MER 2021
Mount Clare Junction
LCC for the Rest of Us - Overview
(Layout Command & Control)

Presented by: Dick Bronson

RR-CirKits, Inc.

LCC for the Rest of Us.

Part 1 (overview)

www.rr-cirkits.com/clinics/MER-2021-Signaling with LCC-A.pdf

Part 2 (details)

www.rr-cirkits.com/clinics/MER-2021-Signaling with LCC-B.pdf

What is LCC?

LCC is an information highway for your model railroad layout

What is LCC?

- **LCC is a common language for layout elements to talk to each other**

- Signals
- Turnouts
- Detectors
- Lights
- Panels
- PCs / Smart Phones
- Boosters
- Command Stations
- Throttles
- Power Managers
- Trains
- etc…

What is LCC *NOT*?

LCC does NOT replace DCC.

On the track – DCC

Beside the track – LCC

LCC is not dependent on DCC,

can run on DC or Märklin layouts

not locked to the DCC manufacturer

Why LCC?

- The critical role of software and its importance to our hobby, both now and for years to come, cannot be overstated. Our new digital world and its eventual successors will underpin much of our future innovations across a wide range of technological advancements.
- The ability of LCC nodes to be quickly and easily upgraded over the bus without the need to return them to the manufacturer, purchase replacement chips, or to even access them physically is a key new benefit.

Why LCC?

- Last week it was said on the NCE list that LCC was a solution looking for a problem.
- That is true, but it actuallypoints out the problem. We have LocoNet, CMRI, XpressNet, MERG, plus other proprietary methods to connect our devices.
- Not one of these can connect to another. (except via JMRI and a central computer)

Why LCC?

- Many of us simply use the DCC itself to control devices. That has its own problems.
- First and most restricting, DCC is a one way street. Has anyone here ever seen a DCC connected block detector, fascia control button, turnout position feedback contact, or any other input?
- Second, DCC does have a fixed, limited, bandwidth. Control traffic competes with the repetitive locomotive control information. This is probably not an issue on a 4x8 class layout, but not a good basis for expansion over the next 20-30 years.

Why LCC?

- DCC is a Master–Slave system. This means that there is really no practical way for more than a single command station to control any given layout.
- DCC has a fixed address space. This was seen to be sufficient in the 20th century when it was designed, but today folks run into the limits, and tomorrow requires looking for new tools.
- A single LCC node has a larger address space than many DCC systems can muster.

Other Options

- What about CMRI, XpressNet, MERG, plus the LocoNet and other proprietary methods used to connect our devices?
- Most of these solutions originated due to the difficulty in using the DCC bus for any input information.
- Some are Master–Slave and others have limited accessibility due to licensing.

A solution is proposed

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

A solution is proposed

- The bus must use license free commercial standards for its communications as much as is possible.
- It should be robust and viable even into the next generation of electronic products.
- It should be a peer-peer design with no requirements for any central control station.
- Any two devices from any manufacturers must be able to directly exchange data with each other.
- The Open LCB group was chosen to develop this.

- The result was a set of protocols that can be sent over any media. For example, EtherNet, Wi-Fi, CAN (Control Area Network), and others. (some say tin cans and string, but don’t believe it)
- The NMRA calls this Open LCB standard LCC. Layout Command and Control. LCC is NOT a replacement for DCC. (unless you consider it replacing DCC accessory decoders)
- LCC can run along side of DCC, AC, DC, DCS, TMCC, RailPro, Battery power, etc. It is not a way to power your trains, it is a way to control your entire layout.

•LCC Vocabulary

First lets explain some LCC terms to eliminate any mystery.

Message protocol – With a wire we normally carry the state of something by putting a voltage on the wire. If a switch is closed it supplies power to a lamp, or things like that. We model railroaders all understand that. With a message based protocol we no longer place a voltage on the wire. Instead we send two (at least) messages over the wire.

One message says ‘the switch just closed’, and another message says ‘the switch just opened’. For a single switch this is way too complicated. However by allowing many messages you can send information about many different switches over that same wire. I have visited layouts with wrist sized cables coming from a control panel, each carrying information about one switch or one lamp. By using a message protocol that same panel could have been connected using just a few wires.

- Node – A device that connects to an LCC network and interacts with it. The RR-CirKits Tower LCC is a node. The LCC Buffer-USB is not a node because it simply passes information from the network to a computer. However the computer then becomes a node because it is now connected to the network and interacts with it. For example, a node is what changes a switch closure into a message, or a message into a voltage change to light a LED or control a turnout.

- Node ID – The Node ID is a unique 48 bit address used to identify each LCC node. This allows up to 16,777,215 nodes per manufacturer x 254 NMRA listed manufacturers, and is expandable to individuals and other groups such as non-NMRA manufacturers.
- Bus – The method by which one node connects to another node. One common LCC bus is the CAN bus used in automobiles and aircraft carriers. However LCC may also use Wi-Fi or Ethernet as its bus. (tin cans and string have not been tried, but probably will not work)

- CDI – **C**onfiguration **D**escription **I**nformation. This includes both the description of what the node does, and any settings that you may have configured in it. The CDI is actually saved in the node itself, not some computer program or file.
- Producer – Some item that produces an EventID in response to some event on the layout.
- Consumer – Some item that responds to (consumes) an EventID to do something on the layout. Note: Normally the same EventID is used as both producer and consumer based on viewpoint.

- Event – Anything that happens or should happen on your layout.
- EventID – A message used by the LCC to **ID**entify some **Event** that happens on your layout. These EventIDs are simply unique numbers sent over the bus to represent an event. (these are the numbers that we are going to ignore in this article)
- Globally Unique EventID – A message number that it large enough to guarantee that it is unique in the world.

- Yes, we once heard that in 1981 Bill Gates allegedly said that 640K was more than enough memory space. (but he denies ever having said that) Suffice it to say, you are not going to run out of unique message numbers on your layout. (or even all the layouts in the world even counting G scale)

Why use such large Numbers?

Maybe you have already guessed that LCC has a theoretical message address space larger than the US national debt expressed in pennies. If that is true, (and it is) then how can anyone possibly understand it or keep track of them. Fortunately the answer to that question is; the very reason that the address space is so large in the first place is so that we don’t need to assign and keep track of these numbers at all.

Because the message (EventID) numbers used by LCC are ‘Globally Unique’, no two messages that we will ever find in use on one LCC system will be the same as those found on any other system unless we make it so.

That actually solves a huge issue right at the start. Unlike any legacy model railroad control systems, there is no reason that we ever need to assign and keep track of these numbers ourselves in order to avoid potential conflicts. Conflicts will never happen in the first place. How much modeler’s time is spent keeping track of cab numbers or loco numbers to prevent conflicts on our layouts, or even worse in our clubs where folks come and go.

If you have been doing turnout control or signaling using DCC then you already know that you have even more accessory address numbers to keep track of. Probably in your modular group there is one person in charge of assigning address ranges to cabs, modules, and individual club members, all of this work is simply to avoid potential conflicts.

With LCC that member can spend his time building scenery or running trains, rather than assigning and keeping track of random numbers.

You may say; “OK if we need so many message ID numbers to prevent any possible conflicts, how can I possibly set things up without ever using them?” The answer to that question is actually pretty simple. As humans we give names to things. I live in a town called Waxhaw in NC. Its ZIP code is 28173. If you were to ask me; “Where do you live?” what would you expect me to say? I could answer “Waxhaw”, or I could answer “28173”, as technically both are correct. However, if I actually did say “28173” you would probably look at me funny and step back to give me some more space. However, a mail sorting machine in Chicago does not look at the word “Waxhaw” in my address, because it is meaningless information to a sorting machine in Chicago, even though there is no other town in the US (or probably the world) named Waxhaw. It reads the “28173” portion of the address and routes it onto a mail truck bound for 28--- which carries it off to North Carolina, and eventually to Waxhaw.

It is the same thing with LCC. The electronics needs to see a unique number, but you as the layout owner only want to see “CP 106 Catskill North - diverging” or “Block 27.1 occupied” or however you like to identify things on your own layout. You have spent many hours creating the “Catskill” passing siding with its control points 105 and 106, so you will remember where it is. If the turnout is called 12414 or MT14, then maybe not so much.

LCC Basic Concepts

- One of the early assumptions made by the OpenLCB developers was that the nodes should be peer-peer, globally unique, and self describing.
- Without these features LCC has little to differentiate itself from other legacy solutions. Granted it can be faster, and more reliable, but that is not in itself any reason to make a fundamental change in how we do things.

LCC Basic Concepts - Peer-Peer

- In general networks come in two varieties. They are
  Master–Slave and
  Peer–Peer
- As implied, **Master–Slave** has a master device that controls all communications with remote nodes. (slaves) Usually this is done in a round robin method where the master unit polls each slave in turn to send/receive data.
- **Peer–Peer** on the other hand allows each node to communicate directly with every other node on an equal footing.

LCC Basic Concepts - Peer-Peer

- The advantage of a **Master–Slave** network is that the master controls the network timing and there are no collisions to detect and/or resolve. The big disadvantage is that nodes can not know the status of any other nodes. The (single) master device decides what will be done with all information.
- The advantage of a **Peer–Peer** network is that any/every node can watch and/or act on information from any other node. The disadvantage is that each node needs to monitor for any message conflicts and resolve them.

LCC Basic Concepts - Globally Unique

- In addition, every legacy model railroad control network that I am aware of is plagued by an identity crisis. To communicate on a network the first order of business is to give each device an ”address”. This is required in order to create a sense of identity and give order to network communications. Usually this is the first step in configuration. For example, you are not allowed to simply plop down a new locomotive on a layout and expect it to operate in an independent manner without first assigning it a unique (to that layout) address.

LCC Basic Concepts - Globally Unique

- The OpenLCB design folks recognized this flaw, and established an addressing convention large enough to allow factory preassigned globally unique addresses to each potential LCC node, world wide. Each NMRA DCC manufacturer could build over 16,000,000 nodes before they will run out of their own unique addresses and need to apply to the NMRA for a new address range. I think that the OpenLCB guys have this requirement covered for long enough that I will not be worried about it.

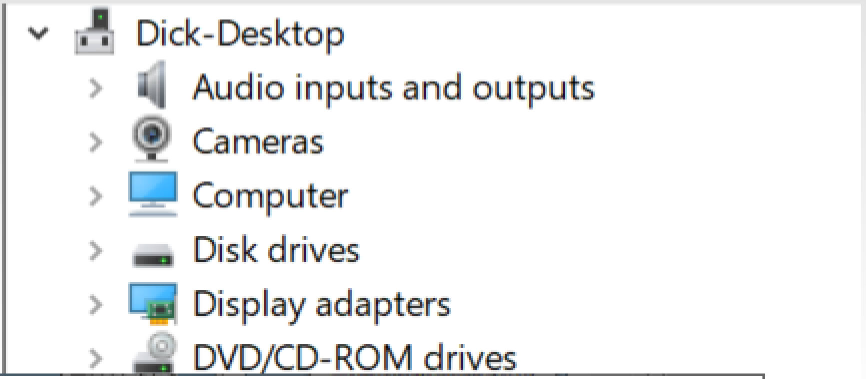
LCC Basic Concepts - Globally Unique

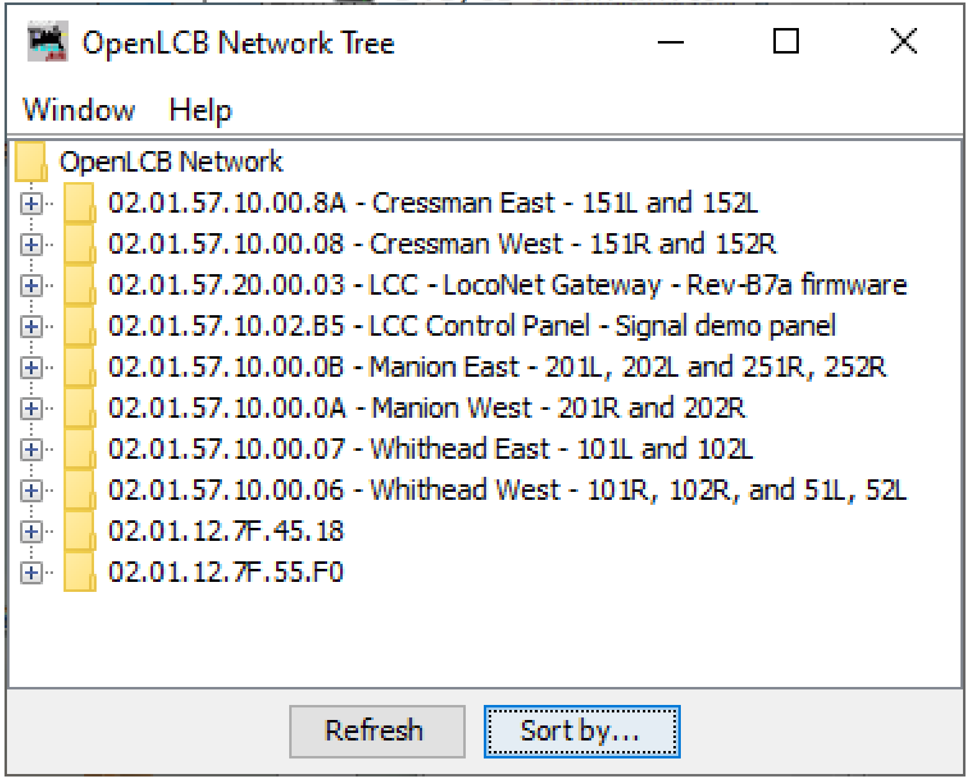
- Some of you may be old enough to remember when adding a serial port or printer port to your computer required you to set address switches and IRQ option jumpers on the I/O board.
- Is there anyone here today that did that with the latest computer that they purchased?
- Is there anyone here today that had to assign the address of the latest piece of model railroad electronics that they purchased? Enough said...

LCC Basic Concepts - Self Describing

- When I add a new device to my modern computer, I expect it to automatically show up in the device manager as seen here.
- Why shouldn't we expect the same thing when we add a new node to our layout?

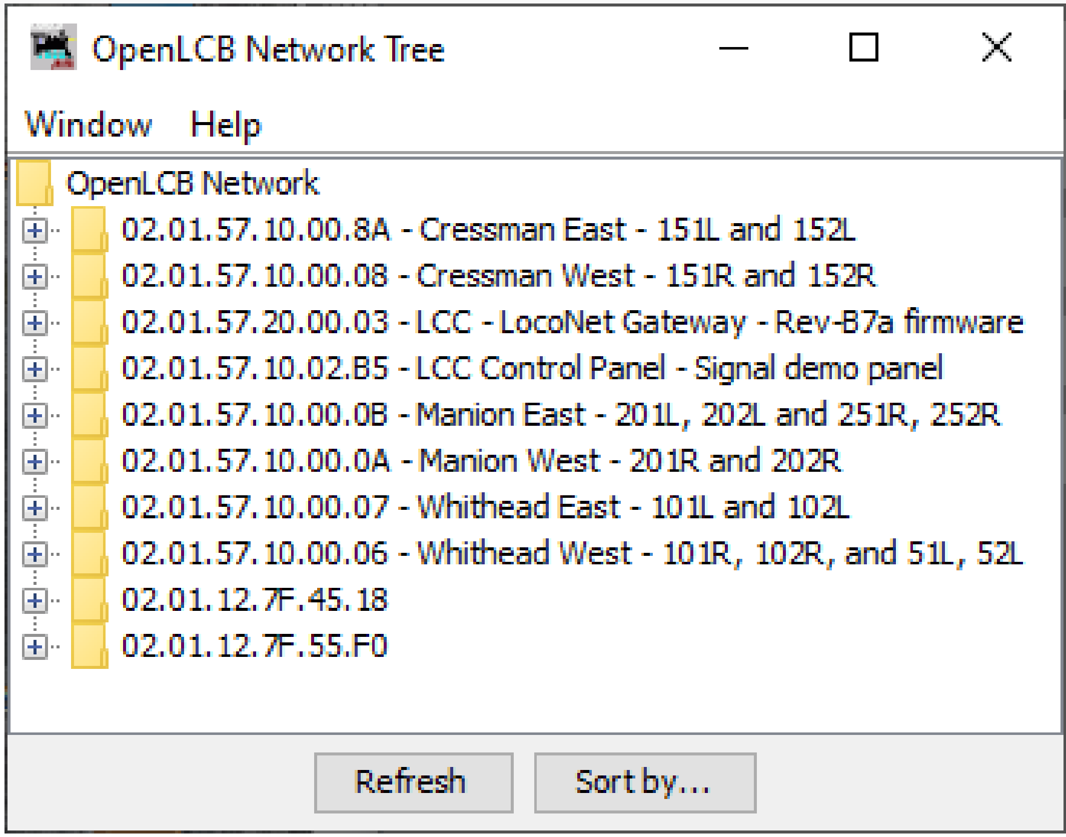
  Now, with LCC, that is a 21st century reality.


Dick-Desktop
Audio inputs and outputs
Cameras
Computer
Disk drives
Display adapters
DVD/CD-ROM drives


OpenLCB Network Tree
Window Help
OpenLCB Network
02.01.57.10.00.8A - Cressman East - 151L and 152L
02.01.57.10.00.08 - Cressman West - 151R and 152R
02.01.57.20.00.03 - LCC - LocoNet Gateway - Rev-B7a firmware
02.01.57.10.02.B5 - LCC Control Panel - Signal demo panel
02.01.57.10.00.0B - Manion East - 201L, 202L and 251R, 252R
02.01.57.10.00.0A - Manion West - 201R and 202R
02.01.57.10.00.07 - Whithead East - 101L and 102L
02.01.57.10.00.06 - Whithead West - 101R, 102R, and 51L, 52L
02.01.12.7F.45.18
02.01.12.7F.55.F0

LCC Basic Concepts - Self Describing

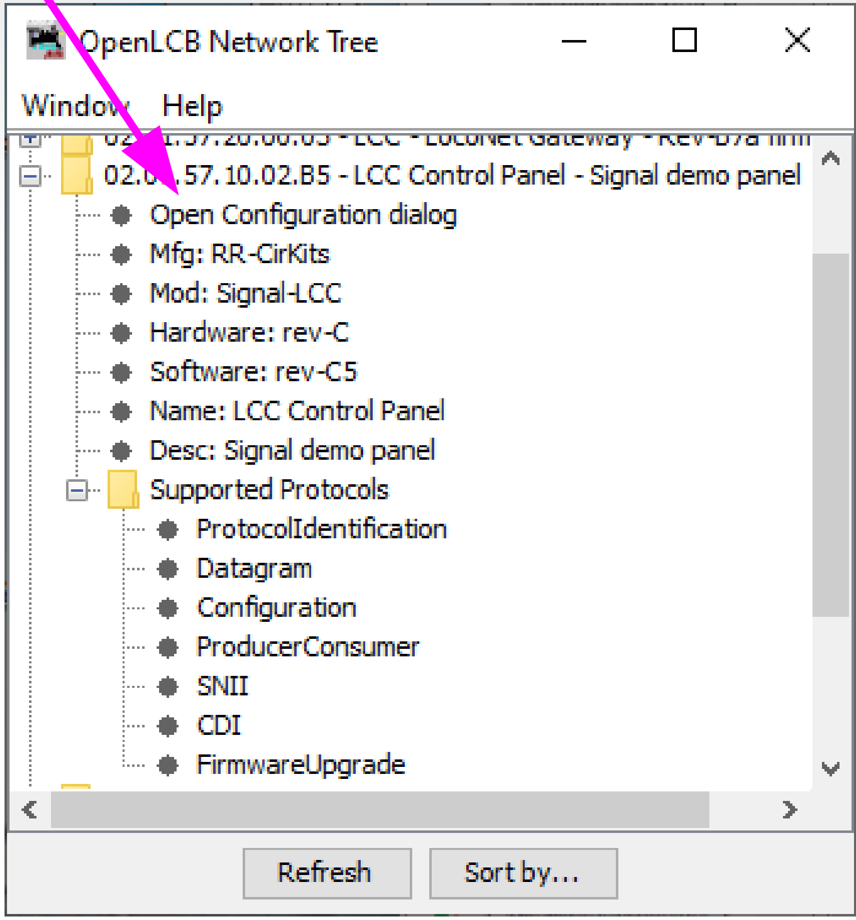
- With DecoderPro if you want to program a decoder you need to first find the correct ”decoder file” and then place the decoder in programming mode, maybe with a jumper or by putting it on a dedicated ’programming’ track.
- With LCC you don’t tell the system what hardware you have added. The system tells you what hardware you have added, what you have named it, and what capabilities it has.



```
OpenLCB Network
02.01.57.10.00.8A - Cressman East - 151L and 152L
02.01.57.10.00.08 - Cressman West - 151R and 152R
02.01.57.20.00.03 - LCC - LocoNet Gateway - Rev-B7a firmware
02.01.57.10.02.B5 - LCC Control Panel - Signal demo panel
02.01.57.10.00.0B - Manion East - 201L, 202L and 251R, 252R
02.01.57.10.00.0A - Manion West - 201R and 202R
02.01.57.10.00.07 - Whithead East - 101L and 102L
02.01.57.10.00.06 - Whithead West - 101R, 102R, and 51L, 52L
02.01.12.7F.45.18
02.01.12.7F.55.F0
```

LCC Basic Concepts - Self Describing

- With LCC you simply select "Open Configuration dialog" and all required information is read from the node itself.
- This opens a DecoderPro like window that allows you to make changes.

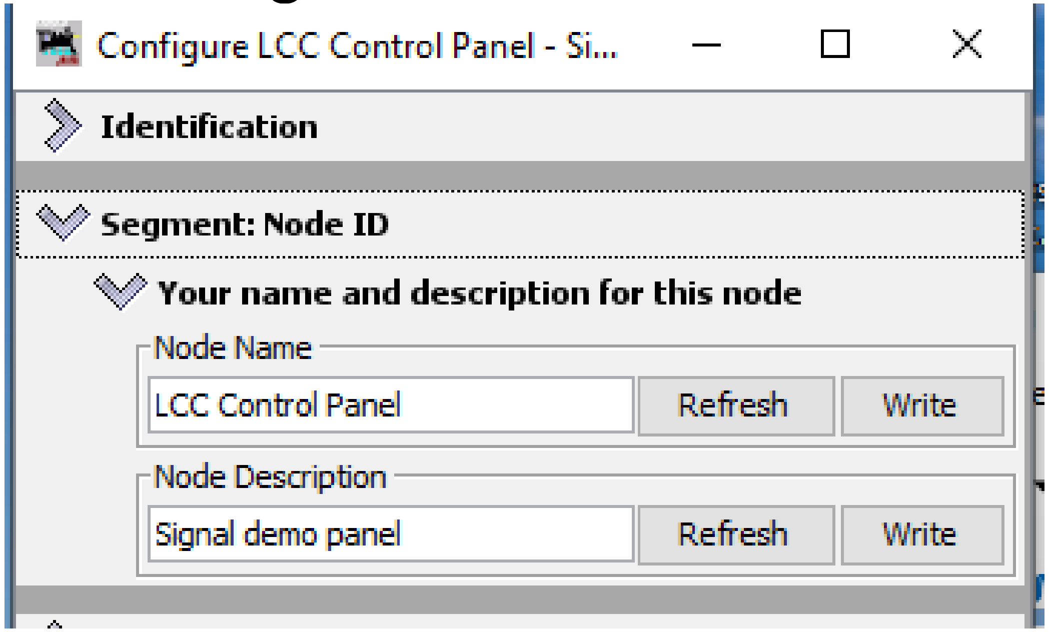


OpenLCB Network Tree

Window Help

02.01.57.20.00.03 - LCC - LocoNet Gateway - Rev-D7a firm

02.01.57.10.02.B5 - LCC Control Panel - Signal demo panel
- Open Configuration dialog
- Mfg: RR-CirKits
- Mod: Signal-LCC
- Hardware: rev-C
- Software: rev-C5
- Name: LCC Control Panel
- Desc: Signal demo panel
- Supported Protocols
  - ProtocolIdentification
  - Datagram
  - Configuration
  - ProducerConsumer
  - SNII
  - CDI
  - FirmwareUpgrade



Configure LCC Control Panel - Si...

Identification

Segment: Node ID

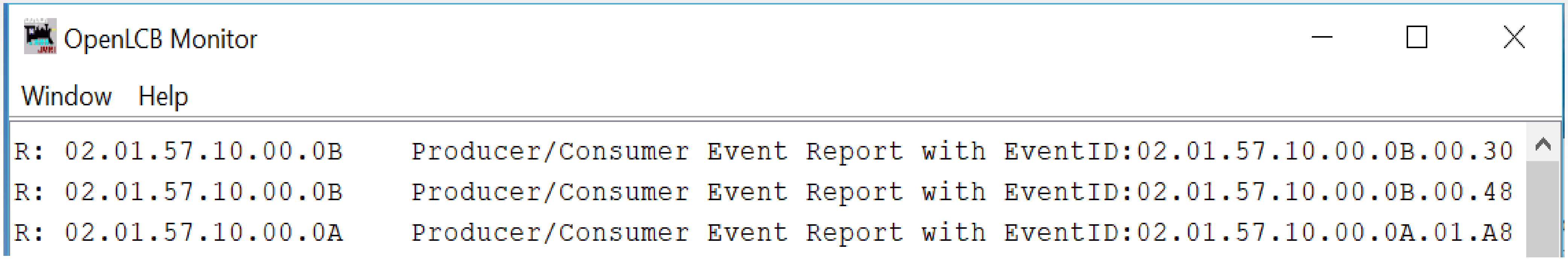
Your name and description for this node

Node Name: LCC Control Panel

Node Description: Signal demo panel

LCC Basic Concepts - Event driven

- ”Event driven” means that the nodes communicate with one another by sending messages whenever something happens.



```
R: 02.01.57.10.00.0B    Producer/Consumer Event Report with EventID:02.01.57.10.00.0B.00.30
R: 02.01.57.10.00.0B    Producer/Consumer Event Report with EventID:02.01.57.10.00.0B.00.48
R: 02.01.57.10.00.0A    Producer/Consumer Event Report with EventID:02.01.57.10.00.0A.01.A8
```

The event messages look like this, but you didn’t really need to know that, anymore than you need to know the format or content of the messages that your car’s O2 sensor uses when it sends messages to the ECU. If there is anything to remember, it is that 02.01.57.10.00.0B.00.30 (144,492,389,284,380,720) is a really large number that isn’t in any immediate danger of causing conflicts with other events from other devices on your layout.

LCC Basic Concepts - Event driven

- I will repeat this bottom line again here. EventIDs are simply magic numbers that represent your information on the bus, or over the air. There is no reason that you should ever need to type one out manually. There is little if any reason (other than curiosity) that you would ever need to know any details of what they mean. (which isn't a whole lot anyway)

LCC Basic Concepts - Event driven

- Its the Event ma'am, just the Event.
- In previous control systems that use a bus and events, (e.g. LocoNet and in a lesser sense CMRI) the events or messages sent on the bus have two parts, first an identifier number (address), and second the message type. This follows the original code line concept where each event was a hard coded station number plus one or more commands. For example: *turnout #23 set normal*.

LCC Basic Concepts - Event driven

- This is:

  1. a Turnout command
  2. for station #23
  3. set to normal

- A matching command with a predefined one bit different would mean *turnout #23 set to reverse.* Another one bit change would create *turnout #24 set to normal* etc.
- Sometimes the size of the DCC command space and the protocol design limits the number of possible options to a predefined set. (e.g. 2048 turnouts, 4096 sensors, etc.)

LCC Basic Concepts - Event driven

- For example turnouts only have two options, normal and reverse. If you have a three way turnout, (very rare on the prototype) sorry, you need to think of it as 2 two position turnouts. Have a three color signal, sorry, you need to think of that as either three different on, off, messages, (CMRI) or else combine two 2 position messages. (LocoNet) What about a more typical eastern US speed signal with 5, 6, or even more aspects?

LCC Basic Concepts - Event driven

- In the LCC world an event has no predefined meanings. None, Keiner, Nada! An LCC event simply says; ’something has happened’, or ’something should happen.’ How it is defined is 100% up to you, the user. In our previous example it could still mean *turnout #23 set normal*. However with LCC 'turnout #23' is just what you call it on your layout, not that it was pin 23 on some brand of hardware controller. *Set normal* just means that the event moves the turnout to normal. Undoubtedly you will want another event to move the turnout back, however that will be a completely different event with a different meaning. (e.g. *turnout #23 set reverse)*

LCC Basic Concepts - Producer Consumer

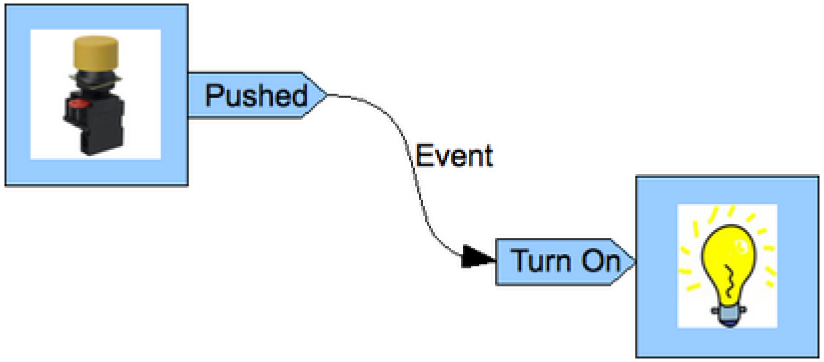
- **Producer - Consumer** You will probably hear LCC folks throwing around terms like Producer and Consumer. They aren't talking about a big business takeover. They are just trying to sound educated. <G> The Producer-Consumer control concept is used for process controls and software queuing.
  - ***Producer*** simply means that some device can create (produce) an Event. Some examples might be a push button or block detector.
  - ***Consumer*** just means that some device can respond to (consume) an Event. It could be a lamp, a turnout driver, or anything else that you can control.
  - ***Events*** can have from 1 to many *Producers*. Events can have from 0 to many *Consumers*. Events are simply messages on the bus that say that something has happened or should happen.

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

LCC Basic Concepts - Producer Consumer

To elaborate a little bit. For an event to happen something must have sent it. Therefor there has to be at least one *producer*. In the LCC world it is possible for many different *Producers* to create the same event. For example you might want to have turnout control buttons track side and on a remote panel. Thus the statement that every Event has one or more *producers*.

Producer



Consumer

LCC Basic Concepts - Producer Consumer

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

Step 1. Basic Equipment

- An LCC network needs something to connect one node to another. For sake of this article we are assuming a wired network using the CAN bus over RJ45 network cables. There are other options such as Wi-Fi but that is beyond the scope of this introduction.
- An LCC network really should have access to a computer interface to make it easy to configure. An LCC Buffer-USB is shown. It is only required during configuration or when using JMRI or another program as a node.

- An LCC network also needs power for its nodes. This may be supplied directly to the nodes if so equipped, or it may come from the network cable for low powered nodes. An LCC Power-Point powers the Signal Demo network over the CAT5 cables.
- An LCC network using the CAN bus needs bus terminators at each end of the cable.

Step 2. Hooking it all together.

Each node and other LCC device has two connectors. Plug CAT5 cables between them in a series string from one to the next in any order desired. Plug the two terminators into the unused end connections. Apply power to the network at the power point. You are now done and have an operating LCC network.

Obviously, on a large network I would distribute multiple power injection points evenly around the cable. CAT5 cable is only good for carrying ½ to 1 amp in any given section. On the other hand it is very convenient to run individual nodes directly from the network cable rather than supplying power over separate wires and power supplies to each one.

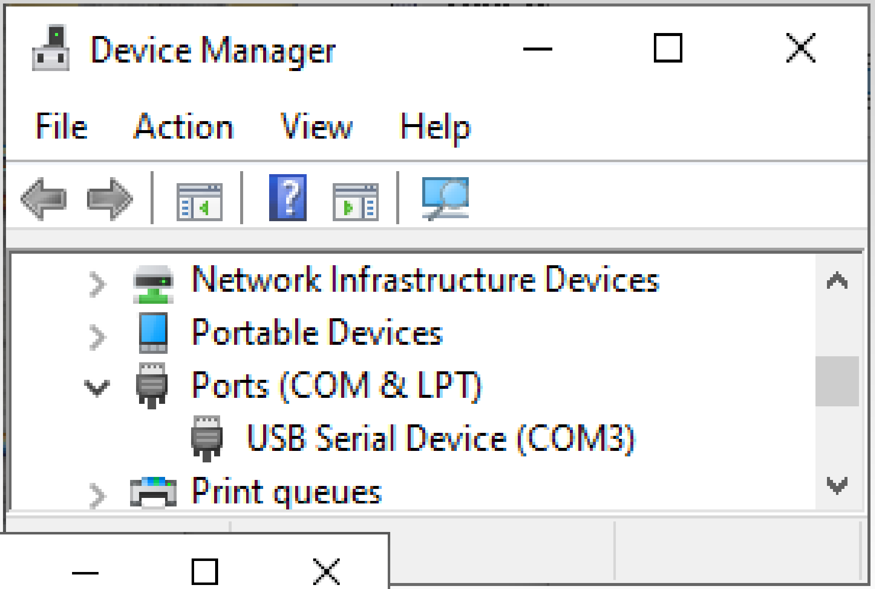
Step 3. Connect JMRI to our LCC Network.

Once we have an LCC network running we need to hook it up to JMRI. (available free at www.jmri.org) There are other programs available to do this configuration step, but we will use JMRI because it is pretty easily available for all platforms, and its functionality with LCC is superb.

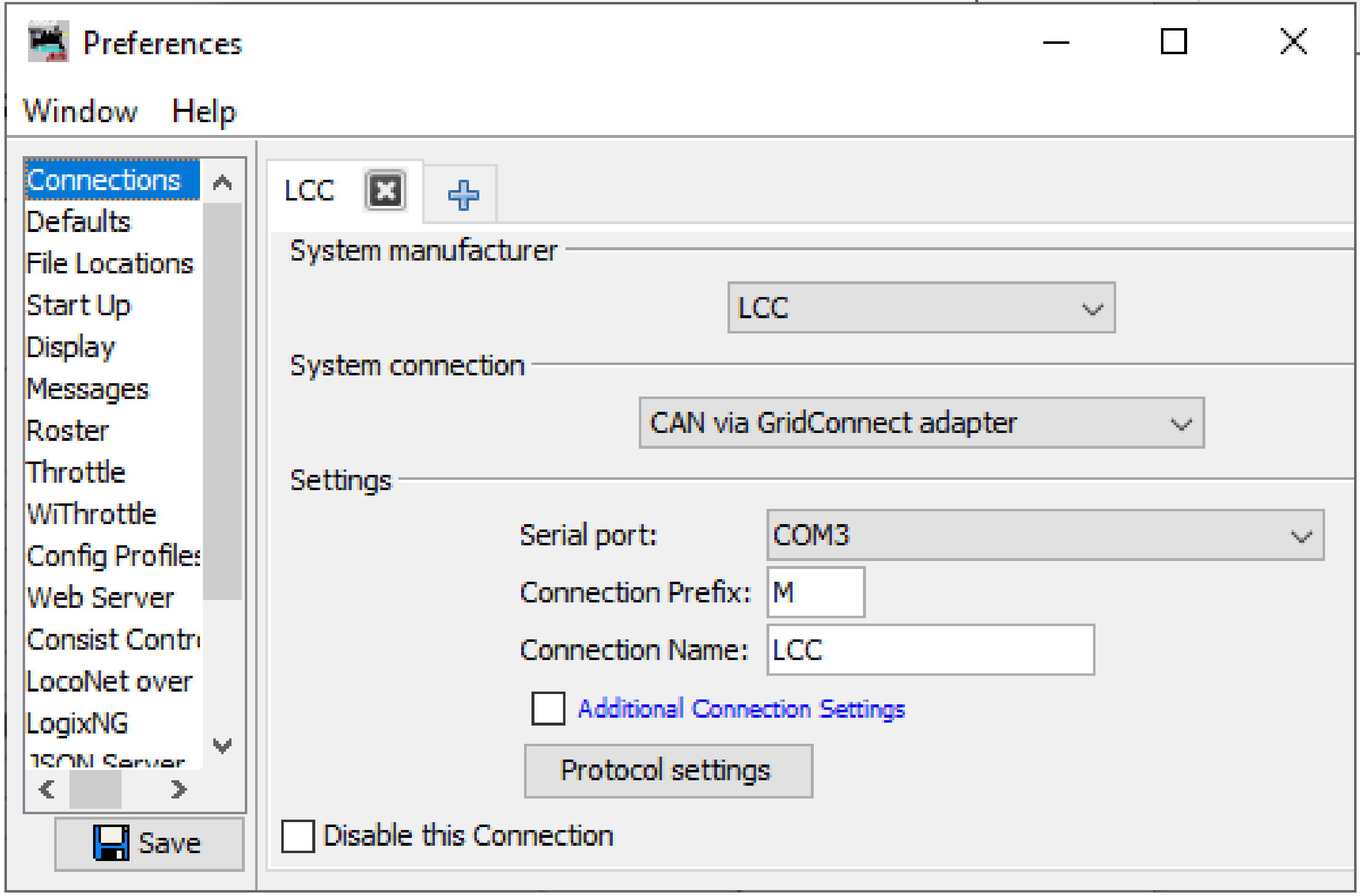
Open your Device Manager on Windows. (or equivalent for Linux or Mac)

Plug in the LCC Buffer-USB. (or other interface) notice which new connection appears in the *Ports (COM & LPT)* section and make a note of it.

- In this example it was "USB Serial Device (COM3)" that appeared.
- Now start JMRI and navigate to the 'Edit – Preferences – Connections' window.



Network Infrastructure Devices
Portable Devices
Ports (COM & LPT)
USB Serial Device (COM3)
Print queues

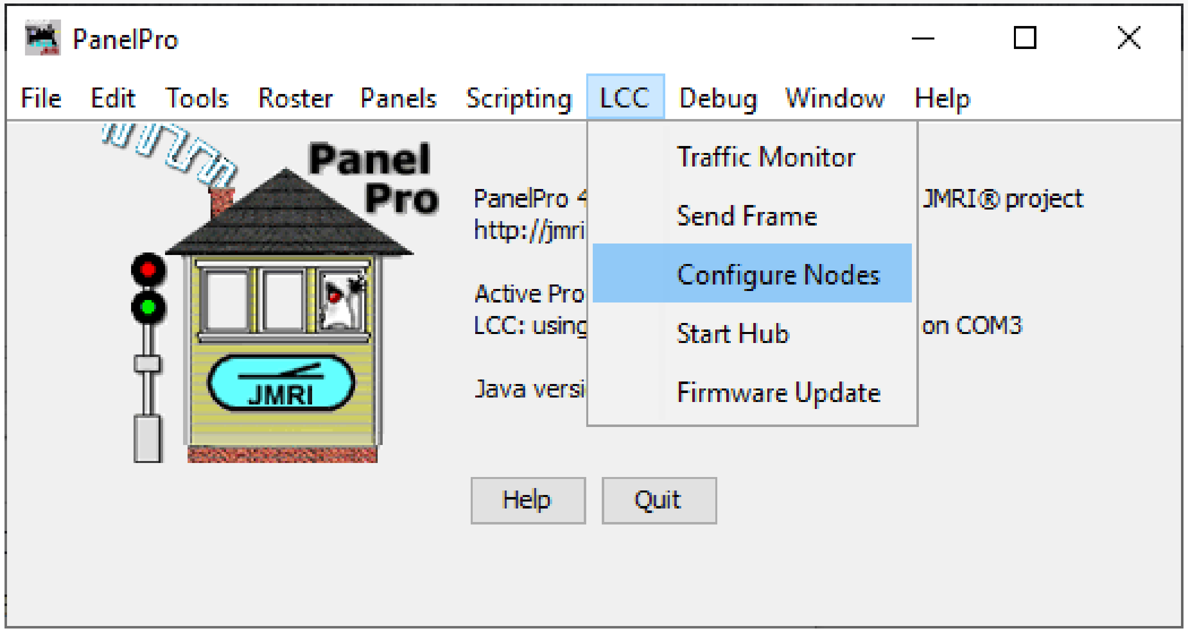


System manufacturer: LCC
System connection: CAN via GridConnect adapter
Settings
Serial port: COM3
Connection Prefix: M
Connection Name: LCC

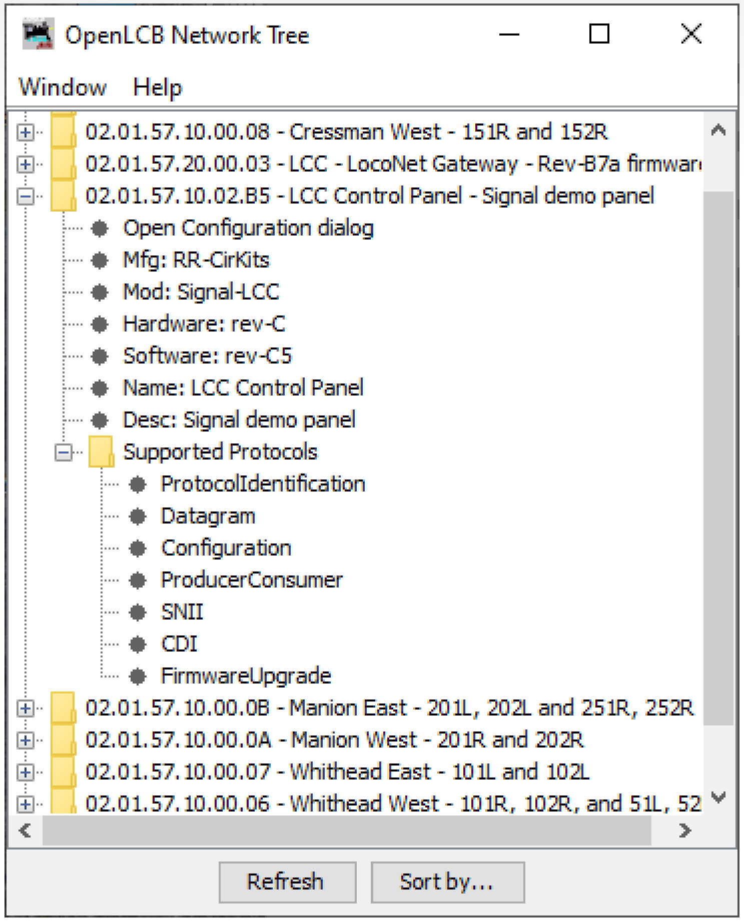
- Enter the ‘System manufacturer’ as ‘LCC’, the ‘System connection’ as ‘CAN via LCC Buffer-USB’, (or whatever your interface device is called) and the ‘Serial port:’ as ‘COM3’ or whatever port your interface appeared on.
- Click on [Save] and wait for JMRI to restart. JMRI should now be communicating with your LCC network as a node so that it can interact just as any other node would. It can then show us what is happening on our network, and modify nodes using the CDI.

Step 4. Getting Started with JMRI

- Once JMRI has restarted, and using PanelPro, we see this window which now knows about the LCC connection.
- Click on the 'LCC' drop down list and select 'Configure Nodes'
- This gives us the Window that we have seen previously.

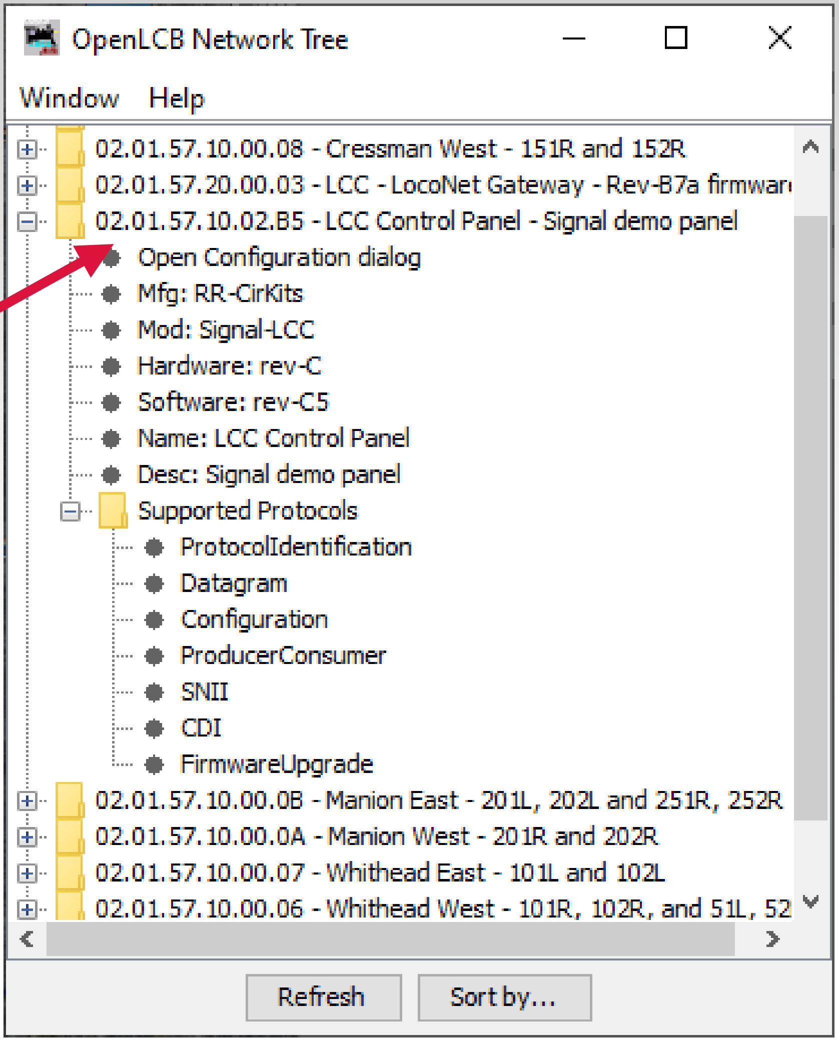


- ‘Configure Nodes’ simply queries the LCC network and creates a list of connected nodes.
- Now we see some more information about the node. We could further explore the ‘Supported Protocols’, but unless you are interested in geeky things, this is of no practical use to us. It simply tells us that this node can send its status information to other nodes when it starts up, can be firmware upgraded over the network, can be configured over the network, supports user names, etc.



```
⊞ 02.01.57.10.00.08 - Cressman West - 151R and 152R
⊞ 02.01.57.20.00.03 - LCC - LocoNet Gateway - Rev-B7a firmwar
⊟ 02.01.57.10.02.B5 - LCC Control Panel - Signal demo panel
    Open Configuration dialog
    Mfg: RR-CirKits
    Mod: Signal-LCC
    Hardware: rev-C
    Software: rev-C5
    Name: LCC Control Panel
    Desc: Signal demo panel
  ⊟ Supported Protocols
      ProtocolIdentification
      Datagram
      Configuration
      ProducerConsumer
      SNII
      CDI
      FirmwareUpgrade
⊞ 02.01.57.10.00.0B - Manion East - 201L, 202L and 251R, 252R
⊞ 02.01.57.10.00.0A - Manion West - 201R and 202R
⊞ 02.01.57.10.00.07 - Whithead East - 101L and 102L
⊞ 02.01.57.10.00.06 - Whithead West - 101R, 102R, and 51L, 52
```

- Of note is that all of this information is stored in the node itself, not in JMRI, which is simply serving as a handy tool to let us view and change things in the node.



02.01.57.10.00.08 - Cressman West - 151R and 152R
02.01.57.20.00.03 - LCC - LocoNet Gateway - Rev-B7a firmwar
02.01.57.10.02.B5 - LCC Control Panel - Signal demo panel
Open Configuration dialog
Mfg: RR-CirKits
Mod: Signal-LCC
Hardware: rev-C
Software: rev-C5
Name: LCC Control Panel
Desc: Signal demo panel
Supported Protocols
ProtocolIdentification
Datagram
Configuration
ProducerConsumer
SNII
CDI
FirmwareUpgrade
02.01.57.10.00.0B - Manion East - 201L, 202L and 251R, 252R
02.01.57.10.00.0A - Manion West - 201R and 202R
02.01.57.10.00.07 - Whithead East - 101L and 102L
02.01.57.10.00.06 - Whithead West - 101R, 102R, and 51L, 52

Step 5. The CDI

- The first item we need to check out is the one called "Open Configuration dialog". That opens a new window showing what LCC calls the CDI. (Configuration Description Information) To create the CDI window JMRI does two reads from the node. The first read is the descriptive information itself, stored in the node. It describes all the things that the node can do and how to display it. This part of the CDI is written by the manufacturer, and is similar to a JMRI decoder file, but instead of being part of JMRI, it is stored in the node itself.

- There are a few reasons for this. One, it means that other programs beside JMRI can open and configure the same information.
- Another, it means that the configuration information remains with the node itself. Your club member that does configurations can take the node to his desk at home to work on with his own computer if he wants to. There is no need to synchronize a “Roster” file to carry along with it.
- It also means that if a new node is released you do not need to wait for a new version of JMRI to be released in order to configure it.

Configuration of LCC nodes

- One of the key new concepts in the LCC protocol is that, not only the configuration, but the ’decoder file’ (in JMRI terms) itself should reside in the LCC node. This is an important change from the status quo.
- Originally hardware had a fixed purpose. Each required its own dedicated connections. Lionel crossing gates flashed with contacts triggered by the passing wheels. (blink-blink....blink-blink....)

Configuration of LCC nodes

- Then some devices were connected to a bus. (or track) This required assigning addresses or channels. The usual solution for addressing was to include a set of jumpers or switches for the selection. In some cases it was a plug with different component values.
- As electronics improved the selection of addresses was moved into the device code itself. An example that we are all familiar with is modern DCC mobile decoders.

Configuration of LCC nodes

- One of downsides of this new method is that our decoders now need to be configured with a new (non default) address. That itself was automated by some manufacturers, but it soon became evident that something more was needed than simple interactions through a hand held throttle. Some of todays new decoders have 1000 or more values to configure.

Configuration of LCC nodes

- JMRI and other programs have come to the rescue, but the decoders are now so complex that a ’decoder file’ is required for each locomotive and stored on a computer to help keep track of changes. The DCC specification does not include an easy way to read information from a decoder except very laboriously and slowly over a special connection. (called a programming track)

Configuration of LCC nodes

- This manual address assignment was deemed to be too slow and inflexible for the new LCC equipment. Two key changes were required.
- The first was that any LCC node could be configured in place on the layout at any time with no need to access it for jumper changes or button presses.
- The second was that any information required to configure a node should reside in the node itself, and be available to any configuration tool connected to the network. Now any node could be configured in one place and moved to another with all the information moving with the node itself. This means not only configuration values but user names and comments as well.

Configuration of LCC nodes

- As previously mentioned, a key design choice of LCC was that the manufacturer would assign a node ID during manufacturing in a manner that prevents any duplication of addresses…. Ever, …. anywhere! (similar to Ethernet MAC addresses)
- This manufacturer based address assignment has another unforeseen benefit. Any automatic or user linking of two LCC nodes no longer needs to know anything at all about the rest of the layout in order to prevent unintended conflicts We will take advantage of this for signaling.
- Adding a new node to the layout will never conflict with any already installed devices.

Currently Available LCC Hardware

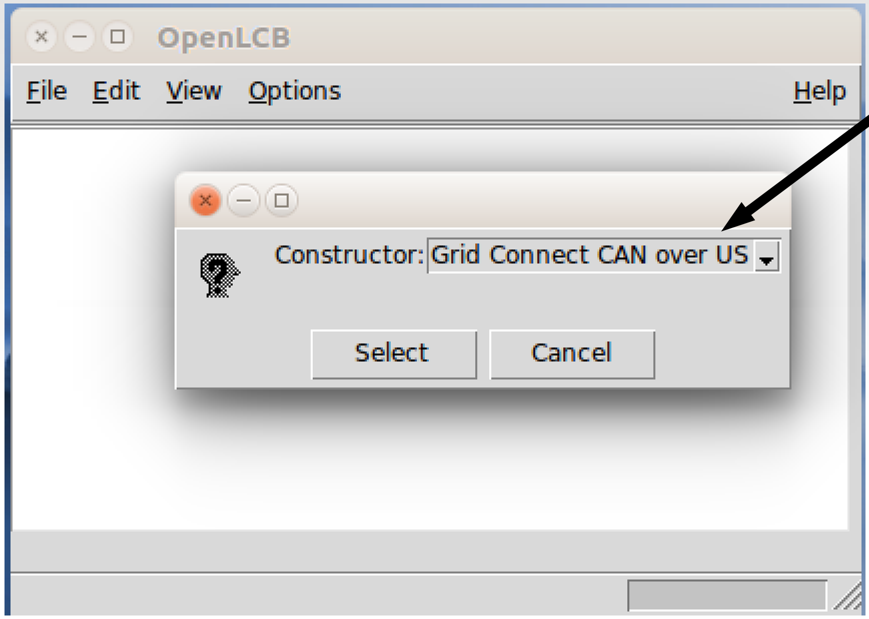
- With the recent addition of an option to place the DCC rail sync information on an otherwise unused pair, CAN can now support smart boosters.
- I have referred to the CAN version of LCC. Remember that the LCC protocol is also capable of being used over different systems, Ethernet, and Wi-Fi also being developed for use by other LCC developers.

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.
- *Servo controllers.
- LocoNet to LCC Gateway. (LCC support for existing products)
- *Ethernet LCC Links.
- *Wireless LCC Links.
- Throttles
- *Smart Boosters, *Command Stations.

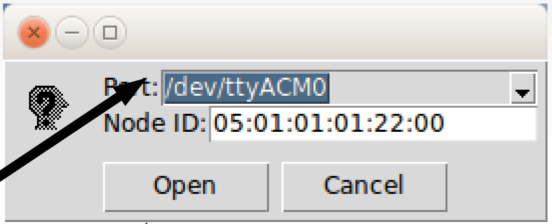
* denotes LCC nodes currently under development in 2021

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.



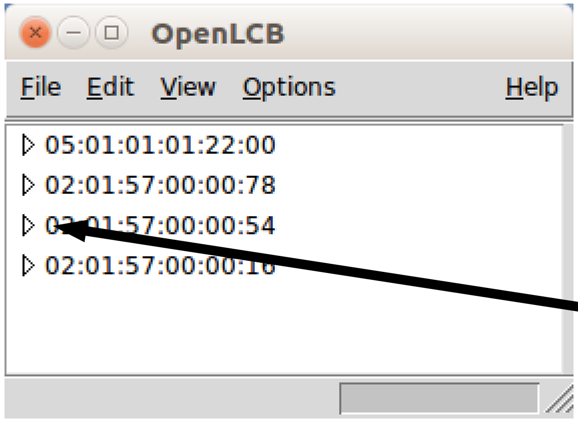
Constructor: Grid Connect CAN over US

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)



05:01:01:01:22:00
02:01:57:00:00:78
02:01:57:00:00:54
02:01:57:00:00:10

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.

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

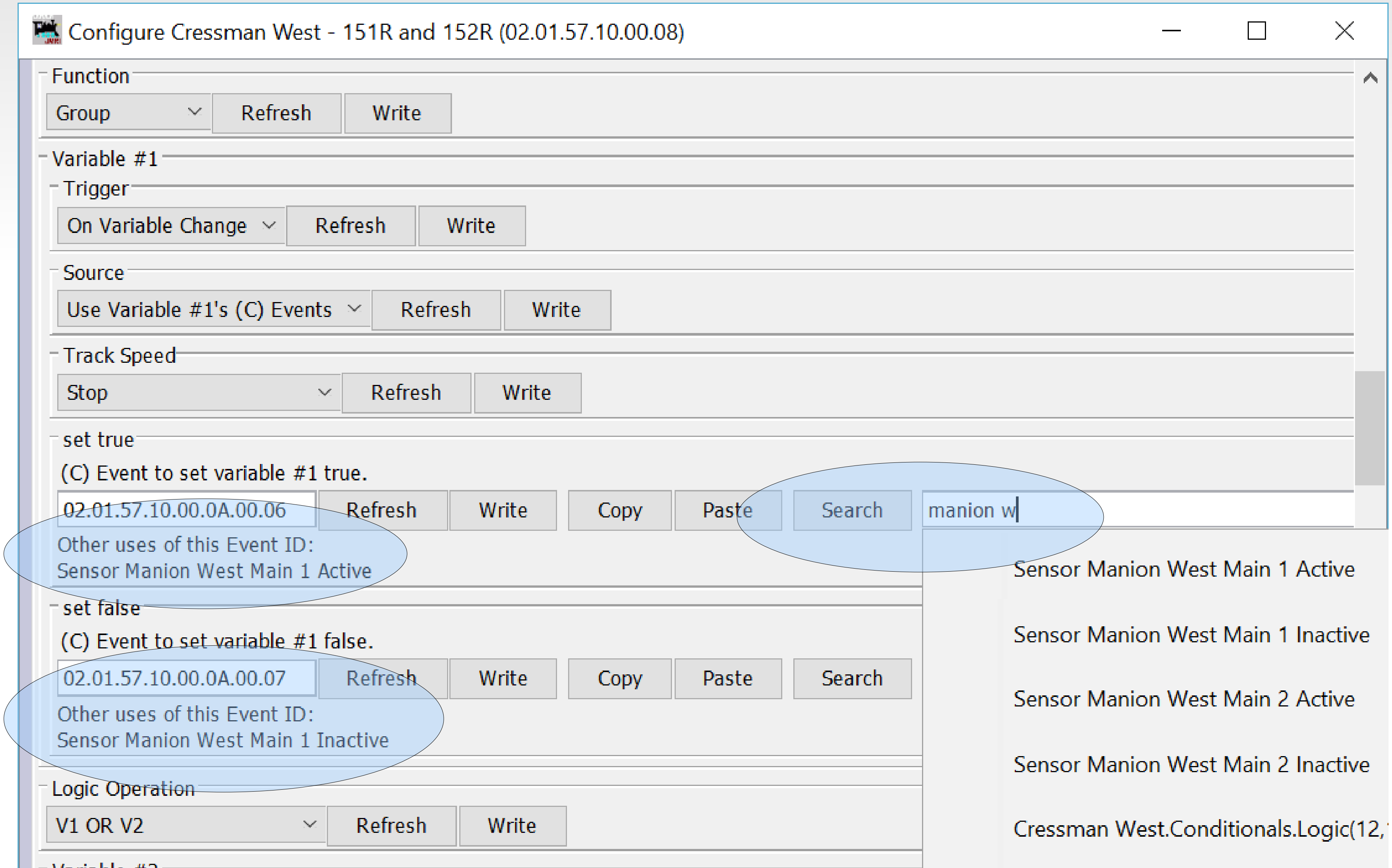
The Future of LCC

- I created the next slide back in 2017. I include it here for a little bit of perspective.

- 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)

The Future of LCC

- Look carefully at a current configuration presentation. Fortunately I have apparently outlived my pessimistic prediction.


Configure Cressman West - 151R and 152R (02.01.57.10.00.08)

Function

Group | Refresh | Write

Variable #1

Trigger

On Variable Change | Refresh | Write

Source

Use Variable #1's (C) Events | Refresh | Write

Track Speed

Stop | Refresh | Write

set true

(C) Event to set variable #1 true.

02.01.57.10.00.0A.00.06 | Refresh | Write | Copy | Paste | Search | manion w

Other uses of this Event ID:
Sensor Manion West Main 1 Active

Sensor Manion West Main 1 Active
Sensor Manion West Main 1 Inactive
Sensor Manion West Main 2 Active
Sensor Manion West Main 2 Inactive
Cressman West.Conditionals.Logic(12,

set false

(C) Event to set variable #1 false.

02.01.57.10.00.0A.00.07 | Refresh | Write | Copy | Paste | Search

Other uses of this Event ID:
Sensor Manion West Main 1 Inactive

Logic Operation

V1 OR V2 | Refresh | Write

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 This requires memory of the two events plus AND logic, or else using detectors that may be stacked so that a train in the island activates both island and approach detectors.
  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 when 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

OpenLCB User Group

https://groups.io/g/layoutcommandcontrol

Started November 2009
Oct 2021 we had over 380 members

Typically a few messages a day. A great source of info.

Info

Users Groups:

https://groups.io/g/openlcb
    https://groups.io/g/layoutcommandcontrol

To Join: openlcb+subscribe@groups.io
    layoutcommandcontrol+subscribe@groups.io

Useful Links:

http://openlcb.org or http://openlcb.com

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

Book: Introduction to Layout Command Control
        by Dana Zimmerli PhD

Questions

- ?