cpNode System Technical Guide

Model Railroad Control Systems, LLP

Version 2.1 09/14/2016

Introduction

The cpNode system takes the leading model railroad layout control bus, CMRInet, which has been used by thousands of model railroaders for 30 years. The cpNode provides a communications and control framework to model prototypical signaling systems as well as other layout automation functions, and extends CMRInet into smaller configurations. A cpNode can provide real time control over devices such as servos and integrated analog/digital conversion. cpNodes can be used to extend existing CMRInet installations and for new model railroads.

The cpNode system provides an economic, simple, durable, and capable contemporary solution for making model railroads operate like prototypes. With it, signals can be controlled by block occupancy and turnout position information as in prototypical signaling systems. Track diagrams (model boards) can be created with hard control panels or computer displays, providing interlocking tower or CTC machine views of the layout. Operators can control model operations from remote towers or local fascia control panels.

Note: This version of the System Technical Guide is for system board versions v1.6 or later.

Model Railroad Control Systems provides complete instructions for the creation of your railroad's operating scheme, which can reflect these prototypical systems. Online information and consulting resources are available if you need them.

Model Railroad Control Systems
www.modelrailroadcontrolsystems.com

Chuck Catania, *chuck@modelrailroadcontrolsystems.com*
Seth Neumann, *seth@modelrailroadcontrolsystems.com*

Table of Contents

1. cpNode Hardware Environment 3
2. cpNode Model Railroad Layout Connection Example 5
3. cpNode Electrical Requirements 6
4. Electrical Ratings (DC) 6
5. arduino™ Integrated Development Environment (IDE) 6
5.1. Setting up the Software Environment 6
5.2. Loading Sketches Into the BBLeo/arduino Board 7
6. Connecting a cpNode to CMRInet (RS-422/RS-485) 9
7. Setting Node Configuration Parameters 13
7.1. Kernel Sketch Node Configuration Section 13
7.2. CMRInet Node Addressing 13
7.3. Cmrinet (Rs-422/485) Line Speed 14
8. Configuring The Kernel Sketch Standard I/O Maps 14
9. Standard Input/Output Port Map Configurations 16
9.1. #define base_Node 16
9.2. #define base_node_8in8out 16
9.3. #define base_node_8out8in 17
9.4. #DEFINE BASE_NODE_12OUT4IN 17
9.5. #define base_node_16in 18
9.6. #define base_node_16out 18
9.7. #define base_node_rsmc 19
9.8. #define base_node_rsmc_LOCK 20
10. Configuring a cpnode using JMRI 20
11. Input/Output Expander (IOX) Port Map Configurations 21
11.1. IOX Interconnection 22
11.2. Setting IOX Board Addresses 22
11.3. Input/Output Extender Port Maps 24
11.4. IOX Port Assignment Sketch Code 25
11.5. IOX Port Assignment Example 26
11.6. IOX32 Port Assignment Example 27
12. cpNode CMRInet Line Speed Selection 28
13. LED Connections - Nommon Anode/Common Cathode 28
13.1. Wiring Common Anode/Common Cathode LEDs 29
14. BBLeo to cpnode Header Pin Location 31
15. Resources For CMRInet, CPNODE, ARDUINO, JMRI 32

Table of Figures

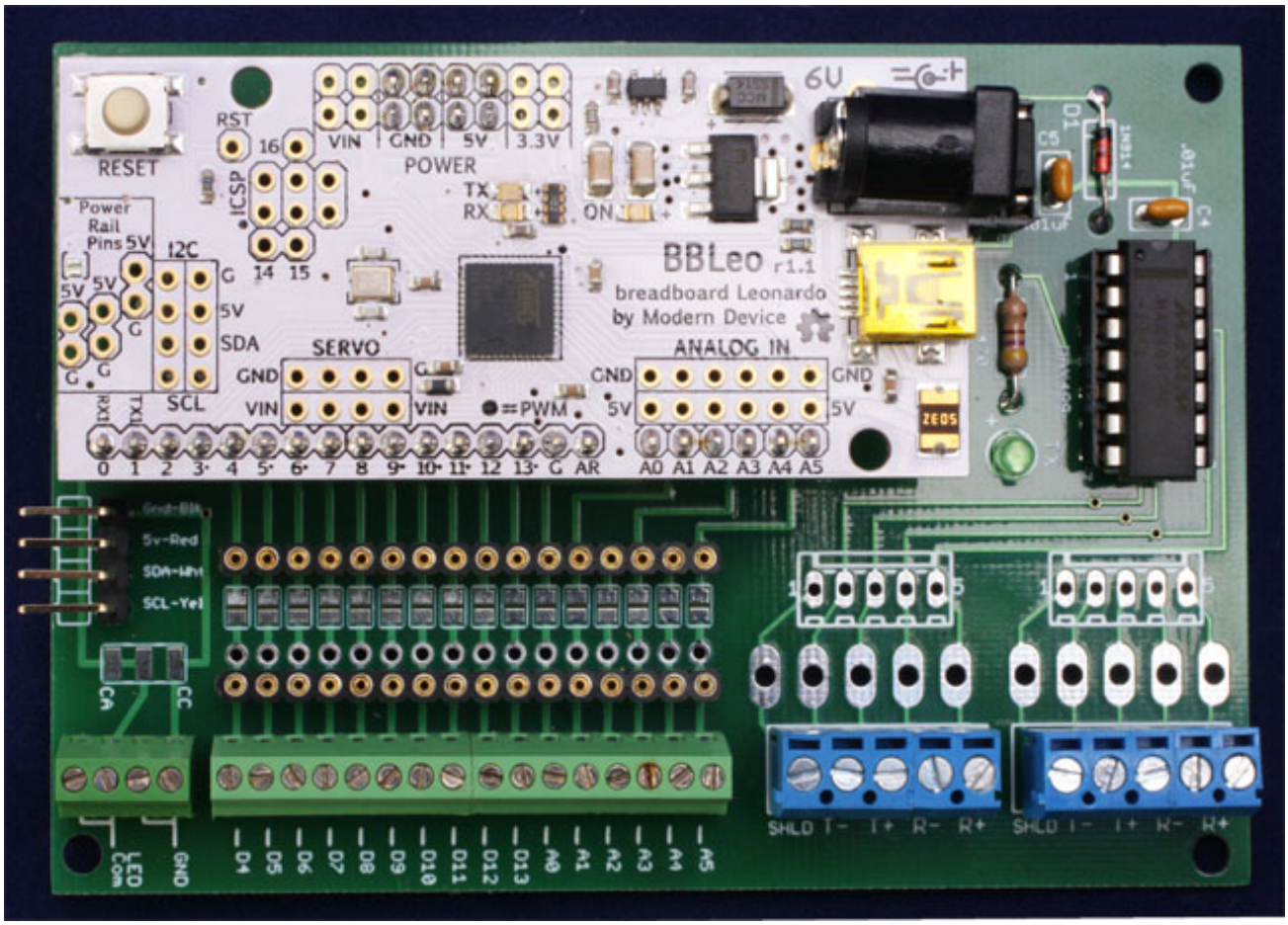
Figure 1 cpNode Functional Sections 4
Figure 2 Example - cpNode Model Railroad Layout Connection 5
Figure 3 cpNode Electrical Specifications 6
Figure 4 CMRInet Network 9
Figure 5 3.5 mm Screw Terminal Connection 10
Figure 6 .156" MOLEX Male Header Connection 11
Figure 7 .100" (2.54 mm) Molex KK Connection 12
Figure 8 cpNode Input/Output Port Map 14
Figure 9 cpNode IO Configuration Selection 15
Figure 10 I2c Interconnet Cable 22
Figure 11 I/O Expander Board Address Jumpers 23
Figure 12 I/O Expander (IOX) Port Maps 24
Figure 13 I/O Expander Port Enable Define 25
Figure 14 IOX Port Direction Assignment Map 25
Figure 15 Example IOX Port Direction Assignment Map 25
Figure 16 JMRI Device Table Name Example IOX16 26
Figure 17 JMRI Device Table Name Example IOX32 27

FIGURE 18 CMRINET LINE SPEED SELECTION 28
FIGURE 19 LED COMMON OPTION SOLDER PAD AND CONNECTION HEADER 29
FIGURE 20 LED COMMON ANODE/COMMON CATHODE CONNECTION DIAGRAM 30
FIGURE 21 BBLEO/CPNODE CONNECTION HEADER PINS 31

1. CPNODE Hardware Environment

The cpNode motherboard consists of five functional sections.

- Arduino (BBLeo) processor board, header connectors for I/O ports, and system power.
- RS-422/485) line driver, automatic transmit enable circuit (AutoRTS), network cable terminal pads.
- Input/output port configuration area and connection pads.
- Input/output extender (IOX) card interface (I2C serial interface).
- Optional remote stall motor controller (RSMC).



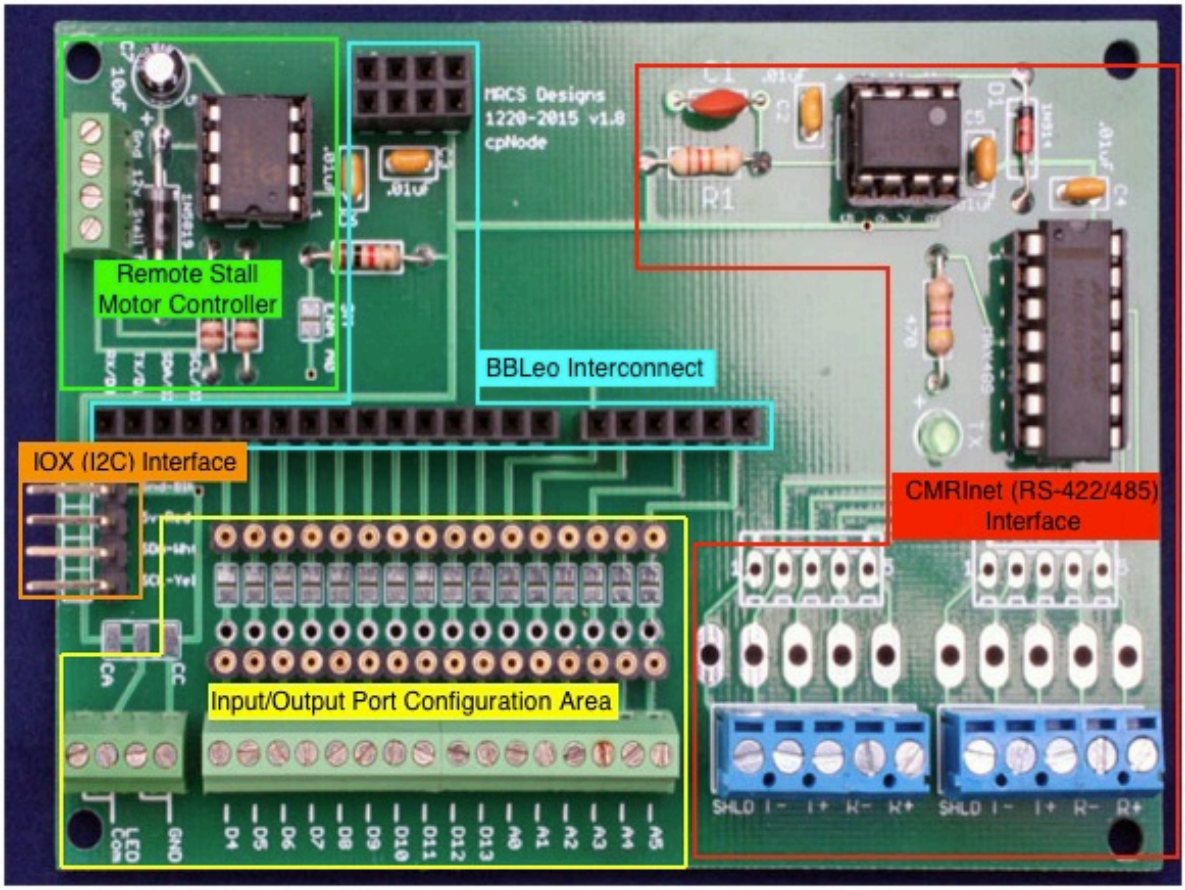


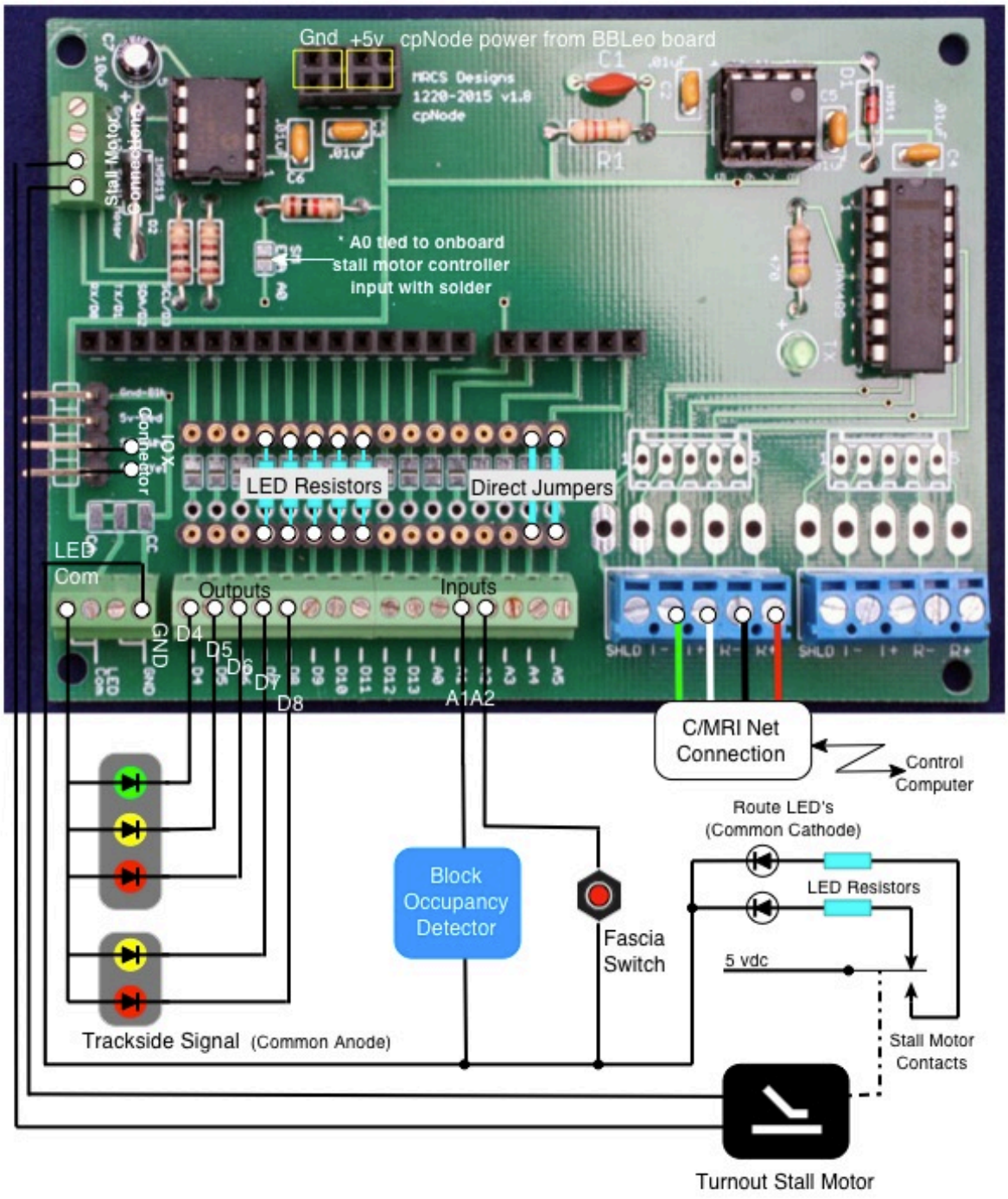
Figure 1 cpNode Functional Sections

The cpNode design is focused on maximum flexibility for connecting devices (e.g. LEDs, sensors, switches, push buttons, servos, etc.) to an Arduino microprocessor system. All connection points are designed with .100" pad spacing to accept a wide variety of connectors and headers. Direct soldering of wires to the pads can also be done, but is not as flexible. Screw terminal blocks are preferred, but are the most expensive option. Male header pin strips are convenient and inexpensive, but require a mating header with crimp pins.

Most of the cost in these systems is in the connectors, so carefully consider your requirements and the resulting costs before choosing a connector type:

- Cost per line
- Does the connector require special tooling? What are the costs?
- Do you need connectors for rapid replacement or to "patch around" un-needed units?
- Is the connector/terminal convenient to use in the place you've got it mounted (e.g. upside down under the layout)?

2. CPNODE Model Railroad Layout Connection Example



3. CPNODE ELECTRICAL REQUIREMENTS

Arduino boards can accept input supply voltages up to 12 Vdc and have onboard power regulation, which supply 5 Vdc to the system. See the individual Arduino processor boards for exact details.

The cpNode system power required is **5-6 Vdc**, at a minimum of **1 Amp**. Standard wall power supplies (wall warts) with sufficient capacity, can be used in the system. A standard 2.1 mm, center positive, power connector is used on the Arduino processor boards.

Arduino ports configured as OUTPUT can supply 40 ma at 5 volts, however, the MAXIMUM current, which can be supplied by the Arduino, is 160 ma. Limiting the current for LEDs to 5ma using a 680 ohm resistor per output would keep the total current within the specification,

4. ELECTRICAL RATINGS (DC)

System Input Voltage	6-12 Vdc
System Input Voltage (limits)	5-12 Vdc
cpNode Operating Voltage	**5-6 Vdc**
cpNode Minimum power supply current	**1A**
I/O pin Maximum voltage	**5 Vdc**
Maximum current per I/O pin	40 mA, sourcing or sinking
Maximum I/O port current (total)	**160 mA**
Optimum LED drive current per pin	5-10 mA

Figure 3 cpNode Electrical Specifications

5. ARDUINO™ Integrated Development Environment (IDE)

Software, which runs in the cpNode, is created using the Arduino Integrated Development Environment (IDE). The software is known as a "sketch", named after the original use of the IDE at the Interaction Design Institute in Italy.

The IDE provides software development tools and libraries used in the creation of software. This application software is free and runs on Macintosh, Windows, and Unix. You need to use version 1.0.5 or greater.

Kernel Sketch version 1.5 or later is compatible with all released versions of the Arduino IDE. Note: IDE version 1.5.6-r2 is recommended for Kernel sketch versions earlier than 1.5.

5.1. Setting up the Software Environment

The IDE software is located on the Arduino web site http://arduino.cc/en/Main/Software There are versions for Windows, Macintosh OS X, and Linux. Download the version for the operating system you are using.

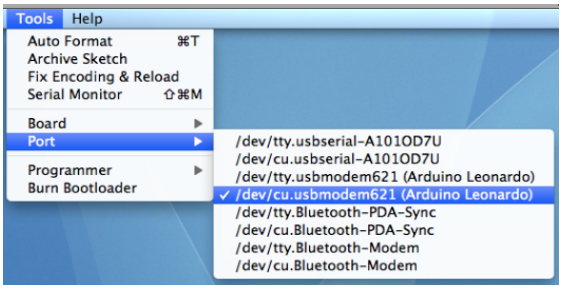
Note: The cpNode uses the Modern Devices BBLeo as the Arduino. The BBLeo is a Leonardo configuration within the supported Arduino hardware family.

5.2. Loading Sketches Into the BBLEO/ARDUINO Board

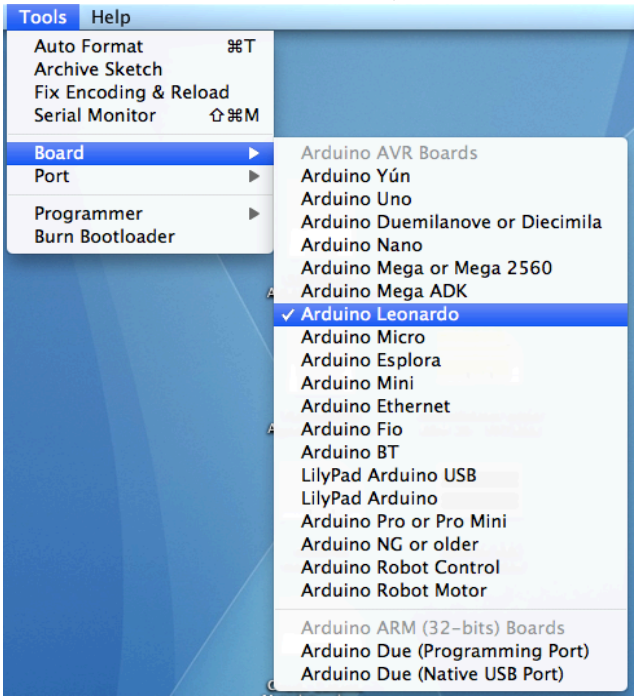
Sketches are uploaded into the Arduino through a USB port assigned in the IDE under the Tools menu. To load a sketch into the BBLeo:

1. Connect the BBLeo to the computer system, which runs the IDE using a mini-USB cable.

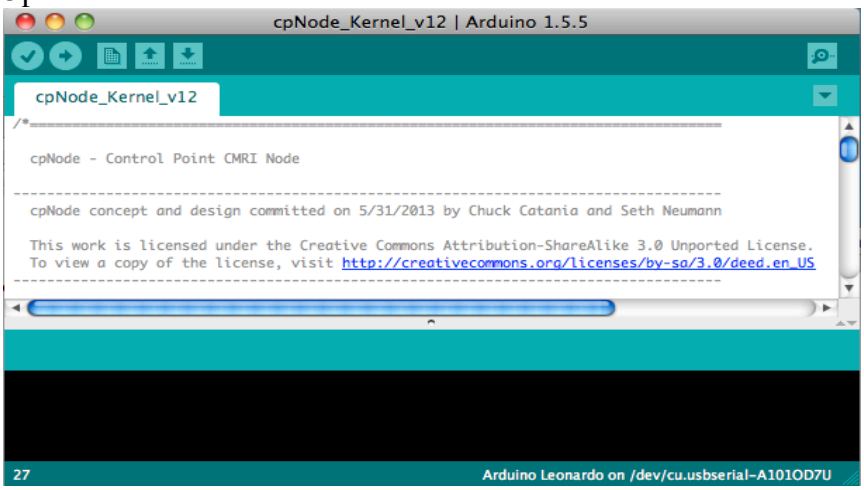
2. From the Tools -> Port menu, select the USB port connected to the BBLeo.



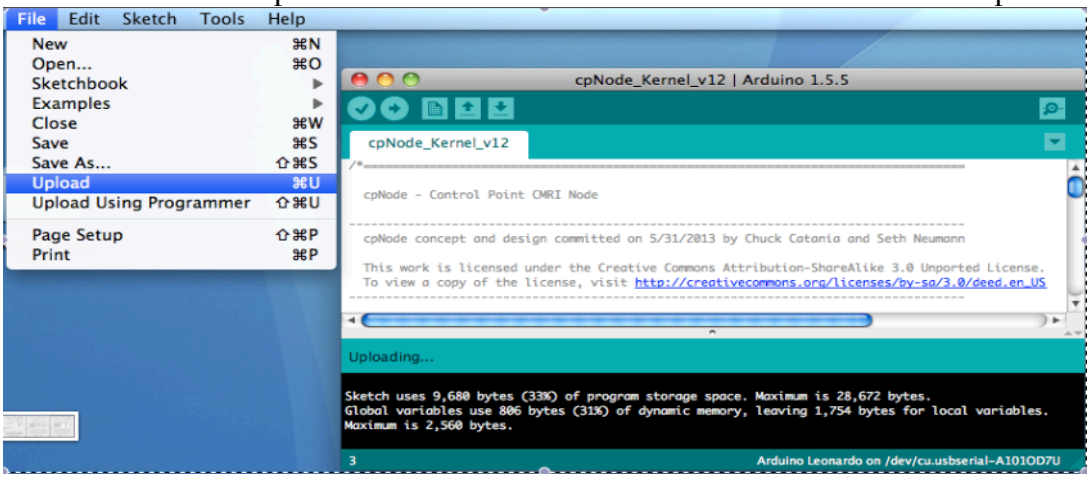
3. From the Tools -> Board menu, select Arduino Leonardo.



4. Open the sketch code file in the IDE.



5. From the File menu, select Upload. The sketch will be compiled first, and then the upload will start. The lower portion of the IDE window will show the status of the Uploading...



6. During the upload, the onboard red and green transmit/receive LEDs will flash indicating proper communication between the IDE and the BBLeo. When the upload has completed, "Done" will be displayed in the status/error portion of the IDE window.

7. Any errors, either during compilation or upload, will be displayed in the lower portion of the IDE window. The text will be displayed in a font color other than white.

6. Connecting a CPNODE to CMRINET (RS-422/RS-485)

The serial communication connection to all CMRInet network nodes is through a four-wire cable. One pair of wires is for transmission, the other pair for receive. The network is defined as half-duplex, RS-422/485. The supported network speeds are standard for serial communications, 9600, 19200, 28800, 38400, 57600, and 115200 bits per second (BPS).

The connection from the control computer to the CMRInet network is through an interface device, which converts USB or RS-232 signals to RS-422/RS-485. In the JLC Enterprises product line is a converter (RS485 Card). Commercially available interfaces, also known as dongles, are also usable. US Converters (Model XS890) is one supplier of these interfaces.

Nodes in a CMRInet network are connected daisy chain fashion, node to node. The cpNode has two connectors for connecting the cpNode to the network. The onboard connector circuit paths are in parallel on the board. Either connector can be used for input or output.

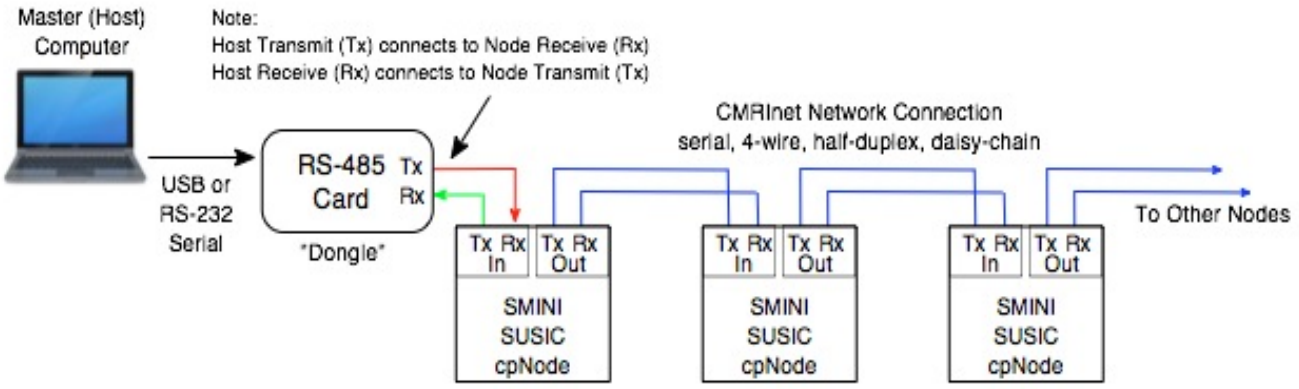


Figure 4 CMRInet Network

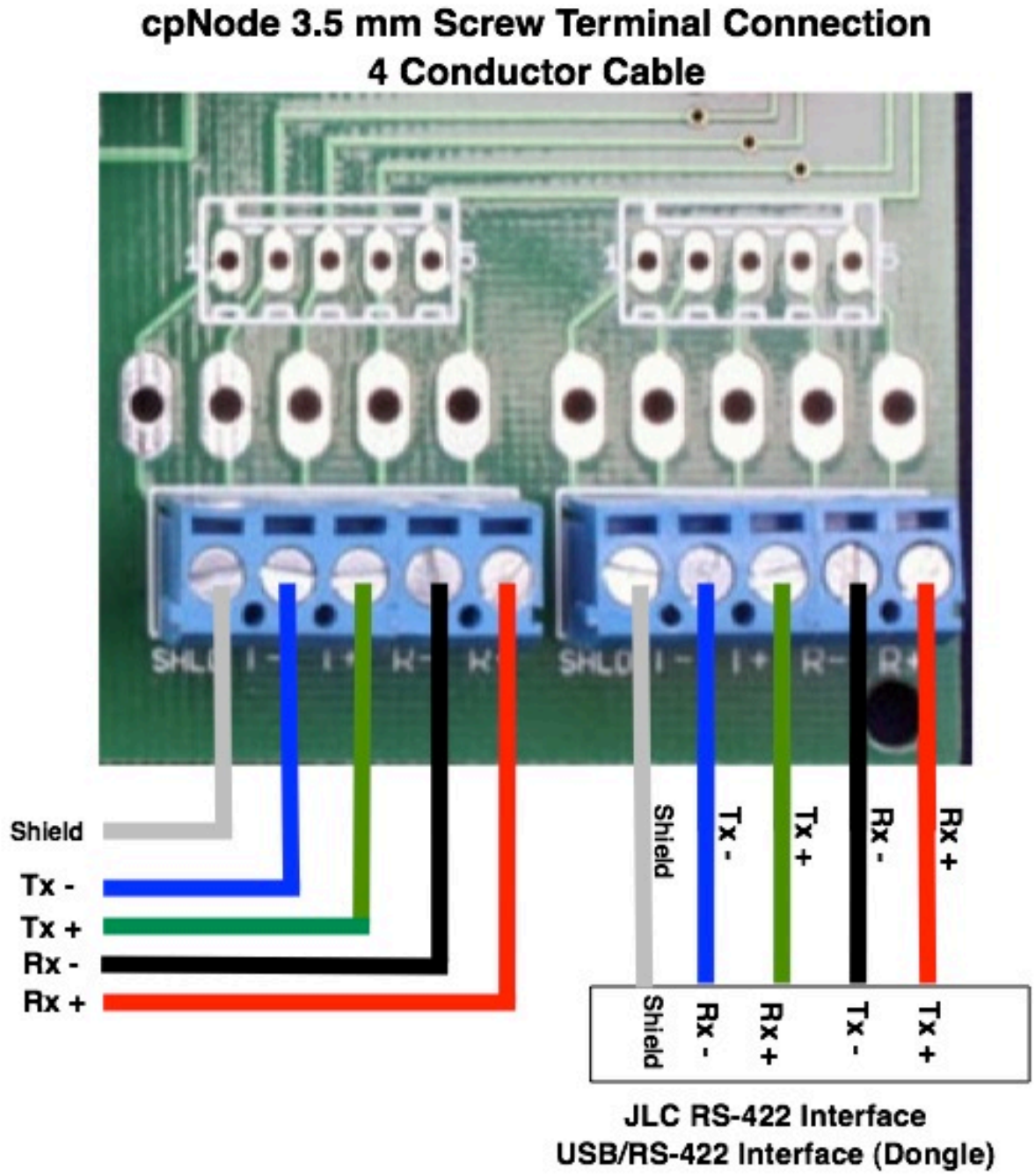


Figure 5 3.5 mm Screw Terminal Connection

There are three connection schemes available on the cpNode, 3.5 mm screw terminals, .100" header pads, and .156 male MOLEX pins. Screw terminals facilitate direct connection of four conductor cable, including the shield or drain wire. The drain terminal is only a tie point for the drain wire. No connection is made to the cpNode ground plane.

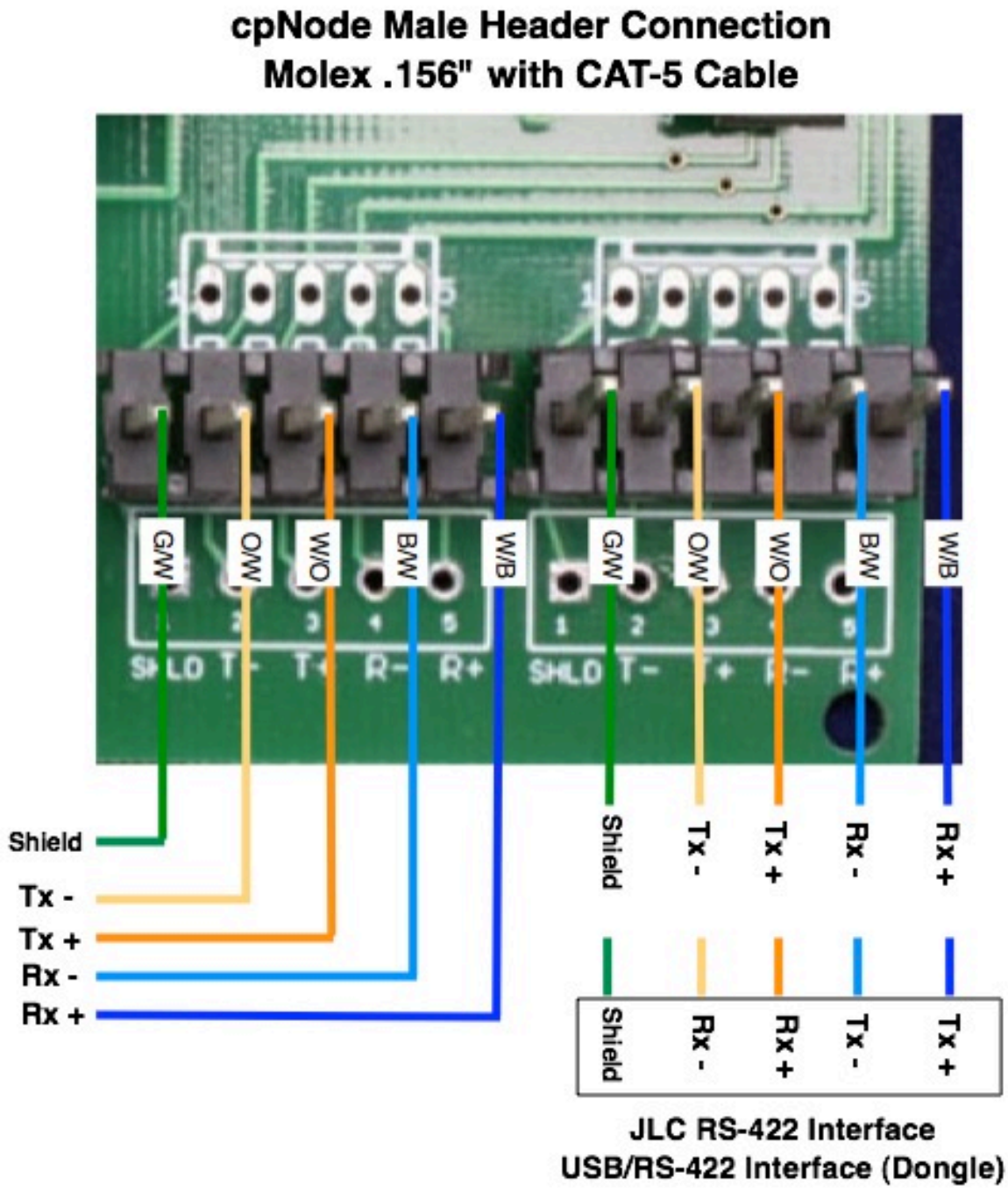


Figure 6 .156" MOLEX Male Header Connection

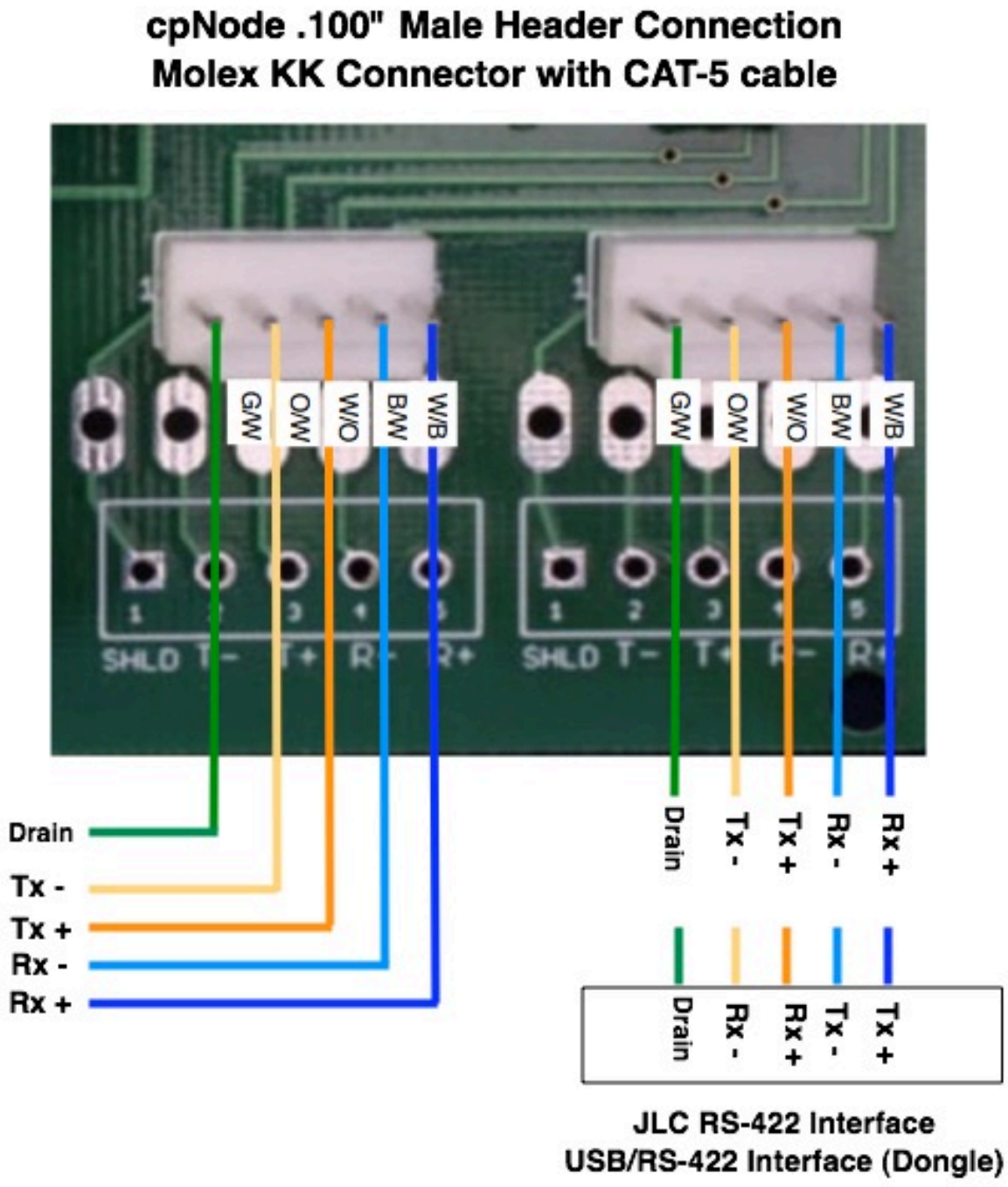


Figure 7 .100" (2.54 mm) Molex KK Connection

The .100" header pads facilitate using male header pins and female connectors with crimp pins. A common type of cable used with this arrangement would be CAT-5. It is advised to use two of the wire pairs in the cable, retaining the color code of the parings. That is, keep the transmit wires paired and the receive wires paired by their respective color code. (White/Blue, Blue/White) and, (White/Orange, Orange/White)

cpNodes are designed to work interchangeably with your existing JLC equipment and can be mixed with existing SMINI and SUSIC nodes, as well as in all cpNode environments. All of your existing CMRInet hardware will continue to work as it has in the past.

7. Setting Node Configuration Parameters

The cpNode operates in the CMRInet serial protocol environment. Each node in the network needs to have a unique address. These addresses are in the range of 0 to 127. JLC Enterprises nodes, the SMINI and SUSIC, have DIP switches on board to set the address. The cpNode address is set in the sketch.

7.1. Kernel Sketch Node Configuration Section

There are two variables, nodeID and CMRINET_SPEED, which can be modified to set the node address and communication line speed. These variables are located near line 106 in the Kernel sketch.
If changes are made, be sure to Save the file.

```
/*=============================================*/
/*=====    NODE CONFIGURATION PARAMETERS     =====*/
/*=============================================*/
                                                //
int nodeID = 20;                                //
                                                //
const long CMRINET_SPEED =  9600;               //
                                                //
/*=============================================*/
/*=====    NODE CONFIGURATION PARAMETERS     =====*/
/*=============================================*/
```

Note: It is suggested to name each of the modified sketches with a name reflecting the address and track controlled track section (e.g. Node20_Aromas). Save the modifications using the Save As menu item.

7.2. CMRINET Node Addressing

The node address for a cpNode is stored in the sketch code, and must be set in the sketch and uploaded into the BBLeo. The address is hard coded into the sketch in order to conserve port bits in the cpNode. Generally, each cpNode is configured and installed for a specific track configuration and once installed, remains in that configuration.
Using the IDE editor, change the value of the variable **nodeID** to the node address. The CMRINet protocol allows for addresses from 0 - 127. Address 20 is the default for the Kernel sketch.

7.3. CMRINET (RS-422/485) Line Speed

Communication line speeds supported by CMRInet nodes are 9600, 19200, 28800, 38400, and 57600 bits per second (BPS). Change the value of the variable **CMRINET_SPEED** to one of the listed values. 9600 BPS is the default for the Kernel sketch.

8. Configuring The Kernel Sketch Standard I/O Maps

The Kernel sketch was created as an example for implementing basic input and output port assignments for the 16 ports on board the cpNode, and for any connected input/output expander (IOX) boards.

Figure 7 shows the port assignment for the cpNode. cpNode ports may be configured as input or output. Using specific IDE libraries, these ports can also be used to drive servomotors, read potentiometer values, and provide variable output voltages.

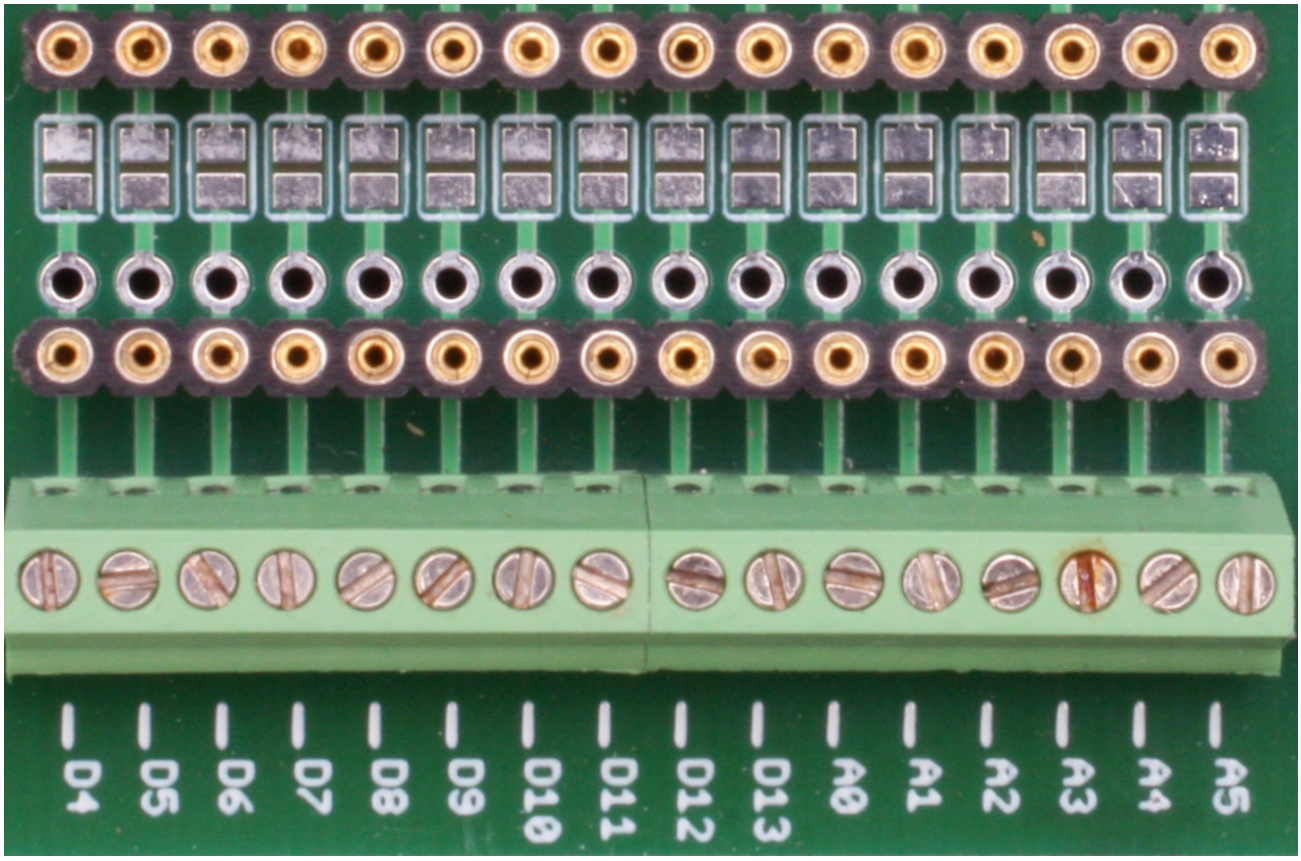


Figure 8 cpNode Input/Output Port Map

The kernel sketch contains a selection of user selectable I/O configurations, which represent standard port assignments. By selecting one of these standard configurations, the sketch code will be set up to read and write the correct port bits from the data bytes sent from the host computer.

The standard port maps are selected in the sketch by un-commenting one of the #define statements. Comments start with double slashes //. Remove both slashes to uncomment the line, insert two slashes to comment the line. Only one #define can be active in the sketch.

The example shows the base node with 8 input bits, and 8 output bits selected.

```
//---------------------------------------------------------
//  Uncomment only one of the standard node configurations
//  #ifdef selects the higher order bit definitions
//---------------------------------------------------------
//#define BASE_NODE
#define BASE_NODE_8IN8OUT
//#define BASE_NODE_8OUT8IN    // cpNode System Test Configuration
//#define BASE_NODE_12OUT4IN
//#define BASE_NODE_16IN
//#define BASE_NODE_16OUT
//#define BASE_NODE_RSMC
//#define BASE_NODE_RSMC_LOCK
```

Figure 9 cpNode IO Configuration Selection

9. Standard Input/Output Port Map Configurations

Each defined configuration provides specific port mapping functions, usable for controlling and managing model railroad layout devices. For example, fascia control panels have switches and buttons, which are inputs, block detectors are also inputs. Throwing turnout motors or setting signal aspect LEDs, require outputs.

The following examples are JMRI sensor and turnout table entries. The system name prefix is "C" for CMRInet, the devices are "T" for turnout (output) and "S" for sensor (input). The node address is designated as "n". Node addresses are in the range of 0 to 127.

9.1. #DEFINE BASE_NODE

The BASE_NODE configuration defines the following:

10 output bits, D4 through D13
6 input bits, A0 through A5

The configuration is designed to support one half of Centralized Traffic Control (CTC) controlled siding. The controlled devices are three block detectors, one stall motor turnout, one dual head mainline signal (G/Y/R)(Y/R), one single head mainline signal (G/Y/R), and one single head low or dwarf siding signal (Y/R).

JMRI System Name	cpNode PORT	JMRI System Name	cpNode PORT
OUTPUTS		**INPUTS**	
CTn001	D4	CSn001	A0
CTn002	D5	CSn002	A1
CTn003	D6	CSn003	A2
CTn004	D7	CSn004	A3
CTn005	D8	CSn005	A4
CTn006	D9	CSn006	A5
CTn007	D10		
CTn008	D11		
CTn009	D12		
CTn010	D13		

9.2. #DEFINE BASE_NODE_8IN8OUT

The BASE_NODE_8IN8OUT configuration defines the following:

8 input bits, D4 through D11
8 output bits, D12 through A5

This configuration provides a standard 8 bits in and 8 bits out.

JMRI System Name	cpNode PORT	JMRI System Name	cpNode PORT
OUTPUTS		**INPUTS**	
CTn001	D12	CSn001	D4
CTn002	D13	CSn002	D5
CTn003	A0	CSn003	D6
CTn004	A1	CSn004	D7
CTn005	A2	CSn005	D8
CTn006	A3	CSn006	D9
CTn007	A4	CSn007	D10
CTn008	A5	CSn008	D11

9.3. #DEFINE BASE_NODE_8OUT8IN

The BASE_NODE_8OUT8IN configuration defines the following:

8 output bits, D4 through D11
8 input bits, D12 through A5

This configuration provides a standard 8 bits in and 8 bits out, ordered with the outputs assigned to the low order byte.

JMRI System Name	cpNode PORT	JMRI System Name	cpNode PORT
OUTPUTS		**INPUTS**	
CTn001	D4	CSn001	D12
CTn002	D5	CSn002	D13
CTn003	D6	CSn003	A0
CTn004	D7	CSn004	A1
CTn005	D8	CSn005	A2
CTn006	D9	CSn006	A3
CTn007	D10	CSn007	A4
CTn008	D11	CSn008	A5

9.4. #DEFINE BASE_NODE_12OUT4IN

The BASE_NODE_12OUT4IN configuration defines the following:

12 output bits, D4 through A1
4 input bits, A2 through A5

This configuration provides 12 output bits and 4 input bits. The configuration useful for driving 3 LED signal heads, and interfacing to block detectors on modular layouts.

JMRI System Name	cpNode PORT	JMRI System Name	cpNode PORT
OUTPUTS		**INPUTS**	
CTn001	D4	CSn001	A2
CTn002	D5	CSn002	A3
CTn003	D6	CSn003	A4
CTn004	D7	CSn004	A5
CTn005	D8		
CTn006	D9		
CTn007	D10		
CTn008	D11		
CTn009	D12		
CTn010	D13		
CTn011	A0		
CTn012	A1		

9.5. #DEFINE BASE_NODE_16IN

The BASE_NODE_16IN configuration defines the following:

16 input bits, D4 through A5

This configuration provides a standard 16 bits in.

JMRI System Name	cpNode PORT
INPUTS	
CSn001	D4
CSn002	D5
CSn003	D6
CSn004	D7
CSn005	D8
CSn006	D9
CSn007	D10
CSn008	D11
CSn009	D12
CSn010	D13
CSn011	A0
CSn012	A1
CSn013	A2
CSn014	A3
CSn015	A4
CSn016	A5

9.6. #DEFINE BASE_NODE_16OUT

The BASE_NODE_16OUT configuration defines the following:

16 output bits, D4 through A5

This configuration provides a standard 16 bits out.

JMRI System Name	cpNode PORT
OUTPUTS	
CTn001	D4
CTn002	D5
CTn003	D6
CTn004	D7
CTn005	D8
CTn006	D9
CTn007	D10
CTn008	D11
CTn009	D12
CTn010	D13
CTn011	A0
CTn012	A1
CTn013	A2
CTn014	A3
CTn015	A4
CTn016	A5

9.7. #DEFINE BASE_NODE_RSMC

The BASE_NODE_RSMC configuration defines the following:

11 output bits, D4 through A0
5 input bits, A1 through A5

The configuration is designed to support the onboard remote stall motor controller (RSMC) connected on port A0. As with the BASE_NODE, the port assignments are for a Centralized Traffic Control (CTC) controlled siding.

JMRI System Name	cpNode PORT	JMRI System Name	cpNode PORT
OUTPUTS		**INPUTS**	
CTn001	D4	CSn001	A1
CTn002	D5	CSn002	A2
CTn003	D6	CSn003	A3
CTn004	D7	CSn004	A4
CTn005	D8	CSn005	A5
CTn006	D9		
CTn007	D10		
CTn008	D11		
CTn009	D12		
CTn010	D13		
CTn011	A0		

9.8. #DEFINE BASE_NODE_RSMC_LOCK

The BASE_NODE_RSMC_LOCK configuration defines the following:

12 output bits, D4 through A0
4 input bits, A2 through A5

The configuration is designed to support the onboard remote stall motor controller (RSMC) connected on port A0 and a Tri-Mode Turnout Controller (TMTC) connected to port A1. As with the BASE_NODE, the port assignments are for a Centralized Traffic Control (CTC) controlled siding.

JMRI System Name	cpNode PORT	JMRI System Name	cpNode PORT
OUTPUTS		**INPUTS**	
CTn001	D4	CSn001	A2
CTn002	D5	CSn002	A3
CTn003	D6	CSn003	A4
CTn004	D7	CSn004	A5
CTn005	D8		
CTn006	D9		
CTn007	D10		
CTn008	D11		
CTn009	D12		
CTn010	D13		
CTn011	A0		
CTn012	A1		

10. CONFIGURING A CPNODE USING JMRI

A cpNode can be configured in JMRI as an SMINI or USIC/SUSIC, depending upon the total number of input and output bytes. A cpNode system is based upon 8 bit data bytes.

For a cpNode with no IOX's, set the type to SMINI. JMRI will allocate 3 bytes for inputs and 6 bytes for outputs. As with any CMRI node defined in JMRI, you must have at least one sensor (input) defined for the node to be polled. You can configure a cpNode with 16 outputs, but JMRI will not poll the node. Outputs will be sent to the node when output states change in JMRI.

If you add IOXs, it is best to configure the cpNode in JMRI as a USIC or SUSIC, with input and output cards assigned for the total number of data bytes. The cpNode bytes are always the first two in poll and transmit messages. IOX bytes are appended to the message after the cpNode bytes, in the order defined in the sketch.

The rule of thumb is to count the total number of inputs and outputs in the cpNode hardware configuration. If the number of inputs is less than or equal 3 AND the number of outputs is less than or equal to 6, the SMINI definition will fit. Any I/O configuration greater than 3 inputs and 6 outputs, a USIC or SUSIC with 24 bit (3 bytes) or 32 bit (4 bytes) cards should be chosen.

11. Input/Output Expander (IOX) Port Map Configurations

Input/Output Expander (IOX) boards provide additional data ports that can be added to a cpNode. These boards are attached through the i2C serial bus connector on the cpNode. This bus is four wires and interconnects the IOX boards as a daisy chain. A chip onboard the IOX provides 16 bits of input or output, representing two data bytes of eight bits.

IOX boards provide either 16 bits (IOX16) or 32 bits (IOX32). An IOX32 is two IOX16's packaged together on one board. A maximum of 8 groups of 16 bits can be connected to the expander bus. Using a mix of IOX boards, a total of 128 additional I/O bits can be added to a cpNode.

Data, which are defined as inputs, appear as bytes appended after the onboard cpNode input bytes. The node always sends its' two input bytes regardless of the defined configuration in the sketch, followed by any input bytes from connected I/O expanders.

Ports, which are assigned as inputs in the Kernel sketch, are configured as INPUT_PULLUP, which enables an internal pull-up resistor, per port in the Arduino. No external pull up resistors are needed for inputs defined this way.

Data, which are defined as outputs, appear as bytes after the onboard cpNode output bytes are received. The node always receives its' two output bytes regardless of the defined configuration in the sketch.

The data stream order (sent or received) is always, two onboard data bytes followed by zero to n IOX data bytes. The IOX data bytes are ordered by IOX board address starting the lowest address enabled.

11.1. IOX INTERCONNECTION

IOX boards are connected together using a four wire cable. The maximum total length of this cable is 6 feet (2 meters). On the IOX boards, J1 and J2 are the connectors for the I2C interconnect. IOX boards are connected in a daisy chain manner, from board to board. An established color code for the four wires is as follows:

Pin 1	Yellow	I2C Serial Clock (SCL)
Pin 2	White	I2C Serial Data (SDA)
Pin 3	Red	I2C Power (5vdc)
Pin 4	Black	I2C Ground

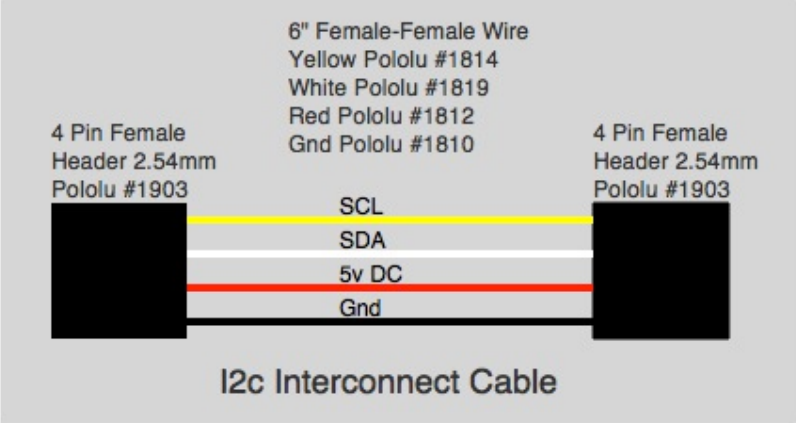


Figure 10 I2c Interconnet Cable

Note: Pololu (www.pololu.com) is one supplier of .100" headers, crimp pins, cables, and screw terminal blocks.

11.2. Setting IOX Board Addresses

Each IOX has a set of plug jumpers, which sets the unique board address on the bus. The address range for the IOX boards is hexadecimal 20 through 27.

Located in the lower left corner of the IOX, are the address jumper pins. By inserting a small jumper plug, vertically as shown in the table, the board address is set. See Figure 3 for setting these jumpers. The jumpers are designated by the vertical colored bar in the table. An IOX32 has two addresses, starting on an even address boundary, e.g, 20,22,24,26. Setting the lower address, left port of the IOX32, automatically sets the next higher address for the right side of the board.

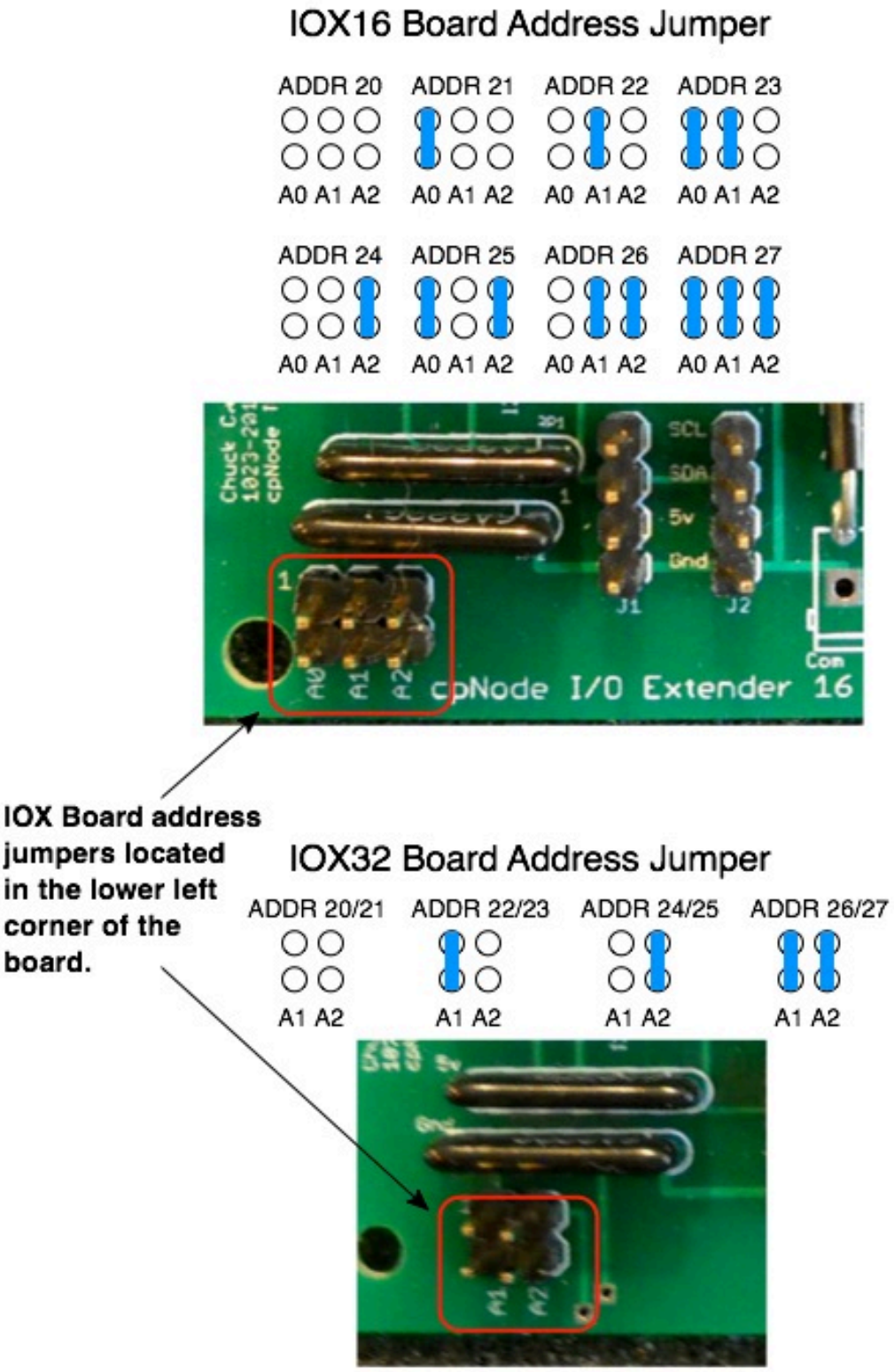


Figure 11 I/O Expander Board Address Jumpers

11.3. Input/Output Extender Port Maps

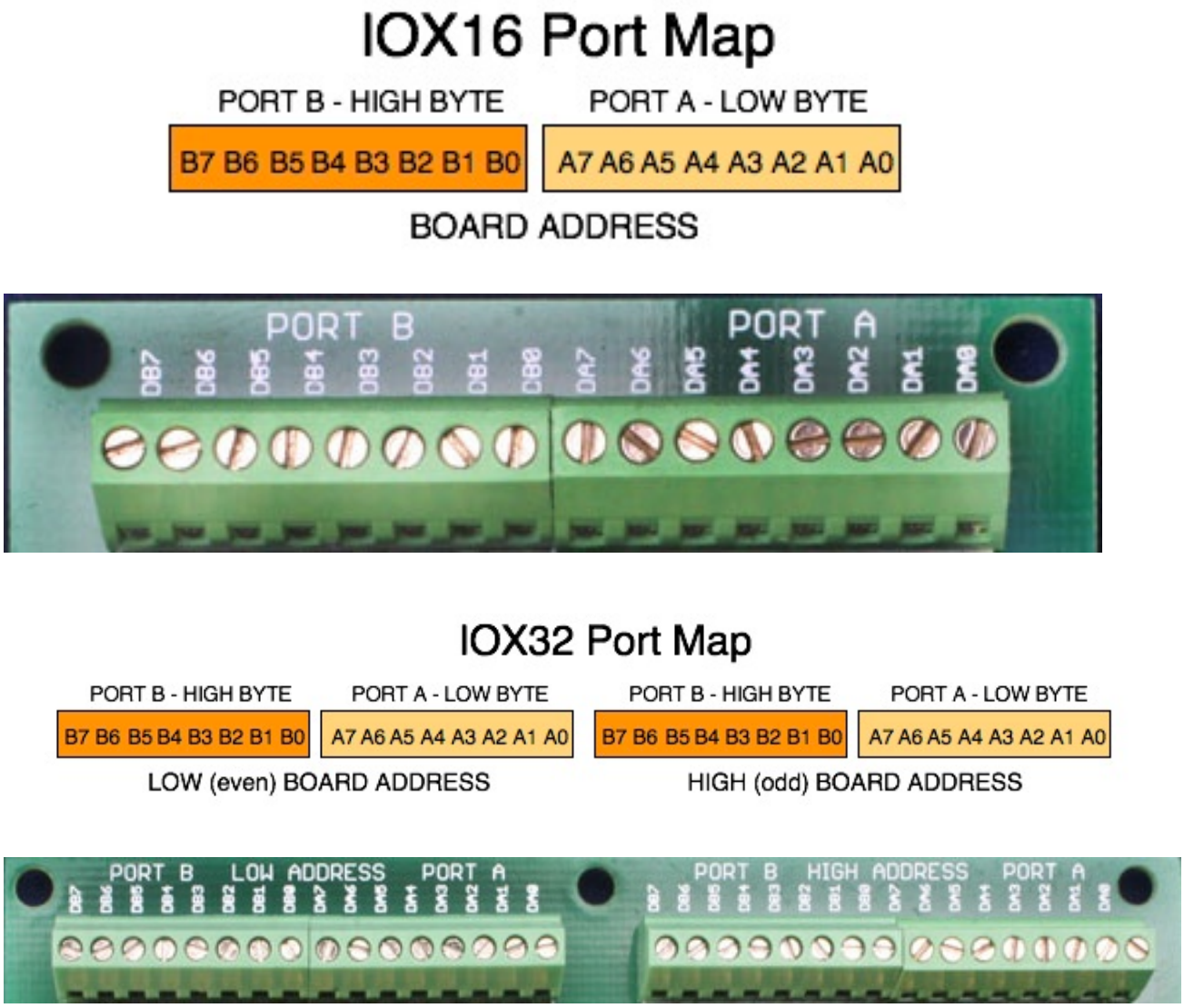


Figure 12 I/O Expander (IOX) Port Maps

11.4. IOX Port Assignment Sketch Code

To configure the standard Kernel sketch for attached I/O Expanders, remove the comment slashes (//) from the #define USE_IOX statement in code.

```
//----------------------
// I/O Expander boards
//----------------------
#define USE_IOX
```

Figure 13 I/O Expander Port Enable Define

Each IOX connected to a cpNode must have the port direction defined in the sketch. The initialization code which runs at node startup time, configures each of the IOX ports, A and B, as input or output, by defined board address.

The IOX_ioMap table has one entry for each possible IOX board and port, organized by board address, starting with address 20. See Figure 9 for what the IOX map looks like in the sketch code.

Each port (A or B) is assigned as input by changing the -1 to a 1, or to an output by entering a 0 (zero). Leave unassigned ports in the table as -1.

```
// I2C port direction bytes 0 = OUTPUT  1 = INPUT  -1 = Not Assigned
//-------------------------------------------------------------------
                            // Addr  0x20    0x21    0x22    0x23    0x24    0x25    0x26    0x27
                            // Port  A  B    A  B    A  B    A  B    A  B    A  B    A  B    A  B
          IOX_ioMap[max_IOX]   =   { -1,-1,   -1,-1,   -1,-1,   -1,-1,   -1,-1,   -1,-1,   -1,-1,   -1,-1 };
```

Figure 14 IOX Port Direction Assignment Map

An example: Two I/O expanders are attached to a cpNode. One IOX32 is set to address 20, with port A set to input and port B set to output, and address 21 set to two outputs. One IOX16 is set to address 22 with port A and B set to input. This is what the IOX_ioMap table would look like:

```
// I2C port direction bytes 0 = OUTPUT  1 = INPUT  -1 = Not Assigned
//-------------------------------------------------------------------
                            // Addr  0x20    0x21    0x22    0x23    0x24    0x25    0x26    0x27
                            // Port  A  B    A  B    A  B    A  B    A  B    A  B    A  B    A  B
          IOX_ioMap[max_IOX]   =   {  1, 0,    0, 0,    1, 1,   -1,-1,   -1,-1,   -1,-1,   -1,-1,   -1,-1 };
```

Figure 15 Example IOX Port Direction Assignment Map

11.5. IOX Port Assignment Example

The following examples are table entries, which could be used for assigning I/O expander port bits to JMRI sensors (input) or turnouts (output). The system name prefix is "C" for CMRInet, the devices are "T" for turnout (output) and "S" for sensor (input). The node address is designated as "n". Node addresses are in the range of 0 to 127.

Data bytes from I/O expanders are placed into the data stream in sequential order starting with the lowest configured IOX board address. This is true for data bytes either inbound or outbound.

The tables shows three configuration examples; all bits are input, all bits are output, eight bits are input and eight bits are output.

JMRI Table System Name	IOX16 Port		JMRI Table System Name	IOX16 Port		JMRI Table System Name	IOX16 Port
ALL INPUTS			**ALL OUTPUTS**			**8 INPUTS**	
CSn017	DA0		CTn017	DA0		CSn017	DA0
CSn018	DA1		CTn018	DA1		CSn018	DA1
CSn019	DA2		CTn019	DA2		CSn019	DA2
CSn020	DA3		CTn020	DA3		CSn020	DA3
CSn021	DA4		CTn021	DA4		CSn021	DA4
CSn022	DA5		CTn022	DA5		CSn022	DA5
CSn023	DA6		CTn023	DA6		CSn023	DA6
CSn024	DA7		CTn024	DA7		CSn024	DA7
CSn025	DB0		CTn025	DB0			
CSn026	DB1		CTn026	DB1		**8 OUTPUTS**	
CSn027	DB2		CTn027	DB2		CTn017	DB0
CSn028	DB3		CTn028	DB3		CTn018	DB1
CSn029	DB4		CTn029	DB4		CTn019	DB2
CSn030	DB5		CTn030	DB5		CTn020	DB3
CSn031	DB6		CTn031	DB6		CTn021	DB4
CSn032	DB7		CTn032	DB7		CTn022	DB5
						CTn023	DB6
						CTn024	DB7

Figure 16 JMRI Device Table Name Example IOX16

11.6. IOX32 Port Assignment Example

JMI Table System Name	IOX32 **Even** Board Address	JMRI Table System Name	IOX32 **Even** Board Address	JMRI Table System Name	IOX32 **Even** Board Address
ALL INPUTS		**ALL OUTPUTS**		**8 INPUTS**	
CSn017	DA0	CTn017	A0	CSn017	A0
CSn018	DA1	CTn018	A1	CSn018	A1
CSn019	DA2	CTn019	A2	CSn019	A2
CSn020	DA3	CTn020	A3	CSn020	A3
CSn021	DA4	CTn021	A4	CSn021	A4
CSn022	DA5	CTn022	A5	CSn022	A5
CSn023	DA6	CTn023	A6	CSn023	A6
CSn024	DA7	CTn024	A7	CSn024	A7
CSn025	DB0	CTn025	B0		
CSn026	DB1	CTn026	B1	**8 OUTPUTS**	
CSn027	DB2	CTn027	B2	CTn017	B0
CSn028	DB3	CTn028	B3	CTn018	B1
CSn029	DB4	CTn029	B4	CTn019	B2
CSn030	DB5	CTn030	B5	CTn020	B3
CSn031	DB6	CTn031	B6	CTn021	B4
CSn032	DB7	CTn032	B7	CTn022	B5
				CTn023	B6
				CTn024	B7
	IOX32 **Odd** Board Address		IOX32 **Odd** Board Address		IOX32 **Odd** Board Address
ALL INPUTS		**ALL OUTPUTS**		**8 INPUTS**	
CSn033	DA0	CTn033	A0	CSn025	A0
CSn034	DA1	CTn034	A1	CSn026	A1
CSn035	DA2	CTn035	A2	CSn027	A2
CSn036	DA3	CTn036	A3	CSn028	A3
CSn037	DA4	CTn037	A4	CSn029	A4
CSn038	DA5	CTn038	A5	CSn030	A5
CSn039	DA6	CTn039	A6	CSn031	A6
CSn040	DA7	CTn040	A7	CSn032	A7
CSn041	DB0	CTn041	B0		
CSn042	DB1	CTn042	B1	**8 OUTPUTS**	
CSn043	DB2	CTn043	B2	CTn025	B0
CSn044	DB3	CTn044	B3	CTn026	B1
CSn045	DB4	CTn045	B4	CTn027	B2
CSn046	DB5	CTn046	B5	CTn028	B3
CSn047	DB6	CTn047	B6	CTn029	B4
CSn048	DB7	CTn048	B7	CTn030	B5
				CTn031	B6
				CTn032	B7

Figure 17 JMRI Device Table Name Example IOX32

12. CPNODE CMRINET LINE SPEED SELECTION

The communication line speed on the cpNode board is selected by the values of R1 and C1. The default values for an assembled and tested board is 12,000 ohms for R1, and .1 microFarad (uF) for C1. A table of R1/C1 values by line speed is on the schematic. For 9600 and 19200 BPS, a value of .1uf for C1, and a value of 12k for R1 can be used.

Note: The AutoRTS circuit will operate properly for all of the standard line speeds with 12K and .1uf values.

* AutoRTS Resistor Value

BPS	R1	C1
9600	12000	.1 uF
19200	6000	.1 uF
28800	3900	.1 uF
38400	3000	.1 uF
57600	2000	.1 uF
115200	1000	.1 uF

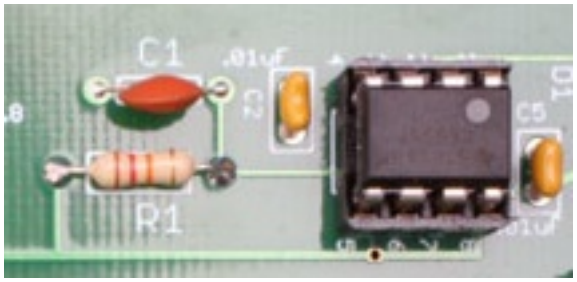


Figure 18 CMRInet Line Speed Selection

13. LED Connections - Common Anode/Common Cathode

Light emitting diodes (LED) can be connected to a cpNode and I/O expanders ports defined as OUTPUT, either as **current sinking** (common anode-CA) or **current sourcing** (common cathode-CC). The "common" is either ground for CC or a +5 Vdc reference for CA.

The cpNode has two headers near the stall motor connector, which can be used to connect to the LED common line. One header is labeled GND and the other LEDCom.

The customary tie point for the LED common would be the LEDCom header. A two position solder pad jumper allows the LEDCom header to either be +5 Vdc for CA, or Gnd for CC. The default is CA with a board trace between the center pad and the pad marked CA.

To configure the LEDCom header for common cathode, cut the trace between the center pad and the CA pad, and put a blob of solder between the center pad and the pad marked CC.

NOTE: Be sure to completely cut the default trace, otherwise, there will be a dead short between the board power and ground. Possible damage to the onboard components or the Arduino could occur.

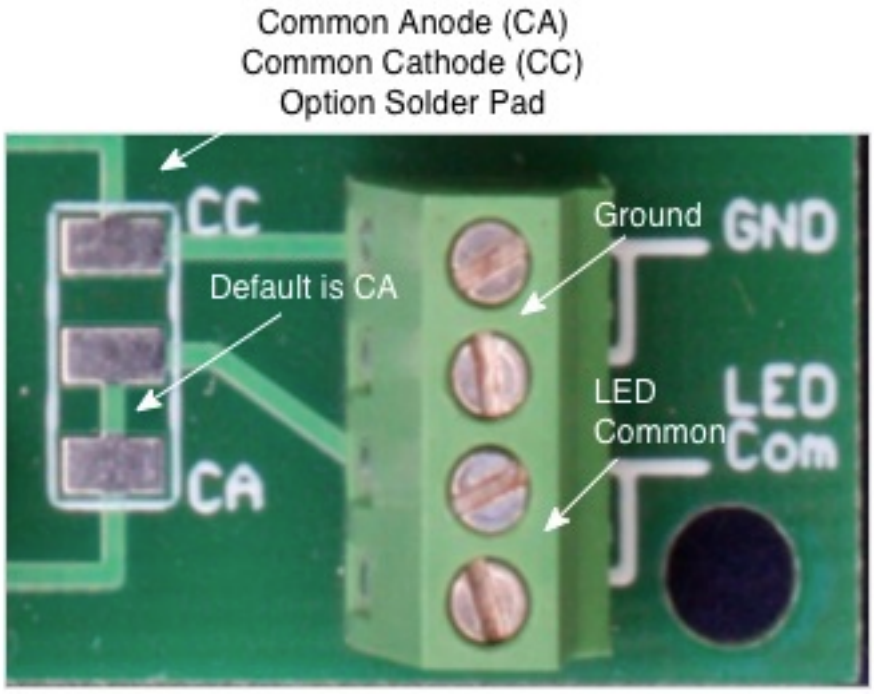


Figure 19 LED Common Option Solder Pad and Connection Header

13.1. Wiring Common Anode/Common Cathode LEDs

Resistors must be placed in series with the LEDs to limit the current to the device. With no resistor, applying voltage to the LED will cause it to glow brightly once, and then burn out.

Resistor values are selected to adjust the desired brightness of the LED. A 1000 ohm resistor is a good, safe starting value for most color LEDs. Bright White LEDs may need a larger resistor value. 4700 ohms is a safe value to start with. The lower the value, the brighter the display.

Note: 5 Vdc maximum on LEDs connected to cpNodes. If you need higher supply voltages to conform to existing standards, use a CSNK breakout board connected to outputs.

While the maximum current per output port at 5 Vdc is 40 ma, the board total is 160ma, so we recommend limiting the current to 10 ma per output. Modern designed LEDs will operate fine at 5 ma.

To determine the value of the current limiting resistor, use the following formula called Ohm's Law. **(Voltage - LED voltage drop 1.4v) / LED Current = Resistor value.**

For example: Assume the Voltage is 3.5 volts (5v - 1.4v), an LED current of 5 ma (.005 amps). The calculation is: 3.5v / .005a = 700 ohms.
The closest standard 10% resistor value is 680 ohms. 1/4 or 1/8 watt resistors will work.

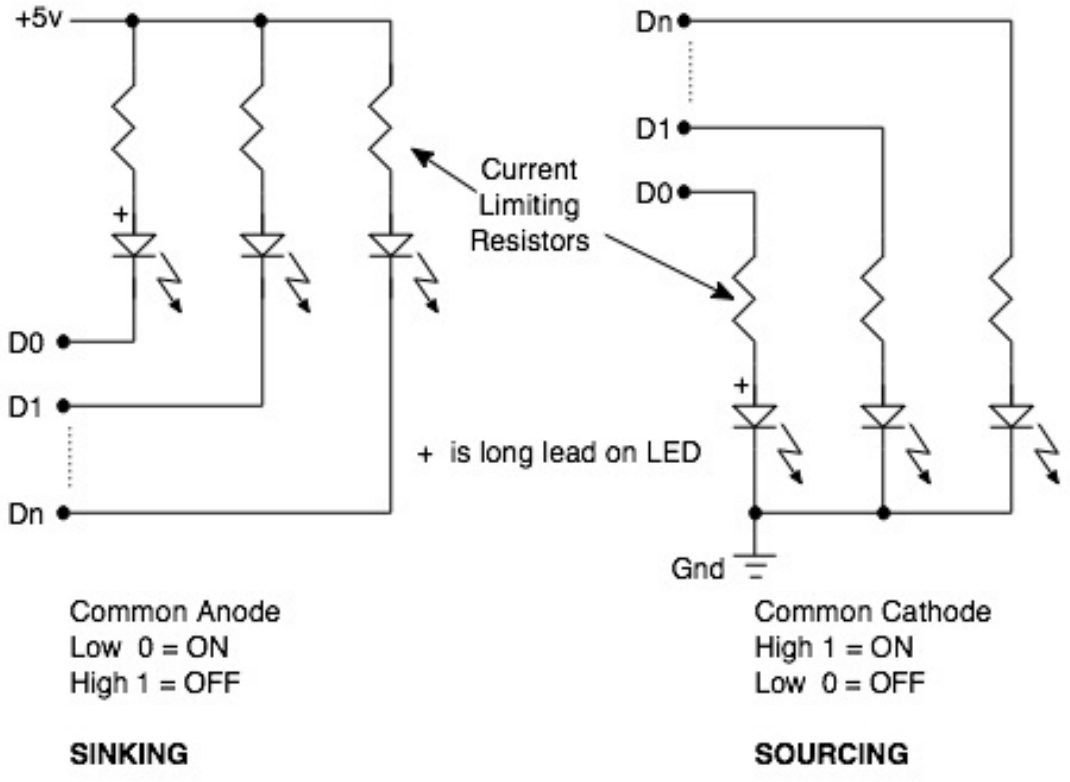
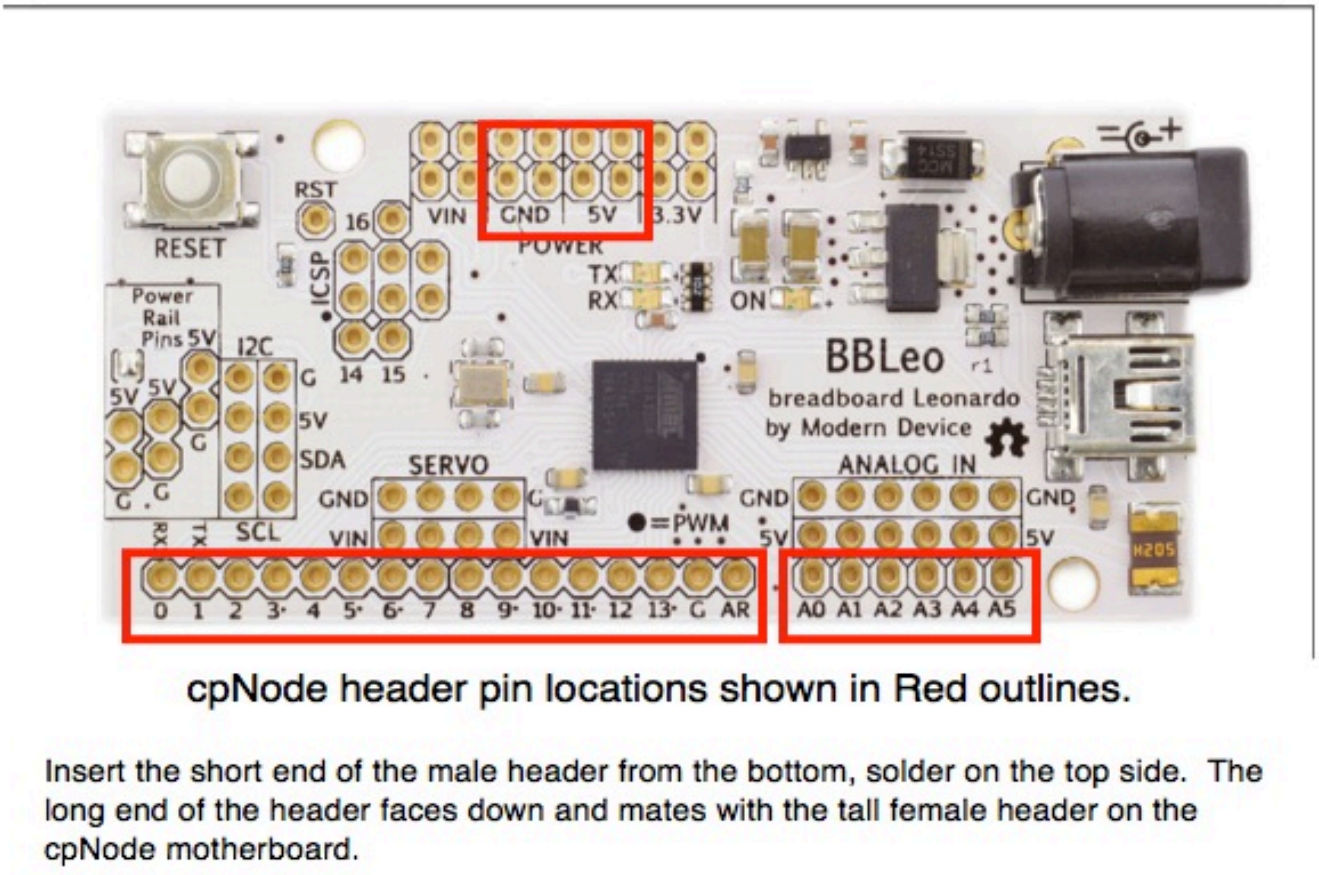


Figure 20 LED Common Anode/Common Cathode Connection Diagram

14. BBLEO to CPNODE Header Pin Location

The BBLeo from Modern Devices is the Arduino style board, which connects to the cpNode motherboard via male header pins. These pins need to be soldered into pads shown in Figure 13. The long side of the male pin needs to be facing down on the underside of the BBLeo board.



cpNode header pin locations shown in Red outlines.

Insert the short end of the male header from the bottom, solder on the top side. The long end of the header faces down and mates with the tall female header on the cpNode motherboard.

Figure 21 BBLeo/cpNode Connection Header Pins

15. Resources For CMRInet, CPNODE, ARDUINO, JMRI

Model Railroad Control Systems (MRCS)
www.modelrailroadcontrolsystems.com
Seth Neumann, seth@modelrailroadcontrolsystems.com
Chuck Catania, chuck@modelrailroadcontrolsystems.com

Yahoo User Groups
Arduini Arduino technology for model railroading
https://groups.yahoo.com/neo/groups/Arduini/info
CMRI_users Original Computer Model Railroad Interface group
https://groups.yahoo.com/neo/groups/CMRI_Users/info
JMRIusers Java Model Railroad Interface software group
https://groups.yahoo.com/neo/groups/jmriusers/info

Official Arduino Web Site
http://arduino.cc/

Licensed Arduino Hardware Suppliers
Modern Device http://moderndevice.com/
Sparkfun https://www.sparkfun.com/
AdaFruit http://www.adafruit.com/

NMRA Layout Control Specification
LCS-9.10 C/MRI Introduction v1.0 (2014.12.01)
http://www.nmra.org/sites/default/files/standards/sandrp/Other_Specifications/lcs-9.10_cmri_intro_v1.0.pdf
LCS-9.10.1 CMRInet v1.1 (2014.12.01)
http://www.nmra.org/sites/default/files/standards/sandrp/Other_Specifications/lcs-9.10.1_cmrinet_v1.1.pdf

Official CMRINET Web Site, Dr. Bruce Chubb, MMR
http://www.jlcenterprises.net/index.htm

SLIQ Electronics - Official CMRINET Hardware Web Site, Marc Robertson
http://sliqelectronics.com/products/

Java Model Railroad Interface (JMRI) - Open Source Model Railroad Software
http://jmri.sourceforge.net/