ESP32 T7S3 MultiFunctionUniversalTurnout Kit Instructions

Robert Heller
The Country Robot
Wendell, MA, USA

April 23, 2023

This is a circuit board that supports a Lily Go T7 S3 (mini32) board to manage multiple model railroad functions as a LCC[1] node. This board provides the following functions:

- Four turnout motors with point sense. This board uses one of four turnout driver daughter boards:
  1. Stall-motor drivers, suitable for Circuitron Tortoise and similar slow motion, stall motor switch machines.
  2. Twin-coil drivers, suitable for Atlas or Peco snap action switch machines.
  3. Single-coil drivers, suitable for Kato snap action switch machines.
  4. Servo drivers, suitable for using RC control servos to operate turnouts.
- Four CT coil occupancy detectors.
- Four driver circuits, suitable for indicator LEDs, decoupling electromagnets, solenoids, relays, etc.
- Four Schmitt-trigger button inputs.
- Sixteen signal lamp drivers.

[1] Layout Command Control – see NMRA's 9.7.x standards, available on the NMRA website https://www.nmra.org/lcc.

1 Assembly

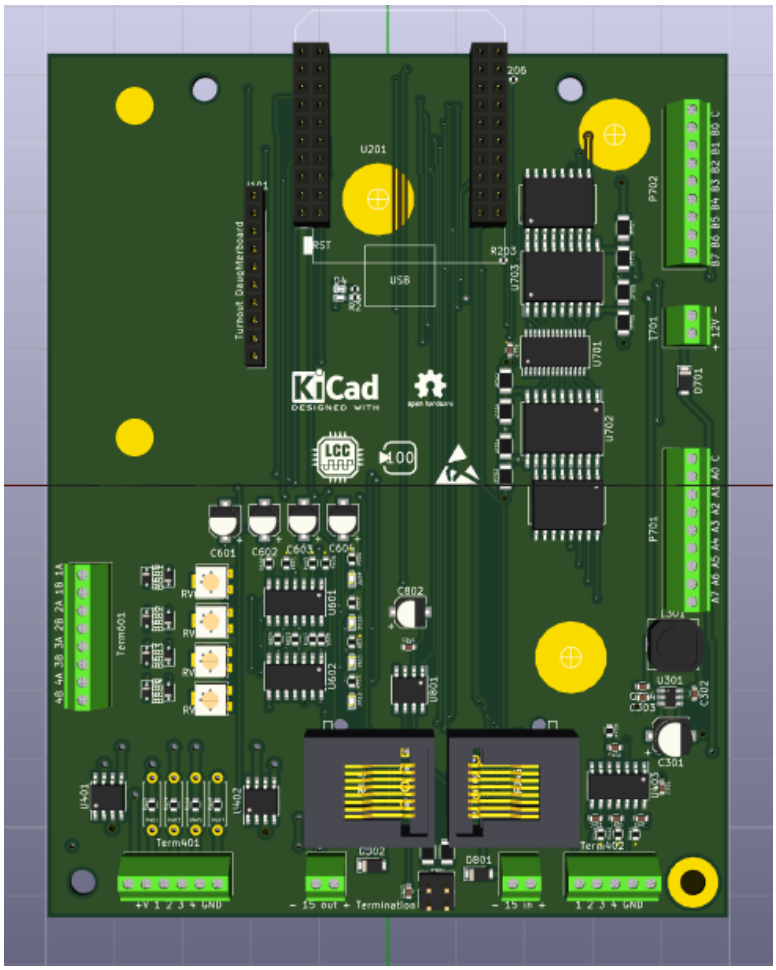

Figure 1: 3D image of the ESP32 T7S3 MultiFunctionUniversalTurnout board

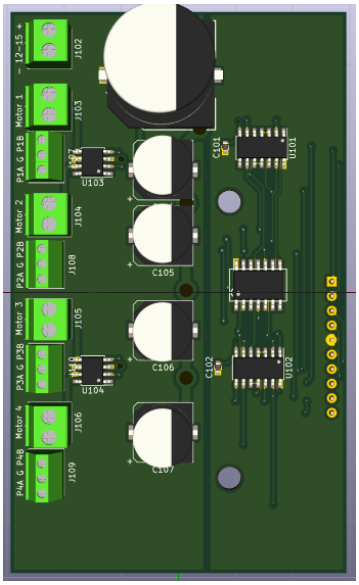

Figure 2: 3D image of the Single Coil Turnout driver daughter board

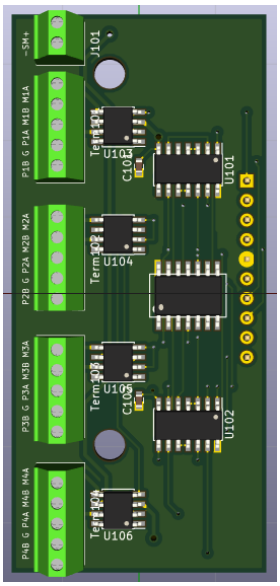

Figure 3: 3D image of the Stall Motor driver daughter board

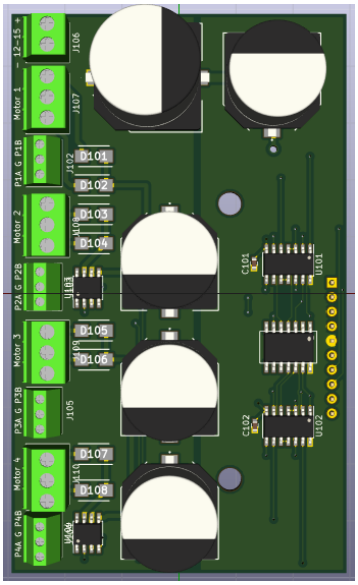

Figure 4: 3D image of the Twin Coil driver daughter board

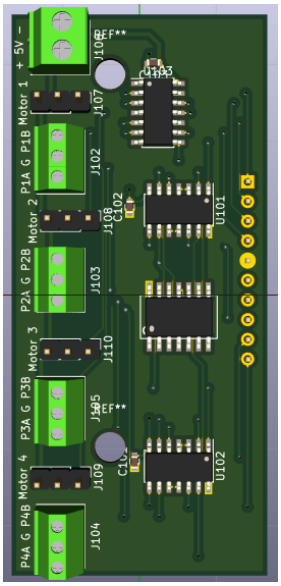

Figure 5: 3D image of the Servo driver daughter board

The boards have all of the SMD parts factory installed. Only the through-hole parts are not soldered to the board. These are the RJ45 Jacks, the Termination Jumper, the turnout daughter board header, the min32 headers, the daughter board support standoffs, and the terminal blocks. The daughter boards also have all of their SMD parts factory installed, and just needs their terminal blocks and board inter-connect pin array headers installed. These are all easy to solder parts. Make sure the headers and terminal blocks are fully seated and square to the boards. A simple trick is to solder one pin and then while pressing the part to the board, reheat that one pin's solder pad and snapping the part to the board and holding it tight and square to the board as the solder cools. Important: make sure the wire entry holes in the terminal blocks are facing the edge of the board! The RJ45 Jacks cannot be installed wrong (they have orientation pins). Start with the shortest parts and work up to the tallest. Note: the pin array on the daughter board goes on the bottom of the board and is soldered on the top side.

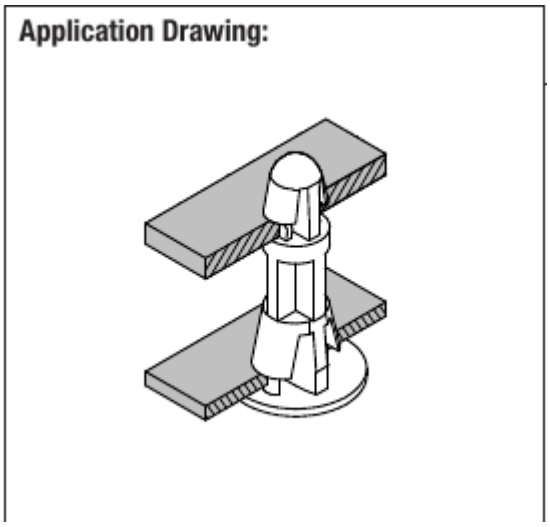

Figure 6: Nylon Standoff Installation

The daughter boards are supported with a pair of nylon standoffs. These are pushed through from the bottom of the main board.

You also need to solder the 2x10 pin headers to the bottom of the T7S3 board.

Once all of the soldering is done, the T7S3 can be installed on the main board. The WiFi antenna extends beyond the top of the board, with the T7S3's USB connector facing into the center of the board. The daughter board's pin array goes in the socket header on the main board and there are two holes in the daughter

board that engage the nylon standoffs.

2 Downloadables and Software Support

As shipped, the TTGO-T7S3 has already had the ESP32-S3-MultiFunctionOpenMRNIDF firmware installed, but if you want to rebuild the code (possibly with customizations) the code is on GitHub in my ESP32-LCC repository: https://github.com/RobertPHeller/ESP32-LCC, in the ESP32-S3-MultiFunctionOpenMRNIDFOpenMRNIDF sub-folder. You will need the Espressif's IoT Development Framework, version 5.0.1 or later, available from https://github.com/espressif/esp-idf. The ESP32-S3-MultiFunctionOpenMRNIDFOpenMRNIDF also uses Mike Dunston's OpenMRNIDF module. Also available in the ESP32-T7S3-MultiFunctionUniversalTurnout/KitBooklet sub-folder are pdfs of the circuit diagrams.

2.1 Building and installing the software

Once you have installed Espressif's IoT Development Framework and cloned the ESP32-LCC repository, you can build the software by cd'ing to the ESP32-S3-MultiFunctionOpenMRNIDF and in bash running these commands in a bash shell:

```
. path-to-Espressifs-IDF/export.sh
idf.py set-target esp32s3
idf.py menuconfig
idf.py build
```

Things to check in the menuconfig (under "Advanced Configuration"):

- Using TTGO-T7S3 Turn this option on.
- Using the servo turnout daughter board If you are using the TS (turnout servo) daughter board, turn this on. If using any of the other turnout daughter boards turn this off.

You can then use the idf.py's flash command to flash the TTGO-T7S3.

2.2 Program Description

The program consists of a main sketch file, and several support files and "components" (libraries).

2.3 Program Startup notes

While most of the common node configuration is accessible using the available CDI configuration tools (JMRI's PanelPro program or MRR SYS OpenLCB program), some of the node's configuration is in separate non-volatile (flash) memory. This includes the node's ID number and some boot options. As installed the node ID is initialized to 05:01:01:01:22:00. **You don't really want to leave this as the node id!** There are two ways to set the node id. One way is to use a CDI configuration tool – the node ID is exposed as a configuration option. Changing the node id forces a "factory reset" on the next boot (and when you update the configuration with a CDI configuration tool, the node will reboot). The other way is during the boot up process. When the node boots, it waits on the USB serial connection for 10 seconds for any "keyboard" response and if it gets a "keyboard" response, it enters a simple command line loop accepting simple commands (mostly single letters). To use this feature you need to connect a a USB cable between the TTGO-T7S3 and a computer (eg a PC or Mac) and then connect to the USB Com port with a terminal program (the Arduino IDE Serial Monitor should work). The commands that the boot startup command line loop supports are:

- **N** Set the node id. Follow the "N" with a 12 digit hex number, with optional periods or colons. Causes a Factory Reset.
- **E** Reset events on boot up.
- **F** Force a Factory Reset.
- **S** WiFi ssid[2]. Enter the ssid after the "S", leading and trailing spaces are stripped off.
- **P** WiFi password[2]. Enter the password after the "P", leading and trailing spaces are stripped off.
- **H** Hostname prefix[2]. Enter the hostname prefix after the "H", leading and trailing spaces are stripped off. The NODE ID in hex is appended. Hostnames are limited to 32 characters.
- **W** Enable WiFi[2]. Enter "YES", "NO", "ON", or "OFF" after the "W".
- **T** Test signal lamps[3].

[2] Only useful if WiFi is enabled.

[3] Each of the signal lamps is flashed for 1 second during boot up.

- **R** Resume.

3 General Wiring Notes

3.1 Main board

There are various terminal blocks, connectors, and jumper blocks on this board.

On the left side is a 10 position socket header for the turnout daughter board and an 8 position terminal block for the four CT coil occupancy detectors, Along the bottom is the 6 position terminal block for the driver outputs, then come the terminal block to extracting power from the LCC bus. Then there is the termination jumper block for the LCC bus. Then there is terminal block for injecting power into the LCC bus. Above the LCC power terminal blocks are a pair of RJ45 connectors. These are for connecting the board to the LCC bus. These connectors are wired in parallel. On the right side of the board are two 9 position terminal blocks for the signal lamp LEDs. Finally, between the terminal blocks for the signal lamp LEDs is a two position terminal block to (optionally) provide power for the signal lamp LEDs.

3.1.1 Turnout Daughter board connector

Starting on the upper left, there is a 10 position socket header for the turnout daughter board. The socket mates with the 10 position pin array on the underside of the Turnout Daughter board. The Turnout Daughter boards' terminal blocks are described below in Sections 3.2, 3.3, 3.4, and 3.5.

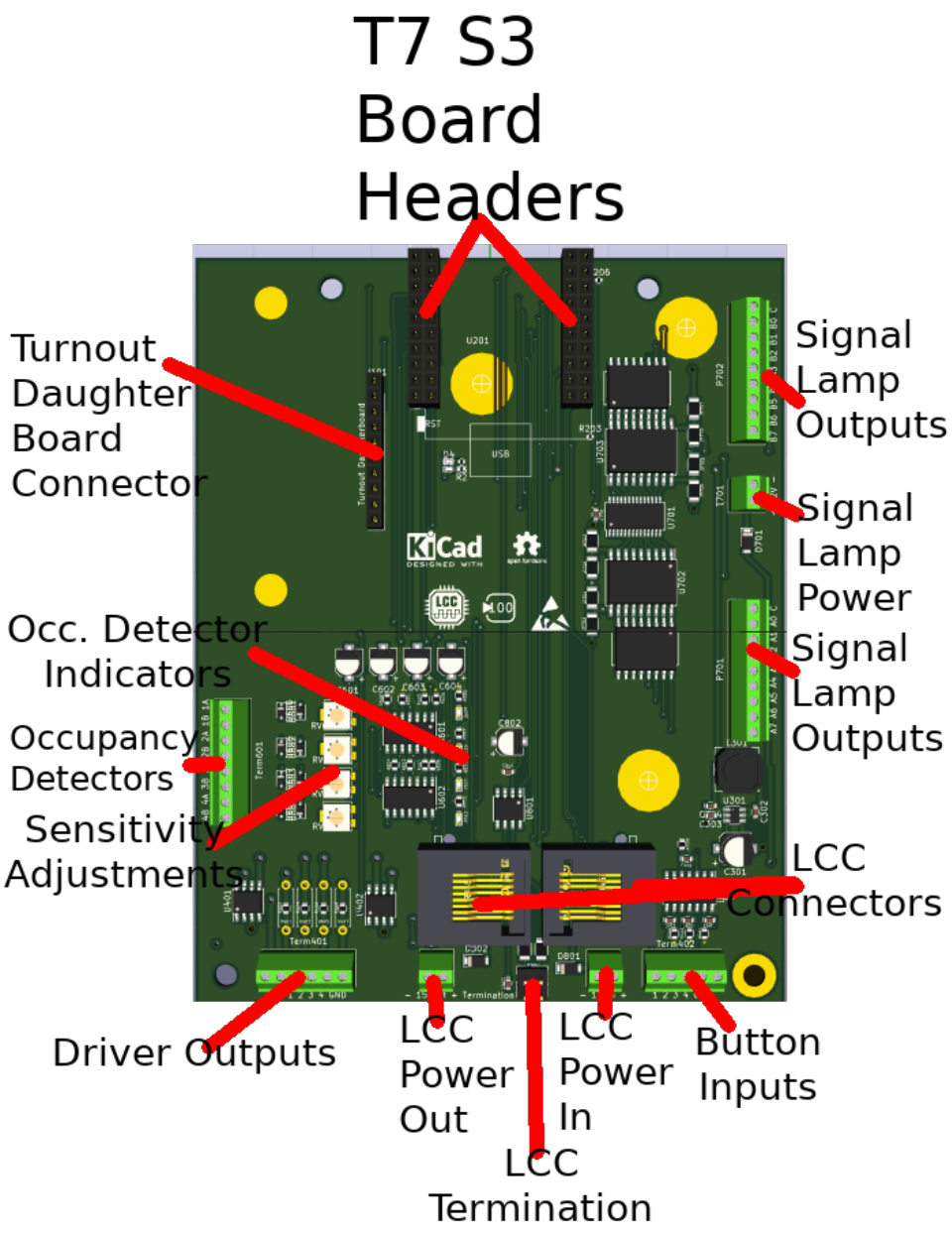

Figure 7: ESP32 T7S3 MultiFunctionUniversalTurnout board, annotated

3.1.2 Occupancy Detectors

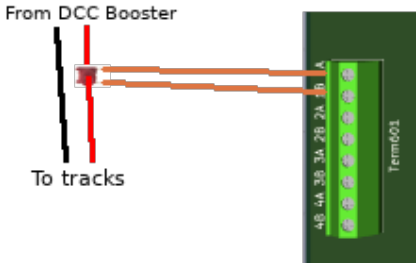

Figure 8: Occupancy Detector Wiring

Then on the lower left side is an 8 position terminal block for the four CT coil occupancy detectors. Each adjacent pair of terminals is for one twisted pair cable to one CT coil for one feeder wire for one block. There is a sensitivity adjustment trimmer and an indicator LED for each detector.

3.1.3 Driver Outputs

There is a 6 position terminal block for the four driver outputs. There are 1500 Ohm load resistors on board, which are suitable for driving LEDs with a 12V supply. There are places for adding through hole load resistors to allow for increased load current. The driver outputs require an external power source, typically 12V. The driver chips are totem pole (push-pull), so can drive both high side and low side, up to 1.5 Amps.

3.1.4 LCC Power

Power can be optionally injected into the LCC bus or extracted from the LCC bus. Power can be injected to into the LCC bus to power this and other boards. Power can also be extracted to power local devices.

3.1.5 LCC Network Connectors

There are two RJ45 connectors for the LCC Network. Connect a CAT5 or CAT6 (Ethernet) cable of at least 1 foot in length to the next and previous node in the network. If this is the last node, only one connector is used and termination jumpers should be installed. See Section 3.1.8.

3.1.6 Button (or switch) Inputs

There is a 5 position terminal block for button (or switch) inputs. These are Schmitt-trigger inputs and are referenced to ground (supplied on the fifth position of the terminal block).

3.1.7 PWM Signal Lamp drivers

There are two 9 position terminal blocks for the sixteen PWM Signal Lamp outputs (two banks of 8) and there is a two position terminal block to provide (optional) external power for the signal lamps. These terminal blocks are on the right side of the board. The boards can be built to be either for common anode or common cathode.

3.1.8 LCC Termination

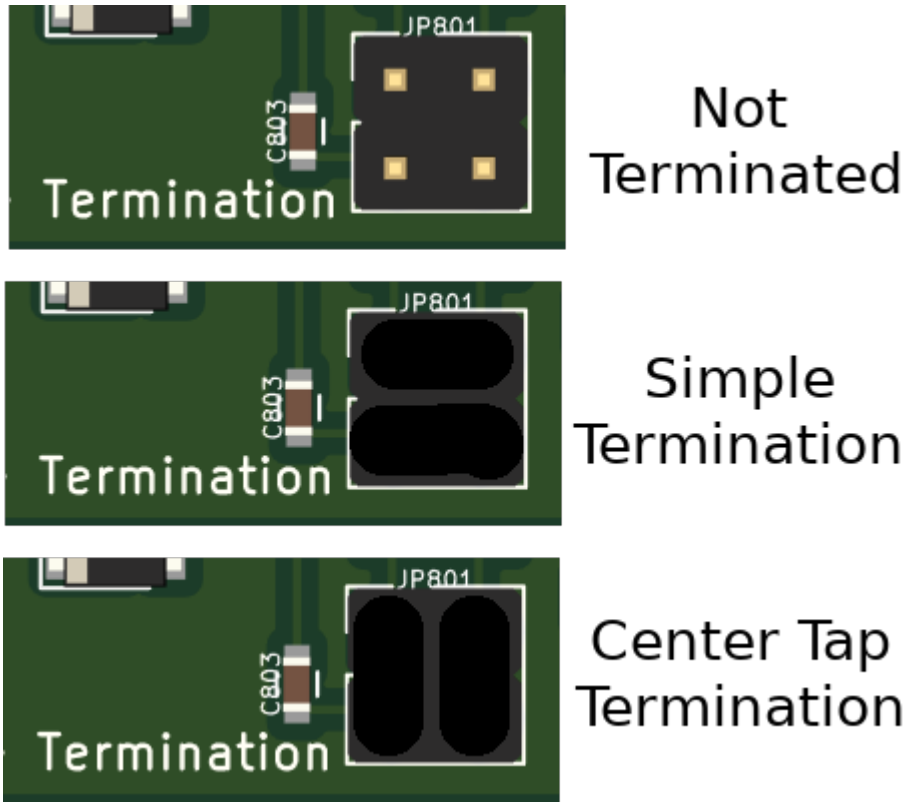

Figure 9: Termination Options

There are three options for termination: none, simple, and center tap, as shown in Figure 9.

3.2 Stall Motor Turnout daughter board

The Stall Motor Turnout daughter board can drive Circuitron Tortoise and similar slow motion, stall motor switch machines. There is a 2 position terminal block for a 12-15 volt supply to drive the stall motors and four 5 position terminal blocks, one for each motor. Each 5 position terminal block has motor A and B connection for the motors and a point A and B and a ground (G) connection for the point sense. The motor A and B connections go to the Tortoises 1 and 8 pins, and the

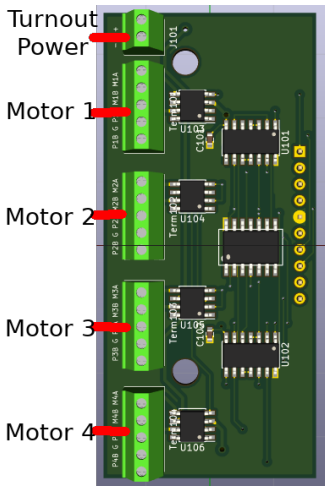

Figure 10: Stall Motor Turnout daughter board, annotated

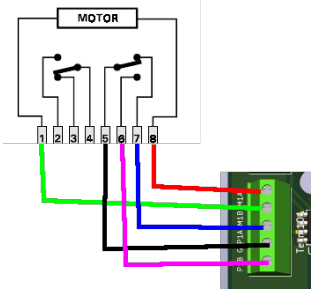

Figure 11: Stall Motor Turnout daughter board, wired to a Tortoise.

other three can be connected to one of the poles of the internal switch contacts to report point sense – the G to the common.

3.3 Servo Turnout daughter board

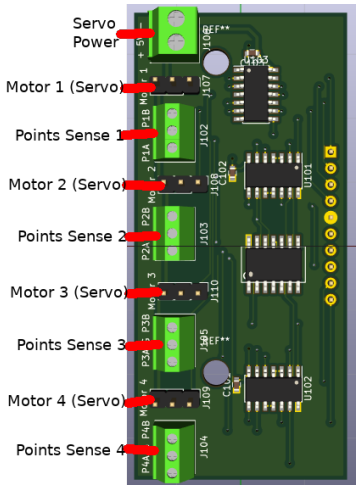

Figure 12: Servo Turnout daughter board, annotated

The Servo Turnout daughter board can drive standard 5V servos. There a 2 position terminal block for a 5 volt power supply to drive the servos, four 3-pin headers for standard servo connectors, along with four 3 position terminal blocks for point sense.

3.4 Single Coil Turnout daughter board

The Single Coil Turnout daughter board can drive single coil snap action turnouts, typically Kato turnout motors. This a capacitive discharge type driver circuit. There is a 2 position terminal block for a 12-15 volt power supply to charge the capacitors. There is a larger 2 position terminal block for each coil, along with a smaller 3 position terminal block point sense for each of four turnouts.

3.5 Twin Coil Turnout daughter board

The Twin Coil Turnout daughter board can drive twin coil snap action turnouts, such as Atlas or Peco switch machines. This a capacitive discharge type driver

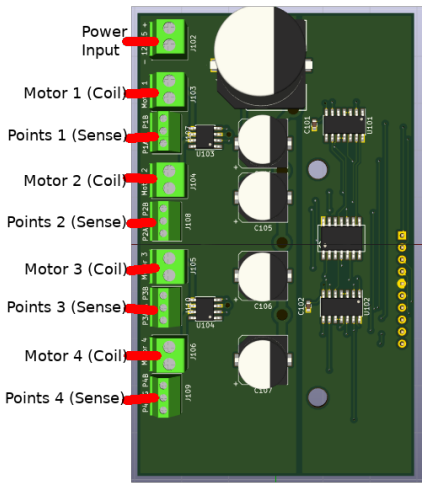

Figure 13: Single Coil Turnout daughter board, annotated

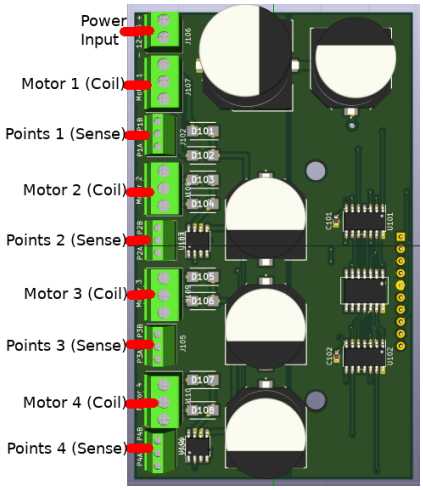

Figure 14: Twin Coil Turnout daughter board, annotated

circuit. There is a 2 position terminal block for a 12-15 volt power supply to charge the capacitors. There is a larger 3 position terminal block for each pair of coils, along with a smaller 3 position terminal block point sense for each of four turnouts.

4 Application Notes

In this section I will present four applications:

1. One end of a siding with local (facia panel) and remote (eg dispatcher or CTC) control in Section 4.1.
2. A Yard Throat with track selection in Section 4.2.
3. A crossing with an interchange track in Section 4.3.
4. Automatic Block Signals in Section 4.4.

4.1 Siding with local control panel

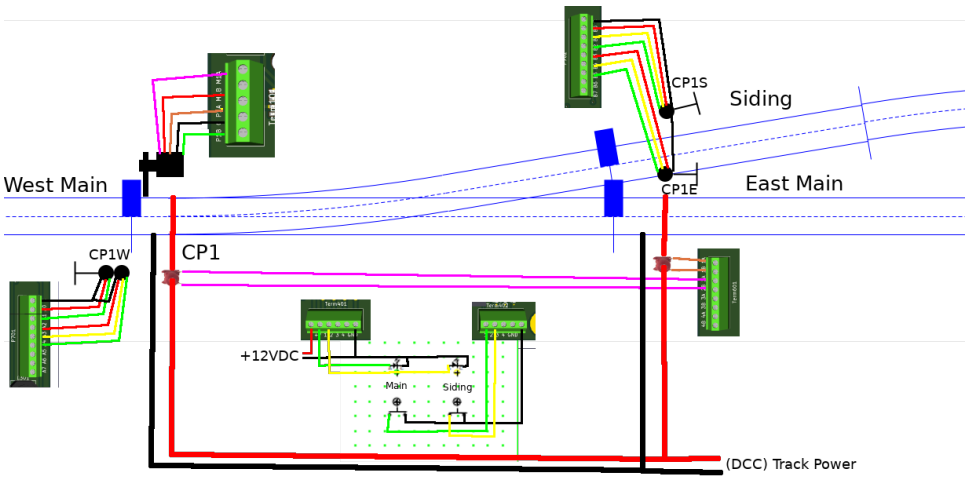

Figure 15: One end of a siding (CP1) with local and remote control, with signals.

This is a basic single turnout with a facia panel with two LEDs indicating the turnout points' position and two push buttons to select the desired turnout position. There also is a two headed (3 over 2) interlocking signal at the turnout's points, and a pair of single head (3 color) signals at the frog ends. The turnout's wiring is shown in Figure 15. We will use one of the turnout drivers (the Stall Motor daughter board is shown, but any of the daughter boards could be used), two of the occupancy detectors, two of the LED drivers, two of button inputs, and 11 of signal lamp driver outputs.

We will start by configuring the user name and description of the node in User Info tab. When we have done this will will be able to find the node by looking for the name.

Next, we will move onto the Board Configuration tab and then the Occupancy Detectors, specificly OC1. Right now, we will just fill in the name (East Main) and create a Block layout control. Similarly for OC2 (CP1 OS).

Similarly we configure the Turnout and Points, filling their names and creating layout control elements for them, along with the veto control element. The layout control elements (or various JMRI table elements).

Next we set up the logic for the Veto. We want to enable the turnout veto in two cases: dispatcher locking or OS occupancy, so we need one logic element for this. We name the Logic "Veto On", copy the OS Occupancy detector events to Variable 2 (Variable 1 will be used by the dispatcher to lock the turnout), set the Logic to V1 OR V2. The actions for both true and false will be send and exit. Action event 1 will be Send Immediately on True and the Turnout Veto On event will be copied to the action event. Action event 2 will be Send Immediately on False and the the Turnout Veto Off event will be copied to the action event.

Next we set up the LED drivers and the button inputs. We copy the turnout's Normal and Reverse motor events to the two button ON events and label the buttons Normal and Reverse. Similarly we copy the Point sensor events to the LEDs' events. For the Normal labeled LED, copy the Point sensor Normal event to the on event, and the Point sensor Reverse event to the off event. Then for the Reverse labeled LED, copy the Point sensor Reverse event to the on event, and the Point sensor Normal event to the off event.

For the signals, we will use additional Logics and three Masts. For each signal will will construct an sequence of logic elements, one for each aspect, starting with the most restrictive and ending with the least restrictive. Each logic element will test one or two conditions and be true for one aspect.

For the signal at the point entry of the turnout we are using a two headed signal, a 3 over 2 (3 lamps (GYR) on the top head, 2 lamps (GR or YR) on the bottom head). This signal will display 4 aspects: Stop (red over red), Approach Limited (red over green or yellow), Approach (yellow over red), and Clear (green over red). These aspects mean:

- Stop (red over red) Absolute stop. Typically because the turnout is currently occupied or the dispatcher forbids traversal.
- Approach Limited (red over green or yellow) Slow speed, taking the siding. This is when the turnout is thrown.

- Approach (yellow over red) Reduced speed continuing on the main. This is when there is a Stop indication at the next signal.
- Clear (green over red) Normal speed continuing on the main. This is when no other cases prevail.

Select the Mast 1 tab under the Rule to aspect tab under Board Configuration. Select Normal under Mast Processing and enter a Mast ID of CP1W for the Mast ID. Then copy the Track Circuit Link Address to Circuit 1 under TRACK CIRCUITS and set the Remote Mast Description to CP1W.

Back at Rule to aspect / Mast 1 scroll down to the Rules. Under Rule 1's Appearance, set Lamp 1 to A0, and Lamp 2 to A2. These are the two red lamps. Now select tab Rule 2. Set Name to 27-Approach-Limited and Track Speed to Limited. Then for its Appearance, set Lamp 1 to A1 and Lamp 2 to A2. Moving on to Rule 3: Set Name to 21-Approach and Track Speed to Approach. Then Lamp 1 to A0, and Lamp 2 A3. Finally for Rule 4: Set Name to 29-Clear and Track Speed to Clear/Procede and Lamp 1 to A0, and Lamp 2 to A4. This completes setting up the mast.

The logic for this mast is:

```
if (TurnoutOS is occupied) then
    send Stop
else if (Turnout is Reversed) then
    send Approach Limited
else if (track circuit(next block) is Stop) then
    send Approach
else
    send Clear
```

We will need 4 successive Logic elements for this mast, but first we need to create a virtual mast for the block identified as East Main. We will use Mast 8 and Circuit 8 for this mast. Select Mast 8, set its Mast Processing to Normal, its Mast ID to CP1ME, and copy its Link Address to Circuit 8. Now right click on the East Main OC in the Layout DB window, and copy its Occupied event to the event to set aspect for Rule 1 (Rule one is already set to Stop speed). Then copy East Main's Clear event to the event to set aspect for Rule 2, then change Rule 2's name to 29-Clear and Track Speed to Clear/Procede. We won't be setting anything for this mast's Lamps, since it is not a "visible" signal. We are just using it to general track circuit signals.

Now we can start to set up the Logic elements. We will start with Logic 4. Set its description to "CP1W Stop", its Group Function to Group, copy CP1 OS OC detector's events to Variable #1's events: true from Occupied, false from Clear. Set the Logic function to V1 only, set Action 1 to Immediate If True and copy the event from Mast 1, Rule 1's "(C) Event to Set Aspect" event to Action 1's event.

Next we set up Logic 5. Set its description to "CP1W Approach Limited", its Group Function to Group, copy the Points 1 sense detector's events to Variable #1's events: true from Reversed, false from Normal – we want *true* if the turnout is set to the siding. Set the Logic function to V1 only, set Action 1 to Immediate If True and copy the event from Mast 1, Rule 2's "(C) Event to Set Aspect" event to Action 1's event.

Next we set up Logic 6. Set its description to "CP1W Approach", its Group Function to Group, its variable source to Track Circuit 8. Use the default speed of Stop. Set the Logic function to V1 only, set Action 1 to Immediate If True and copy the event from Mast 1, Rule 3's "(C) Event to Set Aspect" event to Action 1's event.

Finally we set up Logic 7. Set its description to "CP1W Clear", its Group Function to Last (Single), set its Logic function to null => true, set Action 1 to Immediate If True and copy the event from Mast 1, Rule 4's "(C) Event to Set Aspect" event to Action 1's event.

Next we need to set up the two signals at the frog end of the turnout. They both the same aspects:

- Stop (red) Absolute stop. Typically because the turnout is currently occupied, the points are aligned against travel, or the dispatcher forbids traversal.
- Approach (yellow) Reduced speed continuing on the main. This is when there is a Stop indication at the next signal.
- Clear (green) Normal speed continuing on the main. This is when no other cases prevail.

We will set up these two signals on Mast 2 and Mast 3 and also set up a virtual mast for West Main using Mast 7.

First CP1E on Mast 2:

Set Mast Processing to Normal, Mast ID to CP1E, copy the track circuit address to Circuit 2, along with the Mast ID to the Remote Mast Description. Rule 1's Name will be 0-Stop, its speed Stop, and its Appearance will be Lamp 1: B3. Rule 2's Name will be 21-Approach, its speed Approach, and its Appearance will

be Lamp 1: B4. Rule 3's Name will be 29-Clear, its speed Clear/Procede, and its Appearance will be Lamp 1: B5.

Then CP1S on Mast 3:

Set Mast Processing to Normal, Mast ID to CP1S, copy the track circuit address to Circuit 3, along with the Mast ID to the Remote Mast Description. Rule 1's Name will be 0-Stop, its speed Stop, and its Appearance will be Lamp 1: B0. Rule 2's Name will be 21-Approach, its speed Approach, and its Appearance will be Lamp 1: B1. Rule 3's Name will be 29-Clear, its speed Clear/Procede, and its Appearance will be Lamp 1: B2.

Finally, we will create a virtual mast for West Main using Mast 7. This will be set up much like we did for block identified as East Main, using Mast 7 and Circuit 7. We can either use one of the unused occupancy detectors for this block, but more likely, there will be another occupancy detector for that block.

Next we will set up the logic for signals CP1E (Mast 2) and CP1S (Mast 3). These will each use three logic elements and will be very similar to the CP1W signal, but only using three logics since there are only three aspects for these signals. The only special issue is the logic of the point position: CP1S displays Stop when the points are "normal" and CP1E displays Stop when the points are "reversed", and the Stop logic is ORed with the OS being occupied. Otherwise Approach with the virtual signal at Mast 7 is Stop, and Clear otherwise.

A graph of the Example (CP1) siding Event Id "flow" is shown in Figure 16, with its color coding shown in Table 1. This may look a but intimidating, but each "edge" (line with an arrow) represents a producer / consumer. When multiple edges land at a given spot, the edges all represent the *same* event id. It is easiest to copy the destination (consumer) Event ID to each of the various producers.

Color	Object	What it is
Orange	Node	Occupancy Detectors
Green	Node	Siding normal button and LED
Red	Node	Siding reversed button and LED
Cyan	Node	Logic Elements
Blue	Node	Turnout Outputs
LightSeaGreen	Node	Point Sense Inputs
RoyalBlue	Node	Masts
Green	Edge	Button and LED normal Events
Red	Edge	Button and LED reverse Events
Brown	Edge	Button Release Events
Green	Edge	Turnout Operating Events
Black	Edge	Logic Fall Throughs (Elses)
Orange	Edge	Odd Yard Tracks (1 and 3)
Violet	Edge	Even Yard Tracks (2 and 4)
Cyan	Edge	Mast Rule Selection
LightSeaGreen	Edge	Point Sense events

Table 1: Graph color key

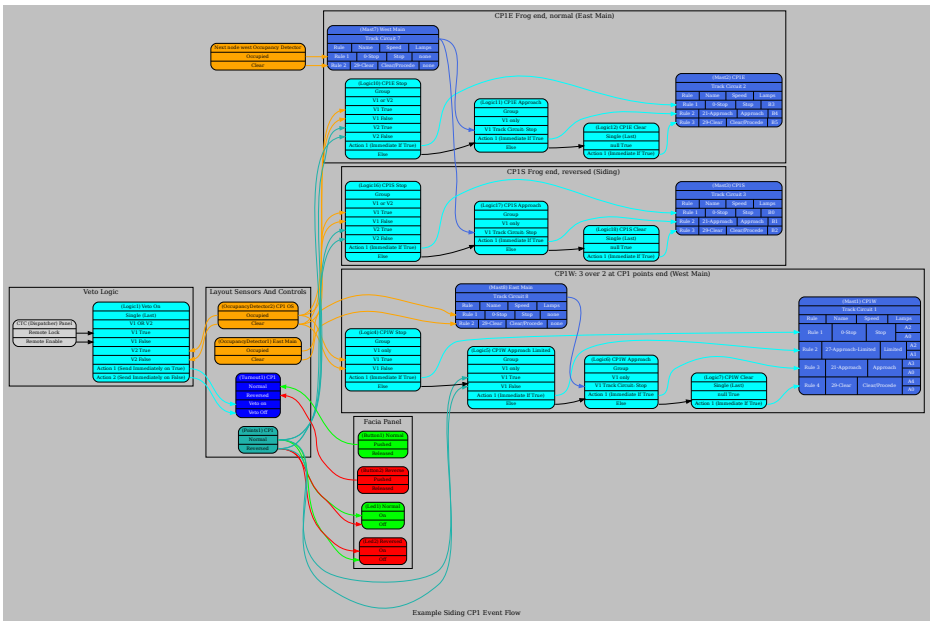

Figure 16: Example Siding CP1 Event Id graph.

4.2 Yard Throat with track selector

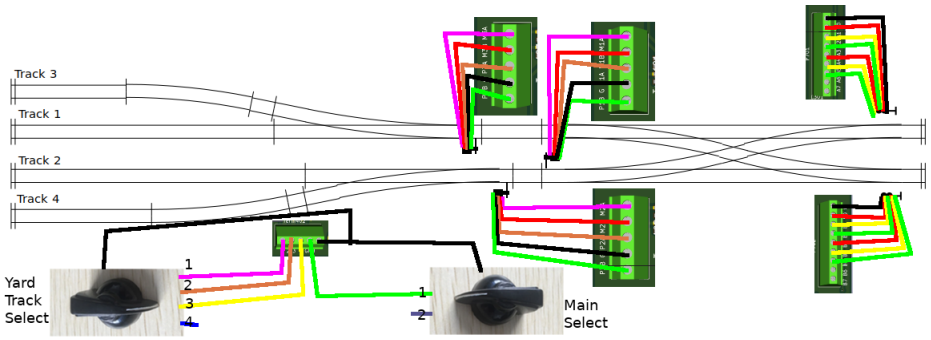

Figure 17: Yard throat with yard ladders and selectors.

This is the entrance to a yard from a double track main line with a scissors type crossover and a simple yard ladder. There are two selector switches, one to select the main track and one to select the yard track. There are a pair of signals at the yard entrance, one for each main line track. These are 3 over 3 signals. The basic wiring is shown in Figure 17. We will use three turnout drivers (the Stall Motor daughter board is shown, but any of the daughter boards could be used), 12 of the signal lamp driver outputs, and all four of the button inputs.

We will start by configuring the user name and description of the node in User Info tab. When we have done this will will be able to find the node by looking for the name.

Next, we will move onto the Board Configuration tab and give names to the Turnout and Points we will be using.

Then we will move onto the button inputs. The rotary switches total 6 positions, but we can get by with only 4 inputs, by leaving one position of each switch unconnected. This works because we get extra state information in the form of "none of the above" – when all three of buttons 1, 2, and 3 are "off", yard track 4 is selected and when button 4 is off, main track 2 is selected.

The logic for the yard track selection is:

```
if button1 is on then
  track 1 selected
else if button2 is on then
  track 2 selected
else if button3 is on then
```

```
  track 3 selected
else
  track 4 selected
```

and the logic for the main track selection is:

```
if button4 is on then
  main track 1 selected
else
  main track 2 selected
```

The logic for the scissors crossover is:

```
if ( track 1 selected OR track 3 selected) AND
   main track 1 selected then
  NORMAL
else if ( track 1 selected OR track 3 selected) AND
        main track 2 selected then
  REVERSE
else if ( track 2 selected OR track 4 selected) AND
        main track 1 selected then
  REVERSE
else if ( track 2 selected OR track 4 selected) AND
        main track 2 selected then
  NORMAL
```

This may *look* formidable, but we can simplify things using a "trick" – a kind of "wired or", by using a common event id to represent a common routing for the odd and even yard tracks. This simplifies things to

```
if common odd tracks AND main track 1 selected then
  NORMAL
else if common even tracks AND main track 2 selected then
  NORMAL
else if common even tracks AND main track 1 selected then
  REVERSE
else if common odd tracks AND main track 2 selected then
  REVERSE
```

We will start by allocating four successive logic elements, setting first three to have a Group Function of Group and the last to have a Group Function of Last (Single). We will set the descriptions of these logic elements to “Track 1”, “Track 2”, “Track 3”, and “Track 4”. The first three will have a Logic function of V1 only, and the last to null => true. We will copy the button event ids from Buttons 1 through 3 to the V1 events of the first three logic elements: the “Button Pushed” event ids to the “set variable true” event ids and the “Button Released‘” event ids to the “set variable false” event ids.

We will set action 1 of each of these logic elements to an action of Immediate if True, and copy the event ids as follows:

- Track 1: to Turnout 3’s Normal.
- Track 2: to Turnout 2’s Normal.
- Track 3: to Turnout 3’s Reverse.
- Track 4: to Turnout 2’s Reverse.

Next we will allocate four more Logic elements. Name these “Main Track 1 Odd”, “Main Track 2 Even”, “Main Track 1 Even”, and “Main Track 2 Odd”. Set the Group Function of all of them to Last (Single). Set the Logic Function of all four to V1 and V2. Copy Button 4’s “Button Pushed”, and “Button Released” event to the V1 “Set variable true” of both “Main Track 1” logics’ “Set variable true” (Button Pushed) and “Set variable false” (Button Released). Copy Button 4’s “Button Pushed”, and “Button Released” event to the V1 “Set variable true” of both “Main Track 2” logics’ “Set variable true” (Button Released) and “Set variable false” (Button Pushed). For the variable V2 events we will copy their events to to the Action 2 events of the Track 1 through Track 4 logics. The odd logics V2 true to get copied to the odd tracks and their V2 false get copied to the even tracks. And it is swapped for the even logics. We will end up with two event ids:

- Odd tracks true / Even tracks false Event: Action 2 events of Track 1 and Track 3 and the V2 true of “Main Track 1 Odd” and “Main Track 2 Odd” and the V2 false of “Main Track 1 Even” and “Main Track 2 Even”.
- Even tracks true / Odd tracks false Event: Action 2 events of Track 2 and Track 4 and the V2 true of “Main Track 1 Even” and “Main Track 2 Even” and the V2 false of “Main Track 1 Odd” and “Main Track 2 Odd”.

Finally, we set up the signals. Each signal is a 3 over 3 signal, with these aspects:

- Stop (Red over Red). Absolute stop, this main is not selected for travel.
- Slow-Limited (Red over Yellow). Slow, crossover and divergent route beyond: either main track 1 to yard track 4 or main track 2 to yard track 3.
- Slow-Medium (Red over Green). Slow, crossover and non-divergent route beyond: either main track 1 to yard track 2 or main track 2 to yard track 1.
- Slow (Yellow over Red). Slow, straight through crossover and divergent route beyond: either main track 1 to yard track 3 or main track 2 to yard track 4.
- Slow-clear (Green over Red). Slow, straight through crossover and non-divergent route beyond: either main track 1 to yard track 1 or main track 2 to yard track 2.

We implement this with a mast and 5 logics for each signal.

Color	Object	What it is
Yellow	Node	Button Inputs
Cyan	Node	Logic Elements
Green	Node	Turnout Outputs
LightSeaGreen	Node	Point Sense Inputs
RoyalBlue	Node	Masts
Yellow	Edge	Button Push Events
Brown	Edge	Button Release Events
Green	Edge	Turnout Operating Events
Black	Edge	Logic Fall Throughs (Elses)
Orange	Edge	Odd Yard Tracks (1 and 3)
Violet	Edge	Even Yard Tracks (2 and 4)
RoyalBlue	Edge	Mast Rule Selection
Cyan	Edge	Logic cascade
LightSeaGreen	Edge	Point Sense events

Table 2: Graph color key

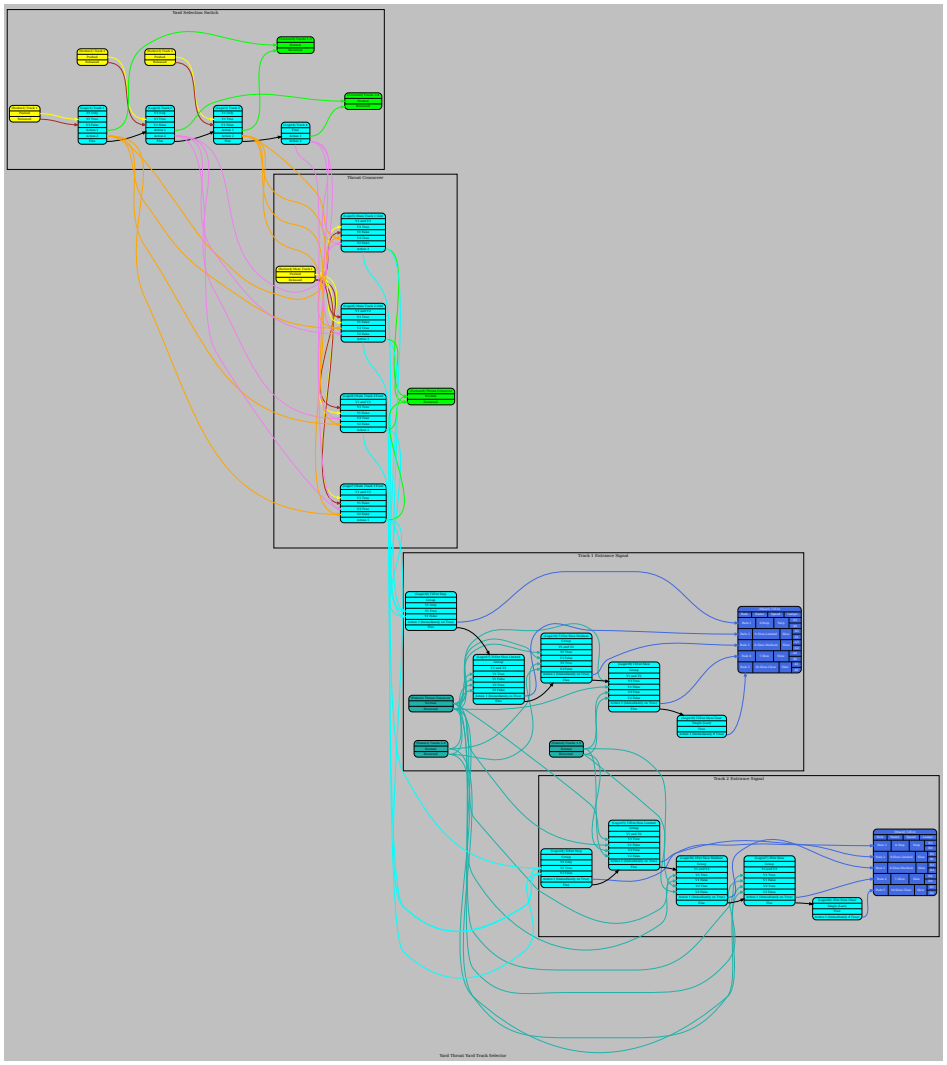

Figure 18: Yard Throat Selection Event Id graph.

A graph of the Yard Throat Selection Event Id “flow” is shown in Figure 18, with its color coding shown in Table 2. This may look a but intimidating, but each “edge” (line with an arrow) represents a producer / consumer. When multiple edges land at a given spot, the edges all represent the *same* event id. It is easiest to copy the destination (consumer) Event ID to each of the various producers.

4.3 Crossing with interchange

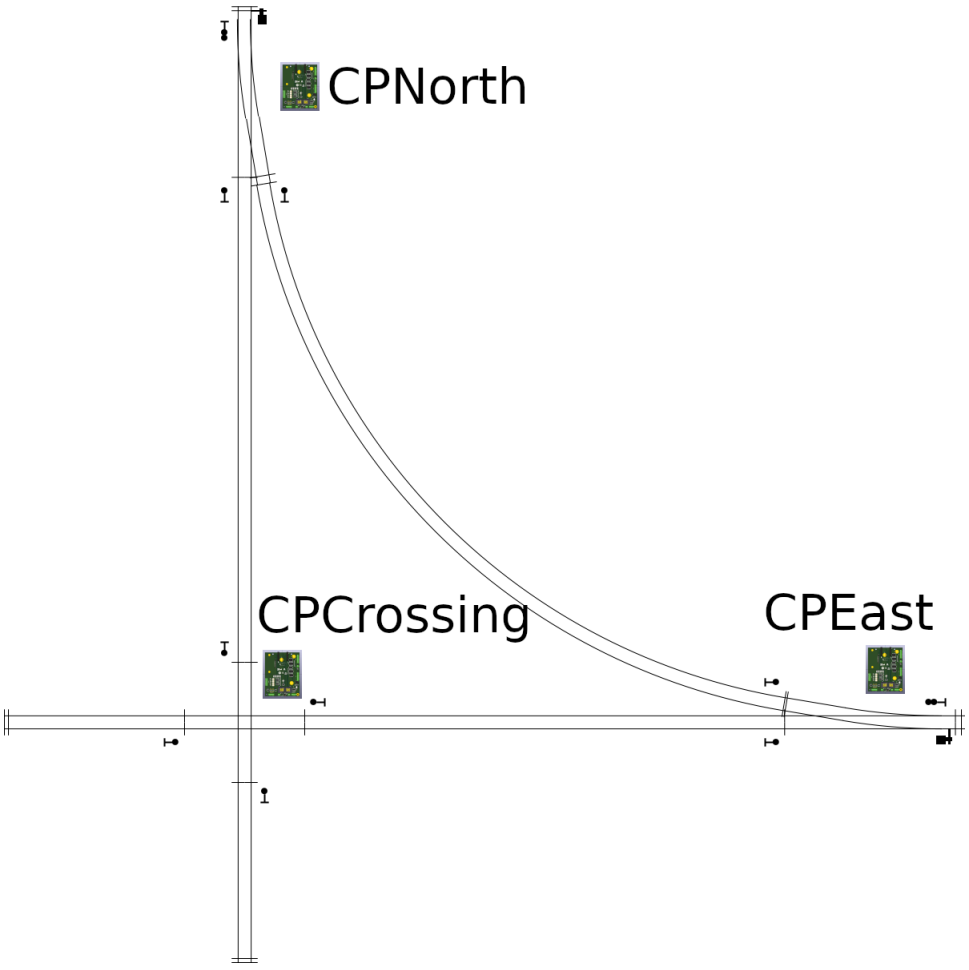

Figure 19: Crossing with Interchange.

This is a level crossing with an interchange track. We will control and manage

this crossing and interchange with three ESP32 T7S3 MultiFunction Universal Turnout nodes, as shown in Figure 19. The three nodes will implement these functions:

CPNorth This node will manage the north turnout and the signals around it. It will also provide occupancy detection for the turnout OS section and the north-south track segment. It will be wired as following:

OC 1 The turnout OS section.

OC 2 The north-south track segment.

Turnout 1 The north turnout motor.

Points 1 The north turnout points.

A0(UR), A1(UY), A2(UG), A3(LR), A4(LY), A5(LG) The signal at the north (points) end of the turnout.

B0(R), B1(Y), B2(G) The interchange segment (divergent frog) end of the turnout.

B3(R), B4(Y), B5(G) The north-south track segment (non-divergent frog) end of the turnout.

CPEast This node will manage the east turnout and the signals around it. It will also provide occupancy detection for the turnout OS section, the east-west track segment, and the interchange track segment. It will be wired as following:

OC 1 The turnout OS section.

OC 2 The east-west track segment.

OC 3 The interchange track segment

Turnout 1 The west turnout motor.

Points 1 The west turnout points.

A0(UR), A1(UY), A2(UG), A3(LR), A4(LY), A5(LG) The signal at the west (points) end of the turnout.

B0(R), B1(Y), B2(G) The interchange segment (divergent frog) end of the turnout.

B3(R), B4(Y), B5(G) The east-west track segment (non-divergent frog) end of the turnout.

CPCrossing This node will manage the crossing diamond, including the signals around it along with occupancy detection of the diamond itself. It will be wired as following:

OC 1 The east-west part of the crossing.

OC 2 The north-south part of the crossing.

A0(R), A1(Y), A2(G) The signal at the west end of the crossing.

A3(R), A4(Y), A5(G) The signal at the east end of the crossing.

B0(R), B1(Y), B2(G) The signal at the north end of the crossing.

B3(R), B4(Y), B5(G) The signal at the south end of the crossing.

The CPNorth and CPEast will actually be configured following the much the same methods as an end of a siding, as describe in Section 4.1[4] Since we are using 3 over 3 signals at the points ends of the turnouts, it is possible to add additional aspects.

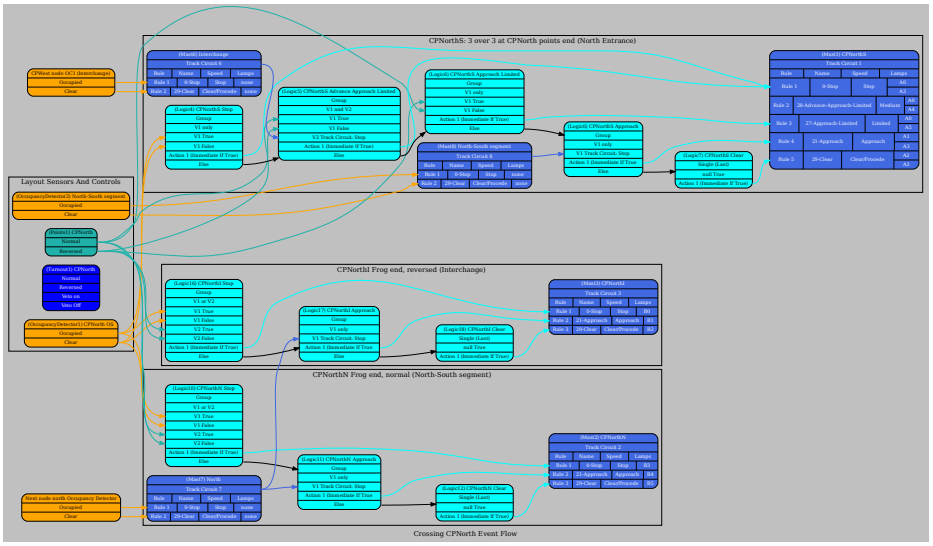

Figure 20: CPNorth Event Id graph.

[4]The local control buttons and LEDs are not shown here - they could be added or left out as desired.

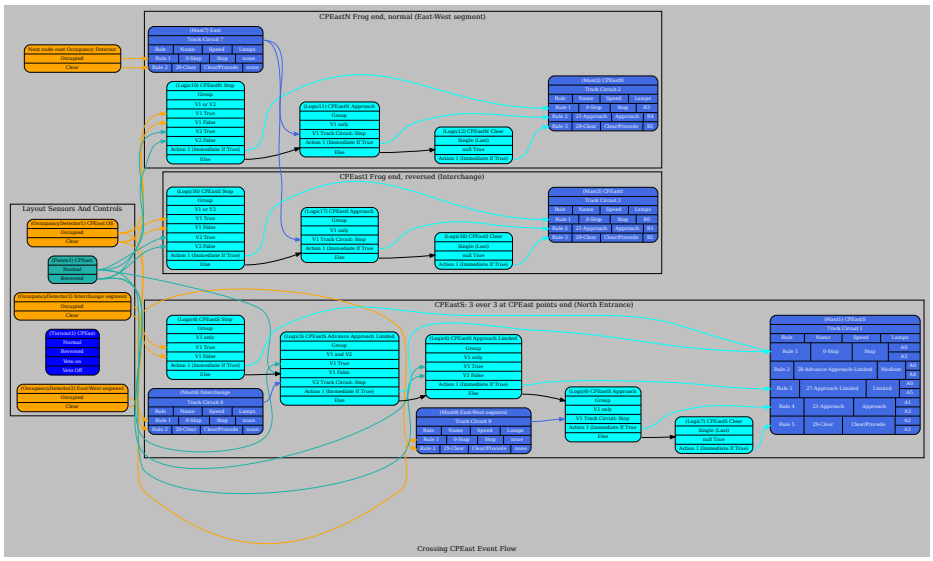

Figure 21: CPEast Event Id graph.

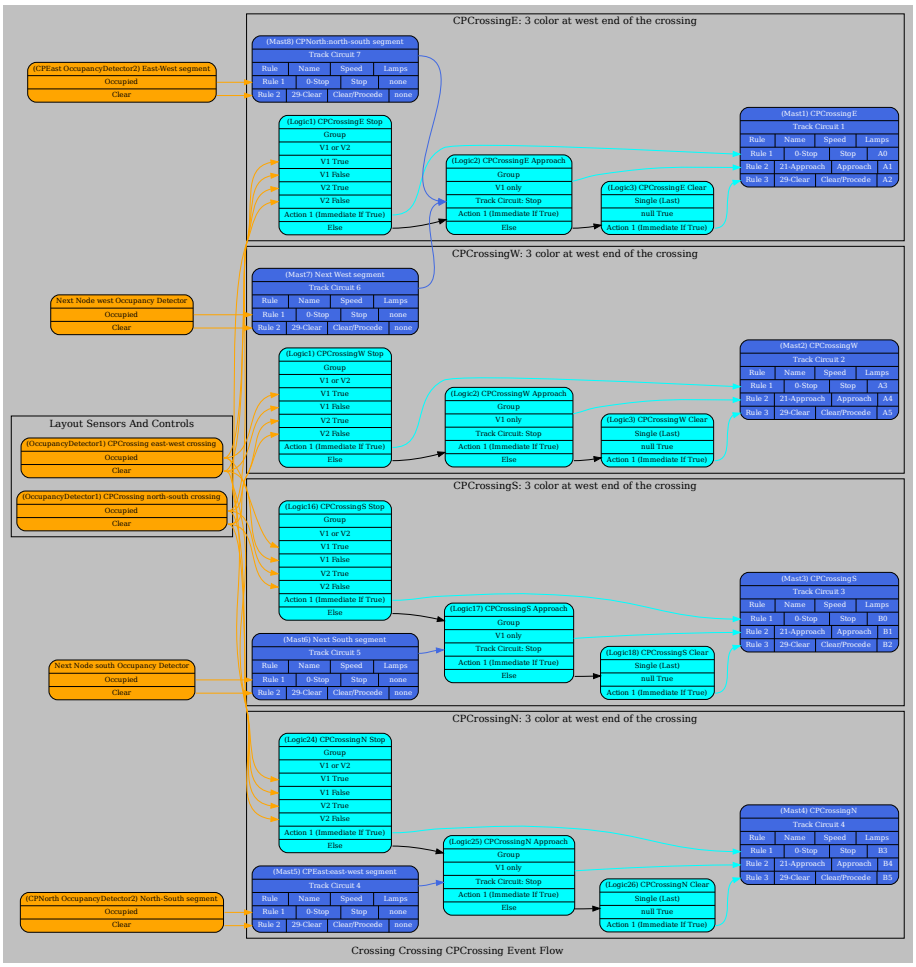

Figure 22: CPCrossing Event Id graph.

Color	Object	What it is
Orange	Node	Occupancy Detectors
Green	Node	Siding normal button and LED
Red	Node	Siding reversed button and LED
Cyan	Node	Logic Elements
Blue	Node	Turnout Outputs
LightSeaGreen	Node	Point Sense Inputs
RoyalBlue	Node	Masts
Green	Edge	Button and LED normal Events
Red	Edge	Button and LED reverse Events
Brown	Edge	Button Release Events
Green	Edge	Turnout Operating Events
Black	Edge	Logic Fall Throughs (Elses)
Orange	Edge	Odd Yard Tracks (1 and 3)
Violet	Edge	Even Yard Tracks (2 and 4)
Cyan	Edge	Mast Rule Selection
LightSeaGreen	Edge	Point Sense events

Table 3: Graph color key

4.4 Automatic Block Signals

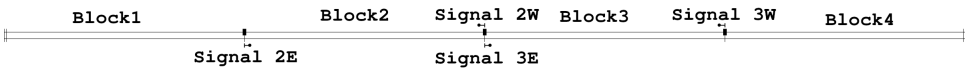

Figure 23: Automatic Block Signaled track.

Figure 23 is a simple bi-directional segment of ABS signaled track, with four blocks and four signals, wired to a node as follows:

OC 1 Block 1

OC 2 Block 2

OC 3 Block 3

OC 4 Block 4

A0(R), A1(Y), A2(G) Signal 2E

A3(R), A4(Y), A5(G) Signal 3E

B0(R), B1(Y), B2(G) Signal 2W

B3(R), B4(Y), B5(G) Signal 3W

Each signal will use three Logic elements:

1. Will be triggered true if the block is occupied and will select the Stop rule.
2. Will be triggered true if the next signal mast is stop (via a track circuit) and will select the Approach rule.
3. Will always be true and will select the Clear rule.

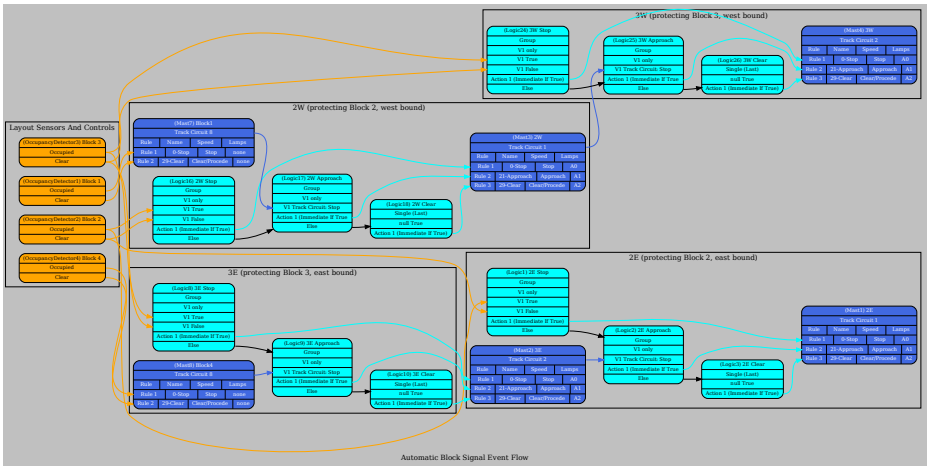

Figure 24: ABSTrack Event Id graph.

Color	Object	What it is
Orange	Node	Occupancy Detectors
Cyan	Node	Logic Elements
RoyalBlue	Node	Masts
Black	Edge	Logic Fall Throughs (Elses)
Cyan	Edge	Mast Rule Selection

Table 4: Graph color key