OpenLCB Working Note	
(title)	
Jan 30, 2021	Preliminary

1 Introduction

A Working Note is an intermediate step in the documentation process. It gathers together the content from various informal development documents, discussions, etc into a single place. One or more Working Notes form the basic for the next step, which is one or more Standard/TechNote pairs.

1.1 Served Use Cases

1.2 Unserved Use Cases

2 Specified Sections

This is the usual section organization for a Technical Note, to accumulate the Standard and Technical Note content in its eventual order.

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

Lorem ipsum dolor sit amet, consectetur adipiscing elit.[1] Fusce ornare mattis justo vitae imperdiet. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

2.4 Message Formats

2.4.1 Get Configuration Options Command

Byte 0	Byte 1
0x20	0x80

[1]See the “Common Information” OpenLCB Technical Note for detailed conventions on bit and byte numbering. Briefly, the least significant bit of a field is numbered with zero in OpenLCB descriptions, but note that other technologies may use other conventions.

2.4.2 Get Configuration Reply

Byte 0	Byte 1	Byte 2-3	Byte 4	Byte 5	Byte 6 (Optional)	Remainder (Optional)
0x20	0x82	Available Commands	Write Lengths	Highest Address Space	Lowest Address Space	Name String

2.4.3 Get Address Space Information Command

Byte 0	Byte 1	Byte 2
0x20	0x84	Address Space

2.4.4 Get Address Space Information Reply

Byte 0	Byte 1	Byte 2	Byte 3-6	Byte 7	Byte 8-11	Byte 12
0x20	0x86	Address Space	Highest Address	Flags	Lowest Address	Description

2.4.5 Freeze Command

Byte 0	Byte 1	Byte 2
0x20	0xA1	Address Space

2.4.6 Unfreeze Command

Byte 0	Byte 1	Byte 2
0x20	0xA0	Address Space

2.4.7 Lock/Reserve Command

Byte 0	Byte 1	Byte 2-7
0x20	0x88	Reserving Node ID

2.4.8 Lock/Reserve Reply

Byte 0	Byte 1	Byte 2-7
0x20	0x8A	Reserved Node ID

2.4.9 Get Unique ID Command

Byte 0	Byte 1	Byte 2
0x20	0x8C	Number requested (1-8, 3 bits); upper bits reserved and must be ignored

2.4.10 Get Unique ID Reply

Byte 0	Byte 1	Remainder
0x20	0x8D	1-8 requested EventIDs, a total of 8-64 bytes

2.4.11 Indicate Command

Byte 0	Byte 1
0x20	0xA3 on 0xA2 off

2.4.12 Update Complete Command

Byte 0	Byte 1
0x20	0xA8

2.4.13 Reset/Reboot Command

Byte 0	Byte 1
0x20	0xA9

2.4.14 Reinitialize/Factory Reset Command

Byte 0	Byte 1	Byte 2-7
0x20	0xAA	Node ID

		Get Config Options	Get Config Options Reply	Get Addr Space Info	Get Address Space Info Reply	Freeze/ Unfreeze
Datagram Type Byte		0x20	0x20	0x20	0x20	0x20
Command Byte (Upper is MSB, lower is LSB)	Command Type	0b10	0b10	0b10	0b10	0b10
	Operation Type	0b0000	0b0000	0b0001	0b0001	0b1000
	Reply	0b0	0b1	0b0	0b1	
	Reserved					0b0
	Freeze/ Unfreeze					Freeze 0b1 Unfreeze 0b0
	Reserved	0b0	0b0	0b0	0b0	
	Byte value	0x80	0x82	0x84	0x86	0xA0-0xA1
Byte 2			Available Commands (2 bytes)	Address Space	Address Space	
Byte 3					Largest Address (4 bytes)	
Byte 4			Write Lengths			
Byte 5			Highest Address Space			
Byte 6			Lowest Address Space (0-1 byte)			
Remainder			Name (0-N bytes, optional		Requires Alignment (4 bits) Low Address Non-zero (1 bit) Read-Only (1 bit)	
					{Lowest Address } (4 bytes)	
					{Desc } (0-N bytes)	

		Lock/ Reserve	Lock/ Reserve Reply	Get Unique ID	Get Unique ID Reply	Indicate	Update Complete	Reset/ Reboot	Reinit/ Factory Reset
Datagram Type Byte		0x20	0x20	0x20	0x20	0x20	0x20	0x20	0x20
Command Byte (Upper is MSB, lower is LSB)	Command Type	0b10	0b10	0b10	0b10	0b10	0b10	0b10	0b10
	Operation Type	0b0010	0b0010	0b0011	0b0011	0b1001	0b1010	0b1010	0b1010
	Reply	0b0	0b1	0b0	0b1				
	Reserved	0b0	0b0	0b0	0b0				
	Start/Stop					Start 0b1 Stop 0b0			
	Sub Command						0b00	0b01	0b10
	Byte value	0x88	0x8A	0x8C	0x8D	0xA2-0xA3	0xA8	0xA9	0xAA
Byte 2		Requesting NodeID	Locking NodeID	Number to reserve (3 bits, 1-8)	New Unique EventID (8 bytes, 1-8 times)				Target NodeID
Byte 3									
Byte 4									
Byte 5									
Byte 6									
Byte 7									
Remainder									

2.4.15 Get Configuration Options Reply

To make it possible to make simple/cheap nodes, not every configuration operation & option needs to be provided. The reply to “Get Configuration Options” provides information that a configuring device can use to control how it communicates with the node so that it only uses available modes.

- Available operations mask (2 bytes, bit coded): Indicate which operations are available so the using software can know whether convenience operations (which are not possible on some hardware) are available.

- 0x8000 Write under mask supported
- 0x4000 Unaligned reads supported. If not set, reads have to start on an address with the low bits, as given by the data size, all zero. For example a 4-byte write must have the low two address bits zero.
- 0x2000 Unaligned writes supported. If not set, reads have to start on an address with the low bits, as given by the data size, all zero. For example a 4-byte write must have the low two address bits zero.
- 0x0800 Read from address space 0xFC available (this is the manufacturer part of Abbreviated CDI)
- 0x0400 Read from address space 0xFB available (this is the user-entered part of Abbreviated CDI)
- 0x0200 Write to address space 0xFB available (this is the user-entered part of Abbreviated CDI)
- Others reserved, must be ignored on receipt and sent as zero.

- Write lengths supported (One byte, bit coded): (provided for devices that can only write certain sizes to memory) (at least one bit must be set)
  - 0x80 1 byte write
  - 0x40 2 byte write
  - 0x20 4 byte writes
  - 0x10 64 byte writes (full datagram, but not 63 bytes or arbitrary length, just exactly 64)
  - 0x02 arbitrary writes of any length OK
  - 0x01 stream writes supported (stream support will identify buffer size)
  - Others reserved, must be ignored on receipt and sent as zero.
- Highest Address Space (byte): Highest number space available. Not all up to that need be available, but sparse allocation will slow down the process as “Get Address Space Information” is needed to determine whether they are present.
- Lowest Address spaces (byte): Lowest number space available. Note that spaces 0xFD, 0xFE and 0xFF are assumed to be included even if the low space ↔ high space range doesn't include them. (also 0xFC, 0xFB of Abbreviated Default CDI if bits indicate they're available)

A node that only has the high spaces could have Highest Address Space = 255, Lowest Address Space = 253 or 251.

A node that has additional low address spaces, e.g. to make more memory available with a 28-bit address, could have Highest Address Space = 127, Lowest Address Space = 0 and leave the top spaces assumed.

2.4.16 Get Address Space Information Reply

To ease automated access, a configuring node can inquire about the address spaces in the being-configured node. Whether or not the address space is present, a reply is required.

- Present: This is carried in the lowest bit of the command byte, just below the reply bit
    - 0x01 == 1: Present. == 0 not present.
- Space ID – provided to identify request this reply is in response to
- Highest Address (4 bytes)
- Flags (byte) – (Alignment and size were going to be here but were made global above); Read-Only is LSB, can write if 0, can only read if 1; Non-zero lowest address is 2nd-lowest bit, low address is zero if 0, is non-zero and specified in next four bytes if 1
- Lowest Address (4 bytes) – optional, omit if zero, as that will let reply fit in single CAN frame; if present, “non-zero lowest address” bit in prior byte must be 1.
- Description (variable length) – optional null-terminated string giving the user-readable name of this space

2.4.17 Lock/Reserve and Freeze/Unfreeze

An OpenLCB node can, in general, be configured while the network and even the node itself is operating.

Code can be simplified by disabling operation of a node while it's being configured, so that there's no concern about it trying to react to transient incomplete information. The Freeze/Unfreeze command, if supported, can be used to tell a node that it should “freeze” operation, ignoring inputs, while the configuration is being updated. A reset of the node releases the freeze option, if set.

Although nodes can be configured by multiple other nodes, this can also lead to inconsistencies. The optional Lock/Release command can be used to avoid this. At the start of configuration, a configuring node sends a Lock message with its NodeID. If no node has locked this node, indicated by zero content in the lock memory, the incoming NodeID is placed in the lock memory. If a node has locked this node, the non-zero NodeID in the lock memory is not changed. In either case, the content of the lock memory is returned in the reply. This acts as a test&set operation, and informs the requesting node whether it successfully reserved the node. To release the node, repeat the lock operation with a zero NodeID. The lock memory is set to zero when the node is reset. Note that this is a voluntary protocol in the configuring nodes only; the node being configured does not change it's response to configuration operations when locked or unlocked.

2.4.18 Get Unique EventID

Nodes maintain a list of unique EventIDs for use in configuration. These are allocated based on the node's unique NodeID. This command allows a configuration tool to get new unique EventIDs from the node's pool, for example to interact with the Blue/Gold configuration process. Each request must provide a different EventID, without repeat, even through node resets and factory resets.

2.4.19 Update Complete/Reset/Reboot/Reinitialize

This is a collection of three operations, distinguished by what are normally the flag bits.

The configuration protocol does not specify the meaning of the transferred data. In particular, it doesn't specify when new configuration information takes effect. Depending on how the node is constructed, this might be immediately upon transfer (although this raises issues of write boundaries), or when an entire sequence of transfers is complete. “Update Complete” is the command that indicates that a series of configuration writes is consistent and complete, and the node can put it into effect. Nodes do not have to require this operation, but receiving it must be permitted. Configuration tools should send it at the end of operations. Nodes may, but are not required to, reset after sending the reply to this message.

The “Reboot/Reset” command is meant to reinitialize a node, equivalent to powering it up. Nodes should finish any pending operations, e.g. non-volatile memory writes, before doing the initialization. It's expected that the datagram reply will be sent before the reset, but this might not be entirely reliable. Configuration tools should not count on the reply. The configuring node will receive a “Node Initialization Complete” when the node is back up. This operation must not reset any configuration information to default contents.

“Reinitialize/Factory Reset” is similar, but includes restoring the node's configuration as if factory reset. (This may require creating new unique EventIDs, see other note) This is a heavy-weight operation which may require some form of interlock, e.g. the user pressing a button, to prevent inadvertent data loss. As a small safety precaution, the NodeID of the note being reset is redundantly carried in the data part of the datagram.

2.4.20 Indicate

(This is likely to move to a separate protocol; note that “indicate” and “ident” are completely different things, with “ident” in a separate protocol already)

This command tells the board to somehow identify itself to the user, for example by flashing a LED or operating it's outputs. This allows the user to be absolutely sure that he's configuring the correct board. “Start” (bit 0 = 1) means that the board should start indicating, and “Stop” (bit 0 = 0) means that the board should stop indicating. The data portion carries information that lets the board know what kind of indication to do. It's not always appropriate to operate outputs if they're e.g. driving large mechanical systems like doors.

2.4.20.1 Get Configuration Options Reply

No extra information given

2.4.20.2 Get Address Space Information Reply

No extra information given

2.4.20.3 Lock/Reserve and Freeze/Unfreeze

No extra information given

2.4.20.4 Get Unique EventID

Nodes maintain a list of unique EventIDs for use in configuration. These are allocated based on the node's unique NodeID. This command allows a configuration tool to get new unique EventIDs from the

node's pool, for example to interact with the Blue/Gold configuration process. Each request must provide a different EventID, without repeat, even through node resets and factory resets.

2.4.20.5 Update Complete/Reset/Reboot/Reinitialize

2.4.20.6 Indicate

No extra information given

2.5 States

2.6 Interactions

3 Background Information

Quisque sollicitudin tempor bibendum. Donec consectetur condimentum sollicitudin. Sed dignissim velit id felis lacinia at eleifend nisi laoreet. Vivamus tristique porta ornare. Vivamus feugiat dolor id lectus aliquet luctus.

180

Table of Contents

