OpenLCB test plan for the Memory Configuration Standard

The OpenLCB Group

February 9, 2024

1 Introduction

This note documents the procedure for testing an OpenLCB implementation against the Memory Configuration Standard.

The tests are traceable to specific sections of the Standard.

The testing assumes that the Device Under Test (DUT) is being exercised by other nodes on the message network, e.g. is responding to enquiries from other parts of the message network.

2 Required Equipment

See the separate "Installing the OpenLCB Test Software" document for initial installation and set up of the test program.

If a direct CAN connection will be used, a supported USB-CAN adapter [1] is required. Connect the adapter to the DUT using a single UTP cable and connect two CAN terminators.

Provide power to the DUT using its recommended method.

3 Set Up

The following steps need to be done once to configure the test program:.

1. Start the test configuration program.
2. Select "Set Up DUT".

[1]See "Installing the OpenLCB Test Software"

3. Get the Node ID from the DUT[2]
4. Enter that Node ID into the program.
5. Configure the test program for the USB-CAN adapter's device address or the TCP hostname and port.
6. Quit the test program and reply "Y" to "Save configuration?" when prompted.

The following steps need to be done at the start of each testing session.

1. Check that the DUT is ready for operation.
2. Start the test program.

4 Memory Configuration Procedure

Select "Memory testing" in the test program, then select each section below in turn. Follow the prompts for when to reset/restart the node and when to check outputs against the node documentation.

A node which does not self-identify in PIP that it supports Memory Configuration will be deemed to have passed these tests. [3]

This plan does not check:

1. Stream-based memory configuration operations
2. Writes to configuration memory. THere's no generally-compatible way to do that. You can't even assume that a read / write / read sequence will return you to the original state, because nodes may react when information is written to them.
3. The Reinitialize/Factory Reset Command and the Get Unique ID Command and reply because these permanently change the state of the node under test.

4.1 Configuration Options testing

This section tests the messages and interaction in Standard sections 4.13 and 4.14.

The tester sends a Get Configuration Options Command datagram. It then checks:

1. That the datagram reply is received.
2. That a Get Configurations Options Reply datagram is received,
3. The reply datagram i sat least six bytes long,

[2]Where do we require this to be marked on a node?

[3]Using the -p option or setting the checkpip default value False will skip this check.

4. That the reserved bits in bytes 2, 3 and 4 are zero.

4.2 Address Space Information testing

This section tests the messages and interaction in Standard sections 4.15 and 4.16.

For each of the required[4] 0xFF, 0xFE and 0xFD memory spaces, the tester sends a Get Configuration Options Command datagram. It then checks:

1. That the datagram reply is received.
2. That a Get Address Space Information Reply datagram is received,
3. The reply datagram i sat least eight bytes long,
4. That the reply datagram refers to the same memory space as the request,
5. That the reply datagram shows the space is present,
6. That the reserved bits in byte 7 are zero.

4.3 Read Operations testing

This section tests the messages and interaction in Standard sections 4.4 and 4.5.

The tester sends Read Commands to space 0XFD, the memory configuration space.

1. A read of 64 bytes from address 0,
2. A read of 2 bytes from address 0,
3. A read of 64 bytes from address (Space Length - 4), which should return four bytes.

Each of these are done once with the address space in byte 1 and once with the address space in byte 6.

For each, the tester checks:

1. The datagram reply has the Reply Pending bit set,
2. The reply datagram is long enough to be checked,
3. It's a Read Reply datagram,
4. The error bit is not set,
5. The space matches (in either form),
6. The starting address matches,

[4]See section 4.2

7. The right number of data bytes are returned.

4.4 Lock/Reserve testing

This section tests the messages and interaction in Standard section 4.17 and 4.18.

The tester requests that the node be reset/restarted then

1. Sends a Lock/Reserve Command with node id A and checks that a Lock/Reserve Reply is received with contents zero.
2. Sends a Lock/Reserve Command with node id B and checks that a Lock/Reserve Reply is received with contents A.
3. Sends a Lock/Reserve Command with node id B and checks that a Lock/Reserve Reply is received with contents A.
4. Sends a Lock/Reserve Command with zero and checks that a Lock/Reserve Reply is received with contents A.
5. Sends a Lock/Reserve Command with node id B and checks that a Lock/Reserve Reply is received with contents zero.
6. Sends a Lock/Reserve Command with node id A and checks that a Lock/Reserve Reply is received with contents B.
7. Sends a Lock/Reserve Command with node id A and checks that a Lock/Reserve Reply is received with contents B.
8. Sends a Lock/Reserve Command with zero and checks that a Lock/Reserve Reply is received with contents B.
9. Sends a Lock/Reserve Command with zero and checks that a Lock/Reserve Reply is received with contents zero.

4.5 Reset/Reboot testing

This section tests the message and interaction in Standard section 4.24 Reset/Reboot Command.

The tester sends a Reset/Reboot Command and then checks that a Node Initialization Complete message is received to indicate a reboot. This may or may not have been preceded by a Datagram Received OK response.