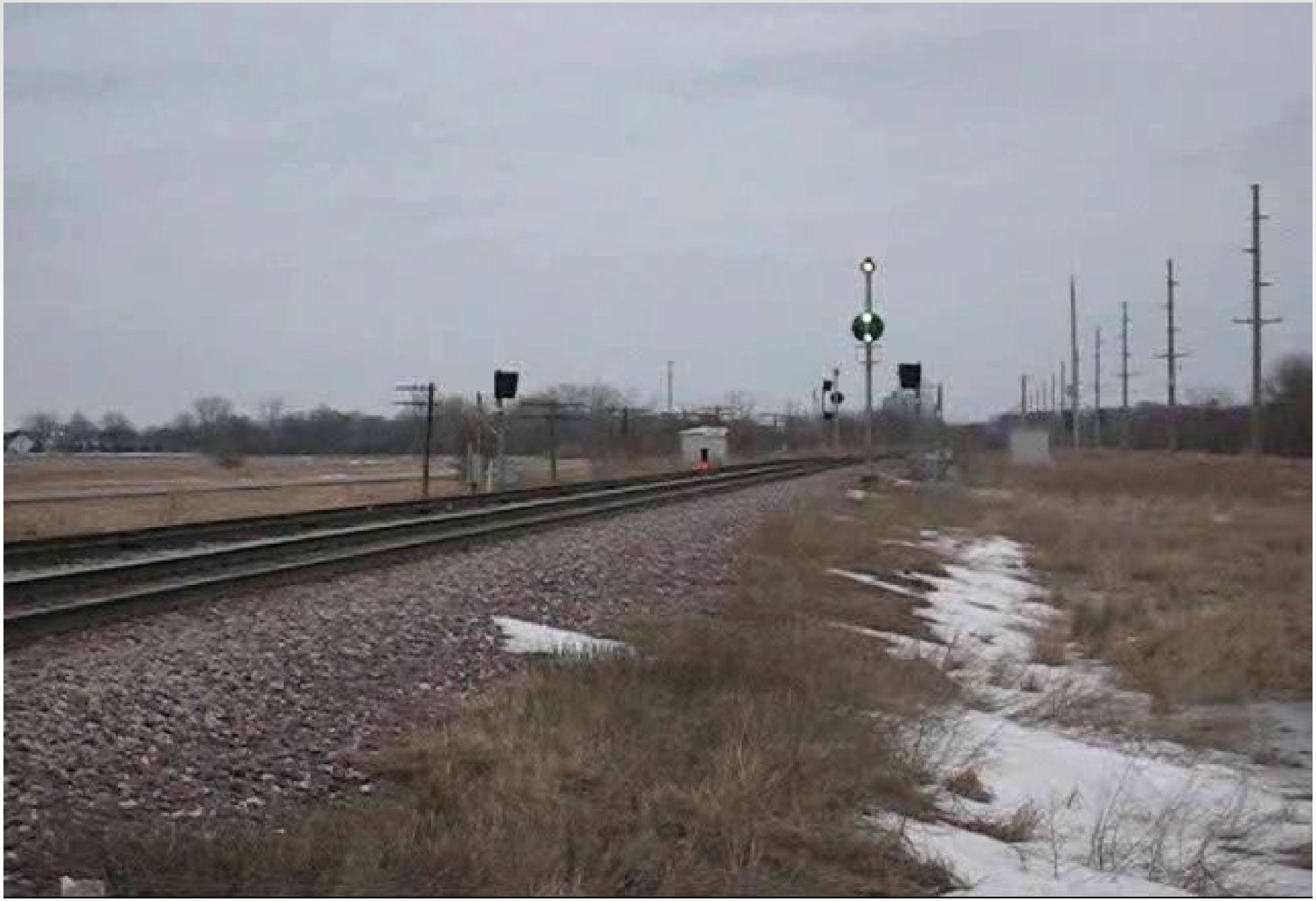
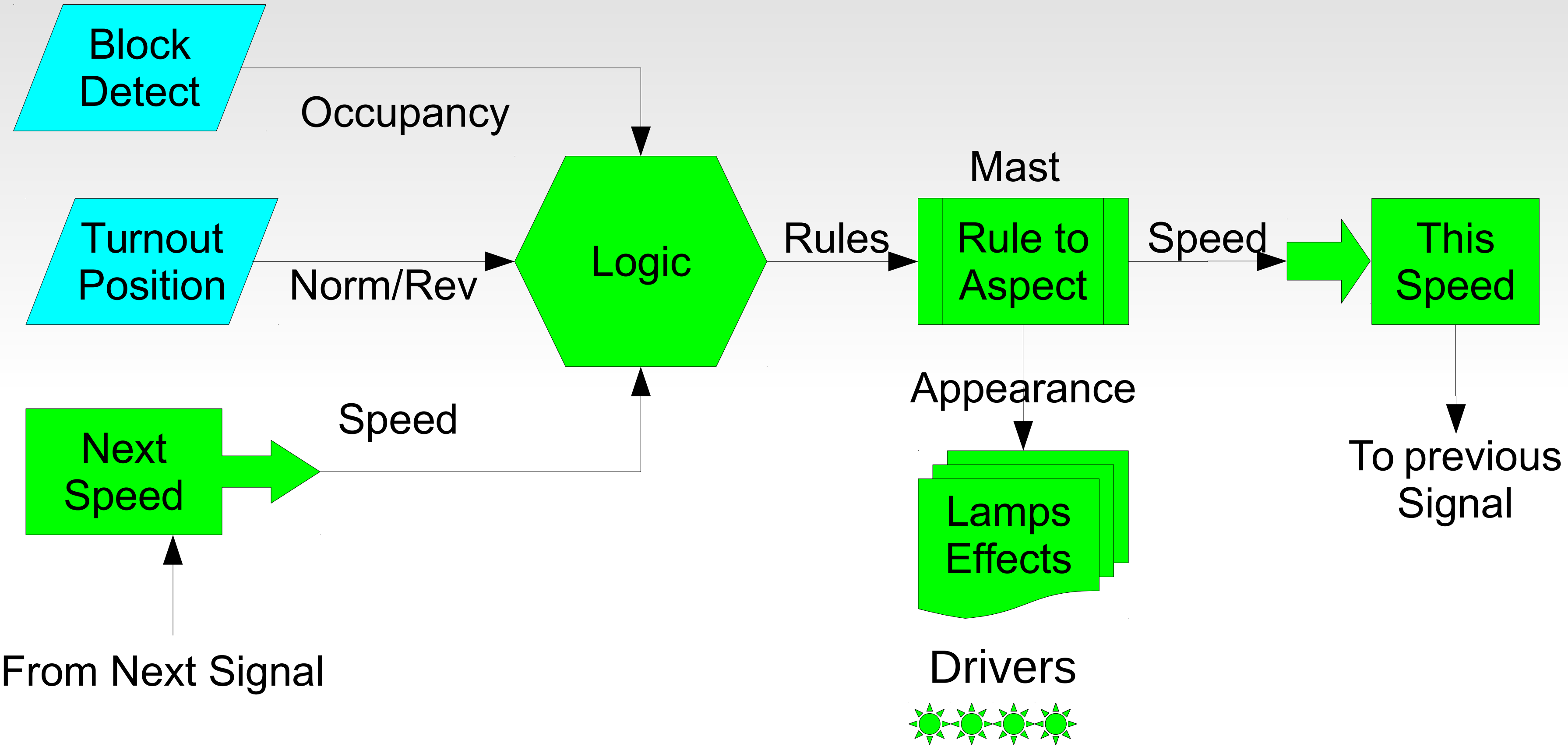
CPL Example



Signal Logic Example



With the Signal LCC all of the control functions required for signaling exist in a single node. Light blue items may be taken care of with a daughter card.

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

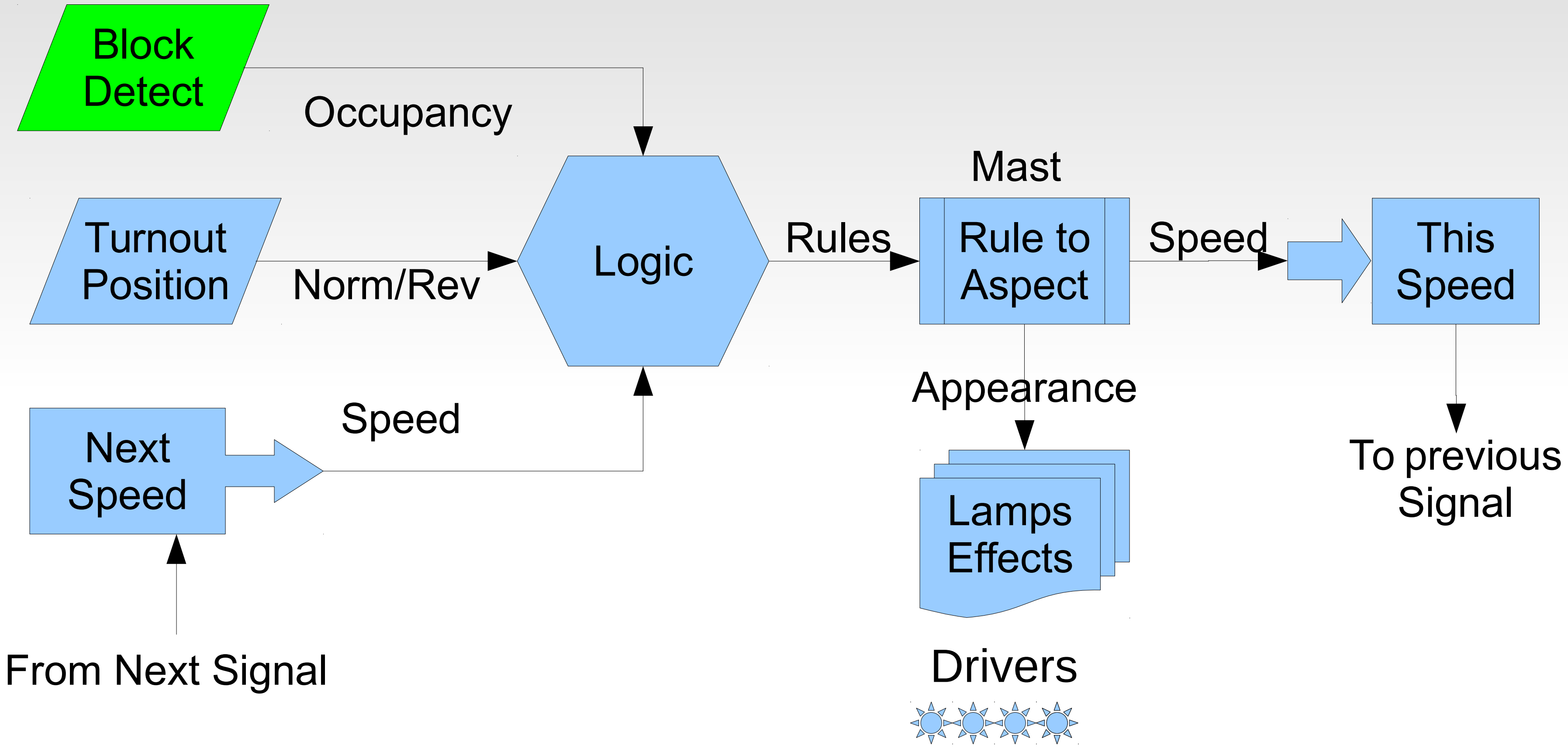
Signal Logic

- Signal Logic

In order to build a signal controller that watches all related status Events from the railroad and CTC panel, and makes independent decisions about the proper signal states and appearances, it must contain internal logic. This logic must either be user controlled or else it must understand all known signaling rules.

Triggering the evaluation of a conditional is done when any monitored event is seen. There are two trigger options. In the first option evaluation of a conditional is only done if the monitored event actually changes the state of the variable. In the second case the evaluation is done when ever the event is seen, even if there is no resulting change to a variable. This allows repeated single events to trigger a conditional multiple times.

Block Detect Example

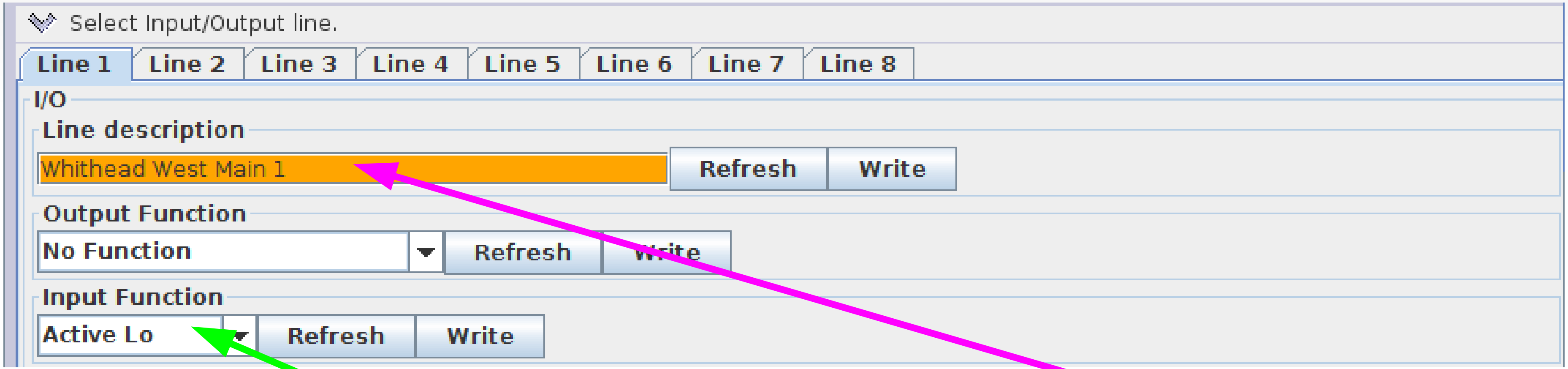


The BOD4 and BOD4-CP cards each include 4 block detector circuits for easy connection to the Signal LCC board. These boards use CT coils to prevent track voltage drop and provide 100% isolation. However they do not work with DC or battery operation.

Block Detector Variables

- Variables

Variables are used to follow the state of objects of interest such as block detectors, turnout positions, etc. Normally two events will allow the variable to follow the state of some object, true/false, normal/diverging, clear/occupied, etc.



Select Input/Output line.

I/O

Line description: Whithead West Main 1

Output Function: No Function

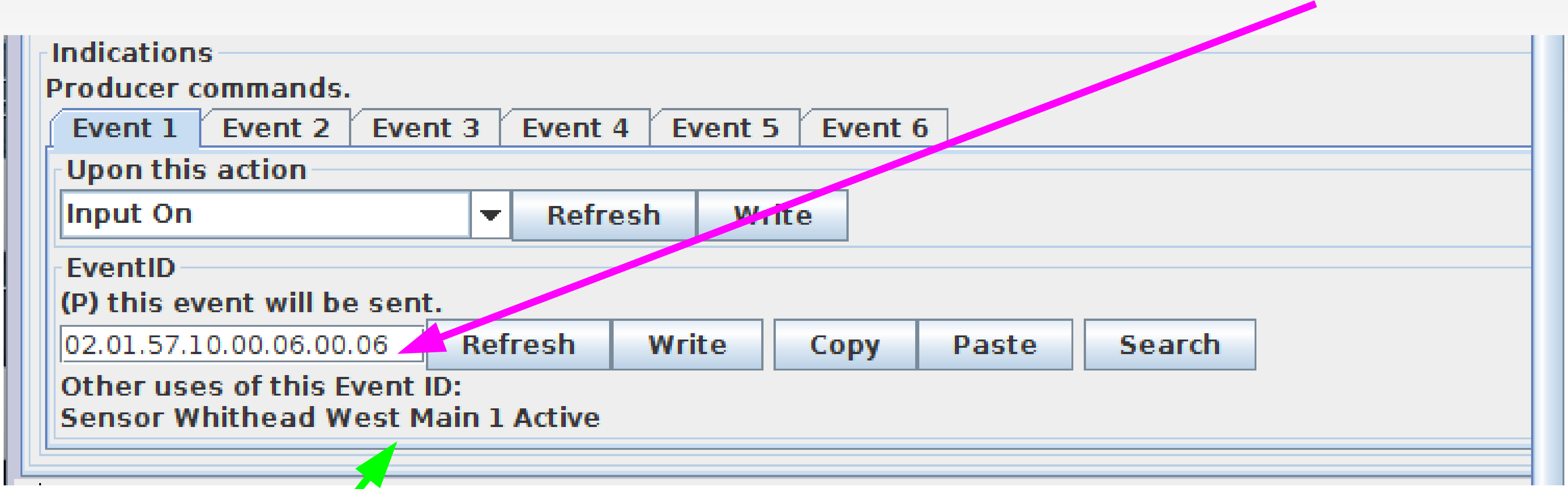
Input Function: Active Lo

Lets start by connecting a block sensor to an input. Its description is ‘Whithead West Main 1’ so we enter it in the description block and ‘Write’ it to the node. Detectors are Input Functions with ‘Active Lo’ so we set that and write it. For a normal Input be sure that the Output Function is set to ’No Function’. Of course ’Line 1’ is the one connected to our first block detector.

Block Detector Variables

- Input (Producer) Events

We now go to the Indications (Producers) for this line, and enable two events. The first (Event 1) will be sent when the Input is ’On’ (goes low in our application) When we need to know if the block goes occupied, we will use this EventID.



Indications

Producer commands.

Upon this action

Input On

EventID

(P) this event will be sent.

02.01.57.10.00.06.00.06

Other uses of this Event ID:

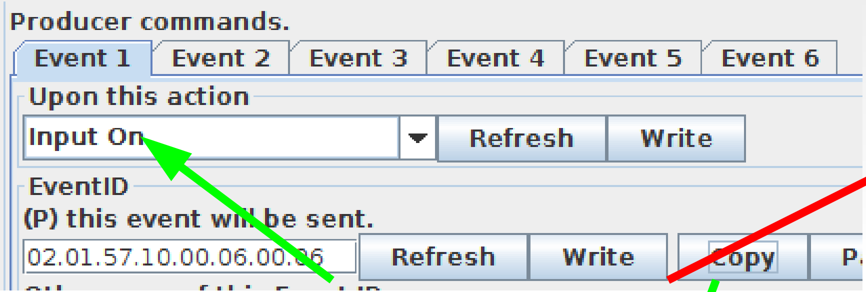
Sensor Whithead West Main 1 Active

Event 2 will be set to ‘Input Off’. Use ‘Copy’ and ‘Paste’ when you need to utilize the magic numbers (EventID) for these events. Its description ‘Whithead West Main 1’ Is noted here to remind you of its function. This information is known because I made a JMRI sensor that follows it. This is a JMRI feature available in the JMRI CDI tool.

Block Detector Variables

- Make Sensor – Make Turnout

To easily create LCC JMRI Sensor entries open the ‘Sensor/Turnout creation’ option found at the bottom of the JMRI CDI window.



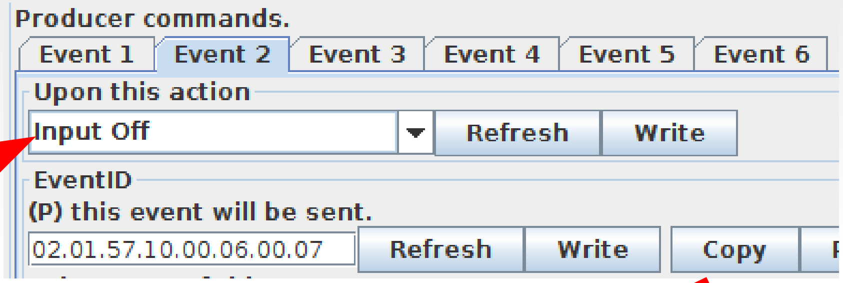
Producer commands.

Upon this action: Input On

EventID

(P) this event will be sent.

02.01.57.10.00.06.00.06



Producer commands.

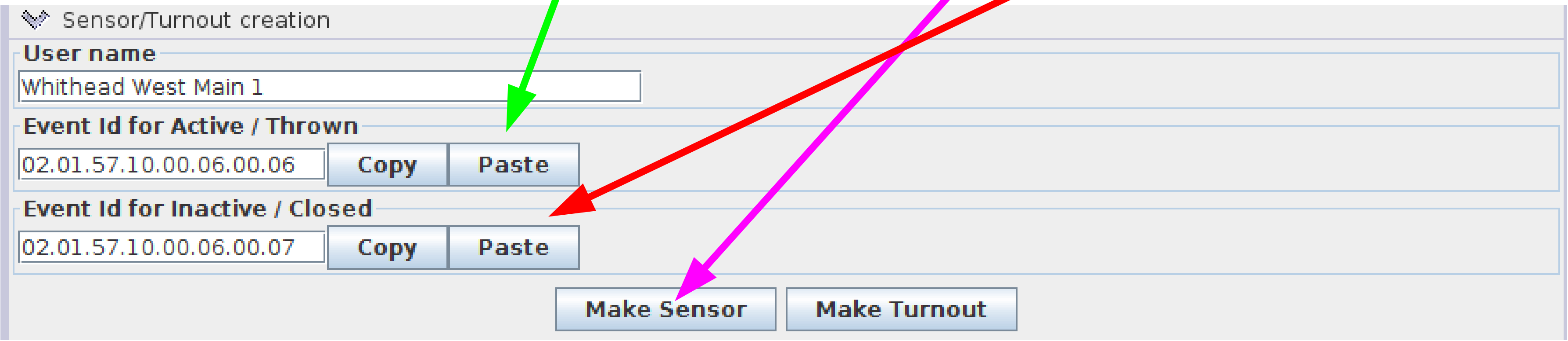
Upon this action: Input Off

EventID

(P) this event will be sent.

02.01.57.10.00.06.00.07

Copy the ‘Input On’ and ‘Input Off’ Events into the creation tool. Use ‘Copy’ and ‘Paste’ to move the events into the tool. Enter a User Name, and then click on ’Make Sensor.’ It will automatically be entered into JMRI. Note: LCC Sensor and Turnout events are the same on the bus, but JMRI needs to know which table to enter them into.



Sensor/Turnout creation

User name: Whithead West Main 1

Event Id for Active / Thrown: 02.01.57.10.00.06.00.06

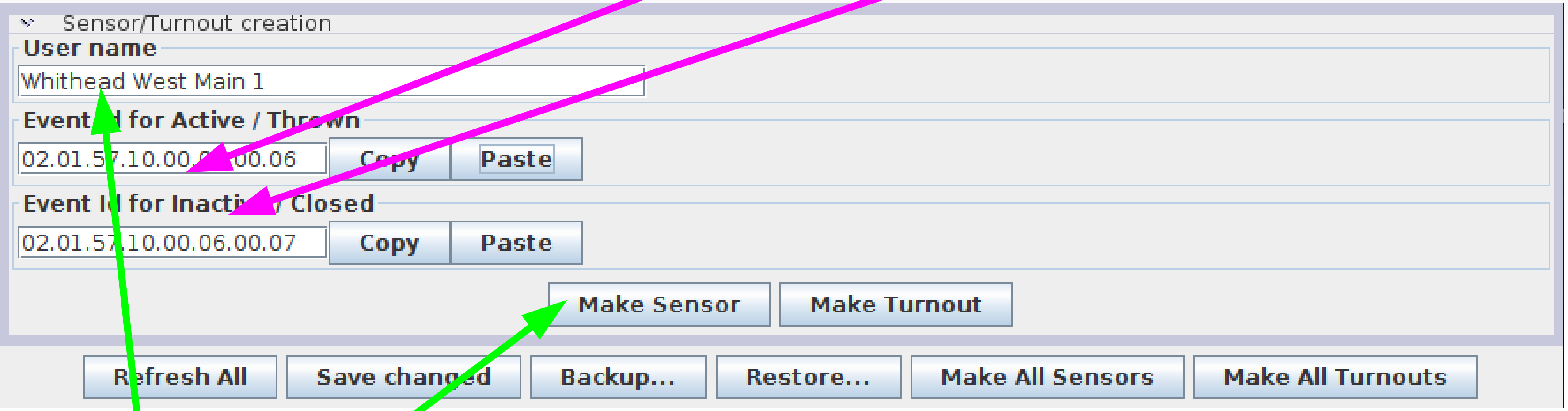
Event Id for Inactive / Closed: 02.01.57.10.00.06.00.07

Be sure to save the JMRI table for future use. Normally this data will become part of a ‘Panels’ file, and be synchronized with the node when the panel is loaded.

Block Detector Variables

- JMRI Sensors

JMRI includes a handy tool at the bottom of the CDI window to make sensors or turnouts from events. LCC Nodes may use two (or more) EventIDs to control sensors and turnouts, so you must use cut/paste to choose the pair that you want for JMRI. For this sensor we use Event 1 and Event 2 that we just defined.



Sensor/Turnout creation

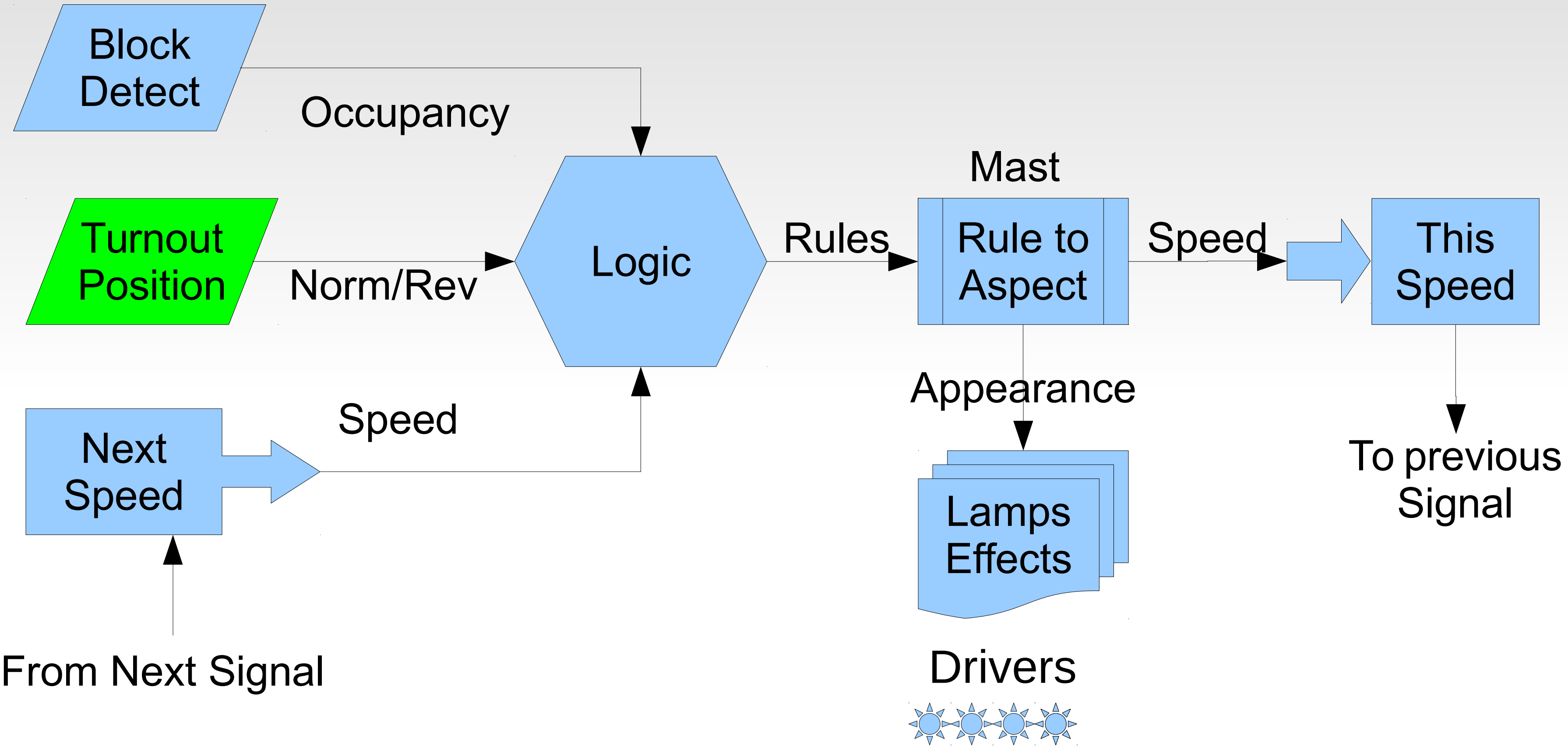
User name: Whithead West Main 1

Event Id for Active / Thrown: 02.01.57.10.00.06.00.06

Event Id for Inactive / Closed: 02.01.57.10.00.06.00.07

Enter the JMRI user name for this sensor, (or turnout) then click on the Make Sensor button. This item will automatically be added to your JMRI Sensor (or turnout) table. Be sure to save the table for future use. Normally this data will become part of a ‘Panels’ file, and be synchronized with the node when loaded.

Turnout Control Example

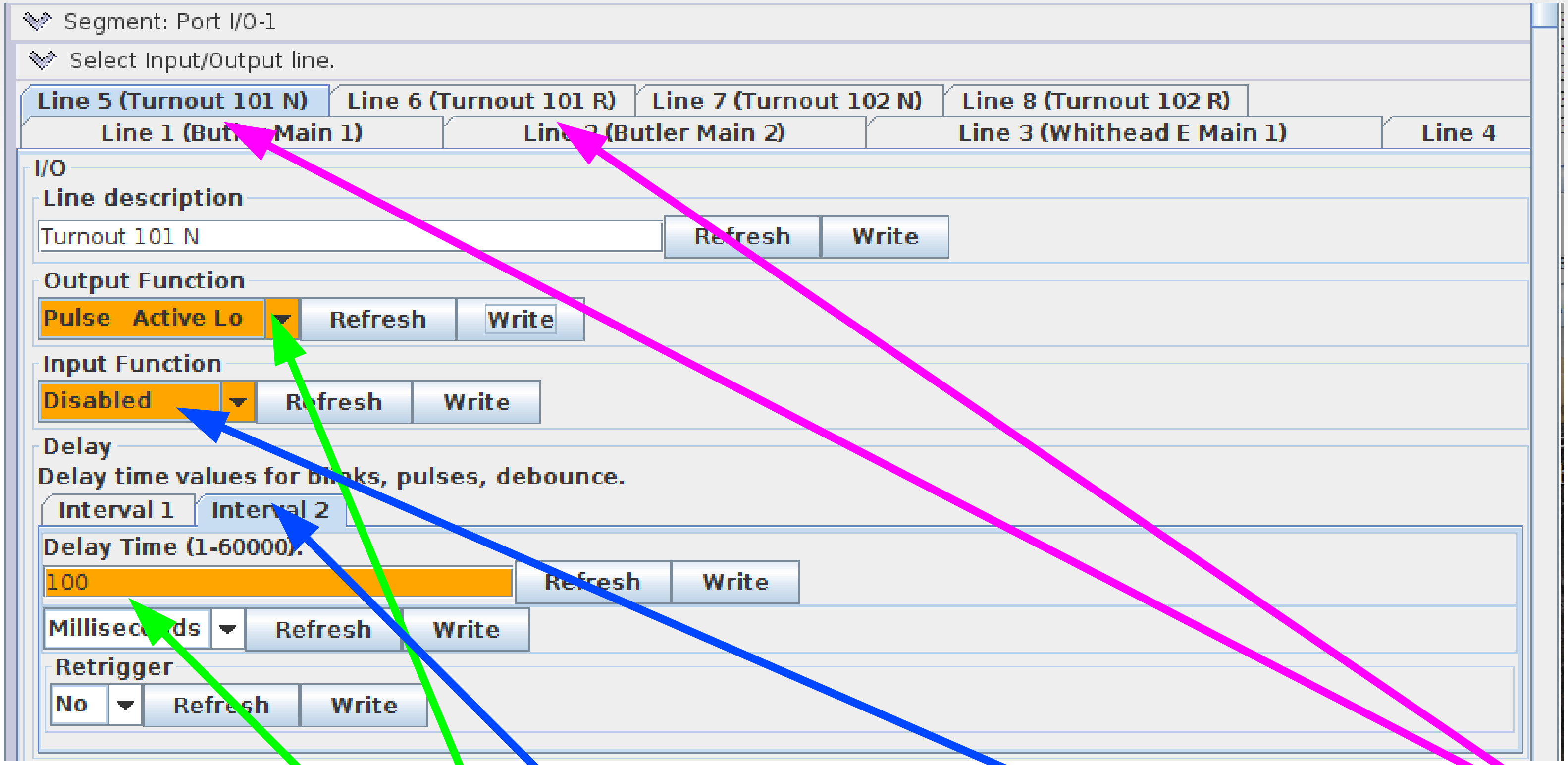


The BOD4-CP cards also include 2 ‘H’ Bridge drivers controlled by the Signal LCC board. These drivers are isolated from the LCC to prevent power supply issues.

Turnout Variables

- Output (Consumer) Events

We now go to the Indications (Producers) for a line. (on board ...07)



Segment: Port I/O-1

Select Input/Output line.

Line 5 (Turnout 101 N) | Line 6 (Turnout 101 R) | Line 7 (Turnout 102 N) | Line 8 (Turnout 102 R)

Line 1 (Butler Main 1) | Line 2 (Butler Main 2) | Line 3 (Whithead E Main 1) | Line 4

I/O

Line description: Turnout 101 N

Output Function: Pulse Active Lo

Input Function: Disabled

Delay

Delay time values for blinks, pulses, debounce.

Interval 1 | Interval 2

Delay Time (1-60000): 100

Milliseconds

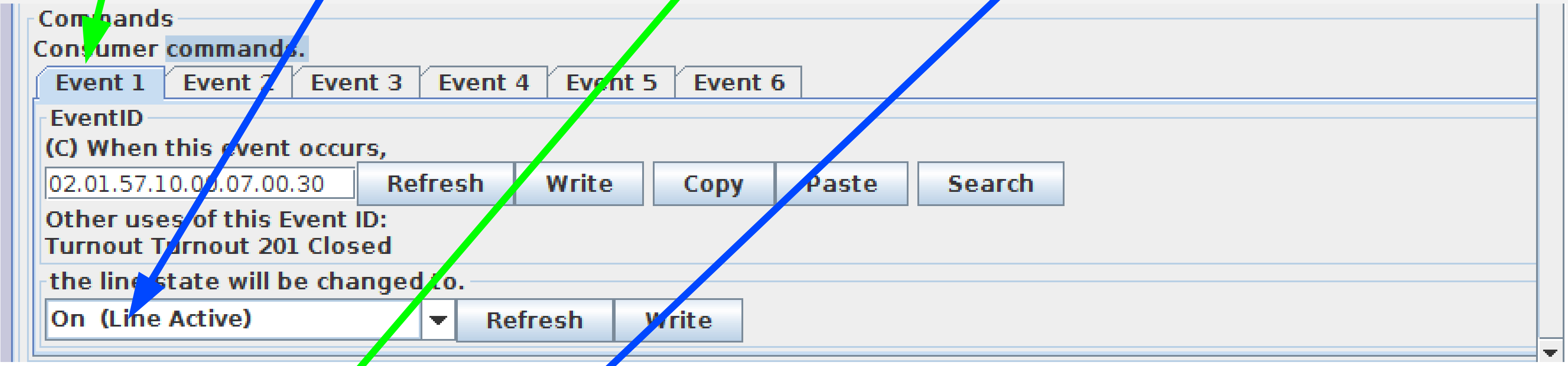
Retrigger: No

Our turnouts are controlled by Kato dual coil solenoids. This requires dual line drivers and 100mS pulse outputs. Normally inputs are disabled for Output Functions. Note: Use Interval 2 for pulse length. Interval 1 is the pulse delay.

Turnout Variables

- Output (Consumer) Events

Event 1 will turn ‘On’ the line and event 2 will turn it ‘Off’. Remember we already specified that ‘On’ just sends a 100mS pulse, so our coils are safe.



Commands

Consumer commands.

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

EventID

(C) When this event occurs,

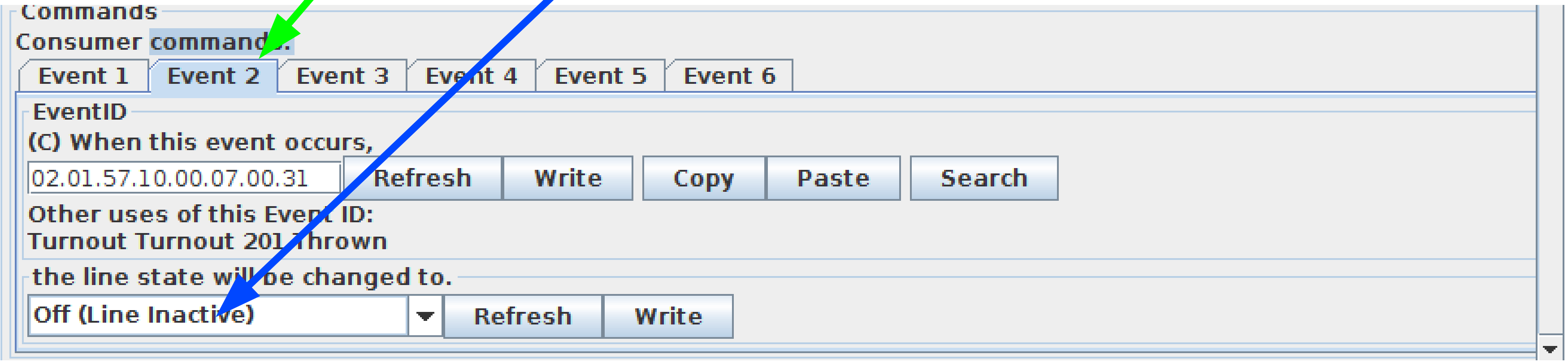
02.01.57.10.00.07.00.30

Other uses of this Event ID:

Turnout Turnout 201 Closed

the line state will be changed to.

On (Line Active)



Commands

Consumer commands.

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

EventID

(C) When this event occurs,

02.01.57.10.00.07.00.31

Other uses of this Event ID:

Turnout Turnout 201 Thrown

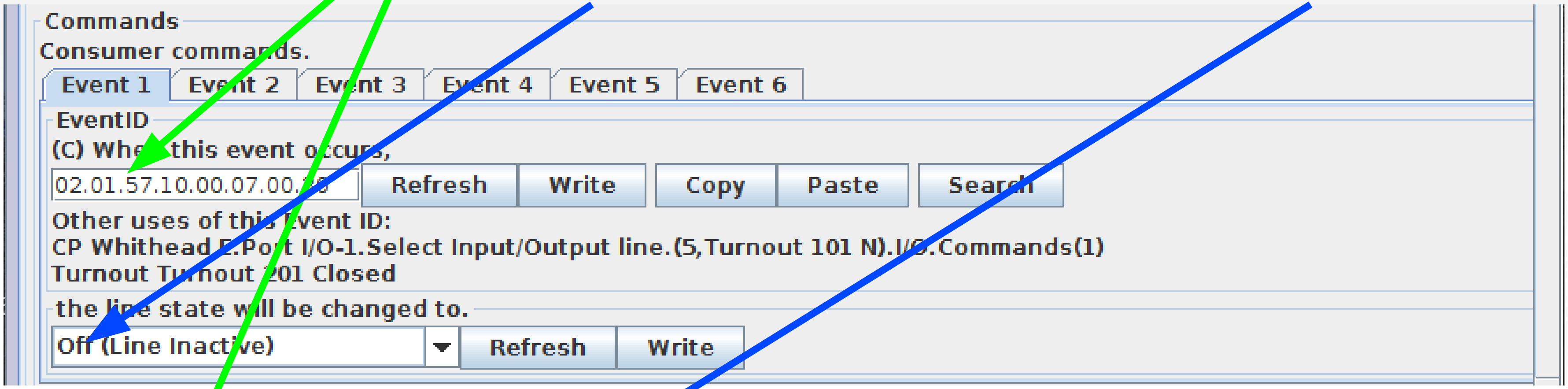
the line state will be changed to.

Off (Line Inactive)

Turnout Variables

- Output (Consumer) Events

To configure the second coil we will do two tricks with events. First we copy and paste the two events from the first line to the second line. Next we reverse their actions. Event 1 will turn ‘Off’ the line and event 2 will turn it ‘On’. Done!



Consumer commands.

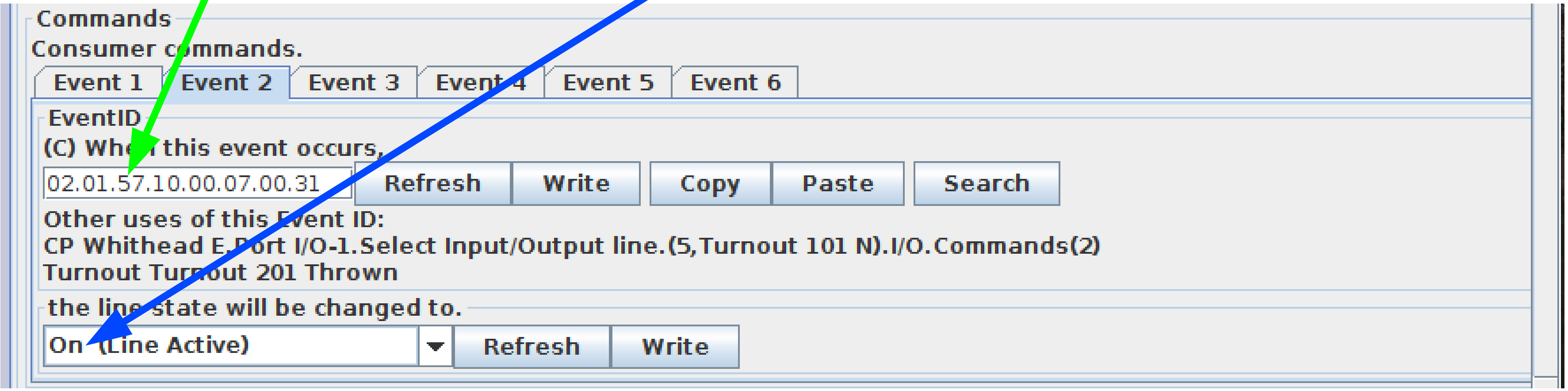
(C) When this event occurs,

02.01.57.10.00.07.00.30

Other uses of this Event ID:
CP Whithead E.Port I/O-1.Select Input/Output line.(5,Turnout 101 N).I/O.Commands(1)
Turnout Turnout 201 Closed

the line state will be changed to.

Off (Line Inactive)



Consumer commands.

(C) When this event occurs,

02.01.57.10.00.07.00.31

Other uses of this Event ID:
CP Whithead E.Port I/O-1.Select Input/Output line.(5,Turnout 101 N).I/O.Commands(2)
Turnout Turnout 201 Thrown

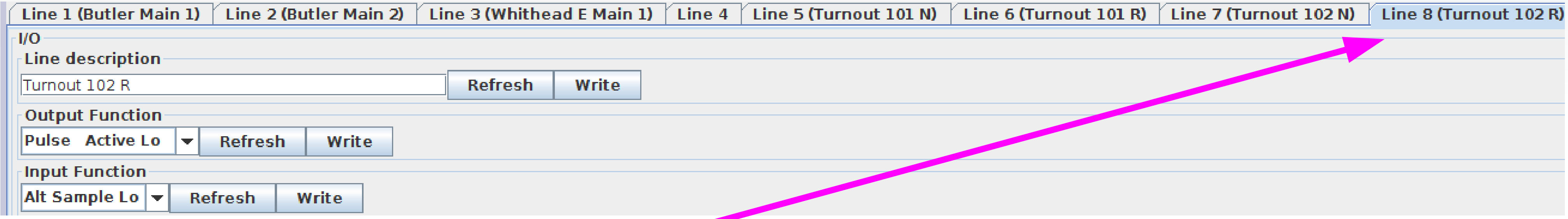
the line state will be changed to.

On (Line Active)

Turnout Control

- Input (Producer) Events.

We now get really fancy. To be compatible with the Berrett Hill Touch Triggers we added a 'Sample' option to our I/O lines. We take advantage of that on the BOD4-CP. Each output driver has a corresponding input line.



Line description: Turnout 102 R

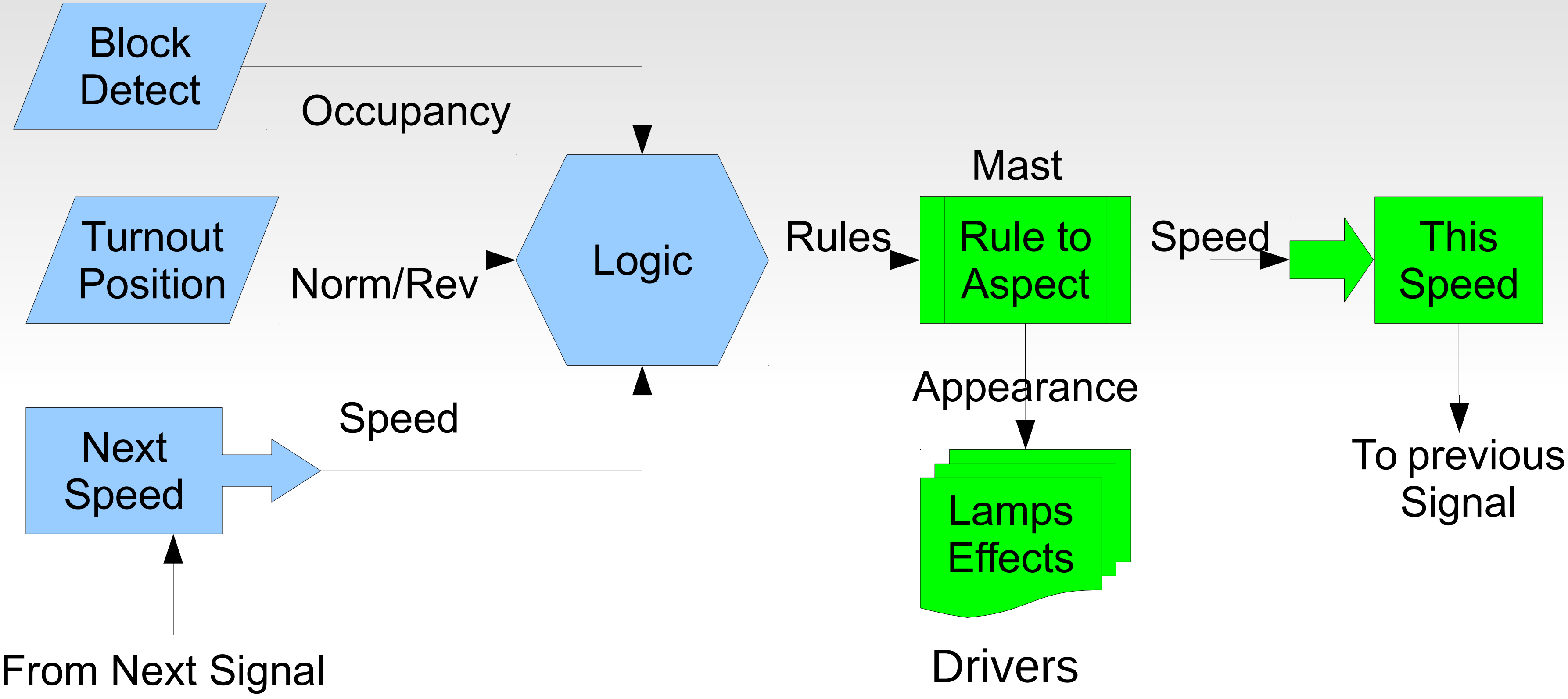
Output Function: Pulse Active Lo

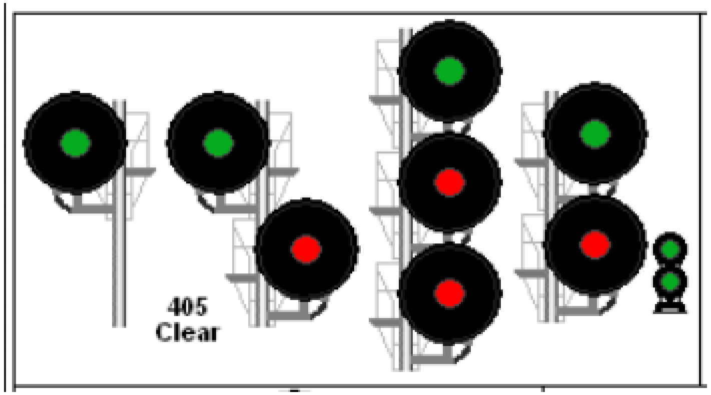
Input Function: Alt Sample Lo

We use the input for Line 8 and connect it to a push button. We set the 'Input Function' to be 'Alt Sample Lo'. This means that each time the input goes low it will alternate the function state. The line still sends its output to drive the turnout as before. We can also use the same line (physical wire) as an input by sampling it.

- For simplicity, have the line send the turnout control events directly. For realism, combine the control events with occupancy and/or panel information that prevents any turnout movement when occupied, or locked.

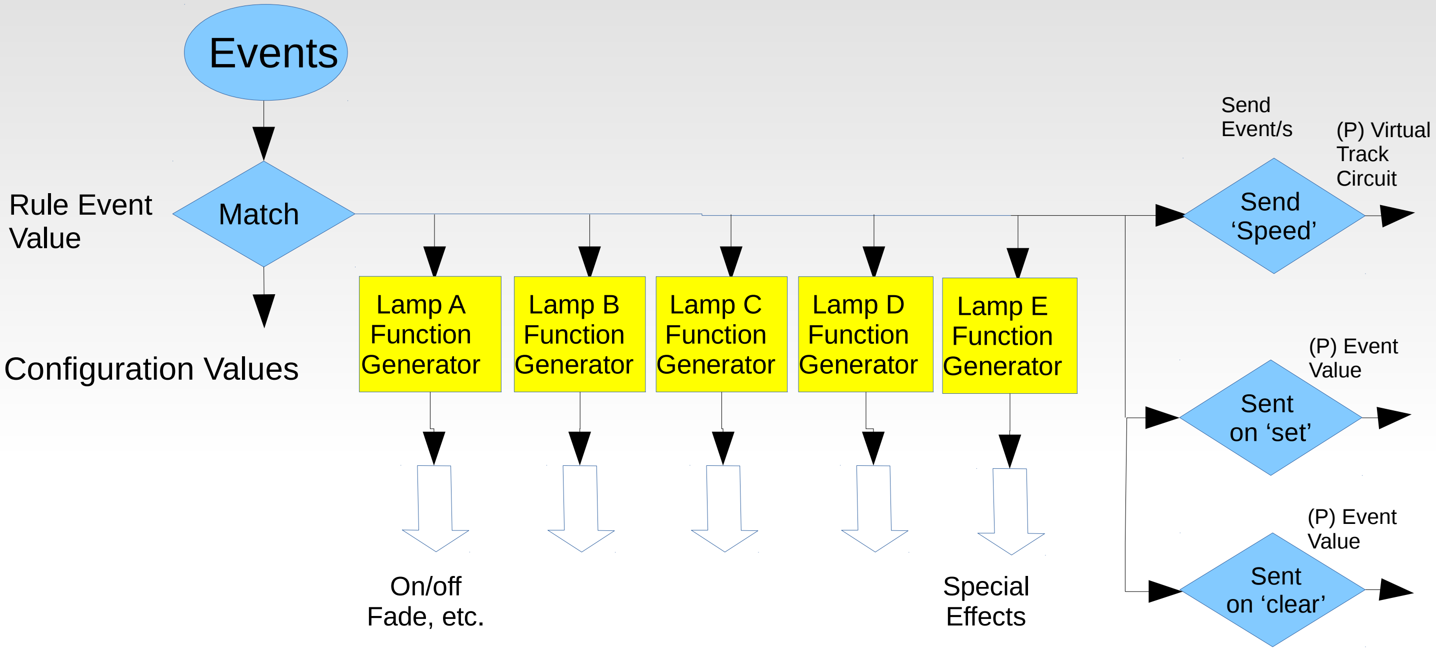
Rule to Aspect





A quick look at any railroad rule book will reveal that the same rule may be displayed in many ways. This means that we need a 'Rule' to 'Aspect' conversion process.

Rule to Aspect



- Signal Masts

This flow chart shows Mast functionality. Any signal rule that is seen (matched) can send up to 4 lamp control messages, an optional special effect, what speed is to be sent by the track circuit, and send optional events. These optional events may be used to send indications to a CTC panel.

Signal Masts

- The EventIDs sent and responded to by each rule are also controlled by the MASTS segment. Because this Rule to Aspect conversion is actually the links between the ’Rule’ events and the actual hardware we call it all ’MASTS’ and treat it as one segment in the CDI.
- The Signal LCC supports 8 Masts, each of which supports up to 8 aspects. If any mast requires more than 8 aspects, then the next ‘mast’ may be logically linked with a previous one.
- A ’Mast’ definition makes two assumptions.

    1) Only one aspect may be shown at a time. Setting any aspect automatically cancels any previous aspect.

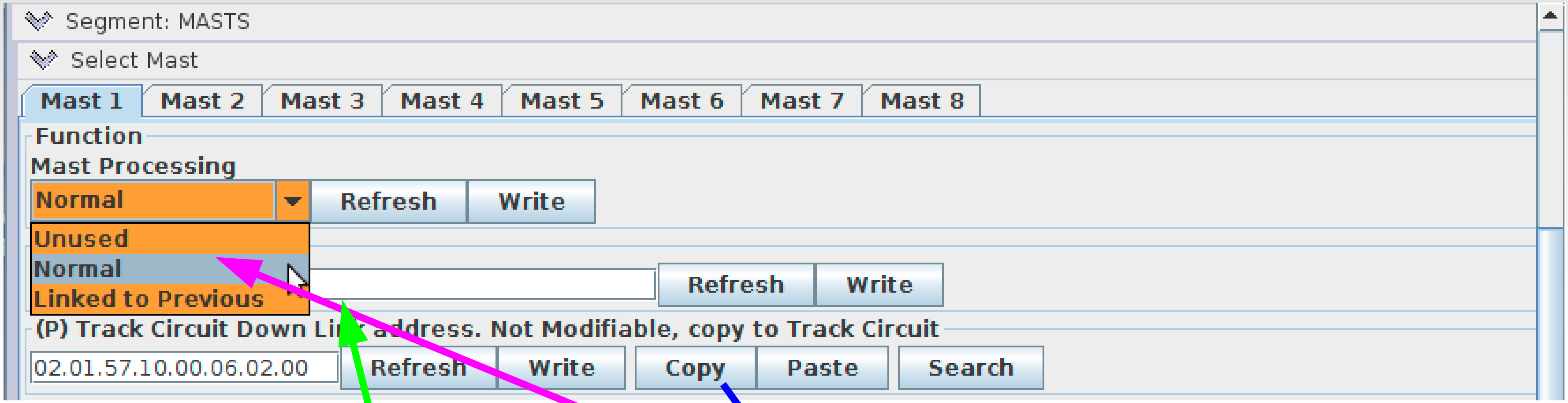
    2) A mast may only set a single speed limit at a time. This ’Speed’ is the currently allowed speed for going past the mast.

- Making a mast ’Linked to Previous’ carries the above assumptions over from any previous mast/masts. Speed is always taken from the first mast.

Signal Masts

- LED Drivers
- Different colors of LEDs have different voltage drops. This drop is subtracted from the drive voltage when calculating the series resistance. A typical red LED operates at 1.9V and a green operates at 3.3V. This means that at 5V the red resistor drops 3.1V and the green resistor drops 1.7V. With the same resistor values, the red LED will draw nearly twice the current as the green. Using a 12V source, the resistor voltage drops are 10.1V and 8.7V respectively, or just a 15% difference in current.
- Sometimes it is easiest to wire 2 LEDs in series for Position Light or Color Position Light signals. The voltage drops of green and yellow LEDs make it difficult or impossible to drive these with 5V supplies. As a result all of our RR-CirKits signal driver boards have always supplied 10V or more to the drive circuits.
- Brightness settings help you match the intensity of LEDs in the same mast.

Signal Mast Setup



Segment: MASTS

Select Mast

Function

Mast Processing: Normal

Unused
Normal
Linked to Previous

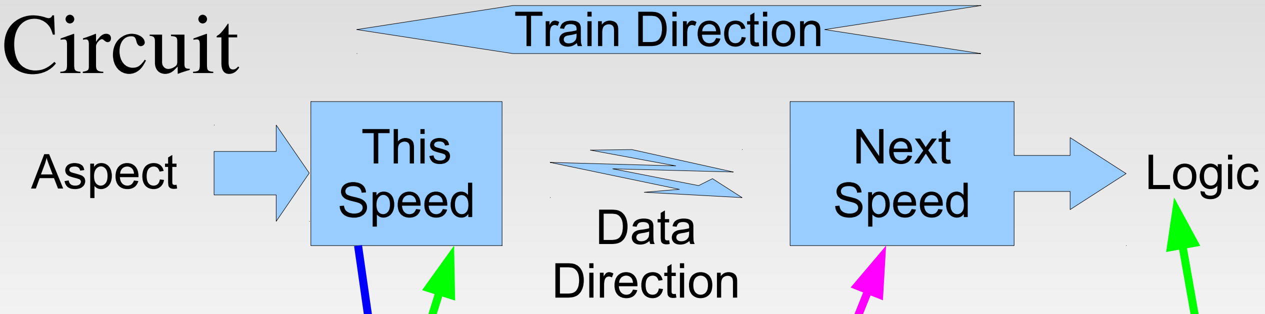
(P) Track Circuit Down Link address. Not Modifiable, copy to Track Circuit

02.01.57.10.00.06.02.00

- Function
- To use a mast you must first change it to ’Normal’ or ’Linked to Previous’.
- Next give it a Mast ID so you can easily find it again later. This could be a CTC panel number, a mile marker, a control point name, etc.
- Track Circuit Down Link Address. This fixed EventID is used as a pointer to the current track speed setting for this mast. Copy this number to any track circuit receive (RX) table to make it easy for logic to follow speed.

Signal Masts

- Track Circuit



Train Direction

Aspect

This Speed

Data Direction

Next Speed

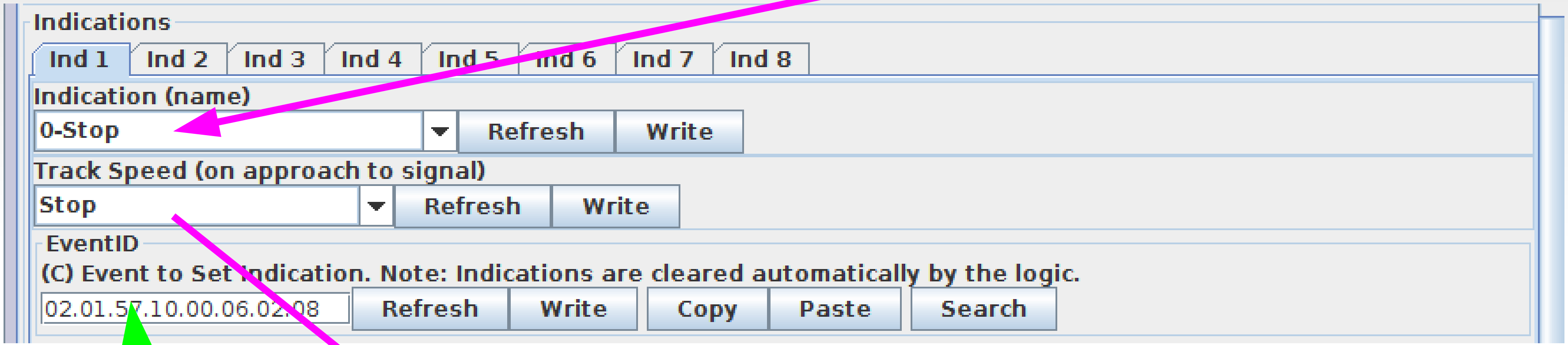
Logic

When calculating signal rules, the most important information from the next signal is the required track speed on approaching that signal. In many cases this information is actually a part of the rule name.

- In modular layout setup, getting this information easily from module to module is the single biggest roadblock to installing authentic signaling. Our Virtual Track Circuit concept is designed to simplify this.
- To link the speeds selected on a mast to the logic of another mast, simply copy the ’Track Circuit Down Link address’ from one mast, and paste it into the ‘Remote Mast Up Link address’ of another. This automatically makes the speed information from one mast available to the logic of another mast without requiring the entry of specific EventID information for each speed change into the appropriate logic conditionals.

Indications (Name)

- Indications tell the crew what to do at a signal. The ’Rule’ or ’Name’ is the shorthand for the Indication.



Indication (name): 0-Stop

Track Speed (on approach to signal): Stop

EventID

(C) Event to Set Indication. Note: Indications are cleared automatically by the logic.

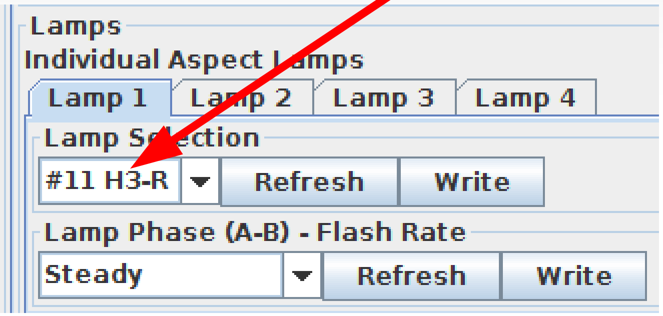
02.01.57.10.00.06.02.08

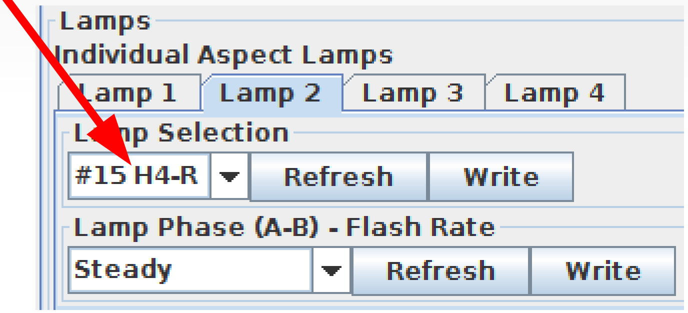
The selected ’Track Speed’ (one of eight possible) is the value that will be sent back to the previous signal over the Virtual Track Circuit. If the names don’t match your rules, simply pick something similar. Its just a number to the track circuit.

- EventID to Set Indication. This is the EventID used by the signal logic to set this signal rule. Remember that the logic may be in a node or JMRI.

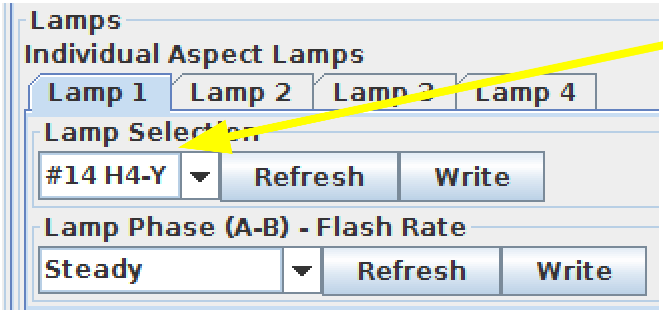
Lamps

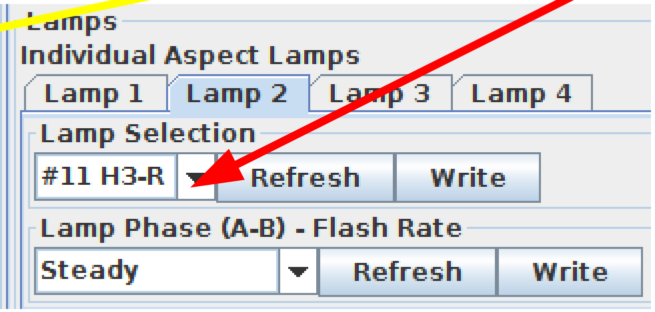
- The bottom line in displaying an aspect is to choose what lamps are lit. After all, that is what the crew (and the visitors) actually see.
- The mast on the signal bridge is dual head searchlights. This means ’Stop’ will display Red over Red. Choose the appropriate lamps to show this.


Lamps
Individual Aspect Lamps
Lamp 1
Lamp Selection
#11 H3-R
Lamp Phase (A-B) - Flash Rate
Steady


Lamps
Individual Aspect Lamps
Lamp 2
Lamp Selection
#15 H4-R
Lamp Phase (A-B) - Flash Rate
Steady

To show Indication 2 - ’Approach’ display Yellow over Red.


Lamps
Individual Aspect Lamps
Lamp 1
Lamp Selection
#14 H4-Y
Lamp Phase (A-B) - Flash Rate
Steady


Lamps
Individual Aspect Lamps
Lamp 2
Lamp Selection
#11 H3-R
Lamp Phase (A-B) - Flash Rate
Steady

- Continue in like manner until you have entered each possible aspect.

Lamps

- Each Indication (Aspect) can be displayed with as many as four lamps. If you have a rare signal aspect that can not be shown with just 4 lighted lamps then you can make a duplicate mast to light any additional lamps. Remember dual lamps that light together only count as a single lamp. (e.g. in Position Lights and Color Position lights) Only lighted lamps count. Only controlled lamps count. A marker that is always lighted can simply be powered on full time.
- Lamp Phase – Flash Rate may be used to flash signals automatically. One common example is ’Advance Approach’ which is commonly displayed with a flashing yellow lamp. Setting an appropriate Flash Rate means that the signal logic doesn’t need to worry about flashing the signal or overloading the bus with unecessary traffic. Providing both A and B phase options is handy for grade crossing flashers or other alternating lamp situations.

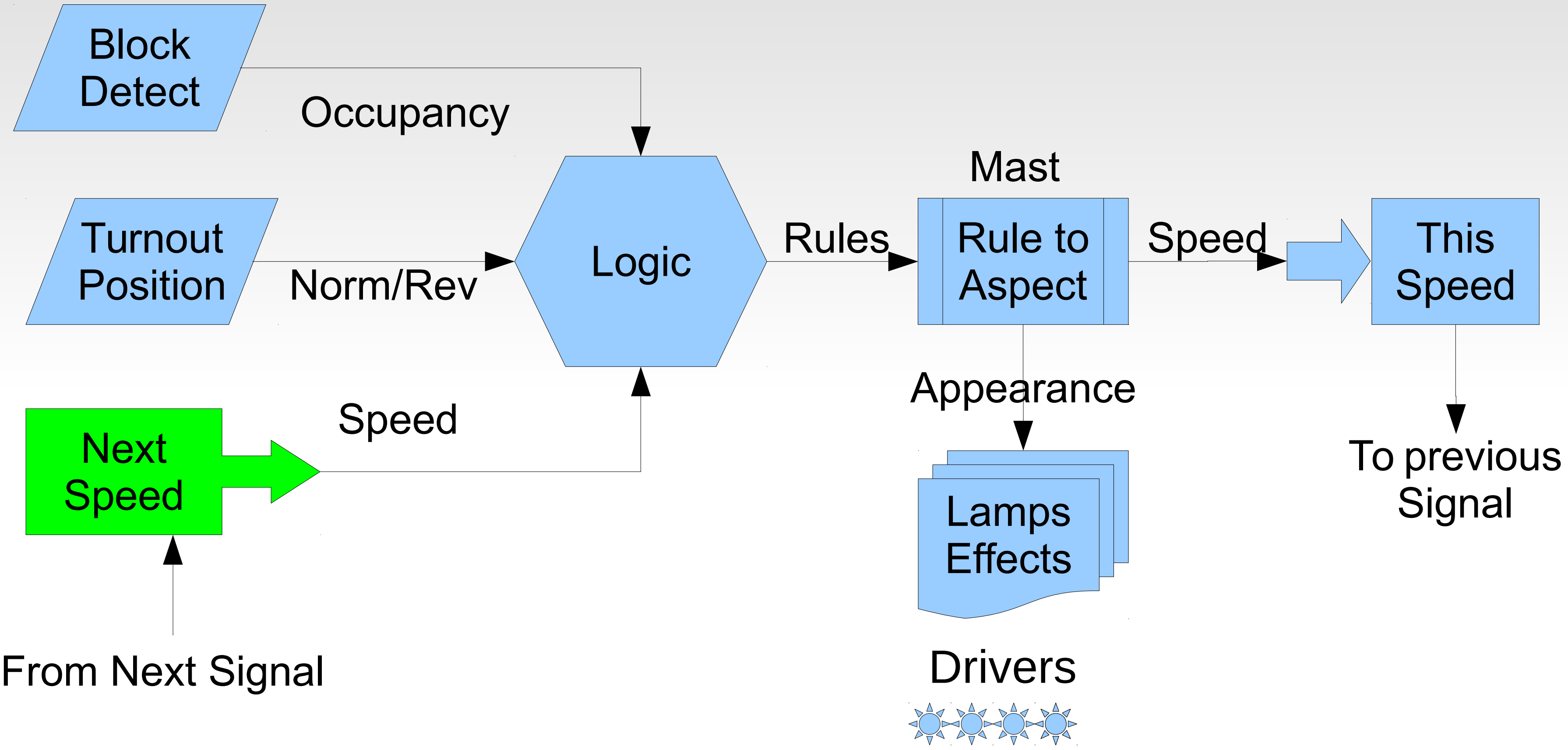
Lamps – Special Effects

- Incandecent fade. Signal lamps are wired differently than standard household lamps. They include a ballast resistor in series with the lamp. This ballast serves two purposes. One is simply to set the brightness of the lamp for maximum life. More importantly when the cold lamp is first powered up it prevents the normal inrush current by dropping most of the voltage across the ballast until the lamp warms up. The visual result of this is that a signal does not blink on rapidly. In fact signals fade on slowly enough to be noted. Of course even houshold incandecents fade off slowly as the lamps cool down again.
- Transition effects. The B&O signal clip we saw earlier shows an interesting transition between Clear and Stop. Not only does it show the fade up and down, but it interjects a brief ’Approach’ into the change. This is an artifact of the relay circuits used to prevent showing a red during green/yellow changes. Selecting ’Transition Down’ as a special effect on ’Stop’ will allow you to do this. (and wow your rivet counting crew)

Lamps – Special Effects

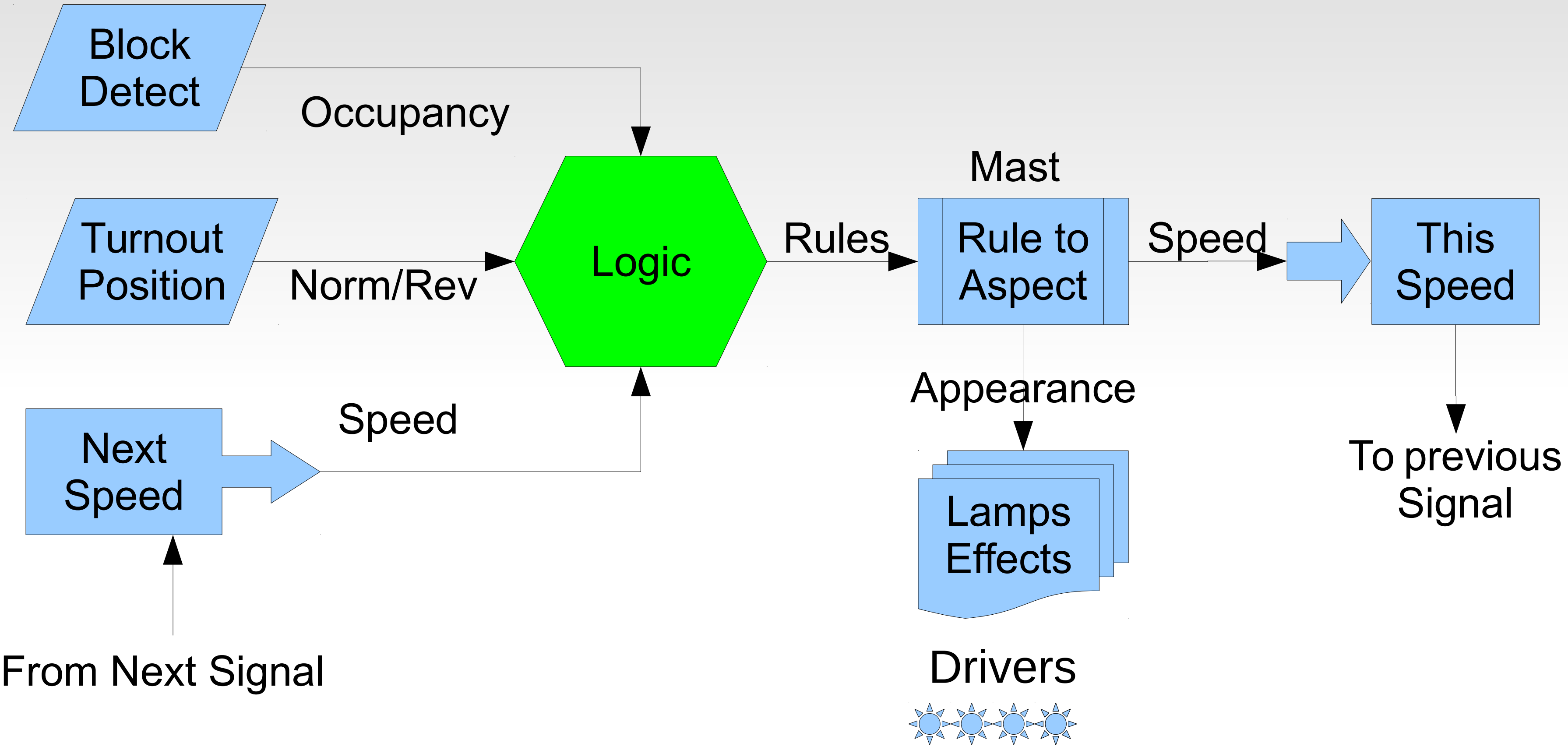
- H2 Red Flicker. Many of you know that real searchlight signals (not just the H2) have an internal arm that swings back and forth in front of the lamp. It hangs by gravity with a red roundel in center position. Displaying either green or yellow requires swinging the arm out of its center position with electromagnets. Not quite as obvious is the fact that you can not change between yellow and green without passing the red between them. This causes the red flash. The other part of the effect is that the arm is free swinging and during a change it will overshoot its position as it settles down. This swings the color roundels past their normal positions which causes the signal to appear to flicker. To show this effect properly on multiple heads use a different ’mast’ for each head and use the same events to drive each ’mast’.
- Strobe lights can be found around the layout. Sometimes it is nice to be able to utilize unused signal outputs for other purposes.

Track Circuits



Paste the next masts ‘Up Link EventID’ from any mast to a track circuit. This creates a virtual link directly into the logic variables by virtual name rather than by using actual event numbers. The logic for a mast can be setup, or mass produced, without knowing any actual mast IDs ahead of time.

Signal Logic



We have covered all the edges. Now we can talk about the central subject, Signal Logic itself.

Signal Logic

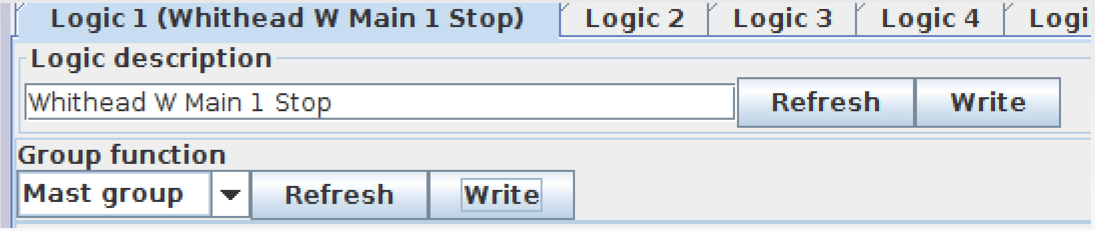
- Signal Logic is just a series of conditions (called conditionals) that are checked to see what signal rule should currently be in effect.
- Logic conditionals should be easy to cascade with the calculations for the most restrictive rules having priority over less restrictive rules. We do this by checking each conditionals in order from top down. Any rule that is found to be true first checks for any more restrictive rule still in effect. (which exits processing if found) Then it sends its appropriate events, and finally skips over any less restrictive rules for the mast.
- Any conditional may directly send up to 4 events representing signal rules (or anything) when it is found to be true. (or false) A cascade option allows even more events to be sent in special situations. Note: this logic may be used for many other purposes than just calculating signal aspects.

Signal Logic

- Logic Functions consist of the usual AND, OR, XOR operators. In addition there are two ’change’ operators. These change the true/false sense of a conditional based on the AND and OR of the variables.
- Additionally we have added a non-standard logic operator called ’AND Then’. This makes it very easy to keep track of train direction. You can simply watch two block detectors and determine train direction by the order in which they are activated.
- A recent addition is the ability to control the action associated with both true and false evaluations of a conditional. These options are to ’Send then Exit Group’, ‘Send then Evaluate Next’, ’Send then Send Next’, ‘Exit Group’, and ’Evaluate Next’. The ‘Send then Send Next’ automatically goes to the next conditional and always treats it as if it were true. This makes it easy to send more than 4 events from a single conditional.

Logic Conditionals

- Normally Signal Logic Conditionals will have a Group function of ’Mast Group’ or else ’Last’.



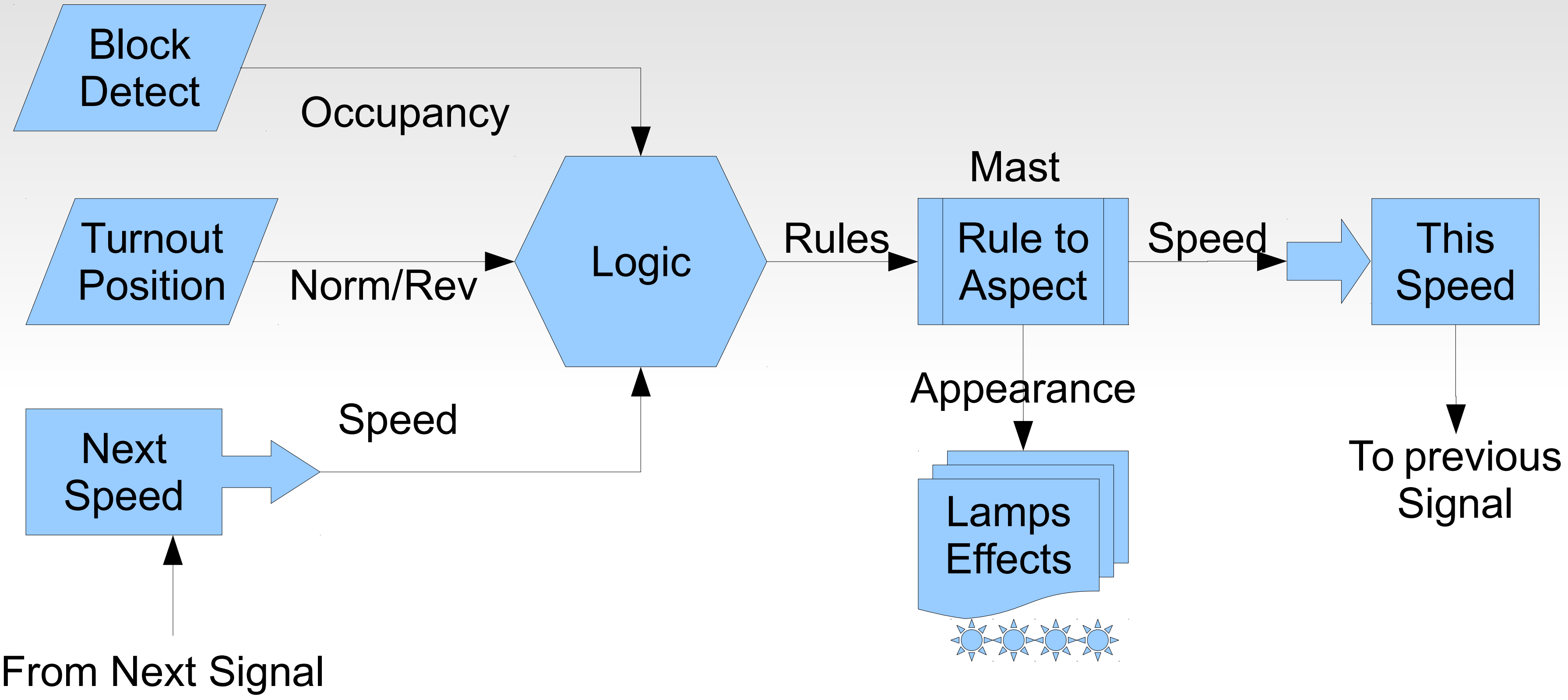
Logic description: Whithead W Main 1 Stop

Group function: Mast group

- The function of a conditional ’Mast Group’ is to pick the most restrictive rule for a mast and send it to the mast table for conversion to the proper aspect.

  It does this by sending the EventID for the first rule that is true, then skipping forward until it is past the last item in the group. As long as you remember to order your checks from most restrictive to least restrictive this works.

Signal Logic Example



The basic signal logic overview.

- Rule logic is calculated using layout status information and next signal speed.
- The resulting ‘Rules’ are converted to lighted lamps, effects, and speeds.

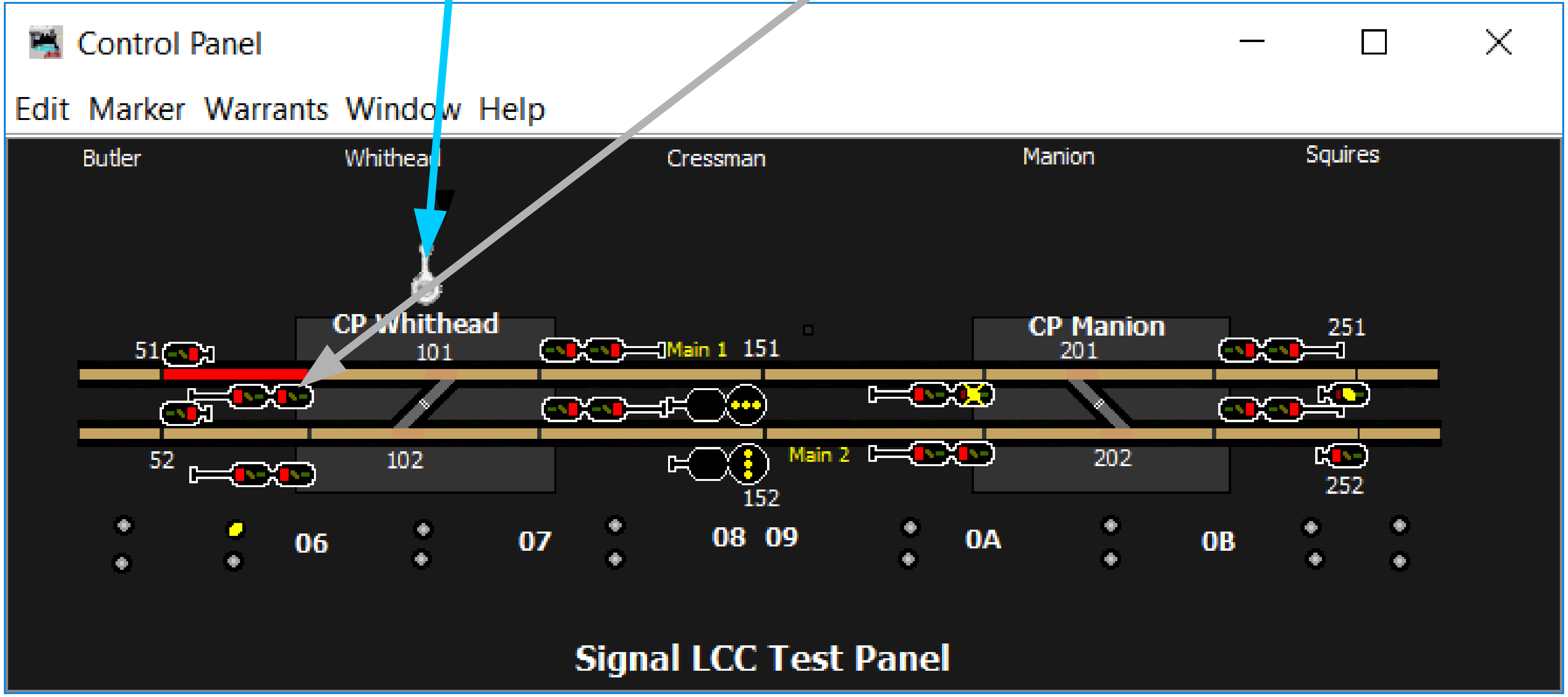
Trailing Point Signal Logic

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / OS occupied	**Not** CTC-Right	OR	OS BOD	Stop	Stop
Siding selected / Main occupied	Turnout Reverse	OR	Main BOD	Stop	Stop
Next Main Stop	Main Mast Stop	null		Approach	Medium
Next Main Medium	Main Mast Medium	null		Approach Medium / Advance Approach	Clear
Next Main Clear	Main Mast Clear	null		Clear	Clear

- To create ’**Not** CTC-Right’ simply reverse the events controlling ’Variable 1’ for that conditional. This data is from the direction lever on a CTC panel.
- First we check for the wrong CTC direction, is the turnout set against us, is the OS occupied, or the track past the turnout occupied. Any of these will set the signal to Stop.
- If the signal has not been set to Stop, then we check to see if the next signal’s speed is ’Stop’. (Main Mast Stop) If so we set this signal to ’Approach’ with a speed of ’Medium’. (or ’Approach’) It is helpful to realize that ’Approach’, when used by itself, is short hand for ’Approach Stop’.
- If the next signal is not Stop, then we check for the next signal’s speed of ’Medium’ (or ’Approach’) and set our aspect appropriately.
- Finally, finding nothing more restrictive, we can set it to ’Clear’.

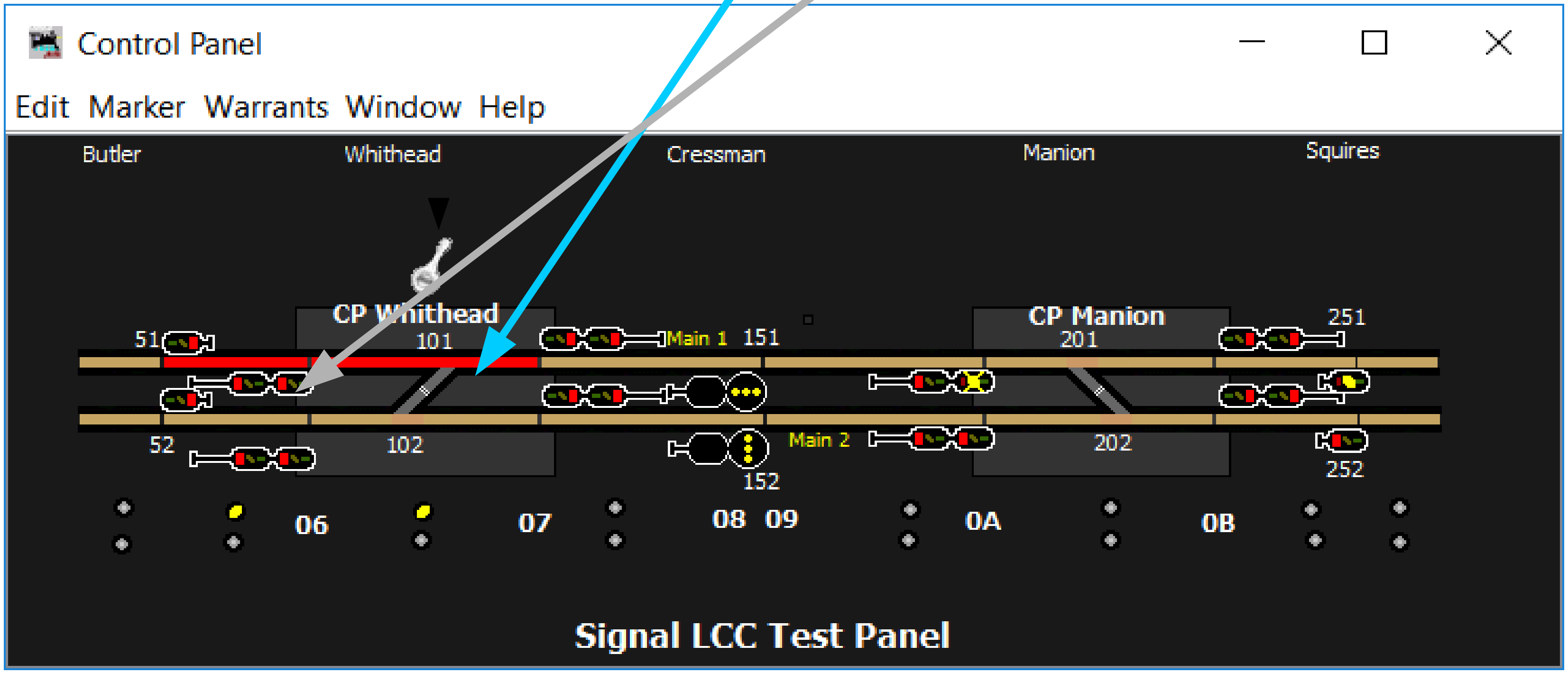
101R

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / OS occupied	**Not** CTC-Right RM1	OR	WH M1 OS BOD	Stop	Stop
Siding selected / Main occupied	101 Reverse	OR	WH East M1 BOD	Stop	Stop
Next Main Stop	151R Stop	null		Approach	Medium
Next Main Medium	151R Medium	null		Approach Medium / Advance Approach	Clear
Next Main Clear	151R Clear	null		Clear	Clear



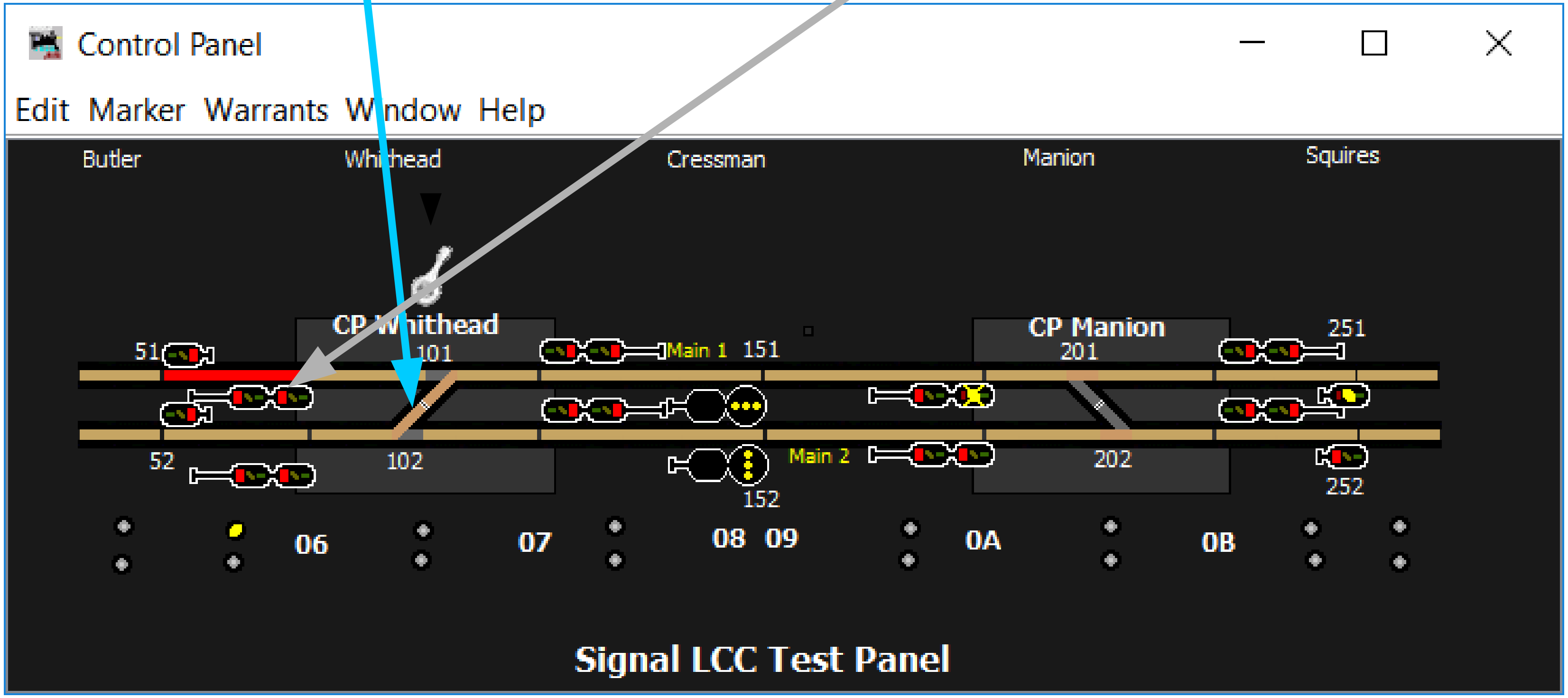
101R

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / OS occupied	**Not** CTC-Right RM1	OR	WH M1 OS BOD	Stop	Stop
Siding selected / Main occupied	101 Reverse	OR	WH East M1 BOD	Stop	Stop
Next Main Stop	151R Stop	null		Approach	Medium
Next Main Medium	151R Medium	null		Approach Medium / Advance Approach	Clear
Next Main Clear	151R Clear	null		Clear	Clear



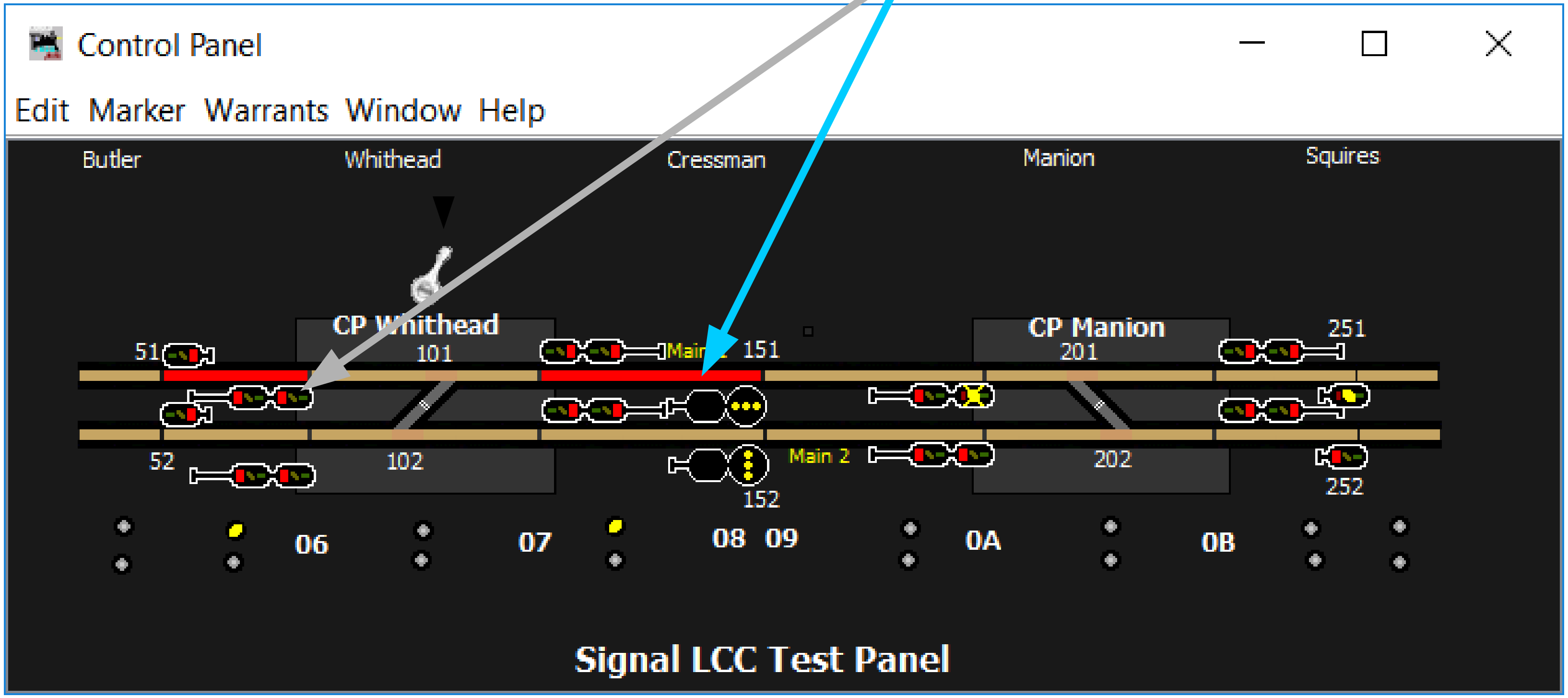
101R

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / OS occupied	**Not** CTC-Right RM1	OR	WH M1 OS BOD	Stop	Stop
Siding selected / Main occupied	101 Reverse	OR	WH East M1 BOD	Stop	Stop
Next Main Stop	151R Stop	null		Approach	Medium
Next Main Medium	151R Medium	null		Approach Medium / Advance Approach	Clear
Next Main Clear	151R Clear	null		Clear	Clear



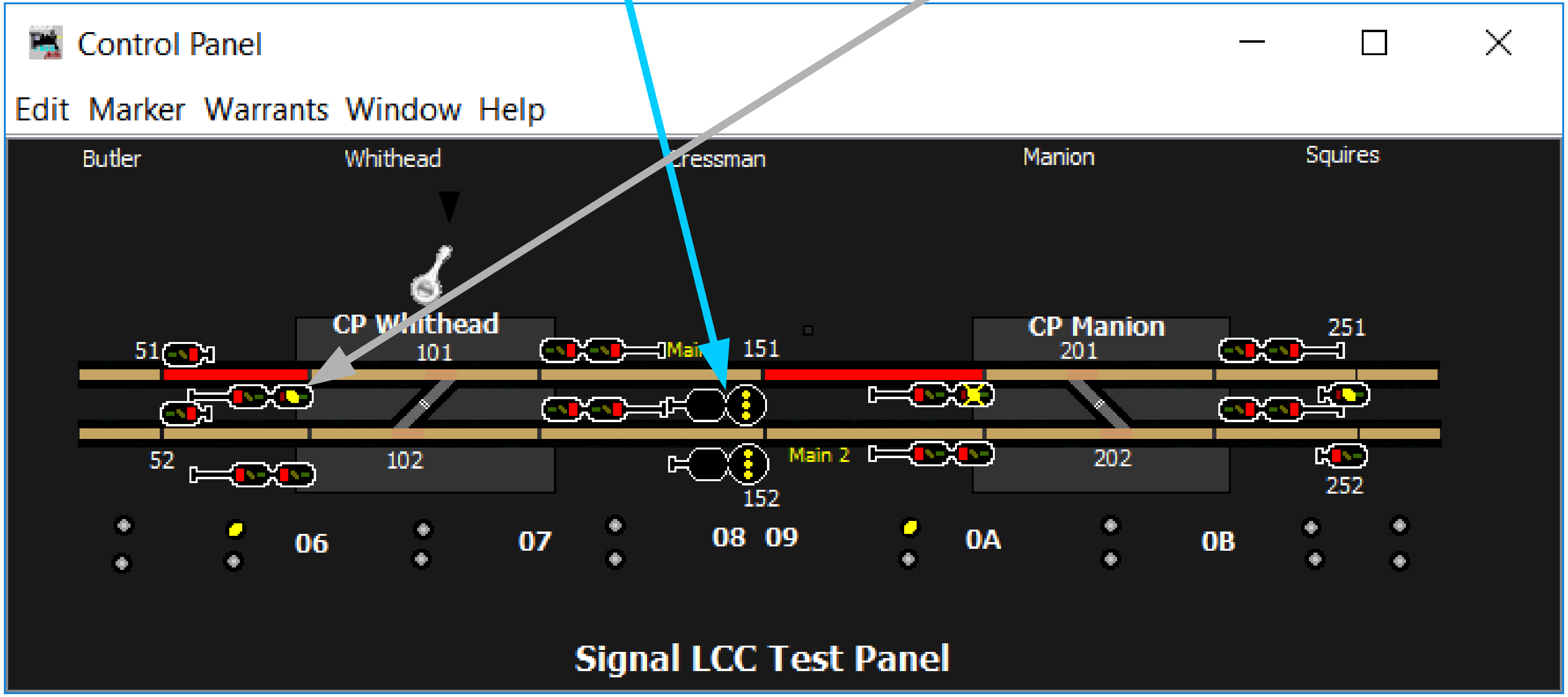
101R

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / OS occupied	**Not** CTC-Right RM1	OR	WH M1 OS BOD	Stop	Stop
Siding selected / Main occupied	101 Reverse	OR	WH East M1 BOD	Stop	Stop
Next Main Stop	151R Stop	null		Approach	Medium
Next Main Medium	151R Medium	null		Approach Medium / Advance Approach	Clear
Next Main Clear	151R Clear	null		Clear	Clear



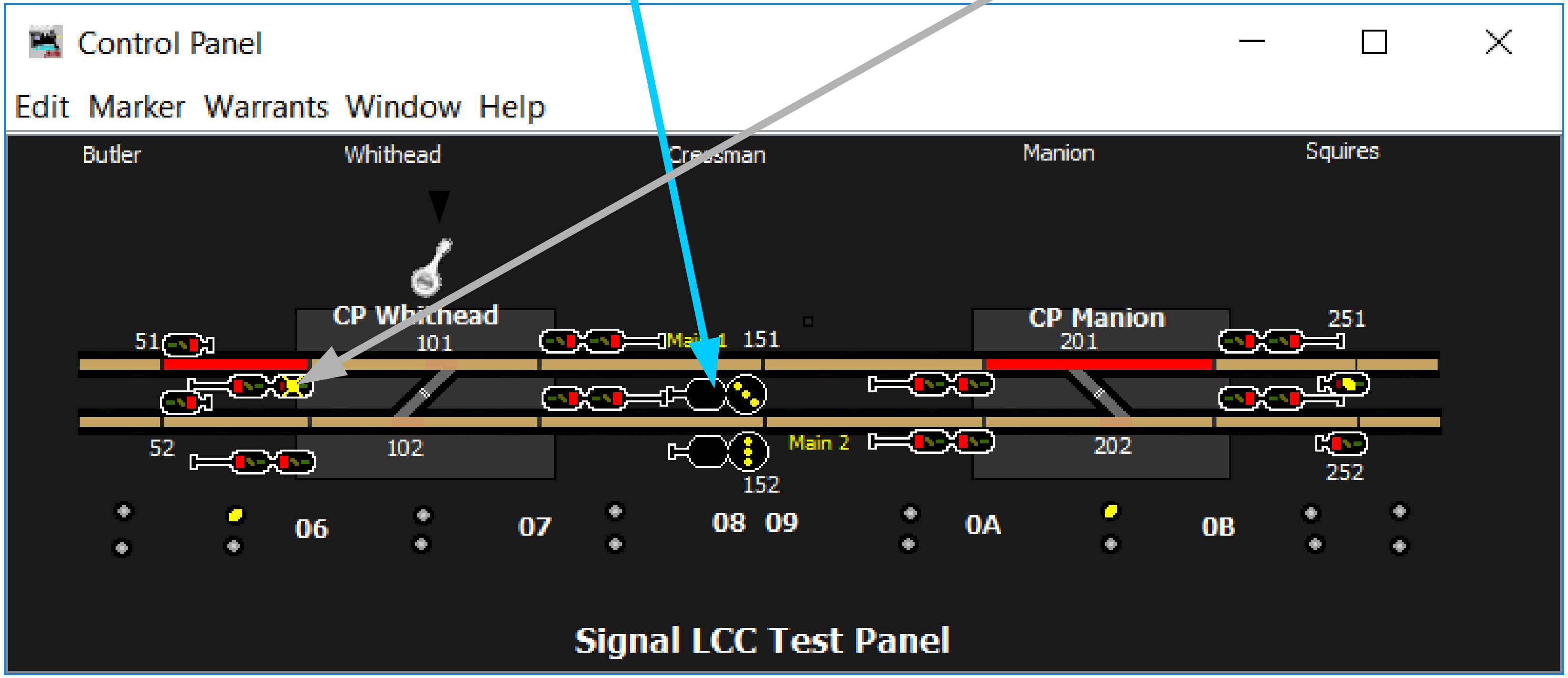
101R

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / OS occupied	**Not** CTC-Right RM1	OR	WH M1 OS BOD	Stop	Stop
Siding selected / Main occupied	101 Reverse	OR	WH East M1 BOD	Stop	Stop
Next Main Stop	151R Stop	null		Approach	Medium
Next Main Medium	151R Medium	null		Approach Medium / Advance Approach	Clear
Next Main Clear	151R Clear	null		Clear	Clear



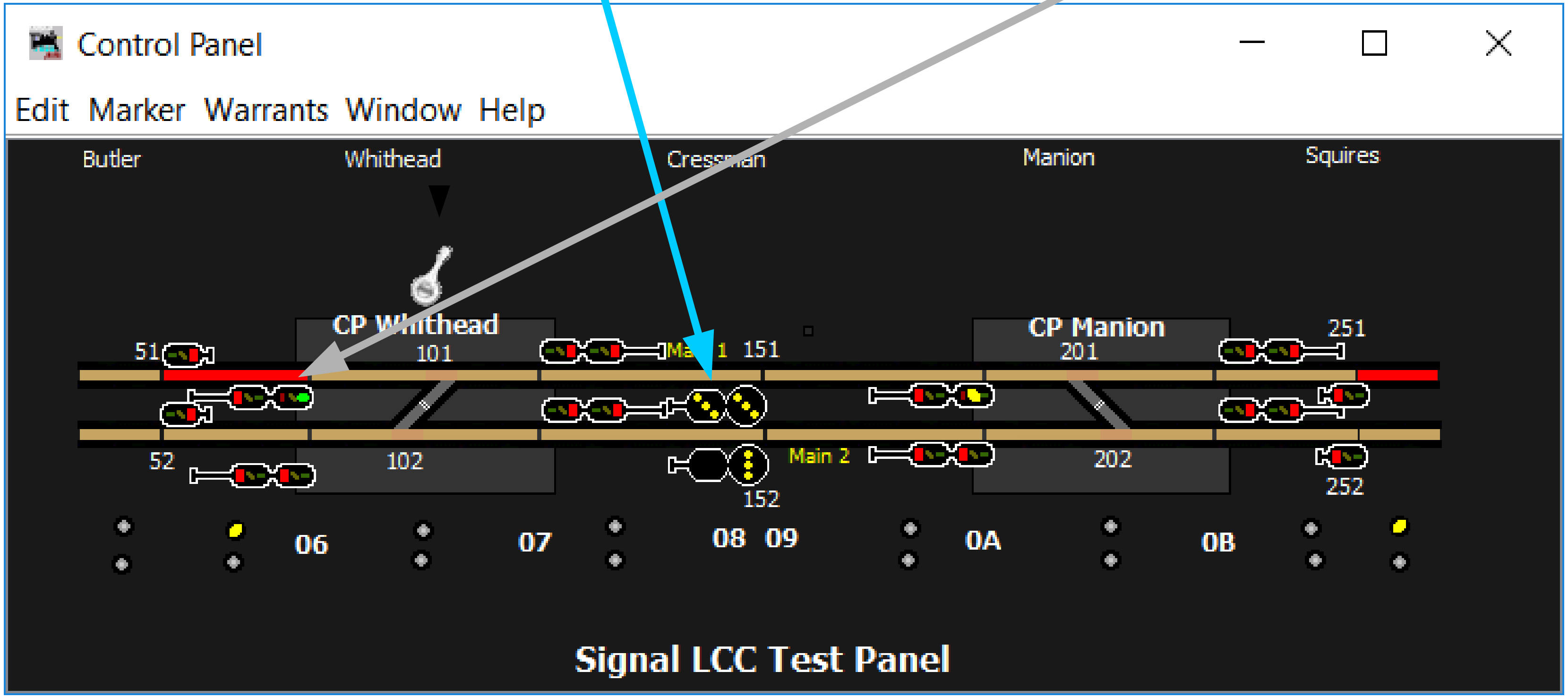
101R

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / OS occupied	**Not** CTC-Right RM1	OR	WH M1 OS BOD	Stop	Stop
Siding selected / Main occupied	101 Reverse	OR	WH East M1 BOD	Stop	Stop
Next Main Stop	151R Stop	null		Approach	Medium
Next Main Medium	151R Medium	null		Approach Medium / Advance Approach	Clear
Next Main Clear	151R Clear	null		Clear	Clear



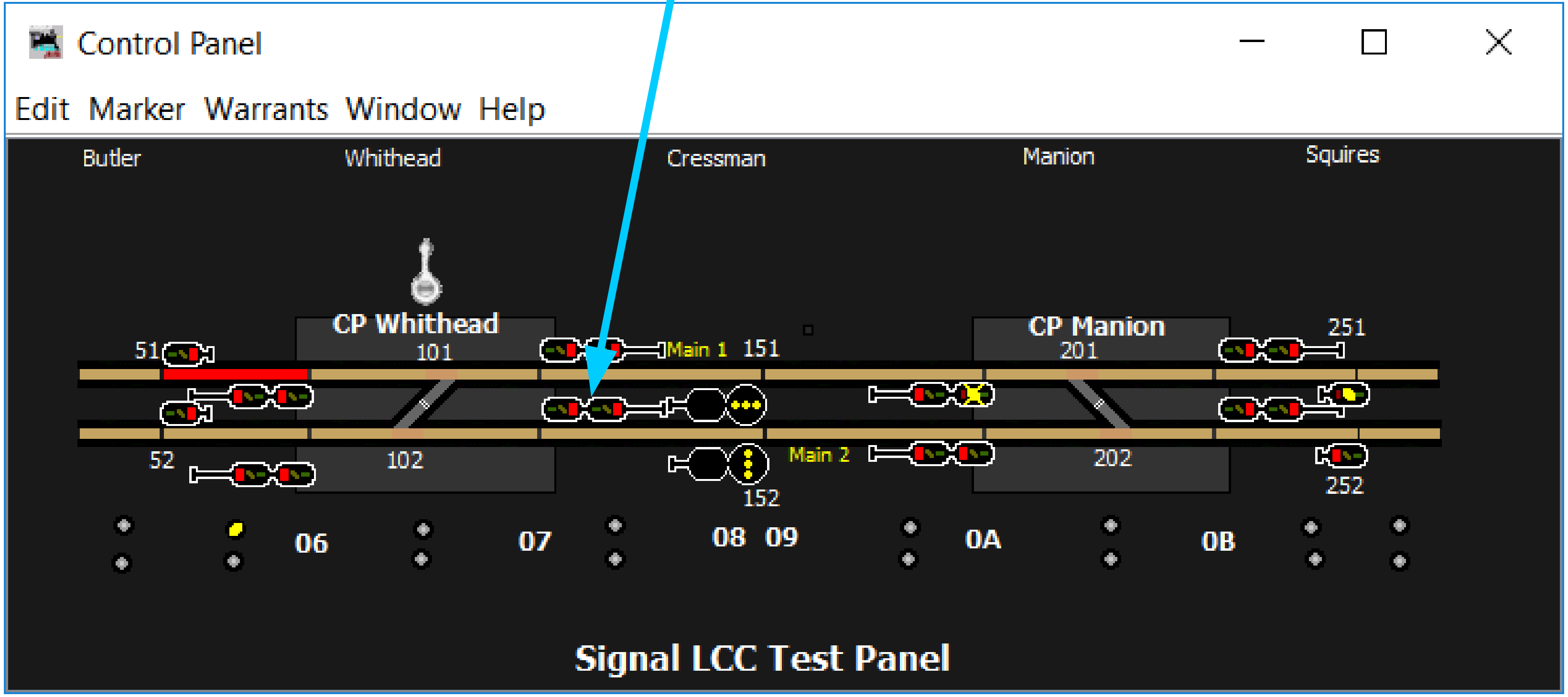
101R

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / OS occupied	**Not** CTC-Right RM1	OR	WH M1 OS BOD	Stop	Stop
Siding selected / Main occupied	101 Reverse	OR	WH East M1 BOD	Stop	Stop
Next Main Stop	151R Stop	null		Approach	Medium
Next Main Medium	151R Medium	null		Approach Medium / Advance Approach	Clear
Next Main Clear	151R Clear	null		Clear	Clear



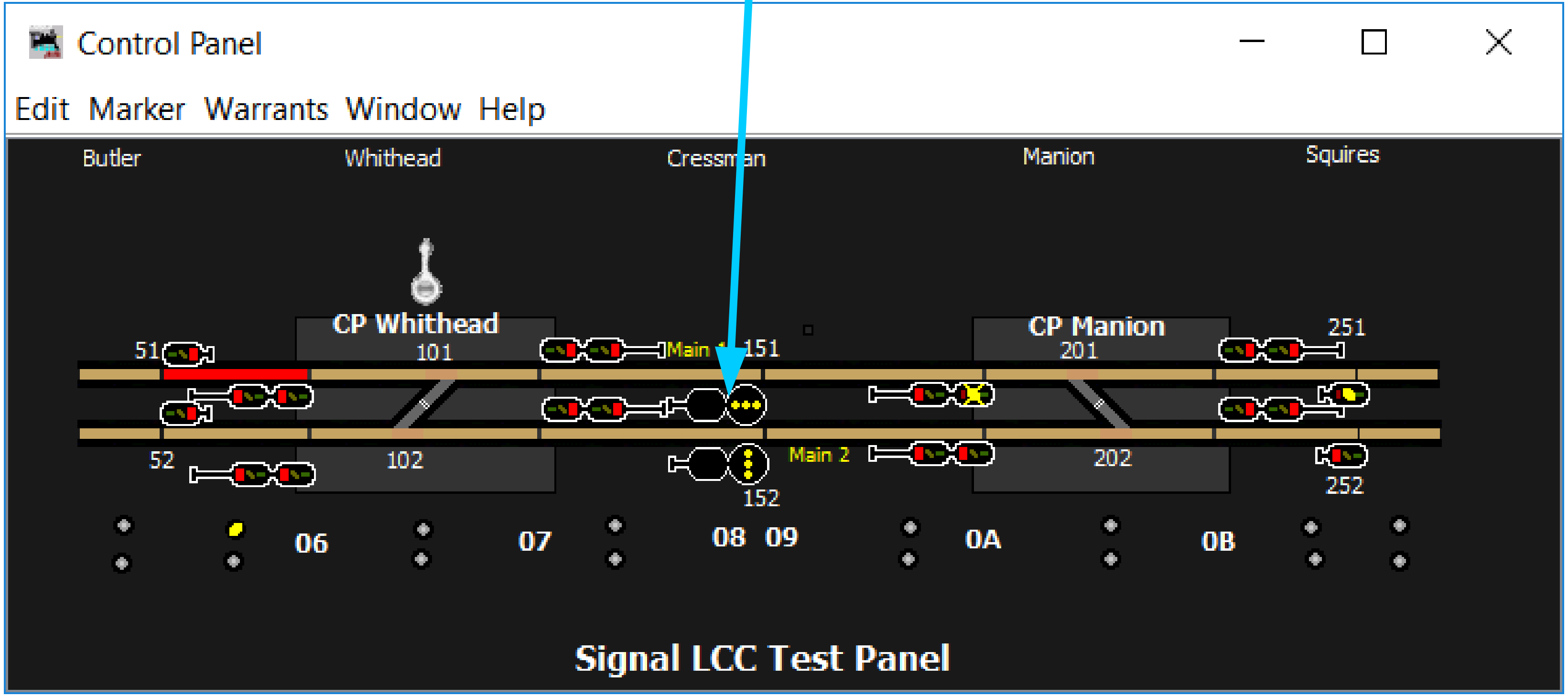
102L

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / OS occupied	**Not** CTC-Left LM2	OR	WH M2 OS BOD	Stop	Stop
Siding selected / Main occupied	102 Reverse	OR	WH West M2 BOD	Stop	Stop
Next Main Stop	52R Stop	null		Approach	Medium
Next Main Medium	52R Medium	null		Approach Medium / Advance Approach	Clear
Next Main Clear	52R Clear	null		Clear	Clear



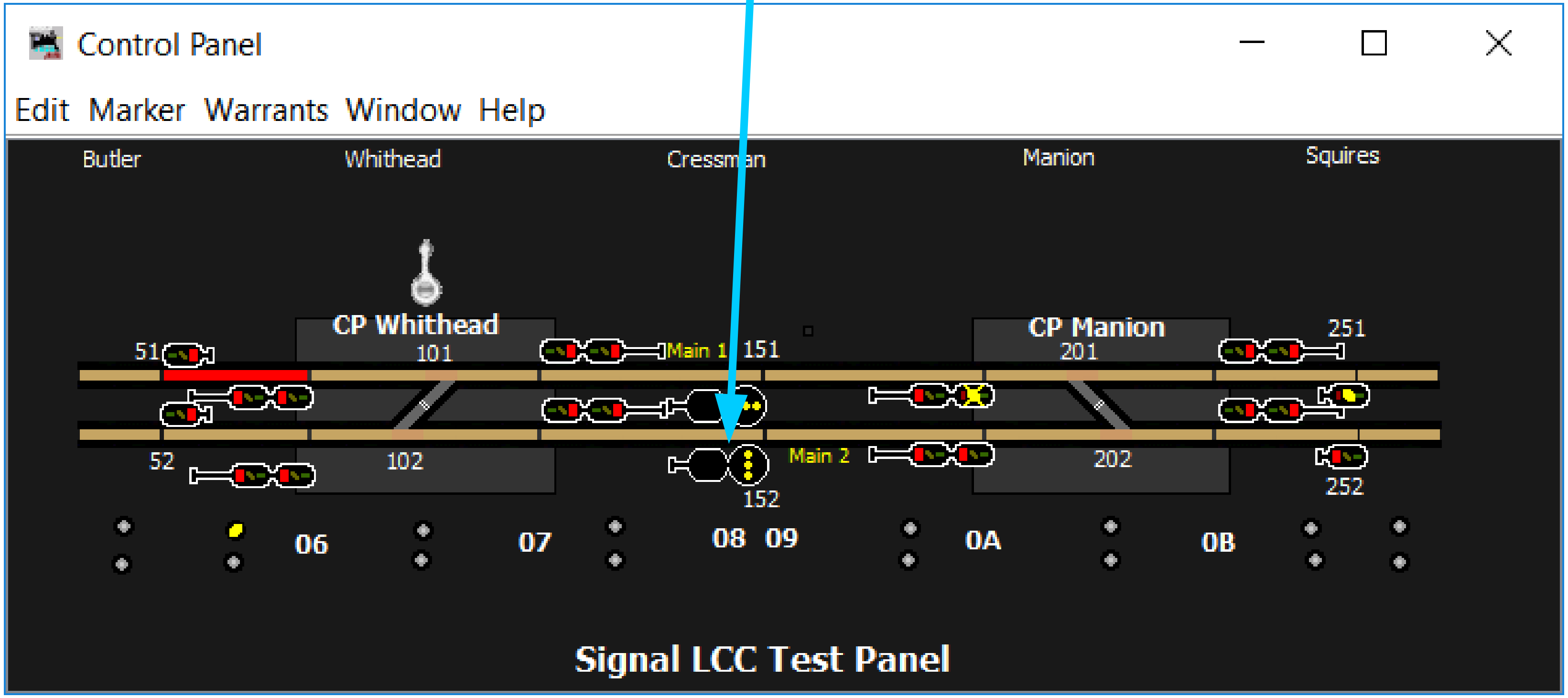
151R

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / Main occupied	**Not** CTC-Right RM1	OR	CM East M1 BOD	Stop	Stop
Next Main Stop	201R Stop	null		Approach	Medium
Next Main Medium	201R Medium	null		Approach Medium / Advance Approach	Clear
Next Main Clear	201R Clear	null		Clear	Clear



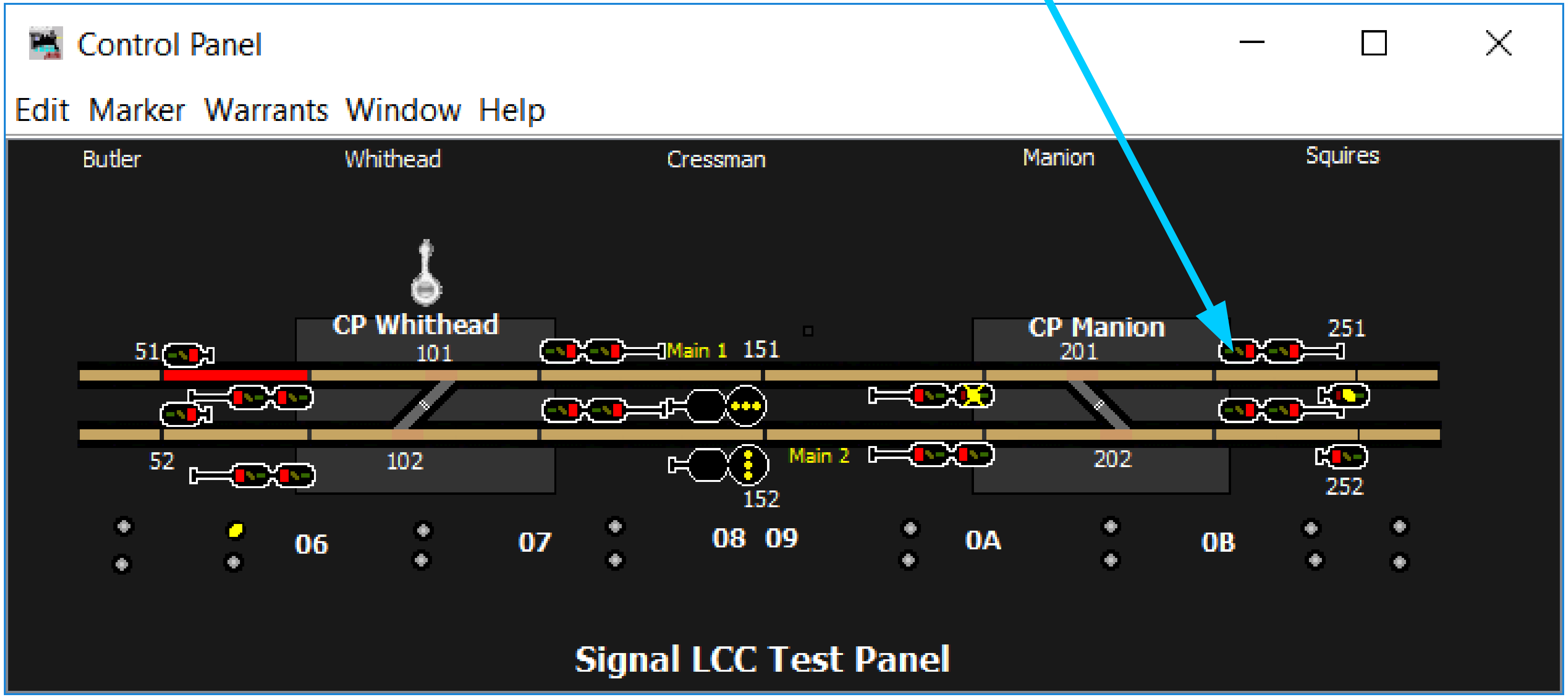
152R

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / Main occupied	**Not** CTC-Right RM2	OR	CM East M2 BOD	Stop	Stop
Next Main Stop	202R Stop	null		Approach	Medium
Next Main Medium	202R Medium	null		Approach Medium / Advance Approach	Clear
Next Main Clear	202R Clear	null		Clear	Clear



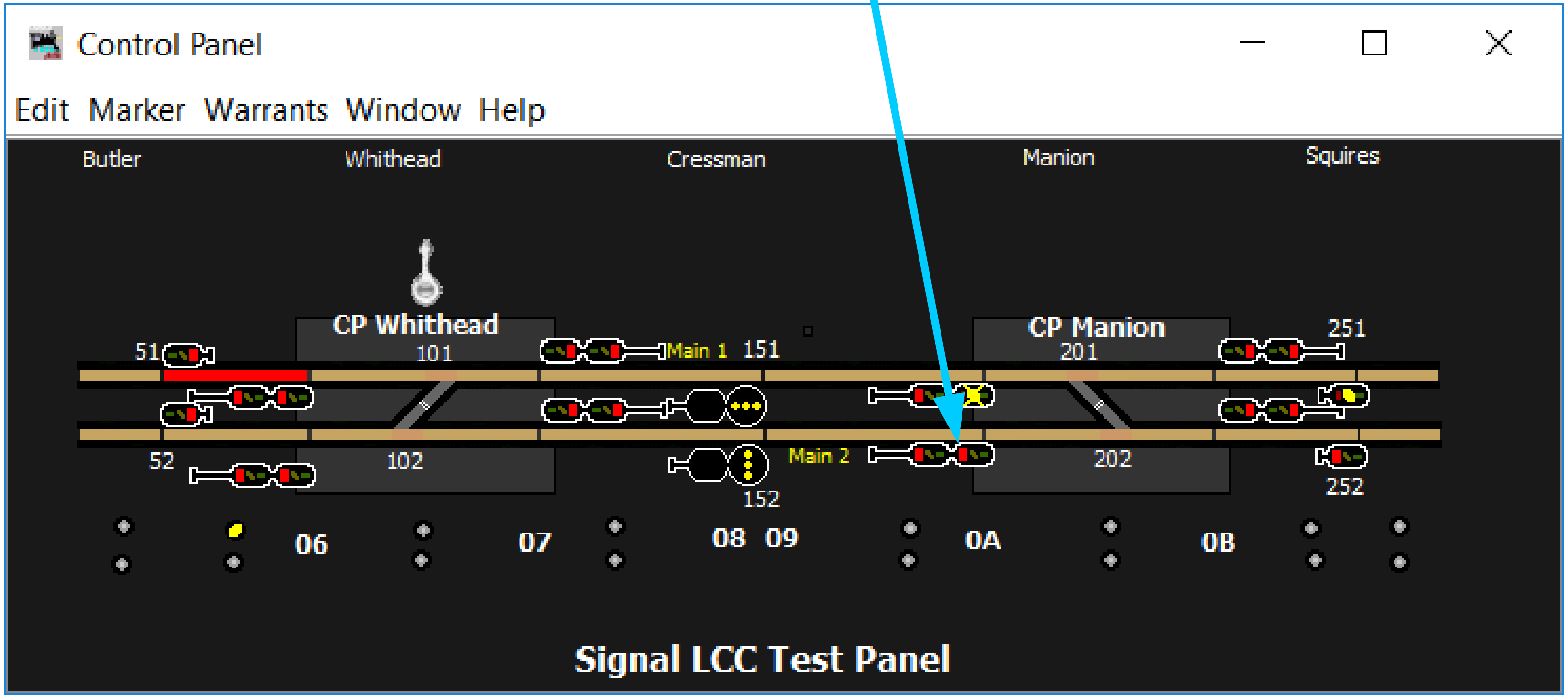
201L

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / OS occupied	**Not** CTC-Left LM1	OR	MA Main1 OS BOD	Stop	Stop
Siding selected / Main occupied	101 Reverse	OR	MA West M1 BOD	Stop	Stop
Next Main Stop	151L Stop	null		Approach	Medium
Next Main Medium	151L Medium	null		Approach Medium / Advance Approach	Clear
Next Main Clear	151L Clear	null		Clear	Clear



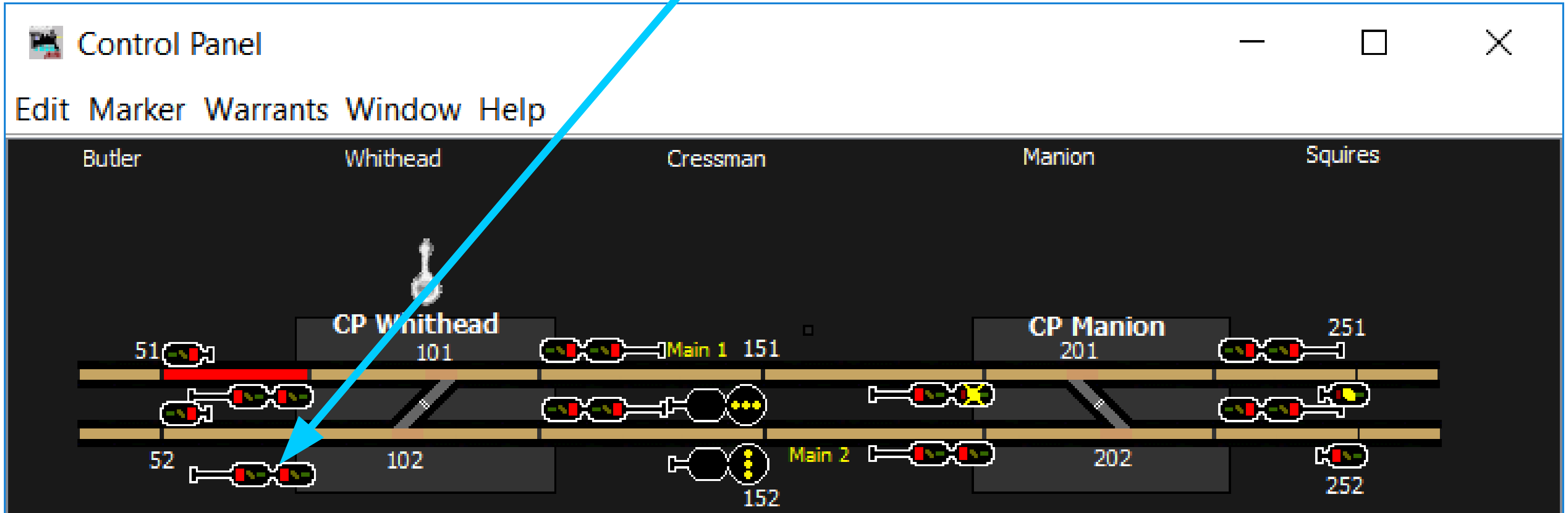
202R

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / OS occupied	**Not** CTC-Right RM2	OR	MA Main2 OS BOD	Stop	Stop
Siding selected / Main occupied	101 Reverse	OR	MA East M2 BOD	Stop	Stop
Next Main Stop	151L Stop	null		Approach	Medium
Next Main Medium	151L Medium	null		Approach Medium / Advance Approach	Clear
Next Main Clear	151L Clear	null		Clear	Clear



Facing Point Logic 102R

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right	**Not** CTC-Right RM2	null		Stop	Stop
Next CTC Left - Rev	Turnout 102 Reverse	AND	Next CTC-Left	Stop	Stop
Next CTC Left - Norm	Turnout Normal	AND	Next CTC-Left	Stop	Stop
OS occupied	OS WH Main 2 BOD	null		Stop	Stop
Siding occupied	Turnout 102 Reverse	AND	WH East M1 BOD	Stop	Stop
Main occupied	Turnout 102 Normal	AND	WH East M2 BOD	Stop	Stop
Next Siding Stop	Turnout 102 Reverse	AND	151R Stop	Medium Approach	Medium
Next Main Stop	Turnout 102 Normal	AND	152R Stop	Approach	Medium
Next Siding Medium	Turnout 102 Reverse	AND	151R Medium	Medium Approach Medium	Medium
Next Main Clear	Turnout 102 Normal	AND	152R Clear	Clear	Clear



Facing Point Signal Logic

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right	Not CTC-Right	null		Stop	Stop
Next CTC Left - Rev	Turnout Reverse	AND	Next CTC-Left	Stop	Stop
Next CTC Left - Norm	Turnout Normal	AND	Next CTC-Left	Stop	Stop
OS occupied	OS BOD	null		Stop	Stop
Siding occupied	Turnout Reverse	AND	Siding BOD	Stop	Stop
Main occupied	Turnout Normal	AND	Main BOD	Stop	Stop
Next Siding Stop	Turnout Reverse	AND	Siding Mast Stop	Medium Approach	Medium
Next Main Stop	Turnout Normal	AND	Main Mast Stop	Approach	Medium
Next Siding Medium	Turnout Reverse	AND	Siding Mast Medium	Medium Approach Medium	Medium
Next Main Clear	Turnout Normal	AND	Main Mast Clear	Clear	Clear

- It should be clear from the above that calculating aspects for the signal prior to this interlocking is simplified by knowing the signal speeds, because there are five different aspects to check, but there are only three different speeds to check. The three different possible medium speed aspects do not cause any change in the signal prior to this one, so it only needs to show Clear, Approach Medium, (or Advance Approach) Approach, and Stop.

Facing Point Signal Logic

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right	Not CTC-Right	null		Stop	Stop
Next CTC Left - Rev	Turnout Reverse	AND	Next CTC-Left	Stop	Stop
Next CTC Left - Norm	Turnout Normal	AND	Next CTC-Left	Stop	Stop
OS occupied	OS BOD	null		Stop	Stop
Siding occupied	Turnout Reverse	AND	Siding BOD	Stop	Stop
Main occupied	Turnout Normal	AND	Main BOD	Stop	Stop
Next Siding Stop	Turnout Reverse	AND	Siding Mast Stop	Medium Approach	Medium
Next Main Stop	Turnout Normal	AND	Main Mast Stop	Approach	Medium
Next Siding Medium	Turnout Reverse	AND	Siding Mast Medium	Medium Approach Medium	Medium
Next Main Clear	Turnout Normal	AND	Main Mast Clear	Clear	Clear

- It should be clear from the above that calculating aspects for the signal prior to this interlocking is simplified by knowing the signal speeds, because there are five different aspects to check, but there are only three different speeds to check. The three different possible medium speed aspects do not cause any change in the signal prior to this one, so it only needs to show Clear, Approach Medium, (or Advance Approach) Approach, and Stop.

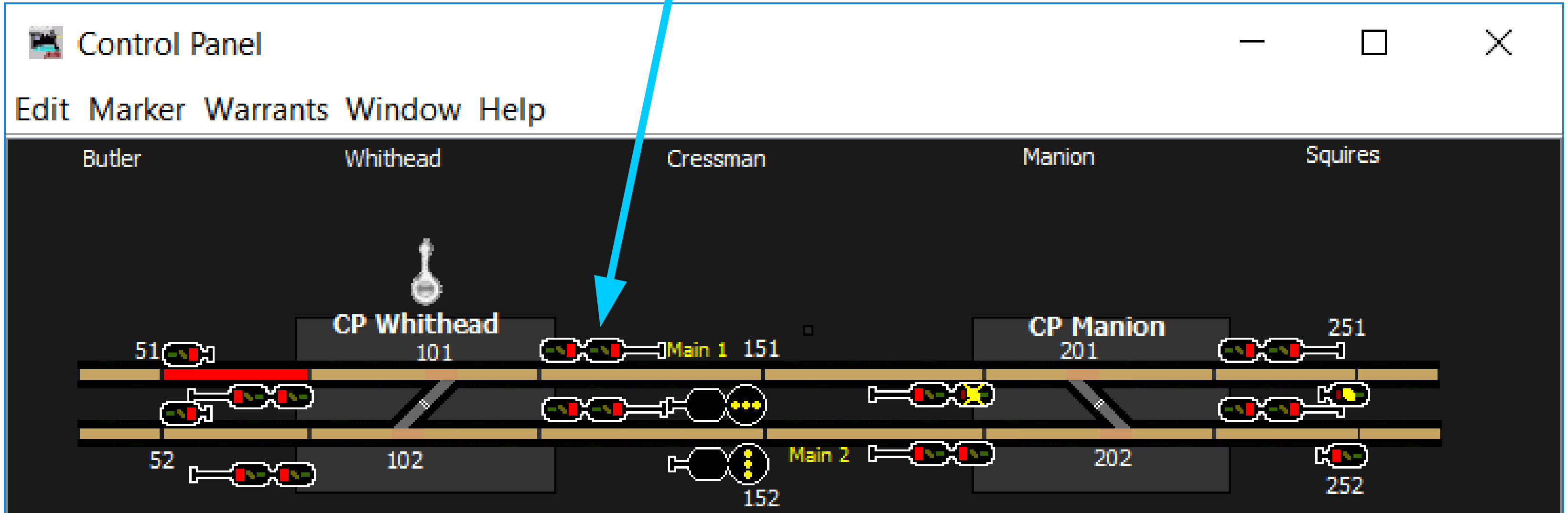
Facing Point Signal Logic

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right	Not CTC-Right	null		Stop	Stop
Next CTC Left - Rev	Turnout Reverse	AND	Next CTC-Left	Stop	Stop
Next CTC Left - Norm	Turnout Normal	AND	Next CTC-Left	Stop	Stop
OS occupied	OS BOD	null		Stop	Stop
Siding occupied	Turnout Reverse	AND	Siding BOD	Stop	Stop
Main occupied	Turnout Normal	AND	Main BOD	Stop	Stop
Next Siding Stop	Turnout Reverse	AND	Siding Mast Stop	Medium Approach	Medium
Next Main Stop	Turnout Normal	AND	Main Mast Stop	Approach	Medium
Next Siding Medium	Turnout Reverse	AND	Siding Mast Medium	Medium Approach Medium	Medium
Next Main Clear	Turnout Normal	AND	Main Mast Clear	Clear	Clear

- It should be clear from the above that calculating aspects for the signal prior to this interlocking is simplified by knowing the signal speeds, because there are five different aspects to check, but there are only three different speeds to check. The three different possible medium speed aspects do not cause any change in the signal prior to this one, so it only needs to show Clear, Approach Medium, (or Advance Approach) Approach, and Stop.

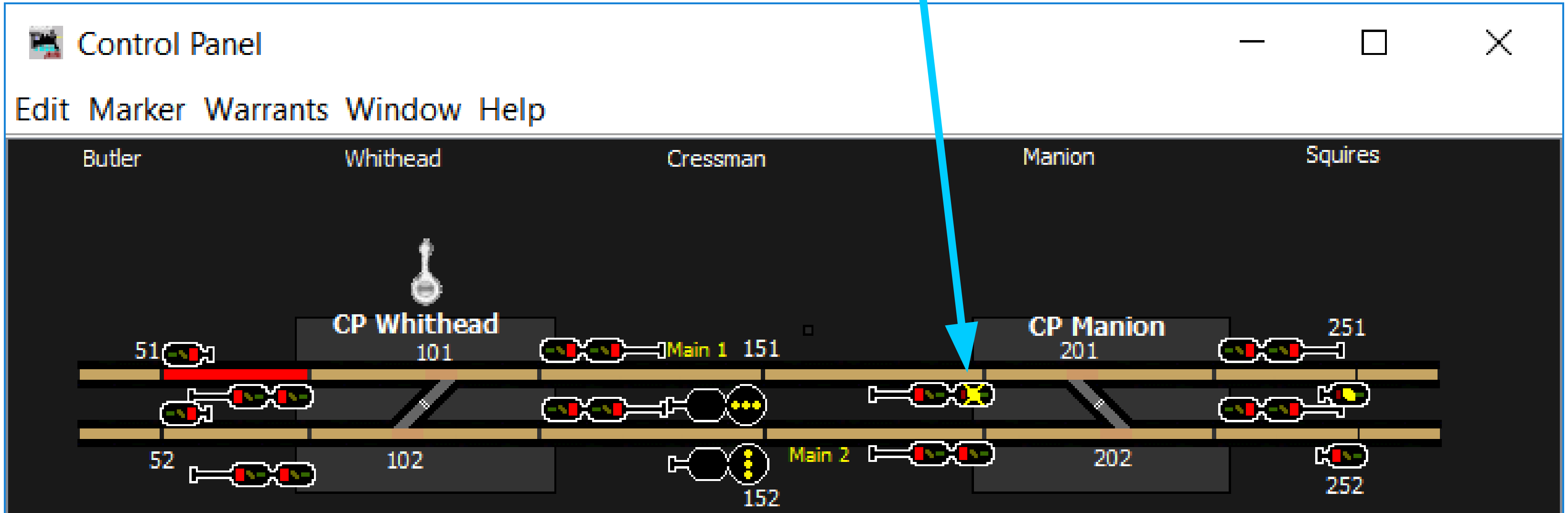
101L

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right	**Not** CTC-Left LM1	null		Stop	Stop
Next CTC Left - Rev	Turnout 101 Reverse	AND	Next CTC-Left	Stop	Stop
Next CTC Left - Norm	Turnout Normal	AND	Next CTC-Left	Stop	Stop
OS occupied	OS WH Main 1 BOD	null		Stop	Stop
Siding occupied	Turnout 101 Reverse	AND	WH West M2 BOD	Stop	Stop
Main occupied	Turnout 101 Normal	AND	WH West M2 BOD	Stop	Stop
Next Siding Stop	Turnout 101 Reverse	AND	52L Stop	Medium Approach	Medium
Next Main Stop	Turnout 101 Normal	AND	51L Stop	Approach	Medium
Next Siding Medium	Turnout 101 Reverse	AND	52L Medium	Medium Approach Medium	Medium
Next Main Clear	Turnout 101 Normal	AND	51L Clear	Clear	Clear



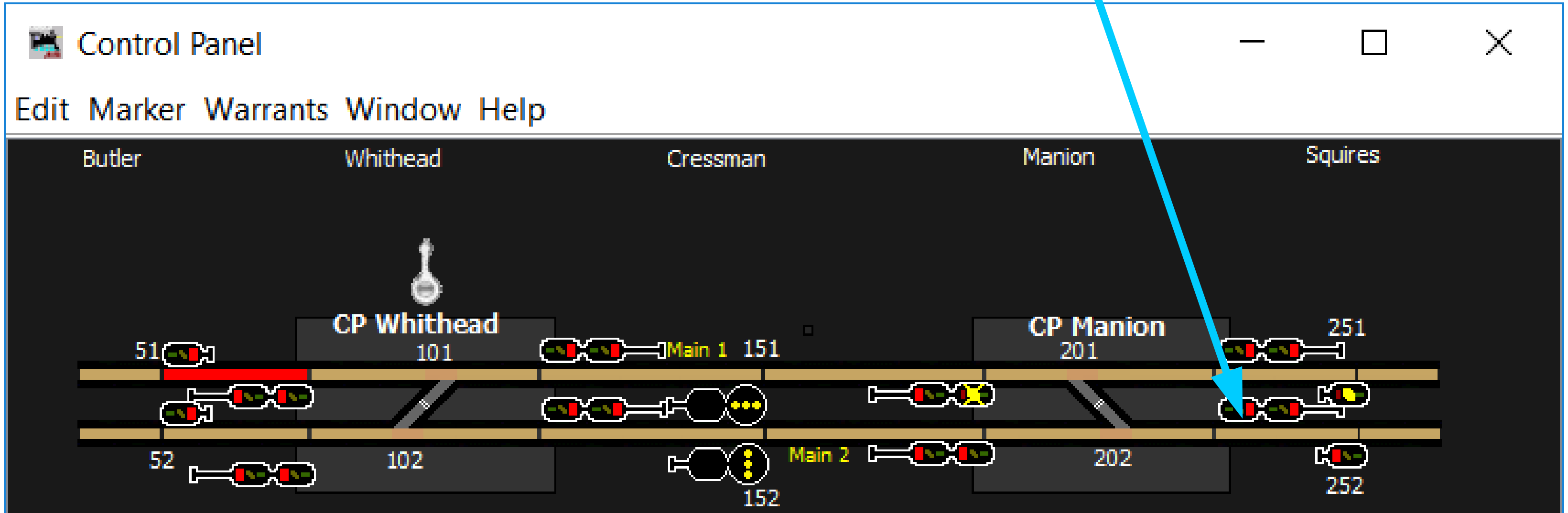
201R

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right	**Not** CTC-Right RM1	null		Stop	Stop
Next CTC Left - Rev	Turnout 201 Reverse	AND	Next CTC-Left	Stop	Stop
Next CTC Left - Norm	Turnout 201 Normal	AND	Next CTC-Left	Stop	Stop
OS occupied	OS MA Main 1 BOD	null		Stop	Stop
Siding occupied	Turnout 201 Reverse	AND	MA East M2 BOD	Stop	Stop
Main occupied	Turnout 201 Normal	AND	MA East M1 BOD	Stop	Stop
Next Siding Stop	Turnout 201 Reverse	AND	252R Mast Stop	Medium Approach	Medium
Next Main Stop	Turnout 201 Normal	AND	251R Mast Stop	Approach	Medium
Next Siding Medium	Turnout 201 Reverse	AND	252R Mast Medium	Medium Approach Medium	Medium
Next Main Clear	Turnout 201 Normal	AND	251R Main Mast Clear	Clear	Clear



202L

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right	**Not** CTC-Left LM2	null		Stop	Stop
Next CTC Left - Rev	Turnout Reverse	AND	Next CTC-Left	Stop	Stop
Next CTC Left - Norm	Turnout Normal	AND	Next CTC-Left	Stop	Stop
OS occupied	OS BOD	null		Stop	Stop
Siding occupied	Turnout 202 Reverse	AND	Siding BOD	Stop	Stop
Main occupied	Turnout 202 Normal	AND	Main BOD	Stop	Stop
Next Siding Stop	Turnout 202 Reverse	AND	Siding Mast Stop	Medium Approach	Medium
Next Main Stop	Turnout 202 Normal	AND	Main Mast Stop	Approach	Medium
Next Siding Medium	Turnout 202 Reverse	AND	Siding Mast Medium	Medium Approach Medium	Medium
Next Main Clear	Turnout 202 Normal	AND	Main Mast Clear	Clear	Clear

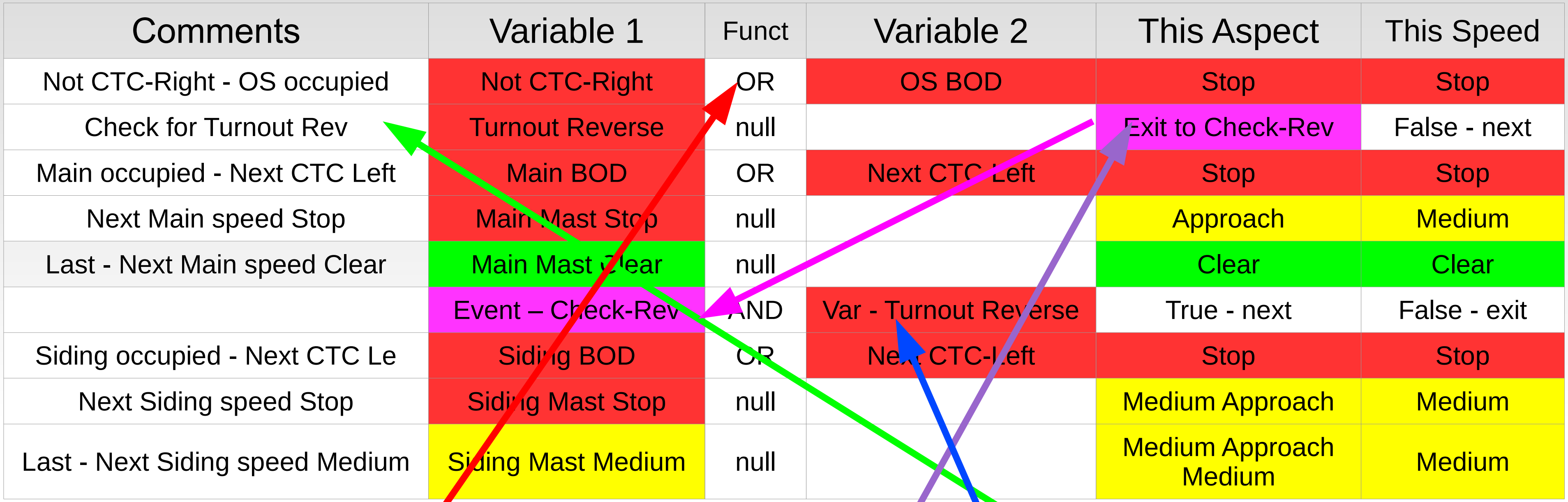


Facing Signal Logic A

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right	Not CTC-Right	null		Stop	Stop
Next CTC Left - Rev	Turnout Reverse	AND	Next CTC-Left	Stop	Stop
Next CTC Left - Norm	Turnout Normal	AND	Next CTC-Left	Stop	Stop
OS occupied	OS BOD	null		Stop	Stop
Siding occupied	Turnout Reverse	AND	Siding BOD	Stop	Stop
Main occupied	Turnout Normal	AND	Main BOD	Stop	Stop
Next Siding Stop	Turnout Reverse	AND	Siding Mast Stop	Medium Approach	Medium
Next Main Stop	Turnout Normal	AND	Main Mast Stop	Approach	Medium
Next Siding Medium	Turnout Reverse	AND	Siding Mast Medium	Medium Approach Medium	Medium
Next Main Clear	Turnout Normal	AND	Main Mast Clear	Clear	Clear

- Here we show two alternate ways to calculate the same facing point logic. The first way simply checks out each combination in order of permisiveness.

Facing Signal Logic B



Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right - OS occupied	Not CTC-Right	OR	OS BOD	Stop	Stop
Check for Turnout Rev	Turnout Reverse	null		Exit to Check-Rev	False - next
Main occupied - Next CTC Left	Main BOD	OR	Next CTC Left	Stop	Stop
Next Main speed Stop	Main Mast Stop	null		Approach	Medium
Last - Next Main speed Clear	Main Mast Clear	null		Clear	Clear
	Event – Check-Rev	AND	Var - Turnout Reverse	True - next	False - exit
Siding occupied - Next CTC Le	Siding BOD	OR	Next CTC-Left	Stop	Stop
Next Siding speed Stop	Siding Mast Stop	null		Medium Approach	Medium
Last - Next Siding speed Medium	Siding Mast Medium	null		Medium Approach Medium	Medium

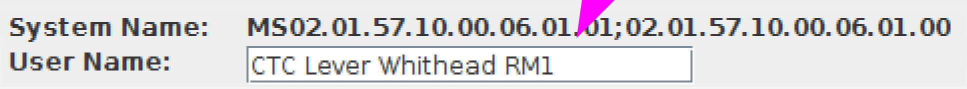
- As in the previous slide, first we check a couple of Stop variables, the CTC direction being against you, OR the OS itself being occupied.

- Now take advantage of the event nature of the LCC to simplify our logic a little bit. Instead of checking the turnout position multiple times as we proceed, we will just check it once. This only saves us one conditional this time, but is better for more complex conditionals.

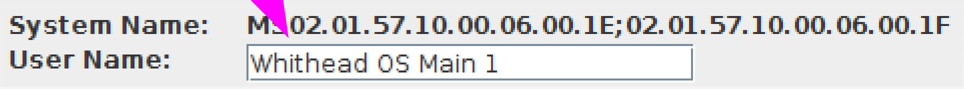
If the turnout is Reverse, then we jump ahead and check things out on the siding (branch). If not we continue checking the main. For this to work properly the ’Check-Rev’ variable triggers on ’Event’, and we also need to check to be sure the variable is still true, in case this siding group evaluation was incidentally triggered by some other change.

Logic

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / OS occupied	**Not** CTC-Right	OR	OS BOD	Stop	Stop

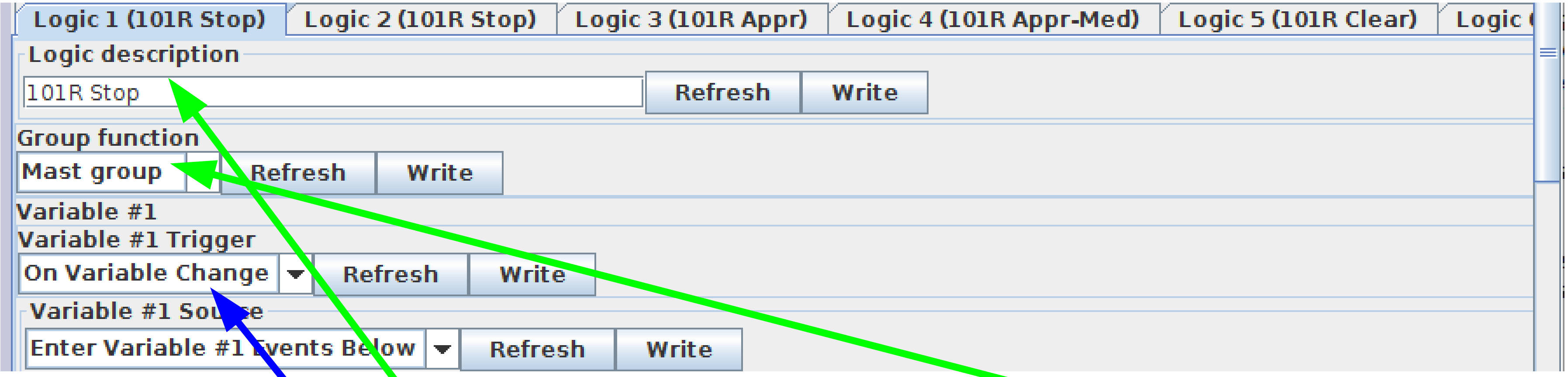


System Name: MS02.01.57.10.00.06.01.01;02.01.57.10.00.06.01.00
User Name: CTC Lever Whithead RM1



System Name: MS02.01.57.10.00.06.00.1E;02.01.57.10.00.06.00.1F
User Name: Whithead OS Main 1

- These two variables as seen in JMRI. I used the Sensor/Turnout creation tool to enter them.



Logic description: 101R Stop

Group function: Mast group

Variable #1

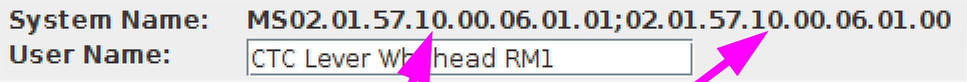
Variable #1 Trigger: On Variable Change

Variable #1 Source: Enter Variable #1 Events Below

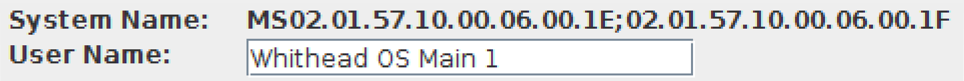
Enter the logic description and set the function to Mast Group. Logic defaults to watching variable changes.

Logic

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / OS occupied	**Not** CTC-Right	OR	OS BOD	Stop	Stop

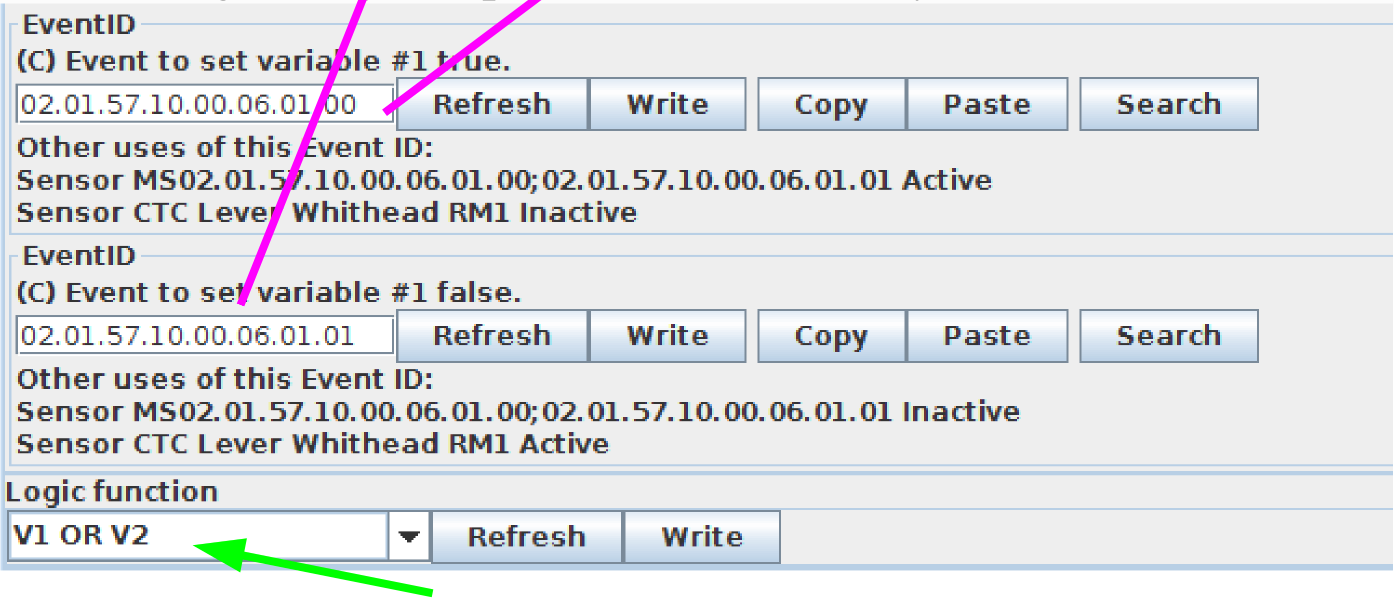


System Name: MS02.01.57.10.00.06.01.01;02.01.57.10.00.06.01.00
User Name: CTC Lever Whithead RM1



System Name: MS02.01.57.10.00.06.00.1E;02.01.57.10.00.06.00.1F
User Name: Whithead OS Main 1

- I actually used the default EventIDs found in variable #1 to create my lever. EventIDs are globally unique so I had no worry about conflicts in meanings.



EventID
(C) Event to set variable #1 true.
02.01.57.10.00.06.01.00
Other uses of this Event ID:
Sensor MS02.01.57.10.00.06.01.00;02.01.57.10.00.06.01.01 Active
Sensor CTC Lever Whithead RM1 Inactive

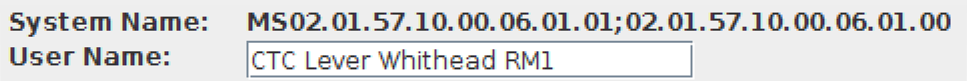
EventID
(C) Event to set variable #1 false.
02.01.57.10.00.06.01.01
Other uses of this Event ID:
Sensor MS02.01.57.10.00.06.01.00;02.01.57.10.00.06.01.01 Inactive
Sensor CTC Lever Whithead RM1 Active

Logic function
V1 OR V2

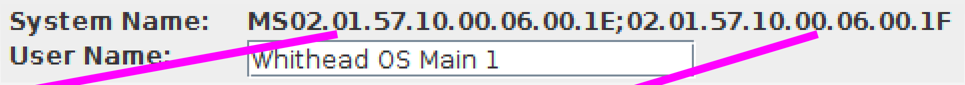
- Enter the logic function. In this case it is ’OR’.

Logic

Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / OS occupied	**Not** CTC-Right	OR	OS BOD	Stop	Stop

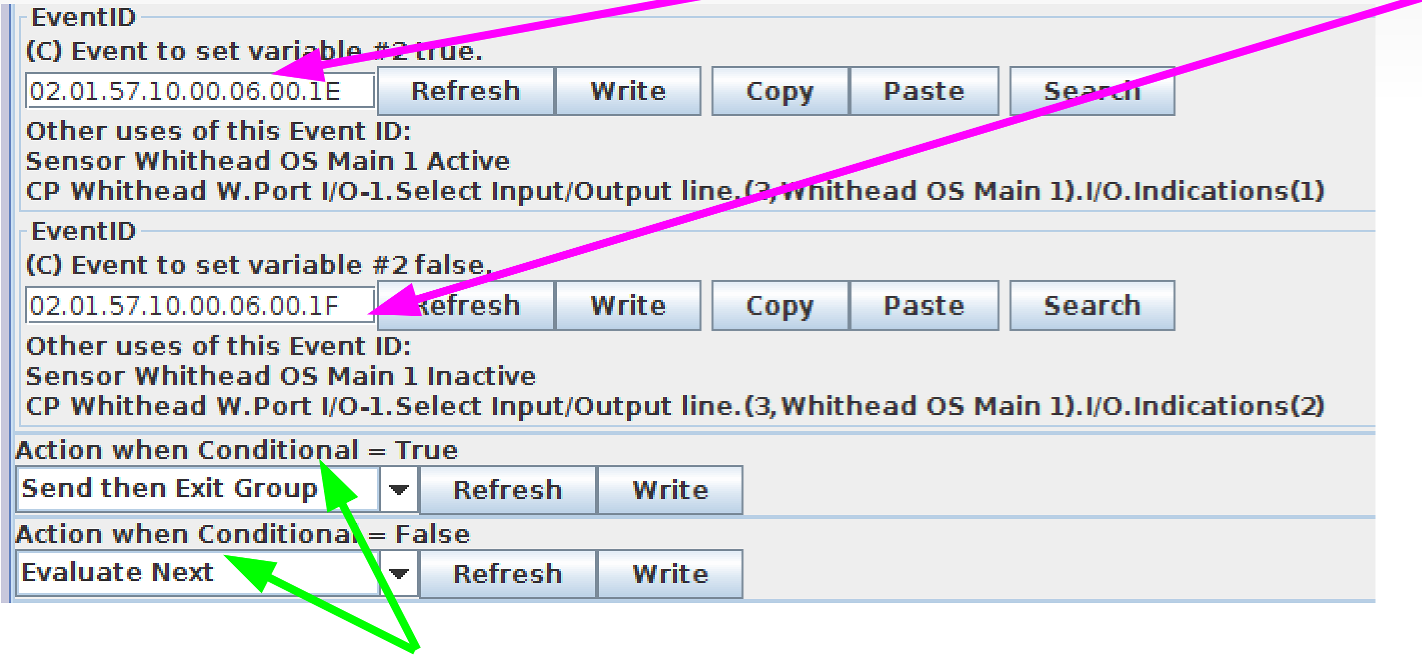


System Name: MS02.01.57.10.00.06.01.01;02.01.57.10.00.06.01.00
User Name: CTC Lever Whithead RM1



System Name: MS02.01.57.10.00.06.00.1E;02.01.57.10.00.06.00.1F
User Name: Whithead OS Main 1

- For the block detector I copy/pasted from the I/O line into Variable #2.



EventID
(C) Event to set variable #2 true.
02.01.57.10.00.06.00.1E

Other uses of this Event ID:
Sensor Whithead OS Main 1 Active
CP Whithead W.Port I/O-1.Select Input/Output line.(3,Whithead OS Main 1).I/O.Indications(1)

EventID
(C) Event to set variable #2 false.
02.01.57.10.00.06.00.1F

Other uses of this Event ID:
Sensor Whithead OS Main 1 Inactive
CP Whithead W.Port I/O-1.Select Input/Output line.(3,Whithead OS Main 1).I/O.Indications(2)

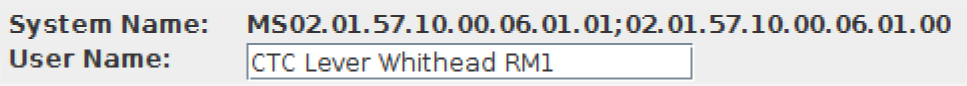
Action when Conditional = True
Send then Exit Group

Action when Conditional = False
Evaluate Next

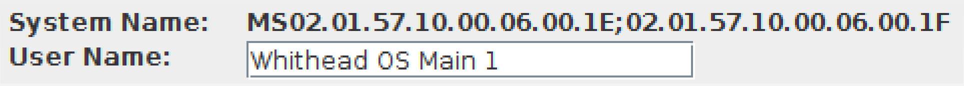
- These default actions are normal for mast logic conditionals. If the condition is true, then any actions are sent, and all less restrictive aspects are skipped.

Logic

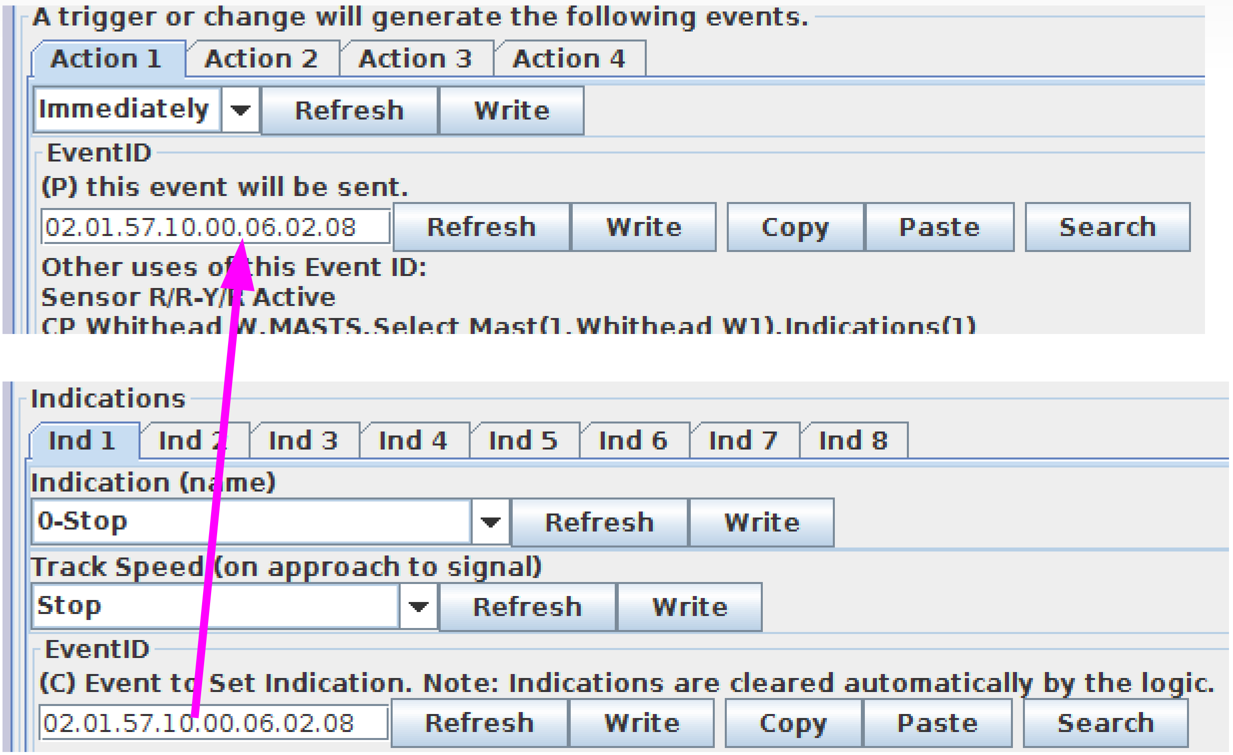
Comments	Variable 1	Funct	Variable 2	This Aspect	This Speed
Not CTC-Right / OS occupied	**Not** CTC-Right	OR	OS BOD	Stop	Stop



System Name: MS02.01.57.10.00.06.01.01;02.01.57.10.00.06.01.00
User Name: CTC Lever Whithead RM1



System Name: MS02.01.57.10.00.06.00.1E;02.01.57.10.00.06.00.1F
User Name: Whithead OS Main 1



A trigger or change will generate the following events.

EventID
(P) this event will be sent.
02.01.57.10.00.06.02.08

Other uses of this Event ID:
Sensor R/R-Y/R Active
CP Whithead W.MASTS.Select Mast(1.Whithead W1).Indications(1)

Indications

Indication (name)
0-Stop

Track Speed (on approach to signal)
Stop

EventID
(C) Event to Set Indication. Note: Indications are cleared automatically by the logic.
02.01.57.10.00.06.02.08

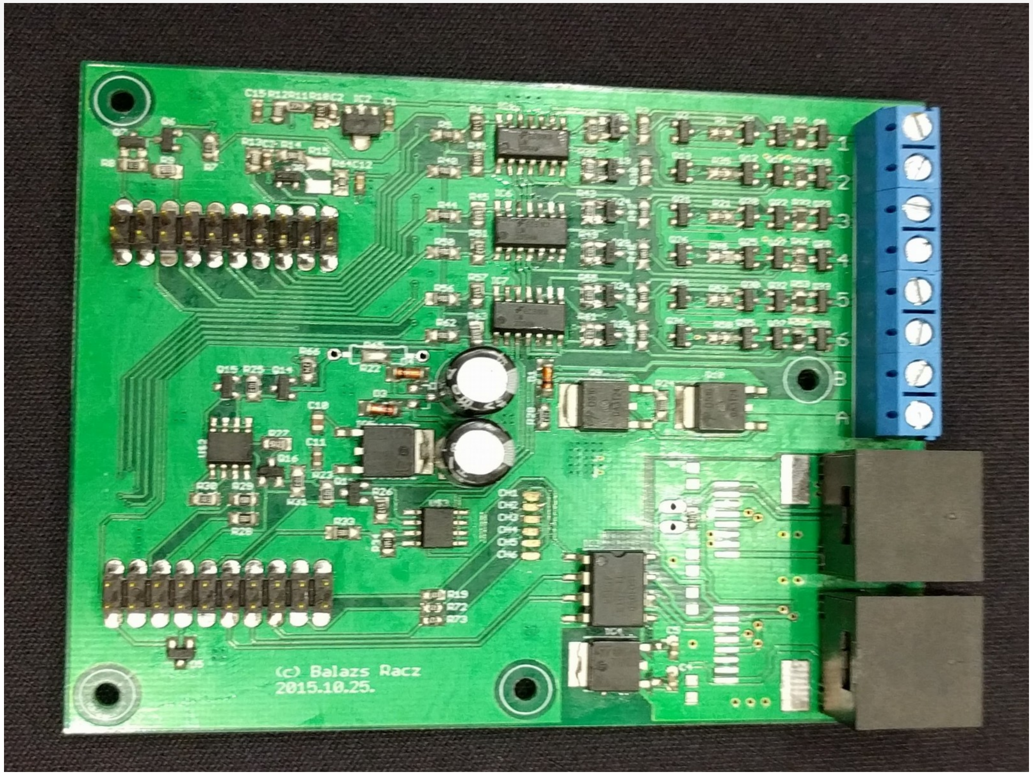
- I then copied the event that sets the Signal rule to 'Stop' into 'Action 1' of the logic. Therefore anytime the CTC direction lever is not 'Traffic Right' or if the OS section is occupied, then the signal will be set to 'Stop'.

A glimpse into the future

- Following are some known LCC related developments

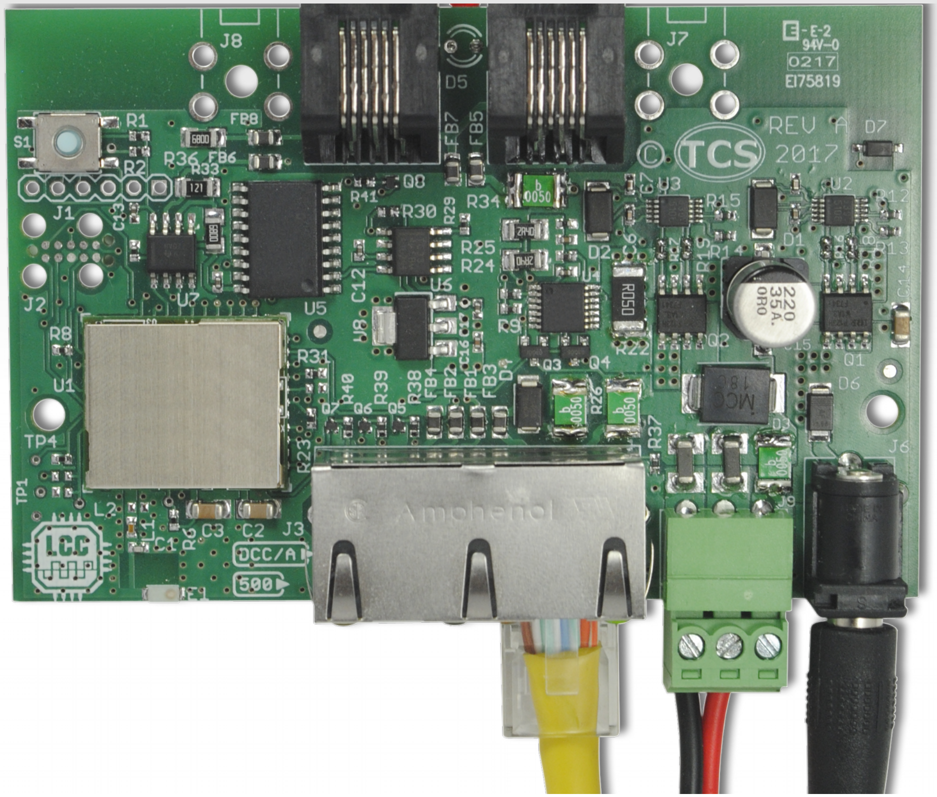
Coming LCC Hardware

- RailCom Detector under development.
  (development image shown)



8-channel
Block occupancy detector
    Adjustable sensitivity
Feedback via LCC
Circuit breaker
    Adjustable current limit
Turn off staging track
Railcom to determine which
    train is on the track
Loco CV readout POM
Staggered block power
    turn-on
Auto reverse channel

Upcoming products -- TCS



This is a DCC command station with an LCC port

Built-in booster for ~2A

Connects via CAN and WiFi, bridges between them

Native OpenLCB throttles

Connect computer or throttle wirelessly via WiFi

Connect NCE wired throttles or NCE wireless base station

From **TCS,** the creators of

WOWSound™

Upcoming products -- TCS

- First Wireless OpenLCB throttle
- Connects via WiFi
- Cross-industry robust technology
- Works with home WiFi and off the shelf components
- Connects to any standard OpenLCB command station
- Large, easy-to-read, backlit display
- Replaceable batteries: 8+ hours of continuous use per charge.
- Option: use with any JMRI layout as a WiThrottle

The Future of LCC

- Smart Detector, Railcom, Circuit Breaker, Reversers
- Simple Detector, CT coil based.
- Stall Motor Driver (Support for ganged Tortoises, MP1, etc.)
- Dual Coil Solenoid Driver.
- LocoNet to LCC Gateway. (LCC support for existing products)
- Ethernet Links.
- Wireless Links.
- Throttles
- Smart Boosters, Command Stations.

LCC Configuration Tools

- Because all the configuration information as well as the values, user names, and coments reside permanently in the nodes themselves, it is easy to use different configuration tools interchangeably. There is no need to synchronize them externally or move files around from computer to computer.

LCC Configuration Tools

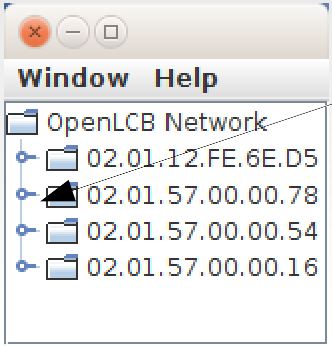
- This node information is stored in the node as a CDI file. (Configuration Description Information) The CDI is in .xml format, but because it references internal register locations it is not advisable to attempt making any changes manually.

Example CDI info as stored in a node:

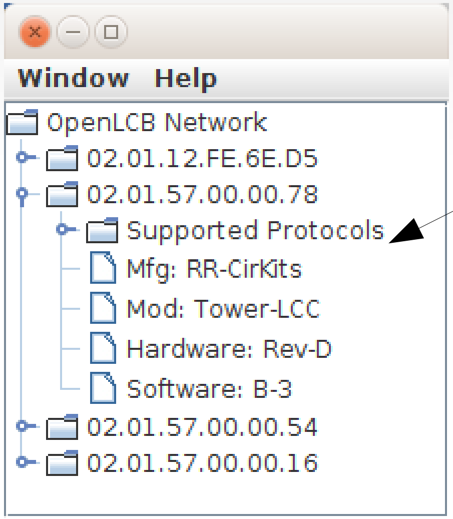
```
<?xml version='1.0'?>
<cdi xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:noNamespaceSchemaLocation=
        'http://openlcb.org/schema/cdi/1/1/cdi.xsd'>
<identification>
<manufacturer>RR-CirKits</manufacturer>
<model>Signal-LCC</model>
<hardwareVersion>rev-A</hardwareVersion>
<softwareVersion>A-4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

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

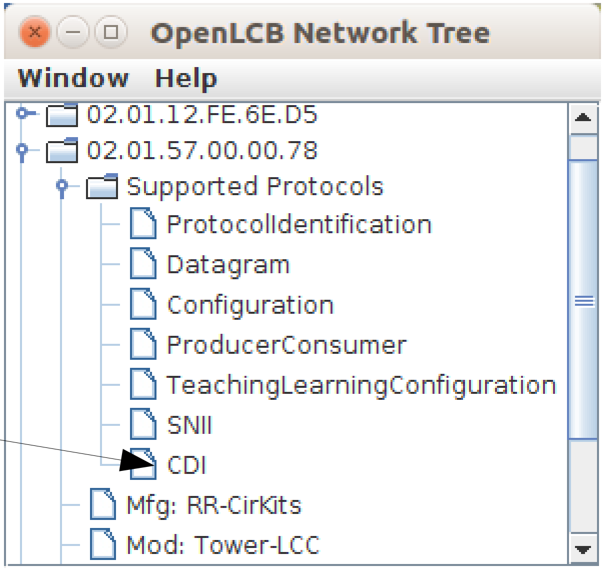
  Select OpenLCB and choose ’Configure Nodes’



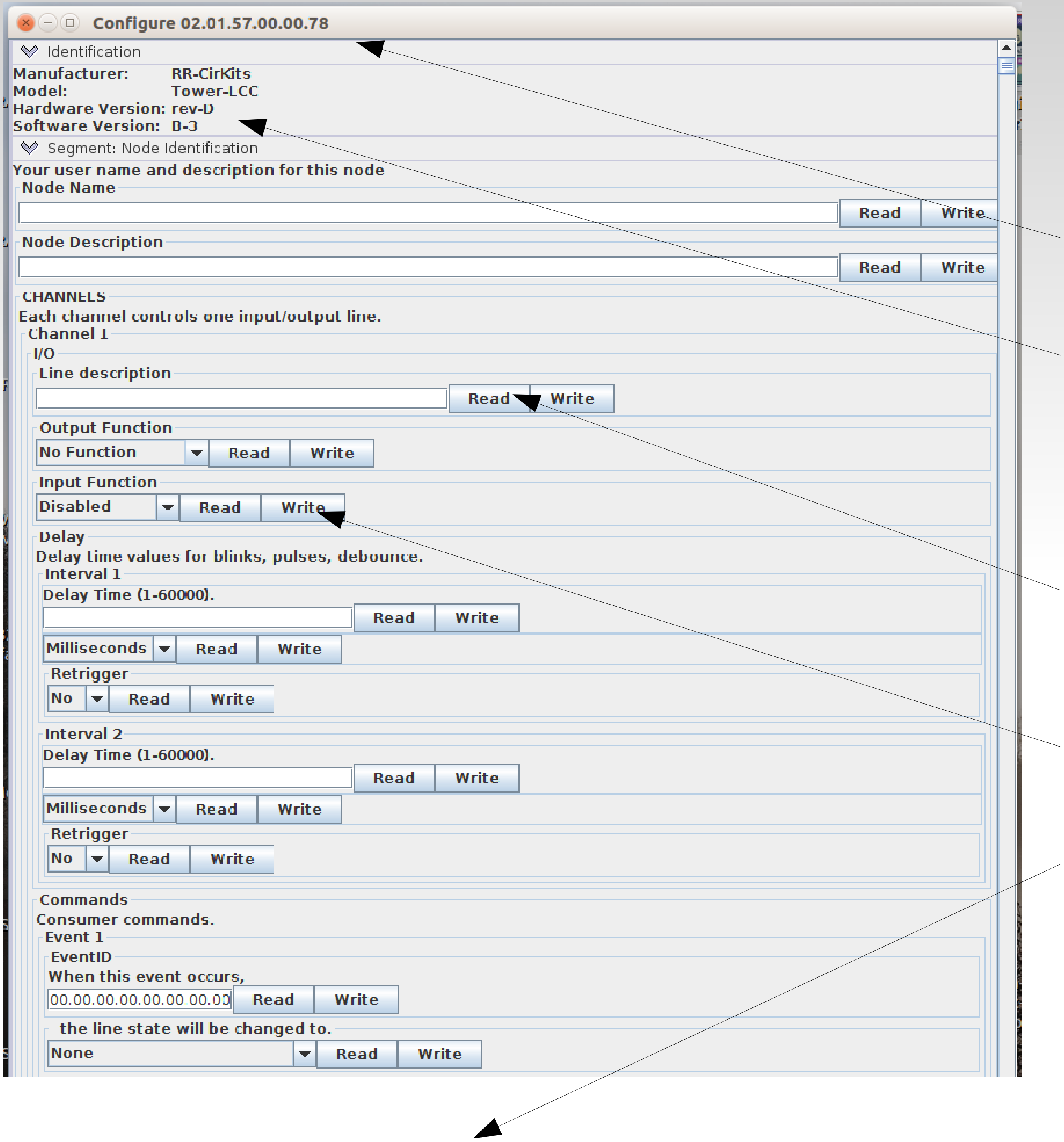
Next open the node you need to configure.



Open ’Supported Protocols’.



Then choose ’CDI’ to open the JMRI CDI tool and read the node.



Configure 02.01.57.00.00.78

Identification

Manufacturer: RR-CirKits
Model: Tower-LCC
Hardware Version: rev-D
Software Version: B-3

Segment: Node Identification

Your user name and description for this node

Node Name

Node Description

CHANNELS

Each channel controls one input/output line.

Channel 1

I/O

Line description

Output Function: No Function

Input Function: Disabled

Delay

Delay time values for blinks, pulses, debounce.

Interval 1

Delay Time (1-60000).

Milliseconds

Retrigger: No

Interval 2

Delay Time (1-60000).

Milliseconds

Retrigger: No

Commands

Consumer commands.

Event 1

EventID

When this event occurs, 00.00.00.00.00.00.00.00

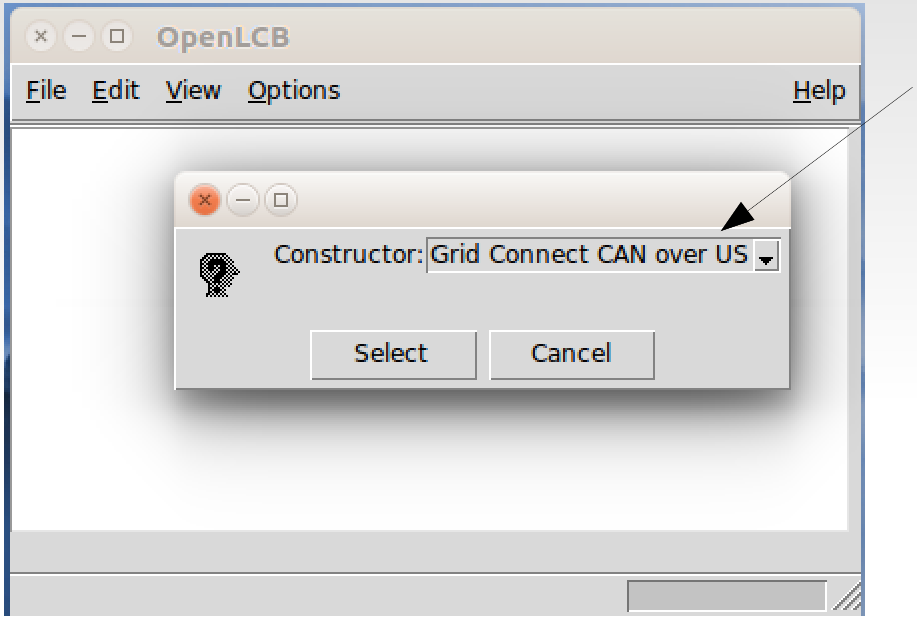
the line state will be changed to. None

This will open the JMRI CDI tool window and allow you to read and write data to the node. The window header shows the node ID that is open and the Identification shows some basic data about the node.

The actual data will not show up unless you choose to 'Read' it from the node. If you make any changes to the information, then you must 'Write' the data to store it into the node.

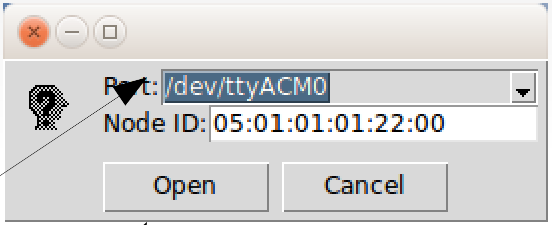
There is a 'Read All' button at the bottom of the window, but be forwarned, it takes a lot of time to read all of the data in.

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



Constructor: Grid Connect CAN over US

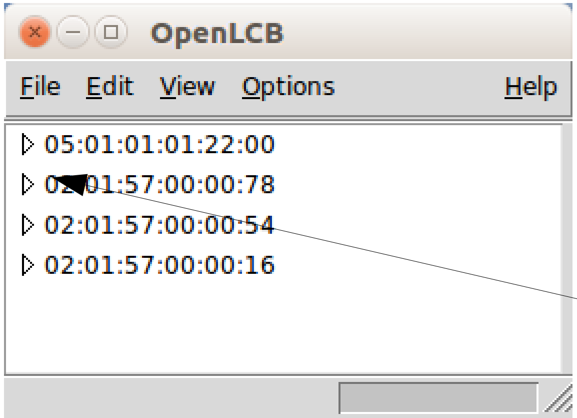
If you are using the LCC Buffer-USB as your interface device, then select ’Grid Connect CAN over USB’ .



Port: /dev/ttyACM0
Node ID: 05:01:01:01:22:00

Next select the proper COM port. (this example is on Linux)

Once you click on ’Open’ a similar window to the one you saw in JMRI will open. The first entry is the program connection itself. The other entries are a list of the attached nodes.



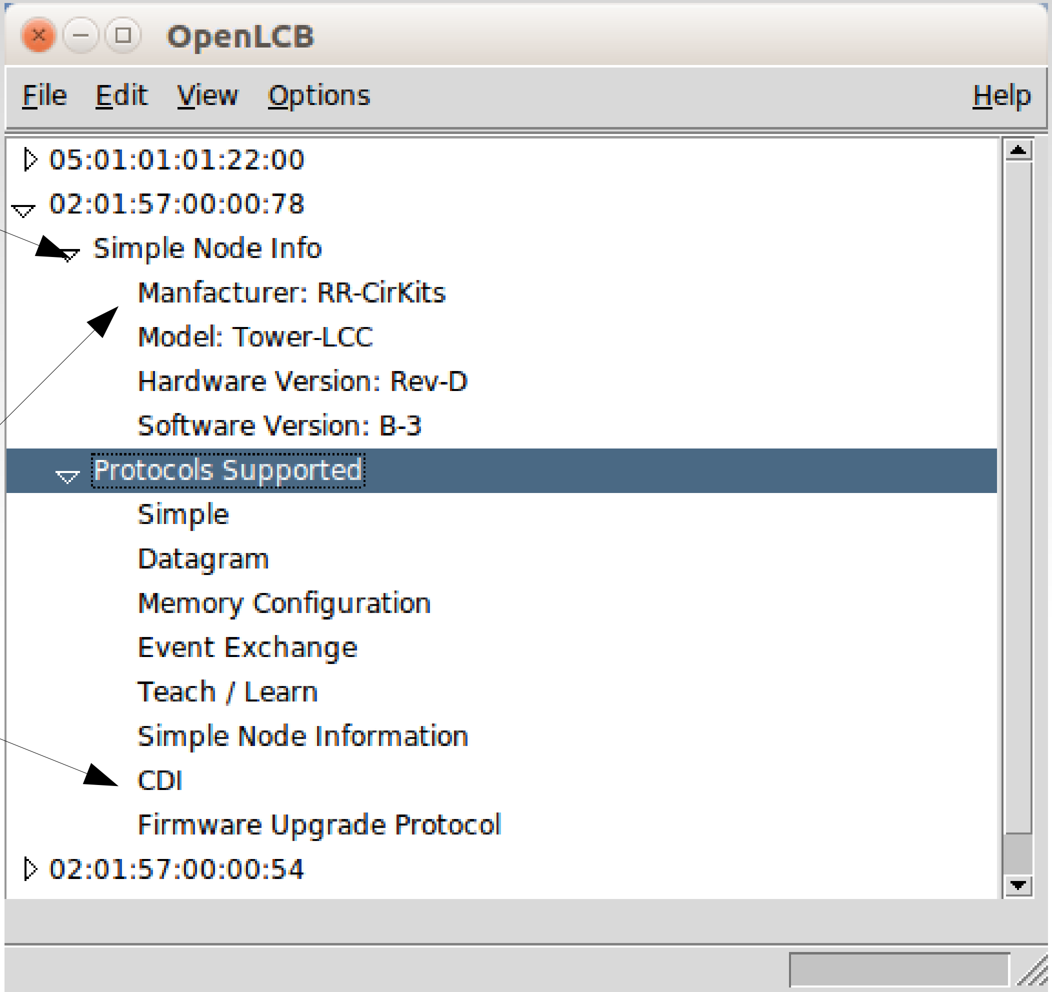
▷ 05:01:01:01:22:00
▷ 02:01:57:00:00:78
▷ 02:01:57:00:00:54
▷ 02:01:57:00:00:16

As in JMRI, open the node you choose to configure by expanding its tree view.

Robert’s CDI tool opens a bit differently than JMRI. You need to drill down in the tree to see more information.

However, because the information actually is stored in the node, you should see exactly the same data.

The CDI tool is started by clicking on ’CDI’ just as it was in JMRI. However, If you missed the LCC traffic indicators, there is no visual feedback that anything has happened, and it may take a long time before the CDI window loads and finally opens. Resist the temptation to click it again.



05:01:01:01:22:00
02:01:57:00:00:78
Simple Node Info
Manfacturer: RR-CirKits
Model: Tower-LCC
Hardware Version: Rev-D
Software Version: B-3
Protocols Supported
Simple
Datagram
Memory Configuration
Event Exchange
Teach / Learn
Simple Node Information
CDI
Firmware Upgrade Protocol
02:01:57:00:00:54

Be patient and you will be awarded with a usable presentation. (Similar to the new JMRI view)

Again the Node ID is found at the top of the window.

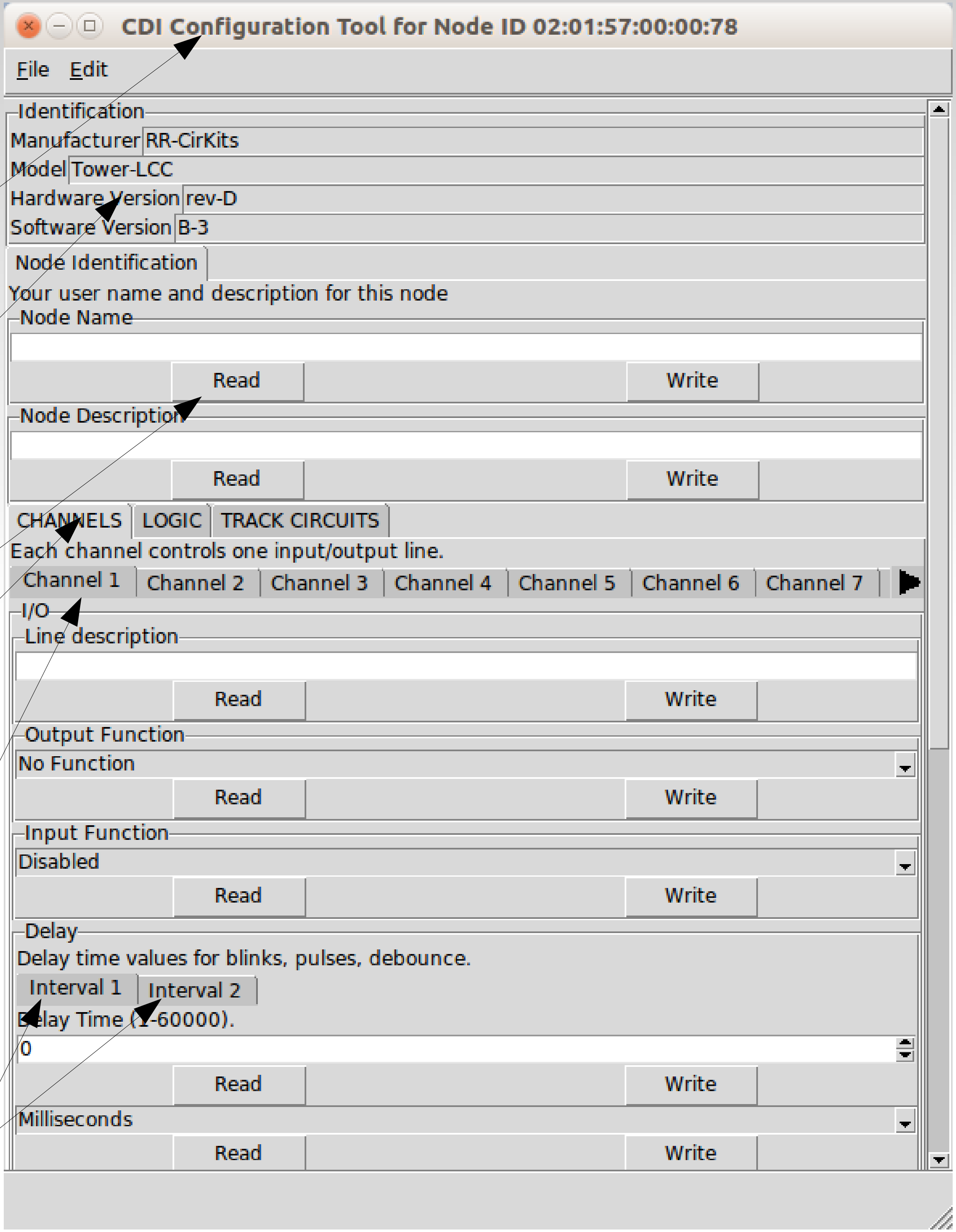
Node Identification follows.

Next is any Name and Description that you have given to the node. (be sure to click on 'Read' to see it.

The key difference is that the data is presented in a tab selected format. Note: the JMRI developers have created a similar improvement.

In this example we have selected 'CHANNELS' and 'Channel 1'.

In like manner, any repeated similar items are presented as tab choices.



CDI Configuration Tool for Node ID 02:01:57:00:00:78

File Edit

Identification

Manufacturer: RR-CirKits

Model: Tower-LCC

Hardware Version: rev-D

Software Version: B-3

Node Identification

Your user name and description for this node

Node Name

Node Description

CHANNELS | LOGIC | TRACK CIRCUITS

Each channel controls one input/output line.

Channel 1 | Channel 2 | Channel 3 | Channel 4 | Channel 5 | Channel 6 | Channel 7

I/O

Line description

Output Function: No Function

Input Function: Disabled

Delay

Delay time values for blinks, pulses, debounce.

Interval 1 | Interval 2

Delay Time (1-60000).

0

Milliseconds

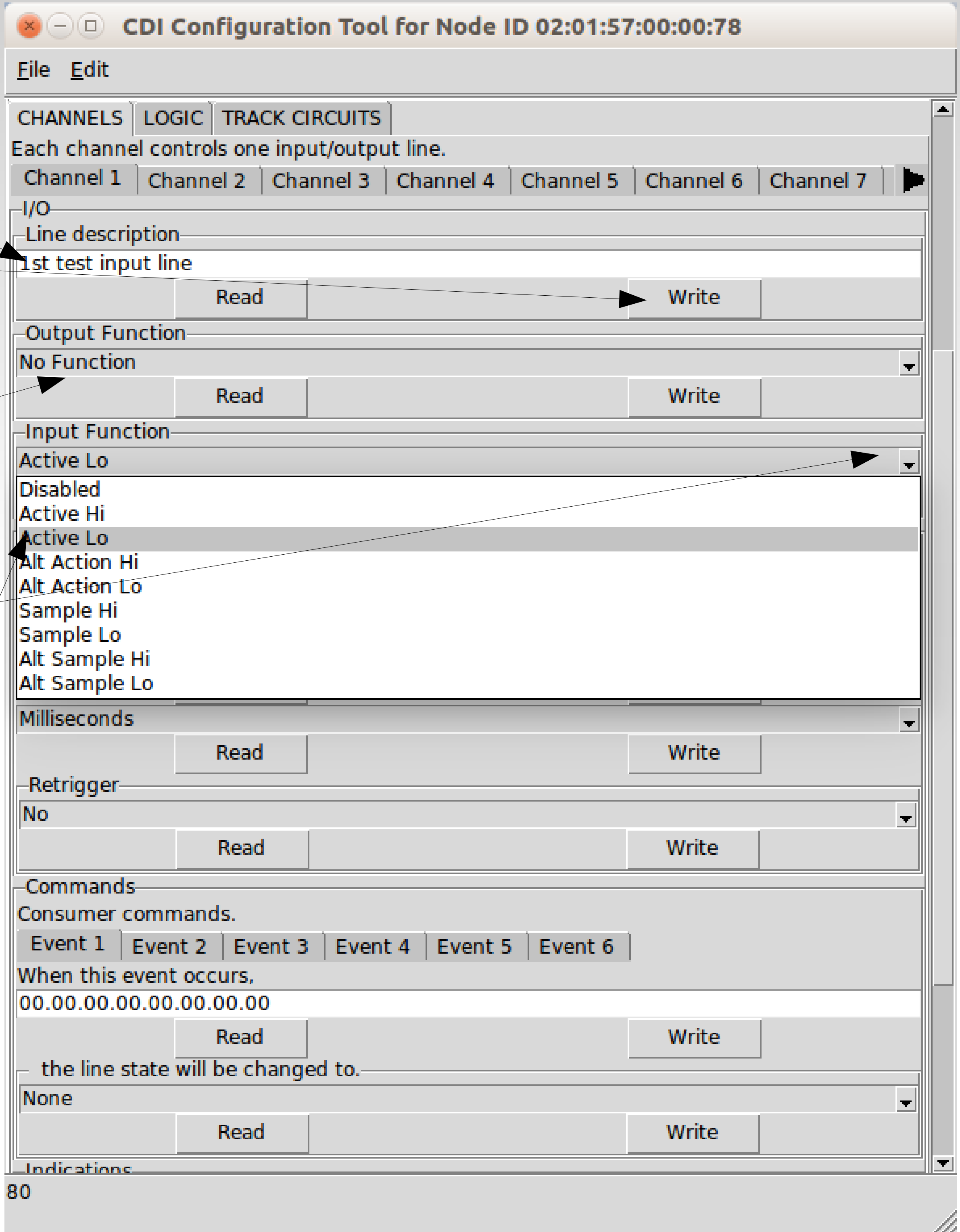
We have now entered a user comment for the line.

Be sure to click on ’Write’ to save the item.

Normally you will need to set the Output as ’No Function’ in order to use the line as an Input.

A list select arrow will present you with valid choices for some items.

In this example we have chosen the input to be ’Active Lo’. It responds as ’On’ when being pulled low. (called negative logic) This is usual for many detectors and push buttons that turn ’on’ by switching to the common ground.



CDI Configuration Tool for Node ID 02:01:57:00:00:78

Each channel controls one input/output line.

I/O

Line description: 1st test input line

Output Function: No Function

Input Function: Active Lo

Milliseconds

Retrigger: No

Commands

Consumer commands.

When this event occurs, 00.00.00.00.00.00.00.00

the line state will be changed to. None

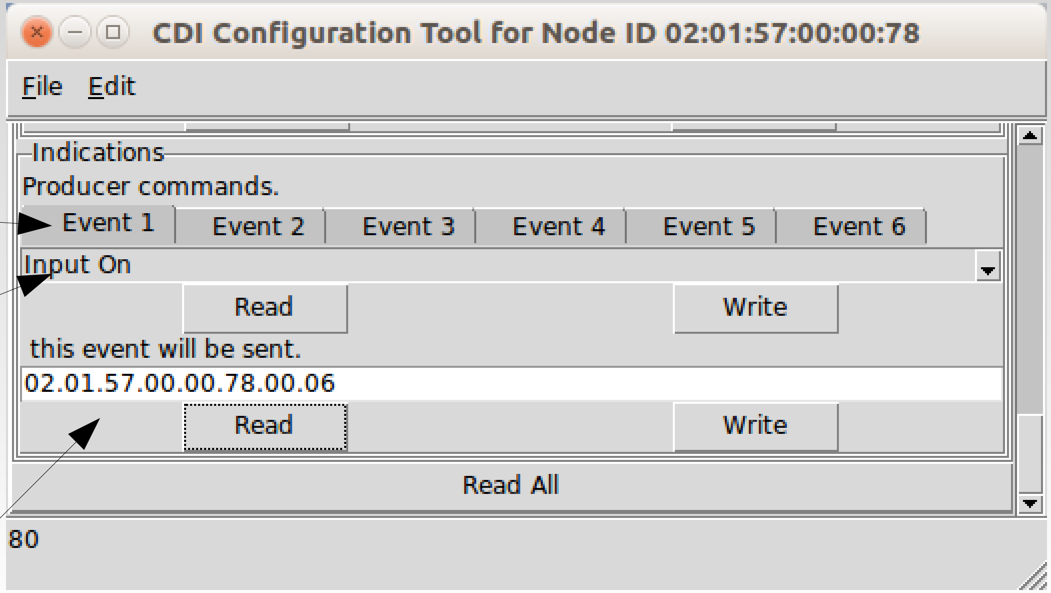
Indications

80

An input line is used to ’Produce’ messages, so scroll down to the ’Indications’ section and pick the first Event.

Select when the event is sent. For this first event it will when the input is ’On’. (low per our initial setting)

For the event number you can either copy an event into the box from someplace else, or else click on ’Read’ to get a new event.



CDI Configuration Tool for Node ID 02:01:57:00:00:78

Indications

Producer commands.

Event 1

Input On

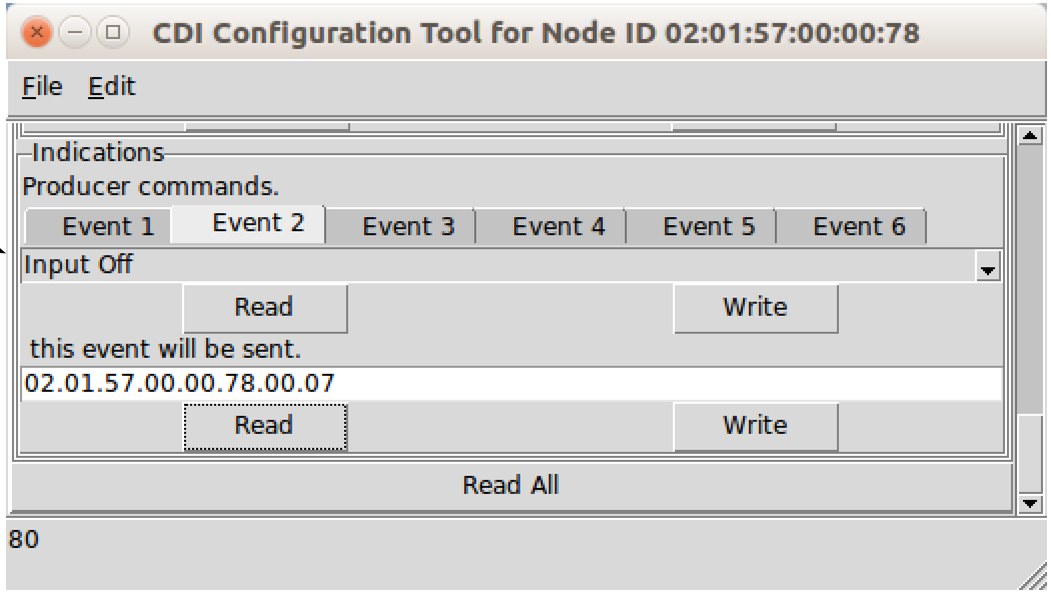
this event will be sent.

02.01.57.00.00.78.00.06

80

Now select ’Event 2’ and enter the data for the ’Input Off’ event.

For simple setups the remaining events will be unused. Our button or detector or whatever is connected to the line will now send 02.01.57.00.00.78.00.06 when pressed (on) and 02.01.57.00.00.78.00.07 when released. (off)



CDI Configuration Tool for Node ID 02:01:57:00:00:78

Indications

Producer commands.

Event 2

Input Off

this event will be sent.

02.01.57.00.00.78.00.07

80

- In like manner outputs (Consumers) may be configured to respond to events. These Events may come from a JMRI program, other inputs, or even logic statements.
- Current configuration tools are still under development. One design target is to eliminate any reference to the actual EventID numbers, and simply use the users own names for items.
- I am not optimistic about seeing that in my lifetime, but once a line is configured you really can ignore the details of each EventID because you will not need to worry about any duplication, and you do not need to know them ahead of time to properly select the hardware like you do on existing networks. In LCC the hardware either offers you a new unused Event, or you may configure it to respond to your own already defined Events. (just copy your EventID to it)

Other Layout Animation

- Signaling is normally the most complex animation applied to a model railroad layout.
- Crossing gates and flashers with or without sound is another closely related animation that is often attempted by modelers. Commercial gate animators have various levels of sophistication, from simple on – off, control to reasonably accurate operation. I have seen designers twist themselves into knots trying to figure out how to do it accurately in both directions. However if you think in terms of Events it is actually very simple. Define two blocks. The first covers the entire gate A*pproach* area. The second covers just the highway portion. We call it the I*sland*.
  The Logic:
  1. Approach clear AND Island clear = gates up (requires memory of the two events plus AND logic)
  2. Approach occupied event = gates down
  3. Island occupied event = gates down
  4. Island clear event = gates up

- Traffic signals. Simple flashers to full four or six cycle control.
- Building lighting and signage.
- Day – Night lighting.
- Street and parking lot lighting.
- Operating bridge spans.
- Warehouse doors.
- Mine skips.
- All of the above could be individual devices, or centrally controlled for even more realism. Building lights could follow room lighting, bright in the evening, off late at night, then on again early in the morning. Traffic signals go to flashing mode late at night. Warehouse doors open as trains arrive. Etc.

Acknowledgements

Key OpenLCB Contributors: Bob Jacobsen, Alex Shepherd, David Harris, Stuart Baker, Balazs Racz, Jim Kueneman, Don Goodman-Wilson, John Plocher

Developer Group

10 to 15 actively working on code at any time
25 to 50 regular contributors and supporters
Many of the same people as supporting JMRI

User Group

Started November 2009
July 2016 we had 226 addresses

NMRA liaison: Stephen Priest
NMRA w.g. chairman: Karl Kobel

Info

Yahoo Users Group

openlcb@yahoogroups.com

LayoutCommandControl@yahoogroups.com

Useful Links

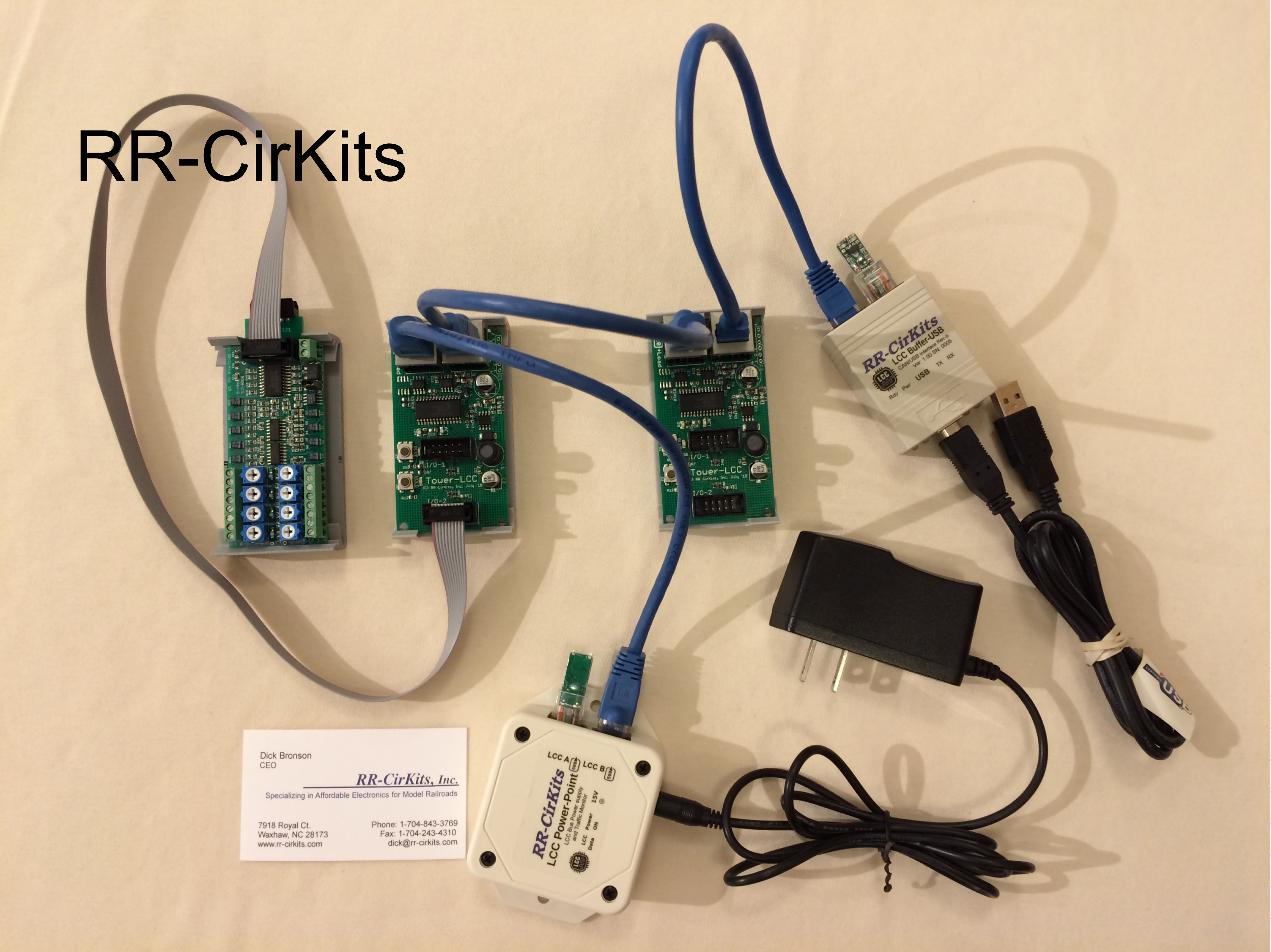
http://openlcb.org

http://openlcb.com

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

OpenLCB LCC

RR-CirKits



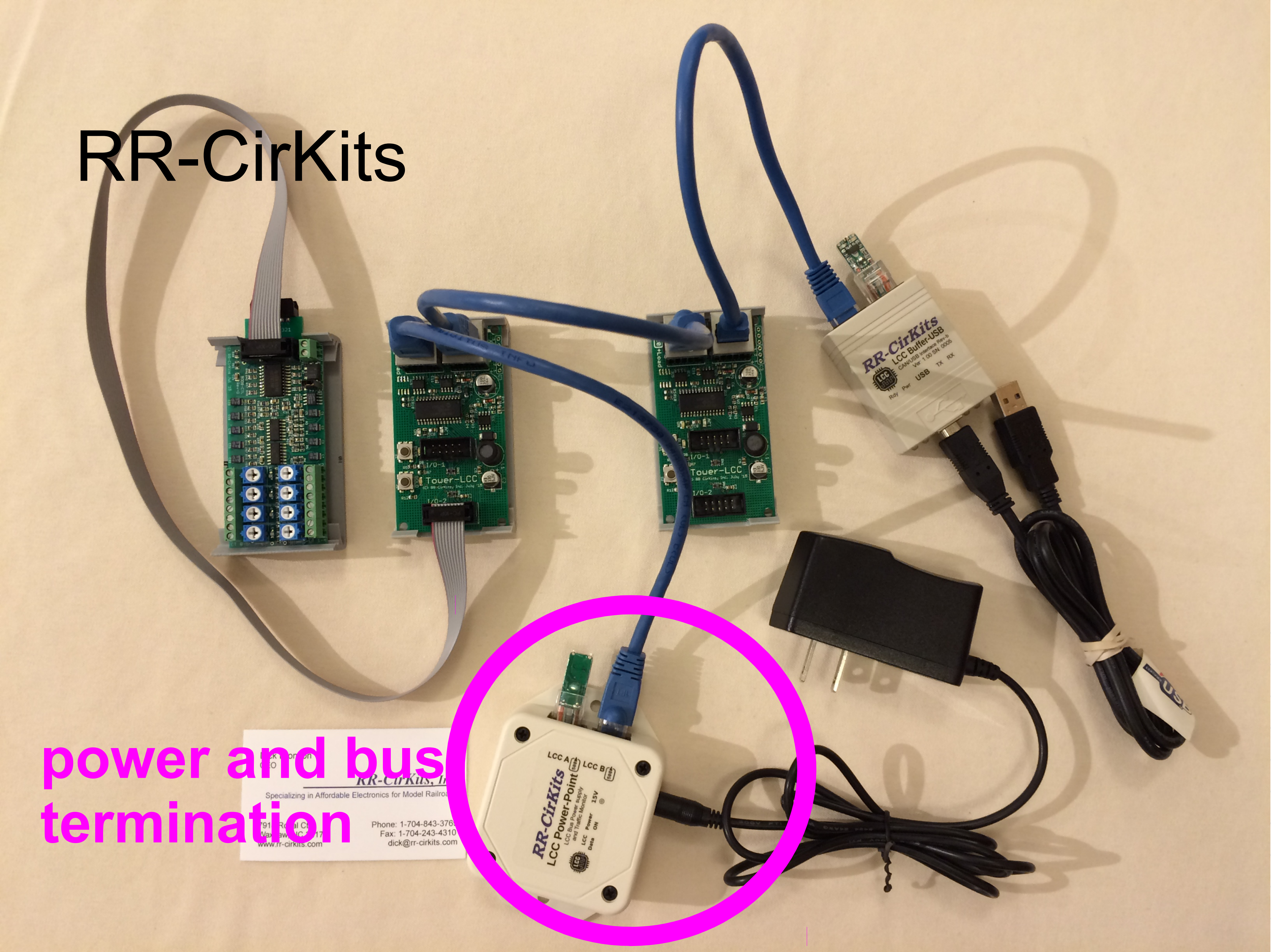
Dick Bronson
CEO

RR-CirKits, Inc.

Specializing in Affordable Electronics for Model Railroads

7918 Royal Ct.
Waxhaw, NC 28173
www.rr-cirkits.com

Phone: 1-704-843-3769
Fax: 1-704-243-4310
dick@rr-cirkits.com

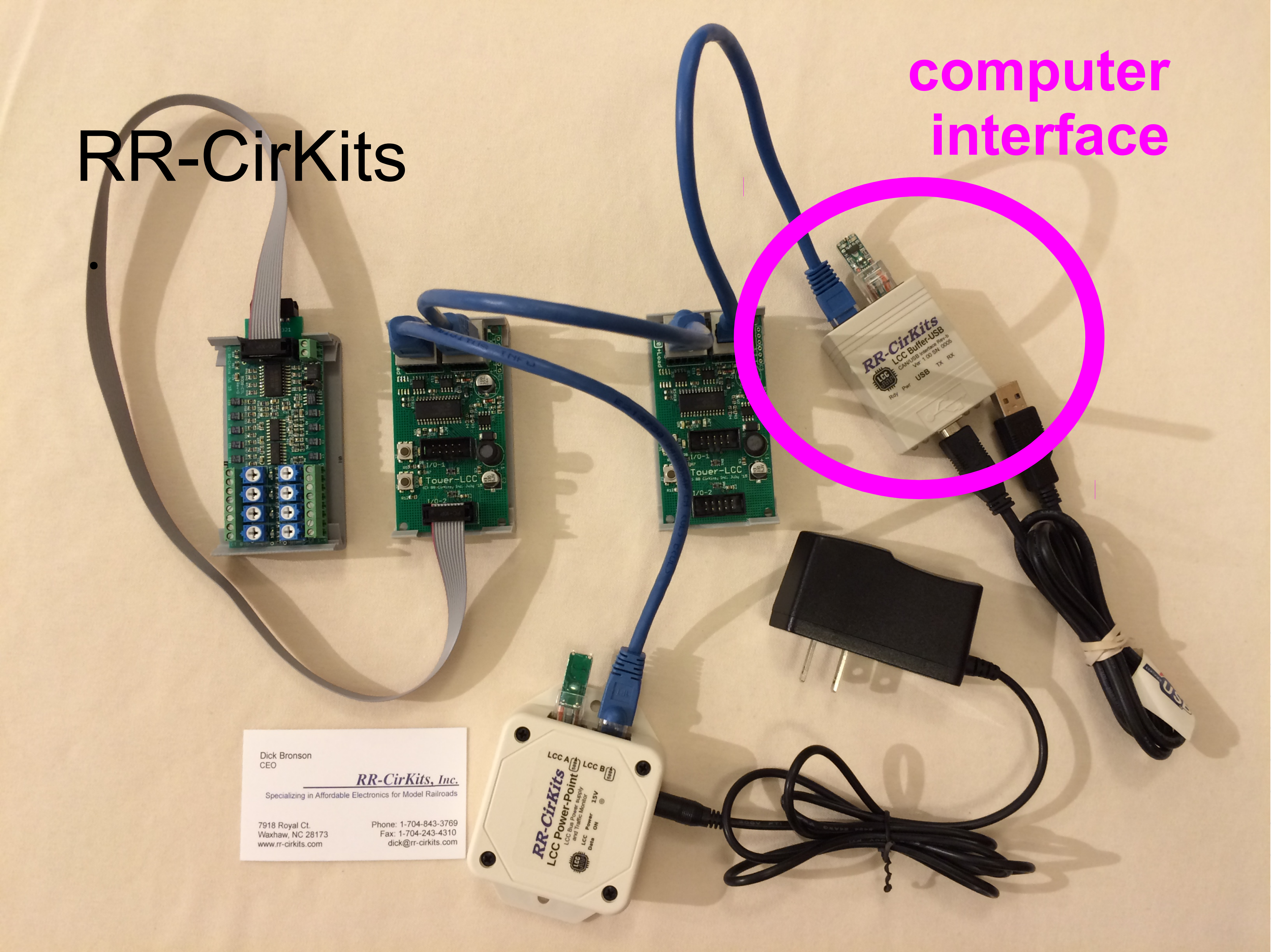


RR-CirKits

power and bus termination

RR-CirKits

computer interface



Dick Bronson
CEO

RR-CirKits, Inc.

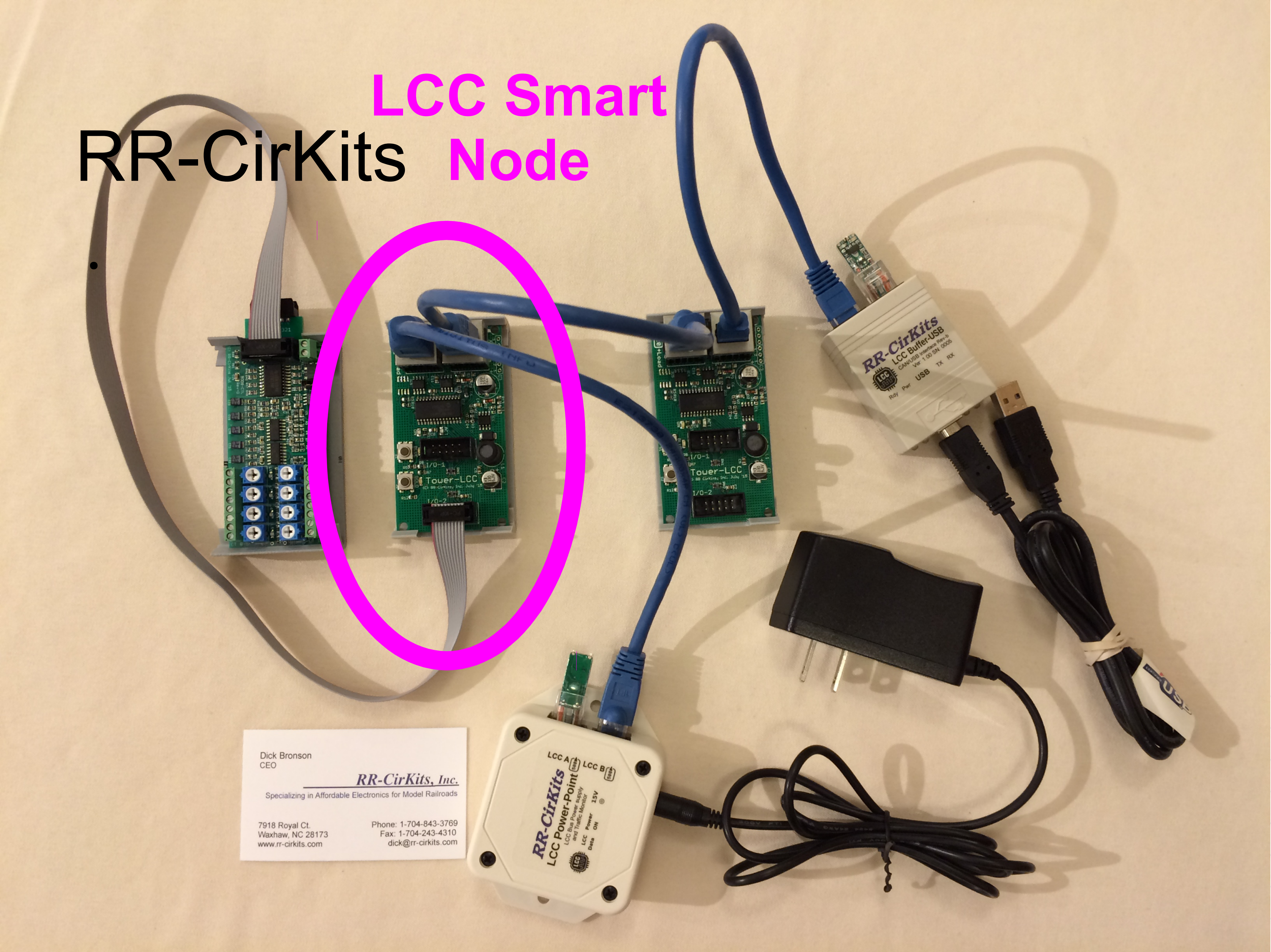
Specializing in Affordable Electronics for Model Railroads

7918 Royal Ct.
Waxhaw, NC 28173
www.rr-cirkits.com

Phone: 1-704-843-3769
Fax: 1-704-243-4310
dick@rr-cirkits.com

RR-CirKits

LCC Smart Node



Dick Bronson
CEO

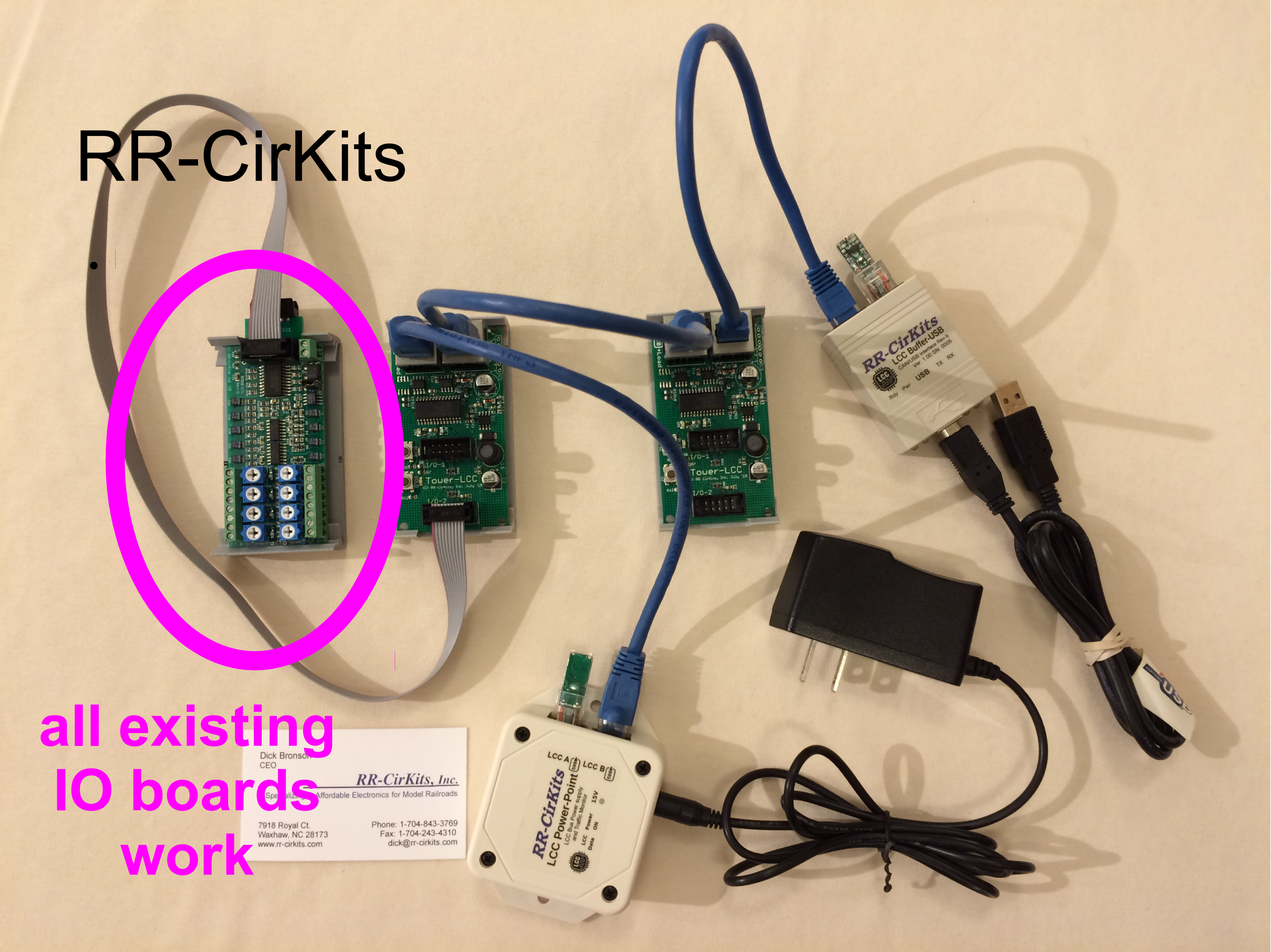
RR-CirKits, Inc.

Specializing in Affordable Electronics for Model Railroads

7918 Royal Ct.
Waxhaw, NC 28173
www.rr-cirkits.com

Phone: 1-704-843-3769
Fax: 1-704-243-4310
dick@rr-cirkits.com

RR-CirKits



all existing IO boards work

Questions

- ?