OpenLCB - *A new way of controlling model railroads*

Executive Summary

This document provides an overview of the OpenLCB™ layout control bus. OpenLCB is intended to control the operation of accessories and trains on a model layout. Further details are available on the web site at http://openlcb.org/ and discussion group at http://groups.yahoo.com/group/OpenLCB.

Guiding principles and design philosophy for OpenLCB:

- Robust yet cost effective, easy to use, plug-n-play, but allows analysis and diagnostics
- Suitable from novice to aggregate and museum layouts.
- Top-down design tested with bottom-up implementations.
- Uniform and regular structure for use across multiple networking technologies.
- Maintains manufacturer's choice of implementation, and eases further development.
- Uses existing standardized and commodity hardware/software
- No requirement for a PC or dedicated tools for simple layouts, but the ability to use these as the layout expands.
- Designed for the future and not focused solely on current technology choices.
- Standards conformance validation requires minimal effort.

OpenLCB defines a set of common messages to provide consumer-producer events, datagram messages, and streams that are media and transport agnostic. IOpenLCB specifies how these are to be implemented on a variety of transports, including but not exclusive to: CAN, Ethernet, and wireless. Using these, OpenLCB specifies facilities for layout control, node configuration, gateways to existing systems, and future expansions.

OpenLCB is designed to use one or more net-segments which may involve one or more types of transport media. OpenLCB allows the ease of use to implement a novice-layout involving only one segment, to very large layouts involving many segments. This is accomplished by defining both simple messaging for use on small layouts, but also more sophisticated mechanisms such as routing, which allow very large layouts to function.

Introduction

This document provides an overview of the OpenLCB layout control bus that is intended to control the operation of accessories and trains on a model layout. This document is intended to give the reader a good appreciation of the main features and benefits of the OpenLCB design. Please note that OpenLCB is still in the design and prototyping phase. To assist in that, we describe the OpenLCB design in sufficient detail for others to review and comment on the work to date. This document should be considered a draft.

Guiding Principles and Design Philosophy

While this document is meant to be an overview or summary of OpenLCB it still contains a lot of higher level details. So here is a list of the guiding principles and design philosophy we have used:

- The OpenLCB must be robust and yet be cost effective, easy to use, plug-n-play, and provide means to attach to a network monitoring tool to analyze behavior and diagnose faults.
- A single system suitable for novices, yet works for large modular meets and museum layouts.
- The top-down design with bottom-up implementation reality checks.
- Elegant structure and uniformity that can be applied across multiple networking technologies.
- Openness to allow manufacturers choice of implementation, and allow further development. No fixed media, microcomputers, etc.
- Use of existing standardized and commodity hardware/software: Consumer/Producer; CAN, Wireless, Ethernet TCP/IP, routing, etc.
- Configuration and running must not require a PC or complicated tools for simple layouts. However as the layout expands and more complicated configuration choices are made, the system should be able to simply expand, without requiring the user to reconfigure existing devices or even start over.
- Design for the future and not be focused solely on current technology choices. A good OpenLCB design should last many years despite changes in networking technologies over that time. The design needs to be adaptable to exploit new technologies as they develop.
- Validation of standards conformance must require minimal effort, yet be comprehensive.

When considering the choices commonly available for initial implementation we cannot ignore the CAN 2.0 (ISO 11989-*) networking technology from Bosch, as it is a very robust and accessible technology in 8-bit microprocessors. OpenLCB defines how to use CAN as an effective low-cost interconnection. To run a layout in the manner similarly to what can be done now using DCC vendor equipment, CAN is very adequate and can handle those duties fine. However it has very limited frame sizes (only 11 bytes), bandwidth, and maximum number of nodes, so it can't do everything needed by advanced model railroaders. We have therefore deliberately not limited ourselves to what can be done on CAN networks because, in the expected life of OpenLCB, other networking technology choices

(particularly wireless) will become commodities, and therefore good candidates for use by model railroaders via OpenLCB.

We have not tried to overly minimize the sizes of the data items in the OpenLCB messages. We have deliberately made Node Ids (48-bits) and Event Ids (64-bits) outrageously large. This choice was to minimize the NMRA management overhead required to manage the allocation of globally unique node addresses and also provide a way to then generate globally unique Event Ids. This greatly simplifies the way modular layout clubs configure their systems, as modules can be arranged in any combination, and, with purpose-built configuration tools, even that can be substantially automated.

While these choices do have a system resource impact, our design pushes the bulk of the additional resource requirements into the gateway nodes as they are likely to be few in number; many layouts won't even require one. Gateways will likely to be PC-class devices, easily containing sufficient resources. OpenLCB design decisions keep the resource impact on small CAN based nodes minimal.

What is OpenLCB?

OpenLCB defines a family of inter-operating protocols for controlling model railroads. It is structured around some common core concepts and separate implementations. The OpenLCB will be capable of supporting Accessory Control, Advanced Locomotive Control and even Multimedia and Rich Text features on the higher bandwidth networks. This protocol proposal will focus on OpenLCB Accessory Control with emphasis on CAN 2.0B, but allowance has to be made for simple Locomotive Control and implementation of the OpenLCB protocol on other network technologies like Ethernet, WiFi and other wireless networking technologies, and the global Internet.

This proposal is based on a series of use-cases *(see the Use Cases section)* that start with a novice modeler connecting up two boards and extend to huge modular layouts put together by several clubs. To achieve this range, and to allow for future technologies, OpenLCB is defined so that it can communicate via multiple media. These include CAN, Ethernet and WiFi. In addition, OpenLCB software running in a computer provide valid, albeit virtual, OpenLCB nodes.

Key Features of the OpenLCB Protocol Proposal

The key features of this proposal are:

- The basic communicating unit is called a "Node". A node might be a single board, or a single program in a PC, or any other addressable item. A node may control any number of actual devices on a layout. Nodes exchange messages to control the layout.
- At it's basic level, OpenLCB defines a transport layer for reliable transport of data between multiple nodes. This transport level also has provision for interconnection of independent OpenLCB segments and filtering of messages to reduce traffic.
- Within that transport layer, OpenLCB defines a global set of common message types, and specifies how the message types are to be used for standard and optional interactions.
- The structure and format of the common messages is separately specified for each of several wire protocols. The initial wire protocols are CAN and TCP/IP point-to-point; basic structure for additional ones has been provided. OpenLCB wire protocols may define additional

interactions and messages for specific transport-level uses, e.g. for housekeeping on the local connection.

- The OpenLCB as a whole is comprised of one or more segments interconnected with repeater, bridge or gateway devices that ensure all nodes are able to communicate.
- Every node has a 48-bit "Node ID number" (NID) which is assigned to assure that it is globally unique. NID ranges are assigned to distributed to manufactures, groups and individuals by a method described in detail in another document. Each manufacturer that has a NMRA DCC Manufacturer ID will have a 24-bit address space allocated for their exclusive use. In most cases, users would buy nodes with IDs preassigned by the manufacturer. However, some users will want IDs for locally developed nodes, and therefore, large clubs or societies like the NMRA, MERG, FREMO, NTRAK, oNeTRAK etc. will each be issued a large number range. The group can then subdivide, possibly by a membership number. Each member of these groups will be automatically allocated NIDs for their own use for DIY-built nodes. In addition, groups or individuals may also request a additional ranges of NIDs, if these are required.
- The OpenLCB is a flat structure, where all nodes are equally accessible and addressable. There is no hierarchy of nodes, node aggregation, tree structures, or segmentation that necessarily filters OpenLCB traffic.
- Athough all OpenLCB nodes have the potential to see all OpenLCB messages, the protocol uses interest-based routing. This is an effective way to reduce network traffic in large OpenLCBs with many interconnected network segments. OpenLCB gateway nodes dynamically learn which nodes on which segments are interested in specific events, and only send those events to those segments.
- The Source-node's ID is included in every OpenLCB message, and allows each message to be traced back to its originator node.
- Above the basic transport level, OpenLCB defines a small number of basic types of communication:
    - Events - Fixed length identifiers that can propagate from multiple source nodes to multiple destination nodes to provide Producer/Consumer communications.
    - Datagrams– Short messages that are transported to a specific destination node. These are used for configuration and other protocols that require messages to be reliably delivered to specific nodes. Use will likely include throttle and train control messages,
    - Streams - An efficient way to move a large number of bytes from one node to another.

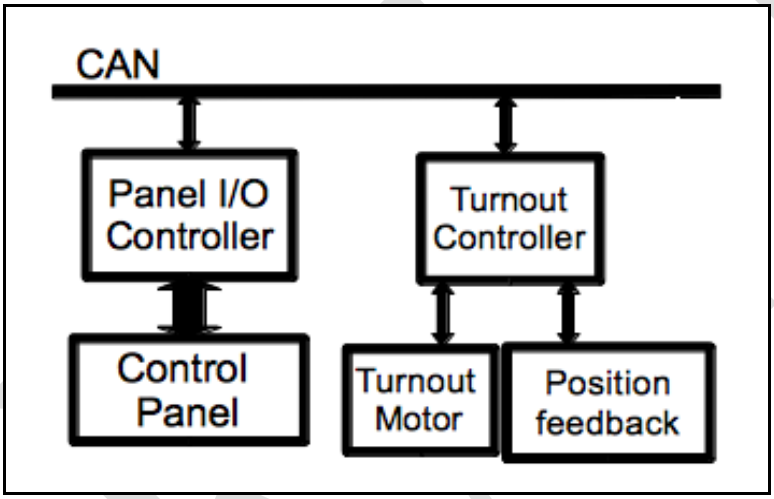
  By limiting communications to a small number of orthogonal forms, OpenLCB simplifies communications code in small nodes, yet still allows very powerful functions to be built with these communications methods. This is a similar method to how the World Wide Web and other valuable communications tools were built on top of the simple TCP/IP and UDP/IP protocols.
- Above the basic communications types, OpenLCB provides protocols for:
    - Layout control operations

- Node configuration
- Function gateways to other layout control equipment, including DCC
- Error logging
- Locomotive control (although not currently part of the protocol specification)

Amble room is left in the specification for additional protocols to be added for new functionality, even model railroad applications that haven't been thought of yet.

OpenLCB Network Topology

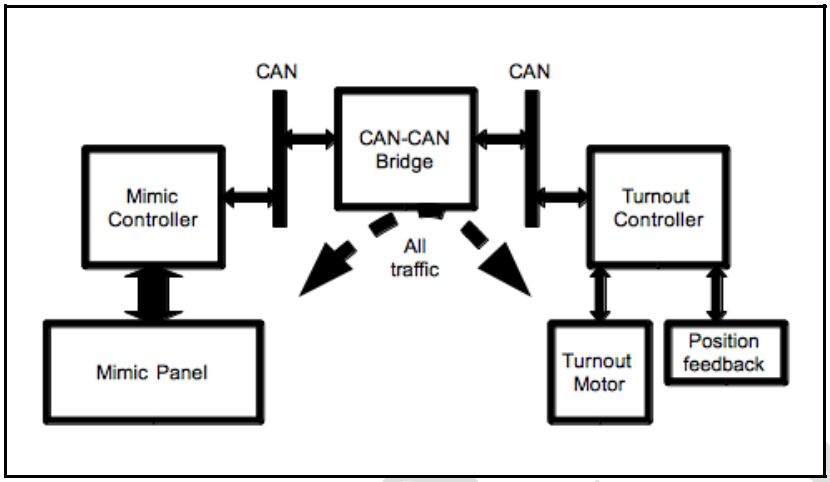
A layout with a CAN-based OpenLCB will require at least one OpenLCB network segment with two Nodes as in the example shown in the diagram below.



The diagram shows CAN as the transport, but a layout could also Wireless, Ethernet ,WiFi or be a combination of several technologies in the future. A layout may have, for example, a lengthy CAN based OpenLCB segment with many nodes, running around the entire layout and have a comparatively short Ethernet segment that provides a convenient connection to a single PC. It is likely that entry level OpenLCB compatible DCC Command Stations will just have a CAN OpenLCB interface and perhaps a USB PC interface. Medium level DCC Command Stations might add Ethernet and/or Wireless interfaces, whereas top end DCC Command Stations may also add WiFi. It is also likely that manufacturers will provide various interface devices (repeaters, bridges or gateways) to link each of the OpenLCB segment types like CAN to Ethernet and/or CAN to USB (for PC access), and CAN to Wireless and/or Ethernet to Wireless (for wireless throttles). WiFi interfaces are likely to use consumer grade Ethernet to WiFi Access Points but some markets may opt for a WiFi interface in the DCC Command Station. This will allow use of Wifi enabled cellular phones to access the OpenLCB directly without needing a PC.

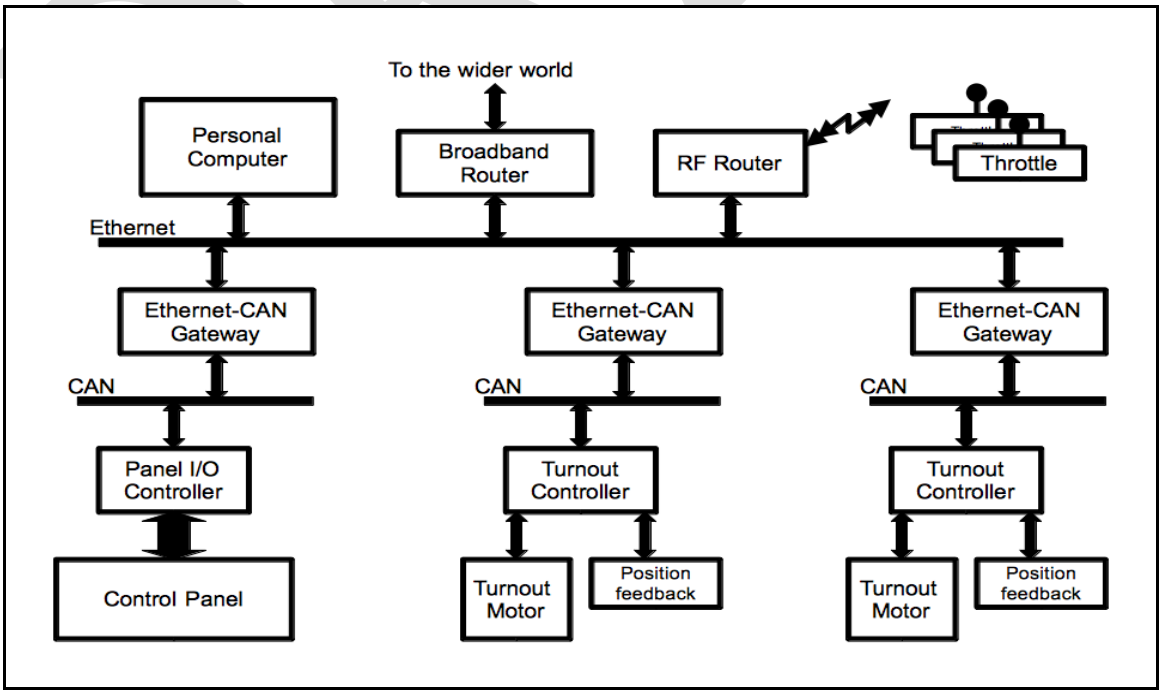
Each segment of a OpenLCB is, by default, transparently connected to all other segments. A single CAN cable would be a OpenLCB segment, as would two CAN cables connected by a repeater or bridge. An Ethernet link or TCP/IP network connection would also form an individual segment.

Two or more segments can be connected by repeater or bridge devices as shown in the diagram below.



Simple repeater or bridge devices do not filter any messages and all messages are forwarded to all other segments. There is a limit to the number of these repeaters before bus latency and traffic become limiting factors. More intelligent interconnection devices, called gateways, can reduce required bandwidth by automatically filtering messages that are not needed on other segments. OpenLCB provides the information that the gateways need to configure their own operations, with no knowledge required from the user.

On a more complex layout, a CAN segment with a number of nodes can be connected via a CAN-USB, or CAN-Ethernet bridge, to a PC running several separate control programs. For a modular layout, CAN segments consisting of a series of modules can be connected to an Ethernet backbone, which in turn is connected to other CAN segments made from other collections of modules. Several PCs could then be connected via the Ethernet, and interact with all the modules via the individual CAN segments.



The large Node IDs and Event ID of OpenLCB enable layouts to grow incrementally to as large as desired without ever having to reconfigure nodes due to addressing conflicts. Two modular clubs can interconnect their entire operating layouts and only have to configure the small number of messages that need to cross between them.

Producer/Consumer Events

The OpenLCB makes use of stateless messages called Events, which are part of a model formally called "The Consumer-Producer Model". This is a relatively new and powerful concept that is well established in the industrial control industry. In it a producer-node reacts to a change in its environment by producing (sending) an Event-message. Similarly, a consumer-node consumes (receives and processes) this message and causes some Action to happen. For example, depressing a push-button would cause the node to which it is connected to produce an Event, and, on receiving this message, a consumer-node would cause its attached turnout to move to a new position.

Events are used to allow multi-node to multi-node interactions. One or more nodes can produce any particular Event, and none or more nodes can consume that Event.

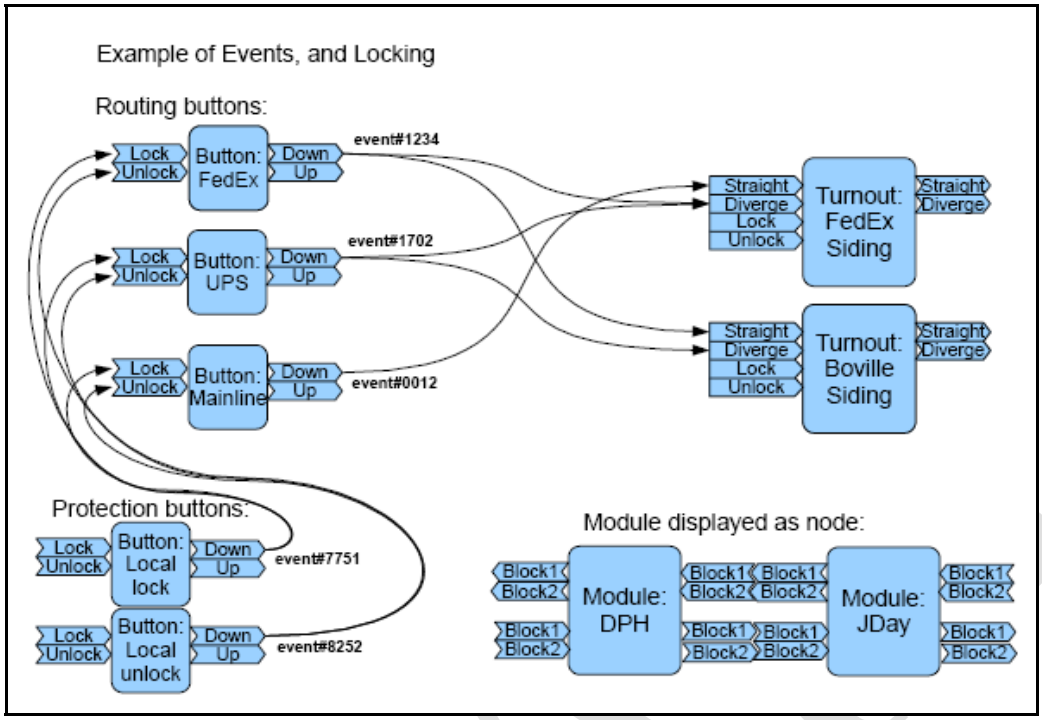
The Producer-Consumer model has some specific details that need to be stressed[1]. Events indicate only that a change has occurred, but do not carry state information --- no information as to what changed can be inferred from the Event. A user could have buttons in several locations that issue a "Set yard throat to track 4" event, and the throat turnouts respond to that event regardless of which location issued it.

Similarly, the eventual Action(s) cannot always be inferred from the Event. In the yard throat example, some turnouts will move one way, others will move the other, and some won't move at all. The Event represents an abstract concept, rather than carrying actual specification of what occurred or has to change. For example, an Event might represent "Night-time running", and it could be produced by one or more of: a push-button; a timer; or a program in a computer, while it might be consumed by nodes that control: street-lights; station-displays or even automated train-routers.

Although it may take a little getting used to, the abstract nature of Events allows them to be very flexible and powerful, as they can link any consumer to any producer. This means that users can pretty much do what they like and not be constrained by the conventional control expectations like controlling a turnout position only from a toggle switch. For instance you could control a turnout position from the time of day clock that produces different Events at different times, and also use those same events to control room and layout lighting. At the same time, Events are still usable for simple commands commonly used on model railroads, such as "Turn on output 3".

[1]More information on using the Producer-Consumer model for model railroad applications is available in a separate note.

The diagram below shows how Events from various push buttons and turnout controllers are able to be configured to Produce and Consume different Events to effect a route selector complete with route and button locking.



Node-Resident Configuration Definitions

All OpenLCB nodes will contain a compressed document in standard form that contains all the relevant information needed by an external configuration tool to know how to configure the node. Configuration tools can use that information to provide some form of suitable Graphical User Interface to allow the user to easily and intuitively configure all aspects of the node's capabilities. An important design choice was to embed this information into each node so that the system has all it needs to configure the node without having to source some file external to the OpenLCB from the manufacturer or some other on-line repository via the Internet or a CD/DVD etc. While configuration tools are likely to be programs running on a PC, they could also be a hand-held device like mobile phone or PDA or even a custom built configuration tool unit. OpenLCB provides the structure for all of these to interoperate.

CAN Node Id Alias Self Assignment

Because of the small CAN frame sizes it is not possible to fit a 48-bit Source Node Id (SID) into an 11-byte Extended CAN 2.0 frame and have much space left for anything else. So to make OpenLCB work on CAN we have used a shortened form of the NID called an Node Id Alias or NIDa. This is a 12bit number that is dynamically assigned at system start-up by each node using its NID as a seed to create a proposed 12-bit Node Id Alias. The node then probes the CAN bus with a sequence of CAN frames to ensure that no other node is using the same Node Id Alias. In the common case of a collision, the involved nodes back off and attempt the algorithm again with new Node Id Alias values. Because nodes have unique Node Id values, this algorithm is guaranteed to converge to non-conflicting aliases even when very large networks are simultaneously powered-up. This property has been confirmed by analysis, simulation and prototyping.

Use Cases

The table below describes the Use Cases[2] that were and are being considered for the OpenLCB design.

Entry Level User	A model railroader wants to learn about OpenLCB. He buys two inexpensive boards at his local hobby shop. He connects them on his layout to two pushbuttons and a turnout motor, does a simple configuration, and is able to control the turnout with the pushbutton. No other boards or tools are needed.
Expanding A Small Layout	A user has four OpenLCB boards working on his layout. He buys one more input board to provide extra pushbuttons to operate his yard ladder from the other end of the layout. He's able to configure the new board and put it into operation. He doesn't have to reconfigure any of the existing boards, nor does he need records of his configuration nor a configuration tool.
Mid-size Layout Problem	A user has twelve OpenLCB boards operating. Something changes, and the system is no longer reliable. Certain operations no longer work, and others only work sometimes. The user attempts some debugging steps using features of the existing hardware, but is unable to locate the problem, so buys/borrows a diagnostic device. The device is able to track all traffic on the network, interrogate each existing board, and in general indicate where the problem lies.
Large Layout Upgrade	A model railroad has 40 OpenLCB boards controlling a large layout. To upgrade a major yard, the user wants to replace 12 boards with new ones that offer new (non-OpenLCB) features. He captures the existing setup, replaces old boards with new ones, configures the equipment and is back in operation. No reconfiguration of non-replaced boards is required.
Distant Control Panel	A model railroad has turnouts and signals controlled via OpenLCB. The owner wants to put a physical control panel in his house, a separate building, and operate it via OpenLCB. He adds a gateway board to his existing OpenLCB, which communicates over his existing home wired or wireless network with another gateway board in the other building. That second gateway drives a small CAN network attached to the nodes that drive the control panel.
Large Layout Expands	A large model using CAN, but not any attached computers, grows enough that it needs to have another CAN segment. A new segment is added, connected by a bridge to the first. It is not necessary to reconfiguring any the existing nodes, nor is it necessary to check for address or event conflicts in the new hardware.
Connect Multiple Programs	The user wants to run multiple programs from multiple vendors in his home computer that simultaneously connect to the layout OpenLCB and control/monitor it. The computer is connected to the OpenLCB via a USB or Ethernet connection.
Remote	A model railroad has turnouts and signals controlled via OpenLCB. OpenLCB is used

[2]See also the separate note on use cases.

Dispatcher	to install a CTC panel at a distant location. The panel could be either a physical panel or on a computer screen.
Modular Layouts	A modular club has fifty modules, each of which as a CAN OpenLCB with three or four nodes controlling the module. These modules are separately built, with no central administration. They are brought to a central location for a meet, where they are all bolted together in some pre-planned orientation. Some OpenLCBs are connected together electrically via CAN jumper cables, and others are connected to a central Ethernet. The resulting single OpenLCB instance can operate the entire layout from both central and distributed locations. No conflicts between modules and/or nodes need to be resolved.
Remote Diagnostics	A club layout is operated via OpenLCB. There's something not quite right about the signaling, and one of the members wants to check the operation of the signals. He makes a remote connection to the layout and checks the status and configuration of the hardware from his home computer.
Aggregation of Modular Clubs	Dozens of clubs put together dozens of OpenLCB segments and hundreds of modules that have been separately configured. A large FREMO meet would be an example. Collision avoidance: EventIds, NodeIds that are already configured into the modules should already be unique. Conventions and/or tools for connecting events across the boundaries are available so that e.g. signaling systems can work with adjacent modules they've never met before. Automated monitoring tools can see traffic on the entire network.

Typical Equipment

The table below describes the typical equipment that is likely to exist on a OpenLCB equiped layout.

Simple Board	Low cost, low engineering complexity is important.
Multiple I/O Board	Higher cost/complexity board that can do multiple input and output functions.
Simple Diagnostic Tool	A simple diagnostic tool that can plug into a CAN segment and provide limited information. Not based on a computer.
Computer-CAN Interface	Inexpensive module to allow any type of home computer to attach to a CAN segment and interact.
Ethernet-CAN Interface	Module to allow a CAN segment on e.g. a module to be connected to an Ethernet backbone without requiring a local computer.
CAN bridge	Low cost module to connect two CAN segments for greater length or more attached

	nodes.

Glossary of Terms used in OpenLCB

Common Terms

Alias	Short form of a Node ID number which can be mapped back and forth to the full number. Often used as "NID Alias", "DID Alias" or "SID Alias"", which are then written NIDa, DIDa and SIDa.
Board	Not really something that occurs in OpenLCB itself, we need to talk about how the common term "board" maps onto OpenLCB. E.g. A node (board) may connect to several things (devices) on the layout.
Bridge	Connects two OpenLCB segments with minimal changes to the content of the messages. For example, a bridge between two CAN segments would allow more nodes to be attach to the combined segments as if they were one; a bridge between a CAN segment and Ethernet segment would transform message format, but transfer every message.
Device	Not really something that occurs in OpenLCB itself, we need to talk about how the common term "device" maps onto OpenLCB. E.g. A node (board) may connect to several things (devices) on the layout.
DID	Destination Node ID - Node ID of the node to which a specific message is addressed.
Event	OpenLCB allows nodes to notify each other when specific "events" occur on the layout. These in turn can cause nodes to take particular actions. Events are not necessarily attached to a producer ('Button 2 pressed') or attached to a consumer ('Turn off light 4'), but rather to an overall state change ('Set for nighttime operation'). This is called a "Producer/Consumer model".
Event ID (EID)	A 64-bit number that identifies a specific Event. OpenLCB Event IDs must be globally unique. Events are not associated with any particular node. It may be convenient to use e.g. Node ID as a way of numbering them uniquely, but Node IDs and Event IDs are not related.
Gateway	Connects two segments of the OpenLCB, optionally suppressing messages that are known to be not relevant to the far segment. If needed, so translation from one message form to another.
Installation	An OpenLCB installation is the complete set of OpenLCB hardware, nodes, etc, that can be reached from any one of them.
Node	Unit of addressability for the network. Every OpenLCB interaction originates in a node.

	Every board that connects to OpenLCB is at least one node. For example, a simple turnout controller board is one node, while a PC with multiple programs running may contain several nodes.
Node ID (NID)	A 48-bit number identifying a specific node. OpenLCB node IDs must be globally unique, so they form a one-to-one mapping to the nodes themselves.
Node ID Alias (NIDa)	A 12-bit number identifying a specific node on a CAN network. There is a one-to-one mapping from a NID to a NIDa.
Message	The basic unit of OpenLCB communication.
MTI	Message Type Indicator - identifies a common message type. (This term is still under discussion)
Producer / Consumer Event Model	The mechanism whereby Producer or sending Nodes notify interested Consumer or listener Nodes that particular layout events, state changes or control actions have occurred by generating an Event. The Event is simply a OpenLCB Message that contains an Event Id or EID, which is simply a unique number that signals the defined event has occurred.
Repeater	Connects two segments of the same type at the physical level, without transformation of message format
Router	Rare in OpenLCB, these transform one address space into another. For example, these change EventIDs on one side of the Router into another set of EventIDs on the other side.
SID	Source Node ID - Node ID of the node which originated a specific message.
Segment	Subset of an overall OpenLCB installation which is reached via one or more translators and/or gateways. A segment typically uses a specific wire protocol.
Stream	Streams are a method of moving a large number of bytes between two nodes.
Wire Protocol	Version of the OpenLCB common messages, interactions, etc adapted to a particular transport mechanism. Examples are the wire protocols for CAN bus segments and TCP/IP links.

Wire Protocol Terms

Packet	A Packet is a single transmission on a OpenLCB segment.

CAN Wire Protocol Terms

Frame	A Frame is packet as it is defined on the CAN bus. It consists of a 11 or 29 bit CAN header and zero through 8 bytes of data.

Table of Contents