Checking the OpenLCB CAN Frame Level Protocols

The OpenLCB Group

February 25, 2024

1 Introduction

This note documents the procedure for checking an OpenLCB implementation 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. .

2 Frame Level Procedure

Select "Frame Layer 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.

2.1 Initialization

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

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

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

The checker 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 being checked
5. Neither the alias[1] nor the Node ID[2] is zero.

2.2 AME Sequences

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

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

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

2.3 Alias Conflict

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

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

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

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

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

The checker 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.

[1] See section 6.3 of the Standard
[2] See section 5.12 of the Unique Identifiers Standard.

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

If an alias reservation sequence does start, the first frame will be checked for a proper CID frame. In addition, the checker will check that the newly reserved alias is different from the original one.

Having reserved an alias, many nodes will put it into use with an AMD frame. If one is received, it will be checked to see if it contains the new alias and the DBC's node ID.

Once traffic on the bus has paused, indicating the completion of any allocation(s), the checker will emit an AME frame.

1. If an AMD has been received, there should be exactly one reply with the DBC's node ID. It should carry the new alias.
2. If an AMD has not been received, there should be no replies with the DBC's node ID or with the new alias.

Note: The Standards allow other initialization sequences, and this check is not completely general. If a DBC doesn't pass, the sequence should be carefully examined in light of the CAN Frame Transfer Standard to determine if it's just an issue with the check logic.

2.4 Reserved Frame Bit

This section's checks 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 checker 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.