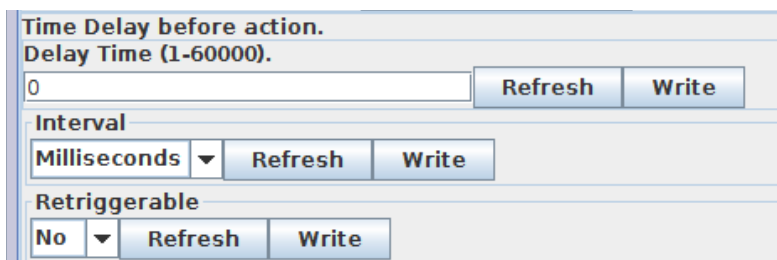
## 7.8.6 Setting up Virtual Code Lines

Use the CDI tools in Segment: Track Circuits to setup virtual links. The EventID for each TX code set will always come from the transmitting node and be entered into the receiving node to avoid accidental reuse of EventID numbers from show to show.

Each node can setup one or more virtual code lines to any other node. For simplicity these virtual links are named 'Circuit 1', 'Circuit 2', etc. It is incumbent upon the user to keep track of which 'Coded' virtual links are created between nodes. Be sure to record which circuit (1-32) is used for each side of the virtual links if you have not standardized these connections. There is no need to use the same 'circuit' number on both sides of any virtual coded track circuits, and in fact they will not normally be matching. Normally these virtual links will follow along with the rails in each block, but there is no actual requirement that they do so.

There is also no requirement that only a single receiver may listen to a given transmitter. The speed information sent over a track circuit might be used both by a signal mast's logic, and by a control panel monitoring the same information.

## 7.9 Delay



The delay time before any action may be set with a resolution of minutes, seconds, or milliseconds. (0-60,000) That gives you resolution enough to create real time day/night lighting controls for your buildings. (or layout room) The 'Re-trigger' option

controls if a running time delay gets reset when it is re-triggered before completion. If so, then the time delay period starts over again. If not, then the re-trigger event is ignored, and the timer continues to run to its original completion. To 'kill' the output of a timer already in progress disable a timer's output by setting the triggering function to no longer be 'true'.

## 7.10 Actions (Producers)

Any logic conditional may generate up to four events when it is true. (or false) Each of these four events may be; ignored with 'none', sent 'Immediately', or sent 'After delay'.

When the logic is being used to calculate signal aspects, then each event produced will normally be used to control the appropriate 'Rule to Aspect' entries in the signal driver section.

## 8 Masts

The primary difference between a simple I/O line and a Signal driver is that there is a way for the user to specify exactly which lights are lit on each signal mast for each signal rule that is called for.

NORAC Definitions:

- Rule/Name - The number and name given to each signal aspect.
- Signal Indication - The required action conveyed by the aspect of a signal. (e.g. Stop, Proceed, Proceed prepared to stop at next signal. Trains exceeding 40 MPH must begin reduction to 40 MPH as soon as engine passes signal displaying approach, etc.)
- Signal Aspect - The signal appearance, which conveys an indication as viewed either (1) from the direction of an approaching train, or (2) on the cab signal display unit in the engine control compartment. (e.g. Red over Red, Green over Red, Flashing Yellow, etc.)
- Signal Mast - The vertical pole on which the signal head(s) or arm(s) is/are mounted. A signal's aspect always includes all heads that are mounted on a single signal mast, not just a single head.
  - One exception to this might be some systems where separate signals are provided on the same mast for switching moves and primary movements. In that case treat the two systems as separate masts in the node even though they are mounted on the same physical mast on the layout.

### 8.1 Mast Processing

To configure each decoder driver board requires the following information: (shown item by item)

Signal Heads										
Assign rules and lamps to a Signal Head.										
Head 22	Head 23	Head 24	Head 25	Head 26	Head 27	Head 28	Head 29	Head 30	Head 31	Head 32
Head 11	Head 12	Head 13	Head 14	Head 15	Head 16	Head 17	Head 18	Head 19	Head 20	Head 21
Head 1 (51R)	Head 2 (52R)	Head 3 (51L)	Head 4 (52L)	Head 5 (252R)	Head 6 (2)	Head 7	Head 8	Head 9	Head 1	Head 1

Select a head to configure. Note that a dual decoder/driver board is configured and counted as two separate heads and you need to account for this in your numbering. You also need separately configure each head's "Appearance" if there are multiple heads on the same mast.



Signal Processing

Enabled

A single head mast, or the top head in a multi-head mast will be "Enabled". Any additional heads in a multi-head mast will be set to "Linked to Previous".

Position

Head position in string (1...32; 0 = skip)

2

Normally the head position will match the physical position in the string. (remember to count dual boards as two positions) However to allow adding one or more decoder/driver units into a string after the fact without reprogramming every head following it, this option allows you to shift the physical position of each configuration entry if required. Note that you will need to change all head position entries following the new added decoder/driver unit to account for their new positions in the string. Note also that you may need to swap head positions in a multi head mast if the physical wiring places the heads in the incorrect order.

Signal Description

52R

This names the tab.

Link Address

(P) Track Circuit Link Address. Copy and Paste into linked Track Circuit (Limited Writeable)

02.01.57.11.00.00.01.A0

This is where you get the pointer for track circuits.

Lamp Fade

None

My example is a dwarf searchlight signal, so there is no "Incandescent Fade".

Lamp 1 Intensity (0-255)

128

Lamp 2 Intensity (0-255)

128

Lamp 3 Intensity (0-255)

128

This section determines the individual LED intensities for the three lamps. Only change these settings if the brightness is not well matched between the LEDs on your signal, or if it is too bright or dim. Also note that if you are driving RGB LEDs you may need to adjust the individual red and green lamp brightness to get a



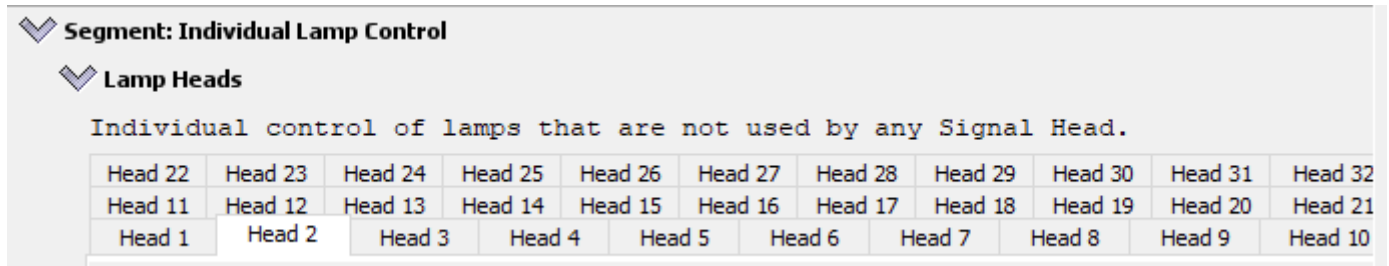
decent yellow color. This board is designed to control RR signals, so you do not have the option to create multiple random colors per aspect. This is the only place that you can adjust the color brightness for mixed color options. I.e. adjusting for a good yellow will effect the steady red and steady green output brightness as well.

Select each rule for this Mast. Note that for a multi head mast the first "Enabled" head's rules are used for ALL linked heads. Therefore each mast is limited to just the first head's 16 rules. (aspects) For "Linked to Previous" heads you will still need to select each rule when configuring the appearance of individual heads, but the rest of the content in these entries will be ignored.

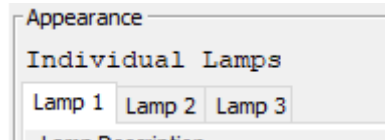
The "Rule Name" and "Track Speed" entries are related to the track circuits. The "Set Aspect" EventID is the EventID that will select this aspect on the mast. "Aspect Set" and "Aspect Cleared" EventIDs are the feedback information from the mast. These EventIDs can be used for signal repeaters or for CTC indication messages.

Select each of the 3 lamp outputs on each decoder/driver board and set the desired output for this selected rule. This section is different from our previous signal drivers because this board is capable of driving three color (RGB) LEDs such as the Ada Fruit NeoPixels that may require multiple outputs to be simultaneously driven for some colors. Normally each rule will only enable a single lamp and effect. In this example Lamp 1 is driving the 'Red' signal lamp so for "Rule 1"

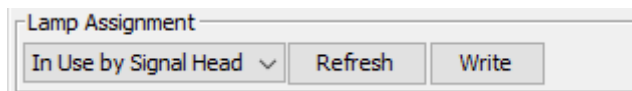
(Stop) it is set to "Steady" and the other two lamps are set to "Not Used".



Check the segment "Individual Lamp Control" and select each head in turn.



Step through each lamp in turn.



Make sure that each individual lamp has "In Use by Signal Head" selected. 'Individual Lamp Control' is normally only used if you are using the Signal LCC-32H board to control physical panel lamps rather than signal masts. In that case you would select "Individual Control" for each lamp (of the 96) and use its EventID pair to turn it on or off. Note that you have individual brightness control for each, so you could assign several lamps to the same EventIDs and use RGB to create any individual color that you want for building lighting or similar. Also note that because there are only two EventIDs available for control the JMRI CDI automatically presents you with "Make Sensor" and "Make Turnout" options to easily enter data into a JMRI table.

•  
•

**DRAFT**

**version**

# from Here.

Information may be incomplete and/or in error as it has been copied from other similar nodes and uncorrected for the specifics of the Signal LCC-32H design.

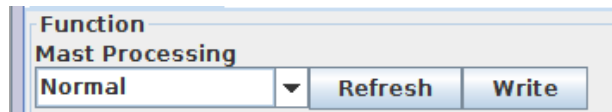
## 8.1 Mast Processing

### 8.1.1 Unused

This signal Mast entry is not used.

### 8.1.2 Normal

This signal Mast can show up to 8 rules.



Function  
Mast Processing  
Normal ▼ Refresh Write

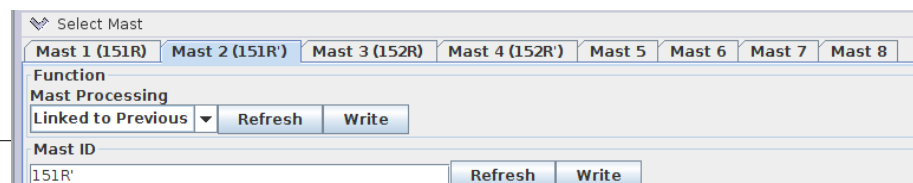
### 8.1.3 Link to Previous

If an individual mast needs to display more than 8 rules, then the *next* mast entry may be linked back to this entry to allow a total of 16 rules. This is done by setting the next mast's Mast Processing value to '*Linked to Previous*'. Normally any single mast will not exceed 16 possible rules, but the '*Linked to Previous*' entry may be used more than once to create signals with very many rules. Note: The 'Link' controls the automatic cancellation of one indication by another such that any aspect selected from one mast will automatically cancel any other aspect from any linked mast. It also directs any specified speed information from either (or all) mast(s) to the 'Track Circuit' of the first mast in a linked group.

## 8.2 Mast ID

This entry is the user's definition given to this

8 Masts



Select Mast  
Mast 1 (151R) Mast 2 (151R) Mast 3 (152R) Mast 4 (152R) Mast 5 Mast 6 Mast 7 Mast 8  
Function  
Mast Processing  
Linked to Previous ▼ Refresh Write  
Mast ID  
151R Refresh Write

signal mast, either by mile marker, control panel number, Station Name, Etc. It will be listed on the selection tab for this mast, so short and succinct is best. In this example the ' after 151R indicates that the Mast 2 entry is an extension of the 151R rules found in Mast 1, and that Mast 4 is an extension of the 152R rules found in Mast 3.

### 8.3 (P) Track Circuit Link Address

This (read only) address is used by the virtual track circuit to send speed information to any other node that requires it. Copy this value into the associated track circuit speed table's '(C) Remote Mast Up Link Address' field. See: section 9 for more information.

### 8.4 Lamp Fade

This entry controls if the lamps should simulate ballasted incandescent lamps or change immediately from one color to the next. The options are Incandescent, or None.

### 8.5 Rules

This entry selects up to 8 indications to be shown by this mast. (16 or more if the 'Link to Previous' option is used)

The screenshot shows a 'Rules' configuration window with tabs for Rule 1 through Rule 8. Rule 1 is selected and contains the following fields and buttons:

- Name:** 29-Clear (dropdown menu), Refresh, Write
- Track Speed (on approach to signal):** Clear/Proceed (dropdown menu), Refresh, Write
- EventID (C) Event to Set Aspect. Note: Aspects are cleared automatically by the logic.**
  - 02.01.57.10.00.08.01.C8 (input field), Refresh, Write, Copy, Paste, Search
  - Other uses of this Event ID: Turnout 151R Clear Thrown, Cressman West.LOGIC.Logic event processing(4,151R Clear).A trigger or change will generate the following events.(1)
- EventID (P) Send this event when the Aspect is set.**
  - 02.01.57.10.00.08.01.C9 (input field), Refresh, Write, Copy, Paste, Search
- EventID (P) Send this event when the Aspect clears.**
  - 02.01.57.10.00.08.01.CA (input field), Refresh, Write, Copy, Paste, Search

**Name (indication)** This selection is for information only and lists common rule book names for signal indications. These may not exactly match your railroad's rule book names. If not, simply choose something similar.

**Track Speed** This selection chooses the rule book track speed allowed upon approach to this signal. This information becomes available to the preceding signal's logic over the virtual track circuit. (RX table) *Note:* These speeds are simply coded indications, and may be redefined in your signal rules. There is no actual speed associated with them.

**(C) Event to Set Aspect** This is the EventID that selects this rule.

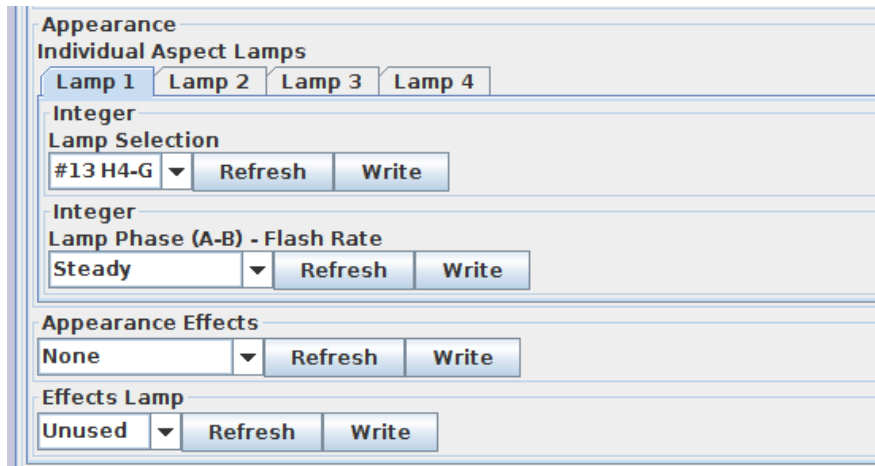
**(P) Send this event when the Aspect is set** This is the EventID that will be sent when this rule is selected.

**(P) Send this event when the Aspect clears** This is the EventID that will be sent when when this rule is cleared. (because another rule was set)

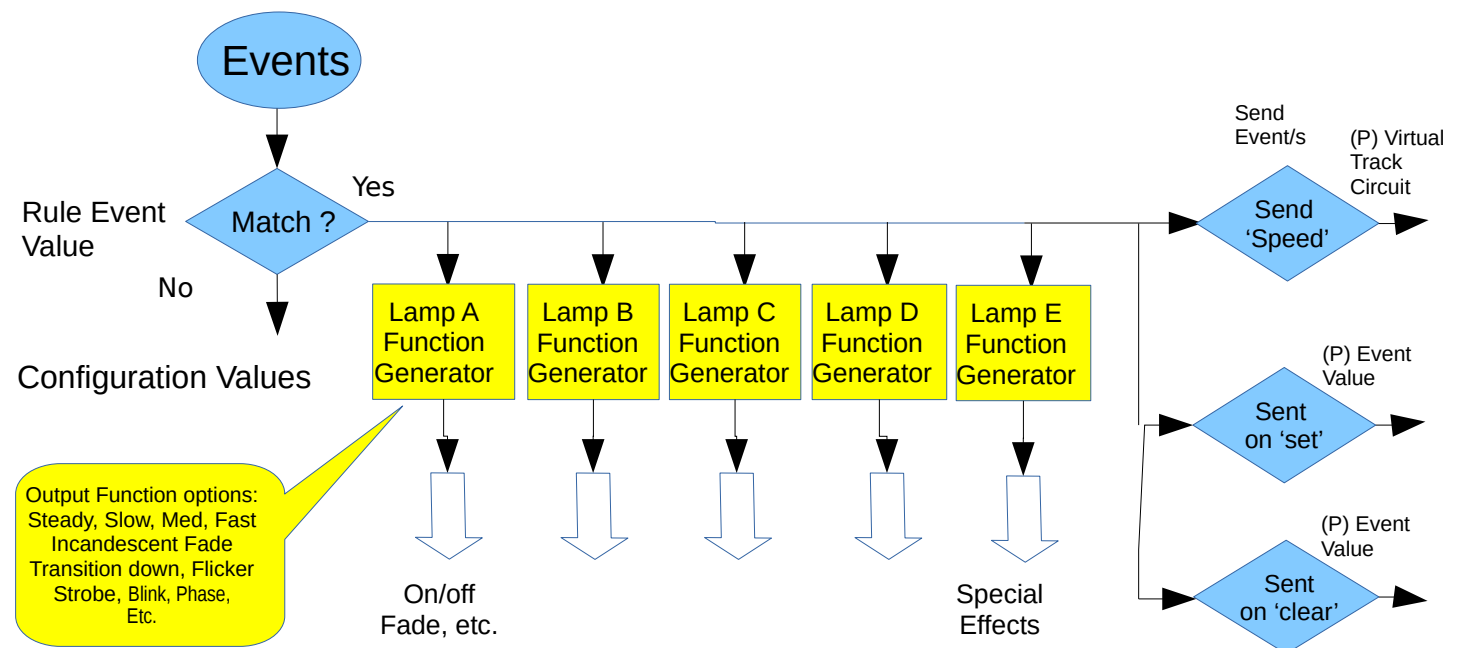
## 8.6 Aspect to Appearance

The LCC sends Signal Aspect (rule) events over the bus. These Signal Aspects are based on the rule book of the rail road being modelled, and originate in the vital signalling logic, which may be located either in internal node based logic, or in an external program such as JMRI.

The 'Aspect to Appearance' Table retrieves these events from the bus, (or internally via direct connection) groups them together into Masts, and determines what lamps (up to 4 per aspect) and lamp effects (1 per aspect) are required to correctly display each Signal Aspect to the user. This table is the key to easily creating any aspect required by any signal rules from any place in the world.



Additionally the Aspect to Appearance table entry also specifies what speed is allowed on approach to the signal. This 'Speed Indication' information is sent via the virtual track circuit to a matching table located in the preceding signal's logic where it may be used to calculate signal rules for that signal.



### **8.6.1 Individual Aspect Lamps**

Each signal Aspect may illuminate up to 4 lamps plus it may use a 5<sup>th</sup> lamp (or one of the first 4) to create a special effect. One common special effect would be to create the red flash seen as a searchlight signal mechanism passes between yellow and green, or green and yellow.

### **8.6.2 Lamp Selection**

Each of 4 lamps may be specified as one of the 16 outputs, or simply left 'Unused'. The lamps are identified as H1-G, H1-Y, H1-R, H1-L, H2-G, etc. but of course any lamp output may be used on any mast as required. The labels simply match the documentation for the board as found on pages 6-7 of this document.

### **8.6.3 Lamp Phase (A-B) - Flash Rate**

This may be used to specify basic functionality. Two phases are specified to allow for simple crossing gate flashers or similar alternating action. Three different common flash rates are offered. This makes it simple to create a realistic 'Flashing Yellow' lamp for 'Advance Approach', or 'Flashing Red' for 'Restricting', without the need for the signal logic to do the flashing.

### **8.6.4 Appearance Effects**

A few common special effects are offered. 'Transition down' is used to automate the effect seen on green CPL signals that causes the yellow lamps to light briefly before the red lights illuminate. Another common effect called 'H2' would be used to flash the red briefly as a mechanical search light moves between its yellow and green positions. The 'Strobe' effect can be used to simulate lamps that repeatedly flash briefly. (e.g. the white strobe often seen with a red highway stop signal)

### **8.6.5 Effects Lamp**

This entry selects the lamp to be used for a special effect. For example, during the H2 searchlight effect the red lamp is flashed briefly. This gives the distinctive red flash between aspects. You could select any lamp to create this effect, but normally it will be the same red lamp used in the mast for 'Stop'.

## **9 Track Circuits**

---

The prototype railroads have the need to send signal speed information from one signal to its previous signal in order to allow the previous signal to calculate the proper aspects to display. These calculations are done locally, (called vital logic)

not at a dispatcher's office, nor by some central computer. This speed information can be sent over local wires from one signal to the next, but that means lots of infrastructure to maintain. Fortunately there is always one pair of conductors that need power on them for detection circuits and that automatically go to the correct place. That is the rails themselves. Even from the earliest semaphore days some basic speed information was passed over the rails simply by switching the polarity of the battery feeding the DC track circuit being fed from one end of a block to the other. This polarity change was used to change the signal from Clear to Approach.

As signaling became more complex and speed signaling was introduced, there was a need to pass more information over the same two rails. Genrakode® and Electrocode® are two brands of equipment that do this by sending a series of pulses rather than just straight DC from one end of the block to the other. These circuits are normally bi-directional, so the transmit and receive circuits switch from one direction to the other every few seconds. We are using the bus, not the rails, for data, so this direction switching and its attendant delays is not required.

## 9.1 Simulating a Code Line with Events

To send this speed information from signal to signal by configuring individual EventIDs is cumbersome because each link needs to be specifically configured to use different EventID numbers for each speed. That makes it difficult or impossible to determine ahead of time what all the EventID numbers will be for a modular setup.

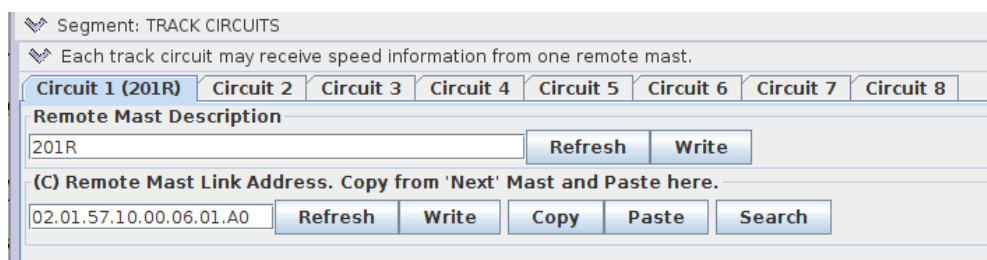
One solution to this dilemma is to use events in a way similar to those used in the actual coded track circuits. We do this by using a single address and fixed offsets for each speed value. Any speed change sends an EventID with a fixed offset from a known value and it gets stored in the receiving (RX) table. We then allow nodes to learn these group addresses of internally generated events simply by a single cut/paste operation in the CDI.

These internal event groups are then made available for use in Logic. To link up these virtual connections at a modular meet the meet coordinator would arrange module panels appropriately, then do a simple cut/paste operation at each module boundary to link the speed tables appropriately.

Because the logic conditionals may optionally refer to the RX table, not the actual EventIDs, there are no last minute changes required to the logic. Also patterns of logic can be repeated from node to node once a template is available for the rules of your railroad.

## 9.2 Linking Virtual Code Lines (Track Circuits)

The EventID for the TX code set will always come from the transmitting node and be entered into the receiving node to avoid accidental reuse of EventID numbers from show to show. See section 8.3 for the sending side.



Each node can setup one or more virtual code lines to any other node. For simplicity these



virtual links are named 'Circuit 1', 'Circuit 2', etc. It is incumbent upon the user to keep track of which 'Track Circuit' links are created between nodes. If you have not standardized these connections, be sure to record which mast (1-8) is used for each side of the virtual links. There is no need to use the same 'Track Circuit' number on both sides of any virtual coded track circuits, and in fact they will not normally be matching. Normally these virtual links will follow along with the rails, but there is no actual requirement that they do so.

Also note that the 'Speeds' are simply names given to the different events for simplicity. If your rules use different names, just adjust appropriately.

### 9.3 Prototype Code Line

In our solution to this problem we follow the prototype design with a few changes. When each prototype code is sent onto the rails there are four data items simultaneously sent in each packet. They are the 'Start' bit, (1) the 'Speed' or 'Indication' code, (2, 3, 4, 6, 7, 8, or 9), the '5' code, (5) and the 'M' code. (M) In order to make this solution more 'EventID' friendly we ignore the '5' and 'M' codes, and replace the '5' code with 'Absolute Stop'.

*Note:* the prototype can only send indications via an unoccupied block, because the train itself is shorting the rails going toward the signal ahead when it is occupied. On the model we do not have that restriction. Therefore we can send either 'Stop' (5) or 'Tumble down' (6) as an indication. This simplifies the coding of APB signals.

To limit the number of indication EventIDs to 8 we have omitted the 'M' code as being unnecessary for our purposes. Further, in the receive tables we automatically cancel any previous speed indication whenever a different one is received. This eliminates the need for extra events to cancel speeds.

### 9.4 Speed indications

One of the items in each 'Name to Aspect' table entry is a speed indication for each aspect. The 'Speed Indication' is the rule book speed that a train must obey as it approaches the signal. In order for a signal's logic to correctly calculate its aspect, it must know the allowable speed upon approaching the next signal. (plus other factors) This is the 'Speed Indication' and it always refers to the speed allowed when entering the block past the one that you are about to enter. I.e. it is the speed allowed at the next signal.

We support the following speed indications as being the most useful for model RR use:

- Stop

TABLE 5-1 CODE RATE AND ASPECT

CODE RATE	ASPECT
7	Clear
4	Advance Approach
3	Approach Limited
8	Approach Medium
2	Approach
9	Approach Slow
6	Accelerated Tumble Down
5	Non-Vital code indicating track occupancy, or a hand-throw switch in the block out of normal correspondence
M	Non-Vital code indicating power off in the block, or a lamp out condition in the block. Power Off will indicate from the east end CP, lamp out from the west end CP

- Restricting/Tumble Down
- Slow
- Medium
- Limited
- Approach
- Approach Medium
- Clear/Proceed

A typical rule book entry might be: “282 - Approach Medium - Proceed approaching the next signal at Medium Speed.” For this aspect you would set the speed entry for this mast to ‘Clear/Proceed’. The fact that you need to slow down as you approach the next signal is unimportant when determining the speed that you must approach this signal.

Another rule book entry might be: “285 - Medium Approach - Proceed prepared to stop at the next signal. Trains exceeding Medium Speed must at once reduce to that speed.” For this aspect you could set the appropriate speed entry to either ‘Medium’ or to ‘Approach’ depending on your signal system. Your logic could then calculate ‘282’ - Approach Medium or else ‘282a’ - Advance Approach.

The sample rule worksheet in section 18.1 shows the normal allowed speed for passing a mast with the most common aspects. If these don’t cover the needs of your prototype you may re-purpose them as required.

Note: Any unused entries may also be re-purposed as you require. These are simply tools to aid in signal aspect calculations. They do not actually control the train’s speed. That is left up to the engine driver.

## 9.5 ABS and APB Signal examples

For examples of using logic to control signals see the Signal LCC-32H manual. (oops, that's this manual) The Tower LCC logic may be used to expand a Signal LCC-32H logic table if more statements are required.

Logic - Description	Group	T	Var #1	Funct	T	Var #2	True-False		Actions	Comment
1 -										
2 -										
3 -										
4 -										
5 -										
6 -										
7 -										
8 -										
9 -										
10 -										
11 -										
12 -										
13 -										
14 -										
15 -										
16 -										
17 -										
18 -										
19 -										
20 -										
21 -										
22 -										
23 -										
24 -										
25 -										
26 -										
27 -										
28 -										
29 -										
30 -										
31 -										
32 -										

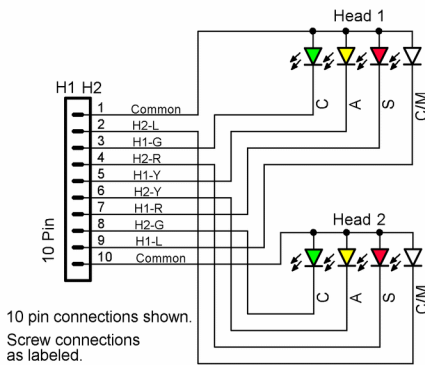
## 9.6 Signal Lamp Wiring

The following examples show how to wire the common signal types and give some simple example Lamp Table entries used to control them.

### 9.6.1 Color Light Signal Connections

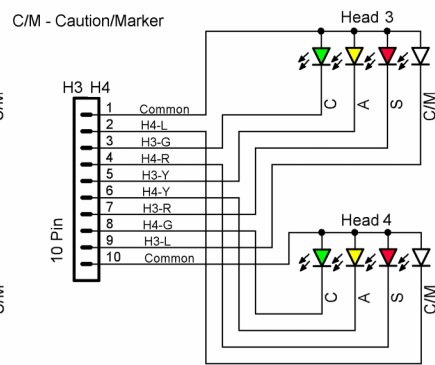
The default Color Light signal wiring for the Signal LCC-32H is shown here. The head names and lamp numbering are just for reference to the table entries. The actual lamp colors and head numbers may be changed to match your specific prototypes requirements.

Color Light Signals using LEDs driven with a Signal LCC.



10 pin connections shown. Screw connections as labeled.

Unused lines may be used as grade crossing flashers, radio tower lamps, commercial sign lighting, etc.



For example, when using 3 color signals the Signal LCC-32H will actually control 5 heads by using the unused Lunar connections to drive the 5th head. Alternately these unused connections may be used for driving Tortoise switch machines, crossing gate flashers, radio tower strobes, advertising signs, or other layout lighting effects.

Signal LCC-32H Signal Head Wiring for Color Light Signals.

Head	Lamp	Signal
Head 1	C	Common
	A	H1-R
	S	H1-L
	C/M	None
Head 2	C	Common
	A	H1-Y
	S	H1-L
	C/M	None
Head 3	C	Common
	A	H3-R
	S	H3-L
	C/M	None
Head 4	C	Common
	A	H3-Y
	S	H3-L
	C/M	None



Aspect #1 (Stop) sets 'Lamp A' to 'H1-R Steady' which is connected to the red LED. This incandescent prototype signal fades from one color to the next. The Signal LCC-32H automatically does this effect if 'Incandescent' is selected for the mast. The 3 unused lamp entries are set to 'None'.



Aspect #2 (Approach) sets 'Lamp A' to 'H1-Y' for the yellow indication. The 3 unused lamp entries are set to 'None'.



Aspect #3 (Clear) sets 'Lamp A' to 'H1-G' for the green indication. The 3 unused lamp entries are set to 'None'.

Aspect #4 (Dark) in this example, is dark so each of its 4 lamp entries are set to 'None'.

'Incandescent' is set for each effect to simulate the slow brightening and dimming of incandescent bulbs with ballast resistors.

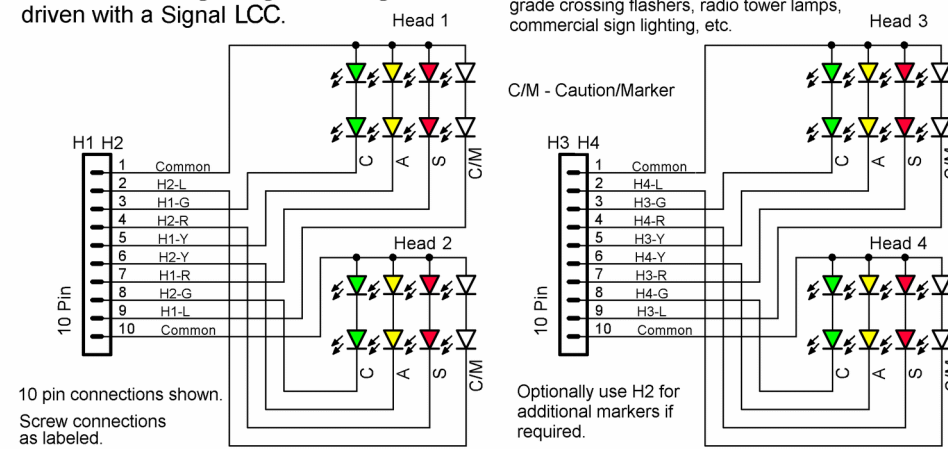
**Tip**

Remember, it requires less command traffic to add flashing aspects in the tables, than if you use 'Dark' and expect JMRI to turn the lamps on and off.

### 9.6.2 Color Position Light Signal Connections

For this example we are using turnout commands for aspect control. A user modeling the B&O railroad's CPL signals might use the first three lamps of Head H1 for the central aspect's color position lights. (two LEDs wired in series or parallel for each aspect color) The fourth lamp plus the 4 lamps of Head H2 could then be used to control up to 5 of the six possible B&O marker lamps.

Color Position Light Signals using LEDs driven with a Signal LCC.



Another B&O mast might add a lunar for Caution and require all 4 lamps of Head H1 for the central aspect. Four markers could then use the 4 lamps from Head H2.

Yet another mast might only need 4 lamps, 3 for the CPL aspect and one used as a single marker. Both the central aspect and markers

#### Signal LCC-32H Signal Head Wiring for Color Position Lights

may all come from one 'head' because there are no restrictions against lighting more than one lamp at a time on the same 'head' or 'mast'. In truth the 'Head' designations are simply a wiring convention carried over from earlier generations of hardware that could only light one lamp per head at a time.

Note: These LEDs are connected 'ca' (Common Anode) to match the LEDs on the mast.

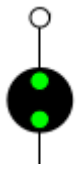


Aspect #1 (Stop) lights 'H1-R Steady' which is connected to the Red LED. This B&O signal lights the approach aspect briefly when dropping from clear to stop. The Signal LCC-32H automatically does this effect if 'Tumble Down' is selected. First the lamp selected as 'Lamp B' is activated for 500ms, then it extinguishes and any lamps selected as 'Lamp A', 'Lamp C', and 'Lamp D' are lighted. The 'Lamp B' position in this case is 'H1-Y'. The 2 unused lamp entries are set to 'None'.



Aspect #2 (Approach) sets 'Lamp A' to 'H1-Y' for the yellow indication and 'Lamp C' to 'H1-L' for the upper center marker. The 2 unused lamp entries are set to 'None'.





Aspect #3 (Clear) is similar to Aspect #1 in that it just blinks the yellow LED 'H1-Y Steady' in 'Lamp B' with the 'Tumble Down' effect, followed by lighting the green LED 'H1-G Steady' in 'Lamp A'. 'Lamp C' set to 'H1-L' for the upper center marker. The unused 'Lamp D' entry is set to 'None'.

Aspect #4 (Dark) in this example, is dark so each of its 4 lamp entries are set to 'None'.

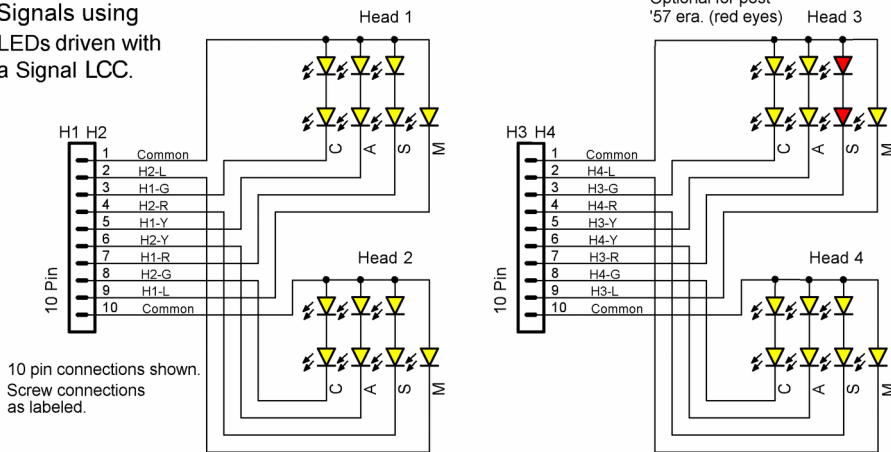
'Incandescent' is set for this mast to simulate the slow brightening and dimming of incandescent bulbs with ballast resistors. The 'Tumble Down' effect automatically fades its lamps.

**Note**

Do not allow a signal using the 'Tumble Down' effect to be flashed by software to show Cab Speed. It will blink the yellow before each green is displayed in an un-prototypical manner. Instead add an additional aspect for 'Cab Speed' that uses the Signal LCC-32H itself to flash the green lamp.

**9.6.3 Position Light Signal Connections**

Position Light Signals using LEDs driven with a Signal LCC.



This shows the simplest wiring option for position light signals. The advantage of serial connection is that all LEDs receive the same current and will show with equal brightness. LEDs are current, not voltage operated devices, and even units from the same batch will often show with different brightnesses when connected in parallel.

*Position Light Signal Wiring*

**Tip**

The wiring for the Conrail Amtrak version of these heads connects two horizontal red LEDs in place of the horizontal yellow LEDs in series on the upper head. (horizontal LEDs do not appear on the lower head) If the red LEDs are too bright simply dim them with the brightness control.

Unfortunately many commercial signal heads use parallel connections because some commonly available signal head drivers are low voltage devices that are unable to drive a signal with 2 yellow LEDs in series. Driving such parallel LEDs is not a problem for the Signal LCC-32H, but it may require that you use different brightness options.

1	Common	1	Common
2	H2-L	2	H4-L
3	H1-G	3	H3-G
4	H2-R	4	H4-R
5	H1-Y	5	H3-Y
6	H2-Y	6	H4-Y
7	H1-R	7	H3-R
8	H2-G	8	H4-G
9	H1-L	9	H3-L
10	Common	10	Common

In the following examples we will use Head 1 as the upper head, and Head 2 as the lower head in a dual head mast.



Rule 281b Aspect #17 (27-Appr-Limited) lights 'H1-Y Steady' which is connected to the H1 45° LEDs, 'H1-L' connected to the upper marker, 'H2-G A Slow' which is connected to the H2 vertical LEDs, and 'H2-L A Slow' connected to the lower marker. This flashes the H2 LEDs.



Rule 281c Aspect #18 (20-Limited-Clear) lights 'H1-R Steady' which is connected to the H1 horizontal LEDs, 'H1-L' connected to the upper marker, 'H2-G A Slow' which is connected to the H2 vertical LEDs, and 'H2-L A Slow' connected to the lower marker. This flashes the H2 LEDs.

**More aspects and examples needed here**



Rule 282 Aspect #19 (25-Appr-Medium) lights 'H1-Y Steady' which is connected to the H1 45° LEDs, and 'H2-G Steady' which is connected to the H2 vertical LEDs.



Rule 283 Aspect #20 (15-Medium-Clear) lights 'H1-Y Steady' which is connected to the H1 45° LEDs, and 'H2-G Steady' which is connected to the H2 vertical LEDs.



Rule 284 Aspect #21 (23-Appr-Slow) lights 'H1-Y Steady' which is connected to the H1 45° LEDs, and 'H2-Y Steady' which is connected to the H2 45° LEDs.



Rule 286 Aspect #22 (11-Medium-Appr) lights 'H1-R Steady' which is connected to the H1 horizontal LEDs, and 'H2-Y A Medium' which is connected to the H2 45° LEDs. It flashes the H2 LEDs.



Rule 288 Aspect #23 (6-Slow-Appr) lights 'H1-R Steady' which is connected to the H1 horizontal LEDs, and 'H2-Y Steady' which is connected to the H2 45° LEDs.



Rule 290 Aspect #22 (4-Restricting) lights 'H1-R Steady' which is connected to the H1 horizontal LEDs, and 'H2-L Steady' which is connected to the H2 135° LEDs.

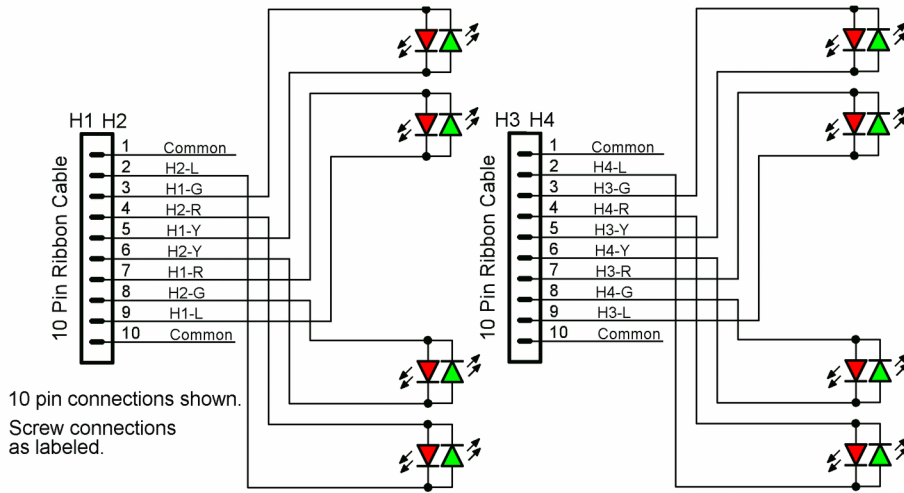


We can see from this example that the Signal LCC-32H can easily control all 8 possible aspects of the PRR dual head masts sent as 8 different NMRA DCC signal aspects for Mast number 17.

### 9.6.4 Searchlight Light Signal Connections (bi-polar)

This CDI shows the settings required for driving bi-polar LEDs to show 3 colors plus the red flash seen when changing between yellow and green mechanical searchlight signals.

Searchlight Signals using bi-polar LEDs driven with a Signal LCC.

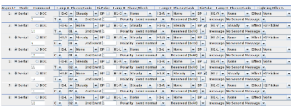


You can see from this wiring diagram that H1-G and H1-Y connect to the first LED. When 'H1-Y' is low and 'H1-G' is high then the red LED lights. When 'H1-Y' is high and 'H1-G' is low then the green LED lights. To make a yellow indication both LEDs need to be lighted by rapidly alternating between the two polarities. If one

color or the other predominates, then use the Signal LCC-32H Brightness pane to adjust as required. You will need to resend the yellow aspect command after each brightness change to reload the PWM (pulse width modulation) registers to their new values.

Note: These LEDs are connected as if they are 'cc' (Common Cathode). Be sure to power cycle the board after changing the 'ca cc' jumper to 'cc'. Connecting as if 'ca' will reverse the red and green colors.

- Aspect #1 (Stop) lights 'H1-G Steady' which is connected to the Red half of the LED. The 3 unused lamp entries are set to 'None'.



- Aspect #2 (Approach) is more complex due to the red flicker effect when a mechanical searchlight changes between green and yellow indications. The Signal LCC-32H automatically does this effect if the 'H2 Flicker' effect is selected. First the lamp selected as 'Lamp B' is activated for 100ms, then it extinguishes and any lamps selected as 'Lamp A', 'Lamp C', and 'Lamp D' are lighted. 'H1-G Steady' is connected to the red LED, so we enter it as 'Lamp B'. Then we select 'Lamp C' as 'H1-G Steady' and 'Lamp D' as 'H1-Y Steady' to light both the red and the green LEDs to yield yellow as the final color. We use 'Lamp C' and 'Lamp D' because we must also check a 'BP' (Bi-Polar) box for the pair to cause the required fast reversal to light them both. 'Lamp A' is set to 'None'.

Aspect #3 (Clear) is simpler in that it just flashes the red LED 'H1-G Steady' in 'Lamp B' followed by the green LED 'H1-Y Steady' in 'Lamp A'. The 2 unused lamp entries are set to 'None'.

Aspect #4 (Dark) in this example, is dark so each of its 4 lamp entries are set to 'None'. The 'Dark' aspect allows the control software to be able to 'Approach Control' the signal by extinguishing the lamp.

**Note**

Do not allow a signal using the 'H2 Flicker' effect to be flashed by software to show Advance Approach. It will flicker the red before each yellow is displayed in an un-prototypical manner. Instead add an additional aspect for 'Advance Approach' that uses the Signal LCC-32H itself to flash the signal.

The second LED is controlled in a like manner using aspects #5 - #8. The difference being that the second LED is connected to 'H1-R' and 'H1-L', and output lines --- and --- are used to control the aspects.

**Tip**

When using Bi-Polar LEDs it is difficult to get a good yellow signal. There are several reasons for this. First, because of the way that these LEDs are manufactured, the green and red dies are not close to one another. This permits both colors to be seen distinctly. Second, as the viewer changes position he will see more or less of each color due to the lens of the LED. To minimize this color shift be sure to locate the LED leads vertically to each other. Finally, neither the red nor the green colors are very close to an actual signal's colors. For best color fidelity consider using the RR-CirKits SS-RGY instead.

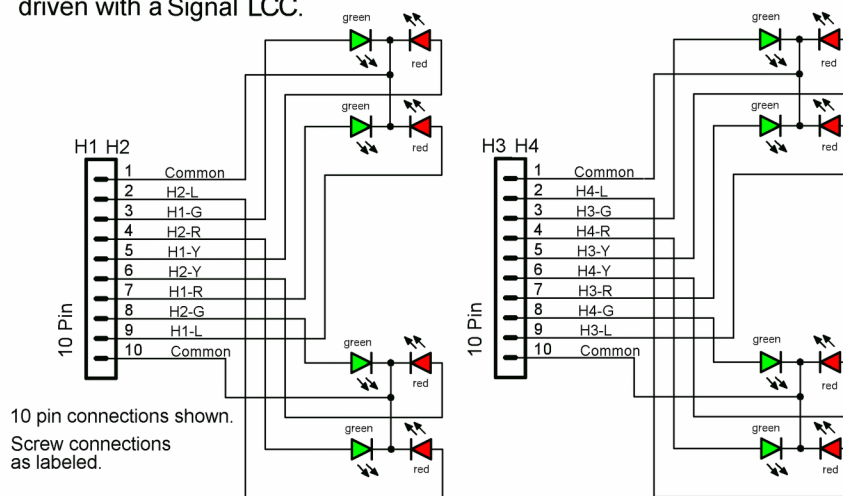
### 9.6.5 Searchlight Light Signal Connections (Dual Color)

Note: These LEDs are connected 'cc' (Common Cathode).

For this example we are using turnout commands for aspect control. The following DecoderPro file shows the settings required for driving dual color LEDs to show 3 colors plus the red flash seen when changing between yellow and green mechanical searchlight signals.

To make a yellow indication both LEDs need to be lighted.

Searchlight Signals using two color LEDs driven with a Signal LCC.



Aspect #1 (Stop) lights 'H1-G Steady' which is connected to the Red half of the LED. The 3 unused lamp entries are set to 'None'.

Aspect #2 (Approach) is more complex due to the red flicker effect when a mechanical searchlight changes between green and yellow indications. The Signal LCC-32H automatically does this effect if 'H2 Flicker' is

Signal LCC-32H Signal Wiring for two color (3 lead R/G) LEDs

selected. First the lamp selected as 'Lamp B' is activated for 100ms, then it extinguishes and any lamps selected as 'Lamp A', 'Lamp C', and 'Lamp D' are lighted. 'H1-G Steady' is connected to the red LED, so we enter it as 'Lamp B'. Then we select 'Lamp A' as 'H1-G Steady' and 'Lamp C' as 'H1-Y Steady' to light both the red and the green LEDs to yield yellow as the final color. 'Lamp D' is set to 'None'.

Aspect #	Color	Lamp A	Lamp B	Lamp C	Lamp D	Signal
1	Red	H1-G Steady	H1-G Steady	H1-G Steady	H1-G Steady	H1-G Steady
2	Green	H1-Y Steady	H1-Y Steady	H1-Y Steady	H1-Y Steady	H1-Y Steady
3	Yellow	H1-G Steady	H1-Y Steady	H1-G Steady	H1-Y Steady	H1-G Steady
4	Dark	None	None	None	None	None
5	Red	H1-R Steady	H1-R Steady	H1-R Steady	H1-R Steady	H1-R Steady
6	Green	H1-L Steady	H1-L Steady	H1-L Steady	H1-L Steady	H1-L Steady
7	Yellow	H1-R Steady	H1-L Steady	H1-R Steady	H1-L Steady	H1-R Steady
8	Dark	None	None	None	None	None

Aspect #3 (Clear) is simpler in that it just flashes the red LED 'H1-G Steady' in 'Lamp B' followed by the green LED 'H1-Y Steady' in 'Lamp A'. The 2 unused lamp entries are set to 'None'.

Aspect #4 (Dark) in this example, is dark so each of its 4 lamp entries are set to 'None'.

---

**Note** Do not allow a signal using the 'H2 Flicker' effect to be flashed by software to show Advance Approach. It will flicker the red before each yellow is displayed in an un-prototypical manner. Instead add an additional aspect for 'Advance Approach' that uses the Signal LCC-32H itself to flash the signal.

---

The second LED using aspects #5 - #8 is controlled in a like manner. The difference being that the second LED is connected to 'H1-R' and 'H1-L', and output lines --- and --- are used to control the aspects.

---

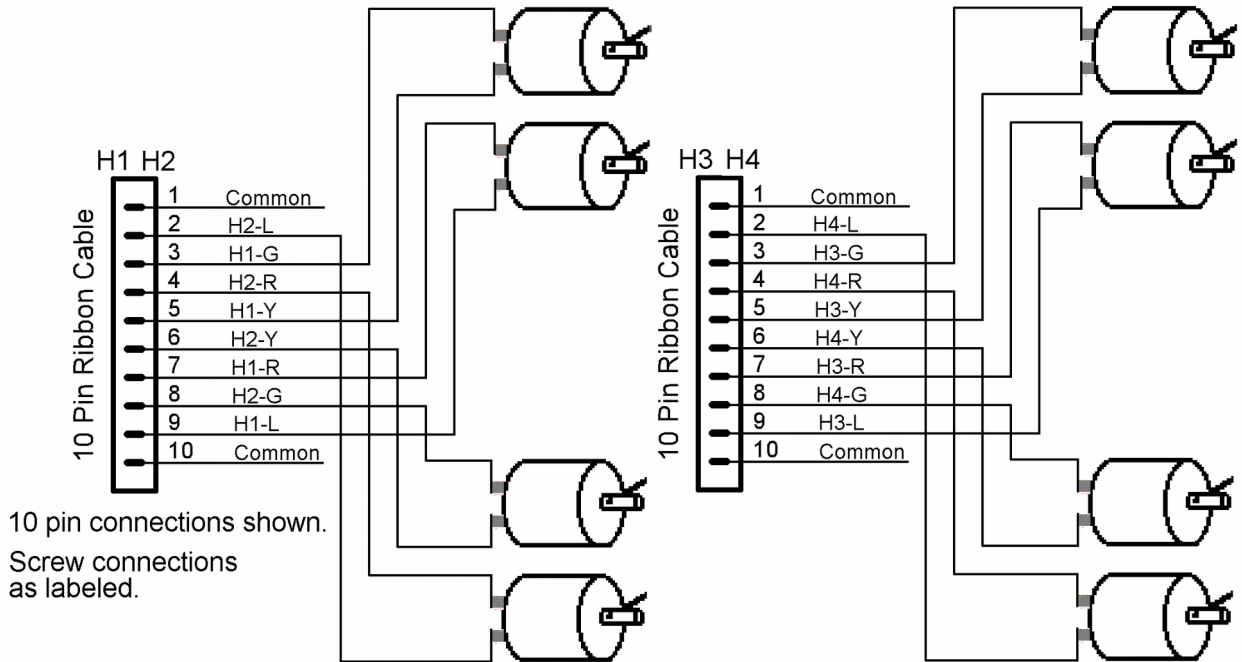
**Tip** When using dual color LEDs it is difficult to get a good yellow signal. The green and red dies are usually closer to each other than in a Bi-Polar LED. However both colors may still be seen distinctly. Second, as the viewer changes position he will see more or less of each color due to the lens of the LED. To minimize this color shift be sure to locate the LED leads horizontally to each other. Finally, neither the red nor the green colors are very close to an actual signal's colors. For best color fidelity consider using the RR-CirKits SS-RGY instead.

---

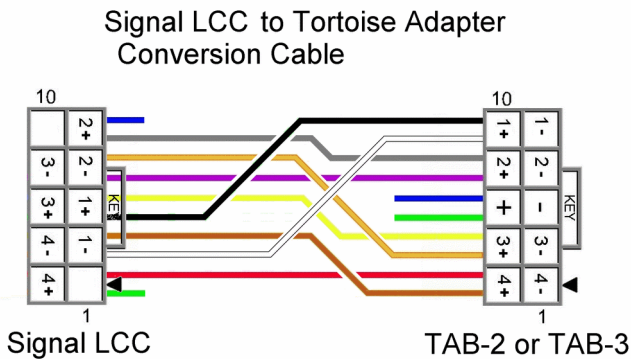
### 9.6.6 Stall Motor Machine Wiring

The following examples show how to wire low current Stall Motors such as Tortoise®, SwitchMaster® or other low current switch machines. The Signal LCC-32H outputs are rated for 20mA maximum. Most Stall Motor machines draw 10 to 20mA when running at a reasonable speed.

## Stall Motor Actuators Driven with a Signal LCC.



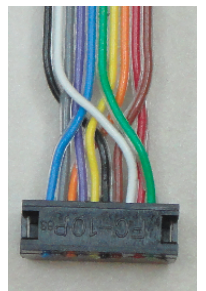
### 9.6.7 Switch Machine Connections



Switch machines represent the simplest possible control because they have just two positions, closed, and thrown. A switch machine is wired the same as a Bi-Polar LED would be, but with no need to show a 'yellow' signal.

RR-CirKits makes an adapter from the Stall Motor Driver board (SMD-8) to a Tortoise® machine. It is called the TAB-2. If you do this wire conversion

at the end of your cable it will allow the Signal LCC-32H to connect to the TAB-2 (or TAB-3) boards for solder less plug connections to your Tortoise® machines. (the blue and green connections are optional for the TAB-2, but supply the lamp circuit on the TAB-3.)



Aspect#	Mode	Command	Lamp A Phase/flash	HiPolar	Lamp B Phase/flash	HiPolar	Lamp C Phase/flash	HiPolar	Lamp D Phase/flash	HiPolar	Lighting Effect
1	Turnout	DCC	H1-G	Steady	Hi	H1-Y	None	Hi	H1-R	None	Effect None
2	Turnout	DCC	H1-Y	Steady	Hi	H1-G	None	Hi	H1-L	None	Effect None

This is the Signal LCC-32H Mast Table entry for a Stall Motor Turnout connected between the H1-G and H1-Y output lines and controlled by ---.

Aspect #1 (--) is easy. Just activate H1-G. The opposite wire will be correct.

Aspect #2 (--) is just as easy. Activate H1-Y. The opposite wire will be correct.

---


**Tip** When using the Signal LCC-32H to drive Stall Motors it may improve their operation to add 330 ohm resistors in parallel with the current limiter resistors to increase the output current available. Also be sure to set the brightness to maximum.

---


### 9.6.8 Semaphore Machine Connections


Semaphore machines using three position Tortoise® units require double outputs from the Signal LCC-32H to allow for three positions using the internal contacts in a Tortoise® actuator with a TAB-3 board. These can be controlled using just 4 aspects with each aspect controlling 4 output lines. The 4th aspect is 'Dark' and will extinguish the Semaphore lamp without moving the blade. Use the 'ca' polarity setting on the Signal LCC-32H. This will require setting the 'brightness' (speed) such that the machine's coasting distance is equal in each direction to maintain a consistent 45 degree positioning. This speed is somewhat faster than a prototype blade normally moves.

Aspect#	Mode	Command	Lamp A Phase/flash	HiPolar	Lamp B Phase/flash	HiPolar	Lamp C Phase/flash	HiPolar	Lamp D Phase/flash	HiPolar	Lighting Effect
1	Turnout	DCC	H1-G	Steady	Hi	H1-Y	None	Hi	H1-R	None	Effect None
2	Turnout	DCC	H1-Y	Steady	Hi	H1-G	None	Hi	H1-L	None	Effect None
3	Turnout	DCC	H1-G	Steady	Hi	H1-L	None	Hi	H1-R	None	Effect None
4	Turnout	DCC	H1-Y	Steady	Hi	H1-R	None	Hi	H1-G	None	Effect None

 Aspect #1 (Stop) enables 'H1-Y Steady' and 'H1-R Steady' which moves the TAB-3 with Tortoise® controlled blade to the horizontal position. The 2 unused lamp entries are set to 'None'.

 Aspect #2 (Approach) enables 'H1-Y Steady' and 'H1-L Steady' which moves the TAB-3 with Tortoise® controlled blade to the 45° position. The 2 unused lamp entries are set to 'None'.

 Aspect #3 (Clear) enables 'H1-G Steady' and 'H1-L Steady' which moves the TAB-3 with Tortoise® controlled blade to the vertical position. The 2 unused lamp entries are set to 'None'.

 **DARK** Aspect #4 (Dark) is dark so each of its 4 entries are set to 'None'. This removes power from the TAB-3 board and the lamp extinguishes without moving the blade from its previous position.

## 10 Rules

---

It is planned to eventually include logic presets for the simplest types of signal rules. These rules will make assumptions about which I/O ports are used for each function such as block occupancy, turnout position, etc. in order to simplify the installation of a signaling system.

## 11 Brightness

---

Each signal lamp may be adjusted for brightness. Minimum brightness is set to 8 and maximum is 255. The brightness of 8 gives just enough dynamic range to allow for fading a lamp even when set to minimum intensity.

## 12 Signal LCC-32H compatible Input/Output Cards

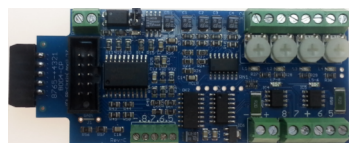
---

The RR-CirKits Signal LCC-32H and its compatible I/O modules are designed to be clipped into Tyco 3-1/4" Snap-Track® mounted to the bench work. (Snap-Track® is a plastic channel designed to mount PC cards to a chassis, not something to run trains on.)

A single Signal LCC-32H or compatible I/O module fits into the 3TK2-1 (single) mounting track. Other widths are available for compact installations using multiple boards. A common interlocking installation would use a double mounting track.

Each I/O module is equipped with connectors to facilitate these I/O board connections. Use IDC connectors and ribbon cables to connect the Signal LCC-32H to the I/O cards.

### 12.1 BOD4-CP (DCC 4 Block Occupancy Detector - Dual Turnout Driver)



*BOD4-CP*

The BOD4-CP is designed to augment the Signal LCC-32H in order to do all of the primary functions required at a typical interlocking. (Control Point) The BOD4-CP does not expect you to re-wire your layout to bring track feeders to remotely located detector cards. The small CT (Current Transformer) detection coils are placed directly on the track feeders where they belong. A short length of Cat-5 cable is the usual way to carry the detected current back to the detector boards. The use of CT coils means that there are no track voltage losses associated with the detectors. Normal detection levels are 1mA. but may be adjusted to higher levels with on board pots.

During a DCC bus power failure the Power-Lok input on the BOD4-CP instantly locks the present state of each block detector. I.e. the state of the layout does NOT change during a DCC power outage, neither to all occupied, nor to all vacant. It just suspends the sending of any occupancy changes until after power is restored and the layout has stabilized again. If you do not want to use the Power-Lok feature, there is a jumper to disable it.



The BOD4-CP outputs are low during detection so the Signal LCC-32H should be configured accordingly.

The BOD4-CP Also includes two dual output turnout drivers and 4 input lines. To take advantage of this dual use the Signal LCC-32H output section needs to be configured as 'Sample' mode. The BOD4-CP includes a screw terminal strip to connect 4 logic level input lines. These lines include internal 10K ohm pull up resistors to +5V, and may be driven by logic level circuits or contacts.

It is designed to drive stall motor turnout machines such as those found in Tortoise® and Switchcraft® machines. It is also suitable for the higher current required by the MTB-MP1 and MTB-MP5 machines. The BOD4-CP Output section is also able to drive solenoid coils or other high voltage medium power devices. Normally the input voltage should not exceed 27VDC. The BOD4-CP board is optically isolated from the driving circuitry to protect the Signal LCC-32H or other control device from these high power outputs. When driving stall motors, single coils or high power loads configure the lines as a steady output.

By using the proper options on the Signal LCC-32H the SCSD-8 may also be used to control dual coil momentary switch machines. In 'Dual Coil' mode the output lines must be paired such that each pair of lines requires just a single event pair. However reverse the two EventIDs. This action will normally require a 0.1 second pulse when driving solenoids.

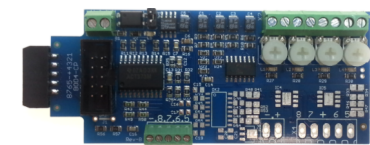
The lines are paired and only the primary event of the first line of each pair will be used to trigger a pulse.

Dual coil operation should not be attempted if the switch machine power supply is not of the capacitive discharge type that will limit the long term current to a low value, or contains proper fusing, in case of hardware or configuration errors.

**Failure to observe this precaution may result in destruction of equipment and could be a fire hazard!**

The BOD4-CP is not designed to drive last generation dual coil (Kemtron style) switch machines that require over 3A to operate properly. The BOD4-CP output current is internally limited to approximately 3A and these old machines will not operate reliably, if at all. Use an SCSD-8, RB-4, or else install more modern machines.

## 12.2 BOD4 (DCC 4 Block Occupancy Detector)



*BOD4* The BOD4 is the detector only version of the BOD4-CP. It is designed to augment the Signal LCC-32H with 4 detection blocks when no high current turnout drivers are required. The BOD4 does not expect you to re-wire your layout to bring track feeders to remotely located detector cards. The small CT (Current Transformer) detection coils are placed directly on the track feeders where they belong. A short length of Cat-5 cable is the usual way to carry the detected current back to the detector boards. The use of CT coils means that there are no track voltage losses associated with the detectors. Normal detection levels are 1mA. but may be adjusted to higher levels with on board pots.

During a DCC bus power failure the Power-Lok input on the BOD4 instantly locks the present state of each block detector. I.e. the state of the layout does NOT



change during a DCC power outage, neither to all occupied, nor to all vacant. It just suspends the sending of any occupancy changes until after power is restored and the layout has stabilized again. If you do not want to use the Power-Lok feature, there is a jumper to disable it.

The BOD4 outputs are low during detection so the Signal LCC-32H should be configured accordingly.

The BOD4 also includes a screw terminal strip for easy connections to the 4 unused Signal LCC-32H logic level I/O lines.

## 12.3 BOD-8 (DCC Block Occupancy Detector - 8 block)

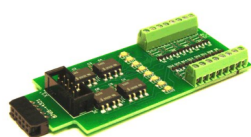


The BOD-8 does not expect you to re-wire your layout to bring track feeders to the detector cards. The small CT (Current Transformer) detection coils are placed directly on the track feeders where they belong. Simple lengths of Cat-5 cable are the usual way to run the signals back to the detector boards. Use of CT coils means that there are no track voltage losses associated with the detectors. Normal detection levels are 1mA. but may be adjusted to higher levels with on board pots.

During a DCC bus power failure the Power-Lok input on the BOD-8 instantly locks the current state of each block detector. I.e. the state of the layout does NOT change during a DCC power outage, neither to all occupied, nor to all vacant. It just suspends sending any occupancy changes until after power is restored and things have stabilized again. If you do not want the feature there is a jumper to disable it.

The BOD-8 outputs are low during detection so the Signal LCC-32H should be configured accordingly.

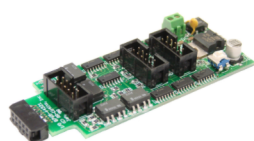
## 12.3 OIB-8 (Opto Isolator Board - 8 input)



This 8 input board is used when a non-isolated source of voltage needs to be monitored and input to the Signal LCC-32H. One example would be to monitor the DCC voltage on a set of points to determine the position of a turnout without using auxiliary contacts.

This board may be configured to monitor the absence or presence of an AC or DC signal. This board requires 10mA. for reliable operation and includes built in current limiters. It is often used for detection on three rail systems by isolation of one rail.

## 12.4 SMD-8 (Stall Motor Driver - 8 line)

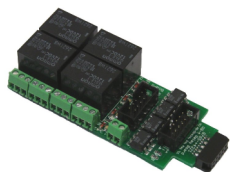


The SMD-8 board contains 8 individual, optically isolated, H-Bridge drivers. This allows the board to be powered from any supply between 9 Volts and 27 Volts. It is primarily designed to drive stall motor turnout machines such as those found in Tortoise® and Switchcraft® machines. It is also suitable for the

higher current required by the MTB-MP1 and MTB-MP5 machines. Do not exceed 20VAC or 27VDC at the power input.

This board includes an adjustable buck switching regulator to allow you to control the speed of your switch machine motors. This regulator can not boost the drive voltage above the supply voltage.

## 12.5 RB-4 (Relay Board - 4 x SPDT)



Relay Board - 4 is a Quad 10A SPDT relay board with logic level drivers. It is suitable for use with Signal LCC-32H or other logic level output devices. It requires 12V auxiliary power to drive the relay coils. Auxiliary power is optically isolated from the logic inputs for double isolation. LED indicators for each relay make it easy to monitor activity.

Includes dual ribbon connectors with offset lines to allow easy connection as output 1-4, or output 5-8, from the Signal LCC-32H, or other driver.

The RB-4 may be connected in parallel with a BOD4 when detection plus high current outputs are required. Use a single 10 wire flat ribbon cable with 3 IDC connectors, and connect it to the 5-8 input.

The RB-4 input lines are active low so all lines on this Signal LCC-32H port should be configured appropriately. This inverted input mode matches most types of driver outputs, and the drive polarity may be easily switched either in the Signal LCC-32H configuration or by reversing the RB-4 output contacts.

## 12.6 BOB-S (Break Out Board)

This board is a convenient way to convert from 10 pin ribbon cable to screw terminals. It may be used for inputs or outputs.

For input lines do not exceed 5V on any input or the Signal LCC-32H will be damaged.

# 13 Trouble shooting

---

## 13.1 Sanity Test

To perform a very basic Signal LCC-32H sanity test perform the following steps:

- Power up the Signal LCC-32H by plugging it into a powered network.
- The green power LED should come on.
- If no other board is present on the network, then the red activity light will begin to flash rapidly while the board seeks to establish an alias.
- The previous output states should be automatically be restored.

If the green power LED does not light, be sure that a power supply is connected to the LCC network segment, and provides at least 9V to the Signal LCC-32H. The green power LED will initially light at much lower voltages, so it is not a reliable indicator of suitable power.

## 13.2 Activity Test

The Signal LCC-32H's input circuit and code sends data directly to the unit's processor, so if you send any command to the unit it should immediately be seen on the command (COM) LED. This test uses the free software available from the JMRI project to watch the test commands. ([www.jmri.org](http://www.jmri.org))

Steps:

- Open the JMRI LCC® Monitor window. Using the JMRI turnout control send a command to any output line on this Signal LCC-32H. The command should appear in the LCC® monitor window and the Signal LCC-32H command (Y) LED should blink.
- The connected output should respond.

If there is activity at the LCC Terminator blue LED, but no activity light at the Signal LCC-32H when events are sent, check the LCC wiring. If the command is seen in the LCC® monitor, but not in the command light, be sure that the command you are sending is configured to respond on this Signal LCC-32H. If there is no activity shown in the LCC® monitor window, check that you have the correct interface selected in the JMRI preferences, and that you have the correct COM port selected.

A flashing yellow lamp on a single board may indicate the failure to complete the alias check. Note that a minimum network will have two nodes, or a single node plus a computer connection.

The Signal LCC-32H is initially configured as simple input lines. You may use a RR-CirKits I/O test board to send events simply by connecting it to the input port and pressing the test buttons.

## 14 Boot Loader

---

### 14.1 Firmware Upgrade

If an update to your Signal LCC-32H firmware is needed, a program such as JMRI version 4.10 or later is required.

To enter Firmware upgrade mode:

- 1) Start JMRI and select "OpenLCB".
- 2) Select 'Firmware Update' from the OpenLCB drop down list.
- 3) Select your 'Target Node ID'.
- 4) Click 'Select' to pick a firmware file.
- 5) From the file menu, select: 'Signal LCC-32H\_rev\_B2\_UPDATE.hex' or the latest upgrade available.
- 6) Optionally you may check the 'Lock Node' check box to take it off line during the upgrade.
- 7) Click the 'Load' button to initiate the upgrade to Signal LCC-32H revision B2.
- 8) Wait until 'updating device firmware..' is complete.

- 9) Switch back to the OpenLCB Network Tree window.
- 10) It should now show 'Mod: Signal LCC-32H' and 'Software: rev B-2'.
- 11) Any errors will be shown in the lower window ticker tape display.

If the node does not automatically enter boot mode and start the upgrade it may be forced into boot mode by un-powering it, then holding down the 'Gold' button as you power it up again. The gold LED should start blinking to indicate that it is in forced boot mode. This will likely be required after a failed upgrade attempt.

## 15 Grounding and Isolation

---

Unlike the LCC Buffer-USB, the Signal LCC-32H is not optically isolated from the LCC bus. This allows for possible ground loop problems between the LCC® and your layout accessory power supplies, so be sure to keep the ground connection to the Power-Point either isolated, or else in common with your layout power source.

Normally all Signal LCC-32H connections will originate or reference to the Signal LCC-32H board itself, so there is no danger of ground loops with these connections. RR-CirKits High power output boards are optically isolated from the Signal LCC-32H ports and use separate power sources.

If you are building your own I/O boards or using third party units be sure to observe the common/isolated ground rules, and never exceed 5V on any I/O pin.

Properly ground your boosters, your power supplies, and your desktop computer through a 3 wire cable, and isolate them from each other via isolated equipment where necessary.

## 16 Warranty Information

---

We offer a one year warranty on the Signal LCC-32H. This device contains no user serviceable parts.

If a defect occurs, please contact RR-CirKits at: [service@rr-cirkits.com](mailto:service@rr-cirkits.com) for a replacement.

## 17 FCC Information

---

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions:

1. This device may not cause harmful interference, and
2. this device must accept any interference received, including interference that may cause undesired operation.

Note: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the

instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Any modifications to this device voids the user's authority to operate under and be in compliance with these regulations. The actual measured radiation from the Signal LCC-32H is much lower than the maximum that is permitted by the FCC Rules, so it is unlikely that this device will cause any RFI problems.

RR-CirKits, Inc.  
7918 Royal Ct.  
Waxhaw, NC USA 28173

<http://www.rr-cirkits.com>  
sales@rr-cirkits.com  
service@rr-cirkits.com  
1-704-843-3769  
Fax: 1-704-243-4310

# 18 Signal LCC-32H Work Sheets

## 18.1 Sample Rule Sheet

Mast Number:				Mast Type:				Comment		
Aspect - Name	Speed	Lamp A	F	Lamp B	F	Lamp C	F		Lamp D	F
0 - Stop	Stop									
1 - Take Siding	Stop									
2 - Stop Orders	Stop									
3 - Stop Proceed	Stop									
4 - Restricting	Slow									
5 - Permissive	Slow									
6 - Slow-Approach	Slow									
7 - Slow	Slow									
8 - Slow-Medium	Slow									
9 - Slow-Limited	Slow									
10 - Slow-Clear	Slow									
11 - Medium-Approach	Medium									
12 - Medium-Slow	Medium									
13 - Medium	Medium									
14 - Medium-Limited	Medium									
15 - Medium-Clear	Medium									
16 - Limited-Approach	Limited									
17 - Limited-Slow	Limited									
18 - Limited-Medium	Limited									
19 - Limited	Limited									
20 - Limited-Clear	Limited									
21 - Approach	Approach									
22 - Advance-Approach	Limited									
23 - Approach-Slow	Approach									
24 - Advance-Approach-Slow	Approach-Medium									
25 - Approach-Medium	Approach-Medium									
26 - Advance-Approach-Medium	Clear									
27 - Approach-Limited	Clear									
28 - Advance-Approach-Limited	Clear									
29 - Clear	Clear									
30 - Cab Speed	Clear									
31 - Dark	Stop									

Note: Speed is the maximum speed allowed when passing a signal showing this aspect. Enter this value in the Track Speed selection box immediately under the Aspect selection.

**F** = Flash speed. Steady, Slow, Medium, Fast. (Phase A / Phase B)

**Comments** = Lamp Fade, Appearance Effects, Etc.

Fill out a copy of this sheet for each mast type based on the rule book for your railroad. Show the lamps used for each type of mast. Lamp names (e.g. H1G, H1Y, Etc.) are for identification purposes only. You may use any lamp drivers in any mast. Standardizing the signal connections for your railroad will simplify the installation. Unused lamps may be used for any purpose.

## 18.2 Sample Logic Sheet

Logic - Description	Group	T	Var #1	Funct	T	Var #2	True-False		Actions	Comment
1 -										
2 -										
3 -										
4 -										
5 -										
6 -										
7 -										
8 -										
9 -										
10 -										
11 -										
12 -										
13 -										
14 -										
15 -										
16 -										
17 -										
18 -										
19 -										
20 -										
21 -										
22 -										
23 -										
24 -										
25 -										
26 -										
27 -										
28 -										
29 -										
30 -										
31 -										
32 -										

Note: Each node contains 32 conditionals. These do not need to reside in the node they control. They are strictly event processors.

**Group** = Blocked, Group, and Single (Last). The Group concept allows you to easily calculate signal aspects. Place the more restrictive aspects first in a group. A 'true' condition can send the aspect and skip over the rest of the conditional statements, thus assuring that only the most restrictive aspect is sent.

**T** = Trigger 'On Matching Event' or 'On Variable Change'

**Funct** = Conditional Function.

**Var** = Variable Events or Mast+Speed

**True-False** = Conditional Actions upon true or false. SX=Send then eXit Group, SN=Send then evaluate Next, X=eXit group, and N=evaluate Next.

**Action** = Delay and Events.