Introduction

In the June issue of Scale Rails, Didrik Voss introduced readers to some of the issues involved in developing an accessory bus for model railroad layouts. He outlined the history of NMRA DCC, his wish to develop a similar standard local control bus (LCB), and the initial mandate of the NMRA Net Working Group which was to be used as a guide for its development.

He also described one of the proposals, S9.5, in the article. This article picks up from there and describes another effort, called OpenLCB, and the NMRAnet/S9.6 proposal being developed by that team.

OpenLCB NMRAnet/S9.6 Proposal

To be a good value for model railroaders, a new model railroad control bus and the pieces it connects should be usable for 20 years or more. DCC, for example, has been in use since the early 90's, and although modern parts can do many things that weren't imagined back then, early and recent DCC decoders and systems generally work together fine.

At the same time, technology is moving forward quickly. The microprocessors that are in DCC decoders today are much more powerful than the ones that were available when DCC was standardized. To have a long life and do neat things, a new layout control bus should be able to take advantage of new technologies as they become available.

Finally, we have to remember the goal: The layout control bus exists to make it easy and cheap for a model railroader to get his layout to work exactly as he wants it to, without having to do a lot of complex things. Only a very few of us consider complex electronics to be a big part of the hobby. For most people, the most important thing a layout control bus can do is to hide all the electronic complexity by having the system itself handle the details for them. DCC is an example of both success and failure in this respect. It's a success, because it lets you run multiple engines independently, without the complexity of DC blocks, switches to route multiple DC throttles to each block, etc. At the same time, many modellers view the CV method of configuring locomotives as too complicated and too much trouble. Programs like DecoderPro have been created by model railroaders to smooth over that complexity, but there are limits to what DecoderPro and its ilk can do given how DCC was defined. A new layout control bus should be designed to remove those issues, and ensure it can grow while retaining simplicity for model railroaders.

The OpenLCB team has put a lot of thought and experimentation into these hard problems, so as to create an integrated solution that will serve novices to experts, and small to large layouts. We have concentrated on providing strong solutions in these areas:

1. Simplicity for the first-time user and small layouts. A novice can buy two S9.6 modules from different manufacturers, hook them together and simply teach them to interact with a few button pushes. No computers or control units needed to hook up something to control your yard throat, for example.
2. Enough capability for even the biggest layouts, such as museums and large modular layout meets: A virtually unlimited number of S9.6 modules can be connected in various ways, including cheap CAN connections, fast Ethernet links, or other types to interconnect huge layouts. S9.6 ensures that modules can be easily interconnected without worrying about address

conflicts or having to pre-assign addresses.

3. Solutions when things get too full or too busy: S9.6 lets the control bus wiring be split into multiple segments when large loads require it. It automatically controls traffic flow to prevent bus overload, so that the model railroader doesn't have to deal with tricky configuration.
4. Protect investment now and long term: S9.6 efficiently bridges to existing equipment so you can continue to benefit from your prior investments in today's DCC and other control systems.
5. Ability to grow with new features: Wireless devices as throttles, radio controls, and even unimagined new technologies will be coming to model railroading over the next twenty years. S9.6 is structured to permit adding completely new capabilities and protocols without interfering with continued use of controls that model railroaders have already installed.

Key to all this are the S9.6 methods for doing this automatically, so it doesn't require the model railroader to learn how to do complex configuration.

Simplicity, when things are new or small:

S9.6 simplifies life for the novice. The novice can simply plug nodes together, program them, and immediately control his accessories. Instead of entering numbers, such as configuring a switch to turn on a light by loading a board with "Send 1057 on", the user can just press a couple buttons on the light board, press some buttons on the board for the switch, and associate those two together permanently. Even complicated control of yard throats, realistic day/night lighting and other applications can be done without looking up numbers and complicated record-keeping.

S9.6 uses the "producer/consumer" communications from the NMRA Detroit meeting of 2007. This is somewhat different from the method used by S9.5 and described by Di Voss in the June 2010 Scale Rails, but the basic idea is the same. Changes in inputs cause "events", which "producers" send through the S9.6 links to "consumers" who are listening for them. The consumers then change the necessary outputs to turn lights on or off, move turnouts, or whatever else is the correct response to the event. This lets S9.6 users concentrate on what they want to control, and not on the nitty-gritty of what messages are needed to do it.

Node numbers are preloaded at the factory, so you don't have to configure them, keep track of them, or even care what they are if you don't want to. No central controller or configuration tool is needed to put an S9.6 system into operation, or to connect new nodes to it. Adding new controls to an existing system never requires reprogramming or reconfiguring of any of the existing nodes because of conflicts with the new hardware.

Enough capability:

As time goes on, layouts get bigger and have more things to control. More layouts have signals installed these days, for example, and the signals are becoming more prototypical and complex. A control system has to be able to grow with the layout. S9.6 starts with a robust connection, the controller area network (CAN) described by Di Voss in the June article. For larger layouts, where a single CAN link has been outgrown, S9.6 can connect segments together with higher-speed links. For example, at the NMRA 2010 meeting in Milwaukee, we showed two CAN segments connected by an Ethernet connection.

S9.6 is designed to handle the special configuration needs of large modular layouts, such as those at annual meets of Ntrak in North America and Fremo in Europe, see the picture to the right. Clubs' nodes cannot conflict, even when multiple clubs get together with modules that were programmed at

home, because S9.6 ensures that all nodes have different serial and event numbers -- none of the modules will need to be re-programmed due to conflicts between their addresses. The meet's organizers only have to configure any new connections and the control system is ready to go, without having to keep lists of who owns what module numbers, nor pre-allocate module numbers, channels numbers, or events. S9.6 also automates and simplifies the set-up and configuration of the modules, allowing the meet organizers to connect to and configure the individual modules, if necessary. Multiple people and computer programs can be simultaneously configuring and testing the layout from computers, including handhelds, all without interfering with each other.

Solutions, when things get too full or too busy:

As stated above, layouts tend to grow. When a layout outgrows a single CAN-segment, it can be easily split into two segments joined by a repeater, bridge or gateway. In addition, multiple CAN-segments can be interconnected by gateways via faster Ethernet,with the gateways automatically controlling segment traffic by reducing unneeded transmissions between the segments. Importantly, S9.6 has methods to do this automatically, and saves the model railroader from having to learn how to do complex configuration.

Protect your investment:

S9.6 let's you keep using your legacy equipment, such as a DCC system or other controls. Intelligent bridges can connect these to S9.6, and further integrate their devices right into the S9.6 event protocol by including their messages inside S9.6 messages. For example, DCC accessory commands are converted to S9.6 events, and they can directly address accessories attached via S9.6 without any extra programming. Other buses expected to be supported are Loconet, Xpressnet, NCE's throttle bus and C/MRI.

Ability to grow with new features:

Layouts have a habit of growing, both in extent and in complexity. The OpenLCB team has designed S9.6 to allow multiple computers to connect to your layout to help design, configure, debug, and if you want, to operate your layout. Ethernet, both wired and wireless, and allows directly-connected nodes, such as iPhone throttles, command stations, and even virtual nodes implemented in software on PCs.

NMRAnet Goals and Measures:

The OpenLCB team has worked hard to integrate the Goals and Measures developed by the NMRAnet committee. See: http://nmranet.sourceforge.net/NMRANetGoalsAndMeasures-2007-10-22.pdf

- **Interoperable** Products from one company can work with products from any other company, and with legacy systems.
- **No Central Control** Products can interact with other products without the need for a central processor.
- **Optional PC Control** A PC can provide higher-level functionality, such as a CTC Interlocking, and system functions such as monitoring, testing, and debugging.
- **Simple** Make it easy for a novice customers to install and configure without technical knowledge (should be easier than DCC systems).
- **Expandable** Allow very large layouts (a higher-level of technical knowledge required is

OK for large layouts).

- **Flexible** Customers can easily connect many-to-many devices, and connect to legacy systems.
- **Extensible** Allow additional functionality to be added easily, whether by the NMRA, or manufacturers.
- **Easy to Implement** Easy and inexpensive to implement products that are reliable and compatible for vendors wishing to create products.
- **Bi-directional** Support bi-directional exchange of information.
- **Free IP** The Standards and/or RPs should be free from intellectual-property restrictions from parties other than the NMRA.
- **Transport Agnostic** Allow different transport mechanisms for messages.
- **Train Agnostic** Support controlling trains when the network is connected to a train-control system.
- **Discoverable** Allow a user to find out what devices are connected and how they're configured on a layout.
- **Self Describing** Devices should describe themselves.
- **Testable** Allow components to be easily tested for compliance.
- **Compliance** A name protected by a trademark that NMRA can use to help ensure compliance.
- **First-Time Use** A new user can buy a device and have it work.

How does it all work?

The OpenLCB team decided the best way to achieve the above goals was to define an extensible family of protocols that are based on using automatically-assigned node numbers and event numbers to ensure uniqueness. These choices allowed us to achieve our priorities and (most of) the formal NMRAnet goals, while at the same time allowing simple nodes to remain simple and flexible.

Automatically-assigned Node Numbers

Each S9.6 node has an identifying number called its ID, which is essentially its serial number, which is unique to it in the whole world. This means that you do not need to supply an ID for it, so you don't have to keep track of what numbers you already have. This number was assigned to the board during its manufacture, and S9.6 defines procedures to ensure that's it's unique forever. S9.6 node IDs are 48-bit numbers, which means that a huge number of them are available and we'll never run out. Anyone who builds S9.6 nodes is given a large batch of serial numbers that they can assign to their nodes. Each manufacturer is assigned IDs, and even you, if you want to build your own nodes, has a reserved series of IDs.

This is similar to the way cell phones work, for example. They're all given unique numbers when they're manufactured, so that the companies can keep track of which one is where without ever having to worry about duplicate phones. At the same time, you don't have to worry about that long serial number because it's entirely managed automatically by the cell phone and the system. The phones know how to talk to each other, and that's enough for you.

Automatically-assigned Event Numbers

The main workhorse on a S9.6 LCB is a "token", which represents an "event" being sent from

producer(s) to consumer(s). A token is an event ID number that is unique to that specific event (yes, unique in the whole world). One or more nodes (producers) can send a token in a message, and all the nodes configured with the matching token will listen (consumers). For example, two buttons on either side of a peninsula might set the turnouts in a station to the main line, or a BOD might turn a signal red.

Configuration tools, which may built into the S9.6 nodes or as separate hardware devices or applications, such as JMRI, configure the token numbers to set up these control connections across the layout. Unless they really want to, users don't need to worry about the specific numbers for tokens, because the nodes exchange them as needed to do the setup. Again, think of a cell phone analogy: If you get a call from me on your phone, you can tell the phone to remember my number from now on. You, the phone user, don't have to remember 10 digit numbers since the phone has the technology to do it for you. You just look up somebody's name and press dial.

One of the main differences between S9.6 and other systems is that it uses very big numbers for tokens: 64-bits long. This has some distinct technical advantages: nodes can automatically assign a unique event number whenever one is needed, so the user doesn't need to think one up nor record it. Any node can teach its events to one or many other node(s), whether it or they be consumers or producers, in one teaching session.

Another advantage is we can place such things as fast-clock time, RFID numbers, and other bus's messages right inside S9.6 events, letting nodes respond to these just like any other event, without needing any extra programming to make it happen. This kind of flexibility & capacity for future growth is key to creating a layout control bus that will last for more than 20 years.

Extensible family of protocols

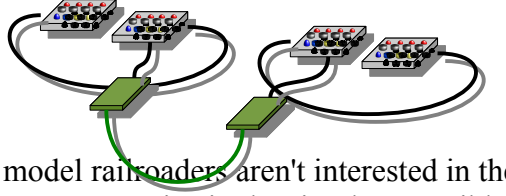
Besides events, S9.6 also uses datagrams and streams to carry larger messages. Datagrams can send up to 72 bytes of information between two nodes in a single operation. Streams can send information from one node to another continuously. These can be used for upgrading node firmware, uploading sound clips, or other uses that need larger amounts of information sent.

Control of size and of traffic

Layouts can grow to the point where there are too many nodes or too much traffic for a single control cable. S9.6 is designed to alleviate both of these problems. The solution is to split the bus into segments, each of which uses a different cable. S9.6 was designed from the ground up run efficiently on a segmented bus. Its use of a large flat ID-space simplifies the problems inherent in this situation, without the need to extend or translate the node or event IDs.

While CAN will likely be enough for small layouts and some medium sized layouts, it's clear that a faster bus will be required for large layouts and in the future. For this reason, S9.6 is also implemented on Ethernet, which can connect multiple CAN segments through intelligent gateways. To help with traffic overload, S9.6 implements automatic analysis, filtering and routing of message traffic such that messages only go to nodes and segments that need them – this is called *interest-based routing* – and reduces segment traffic to mostly local messages only. All of this works without any changes to the basic boards that a beginner first starts with.

Hobbyist involvement



Many model railroaders aren't interested in the technological aspects of a layout control system, and just want to run trains in the simplest possible way. They don't need to worry, because a lot of work has gone into making S9.6 as simple as it possibly can be for the model railroaders using it.

At the same time, there are a few model railroaders who consider working on control system(s) a fun part of the hobby. For them, we published detailed documentation for S9.6 ranging from descriptive examples, sample messages and example applications through formal descriptions and draft standards. Because we believe strongly that S9.6 should work with many different types of control hardware, we've published the code for running OpenLCB in both PIC and AVR microprocessors, including the very popular Arduino series of boards, along with an implementation for use in desktop computer programs such as JMRI. Our website http://www.openlcb.org contains information on how to connect up your prototype board, load our sample programs, and be up and running with your own S9.6 implementation to play with. Like other successful open model railroad collaborations such as JMRI, we're growing a community of developers, users and others who will support each other and move the project forward with great new ideas.

Conclusion:

The OpenLCB team has worked hard to create a set of protocols and methods that is a step forward for modellers of all stripes and layouts of all sizes, while also integrating the goals, mandate, and measures put forth by the NMRanet committee. Special effort was made to make the S9.6 proposal robust but flexible and extendable, while being comprehensive and user-friendly. We hope you'll join us by using it, and by helping us move it forward into the 21st century.

Figures: