OpenLCB

OpenLCB Standard	
Memory Access Configuration Protocol	
Dec 14, 2012	Preliminary

1 Introduction (Informative)

This document defines a protocol for configuring OpenLCB nodes by directly accessing their configuration memory.

2 Intended Use (Informative)

Intended to be used to configure self-contained nodes over their OpenLCB links.

See also the separate note on a Configuration Description Information.

3 References and Context (Normative)

For more information on format and presentation, see:

- OpenLCB Common Information Technical Note

Requires implementation of datagram protocol.

- OpenLCB Datagram Protocol

Must also implement Protocol Identification Protocol, identifying support for this protocol and for datagrams, and perhaps for streams. (That's required to simplify discovery of a datagram-based protocol.)

4 Protocol

Configuration messages use a specific datagram format consisting of the datagram type byte, followed by a single byte combining the operation "Command Type" field and flags. This is then followed by data in an operation-specific format. When present, the four-byte memory address follows first, then the address space byte if present, then other data.

4.1.1 Address Space Size

Configuration addresses are 32 bits. The addressable quantity is the byte.

4.1.2 Address Space Selection

Although a 32-bit address space is large enough to cover combined uses of memory, it can be more convenient to consider separate address spaces in the node. (This can also be considered to be a top digit in a global address space, if you want to, but note that the separate address spaces may cover the same memory objects, e.g. "all memory" and "Event ID configuration" spaces may reference the same configuration memory)

Required space definitions (these may or may not have content on a particular node); these address space numbers can only be used for this, and if the information is available, it must be accessible by these numbers (in addition to any others the designer might provide):

- (0xFF, flag=11) Configuration definition – reading this is how you get the configuration definition
- (0xFE, flag=10) All memory – provides access to "all" memory in the device, where "all" is defined by the designer. Single, flat address space for access. Can be used for e.g. dynamic access to RAM for monitoring & debugging.
- (0xFD, flag=01) Configuration - basic configuration space, with the structure of the 32-bit space defined by the designer.

These three spaces, inclusive, can be addressed without an extra byte in the datagram using control bits in the flag byte. All others need to be specified as a byte value. The high addresses (0xFD through 0xFF) were chosen for the dedicated spaces so that space numbers 00, 01, 02 could be used as a 5th bytes in a contiguous address if desired.

4.1.3 Message Formats

The following table shows available configuration operation formats. All others reserved. They must not be skipped during identification. Items in {} are optional.

Operation Name	**Command Type Field (2 bits)**	**Op Type Field (4 bits)**	**Lower Bits (2 bits)**					
	Command Byte			**Additional Bytes**				
Write	0x0	Stream/Datagram (1 bit) = 0 ReadReply/Write (1 bit) = 0 Under Mask (1 bit) Reserved (1 bit)	Address Space (2 bits)	Address (4 bytes)	{Space} (1)	Data (1-N)		

Operation Name	**Command Type Field (2 bits)**	**Op Type Field (4 bits)**	**Lower Bits (2 bits)**					
	Command Byte			**Additional Bytes**				
Write Stream	0x0	Stream/Datagram (1 bit) = 1 Reserved (1 bit) = 0 Under Mask (1 bit) Reserved (1 bit)	Address Space (2 bits)	Address (4 bytes)	{Space} (1 bytes)	Stream information		
Read-Reply	0x1	Stream=1/Datagram=0 (1 bit) ReadReply/Write (1 bit) = 1 Reserved (2 bits)	Address Space (2 bits)	Address (4 bytes)	{Space} (1 bytes)	Data (1-N bytes)		
Read	0x1	Stream/Datagram (1 bit) = 0 ReadReply/Write (1 bit) = 0 Reserved (2 bits)	Address Space (2 bits)	Address (4 bytes)	{Space} (1 bytes)	Count (1 byte, most significant bit reserved)		
Read Stream	0x1	Stream/Datagram (1 bit) = 1 Reserved (3 bits)	Address Space (2 bits)	Address (4 bytes)	{Space} (1 bytes)	Count (4 bytes)	Stream information	

Operation Name	**Command Type Field (2 bits)**	**Op Type Field (4 bits)**	**Lower Bits (2 bits)**					
	Command Byte			**Additional Bytes**				
Get Configuration Options	0x2	0x0	Reply=0 (1 bit) Reserved (1 bit)					
Get Configuration Options Reply	0x2	0x0	Reply=1 (1 bit) Reserved (1 bit)	Available commands (2 bytes, see section 4.1.4.2 coding)	Write lengths (1 byte, see section 4.1.4.2 coding)	Highest Space (1 byte)	Lowest Space (1 byte)	{Name }
Get Address Space Info	0x2	0x1	Reply=0 (1 bit) Reserved (1 bit)	Space ID (1 byte)				
Get Address Space Info Reply	0x2	0x1	Reply=1 (1 bit) Present/Absent (1 bit)	Space ID (1 byte)	Largest Address (4 bytes)	Requires Alignment (4 bits) Low Address Non-zero (1 bit) Read-Only (1 bit)	{Lowest Address } (4 bytes)	{Desc }
Lock/Reserve	0x2	0x2	Reply=0 (1 bit) Reserved (1 bit)	NodeID (6 bytes)				

Operation Name	**Command Type Field (2 bits)**	**Op Type Field (4 bits)**	**Lower Bits (2 bits)**					
	Command Byte			**Additional Bytes**				
Lock/Reserve Reply	0x2	0x2	Reply=1 (1 bit) Reserved (1 bit)	NodeID (6 bytes)				
Get Unique ID	0x2	0x3	Reply=0 (1 bit) Reserved (1 bit)	Number to reserve (3 bits, 1-7)				
Get Unique ID Reply	0x2	0x3	Reply=1 (1 bit) Reserved (1 bit)	New Unique EventID (8 bytes, 1-7 times)				
Reserved	0x2	0x4-0x7	Reserved (2 bits)					
Freeze/Unfreeze	0x2	0x8	Reserved (1 bit) Freeze/Unfreeze (1bit)					
Indicate	0x2	0x9	Reserved (1 bit) Start/Stop (1 bit)	(somehow identify outputs, LEDs, etc to drive?)				

Operation Name	**Command Type Field (2 bits)**	**Op Type Field (4 bits)**	**Lower Bits (2 bits)**					
	Command Byte			**Additional Bytes**				
Update Complete	0x2	0xA	0x0					
Reset/Reboot	0x2	0xA	0x1					
Reinitialize / Factory Reset	0x2	0xA	0x2	Target NodeID (6 bytes)				
Reserved	0x2	0xA	0x3					
Reserved	0x2	0xB-0xF	Reserved (2 bits)					

4.1.4 Operations

4.1.4.1 Read, Read-Reply

Attempts to read from an invalid location, either outside the available address range in an valid address space, or from an invalid address space, still require a return message, requires a reply datagram with an data length of zero.

In general, a read reply may provide less than the requested data, but always at least one byte if it's a valid read. The maximum read request is 64 bytes when reading via datagrams. When reading via streams, any length up to 0xFF,FF,FF,FF (4GB-1) can be requested; a length of 0x0 means “read forever”.

4.1.4.2 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.

4.1.4.3 Get Address Space Information Reply

To ease automated access, a configuring node can enquire 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

4.1.4.4 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.

4.1.4.5 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.

4.1.4.6 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.

4.1.4.7 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. (This needs to be specified more precisely)

Table of Contents

1 Introduction (Informative)....................1
2 Intended Use (Informative)....................1
3 References and Context (Normative)....................1
4 Protocol....................1
4.1.1 Address Space Size....................1
4.1.2 Address Space Selection....................1
4.1.3 Message Formats....................2
4.1.4 Operations....................6
4.1.4.1 Read, Read-Reply....................6

4.1.4.2 Get Configuration Options Reply.. 6
4.1.4.3 Get Address Space Information Reply...8
4.1.4.4 Lock/Reserve and Freeze/Unfreeze.. 8
4.1.4.5 Get Unique EventID.. 8
4.1.4.6 Update Complete/Reset/Reboot/Reinitialize... 9
4.1.4.7 Indicate...9