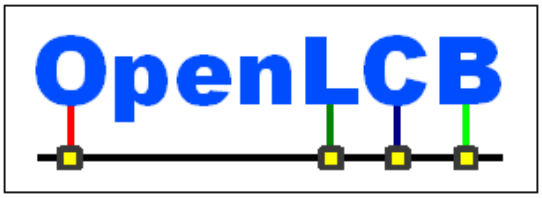

OpenLCB Technical Note	
OpenLCB-CAN Message Network	
10/08/10	Draft

1 Introduction

This explanatory note contains informative discussion and background for the corresponding "OpenLCB CAN Message Network Specification". This explanation is not normative in any way.

2 Annotations to the Specification

This section provides background information on corresponding sections of the Specification document. It's expected that two documents will be read together.

2.1 Introduction

2.2 References and Context

Below this is just a collection of pieces from other docs right now.

The priority bit in the CAN frame is separate from the static priority field in the MTI format specification.

The specific MTI values are being allocated in a separate worksheet (PDF version). In general, the MTI selection is done on the top 8 bits of the variable field. This is mapped to the low MTI byte in a standard format message.

Some MTIs have additional status bits defined as part of the 2nd field. For example, there are two status bits associated with "Consumer Identified" which must be kept in the header since there is no room in the CAN data field. To simplify translation between formats, these are the low bits of the first byte after the MTI in a standard-form message.

- Designers may wish to use CAN hardware filtering, but it can't be assumed to be present. We assign a single bit to indicate "simple node protocol" messages to make simple filtering possible.

Left over from Frame doc

OpenLCB common messages are converted, to the extent possible, to single CAN frames via a one-to-one mapping.

The translations include:

- MTIs may have to be shortened, depending on bit allocations.

Design decisions:

- A specific mapping from the common 16-bit MTI format to a shorter CAN format is documented in a separate worksheet (PDF version). The common form is 16 bits to have lots of space to grow at the high end, but it can be mapped into a 8 bit field for efficient CAN transfer.

2.3 OpenLCB Common Message Format

A common message consists of several parts:

- Node ID Number of the Source node (SID)
- Message Type Indicator (MTI)
- In some cases, the Destination Node ID (DID)
- In some cases, a P/C Event ID (EID)
- Message content, as defined by the particular message type

Each OpenLCB wire protocol may define local representations for each of these components at the transport level and below. This may involve replacement (e.g. using a shorter "Alias" token for the node ID number), reordering, and/or specific representations that differ from the common ones, but in all cases it must be possible to translate from the wire protocol message to a complete common message using only locally available information.

By convention, multi-byte quantities in OpenLCB are represented in big-endian order. The most significant byte is sent first, and stored at the lowest address. This is the same as Ethernet and the common internet protocols, but not the same as the Intel x86 architecture.

Reserved quantities must be created with a zero value unless otherwise specified. When processing a message, any reserved quantities must be ignored unless otherwise specified. When transporting a message, reserved quantities must be transported unchanged. The zero value sometimes indicates a non-initialized value.

Node ID Numbers - NID

A node ID number is 48 bits. The NMRA will allocate number ranges to individuals or organizations upon request, and maintain a public list of assignments. A detailed mechanism for kinds of delegation has been separately proposed.

Message Type Indicator (MTI)

The common Message Type Indicator (MTI) is a 16 bit quantity.

Each specific MTI has a specific defined content documented elsewhere, but there are a few general points.

- Messages are variable length. The specific wire protocol is responsible for carrying length information as needed.
- If the message carries a destination address, that destination node ID (destination address) is the first part of the message content. The form of a Destination Node ID is defined by the particular wire protocol, but must be mappable to the full Node ID of the intended destination.

The common Message Type Indicator (MTI) is a 16 bit quantity. Note that specific wire protocols may remap this.

- The most-significant 5 bits are reserved as 00110; nodes must send and check that value.
- The next 7 bits are used to indicate the message type.
- The top two of these are used to form static priority groups. A 0 bit is considered to have more priority (can be processed first), a 1 bit less priority (can be processed later). The MSB makes a larger statement about priority than the LSB of these two. Priority processing is permitted but not required. The priority group bits are part of the overall message type.
- The next bit is reserved as 0; nodes must send and check that value.
- The lower four bits indicate a specific type.
- The following bit is reserved as 0; nodes much send and check that value.
- The 2nd from-least-significant bit indicates that this message carries a destination node address (DID) when set to 1. Setting 0 means that the message is globally addressed. If a Destination Node ID (DID) is present, it is located at the start of the message content. The form of a Destination Node ID is defined by the particular wire protocol, but must be mappable to the full NID of the intended destination.
- The next-to-least-significant bit indicates this message carries a P/C Event ID field when set to 1. Setting 0 means that the message does not carry a P/C event ID. If a P/C Event ID is present, it's at the start of the message, except after the Destination Node ID, if present.
- The least-significant bit when set to 1 indicates this message carries a flag byte after the DID and/or EID determined by the above bits. The low bits of that byte can be relocated in CAN messages, see the definition of the CAN wire protocol.

We've chosen to allocate bit fields to make decoding simpler; if possible, aligned on nibble boundaries to make it easy to read as hexadecimal numbers. Note that, as a special dispensation for CAN, higher priority messages (MTIs with lower numerical values) may pass lower priority ones; this must be taken into account when designing protocols.

Specific values are allocated and documented in a separate worksheet (PDF version). We keep them in just that one place to avoid conflicting updates.

OpenLCB Common Message Types

(Removed the datagram, stream, event message sections)

The content listed here is in addition to the originating NID, MTI and Destination Node ID, if any. The full format is specified in a separate note.

Some messages may be either addressed to a specific node, or globally addressed to all nodes. These are indicated by separate MTI values (see below).

Base Messages

Initialization Complete

Indicates that the node is complete, and once the message is delivered, reachable on the network.

Content:

- NID of the sending node

Only Global (unaddressed) form available.

Verify Node ID

Content: None

Global (unaddressed) or with specified destination address.

Verified Node ID

Content:

- NID of the sending node – this is sent in full 48 bit format in all wire protocols, even if an alternate form or alias is available elsewhere in the message

Only Global (unaddressed) form available.

Protocol Support Inquiry

Content: None

Destination address required.

Protocol Support Reply

Content: Bytes containing bits identifying the available OpenLCB protocols; see definitions in separate (spreadsheet) (pdf).

Destination address required.

Optional Interaction Rejected

Content:

- Mandatory original MTI
- Optional error code (TBD)
- Optional data content (TBD)

Destination address required.

Terminate Due to Error

Content:

- Mandatory most recent MTI
- Mandatory error code (TBD)
- Optional data content (TBD)

Destination address required.

Standard Interactions

All nodes must be able to take part in all standard interactions.

A) Node Initialization

Newly functional nodes, once their start-up is complete and they are fully operational, must send an "Initialization Complete" message.

- There is no guarantee that any other node is listening for this. No reply is possible.
- Nodes must not emit any other OpenLCB message before the “Initialization Complete” message.

Sending the IC message is required to insure that higher-level tools are notified that they may start to work with the node.

After the IC message is sent, and before any corresponding Producer/Consumer Event Report messages are sent, the node must identify all events produced or consumed on the board via zero or more Identify Consumers, Identify Consumed Range, Identify Producers and Identify Consumed Range messages. These are not required to be in any particular order.

B) Duplicate Node ID Detection

OpenLCB nodes must have unique node IDs. To detect this across the entire connected OpenLCB, all OpenLCB nodes must indicate an error if they detect an incoming message with a Source Node ID equal to their own. If possible, they should indicate it at the board itself using a light or similar. If possible, they should emit a PCER message with the “Duplicate Source ID detected” global event, which will carry the duplicate event ID in the Source Node ID field.

After sending the “Duplicate Source ID detected” global event, the node should not transmit any further messages until reset because this message will be received at the other duplicate-ID node(s), resulting in additional “Duplicate Source ID detected” global events and causing a possible message loop.

To further improve the reliability of this detection, OpenLCB nodes should, but need not, emit a Verified Node ID message every 30 to 90 seconds. As an implementation detail, it's recommended that CAN-attached nodes use their NIDa to pick that interval so that messages don't bunch up.

C) Node ID Discovery

Upon receipt of a Verify Node ID Number message addressed to it, or an unaddressed Verify Node ID Number message, a node will reply with an unaddressed Verified Node ID Number.

This can be used as check that a specific node is still reachable. When wire protocols compress the originating and/or destination NID, this can be used to obtain the full NID.

The standard Verify Node ID Number interaction can be used to get the full 48-bit NID from a node for translation. At power up each node must obtain a alias that is locally unique. Gateways will also have to obtain unique aliases for remote nodes they are proxying on to the segment.

D) Protocol Discovery

OpenLCB defines a number of specific protocols for interacting with nodes, including event exchange, datagrams, streams, configuration, etc. These protocols are optional, in the sense that not every node will implement every one.

To determine which protocols a node implements, a Protocol Support Inquiry message is sent to the specific node. It will reply with a Protocol Support Reply message that contains one or more bytes of data. A specific bit position has been reserved for each defined protocol in a separate spreadsheet (.ods, .pdf form). If the bit is zero or not present, the protocol is not supported and requests to use it will result in a error. If present and 1, the protocol is supported.

It is not necessary to check whether a protocol is supported by a node before attempting to use the protocol. If it's not, the error handling mechanism (see below) will indicate that.

E) Error Handling

There are multiple mandatory error-handling scenarios defined.

Reject Addressed Optional Interaction

- Node A receives an addressed message from Node B that carries Node A's NID.
- The MTI indicates the start of an optional interaction.
- If Node A does not want to take part in the optional interaction, it may send an Optional Interaction Rejected message addressed to Node B with the original MTI in the message content. There is no requirement that OIR be sent; the node may silently ignore the incoming message.

The message content also contains an optional reason code and an optional data value. The use of these fields is to be defined.

Reject Unaddressed Optional Interaction

- Node A receives an unaddressed message from Node B.
- The MTI indicates the start of an optional interaction.

If Node A does not want to take part in the optional interaction, it silently drops the message without reply.

Reject Addressed Standard Interaction Due to Error

- Node A is taking part in an addressed interaction with Node B. Either node may be able to send the next message.
- Some error condition prevents Node A from continuing the interaction.

- To terminate the interaction, Node A sends a Terminate Due to Error message to Node B. It then resets it's state so as to no longer be taking part in the addressed interaction.

The message content contains the most recent MTI received in this interaction, a mandatory reason code and an optional data value. The use of these fields is to be defined.

Table of Contents