# Using the R4_OlcbBasicNode Sketch

## Initial Setup

You will need to obtain the OpenLCB Single Thread library code at:
   https://github.com/openlcb/OpenLCB_Single_Thread
The usual ways should work:
   from the Code dropdown:
   - Download the zip file and use the Arduino IDE's library manager to install it;
   - Hand install it by downloading the zip and unzipping it into your Arduino libraries directory
   - Use git to clone a copy into your Arduino libraries directory:
      git clone https://github.com/openlcb/OpenLCB_Single_Thread.git

Then copy this sketch to your sketch folder.

## Using the Sketch

There are three ways, at least, to run this:
1. Using GCSerial and DEBUG on the Arduino monitor. This will show the debug output, and the message outputs, and you can send LCC messages to the node.
2. Using GCSerial WITHOUT DEBUG, and connecting to JMRI directly through USB.
   This creates a one node network with JMRI. It lets you open the CDI, configure the node and send it commands, and use JMRI panels, etc.
3. Using the R4 CAN driver and DEBUG, but WITHOUT GSerial.
   This requires a CaN transceiver and a SerialCAN connection for your computer.
   The DEBUG output will appear on the Arduino monitor.
   LCC traffic can be followed on the JMRI LCC Traffic monitor.

### Node Identification

You can use the nodeID a given:
```
#define NODE_ADDRESS 2,1,13,0,0,0x08 // DIY range example, not for global use.
```
but you should change it to a nodeID that you control. You can use your MERG membership ID to control 256 nodeIDs, and /or obtain a personal range from
   https://registry.openlcb.org/requestuniqueidrange

### Node Initialization

The node is told to initialize itself to manufacturer's defaults by setting
```
#define RESET_TO_FACTORY_DEFAULTS 1
```
However, if you want the node to retain any configuration changes you make to teh node, then you will need to set this to 0, and reload the firmware, otherwise the node will be reinitialized and the changes will be lost.

# Local testing:

1/ uncomment the `#define DEBUG Serial` line.

2/ uncomment the `#include "GCSerial.h"` line.

3/ comment out the //`#include "R4.h"` line.

4/ Compile the sketch. You should see:

      `#pragma message "Direct to Serial selected (GCSerial)"`
      telling you the sketch is using Gridconnect to Serial.

```
Output   Serial Monitor                                                                        ≡
  In file included from /Users/daveharris/Documents/Arduino/R4_OlcbBasicNode/R4_OlcbBasicNode.ino:20:0:
  /Users/daveharris/Documents/Arduino/libraries/OpenLCB_Single_Thread/src/GCSerial.h:48:17: note: #pragma message: Direct to
  #pragma message "Direct to Serial selected (GCSerial)"
  | | | | | | | |^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Sketch uses 64836 bytes (24%) of program storage space. Maximum is 262144 bytes.
  Global variables use 5120 bytes (15%) of dynamic memory, leaving 27648 bytes for local variables. Maximum is 32768 bytes.
```

5/ Download the sketch to your Uno Minima board.

      You should see:
      OlcbBasicNode
      A dump of the eeprom
      A listing of the node's events

```
Output   Serial Monitor ×

Message (Enter to send message to 'Arduino UNO R4 Minima' on '/dev/cu.usbmodem101')

userInitAll()
printEvents
#  flags  EventID
0:40 : 41 :  02.01.0D.00.00.08.00.00
1:48 : 41 :  02.01.0D.00.00.08.00.01
2:60 : 41 :  02.01.0D.00.00.08.00.02
3:68 : 41 :  02.01.0D.00.00.08.00.03
4:80 : 21 :  02.01.0D.00.00.08.00.04
5:88 : 21 :  02.01.0D.00.00.08.00.05
6:A0 : 21 :  02.01.0D.00.00.08.00.06
7:A8 : 21 :  02.01.0D.00.00.08.00.07
Sorted events
Event Index: 0  EventNum: 0  EventID: 02.01.0D.00.00.08.00.00  Flags: 41
Event Index: 1  EventNum: 1  EventID: 02.01.0D.00.00.08.00.01  Flags: 41
Event Index: 2  EventNum: 2  EventID: 02.01.0D.00.00.08.00.02  Flags: 41
Event Index: 3  EventNum: 3  EventID: 02.01.0D.00.00.08.00.03  Flags: 41
Event Index: 4  EventNum: 4  EventID: 02.01.0D.00.00.08.00.04  Flags: 21
Event Index: 5  EventNum: 5  EventID: 02.01.0D.00.00.08.00.05  Flags: 21
Event Index: 6  EventNum: 6  EventID: 02.01.0D.00.00.08.00.06  Flags: 21
Event Index: 7  EventNum: 7  EventID: 02.01.0D.00.00.08.00.07  Flags: 21
EEPROM: Base Offset: 0 User Base: 8 User Size: 176
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F 0123456789ABCDEF
00 4E 02 01 0D 00 00 08 B7 08 00 EE 00 4F 6C 63 62 N..........Olcb
10 42 61 73 69 63 4E 6F 64 65 00 00 00 00 00 00 00 BasicNode.......
20 54 65 73 74 69 6E 67 00 00 00 00 00 00 00 00 00 Testing........
30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
40 00 00 00 00 00 00 00 00 02 01 0D 00 00 08 00 00 ................
50 02 01 0D 00 00 08 00 01 00 00 00 00 00 00 00 00 ................
60 00 00 00 00 00 00 00 00 02 01 0D 00 00 08 00 02 ................
70 02 01 0D 00 00 08 00 03 00 00 00 00 00 00 00 00 ................
80 00 00 00 00 00 00 00 00 02 01 0D 00 00 08 00 04 ................
90 02 01 0D 00 00 08 00 05 00 00 00 00 00 00 00 00 ................
A0 00 00 00 00 00 00 00 00 02 01 0D 00 00 08 00 06 ................
B0 02 01 0D 00 00 08 00 07 FF FF FF FF FF FF FF FF ................
```

      And a series of LCC message in this form:

      `>>>:X195B4345N02010D0000080000;`
      this is the Gridconnect format.

```
>>>:X17020125N;
 produce true
>>>:X1610D125N;
>>>:X15000125N;
>>>:X14008125N;
 produce true
>>>:X10700125N;
>>>:X10701125N02010D000008;
>>>:X19100125N02010D000008;
>>>:X19547125N02010D0000080000;
>>>:X195B4125N02010D0000080000;
>>>:X19547125N02010D0000080001;
>>>:X19547125N02010D0000080002;
>>>:X195B4125N02010D0000080002;
>>>:X19547125N02010D0000080003;
>>>:X194C7125N02010D0000080004;
>>>:X194C7125N02010D0000080005;
>>>:X194C7125N02010D0000080006;
>>>:X194C7125N02010D0000080007;
```

6/ After it stops printing things, try typing this into the Serial input line:
**:X194B4123N02010D0000080004;**
> The debug output will show that the node is consuming these and taking action.
> On the Arduino monitor you should see:

```
<<<:X195B4123N02010D0000080004;
OpenLcbCore::processEvent: Index4
 In pceCallback index=4
```

> and the LED on the board should light.
> Now type:
**:X194B4123N02010D0000080004;**
> and the LED should should turn off.

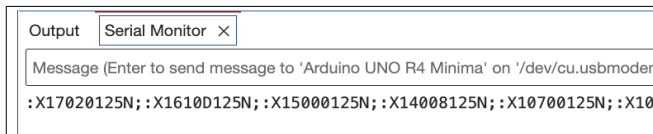7/ Try grounding pin 2 or 3, and ungrounding them, and you should see one or more of:
**>>>:X195B4xxxN02010D00080000;**
**>>>:X195B4xxxN02010D00080000;**
**>>>:X195B4xxxN02010D00080000;**
**>>>:X195B4xxxN02010D00080000;**

```
 produce false
>>>:X195B4125N02010D0000080001;
 produce true
>>>:X195B4125N02010D0000080000;
```
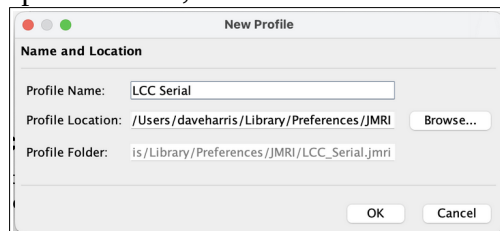
# Local connection to JMRI through USB

1/ Uncomment out **#include "GCSerial.h"** and comment **//#define DEBUG Serial**
  and download the sketch to your board.  The Serial monitor should show some LCC frames:

Output    Serial Monitor ✕

Message (Enter to send message to 'Arduino UNO R4 Minima' on '/dev/cu.usbmoder

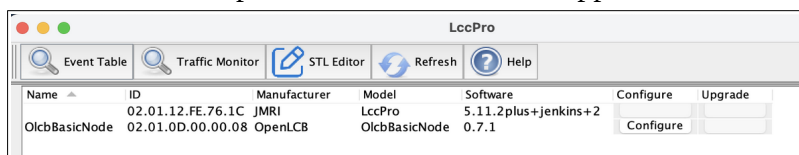:X17020125N;:X1610D125N;:X15000125N;:X14008125N;:X10700125N;:X10

IMPORTANT: Close the Serial window, otherwise it will conflict with JMRI.

2/ Open LCCPro, and click New to make a new JMRI connection, call it "LCC Serial", and click OK:

New Profile

**Name and Location**

Profile Name:    LCC Serial

Profile Location:  /Users/daveharris/Library/Preferences/JMRI    Browse...

Profile Folder:   is/Library/Preferences/JMRI/LCC_Serial.jmri

OK    Cancel

Choose it in the Set Active Profile window, and in the resulting Wizard click OK.  Answer the questions. For the System Connection choose  CAN via Gridconnect. Choose a serial port. IN Additional Connection Settings, set the baud rate to 115200. Choose Next and Finish.

3/  LCCPro should open, and the node should appear in the list of nodes.

LccPro

Event Table    Traffic Monitor    STL Editor    Refresh    Help

| Name | ID | Manufacturer | Model | Software | Configure | Upgrade |
|---|---|---|---|---|---|---|
| | 02.01.12.FE.76.1C | JMRI | LccPro | 5.11.2plus+jenkins+2 | | |
| OlcbBasicNode | 02.01.0D.00.00.08 | OpenLCB | OlcbBasicNode | 0.7.1 | Configure | |

4/ Push the Traffic Monitor button and resize the window that opens smaller.
5/ Push the Configure button on the node's line, and a Configure Node window will open, and start loading the node's CDI. The message traffic will appear in the Traffic window.
6/ The CDI should open. Clink on the >Config line.
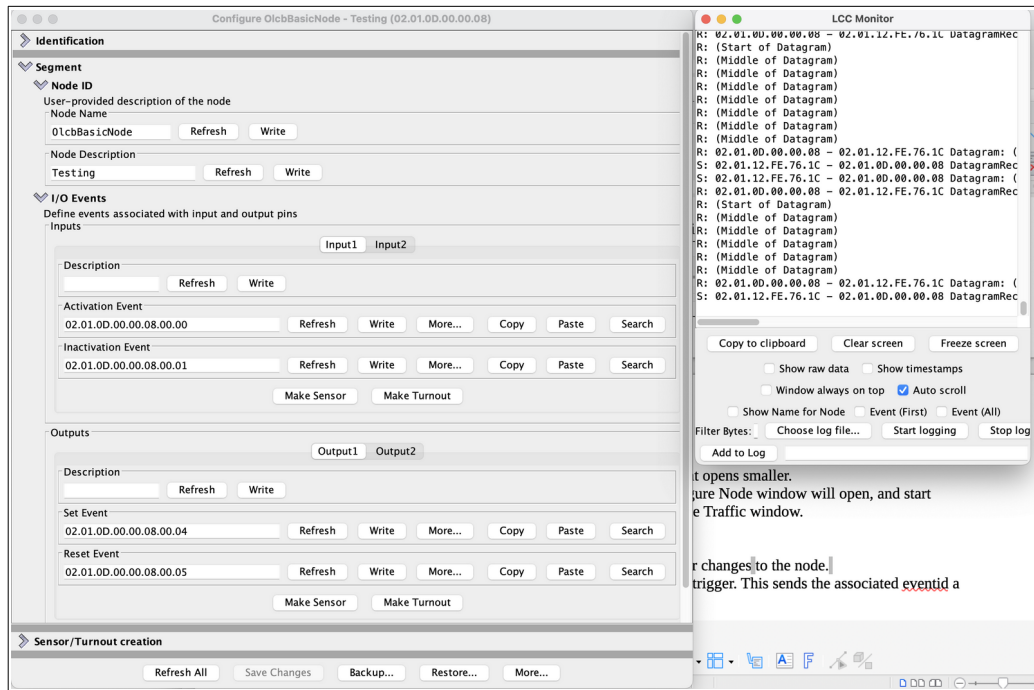


7/ Play with the settings. You need to push Save to write your changes to the node.
8/ One Output 1, click and hold the More button, and choose trigger. This sends the associated eventid a PCER message. The Traffic monitor should show an event, and the LED should turn on.
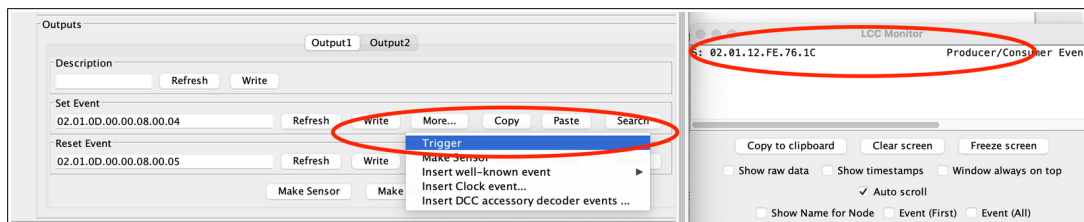


Triggering the off event should turn off the LED.

# Using the CAN network

1/ Uncomment `#include "R4h"` and `#define DEBUG Serial`, and comment out `//#include "GCSerial.h"`
   Recompile, download to the Minima. You can reopen the Arduino Serial Monitor.
2/ Connect a CAN transceiver Tx to pin 4 and Rx to pin 5, and ground. Connect to your CAN bus.
3/ Open LCCPro and make a new profile: LCC, choose the port connected to your Serial-CAN device, and click OK.
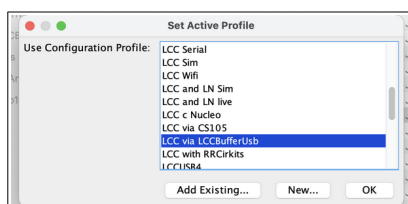
4/ Re-open LCCPro:
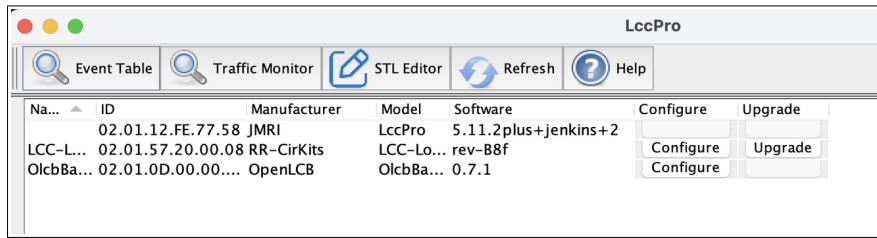


5/ Play.

# Hints:

Remember the set the following lines correctly:

```
#define DEBUG Serial
#include "GCSerial.h"
#include "R4.h"
#define NODE_ADDRESS 2,1,13,0,0,0x08
#define RESET_TO_FACTORY_DEFAULTS 1
```

The RESET line will bite you if you forget. If the node is acting weirdly, then set this to 1, download the sketch to the board, and then set it to 0 and download the sketch again.