OpenLCB Technical Introduction

OpenLCB has been designed as a layered system of communications between "nodes". It provides unified communication and operation from individual low-cost boards through high-end computers.

The bottom layer provide a standard mechanism for sending and receiving messages via reliable transport media[1]. Adaptations are defined for specific transport media, initially Controller Area Network (CAN)[2], TCP/IP[3], in addition to the native format in computer memory. This layer defines mechanisms for declaring that a node is operational, routing and gatewaying messages, enquiring about existing nodes, and parsing the basic messages[4]. The protocol is based on globally unique 6-byte Node IDs. It defines multiple methods for ensuring uniqueness[5] of these Node IDs.

At the next level, OpenLCB defines standard methods for communicating between nodes:

- Datagrams[6] can efficiently move up to 70 bytes from one specific node to another. Buffer management for small nodes is explicitly considered; datagrams are retransmitted when the receiving node is temporarily short of resources.
- Events[7] are short (8-byte) messages that propagate to all nodes that have registered to received them. Typically these are used in Producer/Consumer communication for layout control, but they can also be used for other purposes. Uniqueness of event ID codes is ensured via several mechanisms that can be implemented in even the smallest nodes. The protocol provides messages to enquire about which events a given node can transmit and wishes to receive, which allows automated routing and gatewaying of these messages.
- Streams[8] are an efficient method for moving large amounts of data from one specific node to another. Multiple streams can be exchanged at the same time, both between different nodes and between a specific pair of nodes (except on CAN segments). Buffer management at the end nodes and at any nodes along the path is negotiated through the protocol. The protocol specifies methods for negotiating stream IDs so that independent implementations can be used in the end points.

OpenLCB then builds protocols for common functions on top of these basic communication methods.

- Producer/Consumer Layout Control[9]
- Simple button-based configuration[10]
- Remote configuration[11]
- Fast clock operation and coordination[12]
- Information display[13]

---

1 http://openlcb.org/trunk/documents/
2 http://openlcb.org/trunk/documents/can/index.html
3 http://openlcb.org/trunk/documents/binary/index.html
4
5
6 http://openlcb.org/trunk/documents/
7 http://openlcb.org/trunk/documents/
8 http://openlcb.org/trunk/documents/
9
10
11
12
13

- Node Identification and Confirmation[14]
- Use with external systems, e.g. LocoNet[15], DCC[16]
- Central error logging[17]

(Many of these are works in progress, and there is much to do if people want to get involved) Additional capabilities can be built here, or on top of these.

OpenLCB is being developed via a non-proprietary, collaborative process centered on an openly available documentation[18] and code store[19]. Development proceeds in parallel along several dimensions:

- Documents[20] - Doc package: discusses in detail, not just with tables of bits
- Prototypes – Prototype code[21] is being developed in Java and C/C++. The Java code works both stand-alone and within JMRI[22]. The C/C++ code works both standalone and within the Arduino[23]. This codebase allows easy development of sample implementations, for example the OpenLCB servo driver[24], which was less than 30 lines of code.
- Test code – Test code is being developed in Python[25], Java[26] and C/C++[27] to exercise the OpenLCB design in more detail than can be done with prototypes. This allows simulation of large and small layouts, test of error conditions, and the exploration of special cases needed to create a reliable standard.

The group works via Skype and the OpenLCB Yahoo site, with file collaboration via SVN. All contributions are made available subject to the OpenLCB license terms.[28]

More than just creating specific software and hardware, the goal of the OpenLCB group is to create a ecosystem within which hobbyists and manufacturers can both create great stuff for model railroads. We understand that people will want to use OpenLCB with many types of software and hardware, so we work to ensure that OpenLCB is not implementation specific. OpenLCB has built in extensive tools for discovering nodes and their properties, so that separate implementations can learn about each other and work together.[29] For example, OpenLCB defines how one node can learn about the configuration options available in another without human intervention.[30] By carefully considering how implementations may differ, yet still interoperate reliably, OpenLCB is aiming at a truly general standard.

14
15
16
17
18
19
20 http://openlcb.org/trunk/documents/index.html
21
22
23
24
25
26
27
28
29
30