Installing the OpenLCB Checker Software Basic Version

The OpenLCB Group

February 11, 2024

1 Introduction

This document describes how to obtain and run a set of basic checks for OpenLCB nodes.

The checks are based on the Python 'openlch' module. More information on that can be obtains from its GitHub project site.

For more information on the checks, see the package documentation or the directory of checking plans.

2 Obtaining the Software

The software is distributed as a set of inter-connected Python source files.

2.1 Obtaining and Using via Git

If you're using Git,

```
git clone https://github.com/bobjacobsen/PythonOlcbNode.git
```

will create a PythonOlcbNode directory containing the most recent version of the software. This also contains git tags for the released versions.

2.2 Obtaining by Downloading a .zip File

You can get a download of the most recent released version by going to the project's Github releases web page tag section [1] and clicking the .zip or .tgz icon on the most recent release.

[1] Linked above or see https://github.com/bobjacobsen/PythonOlcbNode/tags

To get the very most recent version, [2] go to the project's Github main web page tag section, click the green Code button, and select "Download Zip".

Expand the downloaded file in a suitable place.

2.3 Prerequisites

You need to have Python 3.10 installed to run the program. Consult your computer's documentation for how to install that. Many computers already have it installed.

To run the CDI checks, the 'xmlschema' Python module must be installed. To do that, enter[3]

```
python3 -m pip install xmlschema
```

3 Configuring for Running

You need to have PYTHONPATH defined to include the main directory. [4] In the Linux and macOS terminals, you can do this with

```
export PYTHONPATH=$PWD
```

while in the distribution directory (where you installed the code) or you can place the same line at the end of your startup configuration file.

If you don't add this to your startup configuration file, you'll have to do this each time you start a terminal session.

Next

```
cd olcbchecker
```

to get to the right directory for running the code.

To start the program:

```
python3.10 control_master.py
```

Depending on your Python installation, this simpler form may also work:

```
./control_master.py
```

[2] But if you want to stay current with development of the tools, you should probably be using Git.
[3] This is the command for Linux nad MacOS; the Windows command may be different.
[4] Eventually, this will no longer be necessary, but not quite yet.

4 Configuring the Checker Program

When you first start the program, you'll be shown a basic menu:

```
OpenLCB checking program
 s Setup

 0 Frame Transport checking
 1 Message Network checking
 2 SNIP checking
 3 Event Transport checking
 4 Datagram Transport checking
 5 Memory Configuration checking

 q  Quit
>>
```

Type 0 and hit return to get the setup menu:

```
The current settings are:
  hostname = None
  portnumber = 12021
  devicename = None
  targetnodeid = None
  ownnodeid = 03.00.00.00.00.01
  checkpip = True
  trace = 10

c change setting
h help
r return
>>
```

At a minimum, you should define how to connect to your OpenLCB network, and the Node ID of the device you want to check.

To change the Node ID, select the"change setting" option and work through the prompts:

```
>> c
enter variable name
>> targetnodeid
enter new value
>> 02.01.57.00.04.9A
The current settings are:
```

```
  hostname = None
  portnumber = 12021
  devicename = None
  targetnodeid = 02.01.57.00.04.9A
  ownnodeid = 03.00.00.00.00.01
  checkpip = True
  trace = 10

c change setting
h help
r return

>>
```

Get the proper value from either a label on the device, or from its documentation. [5]

There are currently two ways to connect the program to your OpenLCB network:

1. Via a USB-CAN adapter, or
2. Via a GridConnect-format TCP/IP connection.

For a USB-CAN connector, define the devicename to be the address of the device in your computer, e.g. /dev/cu.usbmodemCC570001B1 or COM7.

For a TCP/IP link, define the hostname to be the IP address or host name to be used for connecting.

You must specify one or the other of hostname and device name, but not both. When you enter one, the other will be set to None.

When done with setup, select return. You'll be asked if you want to save changes. Select y to save and n to skip saving.

```
>> r

Do you want to save the new settings? (y/n)
 >> y
Stored
Quit and restart the program to put them into effect

OpenLCB checking program
```

[5] Some checks, but not all, can determine the node ID themselves if you leave the value as None. This is only reliable if there's just one node on your OpenLCB network. Note that some OpenLCB hubs add a node of their own to the node being checked.

```
 s Setup

 0 Frame Transport checking
 1 Message Network checking
 2 SNIP checking
 3 Event Transport checking
 4 Datagram Transport checking
 5 Memory Configuration checking

 q  Quit
>>
```

Quit and restart the program to put your changes into effect.

5 Running Checks

5.1 Required Equipment

It's generally best to have the device being checked (DBC) as the only device on the OpenLCB network.

If a direct CAN connection will be used, a supported USB-CAN adapter is required [6]. Connect the adapter to your computer as indicated in its instructions. Connect the adapter to the DBC using a single UTP cable and attach two CAN terminators.

If a TCP/IP GridConnect connection will be used, configure the DBC to connect to the TCP/IP hub when restarted. Note that if the DBC is providing the hub for the connection and restarting the DBC breaks connections to that hub, several restarting checks will indicate problems when the connection breaks.

Provide power to the DBC using its recommended equipment and connections.

6 Technical Information

Your selected defaults are stored in the localoverrides.py file. The original values are stored in the defaults.py file.

Should something corrupt the localoverrides.py file, you can delete it, restart the program, and re-enter your configuration.

[6] The checker has been checked with the RR-CirKits LCC buffer-USB, but others with similar operation characteristics will probably work.