Checking the OpenLCB Simple Node Information Protocol Standard

The OpenLCB Group

February 9, 2024

1 Introduction

This note documents the procedure for checking an OpenLCB implementation against the Simple Node Information Protocol 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".

[1] See "Installing the OpenLCB Checker Software"

3. Get the Node ID from the DBC[2]
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 Simple Node Information Protocol Procedure

Select "SNIP 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 the Simple Node Information Protocol will be deemed to have passed these checks. [3]

3.1 SNIP reply checking

This section checks the format of the reply message in Sections 4.2 and 5.1 of the Standard.

It does this by issuing a Simple Node Information Request message, accumulating the reply(s), and then checking:

1. The message indicates its source is the DBC.
2. The message indicates its destination is the checking node.
3. The version byte at the start of the first section is either 1 or 4.
4. The version byte at the start of the second section is either 1 or 2.
5. There are exactly six zero bytes.
6. Each of the six defined strings is no longer than its defined maximum length.
7. There are no data byte(s) after the sixth zero byte.

[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.