Introduction to LCC®/OpenLCB™

SPROG DCC Ltd

© Copyright 2024 SPROG DCC Ltd

Copyright © September 2024 SPROG DCC Ltd.

This document may be distributed it under the terms of Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/), version 3.0 or later.

Contact:

SPROG DCC Ltd
Anvil House, Butt Lane
Harbury
Leamington Spa CV33 9JL
United Kingdom
https://www.sprog-dcc.co.uk

sprog@sprog-dcc.co.uk

Date	Revision	Comments
November 2023	1	Created
November 2023	1.1	Picture of example LCC Node
March 2024	1.2	Minor rewrites, fix typos
September 2024	1.3	Producer/Consumer Illustrations Summarize cabling requirements

Unless otherwise notes references in this document to LCC apply equally to OpenLCB, and vice-versa.

LCC® is a registered trademark of the NMRA

OpenLCB™ is a trademark of the OpenLCB Group

Contents

© Copyright 2024 SPROG DCC Ltd

1 Introduction

Layout Command Control (LCC) is the new standard Layout Control Bus (LCB) for model railways. LCC is the name for standards adopted by the National Model railroad Association (NMRA) from the open-source Open Layout Control Bus (OpenLCB) standards developed by the OpenLCB Group www.openlcb.org.

You should refer to the LCC and/or OpenLCB standards and technical Notes and other related standards for more in-depth details of LCC and how it operates. This document should give you a basic understanding of the concepts of LCC and how to apply it to your layout.

Some familiarity with the Java Model Railroad Interface (JMRI) software is assumed.

An understanding of bits, bytes and hexadecimal notation will be useful.

2 Basic concepts

2.1 Nodes, Bus and Network

Each "thing" connected to LCC (i.e., an electronic module on a circuit board, a computer program) contains one or more "nodes". The simplest nodes take inputs, such as buttons, switches and occupancy detectors, and control outputs such as signals and lights. More complex nodes might perform internal processing on input and outputs (e.g., signal interlocking, automation) or present a human interface to the layout operator.

Nodes communicate with each other over a collection of wires known as the bus. Inputs on one node may control outputs on a different node. All nodes send messages using a common language, or protocol, defined by the specifications, such that nodes from different manufacturers will work together.

An example LCC node is shown Figure 1.

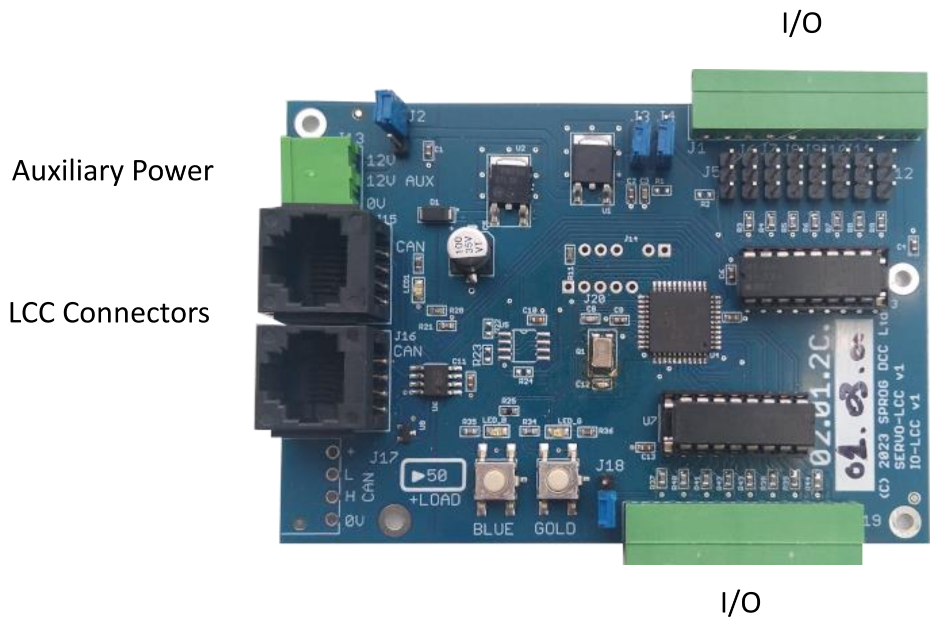


Figure 1- LCC Node

The collection of nodes and the bus can be referred to as the LCC network.

2.2 Events

Events are at the heart of LCC communications and layout control. An event is a numerical value sent in a message that indicates the availability of information. The information could be that something has happened on the layout, e.g., a loco has been detected, a turnout has completed moving or it could indicate user input such as pushing a button on a control panel. Events are split into groups according to their function. We will look at the manufacturer specific events that are used for layout control. Other groups such as the so-called "well-known global identifiers" are beyond the scope of this document, but see the LCC standard S-9.7.0.3 on the NMRA website and related technical note TN-9.7.0.3.

2.2.1 Globally Unique Event IDs

Each LCC node has a globally unique Node Identification (node ID), which is set at the factory and cannot generally be changed by the user. In computer speak, the node ID is 48 bits or 6 bytes, based on the identity of the manufacturer as assigned by the NMRA or the Open LCB group. By default, in a factory fresh node, events are generally formed by adding 16 bits, or 2 bytes, to the end of the node id, creating a 64-bit event ID. Thus, a node can create 65,536 unique events in response to its inputs or internal state. It should be noted, however, that a node may be configured to produce ***any*** event, including events identical to those produced by other nodes.

Keeping the default event IDs, where possible, gives an indication of where the node was produced, as well as it's intended meaning. Alternatively, users can create their own event ID scheme based solely on the events' meaning. Software configuration tools can help in the visualisation of events and their usage, see 4.2 The Event Table.

Nodes can respond to any number of 64-bit events, limited only be the memory and processing power of the node.

2.2.2 State versus Change of State

Consider a pushbutton whose contacts are open when not pushed and closed when pushed. The state of the contacts can be said to open or closed. When the button is pushed, or released, a change of state occurs from open to closed or vice-versa.

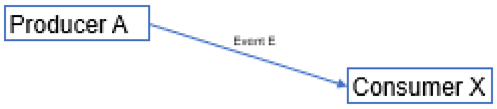
Event are transient and simply indicate such a change of state, e.g., a button was pushed, or a block has become occupied. A different event may be used to indicate when the button is released or the block becomes empty. The node that creates the event may have memory and hold the state of the button or block. Similarly, a node that sees the event may set a flag in memory to hold what it thinks is the state in the sending node. LCC has mechanisms to determine the state, e.g., when the layout is first turned on, and allow nodes to have a consistent view of the layout, e.g., a panel may indicate turnout states, but there is no guarantee that multiple nodes will hold consistent state unless appropriate events are created to signal each change of state.

2.3 The Producer/Consumer Model

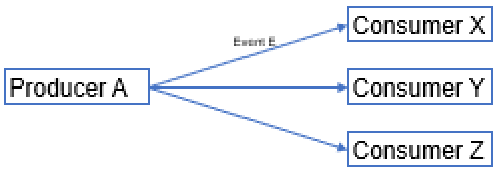
LCC operates on what's called the Producer/Consumer (P/C) model. A node that sends an event is a producer and a node that acts upon an event is a consumer of that event. Nodes are said to produce or consume events. An event might be referred to as a producer event, produced event (just another name for an event that has been produced), consumer event or consumed event (just another name for an event which is consumed).

The P/C model allows a great deal of flexibility in how events are used to control things on the layout. Some examples follow:

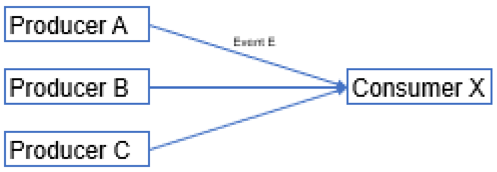
- One-to-one: A single producer and consumer use the event, e.g., a button controlling a turnout.



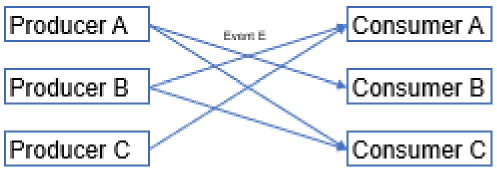
- One-to-many: A single producer sends an event that is consumed by multiple consumers, e.g., an event produced by a button press may be consumed by a turnout controller and a panel display.



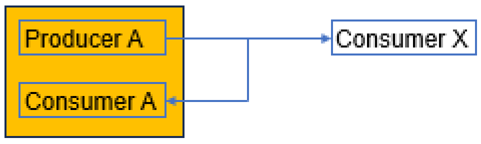
- Many-to-one: Multiple nodes may produce the ***same*** event, e.g., multiple control panels can produce an event to move a turnout.



- Many-to-many: E.g., a combination of One-to-many and many-to-many.



Nodes can be both producers ***and*** consumers, for the same or different events. This means that as well as consuming events from another node, a node may consume the events that it produces itself, e.g., a control panel node may produce an event when a button is pressed, to switch a turnout. The same node may consume this event to set a route indication on the panel.



Nodes may produce events in response to an internal change of state. An example would be a pulsed output. E.g., an event is consumed to turn on an output for a pre-determined time. At the end of the timed period the change of state of the output, from on to off, may produce an event.

Nodes may be able to perform logic operations that combine produced and consumed events, input and output changes of state and internal state to produce new events.

Nothing in the specifications limits the complexity or processing power of a node.

2.4 LCC and Controller Area Network (CAN) Bus

LCC is designed to be flexible and to be applicable to many communications mediums such as Ethernet and wireless.

At the time of writing, the most well specified, and widely used, version of LCC is based on CAN bus, a very robust technology widely used in automotive and industrial applications. Using a well-established standard makes it easy to guarantee operation, at the electrical level, between nodes from different manufacturers. A single CAN bus can connect 50 – 75 nodes over 100s of feet of cable. An LCC network requiring more nodes may be built from multiple CAN bus connections, connected by CAN bus repeaters or other bus technologies. Such layouts are beyond the scope of this document, but see the LCC technical note TN-9.7.1.1 on the NMR a website.

The LCC adaptations to CAN bus do not limit LCC in any way. Long messages using the Datagram or Streaming protocols are fragmented into multiple CAN messages by the producer and reassembled by the consumer.

LCC uses only CAN extended frames and data is sent on the CAN bus at 125 kbits/s.

2.5 I/O Channels

For SPROG DCC Ltd products, we talk about nodes having I/O channels, where each channel is a single output or input (or even both) that can be configured to do something, for example:

- Turn an output channel on when a particular event is consumed.
- Produce an event when the input changes from off to on.
- Control an LED and monitor a button (which we call ButtonLED)

Other manufacturers may use different terminology.

2.5.1 Polarity

Inevitably we need to talk about what is meant by an I/O channel being on or off. These states are defined in terms of the voltage at the connection for the channel.

2.5.1.1 Logic Level

A logic level channel is a channel that works with nominal voltages of 0 and 5 V, although 0 and 3.3 V is becoming increasingly common. There will be some range of acceptable voltage but we will refer to 0 V and 5 V for the remainder of this document.

Logic level inputs will draw very little current from the source of the signal.

Outputs will be capable of sourcing (driving out) or sinking (drawing in) limited current. Typically, enough current is available to illuminate a standard Light Emitting Diode (LED). If higher currents are required, e.g., to drive a relay, than some form of current amplifier such as a transistor is required. This may be included as an option on some modules.

2.5.1.2 Active High

An active high channel defines the on state as 5 V and the off state as 0 V.

An active high input is on when the input is 5 V and off when the input is at 0 V.

An active high output is on when the output is 5 V and driving current into a load.

An active high output is off when the output is 0 V and drawing current from a load.

2.5.1.3 Active Low

An active low channel defines the on state as 0 V and the off state as 5 V. An active channel might be described as “inverted” to describe the swapping, or inversion, of the voltage levels.

An active low input is on when the input is 0 V and off when the input is at 5 V.

An active low output is off when the output is 5 V and driving current into a load.

An active low output is on when the output is 0 V and drawing current from a load.

2.5.1.4 It Can Get Even More Confusing

The way a load is connected can also influence what is perceived as on and off states of an I/O channel.

Consider, for example, an active high output with an LED connected between the output and 0 V. The LED will be on (lit) when the output is on. If, however, the LED is connected between the output and 5 V, the LED will be off when the output is on.

2.5.1.5 Open Collector/Open Drain

Open collector, or open drain, outputs can only draw (sink) current from the circuit they are connected to. For LCC they are generally used where a node is required to switch a device connected to a higher voltage than the LCC power supply voltage, such a relay or high voltage lamp. An open collector output is normally considered to be active low, i.e., it is on when current is being sunk to pull the output voltage level towards 0 V.

Open collector outputs have two main use cases: they allow multiple outputs to be connected together; a load such as a relay can be connected between the output and a positive supply voltage that differs from the LCC supply voltage.

2.5.2 I/O Channel Configuration

Logic v. Open collector is determined by the node hardware. There may be optional components or alternative node variants that determine the type of I/O.

Whether an input signal is active low v. active high is determined by the source of the input. The node should have hardware or a configuration setting to allow either type of input to be used.

Whether an output is required to be active high or active low is determined by the circuit being driven. The node should have hardware or configuration settings to allow either type of output.

2.6 Node Configuration

Node configuration is the process of instructing a node what to do when a change of state happens on its I/O channels or in some internal state. The node must be told which events to consume and what to do in response (e.g., switch an output off). It must also be told which events to produce in response to, e.g., an input switching on.

2.6.1 The Blue/Gold Interface

The OpenLCB defines a simple user interface using two pushbuttons and two LEDs, Blue and Gold (yellow). This allows limited configuration to link (or teach) producer inputs to consumer outputs. Due to the limited utility of the blue/gold interface, it is not supported by all node manufacturers.

SPROG DCC Ltd hardware does not support the Blue/Gold interface. Instead, the Blue and Gold LEDs may be configured as status indicators for the network activity and errors. They may also be controlled by consumed events. The Blue and Gold buttons may be configured to produce events.

2.6.2 Using a Software Tool

The use of a software tool such as JMRI is assumed for node configuration. JMRI includes a number of tools for configuring and visualising a nodes operation.

2.6.2.1 The Configuration Description Information (CDI)

The CDI is information available from an LCC device, via the LCC network, that allows other devices or tools to properly and correctly configure it.

Users of DCC may be familiar with the DecoderPro software that allows DCC decoders to be programmed. DecoderPro uses a set of definition files that tell it how to program each of the many different types of decoders correctly.

The difference with LCC is that the required information is stored in the node itself. Another node, or a software tool, can retrieve the CDI from the node and know that it has the correct version of CDI for that version of the node, even if similar nodes connected to the network are different versions with different CDI.

3 Creating a CAN based LCC Network

As already stated, we can view the collection of nodes and the communication bus as the LCC network. There are additional elements and considerations that are required to create a robust, functioning network.

Nodes are normally connected in a linear “daisy-chain" fashion. If node has only one CAN connector, then a short stub cable may be used to connect it to the network. Nodes should be connected at least 12 inches, 30 cm, apart. Calculations to support the allowable cable length, stub length and node spacing are beyond the scope of this document but see the LCC Technical Note TN-9.7.1.1 on the NMRA website.

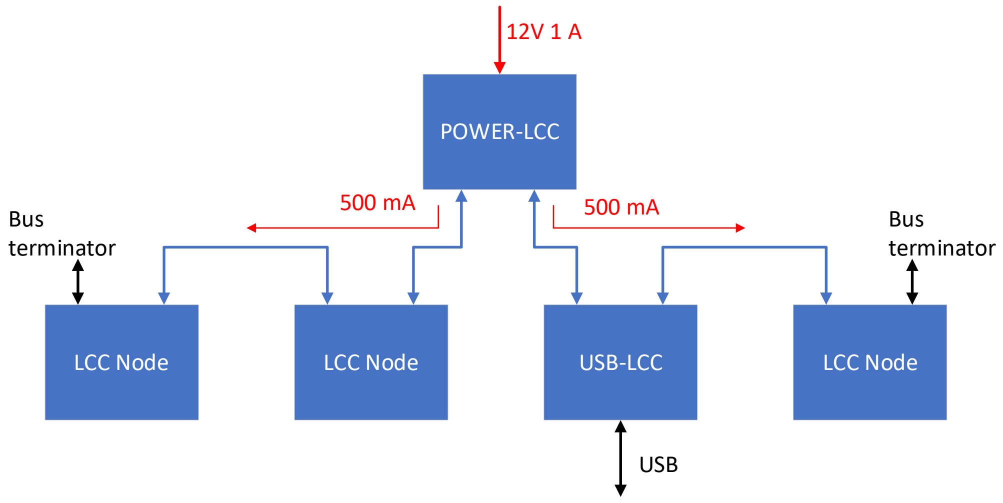


Figure 2 - LCC Network

3.1 Cabling and Connectors

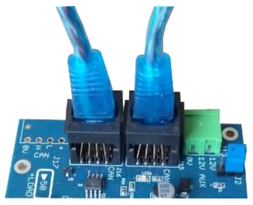


Figure 3 - Network Daisy Chaining

The CAN adaptations for LCC require the use of cables with RJ45 plugs, commonly known as "Ethernet" cables, and RJ45 jacks on the nodes. Nodes should have two RJ45 jacks (or an adapter) to allow the network to be daisy-chained, in the required linear fashion, from node to node.

The CAN adaptations for LCC also have detailed requirements for cabling and users should consult the CAN Physical Layer Specification (S-9.7.1.1) and Technical Note (TN-9.7.1.1) on the NMRA website LCC section.

Some points to note:

- The maximum length of a single segment of CAN bus cable is 1000 ft / 300m
- There should never be less than 1 ft / 30cm of bus cable between any two objects (module or stub cable) that are connected to the bus.
- The recommendations above imply that there is a practical limit of around 48 connections to the bus on a single segment.
- To satisfy the requirements of the CAN specification (ISO11898-2), a 120 ohm resistor (of at least ¼W rating) must be connected across the bus wires at both ends of the bus. See **3.3 Bus Terminators**.

3.2 Power Supplies

A limited amount of power can be distributed to nodes using the LCC cables. Nodes are not allowed to draw more than 500 mA and their maximum current draw must be shown on the node.

Nodes supplying power to the network must do so in the range 9 – 15 V. A label on the node must indicate that it supplies power to the node. Nodes must work on a voltage range of 7.5 – 15 V DC. The lower minimum voltage for operating allows for some voltage drop in the cabling.

Power may be injected into the network at multiple points to avoid large current flowing over long cables causing excessive voltage drop.

LCC Power Point modules allow easy power connection, whilst maintaining the network connection.

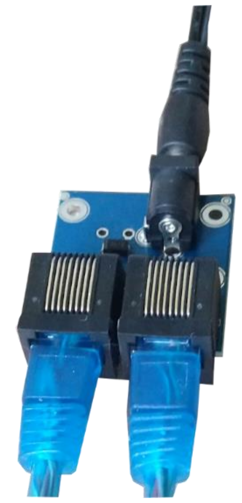


Figure 4 - Power Point

3.3 Bus Terminators

CAN requires "bus terminators" at each end of the network. This is easily achieved by plugging an LCC terminator into the spare RJ45 sockets of the nodes at each end of the network. The two ends of the network must never be connected together to form a ring.

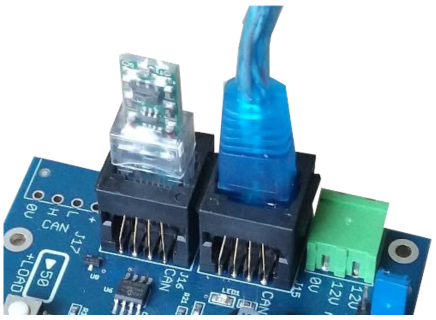


Figure 5 - Bus Termination

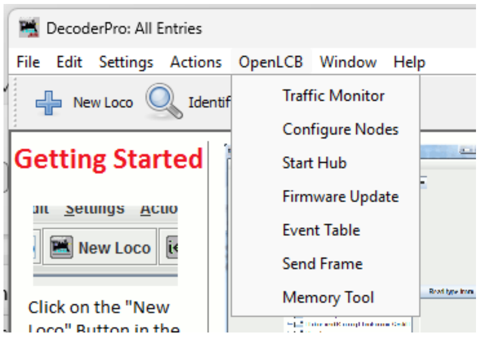
3.4 Computer Interface

A computer interface may be useful, but is not required, for layout operation. Once the LCC nodes are fully configured a computer interface is only required if the layout operation is to be controlled or monitored by computer.

Configuration with a software tool running on a computer requires an interface between e.g., USB or Ethernet and LCC. A generic interface may be used, and some are supported by JMRI, or a more LCC specific interface from an LCC manufacturer may be used, again supported by JMRI.

4 JMRI for LCC

JMRI supports layout control and operation using LCC, just as it does for many other model railway control technologies. In this chapter, however, we will look at some of the tools available for configuring and managing nodes, as shown in Figure 6 - JMRI LCC Tools which can be accessed from the OpenLCB or LCC menu (depending whether the connection preference is OpenLCB or LCC). These two connection methods are the same, in practice.

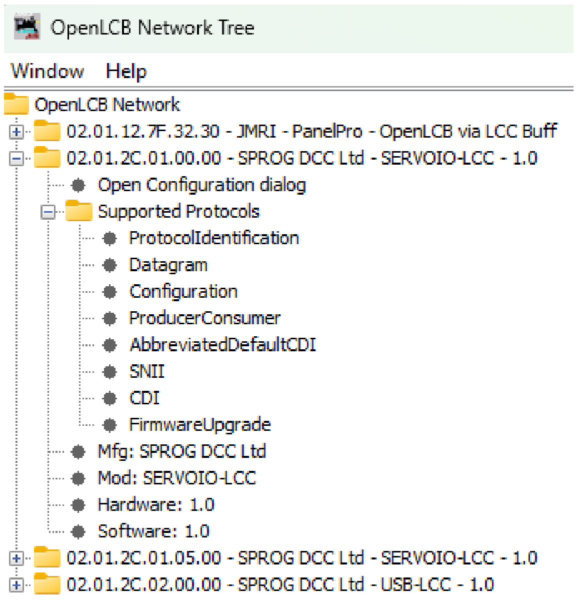


© Copyright 2024 SPROG DCC Ltd

Figure 6 - JMRI LCC Tools

4.1 Configure Nodes

The configure nodes tool is used to read the CDI from nodes, examine and/or make changes to the node configuration and write any changes back to the node. Figure 7- Configure Nodes Tool shows the network tree displayed by the configure nodes tool for a network with three physical nodes. Note that JMRI also behaves as an OpenLCB node, so the tree shows four nodes on total.



```
OpenLCB Network Tree
Window   Help
OpenLCB Network
  02.01.12.7F.32.30 - JMRI - PanelPro - OpenLCB via LCC Buff
  02.01.2C.01.00.00 - SPROG DCC Ltd - SERVOIO-LCC - 1.0
    Open Configuration dialog
    Supported Protocols
      ProtocolIdentification
      Datagram
      Configuration
      ProducerConsumer
      AbbreviatedDefaultCDI
      SNII
      CDI
      FirmwareUpgrade
    Mfg: SPROG DCC Ltd
    Mod: SERVOIO-LCC
    Hardware: 1.0
    Software: 1.0
  02.01.2C.01.05.00 - SPROG DCC Ltd - SERVOIO-LCC - 1.0
  02.01.2C.02.00.00 - SPROG DCC Ltd - USB-LCC - 1.0
```

Figure 7- Configure Nodes Tool

Each node branch includes a unique numeric sequence, e.g., 02.01.2C.01.00.00. This is the Node ID presented as 6 bytes in hexadecimal notation, with periods to separate the bytes. The prefix 02.01 indicates nodes identified by their NMRA assigned manufacturer IDs, 0x12 (18) for JMRI and 0x2C (44) for SPROG DCC Ltd, in the next byte. The three remaining bytes are manufacturer assigned. In this example they are a combination of node type and a unique node serial number, so 01.00.00 and 01.05.00 are the same node type but 02.00.00 is a different node type.

Some basic read only node information is displayed in the network tree, such as manufacturer and version strings.

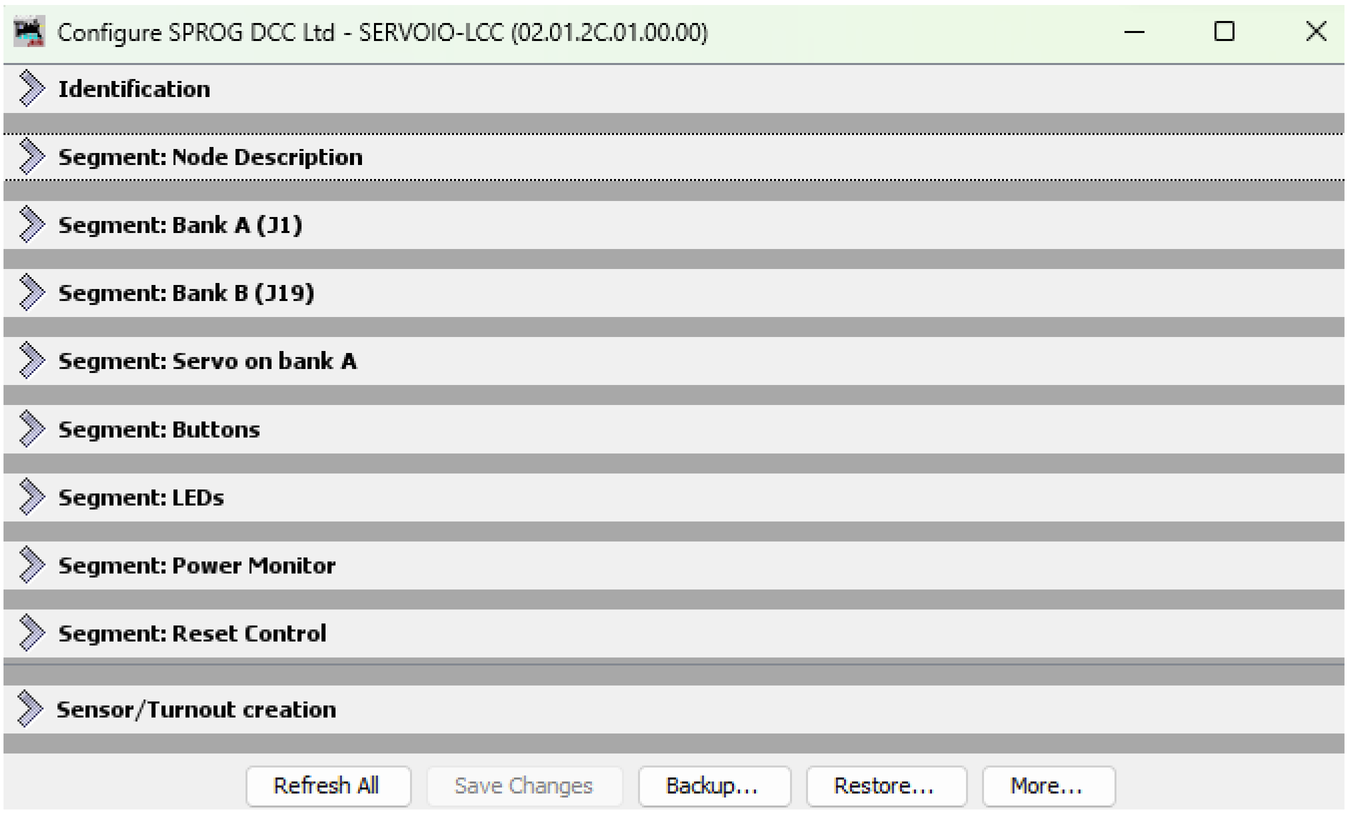
The display for the second node in the list is shown expanded. A discussion of the supported protocols branch is mostly beyond the scope of this document, except to note that the node supports ProducerConsumer (events) and FirmwareUpgrade (see 4.4 Firmware Upgrade). Each protocol is supported by a standard document and/or a technical note – see the NMRA website.

Clicking the open Configuration dialog leaf opens the Configure dialog, shown in Figure 8 - Configure Node.

Initially, all “segments” are collapsed, but can be expanded by clicking the arrows on the left.

The identification and Node Description segments are common to all SPROG DCC Ltd nodes.

© Copyright 2024 SPROG DCC Ltd


Configure SPROG DCC Ltd - SERVOIO-LCC (02.01.2C.01.00.00)

Identification

Segment: Node Description

Segment: Bank A (J1)

Segment: Bank B (J19)

Segment: Servo on bank A

Segment: Buttons

Segment: LEDs

Segment: Power Monitor

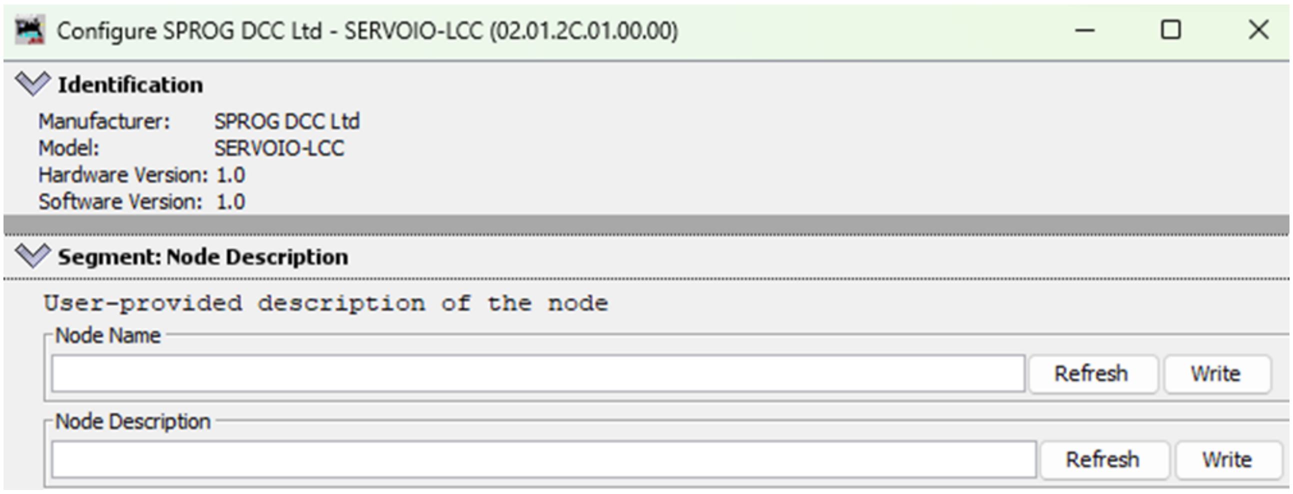
Segment: Reset Control

Sensor/Turnout creation

Figure 8 - Configure Node

The identification segment repeats the basic node information, see Figure 9 - Basic Segments.

The Node Description segment allows a user-friendly name and description to be assigned to the node. This information will be stored in the node when the individual write button for each text box is clicked.


Configure SPROG DCC Ltd - SERVOIO-LCC (02.01.2C.01.00.00)

Identification

Manufacturer:	SPROG DCC Ltd
Model:	SERVOIO-LCC
Hardware Version:	1.0
Software Version:	1.0

Segment: Node Description

User-provided description of the node

Node Name

Node Description

Figure 9 - Basic Segments

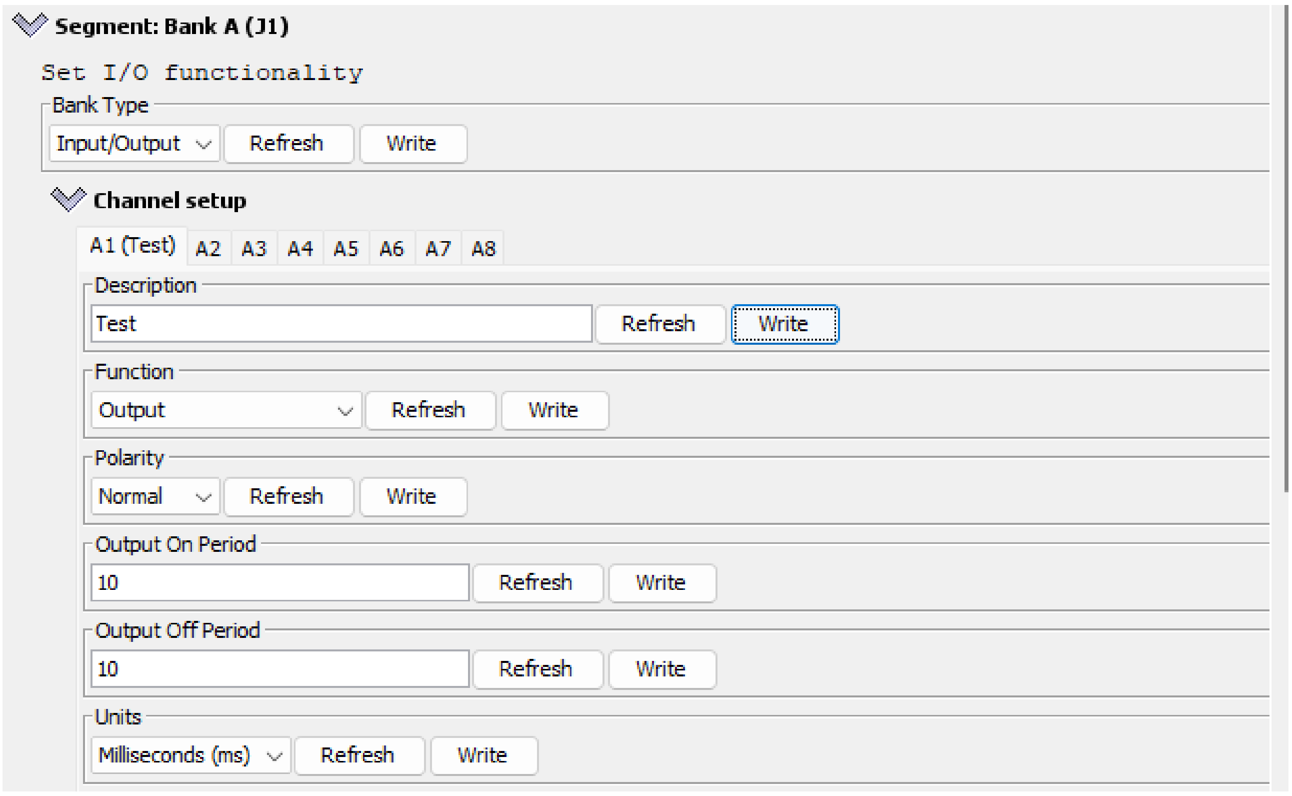
The Save Changes button at the bottom of the configure dialog will write all outstanding changes to the node. If an entry is edited but not written, the original contents can be re-read from the node using the Refresh button.

The remaining segments, apart from Sensor/turnout creation are node specific and will be described in the node manufacturer's document. An example, below, gives an indication of what is possible.

The node shown in the configuration example is an I/O node with multiple channels, see Figure 10 - Setting I/O Functionality. In this case the channels have been arranged in two banks of eight, each with their own segment. As noted previously, the details will be different in other nodes.

Bank A in this node has eight channels, A1 – A8. Channel A1 has been given a user-friendly name "Test" by entering the description and writing it to the node.

Each channel has a number of selectors for Description, Function, Polarity and timed periods for pulsed outputs. Again, the details will vary between node types.



Segment: Bank A (J1)

Set I/O functionality

Bank Type: Input/Output

Channel setup

A1 (Test) A2 A3 A4 A5 A6 A7 A8

Description: Test

Function: Output

Polarity: Normal

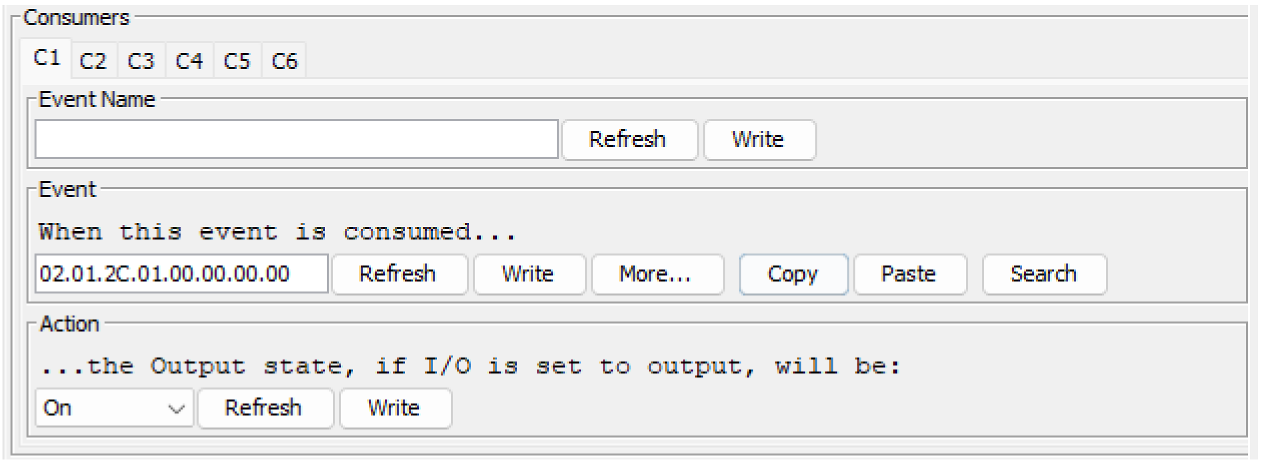
Output On Period: 10

Output Off Period: 10

Units: Milliseconds (ms)

Figure 10 - Setting I/O Functionality

Scrolling down, we come to the Consumers group. In our example each channel can consume up to six events C1 – C6. These can be named in a similar way that channels can be named. In this example, channel A1 consumer C1 (A1C1) is pre-populated with default event 00.00 appended to the Node ID. The action performed when the event is consumed is to turn the output on.


Consumers

C1 C2 C3 C4 C5 C6

Event Name

Event

When this event is consumed...

02.01.2C.01.00.00.00.00

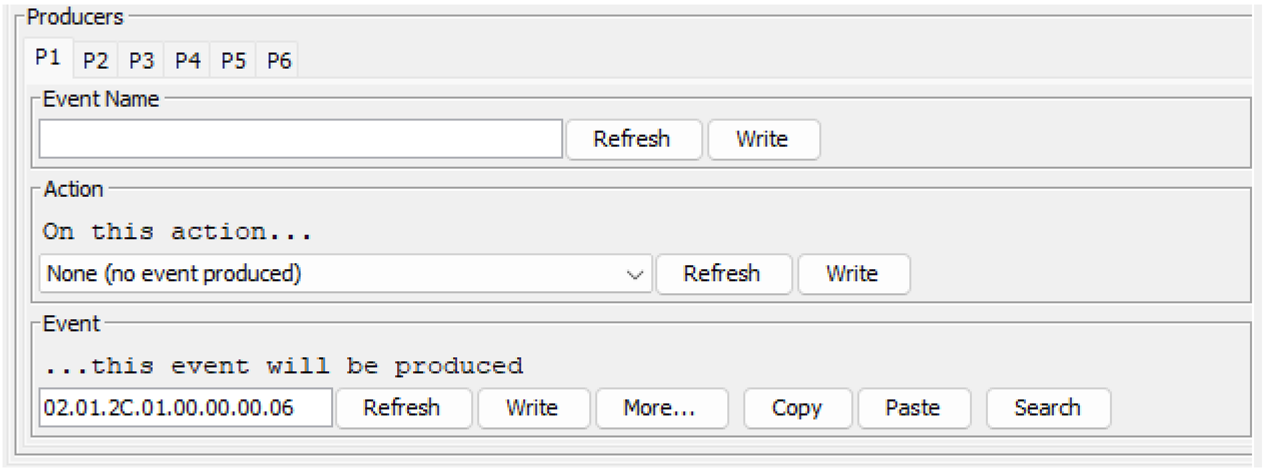
Action

...the Output state, if I/O is set to output, will be:

On

Figure 11 - Consumers

Each channel in our example can produce up to six event P1 – P6. A1 P1 is pre-populated with event 00.06. The action that causes the event to be produced will be node specific. In this example, no action has yet been selected and event P1 will not be produced.


Producers

P1 P2 P3 P4 P5 P6

Event Name

Action

On this action...

None (no event produced)

Event

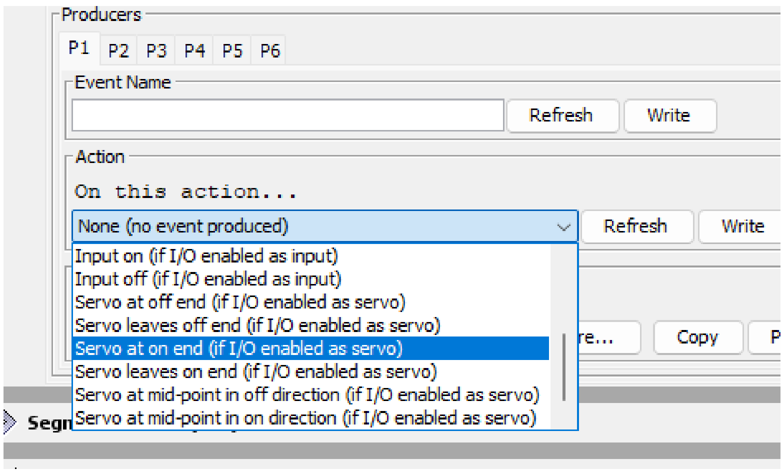
...this event will be produced

02.01.2C.01.00.00.00.06

Figure 12 - Producers

To tie producers and consumers together, the Copy and Paste buttons allow events to copied from, say a producer to a consumer on the same or a different node. This might be referred to as event teaching. A consumer is taught to act on an event from a producer by copying the event from the producer to the consumer. Alternatively, an arbitrary event, according to the user’s own event numbering scheme, may be entered into both the consumer and producer.

A closer look at the actions for producer events Figure 13 - Producer Event Actions shows that events may be produced from and internal change of state. In this example events can be produced based on servo movement.



On this action...

None (no event produced)

- Input on (if I/O enabled as input)
- Input off (if I/O enabled as input)
- Servo at off end (if I/O enabled as servo)
- Servo leaves off end (if I/O enabled as servo)
- Servo at on end (if I/O enabled as servo)
- Servo leaves on end (if I/O enabled as servo)
- Servo at mid-point in off direction (if I/O enabled as servo)
- Servo at mid-point in on direction (if I/O enabled as servo)

Figure 13 - Producer Event Actions

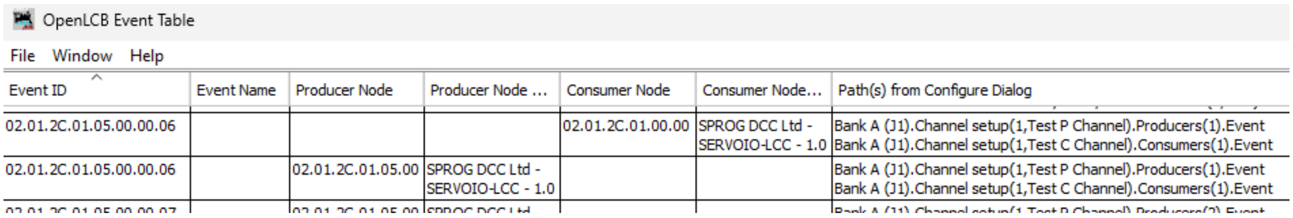
4.2 The Event Table

The Event Table displays the events created on the network, along with the related producers and consumers. It may be exported as a .csv file to create a document recording the events.

The example in Figure 14 - Event Table shows that event 02.01.2C.01.05.00.00.06 has been copied from the producer node 02.01.2C.01.05.00 channel A1, Producer 1 to the consumer node 02.01.2C.01.00.00 channel A1 Consumer 1.

The channel names were set in the node configuration and are displayed in the Paths column.

The Event Name column is a user entered field; it is not read from the node's CDI Event Name.

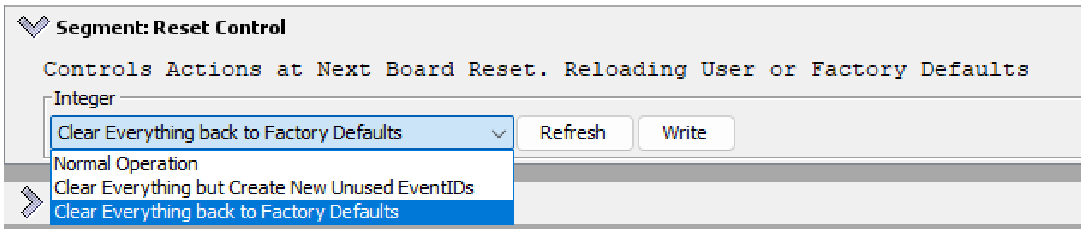


Event ID	Event Name	Producer Node	Producer Node ...	Consumer Node	Consumer Node...	Path(s) from Configure Dialog
02.01.2C.01.05.00.00.06				02.01.2C.01.00.00	SPROG DCC Ltd - SERVOIO-LCC - 1.0	Bank A (J1).Channel setup(1,Test P Channel).Producers(1).Event Bank A (J1).Channel setup(1,Test C Channel).Consumers(1).Event
02.01.2C.01.05.00.00.06		02.01.2C.01.05.00	SPROG DCC Ltd - SERVOIO-LCC - 1.0			Bank A (J1).Channel setup(1,Test P Channel).Producers(1).Event Bank A (J1).Channel setup(1,Test C Channel).Consumers(1).Event

Figure 14 - Event Table

4.3 Node Reset

The reset control segment, if present allows various levels of reset, allowing, e.g., a factory reset of the node.



Controls Actions at Next Board Reset. Reloading User or Factory Defaults

Clear Everything back to Factory Defaults

- Normal Operation
- Clear Everything but Create New Unused EventIDs
- Clear Everything back to Factory Defaults

Figure 15 - Reset Control

© Copyright 2024 SPROG DCC Ltd

4.4 Firmware Upgrade

The LCC specifications include a standardised way to use the existing LCC protocols for upgrading a node's firmware. Firmware programming files are sent over the network using the Datagram or Streaming protocols. The programming files are manufacturer (and node) specific. Contact node manufacturers for details of firmware updates.

4.5 Layout Control

The usual JMRI layout control concepts such as panel, turnouts, sensors, etc., are all supported when connected to an LCC network, just as they are with any other layout control bus.

5 Links

SPROG DCC Ltd website https://www.sprog-dcc.co.uk For all our products and support.

SPROG DCC Ltd Official YouTube Channel https://www.youtube.com/@sprogdcc

OpenLCB group https://openlcb.org The group behind the OpenLCB/LCC standards.

NMRA LCC standards page https://www.nmra.org/lcc The LCC standards adopted by the NMRA.

OpenLCB discussion group https://groups.io/g/openlcb/topics Discussion of OpenLCB topics, more developer focussed.

The NMRA's LCC user group https://groups.io/g/layoutcommandcontrol/topics a good starting point for asking questions of other LCC users.

JMRI users https://groups.io/g/jmriusers/topics JMRI software topics.

JMRI website https://www.jmri.org Download the latest JMRI releases and access support pages.

Book: Introduction to Layout Command Control https://www.amazon.co.uk/Introduction-Layout-Command-Control-Practical/dp/0988825902 focussed on RR-Cirkits products but the concepts are applicable to any LCC hardware.