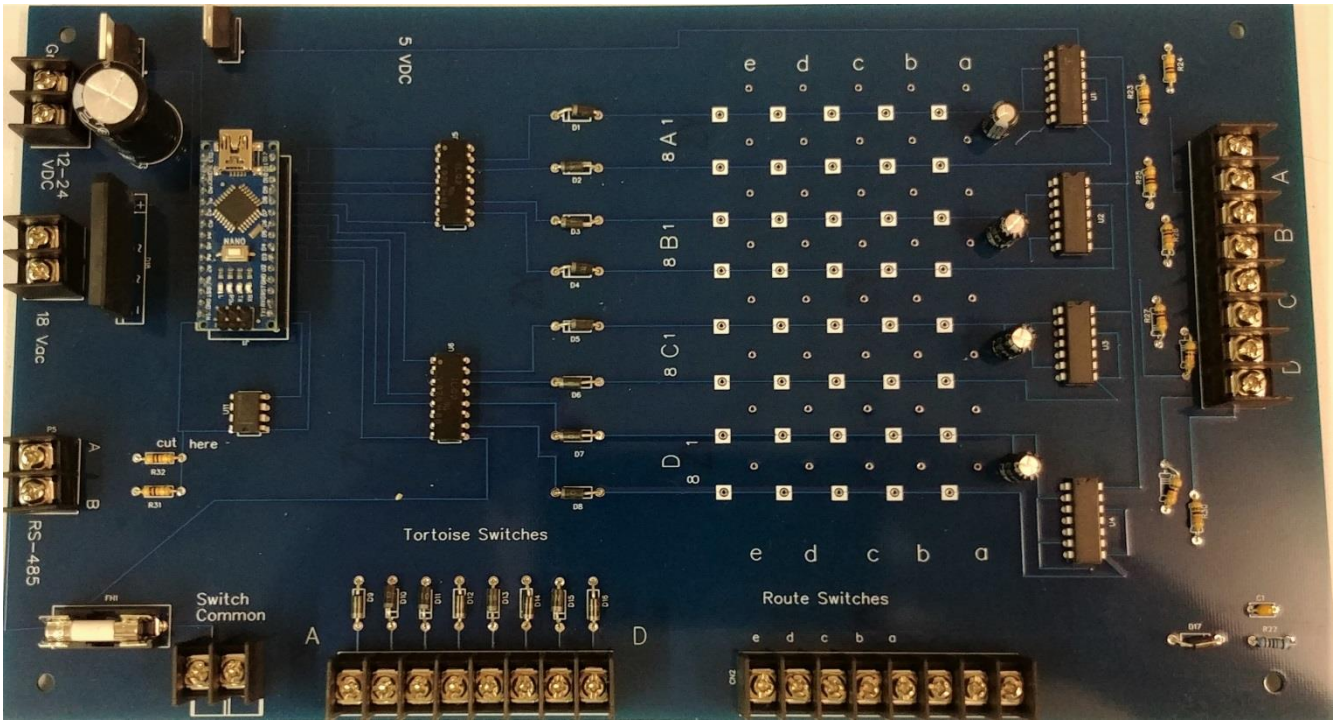
# Tortoise with Matrix Routing Card

This card is designed to provide routing control for four Tortoise type stall motors. The motors may be controlled independently (Tortoise Switches), local routing (Route Switches) or from a computer via the RS-485 connection. It is licensed and distributed under the MIT software license (https://opensource.org/licenses/MIT ) and the TAPR OHL hardware license (https://tapr.org/the-tapr-open-hardware-license/ .)




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 one the these to supply power, never connect both types of power at the same time.**

Just below these terminal strips (shown horizontally) is the 1.6 Amp fuse. This fuse protects the field devices from drawing excessive current and for short circuits.

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.

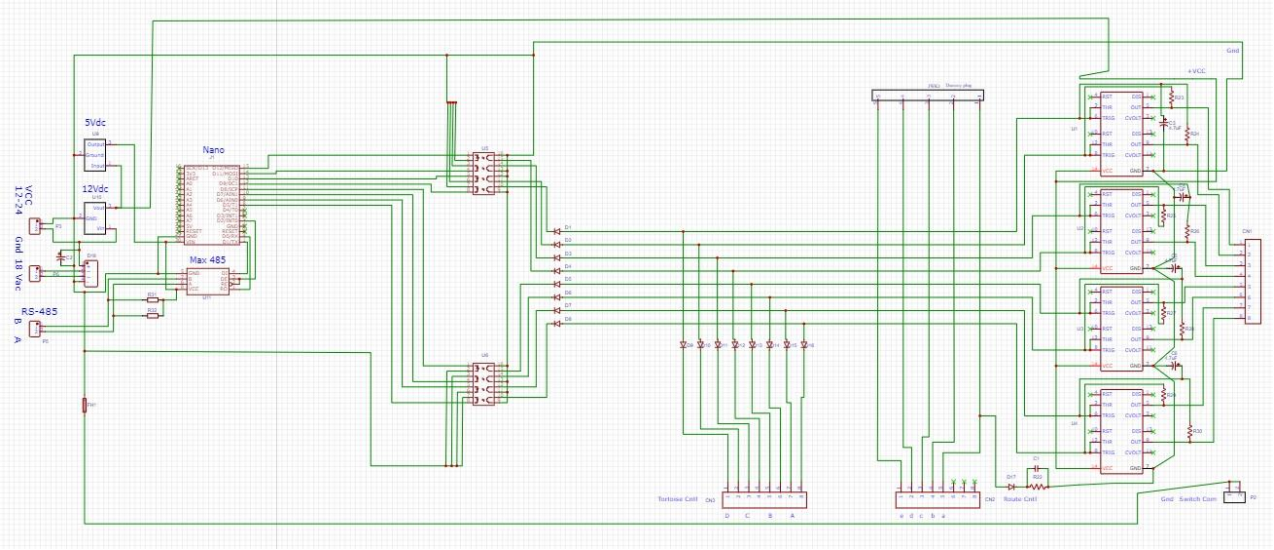
Next to the fuse is the Switch common terminal strip. Route this common (ground) out to all the control switches.

The next two terminal strips provide local control for the motors and routes.

On the right edge are the four output connections to the motors. Note the capital letters A,B,C,D

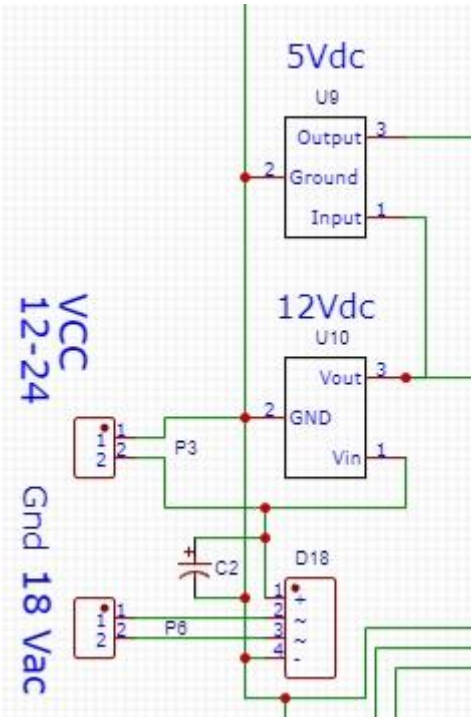
**Schematic Overview:**

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 P6 and connected to a full bridge rectifier D18 (terminals 2 & 3) where it is converted to approximately 25 VDC. The output of the rectifier is applied to a regulator (U10 pin 1) which limits the output (pin 3) to 12 VDC.

The 12 VDC is used in two places:

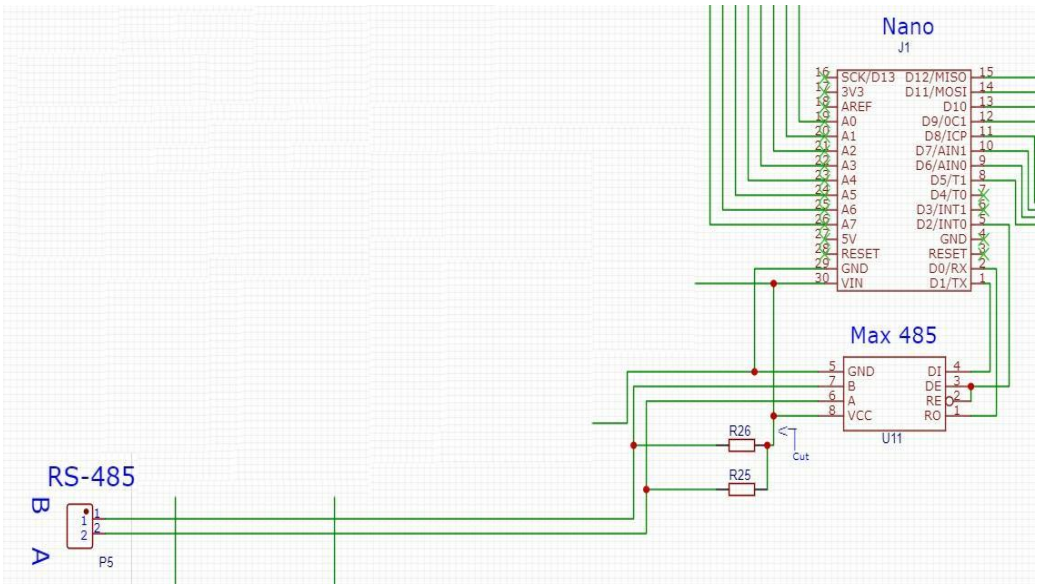
1. Power to control the Tortoise motor
2. The input to U9 (pin 1), the 5 VDC regulator where the output (pin 3) is used for signal LEDs and internal board logic.
3. NOTE: D18-4, U10-2 and U9-2 form a common ground.

12-24 VCC – Is applied to terminal strip P3 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.)

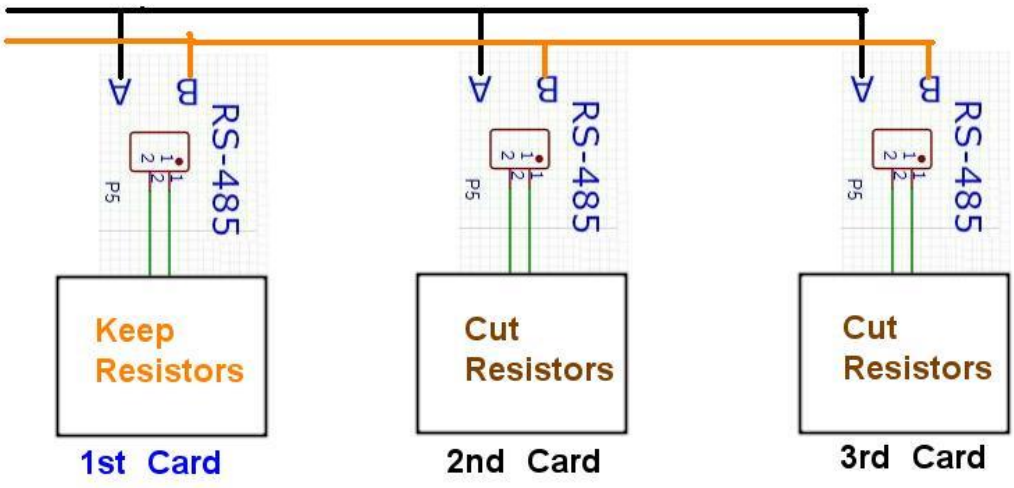



The computer connects at the terminal strip labeled RS-485 and will probably have a USB-to-RS-485 adapter. Make sure to follow the “A” and “B” wiring so that “A” goes to “A” and “B” goes to “B”.

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.

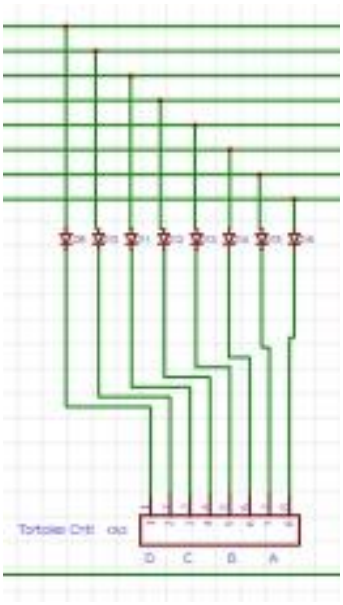



**Ground Switch Com:**


The Signal GRN terminal strip (P2) terminals (1) and (2) are jumpered together on the board. Use these terminals to provide the connection to the common side of the local control switches.

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 5VDC. Nothing on the card will blow this fuse.

**Tortoise Control:**

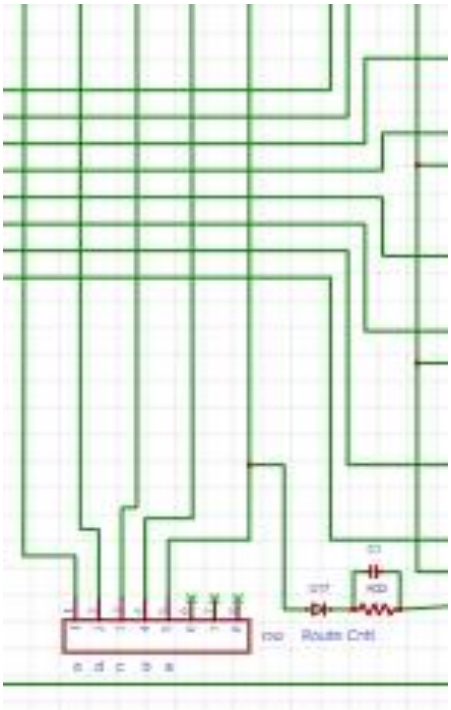


Using a single pole double throw momentary contact switch route the Gnd Switch Com to the common moveable arm of the switch. The two momentary terminals will be connected to the motor control bus. For example motor “A’s” switch will connect to terminates 7 and 8. Now depending on which way the switch is toggled determines the direction the motor will move. The momentary switch does not have to be held for the entire travel time of the motor, a quick pulse of the switch will be captured by the IC chip and will finish the motor movement.

The board will provide local control for four Tortoise type turnout motors. Connect Tortoise terminals (1) and (8) to card terminals (1) and (2) of connector P2. Power for the Tortoise comes from the board.
The Diverging/Mainline flip flop action is created by the IC chip. The Nano chip (not shown) sends commands to the IC chip pins (6) or (8) depending on which turnout setting has been selected. The board also allows manual control of the Tortoise using connections from terminal block CN1 to a single pole / double throw (SPDT) **momentary** switch. Terminal (3 GND) is connected to the common connection on the switch. Terminals (1) and (2) will determine which position the points align too. The momentary switch does

not have to be held for the entire travel time of the motor, a quick pulse of the switch will be captured by the IC chip and will finish the motor movement.

**Route Control:**



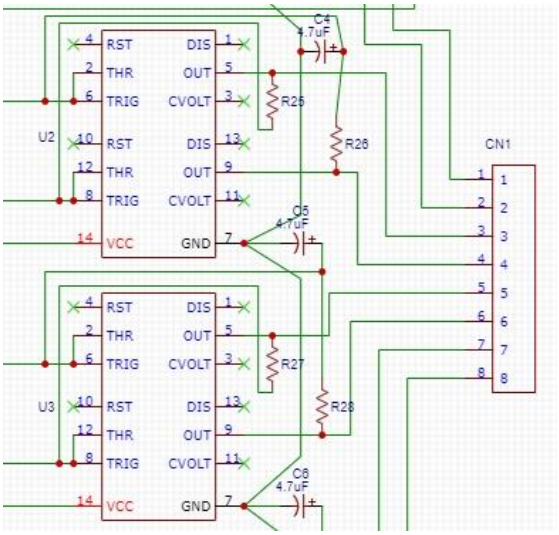
There are five (5) routes available labeled lower case a, b, c, d and e. The wire trace for them is on the bottom side of the board and run perpendicular to the motor control lines. Diodes are used in different combinations to link the proper motor direction for each route (More later.)

Route “a” has an RC circuit that forces this route to be active when power is applied. If this action is not desired cut the wire trace.

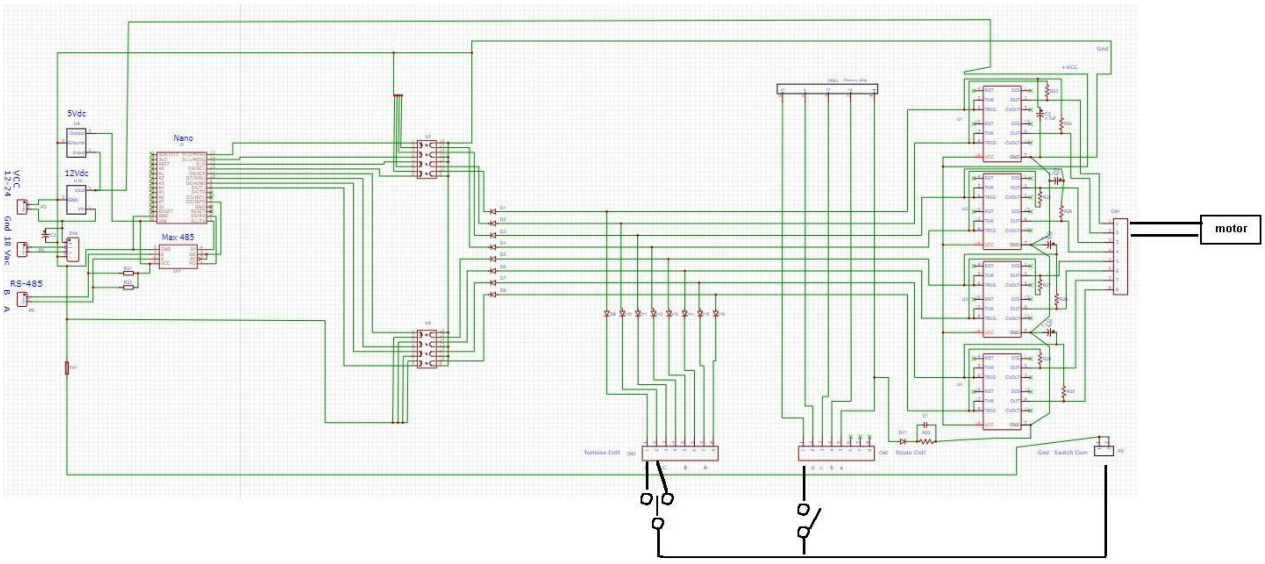
Using a single pole single throw momentary contact switch route the Gnd Switch Com to the common moveable arm of the switch. The momentary terminal will be connected to the route control bus. For example route “a’s” switch will connect to terminate 1. Now depending on which motors have diodes attached to route “a” they will be energized when the switch is closed.

The momentary switch does not have to be held for the entire travel time of the motor; a quick pulse of the switch will be captured by the IC chip and will finish the motor movement.

**Tortoise Outputs:**

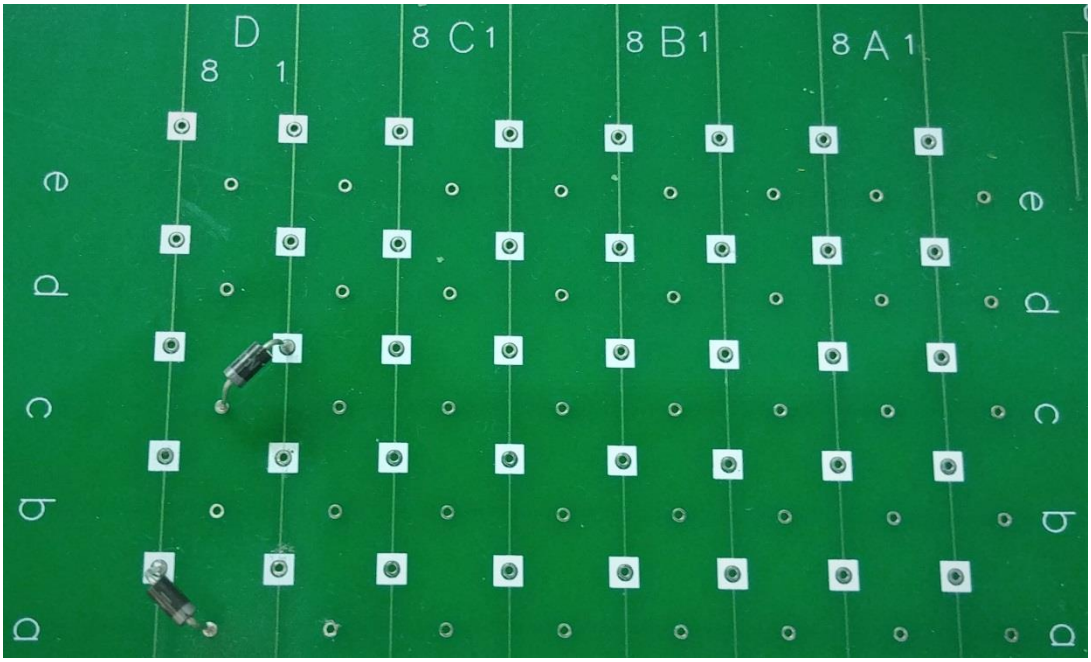



Motor “A” terminates to pins 1 and 2 on CN1. Motor “B” 3 & 4, Motor “C” 5 & 6, and “D” terminate to 7 & 8. Each IC chip has two TRIG pins, these are the control lines that determine which direction the motor will run by reversing the polarity of the two OUT pins that the motor is connected across.




This is an example of a typical tortoise matrix card connection.
Note how the board “Gnd Switch Com” connection is on the common side for all momentary closed control switches.

**Diode Matrix Details:**




The stall motors have capital letters D, C, B, A arranged in vertical rows. Each motor has two control lines labeled 8 and 1, you can see the copper trace connecting each square along the line. The terms 1 and 8 were chosen because they correspond with terminals 1 & 8 on a Tortoise motor (the motor winding.)

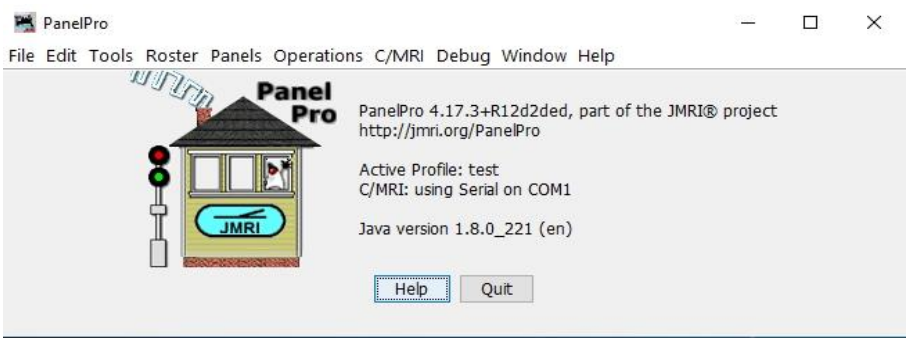
There are five bus horizontal lines labeled with lower case a, b, c, d, e. the copper traces for the bus are on the bottom side of the board so only the small copper circles are visible.

Motor “D” shows how diodes are used to connect a control line to a route. Control line 8 is connected to route “a” by a diode. Control line 1 is connected to route “c” with another diode. Note that the side of the diodes with the silver stripe is connected to the bus side. This is very important as the route matrix depends on the proper diode polarity to work.

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

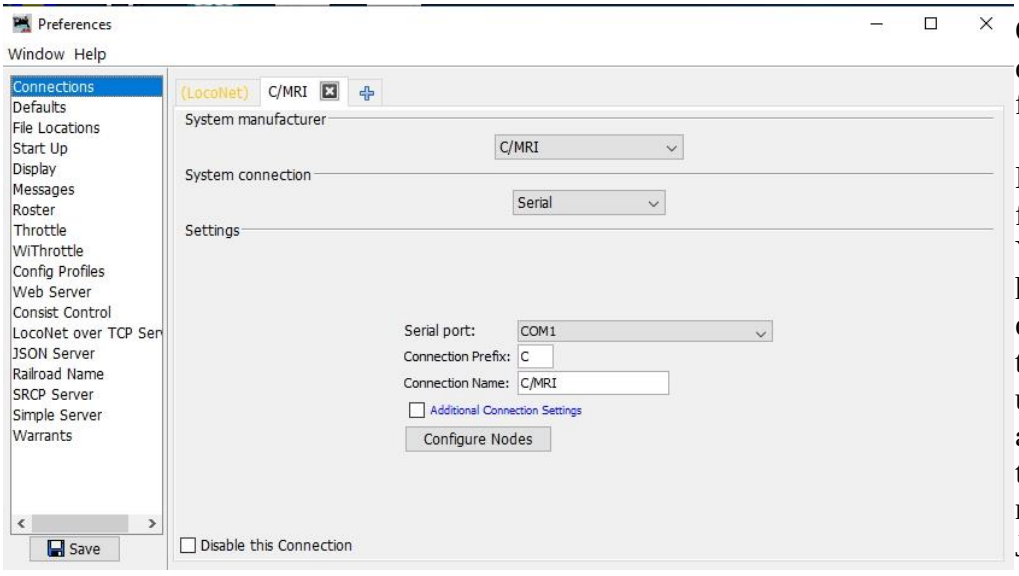
The on-board Arduino chip is programmed to act as a C/MRI SMINI card with the default address of zero (0). Though a true SMINI card can address 24 inputs and 48 outputs the Nano chip cannot handle that much I/O. This Tortoise Matrix Card uses a total of 8 I/O points, of the sixteen (16) available I/O points.

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:

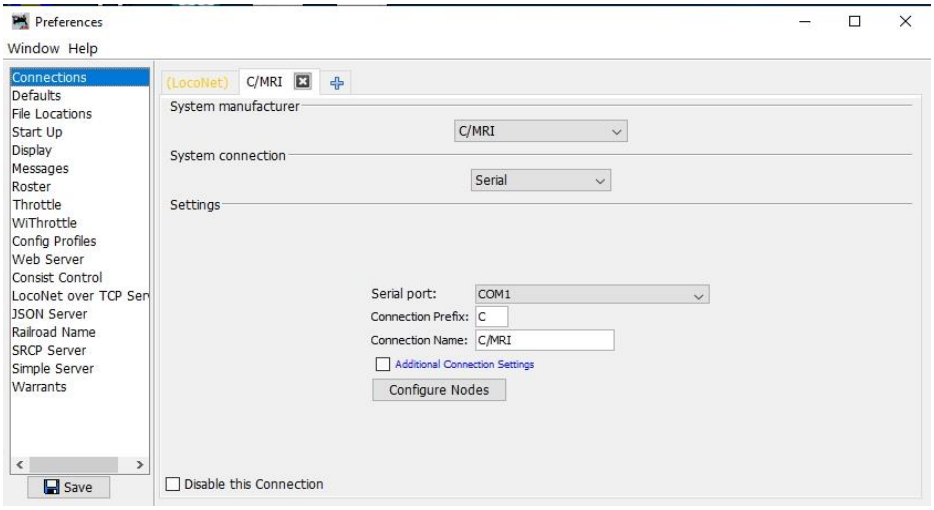
- System Manufacturer – C/MRI
- System Connection – Serial

- Serial Port – COM1 (this may vary depending on how many COM ports your computer has used.

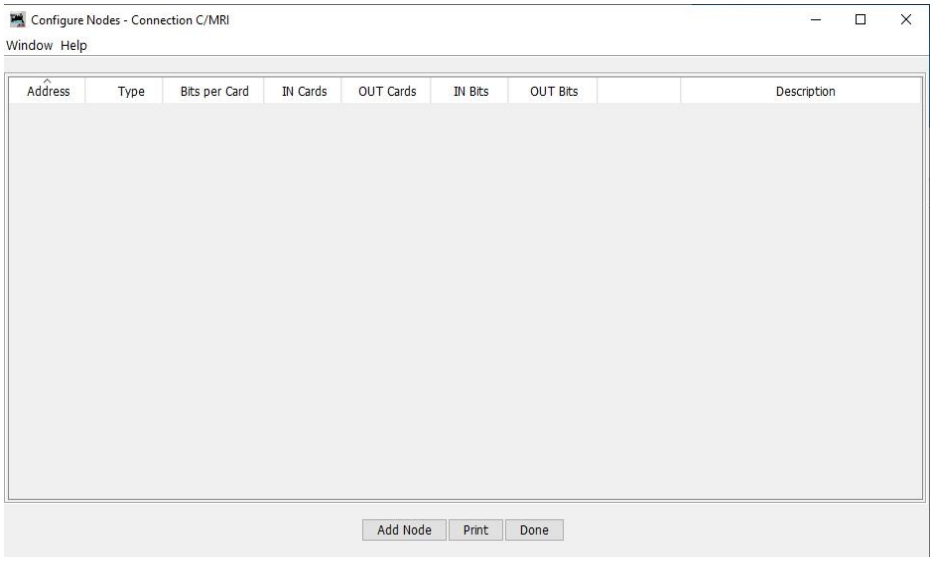
Leave the connection prefix with the default "C."
The Connection Name can be anything you want it to be, I suggest C/MRI as this is the name that will show up on the tab and in other menu choices making it easy to remember what hardware you are talking too.
Additional Connection Setting: set the baud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

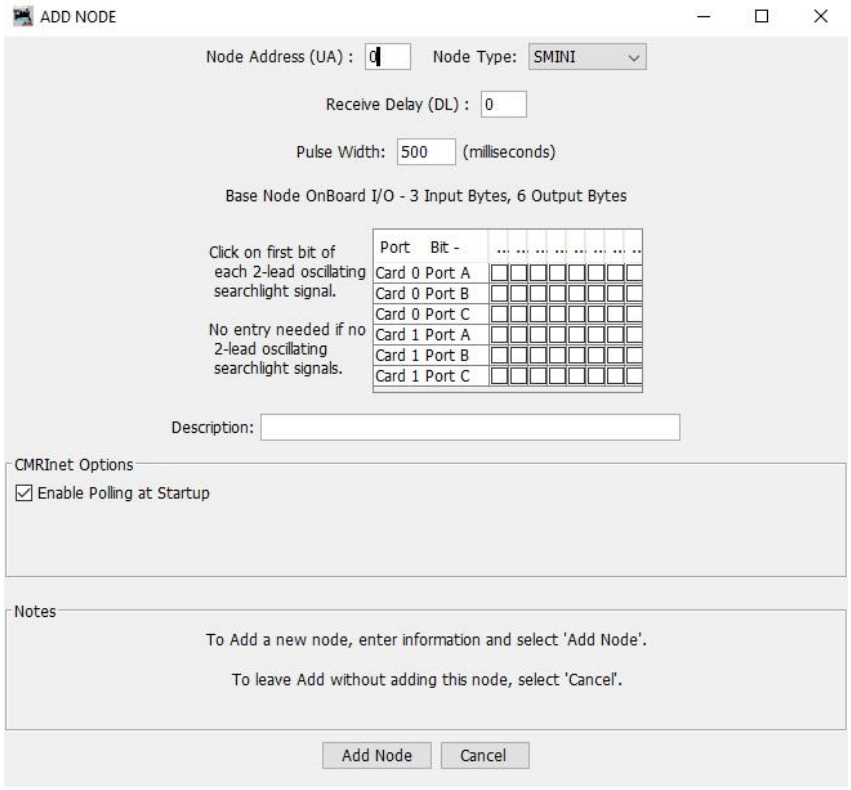
From the splash screen menu go to EDITS/PREFERENCES




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



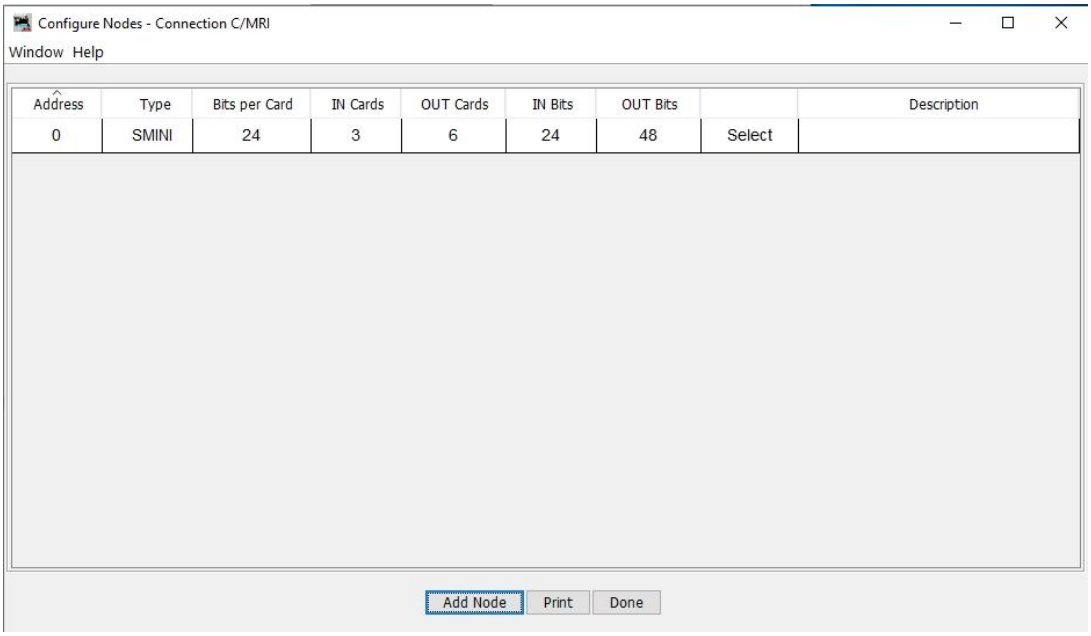

A new blank node list pop-up will be displayed. Click the "Add Node" button to open the next pop-up




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

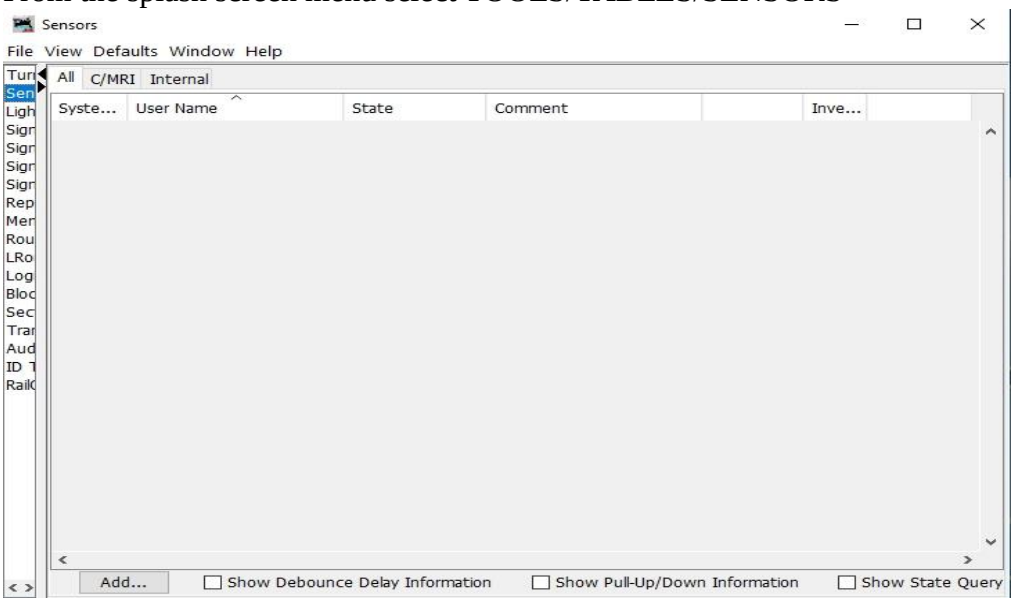



The node configuration pop-up now shows the node you just added. Double check the address (0) and type SMINI before clicking the “Done” button to close this pop-up.

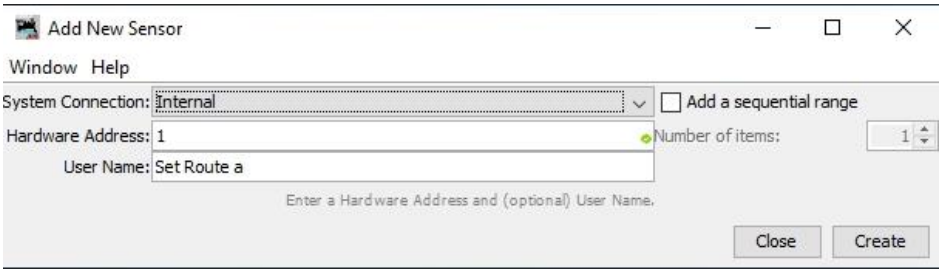
We now have a C/MRI node configured and ready for JMRI to use.

**NOTE: This user manual is written with no other hardware in the system so all addresses start at one (1.) If your system has used some addresses then you must adjust the manual address to the next available address in your system.**

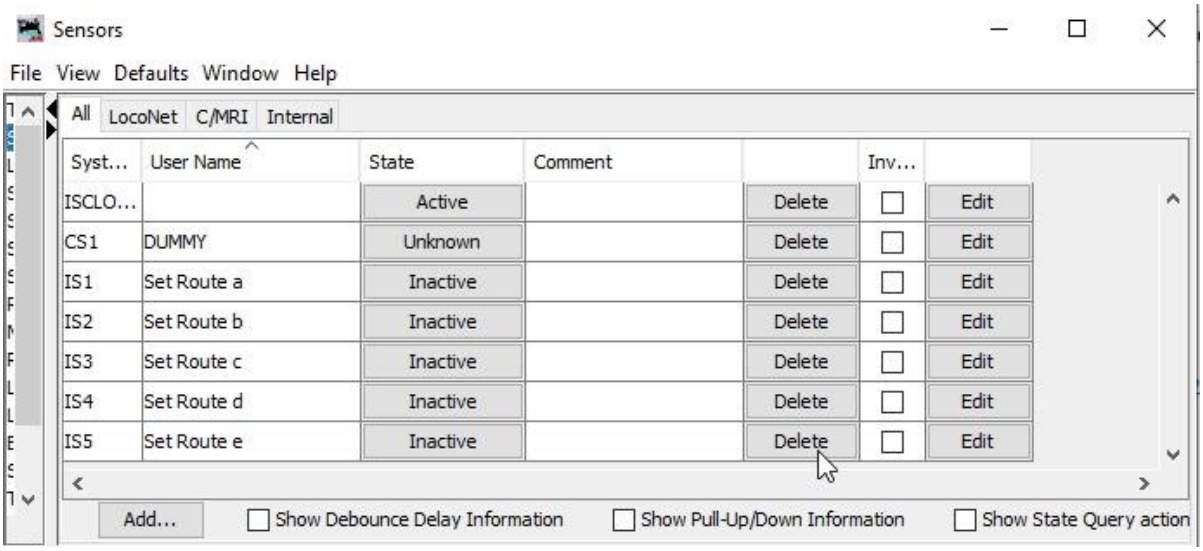
From the splash screen menu select TOOLS/TABLES/SENSORS




Sensors are input signals for JMRI to use for status displays and logic decisions. *We will use five (5) internal sensors to simulate operator momentary push buttons.* Click the "Add" button to get started.



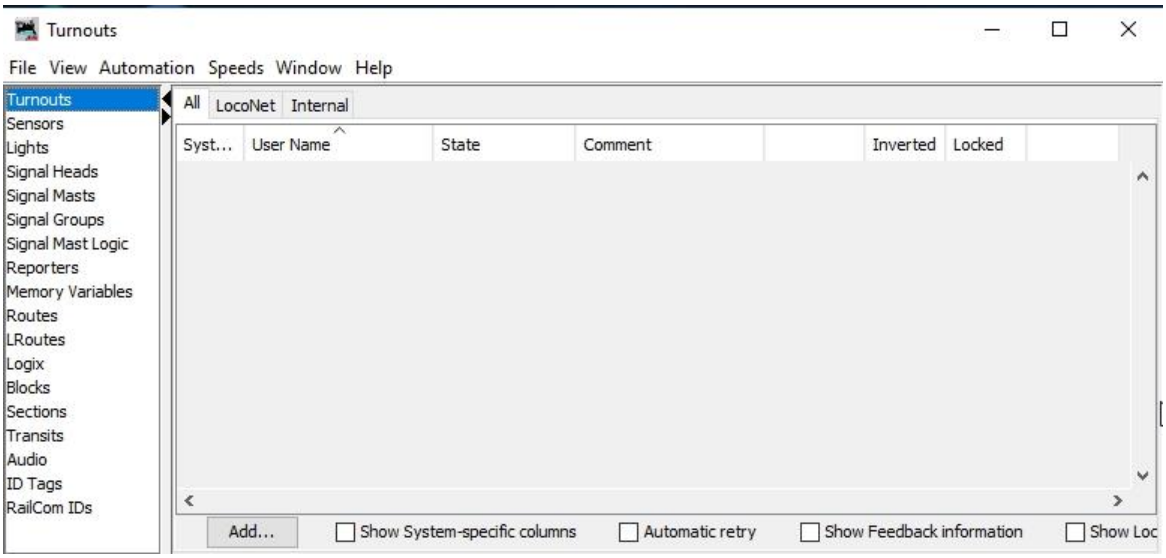

Make sure that the System Connection is set to Internal. We start with Hardware Address 0001 and give it a User Name of our choosing ("Set Route a" is my demo name.) Press the "Create" button and the pop-up will automatically close. Repeat this process to add the other four sensors as shown below.



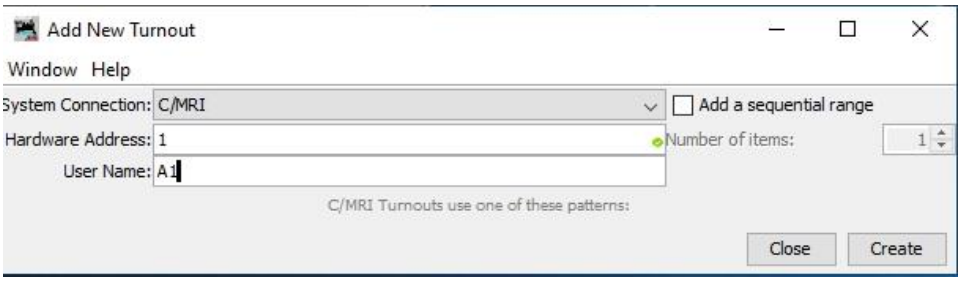

You will test run the card from the Sensors pop-up.
You must simulate the action of the momentary switch by:

- Click the “State” of the sensor (Set Route a) and it will show Active
- Count up to 1003 and click the “State” of the sensor (Set Route a) and it will show Inactive
- Repeat for each route

From the Splash Screen go to TOOLS/TABLES/TURNOUTS



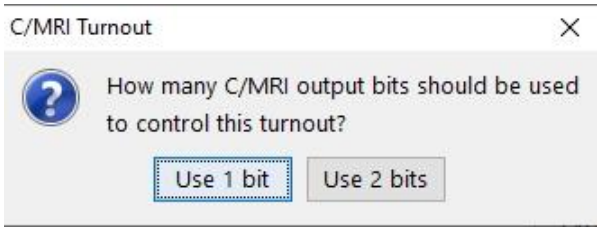

Click the “Add” button



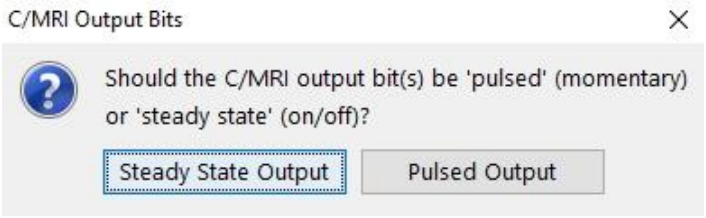

Check that the System Connection setting is C/MRI.
Add a meaningful User Name – A1 (the motor connection we're controlling)
Hardware Address: 1

Click the "Create" button

You then greeted by this pop-up



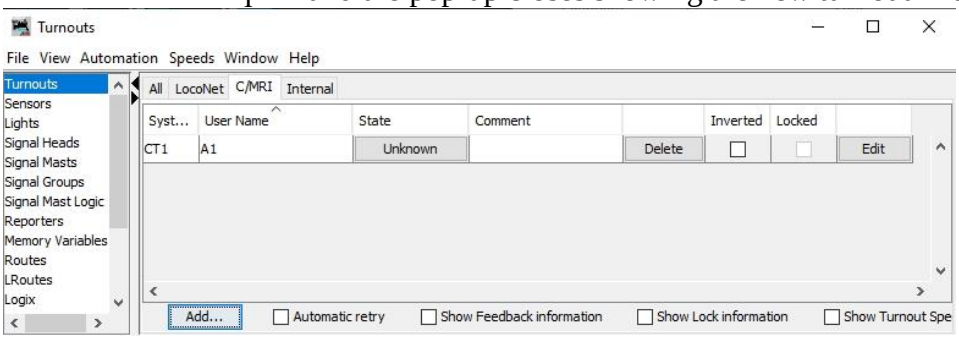

Since we are using Tortoise motors we need two (2) outputs to be able to reverse the motor direction so we select "Use 2 bits."
**NOTE: Since we used 2 bits, addresses 1 and 2 are used for this turnout. The next available address will be bit 3.**



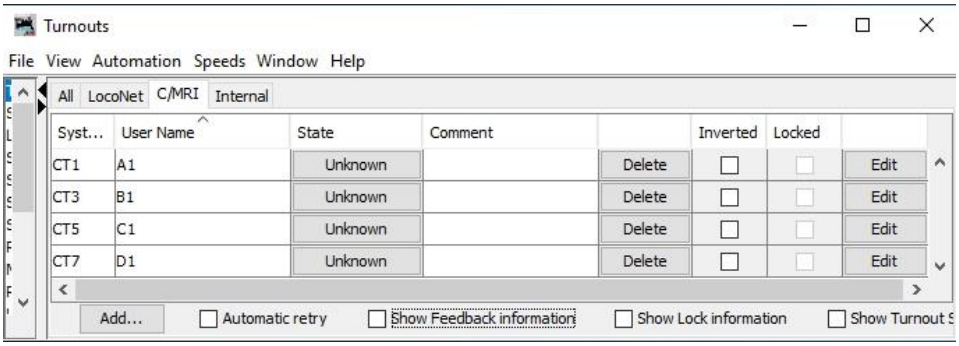

If we select "Steady State Output" the output is set to either On or Off and left in that state until you command it to change. Since we want to allow the operator to choose between using a local fascia mounted switch or a computer output we must release the output state so that the Matrix Board Flip/Flop IC chip can be controlled from either device.

Using the pulsed output setting leaves both bits Off and sending no commands to the turnout (The Matrix Board "remembers" the last commanded state.) This allows the use of the momentary fascia mounted turnout switch to command the turnout and return to the Off state when released.

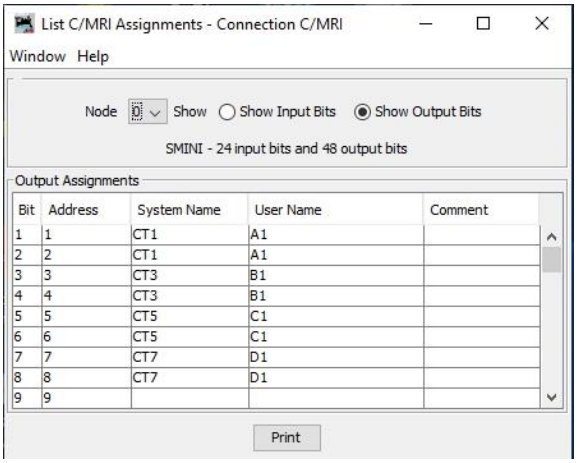
Select “Pulsed Output” and the pop-up closes showing the new turnout in the turnout table.




Repeat this process to add the other three turnouts as shown below.

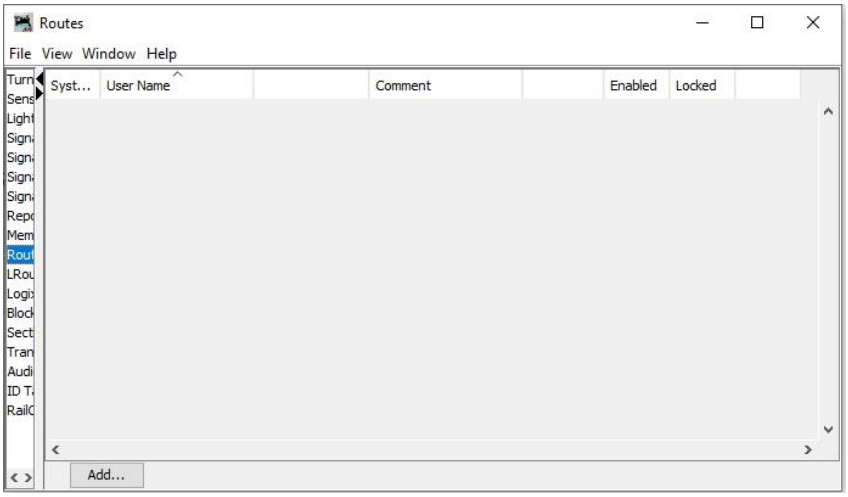



Note System addressing is CT1, CT3, CT5 and CT7
From the splash screen go to C/MRI/List Assignments

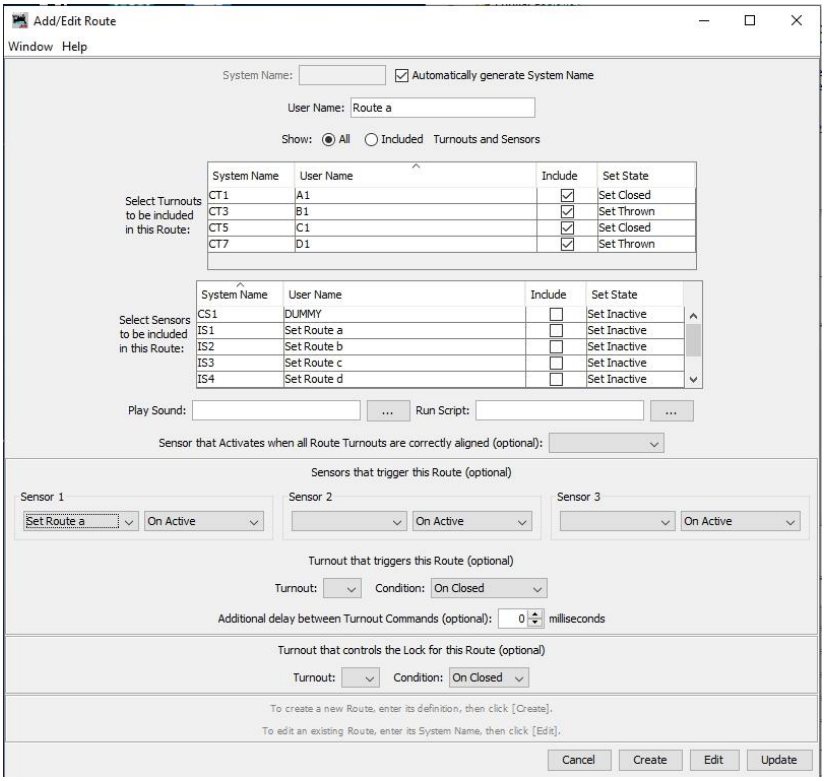



Select Show Output Bits
Note that when JMRI skip address bits it does assigned both bits to the C/MRI output table.

From the splash screen go to Tools/Tables/Routes



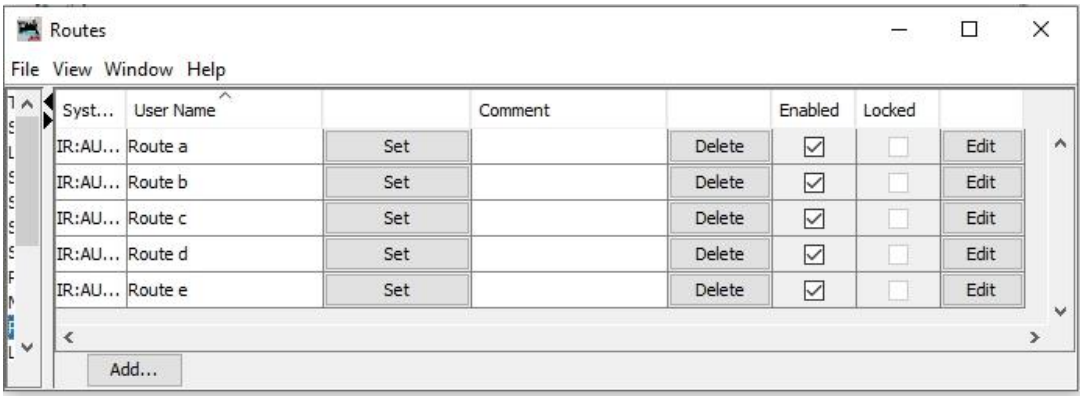

Click the Add button




Check the “Automatically generate the System Name” box.
Give the route a meaningful User Name (Route a” is my demo name.)
JMRI loads a turnout table and a sensor table; these items may be used in a route. As you can see the route options are plentiful and will not be discussed in detail.

In the turnout table check the “Include” box for all 4 turnouts.
Then use the dropdown menu and set the “State” for each turnout
Drop down below the 2 tables and find “Sensors that trigger this Route (Optional)”
Use the dropdown menu for Sensor 1 and select “Set Route a” This is the internal sensor we created to simulate a momentary panel pushbutton.
Leave all other items at the default settings and click the Create button.

The route is now created and added to the route table, but the popup does not close.
Change the popup setting for routes b, c, d and e
Then press the Cancel button and the route table will look like this:




The board test configuration is now complete.

- If you have not followed the popup suggestions to save your work and want to keep this test configuration do the following.

- From the Routes popup (assumed to be still open) go to File/Store/Store Configuration And Panels To File ….
- Use the Store Panels popup and save this configuration (suggested name - Matrix Board Test)
- From the splash screen go to Edit/Preferences
- On the bottom left of the popup click the save button.

You may now test the board one of two ways

- From the Routes popup you can use the Set button for each route
- From the Sensors popup you can use the “State” button to set the sensor to Active. Remember to count up to 1003 and then set the “State” to Inactive. (simulates momentary pushbutton)

**Arduino Code**

```
// Copyright 2020 James W Kelly
// Permission is hereby granted, free of charge, to any person obtaining a copy of this software
// and associated documentation files (the "Software"), to deal in the Software without restriction,
// including without limitation the rights to use, copy, modify, merge, publish, distribute,
// sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is
// furnished to do so, subject to the following conditions:

// The above copyright notice and this permission notice shall be included in all copies or substantial
portions of the Software.

// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
OR IMPLIED,
// INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A
// PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS
OR COPYRIGHT
// HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN
AN ACTION
// OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE
// SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.


#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.

// Asign names to Arduino pins
#define TortoiseA_1 12
#define TortoiseA_2 11
#define TortoiseB_1 10
#define TortoiseB_2 9
#define TortoiseC_1 8
#define TortoiseC_2 7
#define TortoiseD_1 6
#define TortoiseD_2 5

// Asign name to Arduino 485 chip control pin
// and create a bus object
#define DE_PIN 2
Auto485 bus(DE_PIN);
```

```
// ********* change this number (default is 0) to assign
// ********* new address for this CMRI module.
#define CMRI_ADDR 0

// Create a CMRI module object
CMRI cmri(CMRI_ADDR, 24, 48, bus);

void setup() {
    // start 485 communication
    bus.begin(9600,SERIAL_8N2);

    // set Arduino pins to outputs
    pinMode(TortoiseA_1, OUTPUT);
    pinMode(TortoiseA_2, OUTPUT);
    pinMode(TortoiseB_1, OUTPUT);
    pinMode(TortoiseB_2, OUTPUT);
    pinMode(TortoiseC_1, OUTPUT);
    pinMode(TortoiseC_2, OUTPUT);
    pinMode(TortoiseD_1, OUTPUT);
    pinMode(TortoiseD_2, OUTPUT);

    }

void loop() {

     // pass stuff into and out of the CMRI object
     // the object gets and sends data via the bus (485 chip)
     cmri.process();

    // Outputs from C/MRI via JMRI
    // GET the cmri bit setting and WRITE it
    // to the Arduino pin (output)

    digitalWrite(TortoiseA_1, cmri.get_bit(0));
    digitalWrite(TortoiseA_2, cmri.get_bit(1));
    digitalWrite(TortoiseB_1, cmri.get_bit(2));
    digitalWrite(TortoiseB_2, cmri.get_bit(3));
    digitalWrite(TortoiseC_1, cmri.get_bit(4));
    digitalWrite(TortoiseC_2, cmri.get_bit(5));
    digitalWrite(TortoiseD_1, cmri.get_bit(6));
    digitalWrite(TortoiseD_2, cmri.get_bit(7));

}
```