OpenLCB Technical Note	
Method for Allocating MTI Values	
Sep 11, 2011	**Preliminary**

1 Introduction

OpenLCB messages carry a Message Type Indicator (MTI) field to indicate their type. Many nodes will treat MTIs as magic 16-bit numbers, just comparing them for equality to specific values of interest. That's a perfectly fine node implementation strategy.

5 This note describes how the numeric values for those MTIs are allocated. The current allocations are documented in a separate spreadsheet[1]. We keep them in just that one place to avoid conflicting updates. Those allocations are normative. The discussion in this note is not normative on OpenLCB users or node developers, but does describe the methods that are to be used for allocating new MTI values for new OpenLCB message and protocol types.

10 The primary MTI form is a 16-bit number. MTI information may be carried in different formats for different wire protocols. These are documented in the same spreadsheet, and discussed in separate sections below.

To allocate specific MTI values, having a systematic pattern to the values can be more useful.

We've chosen to allocate bit fields to make decoding simpler; if possible, aligned on nibble
15 boundaries to make it easy to read as hexadecimal numbers. We've also used a mix of bit-fields and individual flag bits to increase compatibility when additional MTI values are defined later.

There are two basic approaches to identify classes of message types, such as "addressed" vs "global" messages.

1. Use a dedicated bit field to distinguish the types, e.g. 1 indicates addressed and 0 means
20   global.

2. Encode in the type number, e.g. "A (addressed) is 1", "B (addressed) is 2", "C (global) is 3", "D (addressed) is 4".

The encoded form uses less bits, particularly if there are many classes to distinguish, which would require many dedicated bits. But future expansion is easier with dedicated bit fields,
25 because nodes can do some limited decoding of MTIs even though the node was created before the MTI value was defined. For example, a gateway can determine whether a message is global or addressed to a particular node, even if the MTI was defined after the node was built.

---

[1]See http://openlcb.org/specs/index.html for the current version of the spreadsheet. It provides concrete examples that may help you understand the material in this document.

2 Common MTI Values

30

Common MTI layout:

Field	Reserved	Simple Node flag	Reserved	Message Type Number	Reserved	Dest ID flag	Event ID flag	Flag Byte flag
Size & location mask	2 bits 0xC000	1 bit 0x2000	1 bit 0x1000	8 bits 0x0FF0	1 bit 0x0008	1 bit 0x0004	1 bit 0x0002	1 bit 0x0001
Value(s)	0 (send and check)	0 means for simple node, 1 means not	1 (send and check)	See below	0 (send and check)	1 means Destination ID present, 0 not present	1 means Event ID present, 0 not present	0 means flag byte present, 1 not present

- The two most-significant bits are reserved as 00; nodes must send and check that value.

35
- The next bit is used to indicate messages meant for "simple" or minimal nodes. A 1 in this bit means that these simple nodes can ignore this message. A 0 in this bit means that the simple nodes should process the message. See the Simple Node Protocol Description TN for more information. This bit is reserved to 0 for all addressed message types, as a message specifically delivered to a node should be processed.

40
- The next bit, the least significant of the top nibble, is reserved as 1; nodes must send and check that value.

- The next eight bits form a specific message type number. This has substructure:

Field	Reserved	Static priority groups	Reserved (send and check)	Specific type
Size & location (within 8-bit field)	1 bit 0x80	2 bits 0x60	1 bit 0x10	4 bits 0x0F
Value(s)	0 (send and check)	0 to 3 0 goes first, 3 last if priority processing is	0 (send and check)	See spreadsheet for values & meanings

5   Page 2 of 5 - 9/11/2011

		present		

- The most significant bit is reserved as 0; nodes must send and check that value.

45
- The next 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.

50
- The next 4 bits, forming the low nibble, are used to indicate the specific message type within a priority group.

- The bottom nibble of the MTI is interpreted as flags that define the structure and format of the message type.

- The most significant bit is reserved as 0; nodes must send and check that value.

55
- The 2nd from-least-significant bit when set to 1 indicates that this is an addressed message that carries a destination node address (DID). Setting 0 means that the message is globally addressed. If a Destination Node ID (DID) is present, it is at a specific location in the message content.

60
- 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 is at a specific location in the message content.

- 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 for some wire protocols, see e.g. the definition of the CAN wire protocol.

65 These flags allow nodes to do simple decoding of messages with MTIs that they don't recognize, perhaps because they were defined after the node was created. For example, gateways can use this to control routing of messages that they don't understand, perhaps because they were defined after the gateway was developed.

3 CAN MTI Considerations

70 MTI information is carried in a different format on CAN links to increase bandwidth efficiency, simplify decoding in small processors, and permit use of hardware filtering.

The standard CAN MTI field is 12 bits in the header (messages without destination address) or one byte in the data segment (addressed messages). Since CAN frames only carry 8 data bytes, a 1-byte MTI short form will be used until future expansion makes more necessary. The possibility of longer
75 MTI values has been reserved, see below.

Standard MTIs are mapped to one of seven types:

Type	Meaning
0	Unaddressed MTI for simple nodes
1	Unaddressed MTI for non-simple nodes
2	(Reserved for 16-bit MTI format, not used)
3	(Reserved, not used)
4	Datagram not-last segment
5	Datagram last segment
6	Addressed MTI other than datagram or stream
7	Stream Data

The bit coding of this has been chosen to use the "4" bit to indicate that a destination address is present, directly mapping to the "Destination ID present" bit in the full MTI.

80   The 4, 5 (Datagram) and 7 (Stream Data) values are special cases chosen for efficient processing of large amounts of data on CAN. Most MTIs will map to 0, 1 or 6.

The 0 vs 1 values map directly to the "Simple Node flag" in the full MTI.

3.1 Addressed CAN MTIs

Common MTIs with the "destination ID present" bit set are mapped to and from type 6.

85   In this case, the 12-bit destination alias is placed in the header, and the Message Type Number byte from the full MTI is the first byte of the CAN frame.

Field	CAN prefix	Type	Destination ID
Size & location (within 29-bit CAN Header)	2 bits 0x1800,0000	3 bits 0x0700,0000	12 bits 0x00FF,F000
Value(s)	3	6	12-bit alias for destination node

90   *3.2 Unaddressed CAN MTIs*

Common MTIs with the "destination ID present" bit set are mapped to and from type 0 or 1, depending on whether the "simple node" bit is reset or set.

In this case, the next twelve bits of the CAN header are available for MTI information. The first (most significant) eight bits are used for the Message Number byte. The last (least significant) four bits are
95   used for the Event ID present, the Flag Byte present flag, and any flag bits themselves.

Field	CAN prefix	Type	Number	EID Present	Flags Present	Flag Bits
Size & location (within 29-bit CAN Header)	0x1800,0000	0x0700,0000	0x00FF,0000	0x0000,8000	0x0000,4000	0x0000,3000
Value(s)	3	0 or 1	8 bit MTI type number	1 if Event ID present in CAN data	1 if Flag byte	Flag bits, if present

Flag bit handling is used to get an extra 2 bits of data into the CAN frame, e.g. for certain messages that contain an 8-byte Event ID plus some extra information. If a common MTI has the "Flag Byte" bit set,
100   the low order two bits are moved from the data content into the two "flag bits" in the CAN header.

Table of Contents