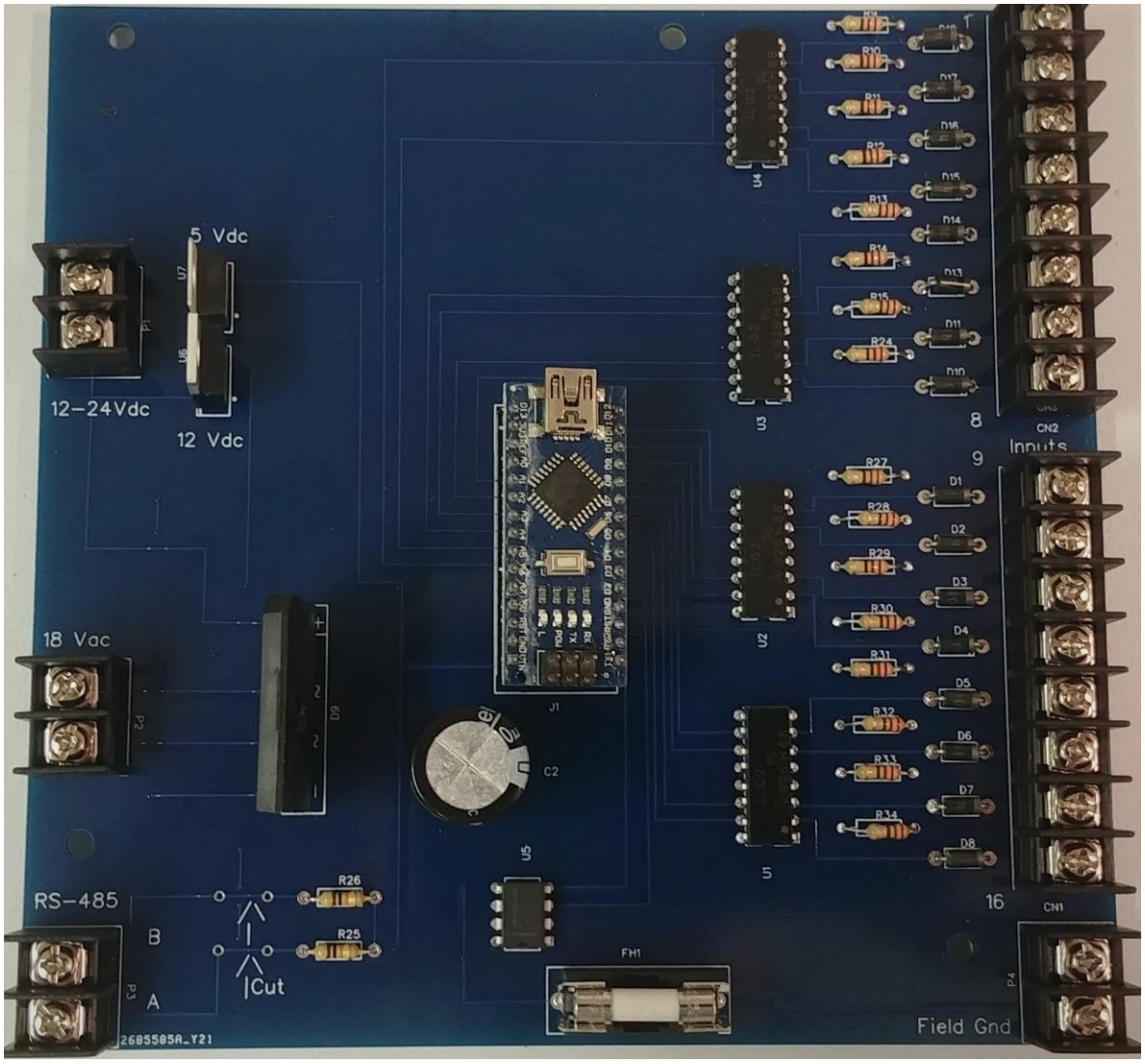
C/MRI 16 Inputs Card

This card is designed as a generic 16 input interface card. The inputs are active low (you apply ground.)



Along the left edge the top terminal strip is for 12 to 24 VDC and the middle terminal strip is for 18 VAC. These are the two types of supply power the board can use. **CAUTION: You must choose just one of these to supply power, never connect both types of power at the same time.**

The bottom terminal strip (still left edge) is the RS-485 connection. The Arduino is programmed to use this port as a C/MRI node (Computer / Model Railroad Interface) and will connect directly to JMRI (Java Model Railroad Interface) as such.

Just to the right of the RS-485 connection is a silk screen label that says “cut here.” This feature is used in conjunction with how many cards you are using. If you are using a single card on the RS-485 connection leave the card as delivered, if however you use more than one card you will need to cut this PCB trace on all but the first card. More details to follow.

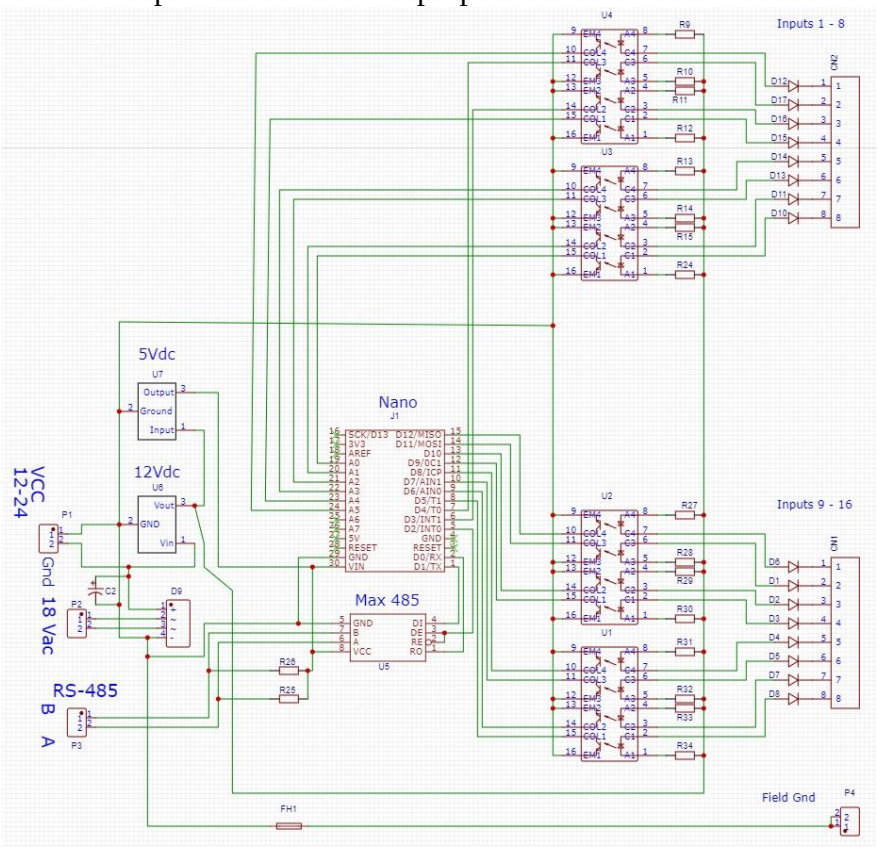
To the bottom right (shown horizontally) is the 1.6 Amp fuse. This fuse protects the field devices from drawing excessive current and the card from short circuits.

The bottom right edge terminal strip is the Field Ground connection. This terminal is provided as a convenient point to source your field devices. More details to follow.

The two top right edge terminal strips are for the sixteen input terminals. Route the field ground through your device and back to one of these inputs.

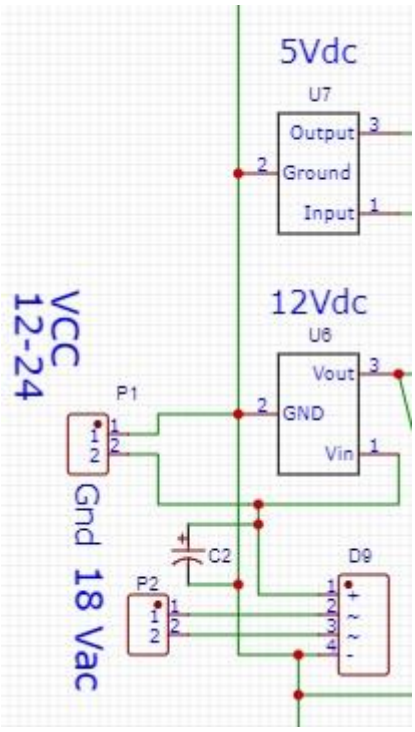
Schematic Overview:

Please don’t be alarmed by the following schematic, mastery is not necessary to use the C/MRI siding card. It is included for a more in-depth understanding of how the card works and will allow advanced users to adapt the card for other purposes.



Blow up the PDF page for more clarity

Power sources:



The board may be run from either a 12-24 VDC or an 18 VAC power source (not both.)

18 VAC – Is applied to terminal strip P2 and connected to a full bridge rectifier D9 (terminals 2 & 3) where it is converted to approximately 25 VDC. The output of the rectifier is applied to a regulator (U6 pin 1) which limits the output (pin 3) to 12 VDC.

The 12 VDC is used in two places:

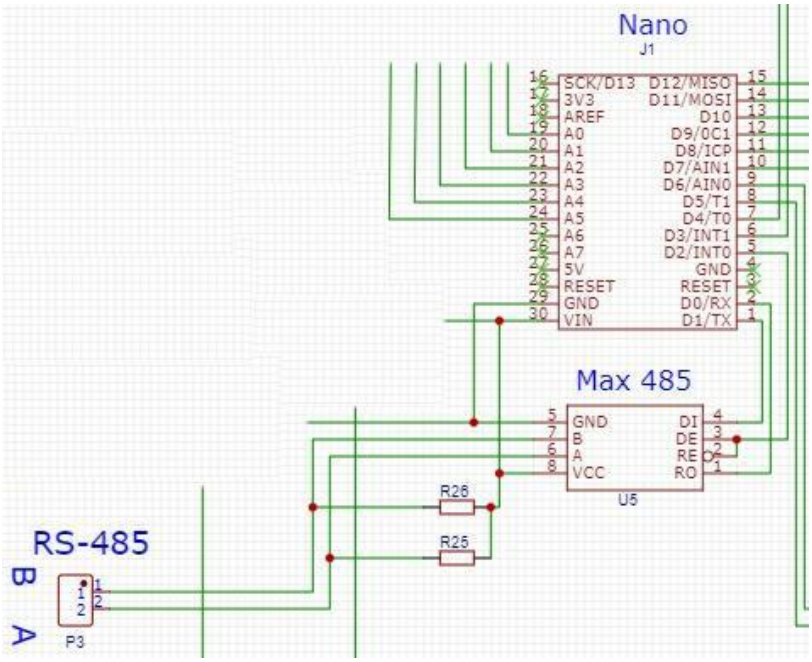
1. Power to source the outputs
2. The input to U7 (pin 1), the 5 VDC regulator where the output (pin 3) is used for the internal board logic.
3. NOTE: D9-4, U6-2 and U7-2 form a common ground.

12-24 VCC – Is applied to terminal strip P1 and connected to the 12 VDC regulator (U10) which in-turn feeds the 5VDC regulator (U9)

CAUTION: Never connect both 18VAC and 12-24VDC to the same board at-the-same-time as this will damage the rectifier (D9) and 12VDC regulator (U10) and possibly the externally connected power sources.

RS-485 Connection:

This is the connection that allows the board to communicate with and external control device, such as a computer running JMRI (Java Model Railroad Interface.) The Arduino on this card uses libraries to make it communicate as a C/MRI card (Computer / Model Railroad Interface.)

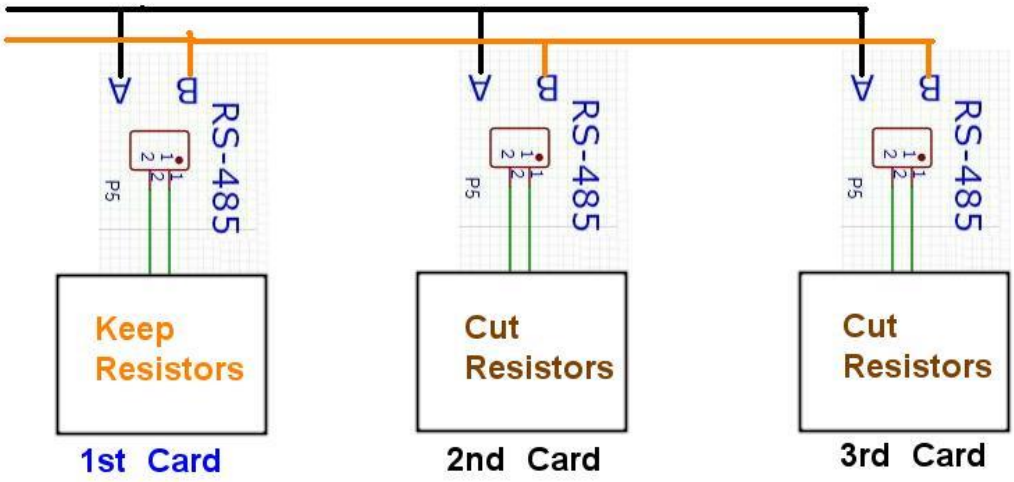


The computer connects at the terminal strip labeled RS-485. The computer will probably have a USB-to-RS-485 adapter. Make sure to follow the "A" and "B" wiring so that "A" goes to "A" and "B" goes to "B".

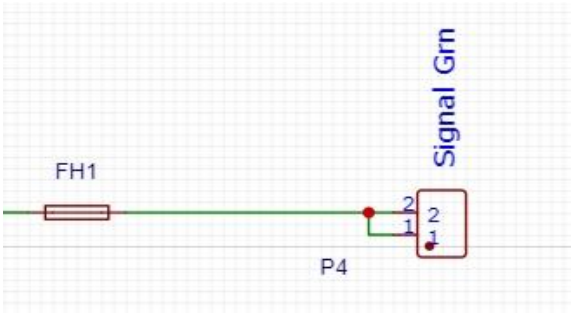
The communications bus needs what are called pull-up resistors to ensure that no noise (unwanted + to - transitions) are seen by the Max 485 chip. Here they are identified as R26 and R25 but the numbers may vary depending on which card you are using. The important thing to know is that the "bus" only needs one set of resistors. So we want to keep the resistors on the first board and disable them on all additional boards. Each card will have a silk trace "Cut" or "Cut Here" point. (Depending on the card this could be labeled on the top-side or the bottom-side of the card.)

If you only have one card on the bus you are done – do nothing with the resistors. If you have two or more cards on the bus then you must,

1. Leave the resistors on the first board untouched (do not cut)
2. On each additional card you must disable these resistors by cutting the copper trace at the point labeled "Cut" or "Cut Here." An X-Acto knife works well for this and the gap only needs to be wide enough to see that the trace has been cut.



Signal Ground:

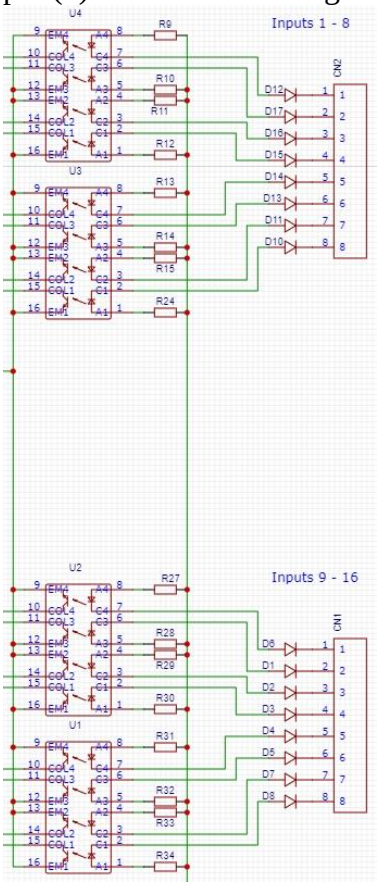


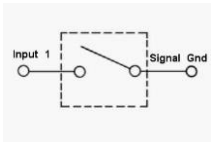
The Signal GND terminal strip (P4) terminals (1) and (2) are jumpered together on the board. Use these terminals to provide the connection to the common side of the field devices to be controlled. You can also use this ground to connect the common side of the input devices.

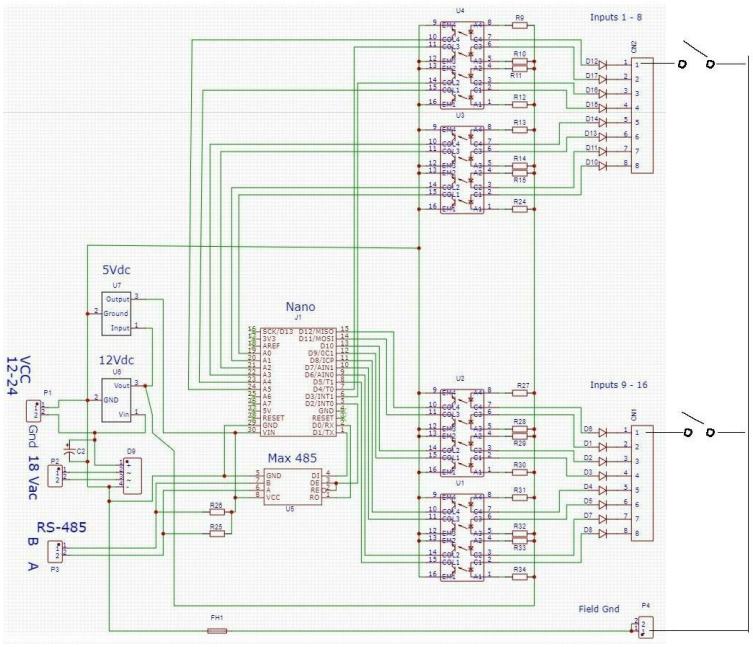
The board comes with a 1.6 Amp fuse but it is safe to use a more common 2 Amp fuse. If this fuse blows then most likely a field device has been shorted to 12VDC. Nothing on the card will blow this fuse.

Inputs 1-16:

There are eight inputs on terminal block CN2. Input 1 is on terminal (1) through Input 8 is on terminal 8. Inputs 9 to 16 are on terminal board CN1. Examining Input 1 we see diode D12, this is a reverse polarity protection diode. D12 is connected to U4 pin (7) this chip provides galvanic isolation for the Nano input terminals. Resistor R9 on pin (8) is a current limiting resistor. Input 1 is active when CN2 pin (1) is connected to Signal GND (active low.)







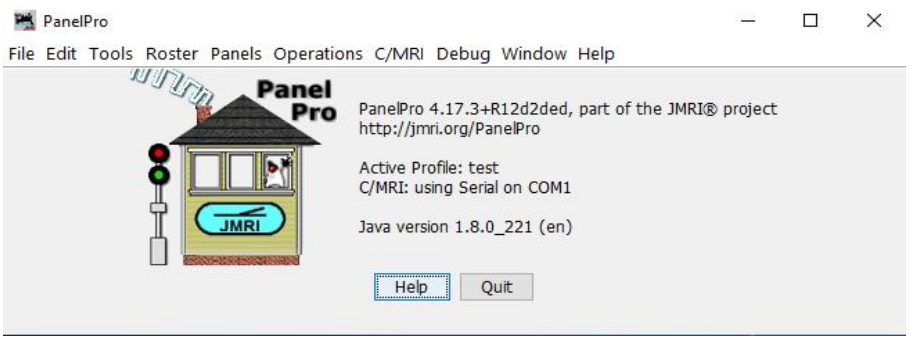
This is an example of a typical input connections.

Note how the board “Field Ground” connection is on the common side for all connections.

Configuring JMRI to use the board as C/MRI hardware:

The on-board Arduino chip is programmed to act as a C/MRI SMINI card with the default address of zero (0). Though a true SMINI card can address 24 inputs and 48 outputs the Nano chip cannot handle that much I/O. This input / output board can only handle a total of 16 I/O points, so all SMINI node I/O points beyond sixteen (16) will be ignored by the Nano chip.

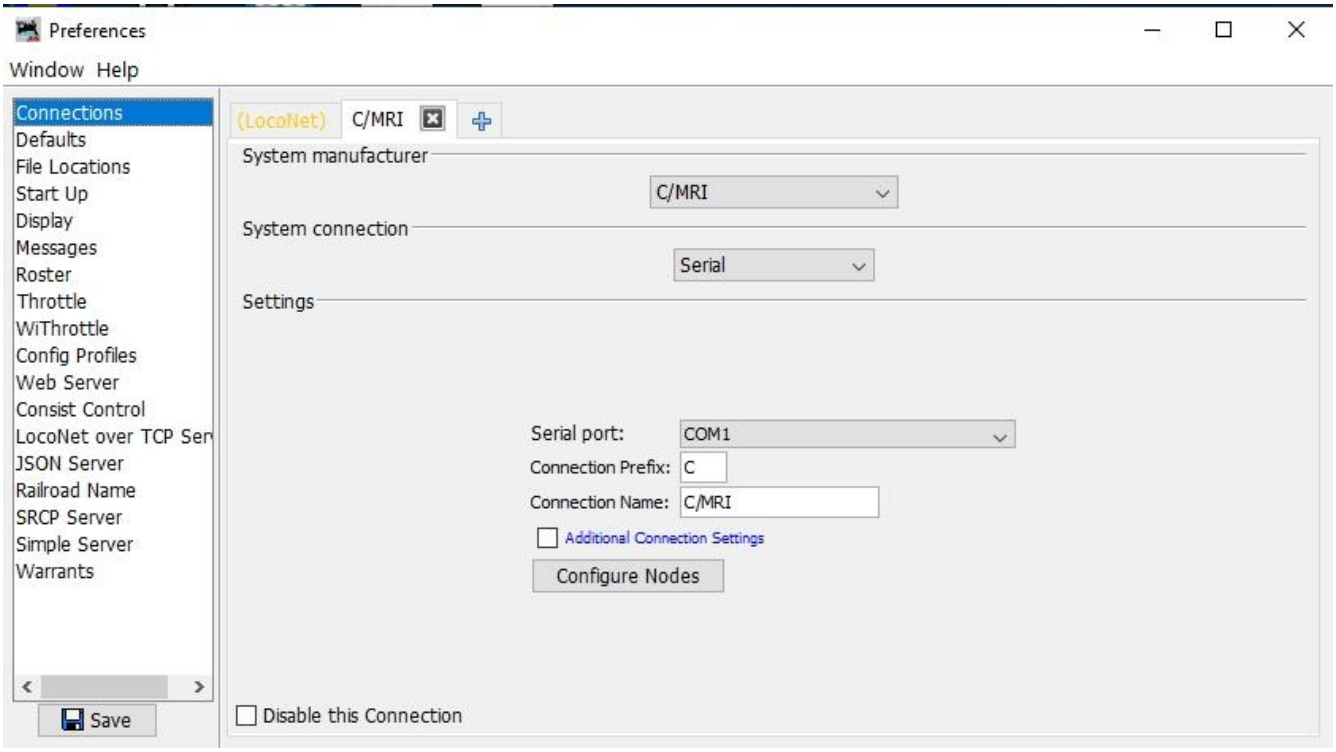
This overview of how to program a C/MRI port is not a comprehensive guide to JMRI but it should be enough information to get you started.



Note that this manual is written using PanelPro version 4.17.3. Some screens may vary depending on your version of PanelPro.

When starting this example your splash screen will not show the C/MRI on COM1. However as we proceed you will be instructed that PanelPro must be restarted and that is when you'll see the C/MRI listing. Note the menu at the top of this splash screen as all references to menu selections will start from this window.

Go to **Edit/Preferences** on this menu to get the following pop-up.



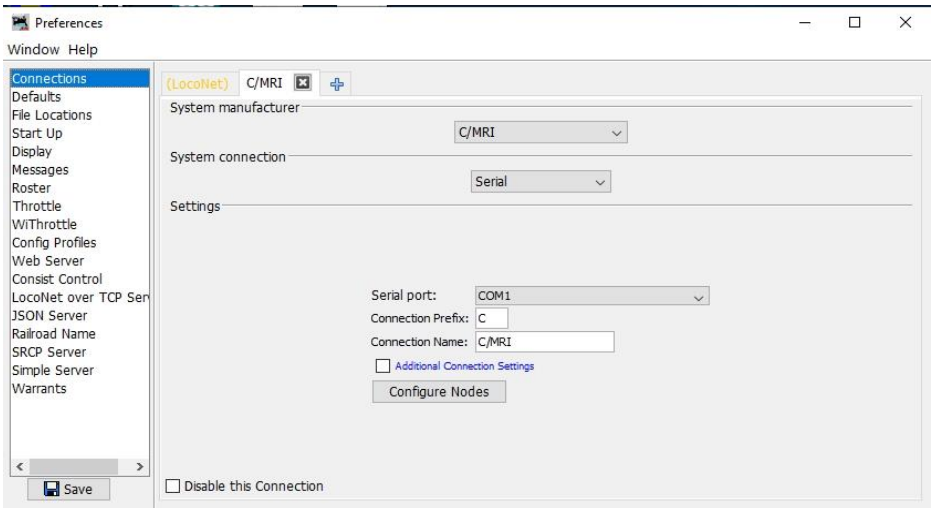
Now select connections from the side-bar menu.
Your window will now have at least two tabs, one is usually named for the DCC system you are using and the other has a big "plus" sign. Click the "plus" sign to add a new connection for JMRI to use.
From the drop-down list boxes select the following:

- System Manufacturer – C/MRI
- System Connection – Serial
- Serial Port – COM1 (this may vary depending on how many COM ports your computer has used.

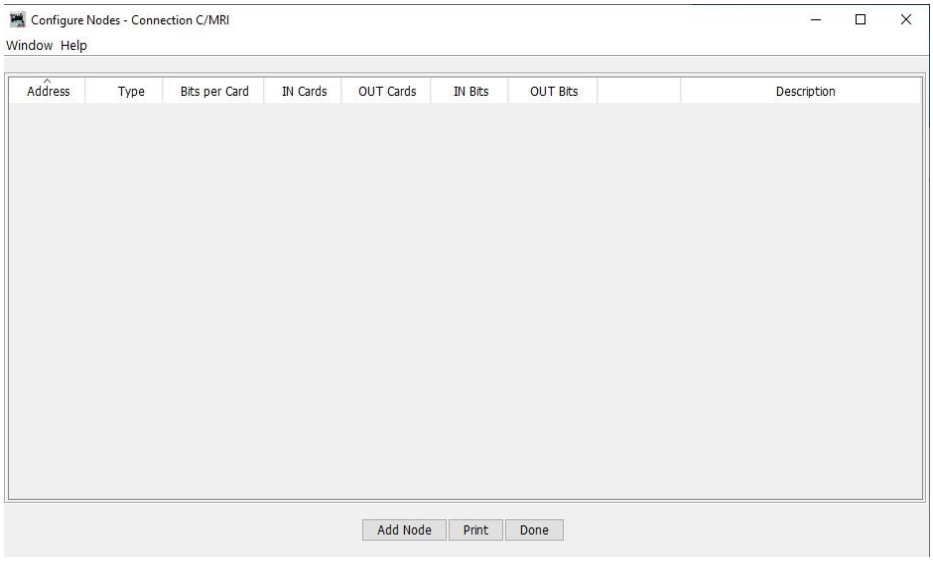
Leave the connection prefix with the default "C."
The Connection Name can be anything you want it to be, I suggest C/MRI as this is the name that will show up on the tab and in other menu choices making it easy to remember what hardware you are talking too.
Additional Connection Settings: set baud rate to 9600

Once you fill in the blanks you must click the SAVE button. A pop-up we tell you that PanelPro must be restarted for the changes to take effect. Answer YES and now you'll see the splash screen as shown above.

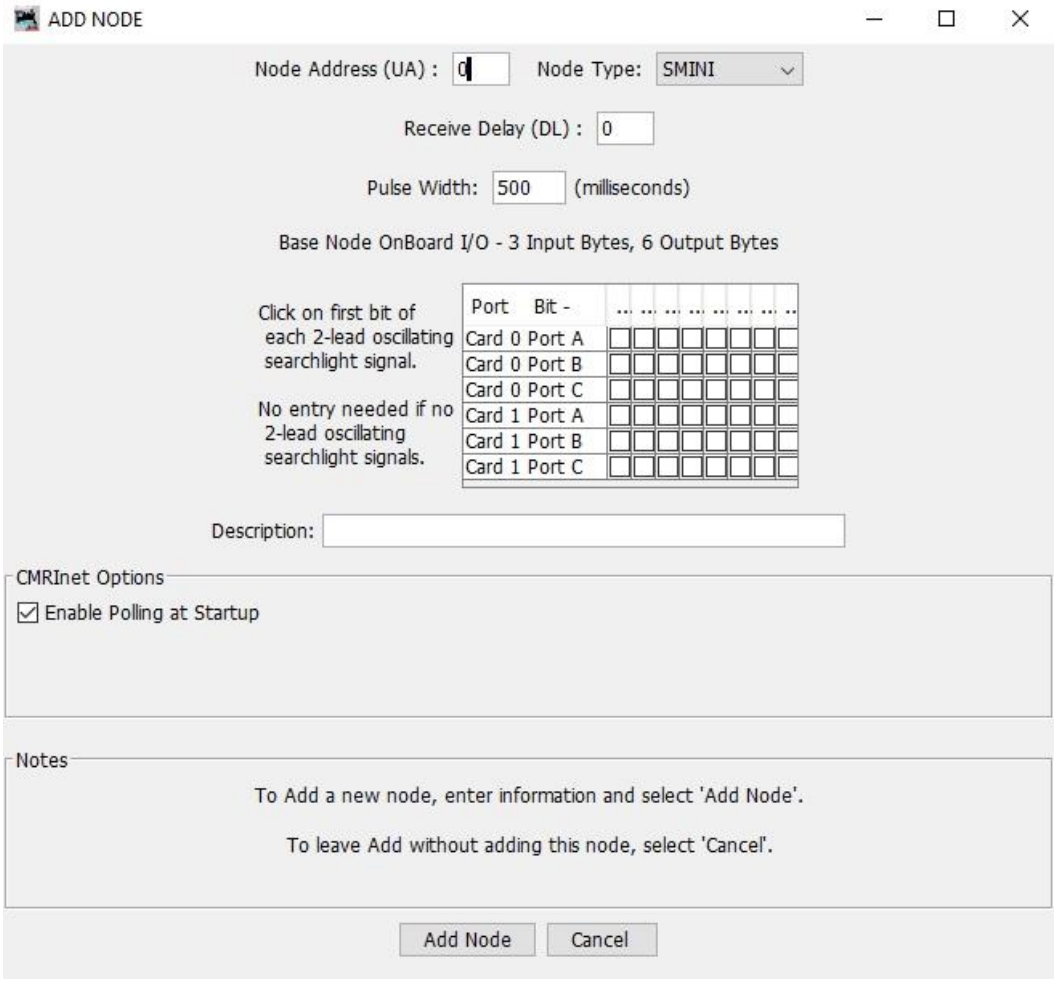
From the splash screen menu go to EDITS/PREFERENCES



Once again select Connections from the side-bar menu. Click the "Configure Nodes" button



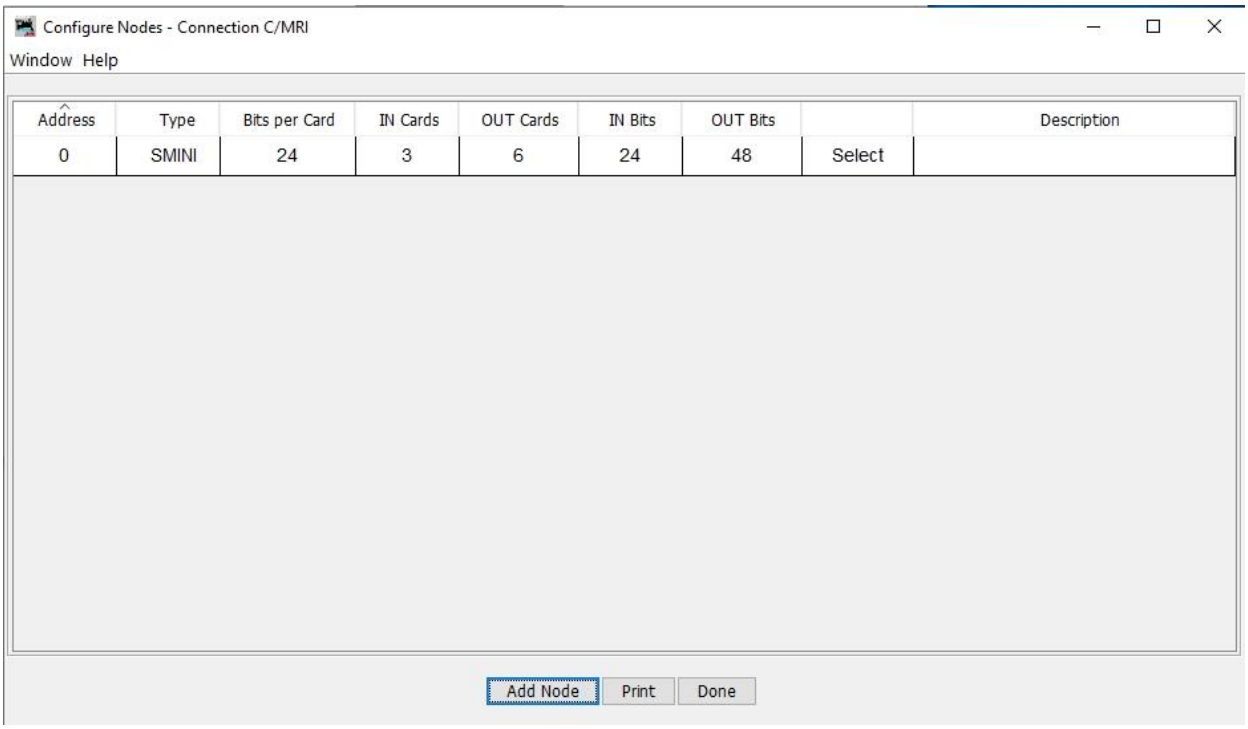
A new blank node list pop-up will be displayed. Click the "Add Node" button to open the next pop-up



C/MRI nodes start at address zero (0) so set that address (the next node if needed would be one (1)) and set the Node Type to SMINI and leave the rest of the boxes set with their default values.

NOTE: You may wish to add a Description to help you remember what this node is used for.

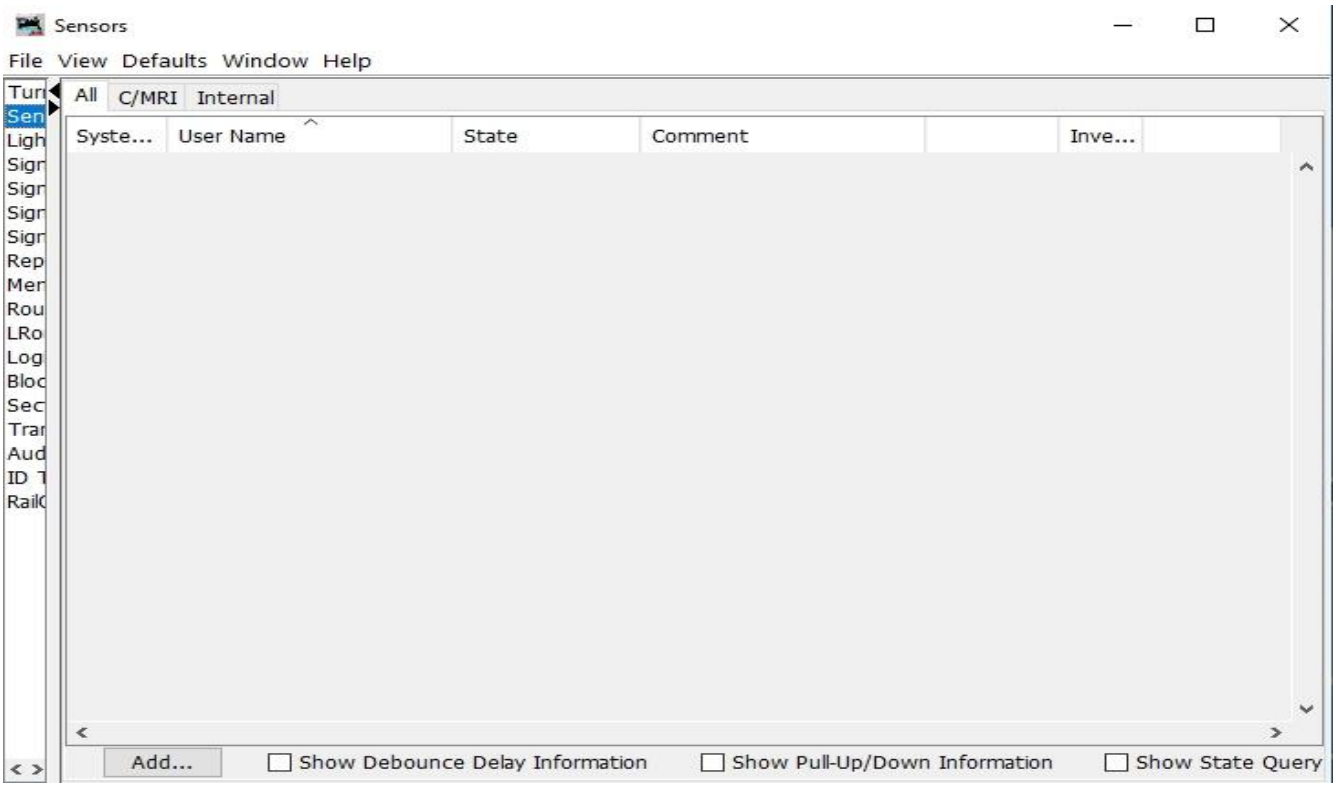
Click to “Add Node” button to close this pop-up



The node configuration pop-up now shows the node you just added. Double check the address (0) and type SMINI before clicking the “Done” button to close this pop-up.

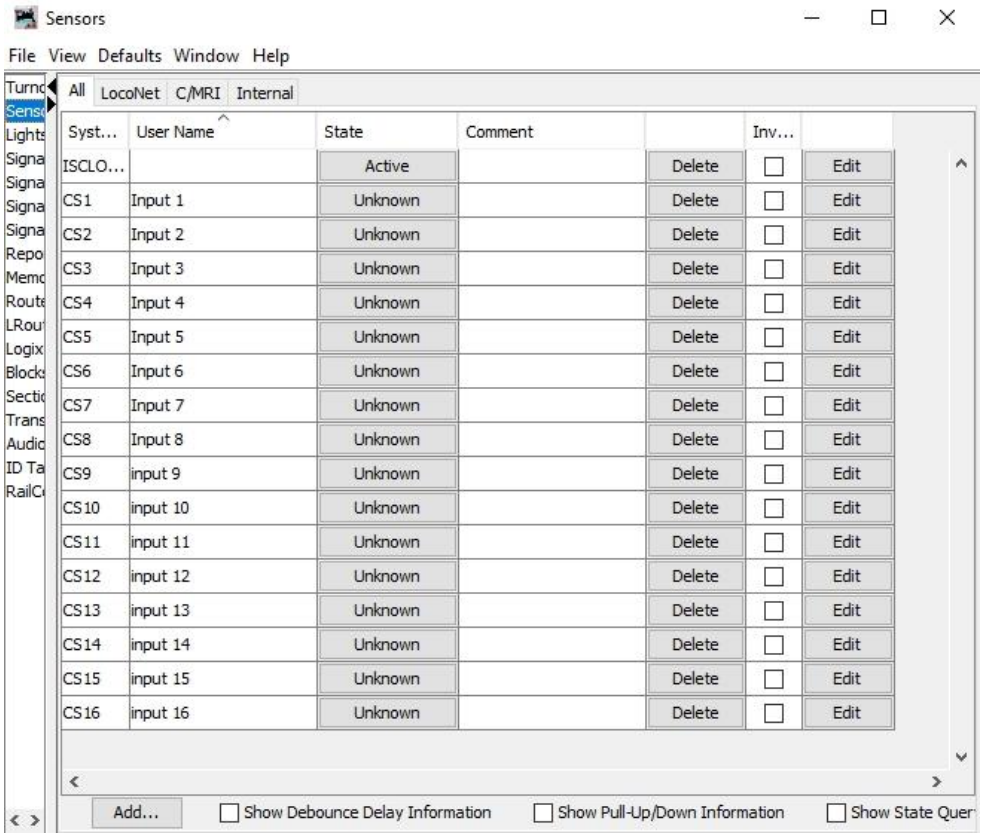
We now have a C/MRI node configured and ready for JMRI to use.

From the splash screen menu select TOOLS/TABLES/SENSORS

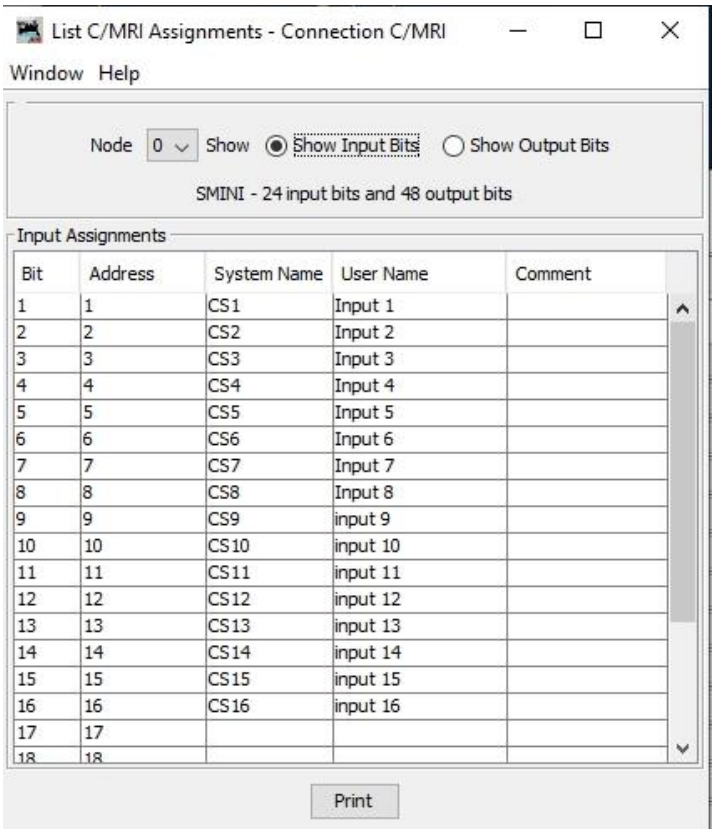


Sensors are input signals for JMRI to use for status displays and logic decisions. We have sixteen (16) inputs on the card to define. Click the “Add” button to get started.

Make sure that the System Connection is set to C/MRI. C/MRI addresses are 4 digits long. The first zero (0) (and sometimes the second) indicates the node being addressed. Bit addresses start with one (1) and again we’ll use only node zero (0) so our addresses will be 0001 – 0016.
We start with Hardware Address 0001 and give it a User Name of our choosing (“Input 1” is my demo name.) Press the “Create” button and the pop-up will automatically close. Repeat this process to add the other fifteen inputs as shown below.


Sensors
File View Defaults Window Help
All LocoNet C/MRI Internal
Syst... User Name State Comment Inv...
ISCLO... Active Delete Edit
CS1 Input 1 Unknown Delete Edit
CS2 Input 2 Unknown Delete Edit
CS3 Input 3 Unknown Delete Edit
CS4 Input 4 Unknown Delete Edit
CS5 Input 5 Unknown Delete Edit
CS6 Input 6 Unknown Delete Edit
CS7 Input 7 Unknown Delete Edit
CS8 Input 8 Unknown Delete Edit
CS9 input 9 Unknown Delete Edit
CS10 input 10 Unknown Delete Edit
CS11 input 11 Unknown Delete Edit
CS12 input 12 Unknown Delete Edit
CS13 input 13 Unknown Delete Edit
CS14 input 14 Unknown Delete Edit
CS15 input 15 Unknown Delete Edit
CS16 input 16 Unknown Delete Edit
Add...
Show Debounce Delay Information
Show Pull-Up/Down Information

From the Splash Screen got to C/MRI/List Assignments



Check that you're on "Node 0"
Click "Show Input Bits"

Here we can see that JMRI has linked its internal "System Name" and "User Name" to bits in the C/MRI hardware.

Arduino Code

```
#include <Auto485.h>
#include <CMRI.h>
//Author: Michael Adams (<http://www.michael.net.nz>)
// Copyright (C) 2012 Michael D K Adams.
// Released under the MIT license.

// sixteen inputs
#define input_1 A5
#define input_2 4
#define input_3 3
#define input_4 A4
#define input_5 A3
#define input_6 A2
#define input_7 A1
#define input_8 A0
#define input_9 12
#define input_10 11
#define input_11 10
#define input_12 9
#define input_13 8
#define input_14 7
#define input_15 6
#define input_16 5

#define DE_PIN 2
Auto485 bus(DE_PIN);

#define CMRI_ADDR 0
CMRI cmri(CMRI_ADDR, 24, 48, bus);

void setup() {
    bus.begin(9600,SERIAL_8N2);

    pinMode(input_1, INPUT_PULLUP);
    pinMode(input_2, INPUT_PULLUP);
    pinMode(input_3, INPUT_PULLUP);
    pinMode(input_4, INPUT_PULLUP);
    pinMode(input_5, INPUT_PULLUP);
    pinMode(input_6, INPUT_PULLUP);
    pinMode(input_7, INPUT_PULLUP);
    pinMode(input_8, INPUT_PULLUP);
    pinMode(input_9, INPUT_PULLUP);
```

```
    pinMode(input_10, INPUT_PULLUP);
    pinMode(input_11, INPUT_PULLUP);
    pinMode(input_12, INPUT_PULLUP);
    pinMode(input_13, INPUT_PULLUP);
    pinMode(input_14, INPUT_PULLUP);
    pinMode(input_15, INPUT_PULLUP);
    pinMode(input_16, INPUT_PULLUP);

}

void loop() {

    // Inputs to C/MRI
    cmri.process();

    boolean a = LOW;
    boolean b = LOW;

    for (int i = 0; i <= 15; i++) {

      switch (i) {
         case 0:
             a = digitalRead(input_1);
             break;
         case 1:
            a = digitalRead(input_2);
            break;
         case 2:
            a = digitalRead(input_3);
            break;
         case 3:
            a = digitalRead(input_4);
            break;
         case 4:
            a = digitalRead(input_5);
            break;
         case 5:
            a = digitalRead(input_6);
            break;
         case 6:
            a = digitalRead(input_7);
            break;
         case 7:
            a = digitalRead(input_8);
            break;
```

```
          case 8:
             a = digitalRead(input_9);
             break;
          case 9:
             a = digitalRead(input_10);
             break;
          case 10:
             a = digitalRead(input_11);
             break;
          case 11:
             a = digitalRead(input_12);
             break;
          case 12:
             a = digitalRead(input_13);
             break;
          case 13:
             a = digitalRead(input_14);
             break;
          case 14:
             a = digitalRead(input_15);
             break;
          case 15:
             a = digitalRead(input_16);
             break;
          default:
             // statements
             break;
       }
       if (a == LOW) {
          b = HIGH;
       }
       else {
         b = LOW;
       }
       cmri.set_bit(i,b);
    }


}
```