Checking the OpenLCB CAN Frame Level Protocols

The OpenLCB Group

February 9, 2024

1 Introduction

This note documents the procedure for checking an OpenLCB implementation against the against the CAN Frame Transfer 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. Quit the checker program and reply "Y" to "Save configuration?" when prompted.

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 Frame Level Procedure

Select "Frame Layer 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.

3.1 Initialization

This section's tests cover Frame Transfer Standard sections 4, 6.1 and section 6.2.1.

The tests assume that the node reserves a single alias at startup.

Follow the prompts when asked to reset or otherwise initialize the DBC.

The tester waits up to 30 seconds for the node to restart and go through a node reservation sequence.

1. All frames carry the same source alias
2. The sequence of four RID frames, a CID frame, and AMD frame are sent
3. The Node ID in the RID frames matches the Node ID in the AMD frame
4. That the Node ID matches that of the node under test
5. Neither the alias[3] nor the Node ID[4] is zero.

3.2 AME Sequences

This section's tests cover Frame Transfer Standard sections 4, 6.1 and section 6.2.3.

The tests assume that the node has previously reserved at least one alias and is in the Permitted state.

[3] See section 6.3 of the Standard
[4] See section 5.12 of the Unique Identifiers Standard.

The tester sends an AME frame with no NodeID and checks for:

1. An AMD frame in response
2. That carries the Node ID of the DBC

The tester sends an AME frame with the Node ID of the DBC and checks the response for:

1. An AMD frame in response
2. That carries the Node ID of the DBC

The tester sends an AME frame with a Node ID different from the Node ID of the DBC and checks for no response.

3.3 Alias Conflict

This section's tests cover Frame Transfer Standard sections 4, 6.1 and section 6.2.5.

The tests assume that the node has previously reserved at least one alias and is in the Permitted state.

The tester sends an AME frame to acquire the DBC's current alias from the AMD response.

The tester sends an CID frame with the DBC's alias and checks for

1. An RID frame in response
2. That carries the source alias of the DBC.

The tester sends an AMD frame with the DBC's alias and checks for

1. An AMR frame in response
2. That carries the source alias of the DBC.

At this point, Frame Transfer Standard section 6.2.5 specifies that the node must stop using that alias. Most nodes will reserve a different one at this point.

If an initialization sequence is not started, the node passes.

If an initialization sequence does start, it will be checked as in the Initialization check above. In addition, the tester will check that the newly reserved alias is different from the original one.

3.4 Reserved Frame Bit

This section's tests cover Frame Transfer Standard sections 4, 6.1 and section 6.2.3., specifically that the 0x1000_0000 bit in the CAN header is properly ignored.

The tester sends an AME frame with zero in the 0x1000_0000 bit and with no NodeID and checks for:

1. An AMD frame in response,
2. That carries the Node ID of the DBC,
3. With the 0x1000_0000 bit set to one.