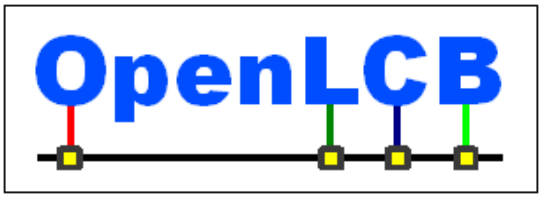

OpenLCB Technical Note	
Remote Button Protocol	
Aug 4, 2012	Preliminary

1 Introduction

(This is an early draft note, a combination of both Standard and Technical Note; the formal separation will come later)

Some OpenLCB nodes can be programmed (configured) via buttons and LEDs on the boards. But sometimes, the board is in an inconvenient location, so you'd like to program it remotely. This note is about a standard way to get access to those buttons and LEDs to work from another location. Note that the configuration protocol (link) can also be used for remote configuration, and that provides the top-end capabilities in this area; this is meant to provide botten-end capabilities to compliment that.

Intended to allow a simple, low cost node to remotely mirror the button/LED UI of another simple, low cost node, thus providing “stoop-less” button programming. Should work well with teach/learn protocol.

Must be inexpensive to implement, requiring absolutely minimal code and data space. Should be orthogonal and simple to understand. Intended to fill an ecological niche well below the full configuration protocol.

Should work with Identify and Indicate to locate a board without having to know the node ID number. Though Identify involves pushing a button on the node, it's still only one button. Indicate can be used to scan and select too.

Buttons and LEDs are entirely under the control of each end of the conversation, so it makes sense to send them all every time. Variable length messages are used to do that.

There's no need for a lot of context information to be sent: Just send the bits as a vector.

Should this be a session-based protocol or not? E.g. should there there be a start and end (so that the far end knows to set updates), or should nodes accept buttons commands and requests for the current LED state? Requesting state requires a lot of bandwidth, because lights blink, so we use “report on” and “report off”.

Controls just buttons, not inputs in general. Matters because the physical inputs might be inputing while the remote is working.

Intended for indicators, e.g. LEDs, that change at visible rates of up to a few Hz, not at 100's of Hertz.

Is there a need for multiple nodes accessing a single one target at the same time? E.g., is there a need for reservation? c.f. Configuration reservation protocol. (should that be made more light-weight and pulled out?)

It would be nice to have a way to enquire about “labels” for buttons and LEDs, so e.g. an iPhone app can make a nicer display of the node. Position info too, so you can show where they are on the node board? Getting a little heavy-weight here, but neither end really needs to use/provide those.

2 Annotations to the Standard

2.1 Introduction

Note that this section of the Standard is informative, not normative.

2.2 Intended Use

Note that this section of the Standard is informative, not normative.

2.3 Reference and Context

3 Messages

Button protocol message (MTI to be detemined; Addressed, simple, medium priority)

Content (after flags and address on CAN):

Control byte:

LSB: 1 – (start/continue) button reports

2nd bit: 1 – contains settings (0 is report or no content, depending on direction)

3rd bit: 1 – contains buttons (0 is contains lights)

rest reserved

0 – N data bytes

Button or light content, as indicated.

Button: 1 indicates pressed

0x80 1st byte – Blue

0x40 1st byte – Gold

(following) buttons as documented by manufacturer, numbered 1 – N

Light: 1 indicates lite

0x80 1st byte – Blue

0x40 1st byte – Gold

(following) lights as documented by manufacturer, numbered 1 – N

4 Protocol

Remote node sends message with “start” set to controlled node.

Remote node can then send “button settings” messages, and provider node will send “light indicator” messages.

Remote note sends message with “start/continue” reset to end the interaction.

5 Discussion

To make this automatic, it would be nice to have a way to enquire about how many buttons there are, how many lights, and perhaps even what they mean. But the goal here is to be as lightweight as possible, for the tiniest of nodes. Even building in a fixed status response takes code space that we don't want to require for tiny nodes.

Does there need to be a way to say “inputs restored to default” when you're done? E.g. “pay attention to what the buttons are actually doing”? Could be piggy-backed on turning off the reporting bit.

Does there need to be a way to say “outputs restored to default” when you're done? Outputs are really under control of the node itself, should be built into the definitions of what the buttons are doing.

There's no provision to write just one button or report one lamp. Not needed, as it's so efficient to provide lots, and we don't want to require the code to do it.

Table of Contents

