Scale Rail Article

Introduction:
The advantages of a bus: reduced wiring, ease of automation, adding a computer. A PD solution would be good, and get everyone onto the same page.

History:
DIDMetaDCC --> MRLCB --> NMRAnet
--> CBUS -->Atbus, -->S9.5, -->S9.6
Thanks to the many contributors: ...

Similarities:
CAN
Producer-consumer

How S9.6 is different:
We have thought about the difficult problems:
how to make it easy for the novice
generalizing the p/c model
simplifying the simple nodes
supporting the very large layouts

Simplicity for the novice, and the expert alike
- no need to learn Ids, channels, etc
- push button programming
- useful things: automating a station's tracks
- automating a yard throat

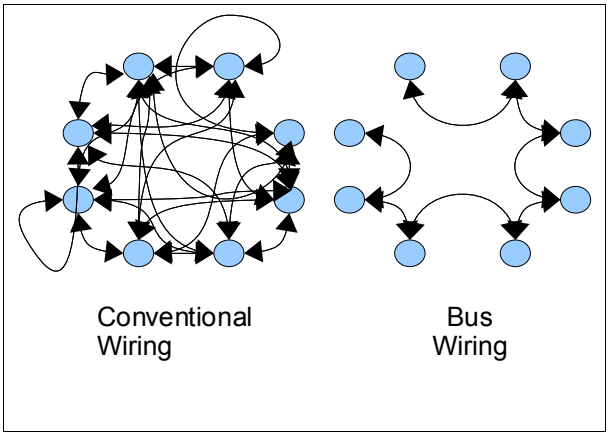
Generalizing the P/C model-
- any node that shares an event# responds to the event
- any node can be taught the event from any other node

Large Ids simplify things:
- allows us to fold other information into the events
- this lets any 'dumb' node respond to the event
  eg: fast clock. n S9.5, this would be a new set of messages, to use them, each node woul dhave to have code added to handle the new set of messages. With S9.6, because of the big-events, the time is folded into the events. A dumb node can respond to the time code, just like any other event. It doesn't need reprogramming.

Support for very large layouts:
- If your bus gets full, either too many nodes, or too much traffic – just split it in two.
- A large layout can be a number of segments joined by a fast bus.
- Traffic is reduced by gateway filtering, messages only go where they are wanted.
- This is called “scaling gracefully”.

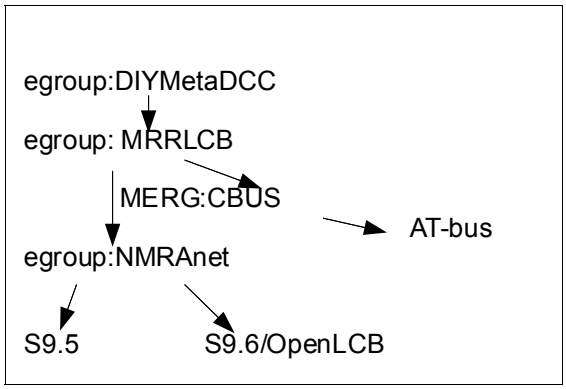
Introduction

Although NMRA has had great success with the standardization of DCC, the rapid advance of electronics has meant that the control of accessories, animation and ... on the layout is much easier. However, the wiring of all these accessories leads to a spider-web under the layout. Unfortunately, one solution that many groups have tried, that of using DCC as an accessory bus, has lead to DCC congestion. Alternate methods include using one of the proprietary CAB or throttle-buses, such as Loconet, Xpressnet ar NCEs Cab-bus. The NMRA decided that a new public domain accessory bus should be developed. [picture of ad hoc wiring vs bus]



History

Numerous people have taken part in a series of internet egroups, some predating the NMRAs interest in the bus. The first group was started by xxx, DIYMetaDIY. The discussion soon moved to MRRLCB, and eventually moved to NMRAnet, when he NMRA got involved. Some progress was made: CAN was recognized as an ideal bus for MRR, haivng been used in automobiles anf truck – robust, error strong. The pc model was introduced to the group by xxx, and was recognize as a powerful method of buoth simplifying and making a bus protocol more powerful. The severe space limitations of Can packets lead to disagreements on how to fit the messages. Two members became eager to build a bus, and developed CBUS at MERG, base on 11-bit headers. Subsequently, another MERG member modified the protocol and develope AT-bus.



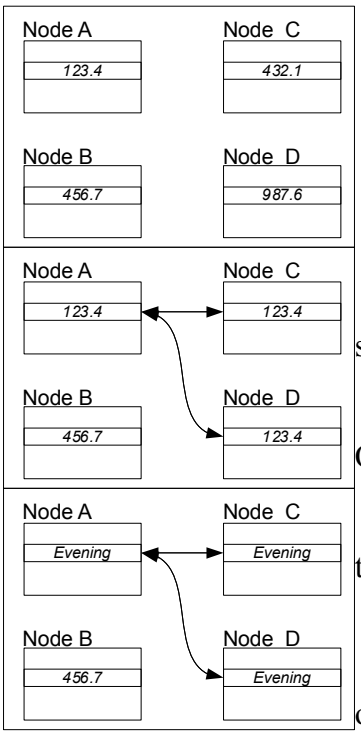
The group continued to discuss the use of CAN, focusing on 29-bit headers. This group eventually split into the S9.5 effort, lead by Don Voss, and the S9.6 effort comprising John Day. David Harris, Bob Jacobsen, Alex Shepherd, and Alan Robinson. Di Voss has done a good job introducing the concepts of an accessory bus, and the concept of p/c, and gives a good description of S9.5. This article will concentrate on the advantages of S9.6.

How S9.6 Differs

The S9.6 group wanted to design a bus standard that added significant function to the existing buses, and one that will be useful into the future, lasting for at least 10-20 years. For this reason the S9.6 design team has spent a lot of time thinking about the hard problems: how to make it simple for the novice while letting it serve very large layouts, and letting it take advantage of future technology while simplifying the implementation of individual nodes.

Easy for the novice

Alice wants to control her three track station. She has goes to her local store and buys one 8-button input node, four turnout-controller nodes, and a power supply. When she gets home, she unpacks the nodes and connects each of the turnout-controllers to her four turnouts, E1, E2, W1, and W2. Then she connects the nodes together with the provided cables. Following the directions, she then uses the blue button on each of the turnout-controllers and ensures the attached turnout changes position.



S9.6 lets you teach nodes to respond to one another by letting you place the same event number into each node. In the upper diagram to the left, the four nodes are represented as boxes containing an event number. However, the nodes have no relationship with each other because they do not share the same event number.

However, by teaching one node's event number to two others, the three nodes can now communicate, and together they form an *event*. This event can represent any conceptual idea, such as 'evening has fallen' – this is shown symbolically in the bottom panel. Node A has taught nodes B and C, and they share the same event number.

One or more nodes can be taught from another node. In S9.6 you can accomplish this be pushing buttons to select the events to be taught on the learning nodes, and then choosing the event number to teach on the teaching node. This method is called the '*blue/gold*' algorithm after the colour of the buttons.

Also, note that in the bottom panel, node B can be taught from any of the other nodes.

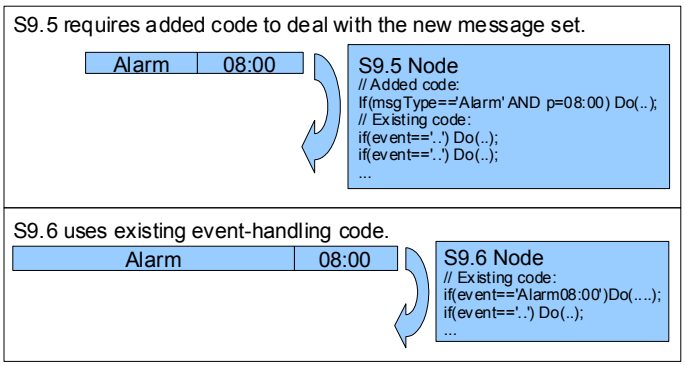
Big Events

Your club does not want to expend any effort pre-allocating numbers, making sure that module configurations don't conflict or anything like that; they just want to plug them together and operate the entire layout from both central and distributed locations. All the modules must still continue to operate without reprogramming or reconfiguration; there's just not enough time for that.

A major difference from previous buses, and the other proposals, is the use of 'big events' - S9.6 uses 48-bit and 64-bit node-ids and event numbers, respectively. This decision was not taken lightly. Big events have many advantages. It simplifies the code of simple nodes, and allows increased flexibility since all nodes can respond to any event. It frees you from remembering node, channel and event numbers since these are pre-assigned when the nodes were made, and each is unique. Since the events are unique and big, your modular club and large layouts do not have to worry about running out of node numbers nor whether two nodes will conflict.

As an example of simplifying nodes while adding functionality, consider adding a fast clock with an alarm function to your layout. The old way would to be to define a new alarm-messages, and to write new code into each node to recognize and process those new messages.

Big-events lets S9.6 to enfold the time-value inside a set of events. Your nodes respond to the alarm-event just like any other event – and no extra code is needed.



This is illustrated in the diagram to the right. The top panel shows a alarm-event and the extra code needed to recognize and process it. The bottom panel sows S9.6's approach, the time is enclosed in a regular event, and your node responds to it just like any other event.

Unique Ids equals no Ids

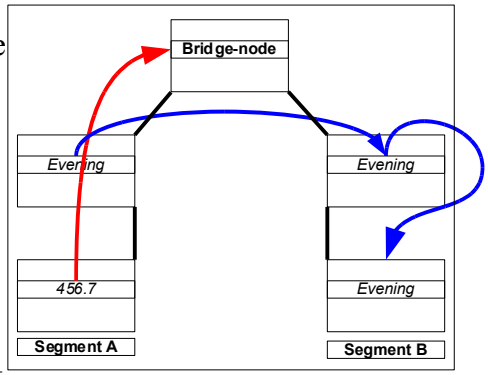
Since S9.6 has big-events, it can afford to assign one to each node at its manufacture. This means that you do not need to set a node's id number when you get it, it already has one and its different from every other node in the world. This means that you don't have to worry about any two nodes conflicting. You can set-up your node, take it to your friend's house, and not worry that it will conflict with his nodes – S9.6 guarantees it.

If you have a large layouts, you don't need to keep detailed lists of node numbers -- a computer can easily query the layout to retrieve all the node numbers. In addition, you can refer to the nodes by name.

Running out of things

Hugh realizes he has a problem – he's maxed out the number of nodes. He talks to the local guru, and the guru says – no problem, just divide your CAN into two parts and join the parts with a CAN-to-CAN bridge. Hugh is doubtful, it can't be that simple, won't he need to change node ids and change his event programming? The guru reassures him – it is that simple – no need to change anything, the nodes will still see all the other nodes events, just as before. Hugh takes him at his word, picks up a bridge and, after five minutes deciding where to break the CAN bus, inserts the bridge and is running trains again.

One of the big problems with buses is that you eventually run out of things: id-numbers, bus capacity and bandwidth. Big events obviously takes care of the first, but S9.6 has also addresses the last two. If you have too many nodes or too much traffic, then you can break your layout into two bus-segments, and join them with a bridge-node, as diagrammed to the right.



This solves the problem of too many nodes on a segment, but not the traffic. S9.6 solves this by by running over fast transports, such as Ethernet. But this doesn't solve the problem if all the messages go everywhere. Partial solutions involve non-linear address spaces and limiting traffic on a message by message basis. S9.6 automatically limits traffic by only sending messages to those nodes that want them. This is called 'interested based filtering', and is illustrated to the right. The blue event is passed by the bridge-node, since two of the nodes on segment B are *interested* in the evemt. However, the red event has been blocked at the bridge, because no node requires it

on segment B. In addition, S9.6 messages are completely routable, and you can use cheap Ethernet routers to do the work. This is implemented now, and your layout can take up acres as space, and still run a single unified bus.

Other good works for Big layouts.

Hugh and John are in charge of setting up the modular meet. They use the configuration program to build a schematic diagram of the entire layout. They then drill down to each interface between layout segments and connect the edge nodes of each segment together. They quite quickly repeat the exercise on each adjacent pair of segments, with the help of the leads of the pairs of segments. Since every node has a unique ID and unique Events, they are not worried about conflicts or having to change ranges of events on any modules – reprogramming hundreds of nodes would not be inviting.

One of the major problems with large modular meets is the time taken to set up. S9.6 gives the meet organizers tools to shorten and ease this effort. Auto-configuration of modules and an automatic over-view of the layout is generated. The meet organizers can drill-down into each bus segment and get reports of all the nodes and their messages, so debugging and tweaking the layout is easy.