Checking the OpenLCB Memory Configuration Standard

The OpenLCB Group

February 10, 2024

1 Introduction

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

The checks are traceable to specific sections of the Standard.

The checking assumes that the Device Being Checked (DBC) is being exercised by other nodes on the message network, e.g. is responding to enquiries from other parts of the message network.

1.1 Required Equipment

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

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

Provide power to the DBC using its recommended method.

2 Set Up

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

1. Start the checker configuration program.
2. Select "Set Up".
3. Get the Node ID from the DBC[2]

[1] See "Installing the OpenLCB Checker Software"

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

4. Enter that Node ID into the program.
5. Configure the checker program for the USB-CAN adapter's device address or the TCP hostname and port.
6. Return from the setup section and reply "Y" to "Save configuration?" when prompted.
7. Quit from the program.

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

1. Check that the DBC is ready for operation.
2. Start the checker program.

3 Memory Configuration Procedure

Select "Memory checking" in the 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 checks. [3]

This plan does not check:

1. Stream-based memory configuration operations
2. Write and Write Under Mask 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 because these permanently change the state of the node being checked.

3.1 Configuration Options checking

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

The checker 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,

[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.

3.2 Address Space Information checking

This section checks 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 checker 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.

3.3 Read Operations checking

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

The checker sends Read Commands to space 0xFD, the memory configuration space.

1. A read of 64 bytes from address 0,
2. A read of 10 bytes from address 0,
3. A read of 2 bytes from address 0,

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 checker 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.

3.4 Lock/Reserve checking

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

The checker 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.

3.5 Reset/Reboot checking

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

The checker 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.