Signalling Application Quick-start Guide

Version 6 – April 2025

This guide is intended to provide an introduction to the Model Railway Signalling Application and the art of the possible in terms of the signalling configurations that can be achieved.

All of the example layouts used in this guide (and other example layout files) are packaged with the application. These can be accessed by selecting File => Examples from the Main Menubar and then selecting the appropriate file.

Other documentation (such as the Application Networking Guide) is also packaged with the application (Select Help => Docs from the Main Menu Bar):
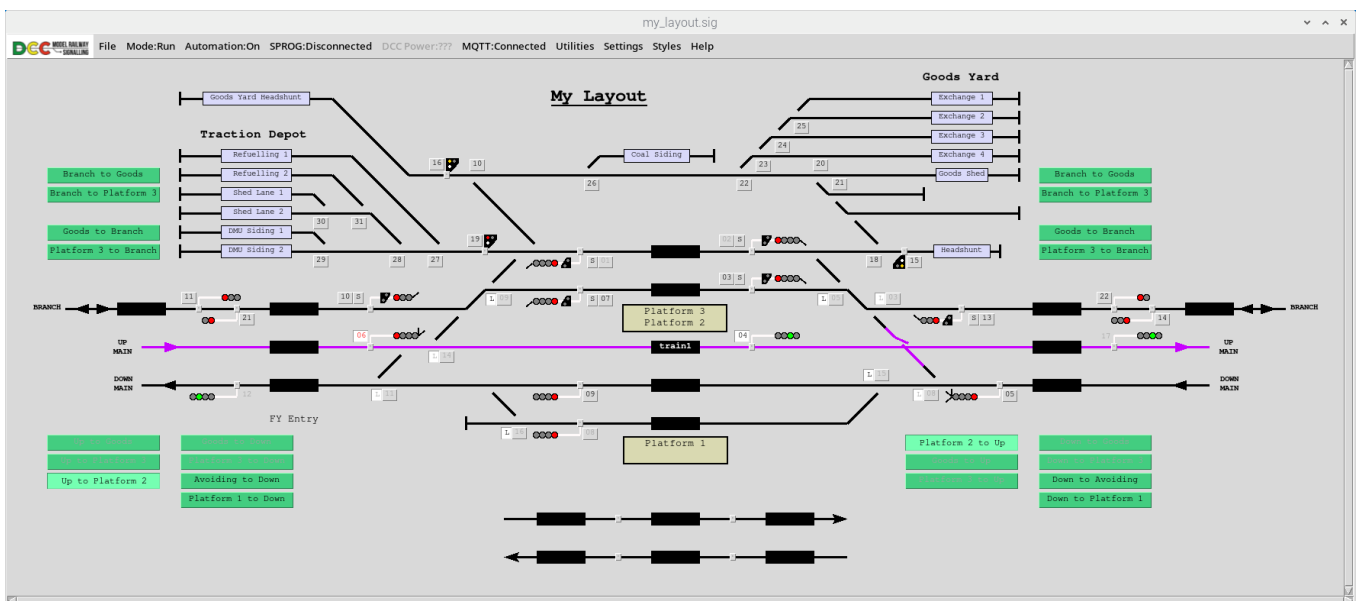
Table of Contents

Introduction

The application enables users to easily create, configure and control prototypical interlocked signalling schemes for model railway layouts, without the need for complex layout wiring. All layout configuration and control is achieved via the application's graphical user interface, avoiding the need for specialist computer skills (if you use PC applications, you should be able to use this).

The application has primarily been developed to run on a Raspberry Pi computer, hosting a Pi-SPROG DCC programmer controller:

- The Raspberry Pi is a low-cost single-board computer which provides a "Windows-like" user experience (and versions of all the usual applications you would expect, such as web-browser, email, office-type applications etc).

- The Pi SPROG DCC programmer controller connects directly to the Raspberry and provides a DCC 'accessory bus' output to control the points and signals out on the layout.

The use of a separate 'accessory bus' makes the system suitable for use with layouts that use DCC or analogue for control of trains (**when used with DCC layouts, the accessory bus for control of the signals/points needs to be electrically separated from the main DCC track bus**).

Several manufactures now provide DCC signals (e.g. Train-Tech from Gaugemaster) and point motors (e.g. Cobalt from DCC Concepts), making this method of control ideal for 'new-build' layouts. There are also numerous DCC signal/point decoders available for those wishing to upgrade to DCC control without the expense of wholesale replacement of their existing units.

The application uses the flexibility of the Raspberry Pi General Purpose Input/Output (GPIO) interface to provide feedback on train location. Simple sensors providing a 'normally-open' output (momentarily closed when triggered) can be connected directly to the appropriate GPIO pins to generate 'signal passed' events as the passing train triggers the sensor (e.g. the slim vertical magnetic sensors from DCC Concepts). These events can then be configured within the application to provide a 'mimic' diagram of train location and provide a level of signal automation.

Note that other sensor types (providing a switched voltage) should <u>never</u> be connected directly to the GPIO pins as this could damage the Raspberry-Pi. In these cases, external opto-isolators should be used - I've been using the PC817 2, 4 or 8 channel opto-isolator modules (available from several Ebay sellers) for my layout. Connection of these is relatively straight forward, but if you have any doubts then seek expert advice.

For added flexibility, the software enables multiple signalling applications to be networked together, making it ideal for control of larger layouts (where the layout gets broken down into multiple signalling areas) or splitting smaller layouts down to individual signal boxes (with simulated block instruments) for real 'true to prototype' operation. Note that in this case only one instance of the application needs to be running on a Raspberry Pi (the instance providing the interface to the DCC bus and track sensor inputs from the GPIO pins). As the application has been designed to be platform independent, other instances can be hosted on Windows or Linux as required.

The importance of research

This is probably the most important (and potentially time consuming) part of the process. If you're reading this document and planning to use the application to develop a signalling scheme for your model railway then I'd recommend building familiarity with British railway signalling practice. There are lots of great resources out there, but some of the best I've come across are:

- https://signalbox.org/ - Comprehensive information on signal types and the 'Block System' and a vast library of signal box diagrams for you to draw inspiration.

- https://en.wikipedia.org/wiki/UK_railway_signalling – Its Wikipedia (enough said).

- http://www.railway-technical.com/signalling/ - A section of the Railway Technical Website covering signalling. There are many great resources on these pages including:

  - http://www.railway-technical.com/signalling/infopaper-6-basic-railway.pdf – A paper (downloadable pdf format) on Basic Railway Signalling.

  - http://www.railway-technical.com/signalling/british-signalling--what.pdf – A paper (downloadable PDF) on "What the driver sees".

But Beware – Railway modeling is always about compromise and that is definitely going to be the case for whatever signalling scheme you design and implement for your layout. Although the application has been developed to add a touch of realism to the operation of your layout, it will never measure up to the million-pound-plus signalling systems of the 'real thing'.

And never forget - Rule 1 of Railway Modelling applies – it's your layout and its therefore entirely up to you how you signal your layout. Hopefully the features provided by the application will enable you to achieve whatever level of realism you want to achieve.
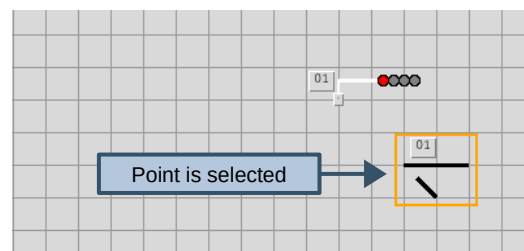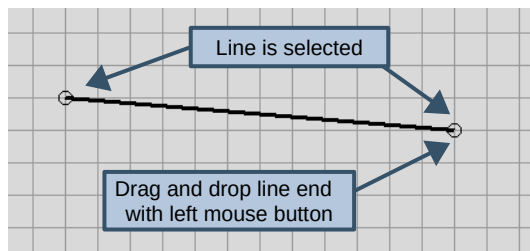
The schematic editor

The application opens in 'Edit' mode with a blank drawing canvas. The panel on the left contains the buttons to add schematic drawing objects to the canvas, whilst the Menubar across the top of the window contains various controls and options for configuring the application.
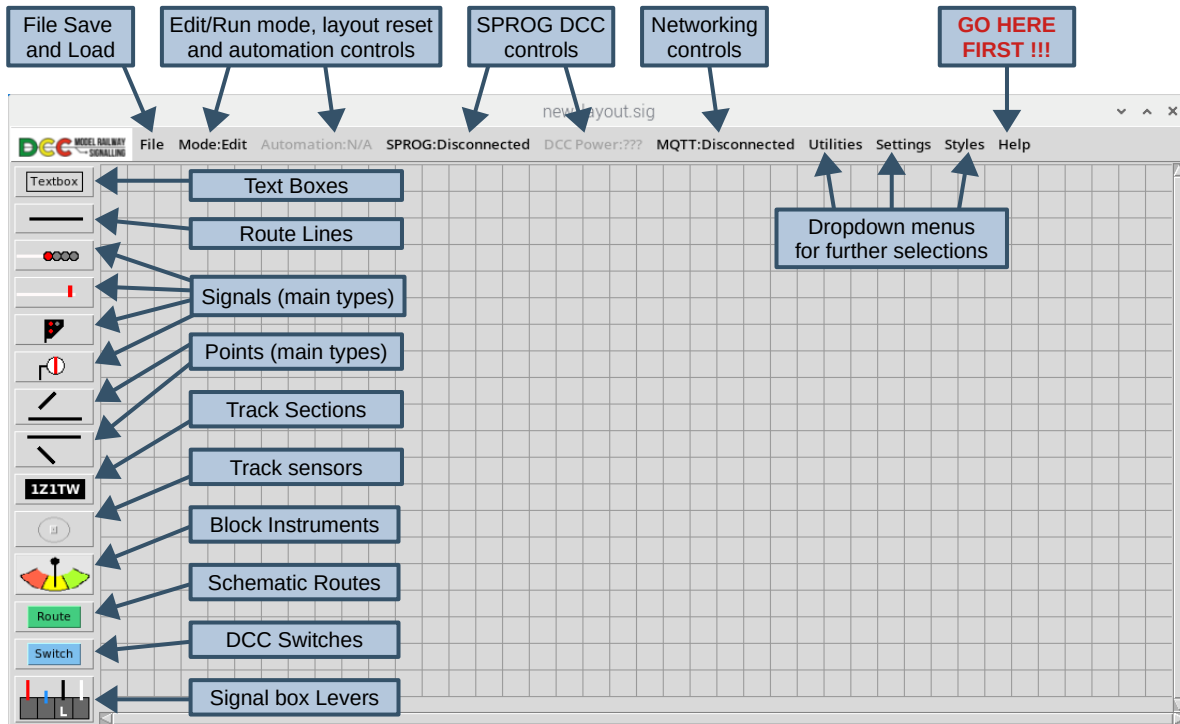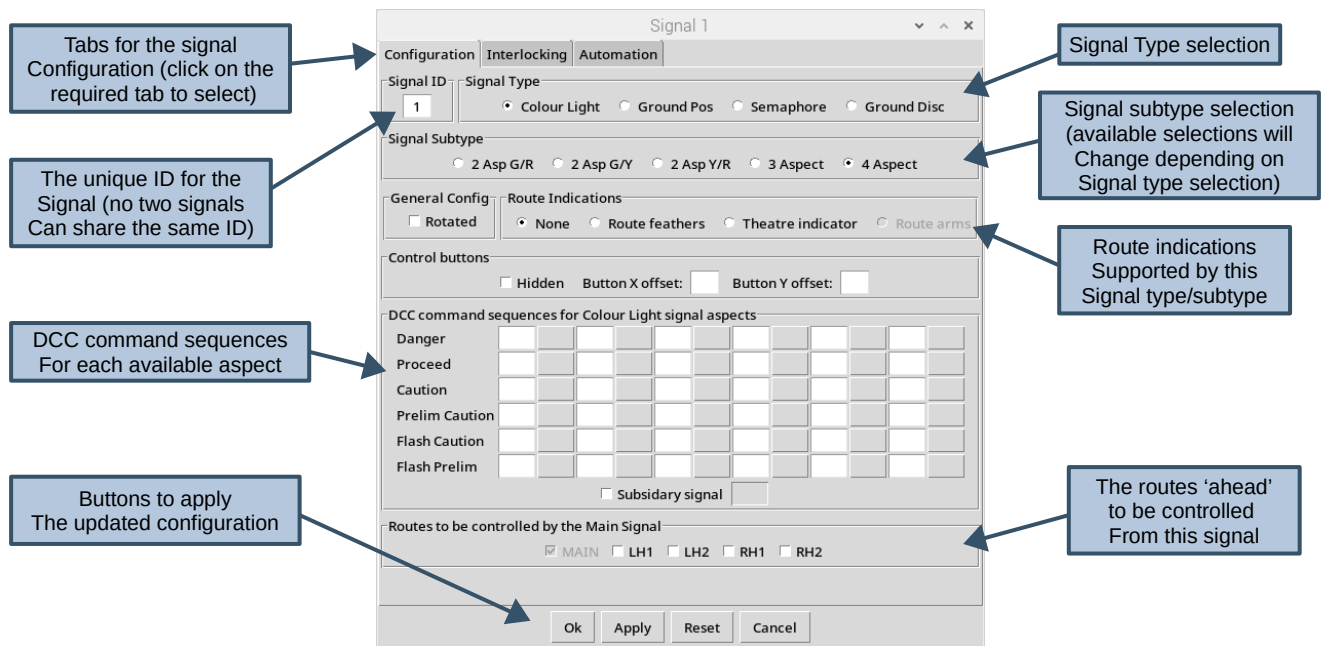

Open the Help Window from the Main Menubar (**Help => Help**) to familiarise yourself with the basic Schematic Editor functions and then practice creating, moving and deleting objects:

- Add objects to the canvas by **left-clicking** the buttons on the left hand side, drag them to the required position on the canvas and then 'place' them via another **left-click**.
- Select objects by **left-clicking** on them (small circles will be displayed at each end of lines to show they are selected - a border will be displayed around other drawing objects)
- Move selected objects by dragging and dropping (**left-click** => **move** => **left-release**).
- Move the ends of a line by selecting the line and then dragging and dropping the line ends
- Rotate selected objects (points and signals) by pressing the '**r**' key
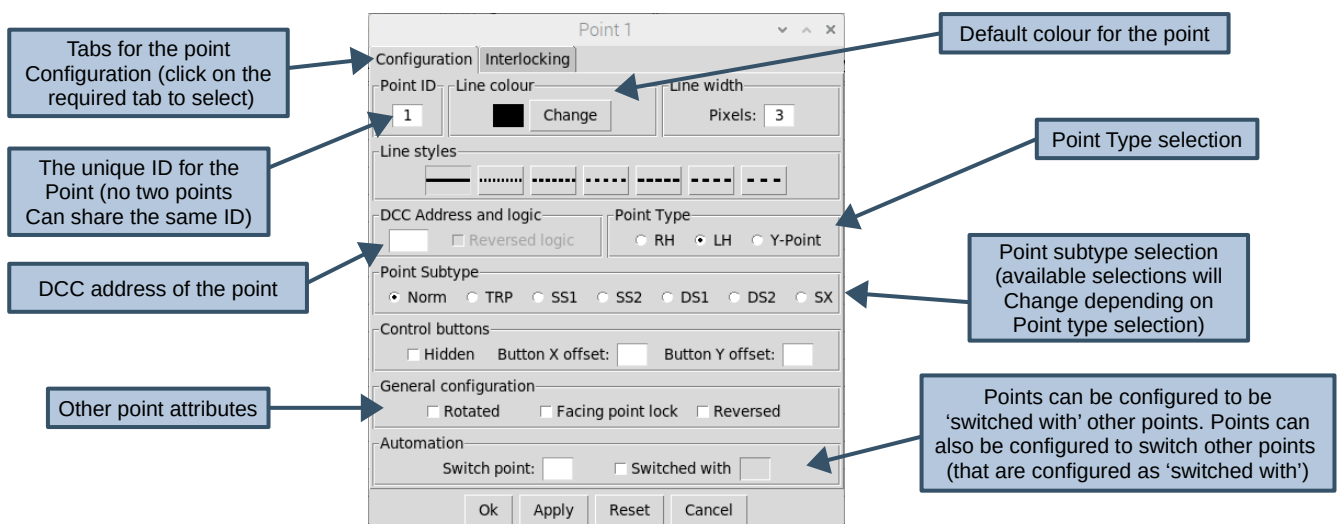- Delete selected objects by using the **backspace** key

Once you are happy with the basic schematic editor functions, try double clicking on some of the objects you have created to bring up their configuration windows. Although the number of settings and options may look daunting, don't worry – The application has built-in help in the form of 'tool-tips' which provide more information on what the settings are for and what type of information should be entered. To bring up the tool-tip for a setting, just hover the cursor over it.

For many schematic objects, the configuration windows allow you to change the type (and sometimes sub-type) of the object. This is particularly important for signals and it allows you to select the most appropriate signals for your layout (most UK signal types are supported):

Tabs for the signal Configuration (click on the required tab to select)

The unique ID for the Signal (no two signals Can share the same ID)

DCC command sequences For each available aspect

Buttons to apply The updated configuration

Signal Type selection

Signal subtype selection (available selections will Change depending on Signal type selection)

Route indications Supported by this Signal type/subtype

The routes 'ahead' to be controlled From this signal

Similarly for points, the type and sub-type options allows you to represent complex track formations (such as single/double slips, scissor crossovers, and trap points). Hover the cursor over the point sub-type selection for further information

Tabs for the point Configuration (click on the required tab to select)

The unique ID for the Point (no two points Can share the same ID)

DCC address of the point

Other point attributes

Default colour for the point

Point Type selection

Point subtype selection (available selections will Change depending on Point type selection)

Points can be configured to be 'switched with' other points. Points can also be configured to switch other points (that are configured as 'switched with')

In the following sections, we'll create an example schematic to demonstrate the main features of the application. Before we begin, we need to clear down the schematic (from the Main Menubar select **File => New) t**o give ourselves a blank canvas.

Quick start example 1 – colour light signals

Drawing your layout schematic

For the Quick Start Example 1, we are going to create a simple layout comprising a single track line serving a small double platformed terminus station (with no run round facilities):
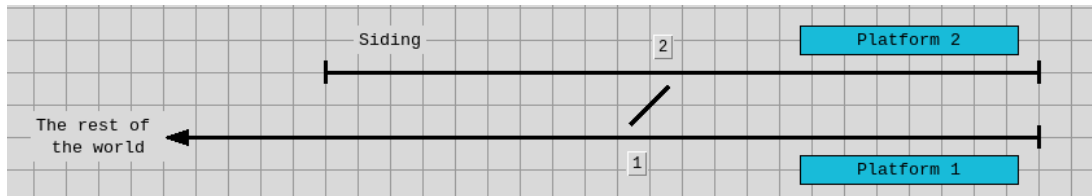

Point 1 (left hand point) can be added straight to the layout and moved into the appropriate position. To create point 2, either add another left hand point to the layout or, alternatively, select point 1 (**Left Click** on the point) and then copy/place (**Ctrl-c** then move/place). When items are created, they are assigned the next available 'one-up' identifier so in this case the added / pasted point will be created as Point 2. Whilst Point 2 remains selected, press the '**r**' key to rotate by 180 degrees, it can then be moved into position to form the required crossover.

Now add the track lines to the schematic. To add end-stops or arrows to the lines, **double-left-click** on the line to bring up the line configuration dialog. A different line end style can then be selected and applied to one or both ends. Once the required selection has been made, click the **OK** button to apply the changes and close the configuration dialog.

Text boxes can be added to annotate the schematic. Once created, double-left click on the 'Text' to bring up the configuration dialog and edit the contents. Note that text boxes will always be sized to fit their contents, but padding (extra lines or spaces) can be used if required. In this example, text boxes with padding before and after the text have been used for the two 'platforms'.

To complete the track layout, we'll configure the points to switch together and add a facing point lock. Firstly, **double-left-click** on point 2 to bring up the configuration dialog, set it to be fully automatic and then **OK**/**Apply** the changes. Now **double-left-click** on Point 1, configure it to switch Point 2 and select the facing point lock. We'll also configure the DCC address we're going to use to switch the points out on the layout (as both points are switched together, we only need a single address to switch both points). Again, click **OK**/**Apply** for the changes to take effect.
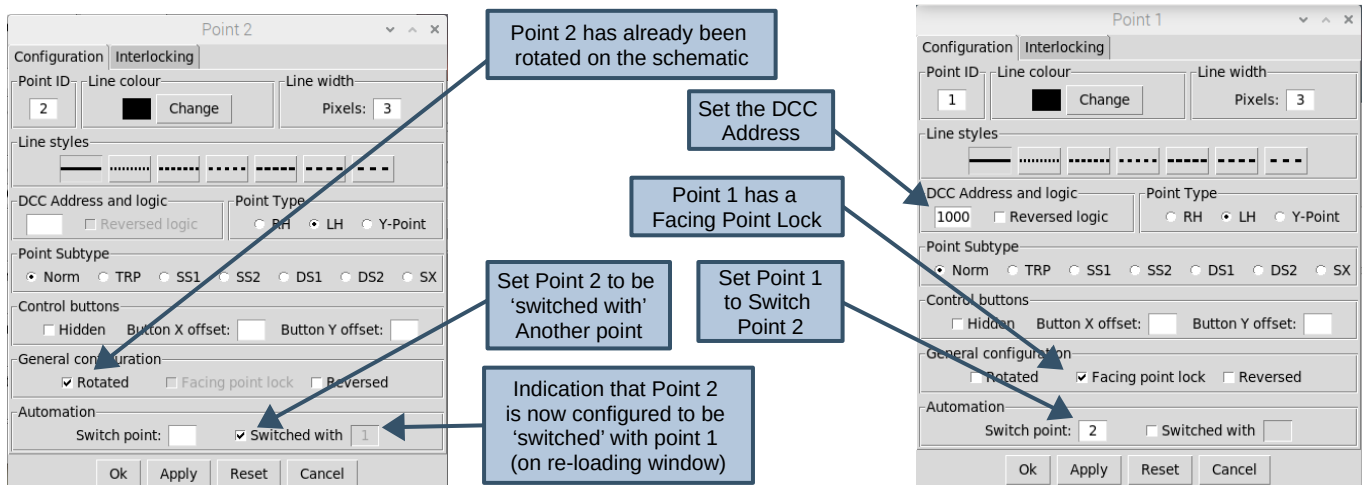
The schematic should now look as follows. Point 2 no longer has a control button as it will now be switched with Point 1, and Point 1 now has an additional 'L' button for the facing point lock (FPL). When this is active, then the main point control button is 'greyed out' and unresponsive. To switch the point, first click on the 'L' button to release the FPL. Once the FPL has been released, you can switch the point and then re-activate the FPL to lock the point in the switched position.
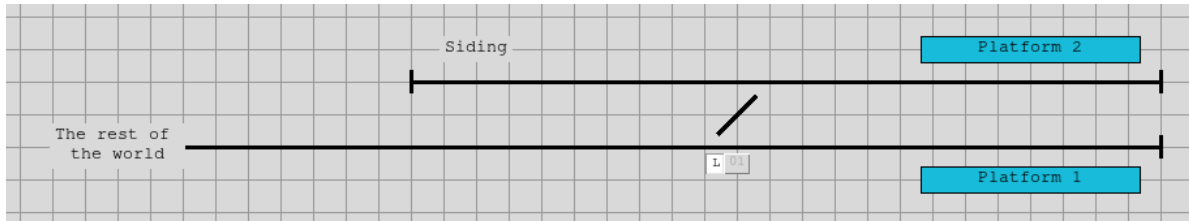
Planning your signalling scheme

The next step is to plan your signalling scheme in terms of the possible train movements on the layout. For this example, the main movements we want to signal are:

- Rest of the world => Platform 1
- Rest of the world => Platform 2
- Platform 1 => Rest of the world
- Platform 2 => Rest of the world

We also want to signal the following shunting movements:

- Platform 2 => Siding
- Siding => Platform 2

This planning, together with the knowledge gleaned from our research in the previous section, allows us to define the signals we want to add to the layout and where they should be positioned.

Adding signals to the schematic

Use the buttons on the left hand side of the window to add signals to the schematic and move them into position (rotating as required). The signals are initially created as four aspect but we'll change them to two aspect as we configure them. Signal 4 will be a ground position (shunting) signal.
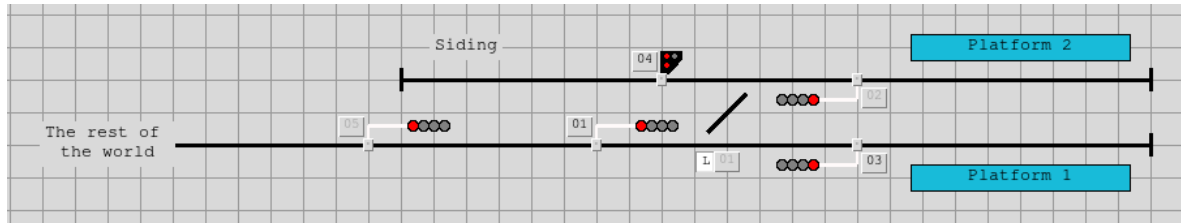
To signal the train movements we have defined, we need to edit Signal 1 to provide a route indication (to differentiate between the routes into Platform 1 and Platform2). At the same time, we'll change it to a two-aspect signal and configure the DCC addresses. As before, **double-left-click** to bring up the configuration dialog and use **Apply** or **OK** to save the changes

We also need to add a subsidiary aspect to Signal 2 to signal the shunting move from platform 2 into the siding (we'll also change this one to a two aspect signal and set up the DCC addresses).

Note that we don't need route indications for Signal 2 as the only route controlled by the main aspect is the departure out to the rest of the world (the MAIN route for this signal). The subsidiary aspect controls the shunting move back into the siding, which is effectively a right hand divergence from the main route (the RH1 route for this signal).

Change the signal to Two-aspect (green/red)

The 2 aspect signal uses a single DCC Address (toggle on/off) to set the aspect

The only route Controlled by the Main signal aspect Is out to the rest Of the world

Select to add a Co-located subsidiary Signal

The subsidiary aspect Uses a single DCC address (toggle On/Off)

The shunting Movement back into The siding is a Right Hand divergence From the main route

Signals 3 and 5 also need to be edited to change them to two-aspect signals and have their DCC addresses configured. Note that Signal 5 should be configured as a two-aspect Green/Yellow signal to act as the distant signal for our layout (as this is a terminus station, we'll configure it as a fixed distant when we specify the interlocking later on).

Signal 4 has been created as a shunting signal and only needs to control a single route (back into Platform 2). We therefore only need to specify the DCC addresses to complete the configuration.

Once all signal configurations have been applied, the schematic should look like this:

The subsidiary aspect is for The RH1 route into the siding

Distant Signal

Single feather indication is for for Platform 2 (no feather indication for the main route into platform 1)
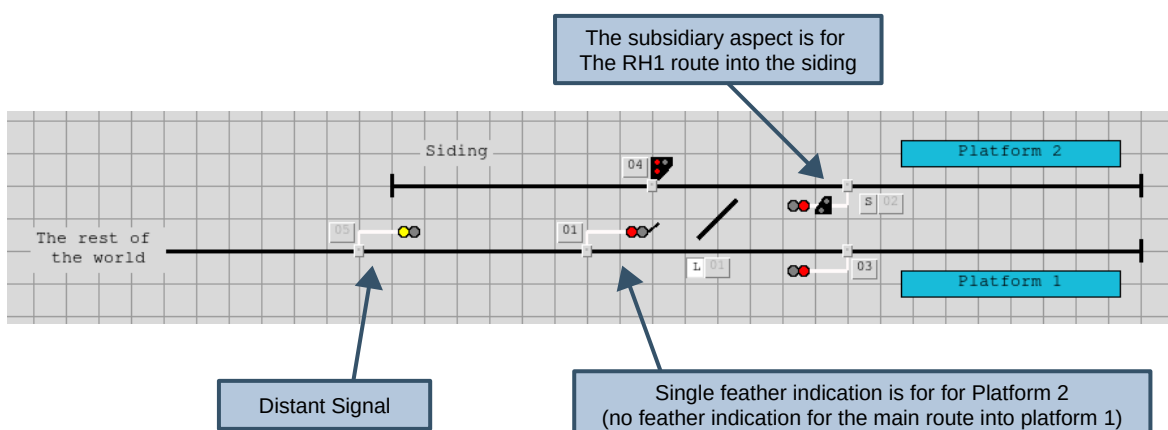
Configuring basic interlocking

The application allows signals to be interlocked with points, conflicting signals, track sections, and block instruments. Initially, we'll configure interlocking with points and conflicting signals. In this context, 'conflicting signals' are any other signals that control a route that would conflict with a route controlled by the signal we are configuring.

All interlocking is defined via the configuration dialog of the appropriate signals. **Double-left-click** on a signal to bring up the configuration dialog and select the **Interlocking** tab.

Signal 1

From the schematic on the previous page, Signal 1 needs to be interlocked with Point 1, but there are two possible routes (into platform 1 and into platform 2). To enable Signal 1 to be cleared for the MAIN route (into platform 1), Point 1 needs to be NORMAL. To enable Signal 1 to be cleared for the LH1 route (into platform2), Point 1 need to be SWITCHED. We do not have to interlock the signal with Point 2 as we have already configured this to be 'switched with' Point 1.

Each 'route' from Signal 1 also needs to be interlocked with any signals that could clear conflicting movements, in this case Signals 2 and 3. Both of these signals only have a single route controlled by the main signal aspect (out to the rest of the world), so we only need to interlock with the MAIN routes controlled by these two signals.

The MAIN route for Signal 1 (into Platform 1) therefore needs to be interlocked with the MAIN route (departing from Platform 1) for Signal 3. Similarly, The LH1 route for Signal 1 (into platform 2) needs to be interlocked with the MAIN route (departing from Platform 2) for Signal 2.

Signal 2

Signal 2 also needs to be interlocked with Point 1, but in this case, the only valid route for the main signal (as opposed to the subsidiary signal) is out from Platform 2 to the rest of the world. Point 1 therefore needs to be SWITCHED to allow Signal 2 to be cleared for the MAIN route.

When Signal 2 was initially configured (see earlier), the subsidiary aspect was configured to allow a shunting move is back into the siding (which is a right-hand diverging route). Point 1 therefore needs to be NORMAL to enable the subsidiary signal to be cleared for the RH1 route.

For the MAIN route (departure from Platform 2 to the rest of the world), Signal 2 needs to be interlocked with the LH1 route of Signal 1. For the RH1 route (back into the siding), Signal 2 needs to be interlocked with the MAIN route of Signal 4.

The MAIN route for Signal 2 to the rest Of the world requires Point 1 to be SWITCHED

The RH1 (shunting) route for Signal 2 back into the siding requires Point 1 to be SWITCHED

The MAIN route for Signal 2 to the rest of the world is interlocked with the LH1 route for Signal 1

The RH1 (shunting) route for Signal 2 back into the Siding is interlocked with the MAIN route for Signal 4

Signal 3

Signal 3 controls a single MAIN route (from Platform 1 out to the rest of the world) which needs to be interlocked with Point 1 (Point 1 needs to be NORMAL to allow Signal 3 to be cleared). The signal also needs to be interlocked with the MAIN route of Signal 1.

The MAIN route for Signal 3 to the rest Of the world requires Point 1 to be NORMAL

The MAIN route for Signal 3 to the rest Of the world Is Interlocked with the MAIN route for Signal 1

Signal 4

Signal 4 controls a single shunting route (from the siding to Platform 2) which needs to be interlocked with Point 1 (Point 1 needs to be NORMAL to allow Signal 4 to be cleared). The signal also needs interlocking with the RH1 route of Signal 2 (the shunting route into the siding).

The MAIN route for Signal 4 into Platform 2 requires Point 1 to be NORMAL

The MAIN route for Signal 4 into Platform 2 is interlocked with the RH1 route for Signal 2

Signal 5

As mentioned earlier, Signal 5 will be configured as a 'fixed distant'. In this case we can just pick any point and configure the signal such it is only unlocked when the point is both SWITCHED and NORMAL (which can never happen, meaning the signal is always locked at Caution).

The MAIN route for Signal 5 requires Point 1 to be both NORMAL and SWITCHED

Testing basic interlocking

Once configured, the basic interlocking can be tested. If a signal or point is 'locked' then the associated control buttons will be 'greyed out' and unresponsive.

Firstly, ensure the layout is in a default state by selecting **Mode => Reset** from the Main Menubar and selecting **OK** in the pop-up dialog to confirm. This will reset all signals to 'ON[1]' and reset all points to NORMAL with their Facing Point Locks (FPLs) ACTIVE.

Signal 5 will be locked as we configured this as a 'fixed distant'. The main aspect of Signal 2 will also be locked as the route controlled by this signal is out from Platform 2 to the rest of the world and the points are currently set back into the siding. The subsidiary aspect of signal 2 (controlling the shunting route back into the siding) will be unlocked.
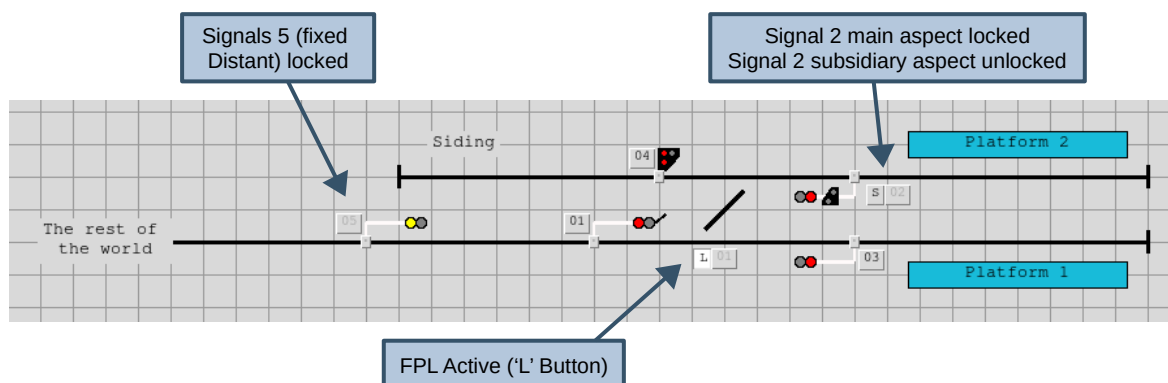
In this state, the interlocking with conflicting signals can be tested. If Signal 4 is switched 'OFF' then the subsidiary aspect of Signal 2 will be locked, and vice versa. Similarly, if signal 1 is switched 'OFF' then Signal 3 will be locked, and vice versa. Note that when a signal is set to 'OFF' the points are locked and cannot be changed until the signals are returned to 'ON'.
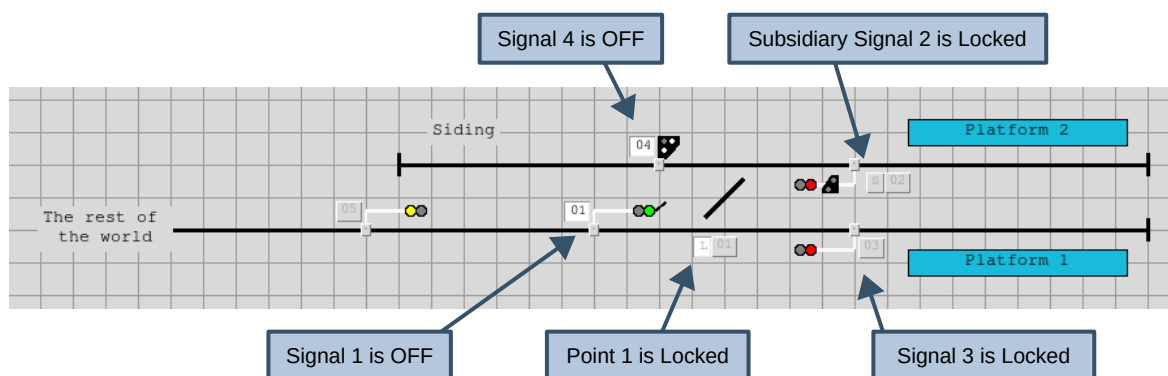
Once all signals have been returned to 'ON' then the points can be changed to test the other signal routes. Click on the '**L**' button to 'release' the Facing Point Lock (FPL) and enable the main point control button. The point can then be switched .

Note that as soon as the point FPL is 'released' then Signals 1, 2, 3 and 4 will be locked, to prevent a train movement being cleared 'through' the points, and will remain locked until the point FPL is 'reactivated' (after the points have been switched).

---

1   When a signal is 'ON', it is displaying its most restrictive aspect (DANGER for Home signals or CAUTION for distant signals). When a signal is 'OFF' it is displaying its least restrictive aspect (PROCEED for most signal types)

Once Point 1 has been switched (and the FPL re-activated) the layout should be as follows:

Signal 4 locked (no route)

Signal 2 main aspect unlocked
Signal 2 subsidiary aspect locked

Signal 1 unlocked

Point 1 switched
with FPL active

Signal 3 locked (no route)
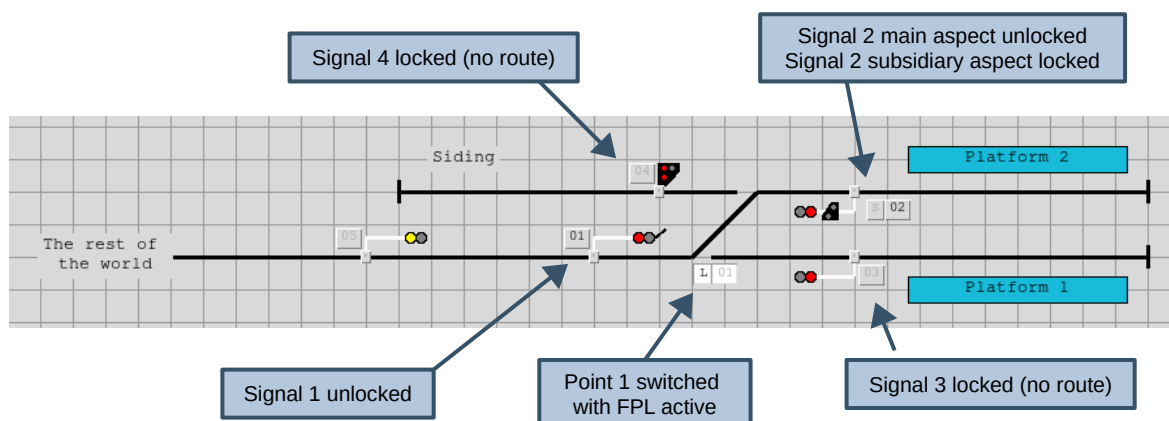
The interlocking with conflicting signals can now be tested in this configuration. If signal 1 is set to 'OFF' then Signal 2 will be locked and vice-versa. Note that when Signal 1 is 'OFF' it will display the appropriate route indication (in this case a left hand feather).

Signal 2 locked

Signal 1 cleared for Platform 2

Point 1 locked
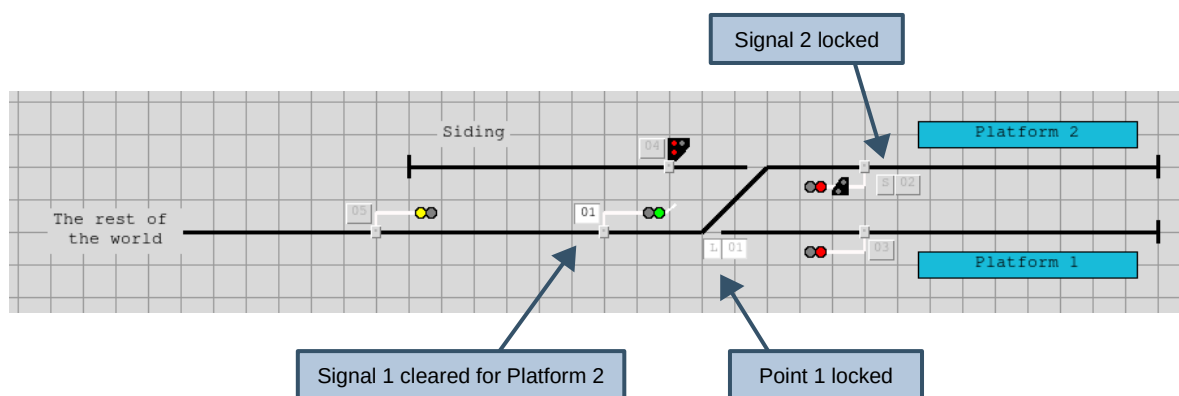
Although interlocking is defined via the signal configuration dialog , the resultant Point interlocking can be viewed (as read only) via the **Interlocking** tab of the point configuration dialog. This shows all the signals/routes which (when cleared for a movement) would lock the point.

We've used Point 1 to 'lock' Signal 5 at caution (to make it a fixed distant). This means it can never be cleared so will never be able to lock Point 1

Point 1 is additionally Interlocked with the RH1 route of signal 2 (Platform 2 => Siding)

Point 1 is interlocked with the MAIN route of all signals

Point 1 is additionally Interlocked with the LH1 route of signal 1 (Into Platform 2)
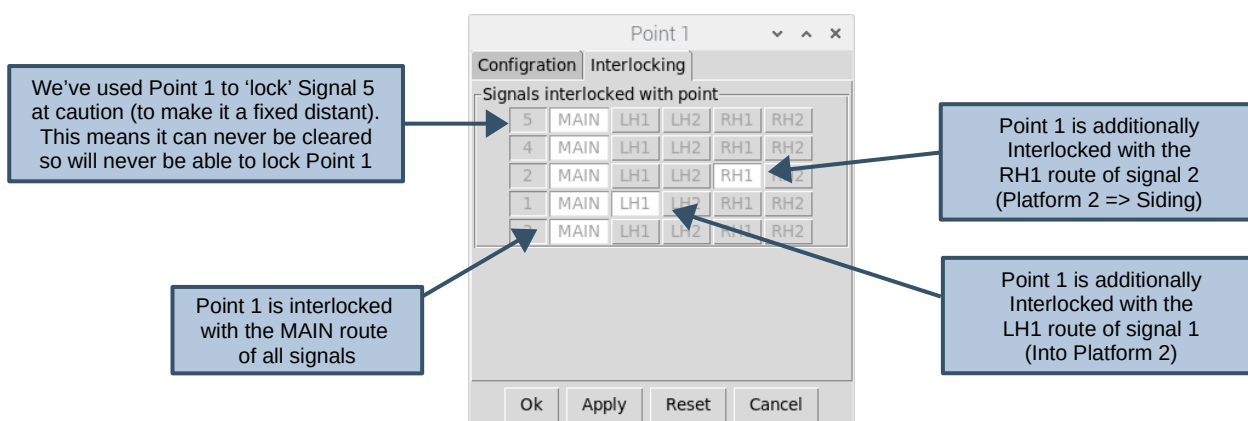
Configuring the DCC bus output

Once you are happy with your layout configuration you can 'lock' the schematic to prevent further editing by setting RUN mode. Either use the keyboard shortcut to toggle into RUN Mode (**Cntl-a**) or select via the Menubar (**Mode => Run**). This also removes the grid lines and the schematic object buttons down the left hand side of the window.

If you are running the application on a Raspberry-Pi with a Pi-SPROG DCC programmer controller then controlling the signals and points out on the layout is simple.

- Select **SPROG => Connect** from the Menubar. If connection is successful then the Menubar will show a SPROG status of CONNECTED and the Menubar DCC Power selection will be enabled.

- Select **DCC Power => ON** from the Menubar to enable DCC Power. If the power was successfully enabled then the Menubar will show a DCC Power Status of ON.

The default configuration should work 'out of the box' for the Pi-SPROG 3 v2 (as long as it has been installed correctly following the instructions provided). If you are using an earlier Pi-SPROG then the baud rate will need reducing - **Menubar => Settings => SPROG**)
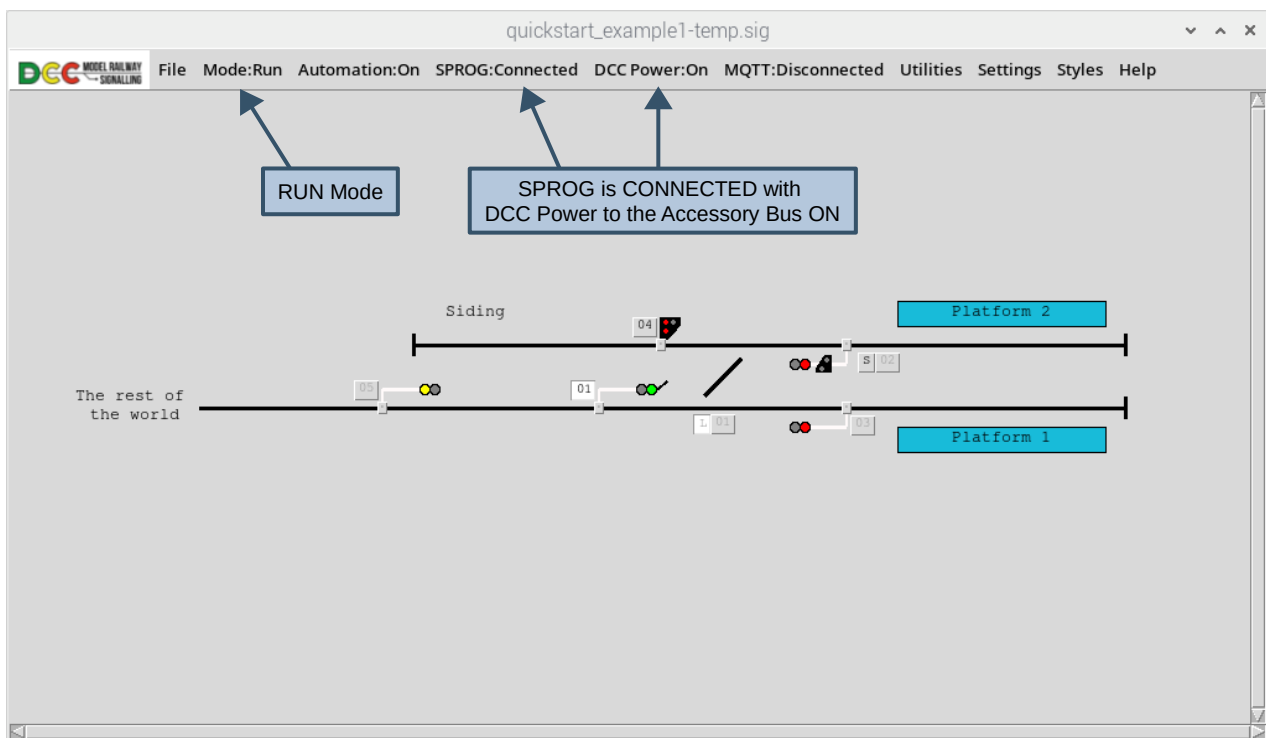
All changes you make to signals or points in the application will now send out the required DCC commands to change the signals and points on the layout accordingly.

Note that the state of the signals and points on the layout may not initially reflect the state of the points and signals shown in the application. To synchronise everything you may need to operate each signal or point on the schematic in turn - to send out the required DCC commands to set the initial state of the signals and points on your layout.

Alternatively, you can save and re-load the layout with the SPROG set to auto connect and apply power on layout load. The application will then send out the required DCC commands to synchronise your physical layout with the loaded state of your schematic. This can be configured via the SPROG settings dialog (from the Main Menubar select **Settings => SPROG**):
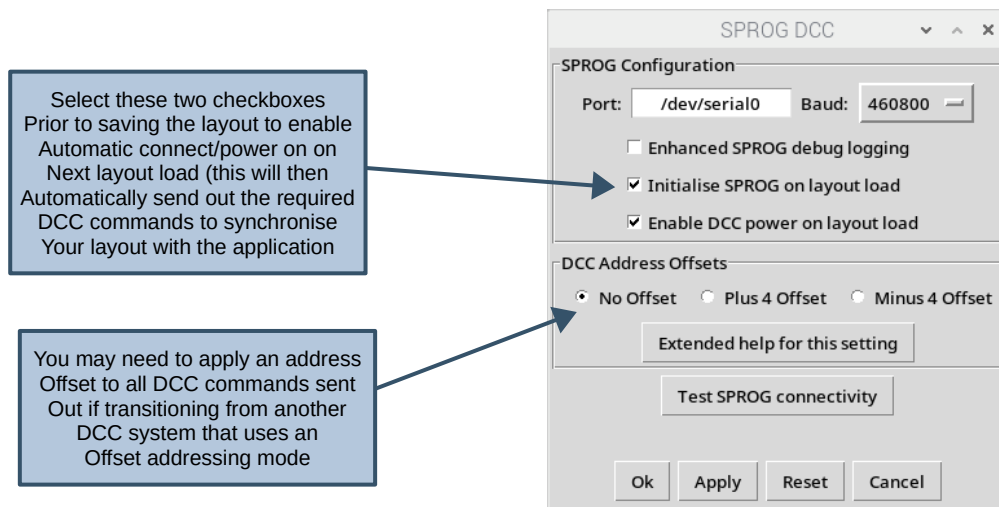
When transitioning from other DCC systems to this application (or swapping between the two) you may find that the DCC commands sent out to the accessory bus (by this system) don't align with the DCC addresses you have previously programmed for your accessories (with your other system).

The reason for this is that some DCC equipment manufacturers have interpreted the NMRA DCC Specification slightly differently, specifically how DCC addresses are encoded into the DCC packet.

If all works as expected then the default setting (No Offset) can be left selected. If you experience a discrepancy in the addressing then you should select either '+4' or '-4' as appropriate (click on the extended help button for more information on address offsets).

If you are still having problems, then you can change the application logging level to DEBUG to see what's going on 'under the hood'. This can be configured via the Logging settings dialog (from the Main Menubar select **Settings => Logging** and then select the required level

A typical log output (in the Terminal window) would be:

Configuring track occupancy

Now we have a schematic that provides basic interlocking of signals / points, and DCC control of the signals and points out on the layout. The next step is to add 'track sections' to provide a 'mimic' diagram of where the trains are on your layout.

Firstly (**in EDIT mode**), add 'track sections' to the schematic and position them accordingly. Track sections should be provided for all positions a train could occupy within the signalling scheme. In this case, trains could occupy Platform 1, Platform 2, the Siding, the section of track between the distant and home signals and the 'fiddle yard'. For this example we're going to represent a 'cassette' fiddle yard, so we'll also add track sections to represent the other cassettes.
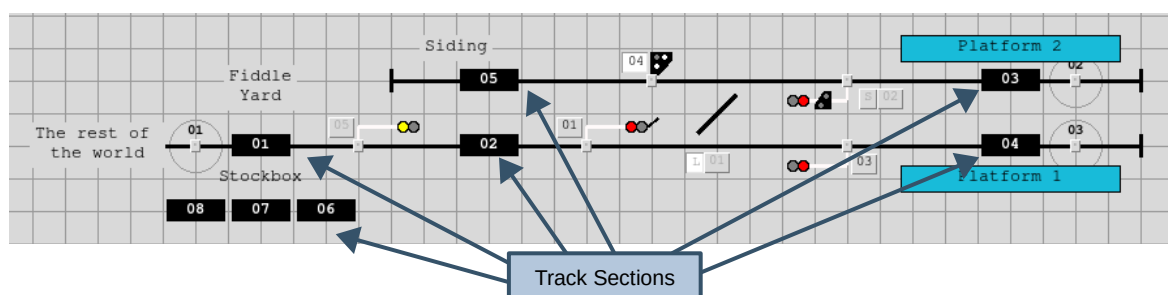
Each signal then needs to be edited to define the track section 'behind' the signal (i.e. the track section that would be 'cleared' when the signal is passed) and the track section 'ahead of' the signal (i.e. the track section that would be set to 'occupied' when the signal is passed). For signals controlling multiple routes, the track section 'ahead' needs to be defined for each route. This configuration is defined via the **Automation** tab of the signal configuration dialog.

Signal 1 controls two routes (into platform 1 and into platform 2) so the configuration would be:
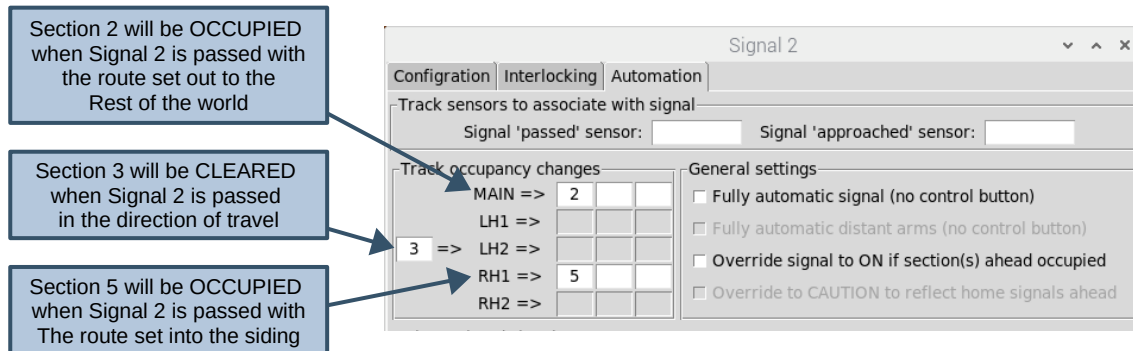
Signal 2 similarly controls two routes (the MAIN route from platform 2 out to the rest of the world and the RH1 shunting route from platform 2 back into the siding) so the configuration would be:

Section 2 will be OCCUPIED when Signal 2 is passed with the route set out to the Rest of the world

Section 3 will be CLEARED when Signal 2 is passed in the direction of travel

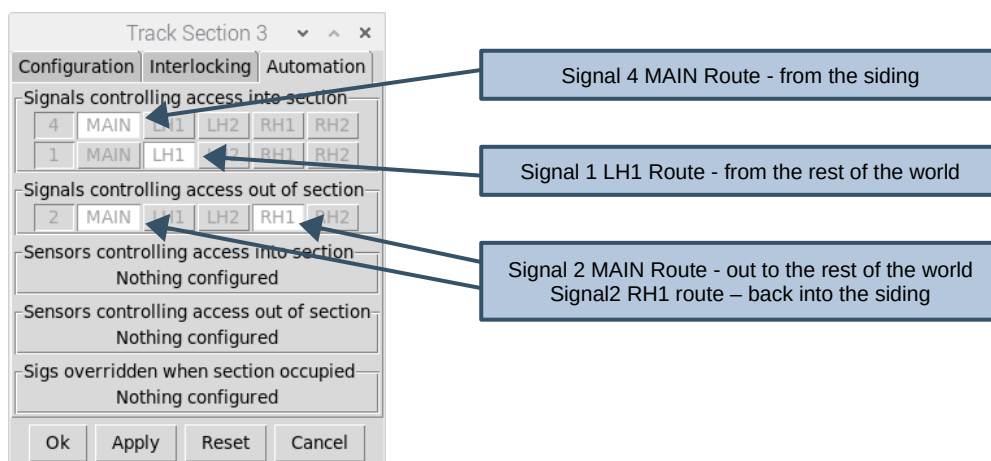Section 5 will be OCCUPIED when Signal 2 is passed with The route set into the siding

All other signals control a single (MAIN) route, so the configuration is straightforward:

- Signal 3 – Section behind is Section 4, Section Ahead is Section 2
- Signal 4 – Section behind is Section 5, Section Ahead is Section 3
- Signal 5 – Section behind is Section 1, Section Ahead is Section 2

Once the sections 'ahead of' and 'behind' each signal have been defined, the resultant configuration of each track section can be viewed (as read only) via the **Automation** tab of the relevant track section configuration dialog. This shows the signals/routes that provide access into the section, and the signals/routes that control access out of the section.

As an example, for Section 3 (Platform 2), there are 2 signals that provide access into the section, the MAIN route from signal 4 (from the siding into Platform 2) and the LH1 route from Signal 1 (from the rest of the world into Platform 2). There are also two routes out of the section (into the siding or out to the rest of the world), both of which are controlled by signal 2 (MAIN and RH1).

Signal 4 MAIN Route - from the siding

Signal 1 LH1 Route - from the rest of the world

Signal 2 MAIN Route - out to the rest of the world
Signal2 RH1 route – back into the siding

Testing track occupancy

Once all signals have been configured, track occupancy can be tested (**in RUN mode**). Although we haven't configured any external GPIO sensors yet, the 'signal passed' events to trigger track occupancy changes can be simulated by clicking on the small buttons at the base of each signal.

On entering RUN mode, all track sections will initially show as CLEAR. To set up the initial track occupancy, **right-click** on a track section and enter an appropriate train designator. Once entered, this will set the track section to OCCUPIED (white text on a black background). Track sections can also be toggled between OCCUPIED and CLEAR by left-clicking on them.

We'll first set up the trains in our cassette fiddle yard. We'll use designators of 'HST1', 'HST2' and 'DMU' to keep things simple (although for your layout you can use any designators you want).

Note that if you want to use longer train designators then you can globally adjust the width of the track section objects via the 'styles' menubar item (from the Main Menubar select **Styles => Track Sections)** to bring up the available options.
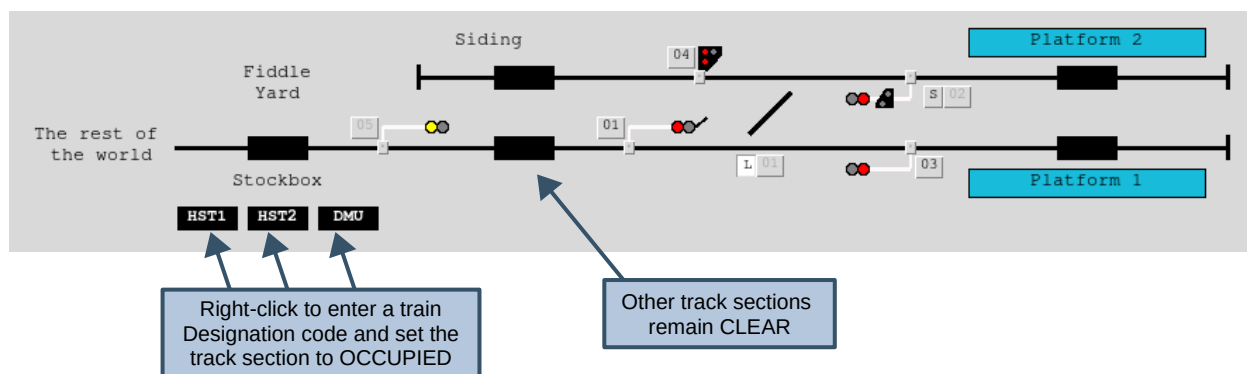
We can then choose the next train we want to run on the layout run into the layout by 'dragging and dropping' its designator from it's 'stock-box' Track Section to the fiddle-yard entry/exit Track Section (**left-click** => **move** => **left-release**).
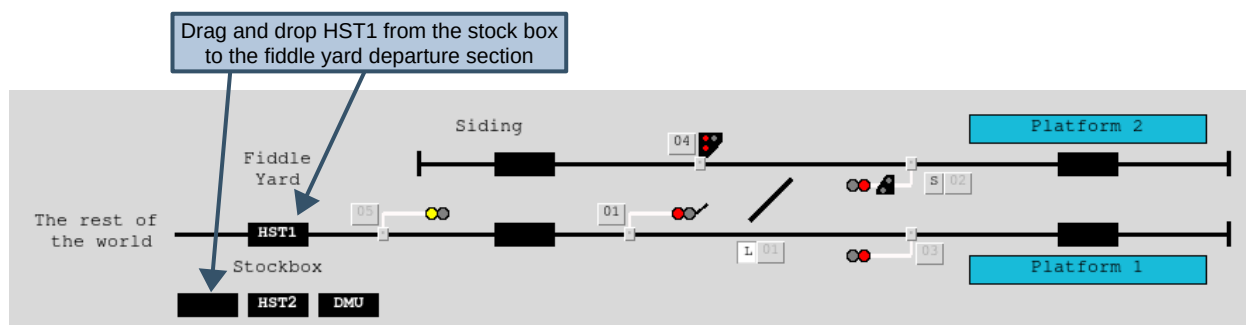
To simulate a train movement from the Fiddle yard into Platform 2, first configure the route into platform 2 (ensure all signals are 'ON', release the FPL for point 1, switch the point and then re-activate the FPL and then clear the route by setting Signal 1 to 'OFF').

Route set and cleared from the
Fiddle Yard into Platform 2
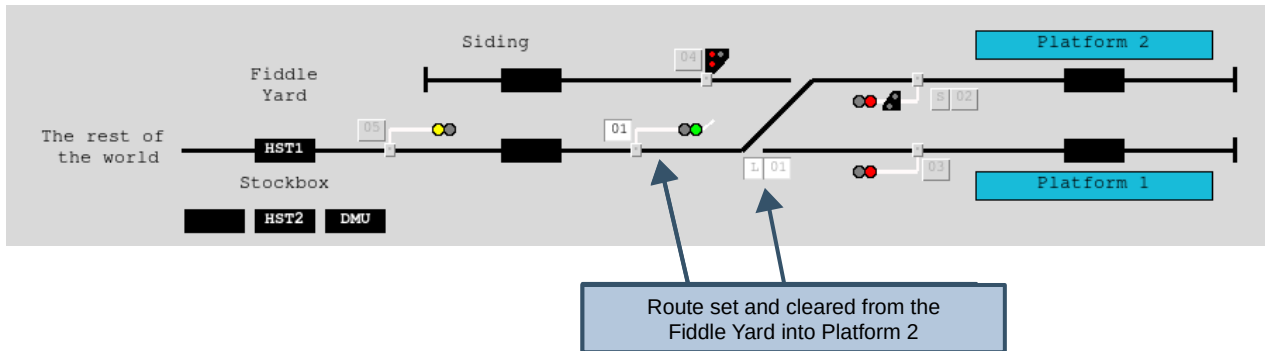
Next, trigger a 'signal passed' event for Signal 5 by **left-clicking** on the small button at the base of the signal to 'pass' the train onto the next track section.


Track Section
Is now CLEAR

Click on the button to simulate
A 'signal passed' event

Next Track Section
Is now OCCUPIED
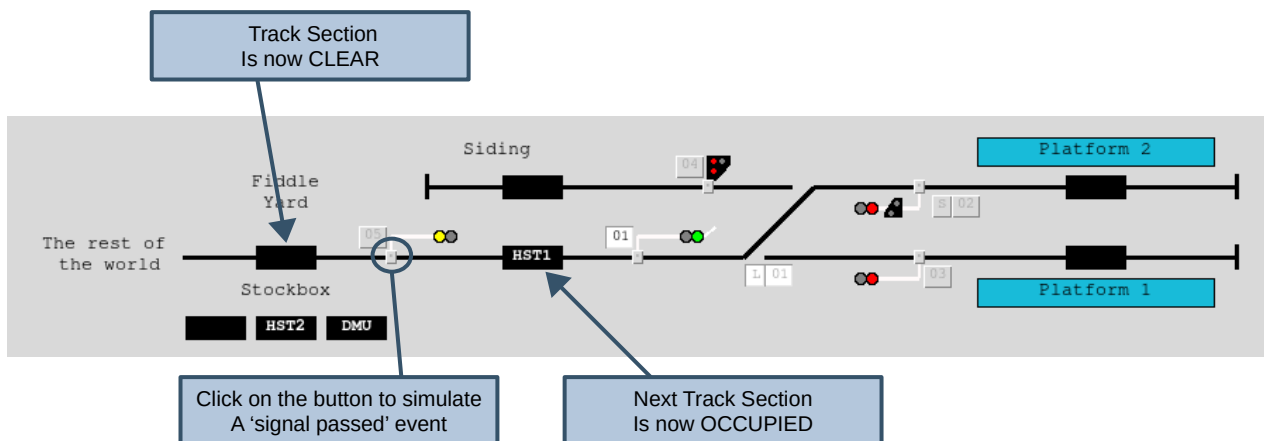
Finally, trigger a 'signal passed' event for Signal 1 to 'pass' the train into Platform 2 and then a 'signal passed' event for Signal 2 (as this signal will also be passed by the train as it heads into Platform 2) – Note that this second event will not trigger a change the track occupancy (the software recognises the signal is controlling a movement in the other direction)

Track Section
Is now CLEAR

Click on the button to simulate
A 'signal passed' event

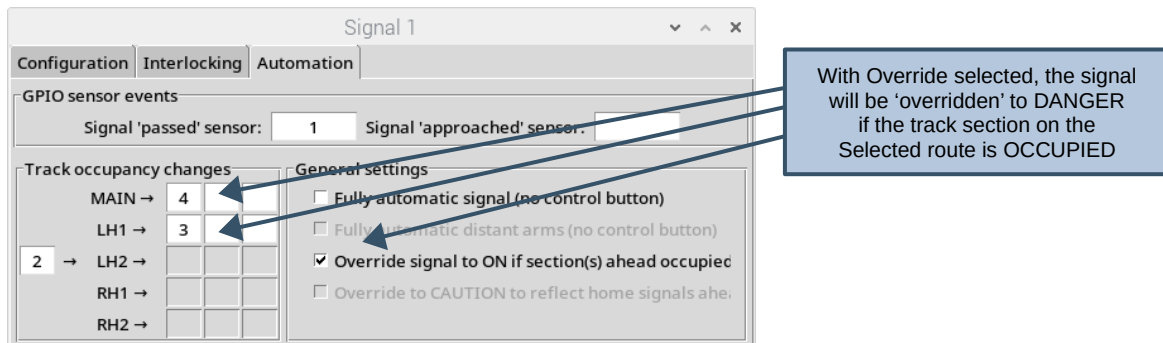Next Track Section
Is now OCCUPIED
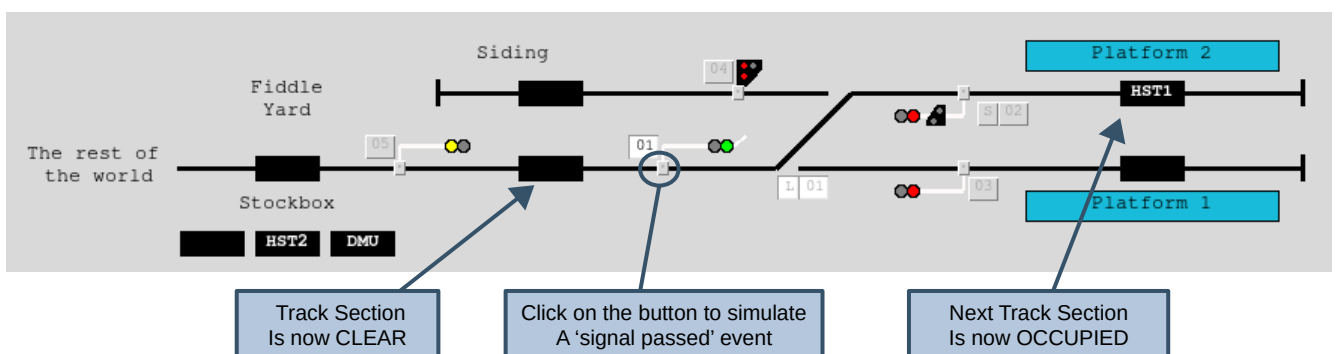
At completion of the train movement, all signals should be returned to 'ON', ready for the next movement. Other movements you could test would be:

- Platform 2 back into the siding
- From the siding into Platform 2
- Platform 2 to the Fiddle Yard
- Fiddle Yard to Platform 1
- Platform 1 to the Fiddle Yard

Automating signals based on track occupancy

Once we have proved that we have correctly configured track occupancy for the schematic, we can now automate the signals such that they are overridden to DANGER once a train has passed (and will remain overridden to DANGER as long as the Track Section ahead is occupied).

In EDIT Mode, **Double-left-click** on Signal 1 to bring up the configuration dialog and select the **Automation** tab. Then select the 'Override signal to on if section(s) ahead are occupied tab'.

With Override selected, the signal will be 'overridden' to DANGER if the track section on the Selected route is OCCUPIED
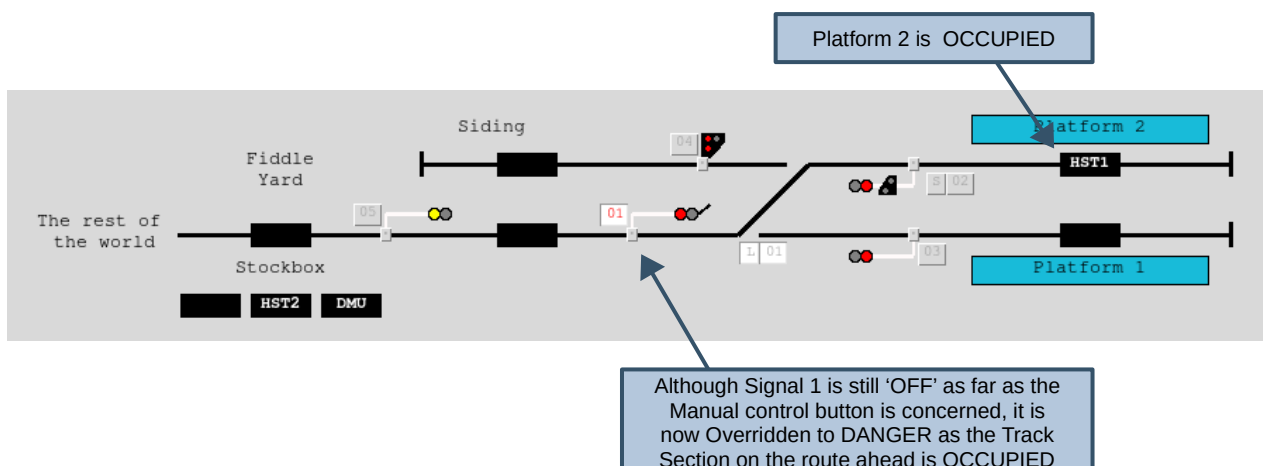
Signals 2 should then be similarly configured, but for these this signal, we also need to take into account the Fiddle Yard entry/exit track section. This Track Section (Track Section 1) should be added to the list of track sections ahead of the signal for the MAIN route:

If Signal 2 is set for the MAIN route Then the signal will be 'overridden' To danger if either of the Track Sections ahead is OCCUPIED
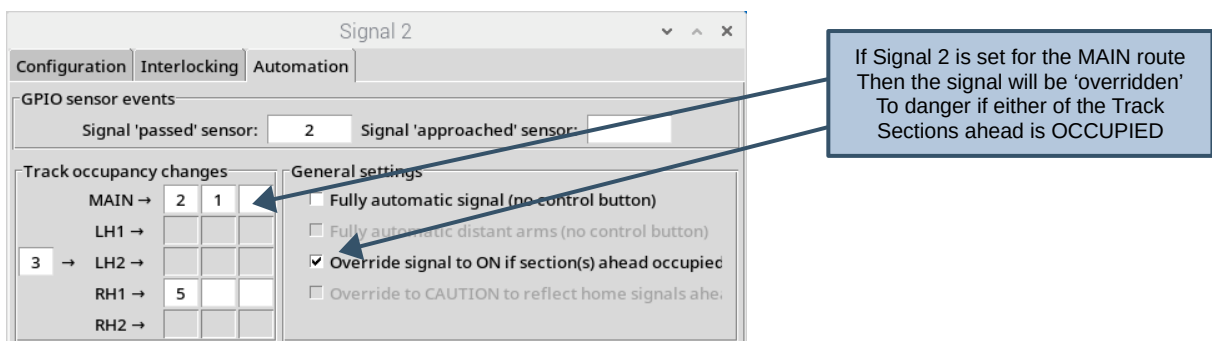
Finally Signal 3 should be configured to be overridden on track sections ahead, also taking into account the Fiddle Yard entry/exit track section. We don't need to configure Signal 5 as this is a 'fixed distant'. We also don't need to configure Signal 4 as this is a shunting signal.

Now when we run trains through the layout to test the track occupancy, the signals will automatically change to DANGER to reflect the passing of the train:

Platform 2 is OCCUPIED

Although Signal 1 is still 'OFF' as far as the Manual control button is concerned, it is now Overridden to DANGER as the Track Section on the route ahead is OCCUPIED

Enabling Signal Passed At Danger (SPAD) warnings

As per the 'real thing', it is the driver's responsibility not to pass a signal at danger (the Signalling application doesn't provide automatic train control), but if you want to shame the drivers then the application can be configured to generate popup Signal Passed At Danger (SPAD) warnings

This is configured via the general settings dialog (from the Menubar select S**ettings => General**) :

Check Box and click OK or APPLY to enable popup SPAD warnings
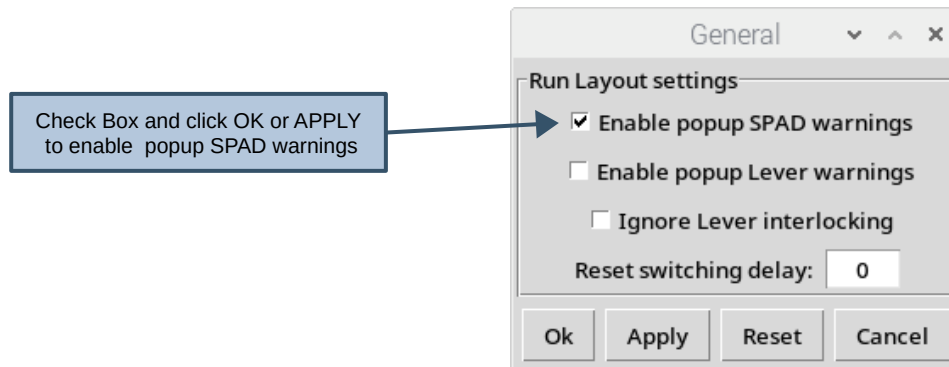
Now, each time a train passes a signal at danger, a popup window will appear with details of the event. If the popup window is left open then subsequent SPAD events will be added to list. If the list gets too long (bad drivers) then the list can be cleared down or the window closed.

Pop up SPAD warnings
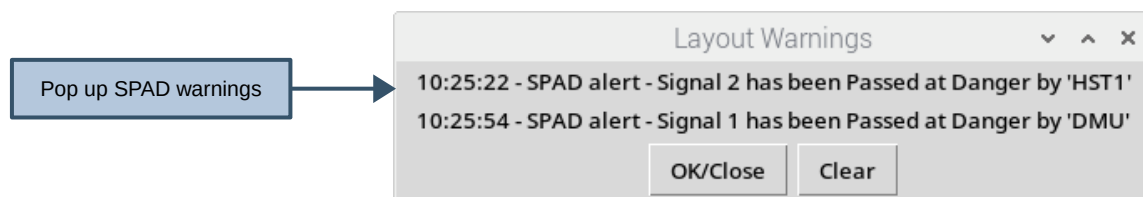
Using external sensors to drive track occupancy

The benefit of using the Raspberry-Pi is that we can connect external sensors into the GPIO ports of the Raspberry-Pi and use them to trigger the 'signal passed' events.

Firstly, we need to define the physical GPIO pins we want to use for each GPIO sensor. To do this, open the GPIO Sensors window by selecting **Settings => GPIO** from the Main Menubar. Individual 'GPIO sensors' can then be associated with each of the GPIO ports.

Note that only a subset of GPIO ports are available for use by the signalling application. Never connect external sensors to the unsupported GPIO inputs (1, 2, 3, 14, 15, 16, 17)

For this example, we'll allocate 'GPIO sensors' to each signal, matching the numbering of each signal. To do this, just enter a unique identifier for the 'GPIO sensor' against the required GPIO port (this identifier will be the identifier we will use when subsequently configuring the signals). We have a total of 5 signals so need to allocate a total of 5 Sensors, each mapped to a GPIO port.
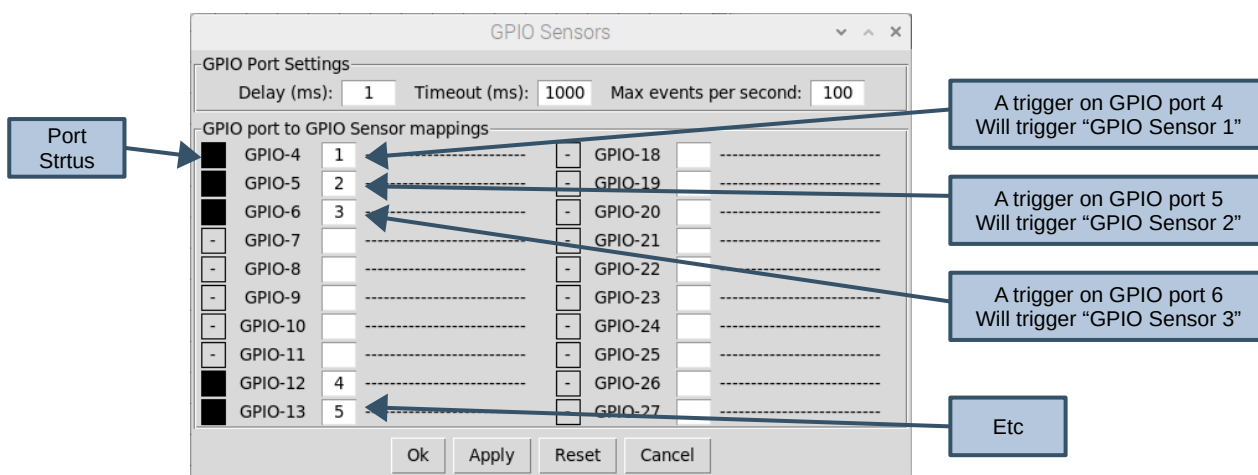
Each signal then needs to be configured to use a GPIO Sensor to generate the 'signal passed' event. This configuration is defined via the **Automation** tab of the signal configuration dialog.
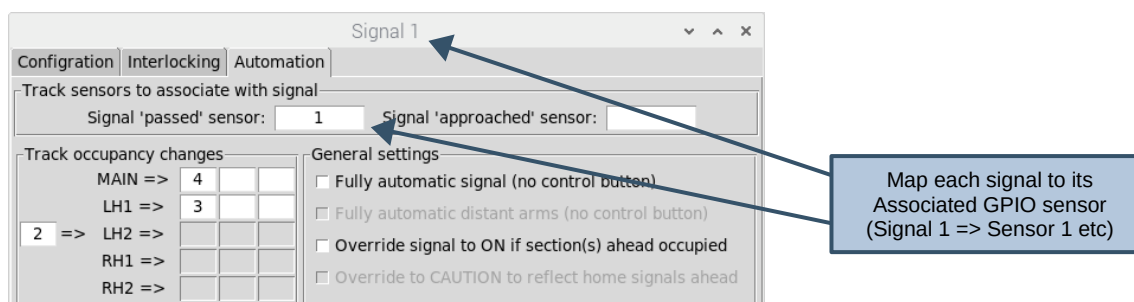

Once all signals have been mapped to their associated GPIO sensors, 'signal passed' events will be triggered whenever the associated GPIO port is connected to 0V.

Normally Open (closed when triggered) sensors can therefore be connected directly between the appropriate GPIO input pin and a 0V DC pin (available on the GPIO header).

Note that other sensor types (providing a switched voltage) should <u>never</u> be connected directly to the GPIO pins as this could damage the Raspberry-Pi. In these cases, external opto-isolators are recommended to protect the GPIO input pins.

Re-opening the GPIO settings window will now display all Signal mappings to provide an overview of the GPIO sensors (and GPIO Ports) used in the layout configuration.
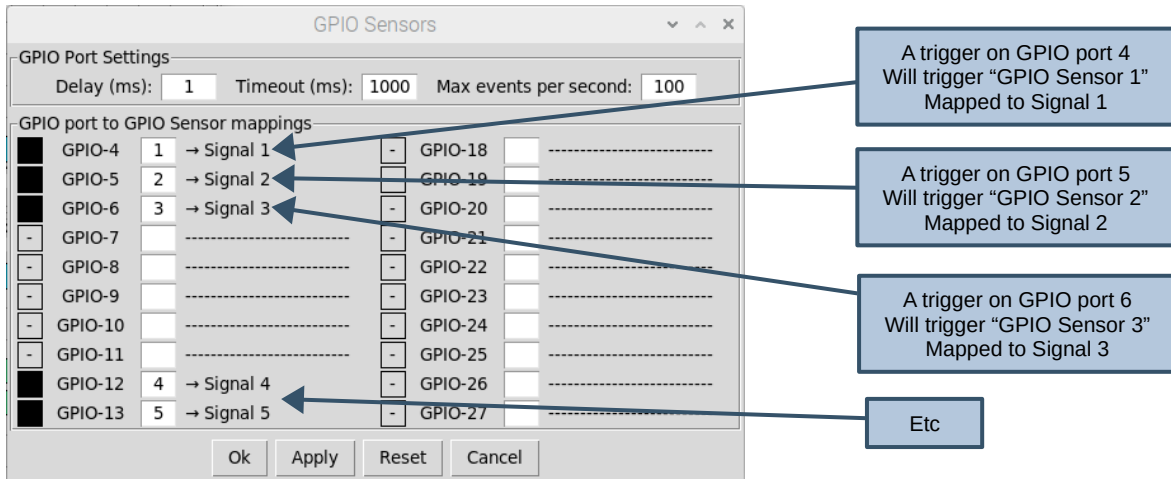
# Configuring interlocking with occupied track sections

So far, we have configured interlocking, track occupancy and signal automation (based on track occupancy), but it is still be possible for signals to be set to 'OFF' if the Track section(s) ahead OCCUPIED (although they would still be OVERRIDDEN to DANGER assuming we have configured the automation correctly as per the previous sections).

Track Section is OCCUPIED

Signal 3 is still Unlocked so Could be changed to OFF
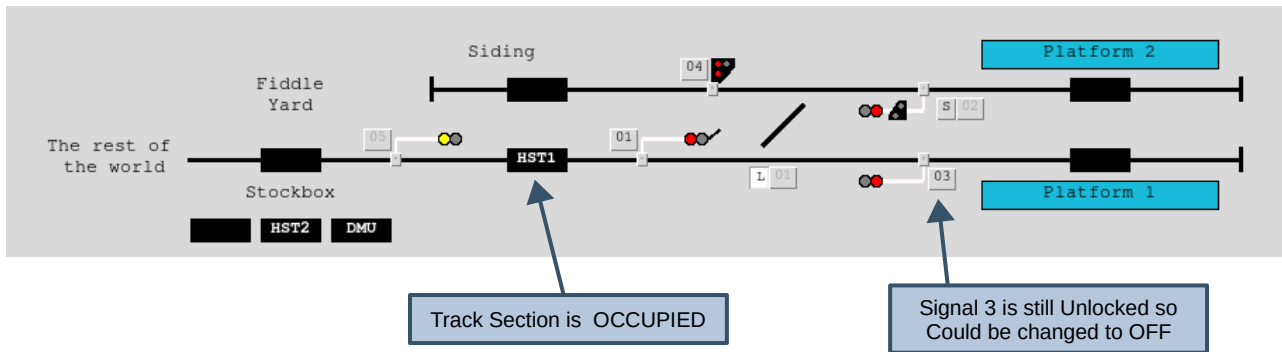
To rectify this, we can also interlock signals with OCCUPIED Track sections. In this case we want to Interlock Signal 3 with Track Section 2 and also Track Section 1 (representing a train already on the single line to/from the rest of the world). To configure Track Section interlocking, open the signal configuration dialog (in EDIT mode) and select the **Interlocking** Tab.

The MAIN route for Signal 3 will be locked (at DANGER) If Track Section 1 or Track Section 2 is OCCUPIED
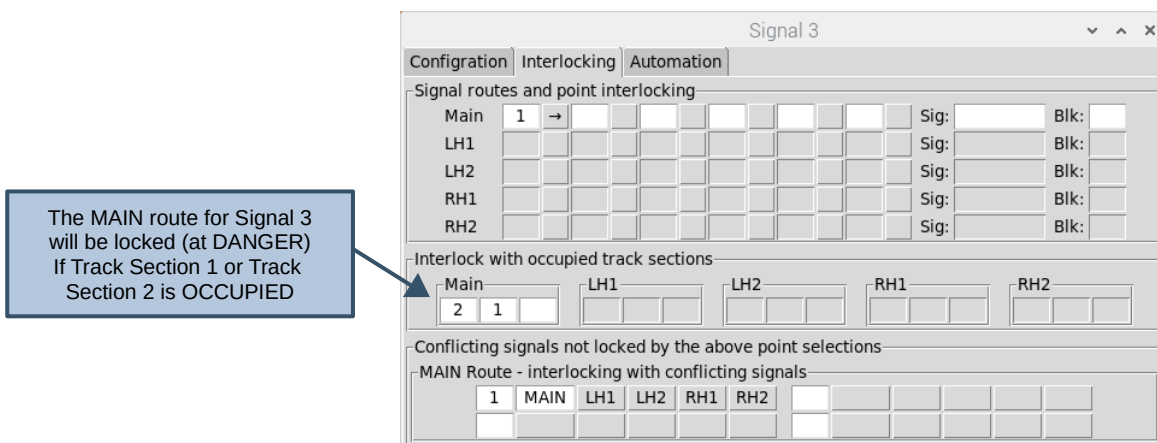
Signal 1 will now be locked (at 'ON') whenever Track Section 1 or 2 is OCCUPIED. Note that if a signal is 'OFF' when a Track Section becomes OCCUPIED then it will only be locked when returned to 'ON' (signals must always be capable of being returned to DANGER).

Track Section is OCCUPIED

Signal 3 is Now Unlocked so Cannot be changed to OFF
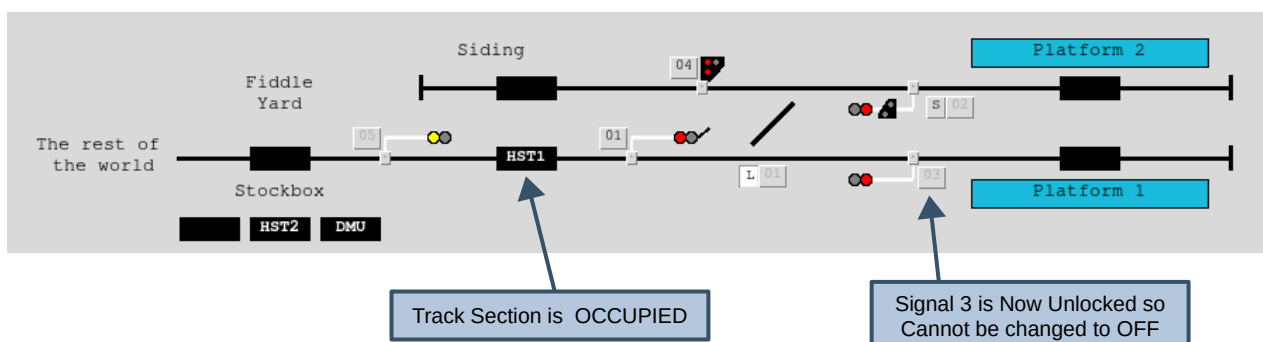
Other signals should be similarly configured to complete the interlocking schema. For example, signal 2 should be interlocked with Track Sections 1 and 2 for the MAIN route (out to the rest of the world) and Track Section 5 for the RH1 route (back into the siding).

The MAIN route for Signal 2 will be locked (at DANGER) If either of the Track sections Out to the rest of the world Are OCCUPIED

The RH1 route for Signal 2 will be locked (at DANGER) If The Track Section for the Siding is OCCUPIED

Once all signals have been configured, the interlocking of signals with Track Sections can be viewed via the **Interlocking** tab of the Track Section configuration dialog (read only).

The MAIN routes for Signal 2 and Signal 3 will be locked When Track Section 2 Is OCCUPIED

Configuring 'one-click' route setting

If you've followed the previous sections then you should now have a fully interlocked layout, with automatic passing of trains between track sections as they move through the layout.

However, setting up routes through your layout might be time consuming (especially for complex, larger layouts), so if you'd rather concentrate on keeping the trains moving, you might want to use 'route buttons' to simplify layout operation. These enable you to automatically set up and clear down routes through your layout with a single click of the mouse.

For this example, we're going to take the basic layout from the previous section and configure four 'route buttons', one for each of the main movements into and out of the layout:

- Route 1 - The rest of the world to Platform 1

- Route 2 - The rest of the world to Platform 2

- Route 3 - Platform 1 to the rest of the world

- Route 4 - Platform 2 to the rest of the world

We're also going to use intermediate 'track sensors'[2] to automatically clear down routes when the train has cleared all signals and points along the route.

Firstly (in EDIT Mode), add the 'track sensors' to the schematic to signify the 'end' of each route. Route 3 and Route 4 share the same end-point (the rest of the world), so you only need to add three track sensors to cover all four routes. You might have to slightly re-jig the schematic from the previous section to make room for the 'track sensors'.
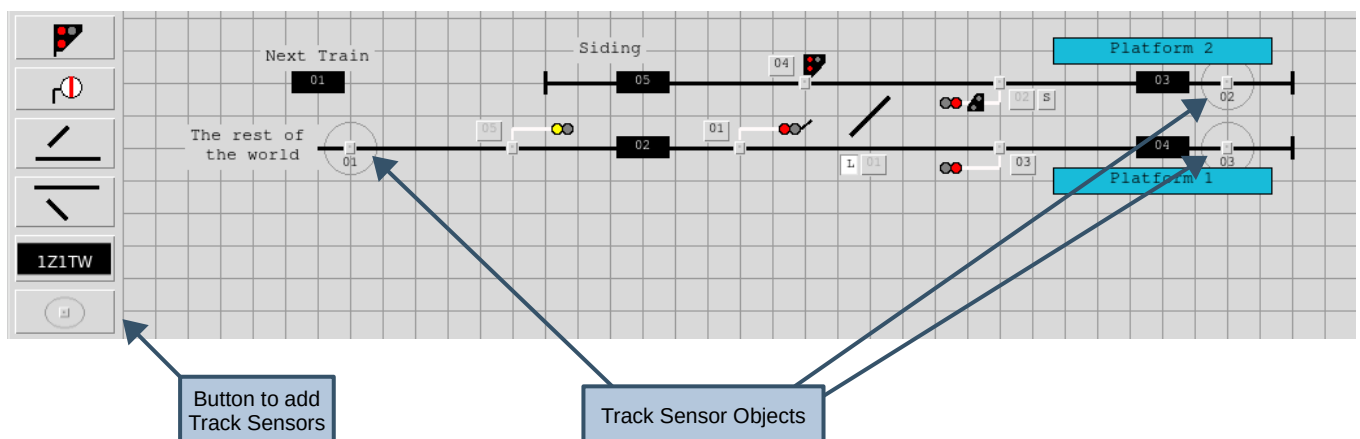
Note that we are going to refer to the 'track sensor' IDs in the route configuration, so make sure you put them in the right place (track sensor 1 going out to the rest of the world, track sensor 2 at the end of platform 2 and track sensor 3 at the end of platform 1).

---

2 The main purpose of intermediate 'track sensors' is to extend train tracking into areas of the layout that aren't fully signalled (e.g. goods yards, fiddle yards etc) or provide a finer granularity of train tracking between signals. They can be mapped to GPIO sensors (similar to how signals can be mapped to GPIO sensors for 'signal passed' events) and configured to pass trains from one track section to another when triggered (again, similar to signals). In this instance we're just going to use them to detect when a train has 'cleared' a route.

For the next step, add 4 'route buttons' to the schematic (which we are going to use to set up and clear down the routes through our layout routes).
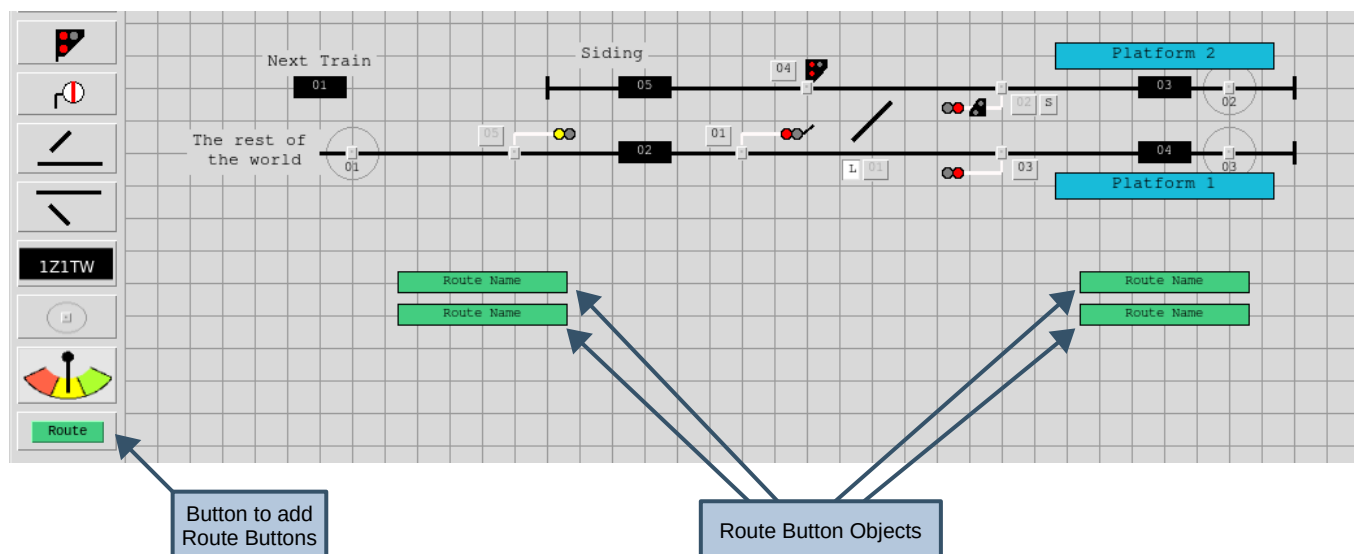

Basic configuration of each route is relatively straightforward in that we just need to define the required point settings for the route and list the signals that need to be cleared. We don't need to worry about point 2 as this is automatically switched with Point 1.

- For Route 1 (into platform 1), we need Point 1 to be 'normal' and Signal 1 to be cleared

- For Route 2 (into platform 2), we need Point 1 to be 'switched' and Signal 1 to be cleared

- For Route 3 (from platform 1), we need Point 1 to be 'normal' and Signal 3 to be cleared

- For Route 3 (from platform 2), we need Point 1 to be 'switched' and Signal 2 to be cleared

Defining the highlighting of the route is slightly more complicated in that we need to specify all points and lines which need to be highlighted (in a different colour) when the route is set. This includes any automatic points along the route.

- To find the ID of an automatic point (if you can't remember it) double click on it to bring up the Point configuration Window.

- Similarly, to find the ID of a route line, double click on the line to bring up the Line's configuration window

The route configuration on the next page assumes the following:
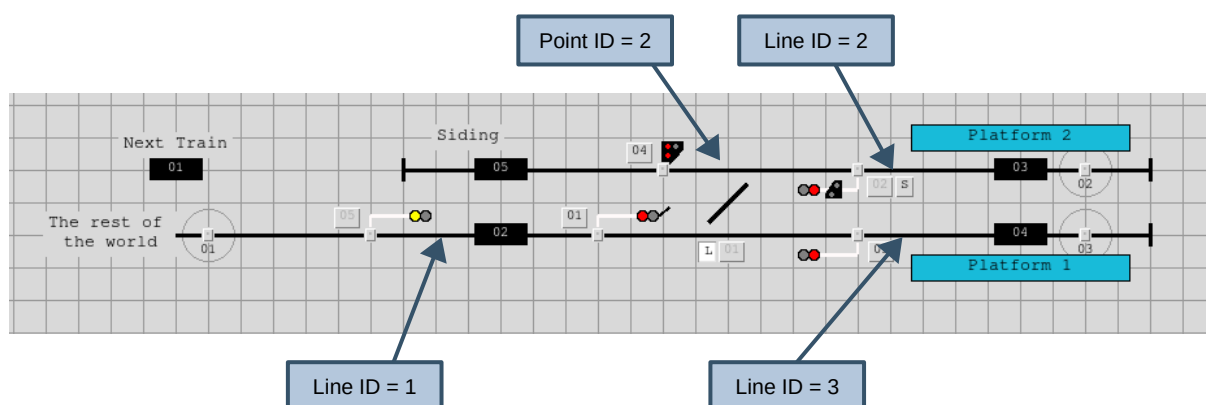
To configure a Route, double click on the button (in EDIT Mode) to bring up the Route's configuration window. For this example we're going to configure all routes with:

- A switching delay of 1 second (1000 milliseconds) – this will be the time delay between each layout change that needs to be made in order to set up or clear down the route – e.g unlocking a FPL, changing a point, re-locking the FPL, changing a signal etc.

- Automatic clear-down of routes (when the track sensor at the end of a route is triggered).

- Reset of points back to their default (un-switched) state when a route is cleared down rather than leaving them in their switched states.
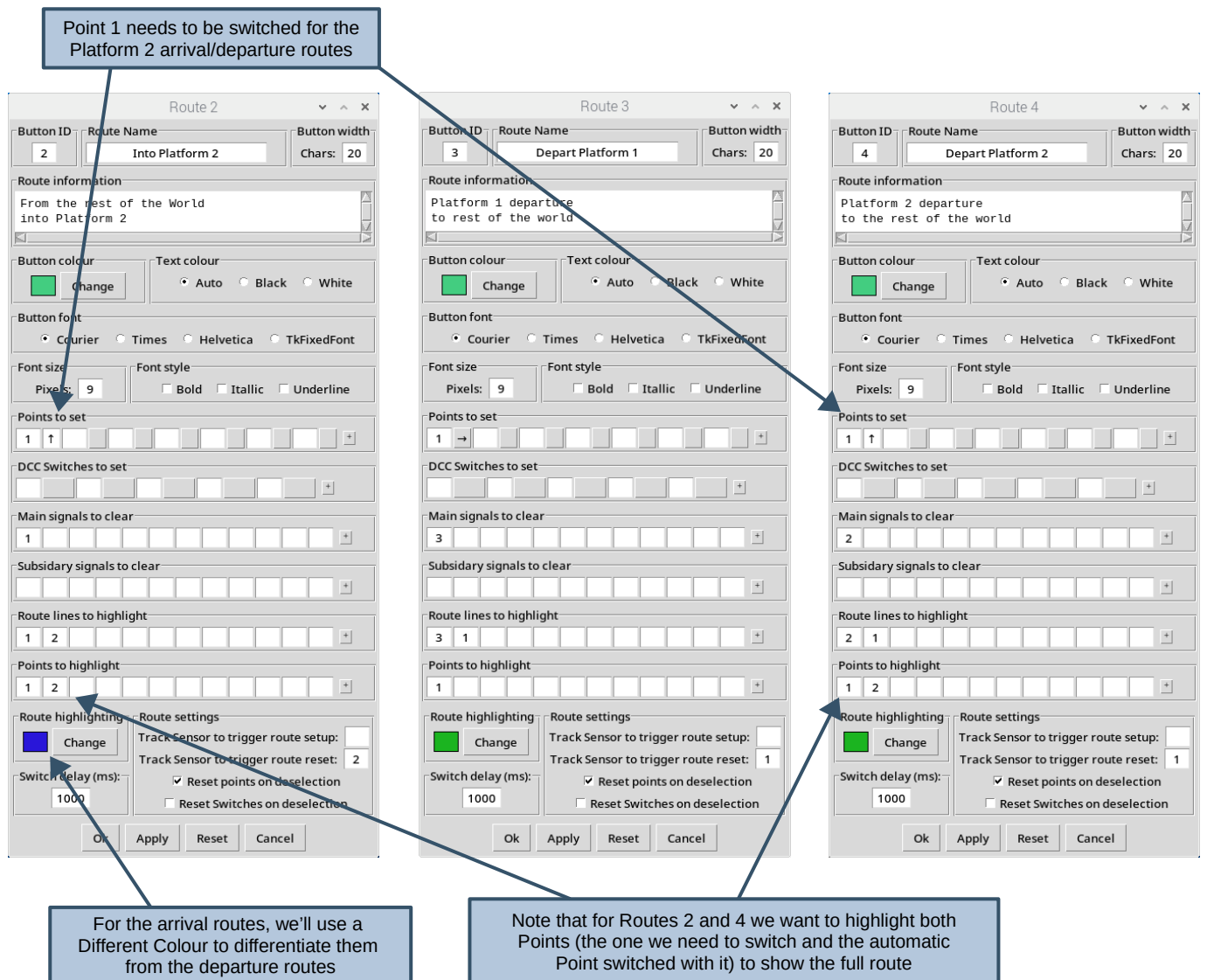
The first Route we are going to set up is Route 1 (into platform 1 from the rest of the world):

Once you are happy with the configuration click on OK to save it and then similarly configure the other 3 routes (see next page for details).

Point 1 needs to be switched for the Platform 2 arrival/departure routes

Route 2   ⌄ ^ ✕

Button ID: 2 | Route Name: Into Platform 2 | Button width: Chars: 20

Route information:
From the rest of the World
into Platform 2

Button colour: [Change] | Text colour: ● Auto ○ Black ○ White

Button font: ● Courier ○ Times ○ Helvetica ○ TkFixedFont

Font size: Pixels: 9 | Font style: ☐ Bold ☐ Itallic ☐ Underline

Points to set: 1 ↑ [+]

DCC Switches to set: [+]

Main signals to clear: 1 [+]

Subsidary signals to clear: [+]

Route lines to highlight: 1 2 [+]

Points to highlight: 1 2 [+]

Route highlighting: [Change] | Route settings:
Track Sensor to trigger route setup:
Track Sensor to trigger route reset: 2
☑ Reset points on deselection
☐ Reset Switches on deselection

Switch delay (ms): 1000

[Ok] [Apply] [Reset] [Cancel]

Route 3   ⌄ ^ ✕

Button ID: 3 | Route Name: Depart Platform 1 | Button width: Chars: 20

Route information:
Platform 1 departure
to rest of the world

Button colour: [Change] | Text colour: ● Auto ○ Black ○ White

Button font: ● Courier ○ Times ○ Helvetica ○ TkFixedFont

Font size: Pixels: 9 | Font style: ☐ Bold ☐ Itallic ☐ Underline

Points to set: 1 → [+]

DCC Switches to set: [+]

Main signals to clear: 3 [+]

Subsidary signals to clear: [+]

Route lines to highlight: 3 1 [+]

Points to highlight: 1 [+]

Route highlighting: [Change] | Route settings:
Track Sensor to trigger route setup:
Track Sensor to trigger route reset: 1
☑ Reset points on deselection
☐ Reset Switches on deselection

Switch delay (ms): 1000

[Ok] [Apply] [Reset] [Cancel]

Route 4   ⌄ ^ ✕

Button ID: 4 | Route Name: Depart Platform 2 | Button width: Chars: 20

Route information:
Platform 2 departure
to the rest of the world

Button colour: [Change] | Text colour: ● Auto ○ Black ○ White

Button font: ● Courier ○ Times ○ Helvetica ○ TkFixedFont

Font size: Pixels: 9 | Font style: ☐ Bold ☐ Itallic ☐ Underline

Points to set: 1 ↑ [+]

DCC Switches to set: [+]

Main signals to clear: 2 [+]

Subsidary signals to clear: [+]

Route lines to highlight: 2 1 [+]

Points to highlight: 1 2 [+]

Route highlighting: [Change] | Route settings:
Track Sensor to trigger route setup:
Track Sensor to trigger route reset: 1
☑ Reset points on deselection
☐ Reset Switches on deselection

Switch delay (ms): 1000

[Ok] [Apply] [Reset] [Cancel]

For the arrival routes, we'll use a Different Colour to differentiate them from the departure routes

Note that for Routes 2 and 4 we want to highlight both Points (the one we need to switch and the automatic Point switched with it) to show the full route

The Routes can now be tested in RUN Mode.

First, click on the Route 2 button (Into Platform 2) and watch the points and signals being changed in sequence to set up the route. Once the route has been successfully set up it should be highlighted:

Next Train   Siding   Platform 2

The rest of the world   Platform 1

Route set up → Into Platform 2 / Into Platform 1

Other Routes are 'locked' and cannot be selected → Depart Platform 2 / Depart Platform 1
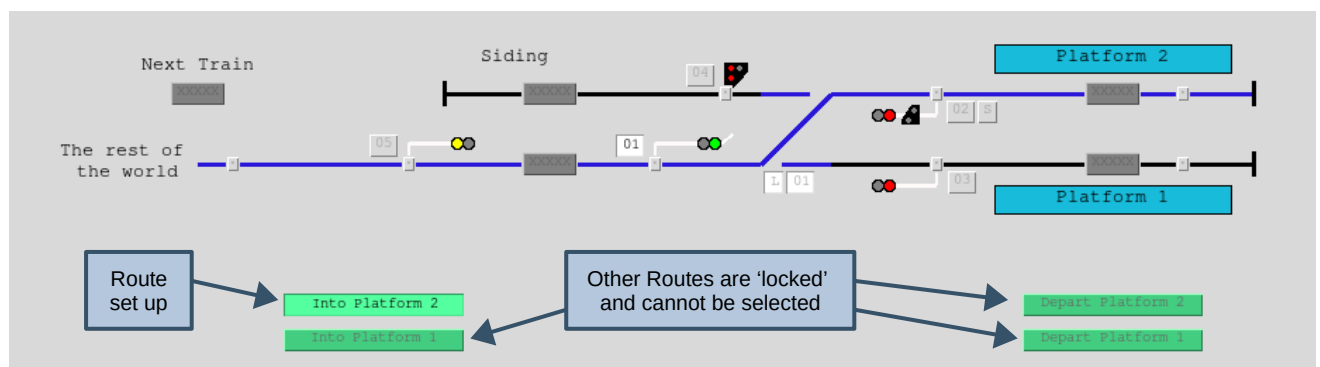
Note that (assuming you have configured interlocking correctly following the previous sections of this guide), the other route buttons will be 'locked'. If you hover the cursor over these buttons, a 'tool tip' will appear to tell you why the route button is 'locked'.

Routes can be cleared down by either deselecting the route button or triggering the track sensor at the end of the route. The signals will revert to ON and the points along the route will change to their default (un-switched) states. Each change is sequenced in a similar fashion to the route set up.

Once the route has been cleared down then it will be un-highlighted (revert to the default point/line colours) and the other route buttons re-enabled ready for you to set up the next movement.
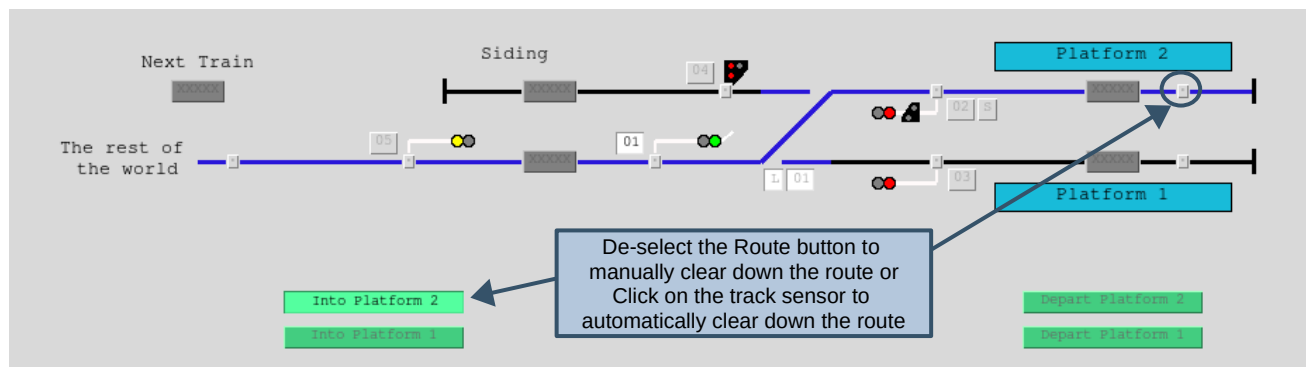

Note that if you make any changes that invalidate a route that has been successfully set-up (for example, changing signal 1 back to DANGER), the route highlighting will be immediately cleared down (to show the route has been invalidated), but any other signals or points along the route will be left in their current states (in this case, point 1 will remain switched).

Full interlocking (assuming you have configured it correctly) is preserved. For example, if you have set up a movement from the siding back into Platform 2 (point 1 un-switched and Signal 4 cleared) then the routes into and out of Platform 2 will be 'locked'.
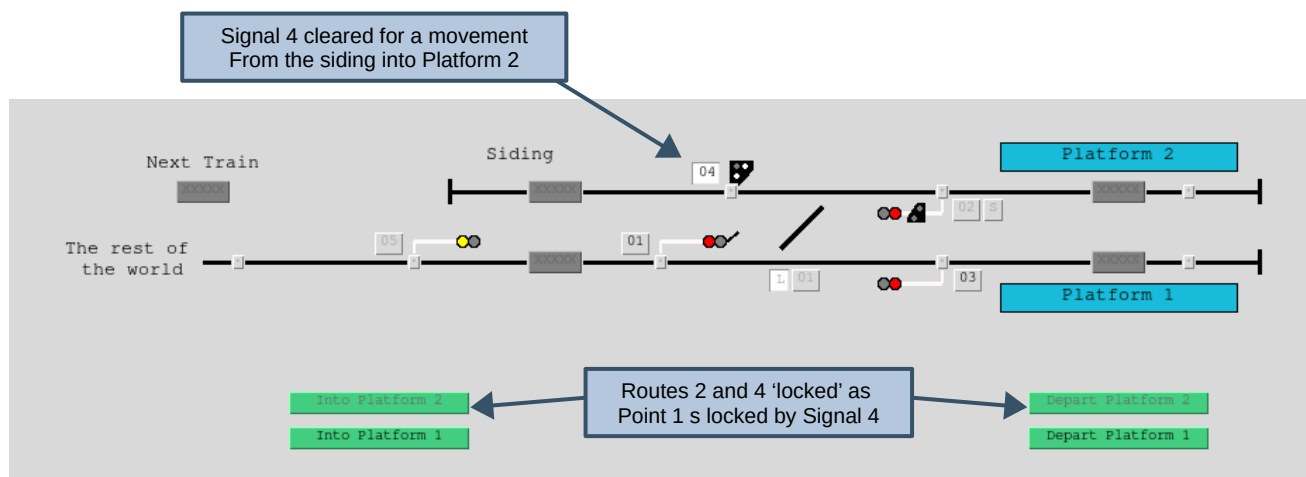
The final part of the configuration is to link external GPIO events to the track sensors so they can be triggered by the trains out on your layout.

The first thing to do is to assign another three GPIO ports to your layout via the GPIO Sensor Settings window (**Settings => GPIO**):
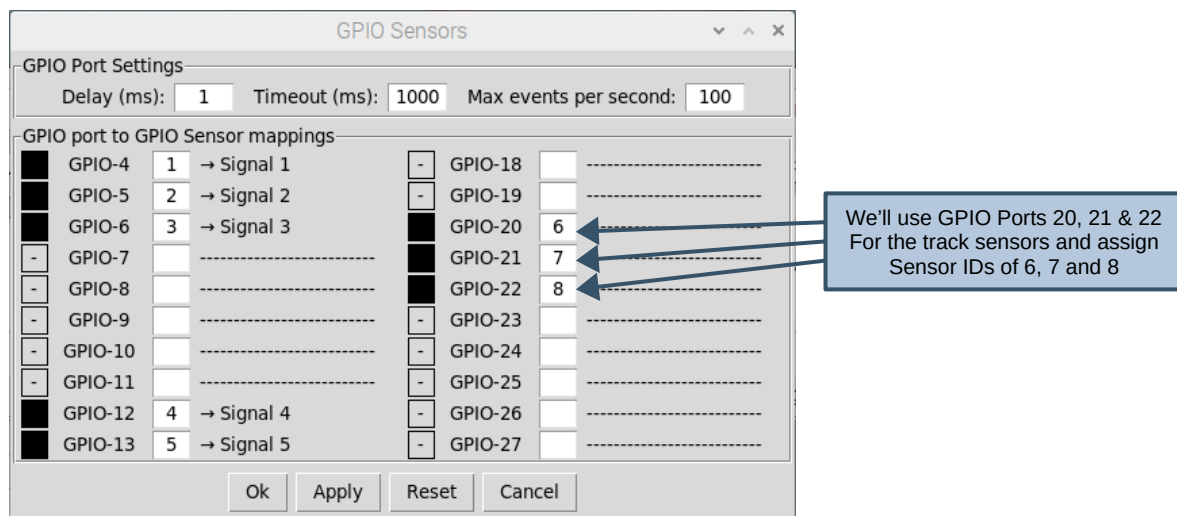
Now, in EDIT Mode, click on each track sensor object in turn and configure the GPIO sensors:

- Track Sensor 1 – GPIO Sensor 6

- Track Sensor 2 – GPIO Sensor 7

- Track Sensor 1 – GPIO Sensor 8
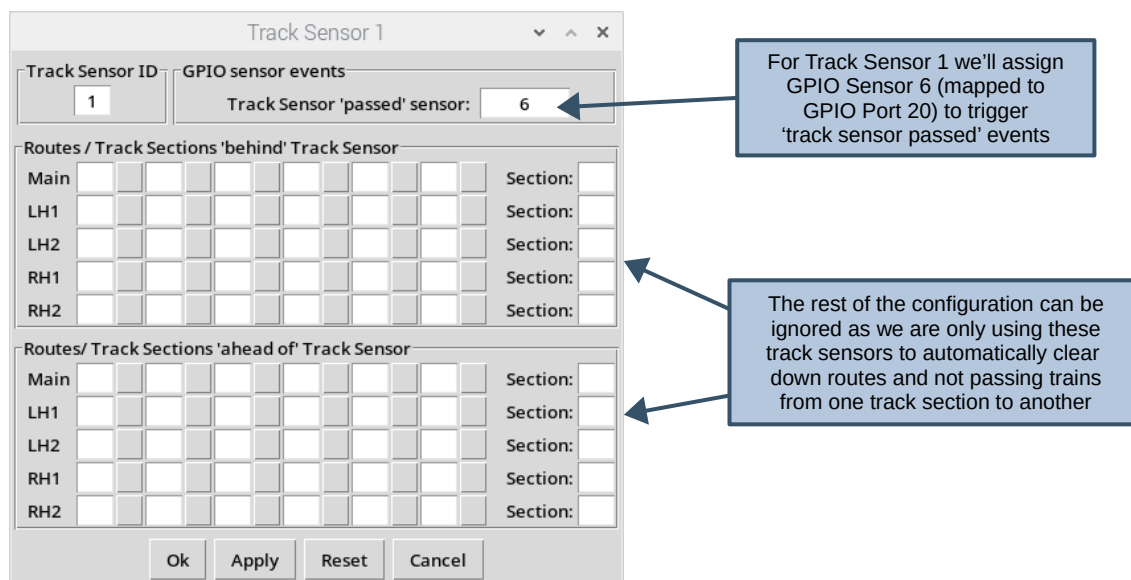
Once all three track sensors have been configured, our 'one click' route configuration is complete.

Saving and loading your layout

Once you are happy with your layout, it can be saved to file (**File => Save** or **File => Save-as** from the Main Menubar – in the case of a 'first time' save or 'save as', this will bring up a dialog to choose the filename and destination folder). Files are saved with a '.sig' extension.

Note that the current state of the layout (in terms of signal and point settings) is saved with the configuration, allowing you to pick up a running session from where you left off.

The current mode is also saved so if your layout was saved in RUN mode it will be loaded in RUN mode (and if saved in EDIT mode it will be loaded in EDIT Mode).

To load your layout select **File => Load** from the Main Menubar. This will bring up a dialog to choose the filename load the layout configuration in the mode/state at the point of save.

When a layout is loaded, the Pi-SPROG settings will default to DISCONNECTED with DCC Power OFF. If you are using the application in a 'fixed' configuration (i.e. with the Pi-SPROG permanently connected to your layout), then you may want to configure the application to automatically connect to the Pi-SPROG and turn on DCC Power on layout file load.

This is achieved via the application settings (select **Settings => SPROG** from the Main Menubar, select the appropriate check-boxes and click on **OK** or **APPLY** to save the changes).
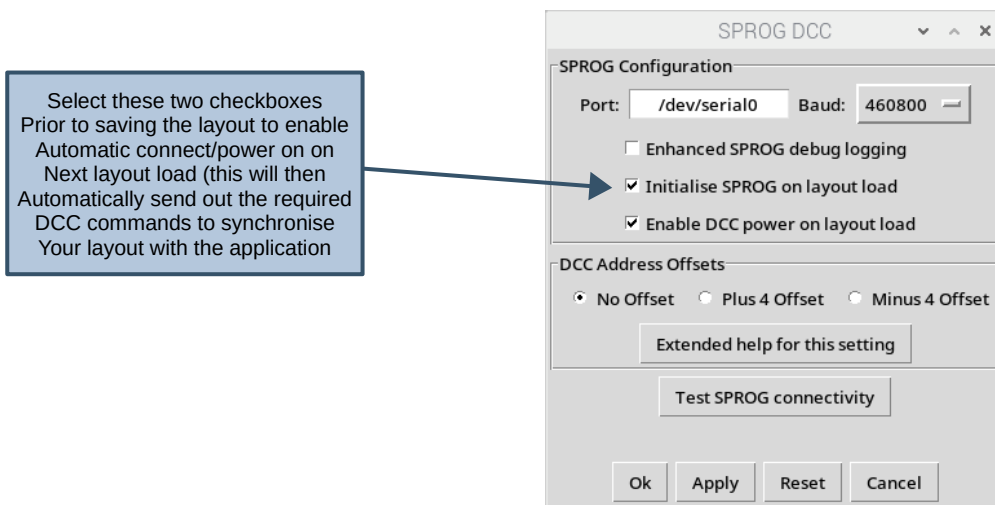
If the layout file is configured to auto connect and apply DCC power then this will also result in DCC commands being sent out to set all signals and points on your layout to to synchronise them with the loaded state of your schematic (useful in case anything has changed between sessions).

One thing to be aware of is that the DCC commands will all be sent out at the same time so, if you have a large layout and a large number of points need to be changed, this may place a significant momentary power load on the DCC accessory bus.

Note that these SPROG settings are specific to the layout configuration (i.e. saved and loaded as part of the layout file) and will not apply to other layouts you load into the application. Similarly the Track Sensor configuration (in terms of GPIO port mappings to 'Track Sensors') is specific to the layout and will be saved / loaded with the rest of the layout configuration.

Quick start example 2 – semaphore signals

In the first quick-start example, we configured a layout using colour light signals. Depending on the period modeled, you might prefer the use of semaphore signals on your layout.

For Quick Start Example 2 we're going to change our colour light signalling scheme (from Quick Start Example 1) to use the equivalent semaphore signal types
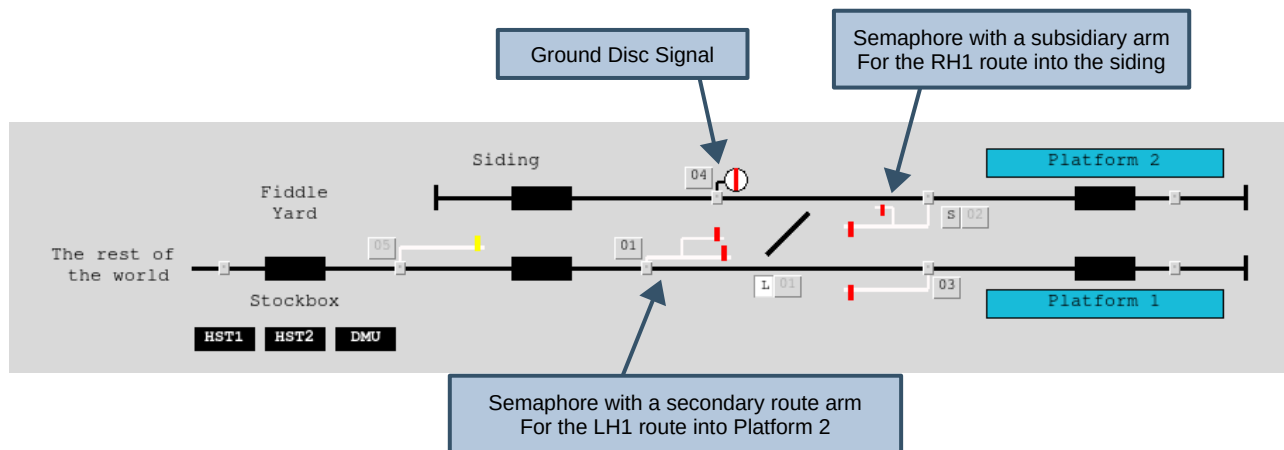
As we have fully configured the colour light signalling scheme in terms of interlocking, track occupancy and automation, its easy to swap between the two signal types by editing each signal in turn ~The only things to change in the configuration are:

- Signal type and subtype – select 'semaphore' and then either home or distant
- The route indications – additional semaphore route arms rather than route feathers
- DCC addressing – each signal arm need to be configured with a single DCC address

Firstly, edit signal 1 and change it to a Semaphore Home signal with Route Arms (a MAIN route arm for Platform 1 and a LH1 route arm for platform 2). Each signal arm should be configured with its corresponding DCC address. The remainder of the configuration (on the Interlocking and Automation tabs) remains unchanged from the colour light signalling example.

The other signals also need to be changed to Semaphore types. Note that for signal 2, the MAIN route (controlled by the main home arm) remains the route out from platform 2 to the rest of the world. A subsidiary arm controls the RH1 route back into the siding.

Changing the other signals (Signals 3, 4 and 5) to semaphore types is relatively straightforward in that only the signal type, sub-type and DCC addresses need to be changed. The remainder of the configuration remains identical to the colour light signalling example.

Quick start example 3 – signal box levers

Simulated signal box levers

For ultimate realism, you can also control your signals and points with virtual signal box levers, allowing to to simulate the layout and operation of a real signal box.

For Quick Start Example 3, we will modify our semaphore signalling example to:

- Add signal box levers and 'connect' them to the signals/points on the schematic.
- Configure the point and signal buttons to be hidden in RUN Mode (this is optional).
- Annotate the points and signals on the schematic with their corresponding lever numbers (alternatively you could use the 'bulk renumbering' utility to align the Point/Signal IDs with the IDs of their corresponding Levers if not hiding the point/signal buttons). Note that you may need to turn off 'snap-to-grid' to position the text boxes appropriately.
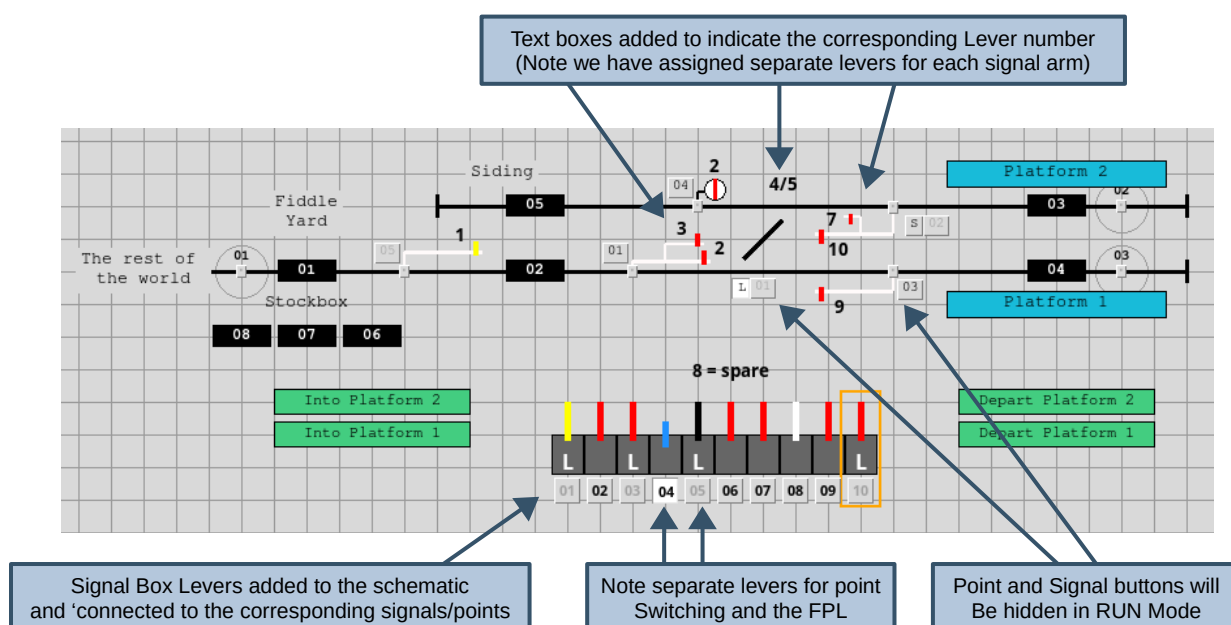
The first step is to add the levers to the schematic. These are initially created as White ('spare') levers , but will automatically inherit the correct lever colour when connected (Yellow for Distant signals, Red for Stop signals, Blue for FPLs and Black for Points).

To connect the levers to their respective signals and points, double-left-click on each Lever in turn to bring up the configuration dialog. For Lever 1 the configuration would be:

For this example, we are going to connect Levers 2 and 3 to different signal arms of Signal 1 (although you could use a single lever to controlling ALL routes if you prefer):
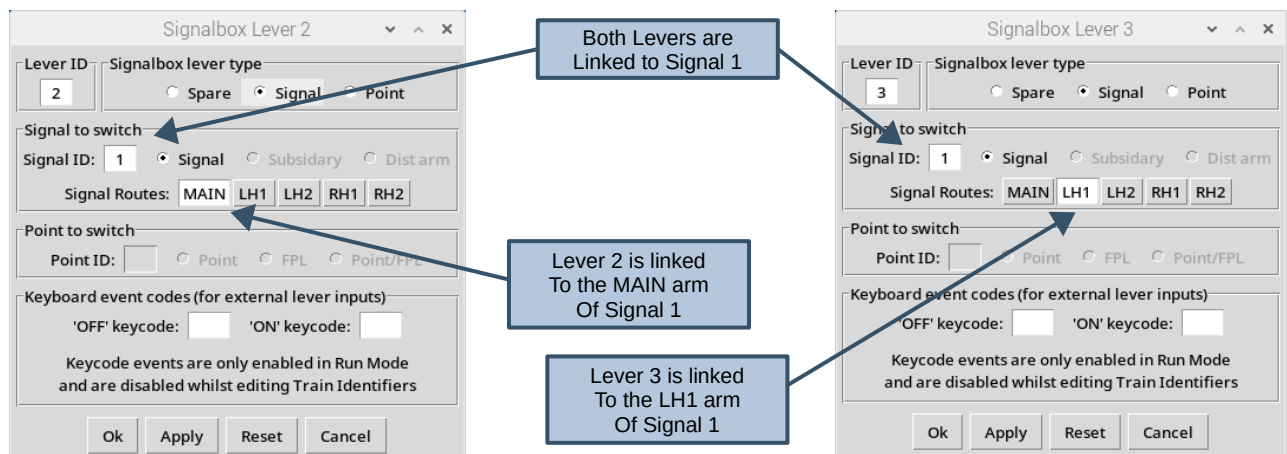

Similarly, we'll use two Levers for Point 1 (one to control the FPL and one to switch the point):
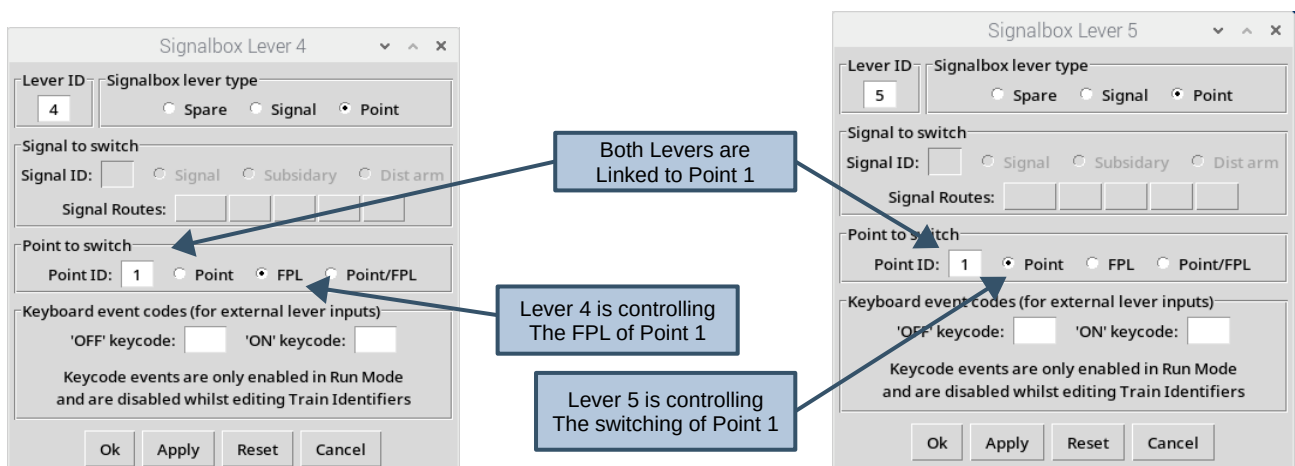
The remaining Levers can then be configured as appropriate (note that we're also going to use separate Levers for each signal arm of Signal 2).

To finalise the schematic, the signals and points on the schematic need to be annotated with their respective lever numbers (by adding text boxes). The the point and signals buttons can then be configured to be hidden in run Mode – simply edit each Point and Signal in turn, select 'Hidden' in the 'Control Buttons' section of the dialog and apply the changes.

Operation is identical to the signal/point control buttons, Levers will only be unlocked when their respective Point or Signal is unlocked. When they are unlocked they can be switched as required.
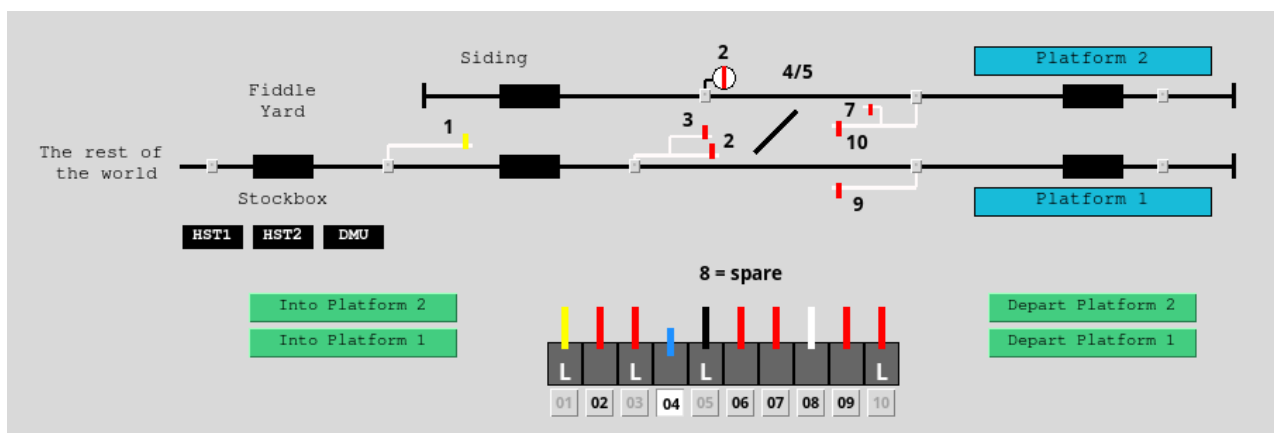
Physical switches/levers

If you prefer to use physical levers or switches for controlling the signals and points on your layout (such as the DCC Concepts Cobalt S Levers) then these can easily be integrated with the application through the use of readily available USB Keyboard encoders.

Choosing and configuring USB keyboard encoders

These devices can be connected into the computer running the application and generate key-press events for the application whenever an input is momentarily activated (they effectively act as an additional USB keyboard for your computer).

Many different units are available from many different suppliers, but we would recommend the Ultimarc I-PAC series of encoders. Although these units are primarily designed for arcade gaming, they have a number of useful features that make it ideal for integrating signal box levers:
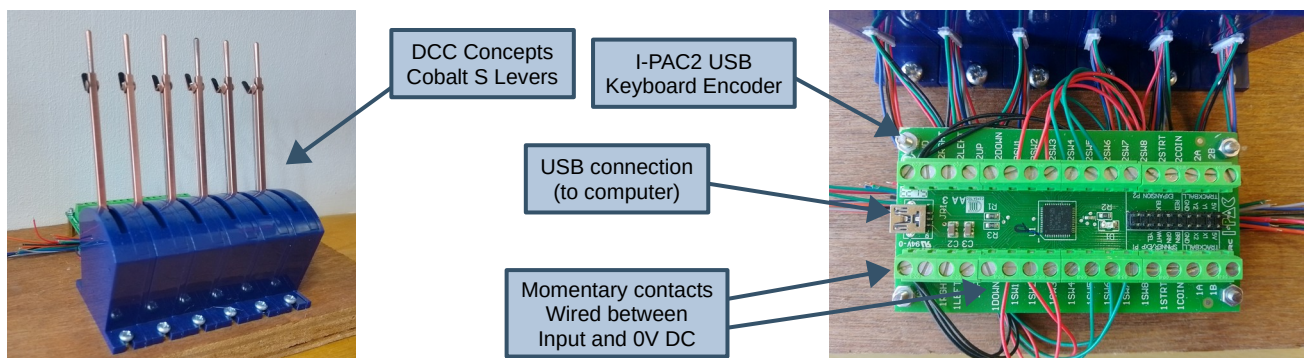
- Units are available that provide either 32 or 56 inputs, which should be sufficient for most signal box lever frames (you will need to use 2 inputs per Lever)

- Each input can be programmed to simulate any keyboard event (printable characters, function keys or other control keys present on a standard PC keyboard). Note that the unit would need to be programmed using the WinIPAC PC application, available to download from the Ultimarc website (this runs on Windows PCs only)

- Multiple units (when programmed with different keyboard mappings) can be used in parallel, allowing you to create large lever frames if your layout is complex.

Although you can program each I-PAC input to simulate any key-press event, you need to avoid key-press events which will have an undesired effect on either the application or the platform you are running the application on:

- The following keys are reserved for the application: Ctrl , A, M, R and the Arrow keys

- Other keys which may produce undesired effects are: F10, Caps-Lock and the Windows key

Choosing and connecting external switches/levers

Any type of external switch or lever can be connected into the keyboard encoder as long as it produces a MOMENTARY output, as we only need a single keyboard event to switch the lever (switches or levers that provide a LATCHING output cannot be used). In this respect, the DCC Concepts Cobalt S Levers are ideal as they provide the required MOMENTARY outputs.

DCC Concepts Cobalt S Levers

I-PAC2 USB Keyboard Encoder

USB connection (to computer)

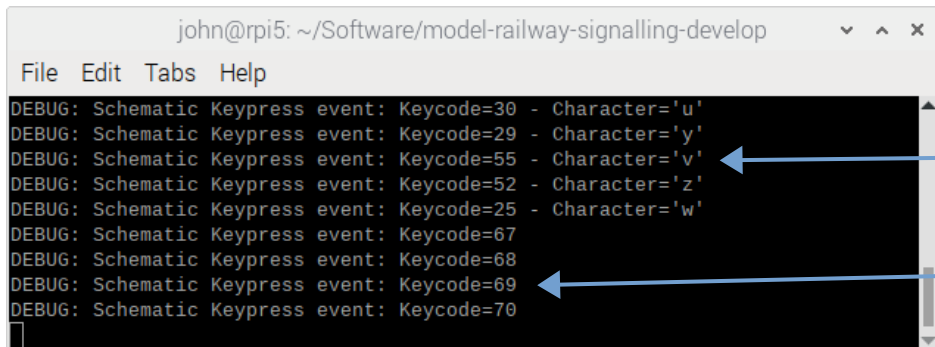Momentary contacts Wired between Input and 0V DC

Configuring signal box levers to use external inputs

Once we have configured the USB keyboard encoder and connected the momentary contacts of the external levers into the USB keyboard encoder, the signal box levers can then to be configured to 'act' on the external key-press events.

The first thing you need to do is to find the 'key-code' for each key-press event produced by the USB keyboard encoder. The application uses 'key-codes' rather than keyboard characters as not all key-press events will produce printable characters (e.g. Function Keys).

The easiest way of finding out the key-codes you have programmed is to set the application into RUN mode and then set the log level to DEBUG (**settings => logging** from the main menubar). Each key-press (with its corresponding key-code) will now be logged to the terminal output:
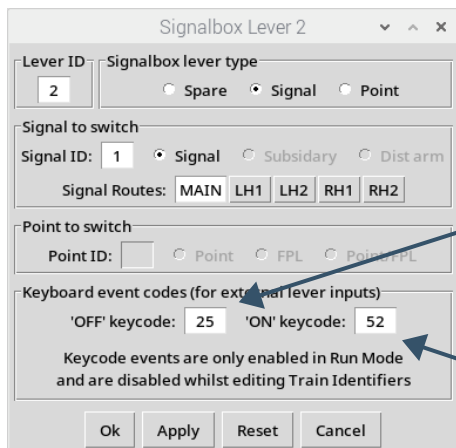

Once the required key-codes are known, then they can be mapped to each signal box lever via the Lever configuration dialog:

Operating with external switches/levers

The signalling application will respect interlocking by default, but the external levers or switches you connect into the system will have no mechanism for interlocking. Each simulated lever on the schematic has a clear indication as to its locking state, and it is therefore the responsibility of the user to respect the interlocking and not 'pull' any levers that are locked.

If required, the application can be configured to generate pop-up warnings if an external lever is operated whilst 'locked' (to shame the signaler).

There is also an option to ignore interlocking (with or without pop-up warnings) if you wanted the state of the schematic (and hence the signals and points on your layout) to always reflect the state of the external levers/switches you are operating the layout with.
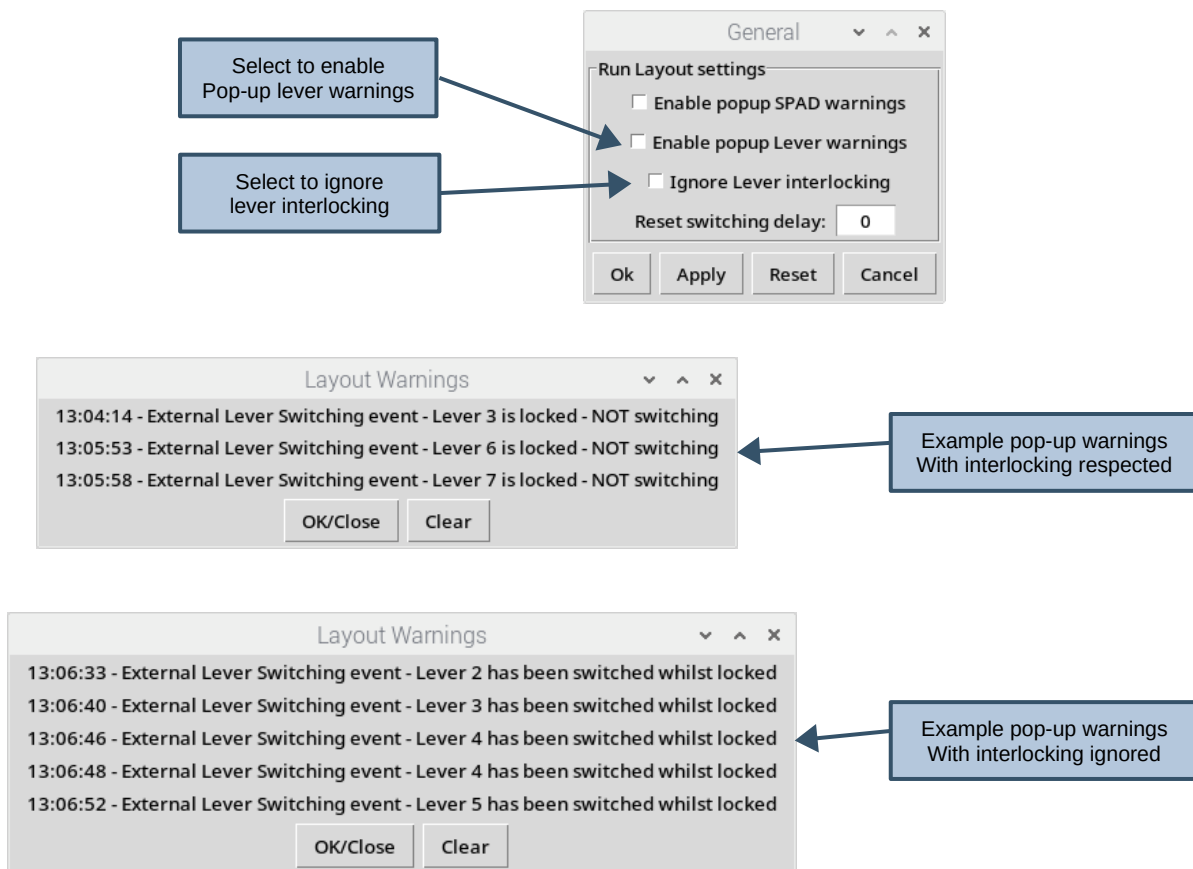
When using external levers it is important to note the following:

- Key-press events will only switch signal box levers in RUN Mode

- Key-press events will only switch signal box levers if the signalling application 'has focus'. If you are doing something in another window, for example editing a text document, then the key-press events will be re-directed to that other window.

- Switching of levers by key-press events will be disabled when editing train designators in Track Sections in RUN Mode (otherwise the characters you type for the train designator may trigger switching of the levers depending on your chosen key-press mappings).

Quick start example 4 - more automation

Now we've mastered the basics, we'll create a new layout to demonstrate some of the advanced automation features provided by the application, with a double track main line junction signalled with 4 aspect colour light signals:
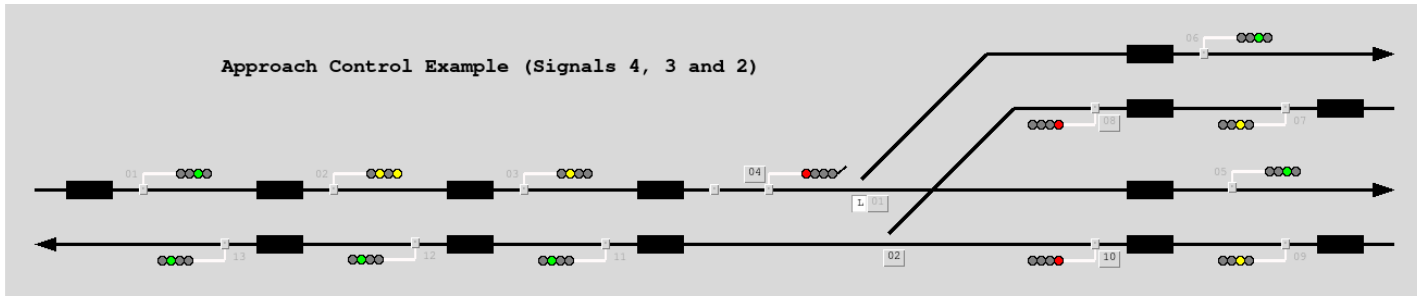
As this layout is quite large, first you need to increase the size of the canvas. To do this, select **Settings => Canvas** from the Main Menubar and increase the width accordingly.
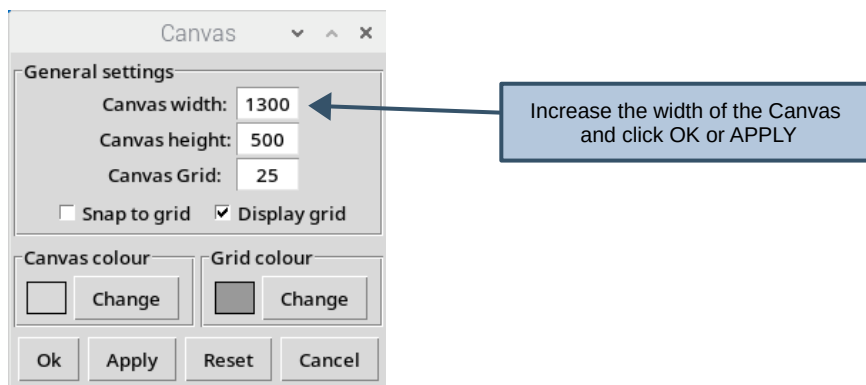
Configuring interlocking

Now draw the basic schematic as per the diagram above and add the signals and track occupancy sections in the appropriate positions. Point 1 should be configured with a FPL, but point 2 can be left 'as is' as this is a trailing point with respect to the direction of travel. Signal 1 should be configured with a left hand route feather for the diverging route.

Interlocking can then be configured for Signals 4, 8 and 10:

- Signal 4 needs to be interlocked with Point 1 (Point 1 needs to be NORMAL for the MAIN route and SWITCHED for the LH1 diverging route).

- Signal 8 needs to be interlocked with Point 2 (Point 2 needs to be SWITCHED).

- Signal 10 needs to be interlocked with Point 2 (Point 2 needs to be NORMAL).

As this layout includes a diamond crossover, we also need to interlock the conflicting Signals:

- Signal 4 needs to be interlocked with the MAIN route of Signal 8.

- Signal 8 needs to be interlocked with the MAIN route of Signal 4.

Configuring track occupancy

The track occupancy configuration should then be defined for each signal:

- Signal 4 needs to be configured with a 'section ahead' for both routes (Section 5 for the MAIN route and Section 6 for the LH1 route) and a 'section behind' (Section 4).

- Signals 6, 5 and 13 only have a 'section behind' configured (no 'section ahead') .

- All other signals have a 'section behind' and a 'section ahead' for the MAIN route.

Configuring basic automation

For this layout, we want all signals to:

- Reflect the state of the 'signals ahead' so when the signal is 'OFF' the *displayed aspect* will take into account the *displayed aspect* of the signal ahead (e.g. if the signal ahead is displaying DANGER, the signal should display CAUTION rather than PROCEED).

- Automatically change to DANGER as soon as a train passes the signal (and then cycle through the aspects back to PROCEED as the train progresses further down the track).

Firstly, configure each signal with details of the 'signal ahead' This is achieved via the **Interlocking** tab of the signal configuration dialog. Note that Signal 4 supports two routes, so we have to specify the signal ahead for each route:
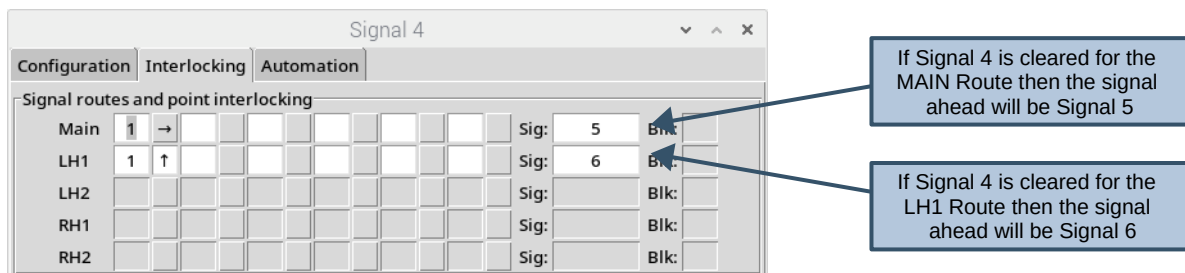
Secondly, configure each signal to be 'Overridden' to 'ON' if the track section ahead of the signal is occupied (so the signal will display DANGER as soon as the train passes the signal and enters the section). At the same time we can also make the signals we don't need to manually control 'fully automatic (without a control button). For this example, the only signals where we need to retain manual control are those signals 'protecting' the junction (Signals 4, 8 and 10). These selections are enabled by checkboxes on the **Automation** tab of the signal configuration dialog.
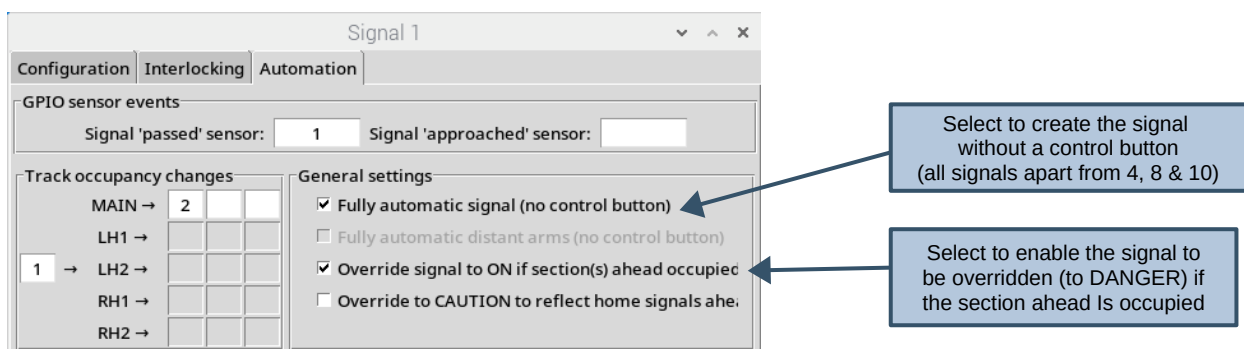
Once all signals have been configured, the overriding of signals by Track Sections can be viewed via the **Automation** tab of the Track Section configuration dialog (read only).

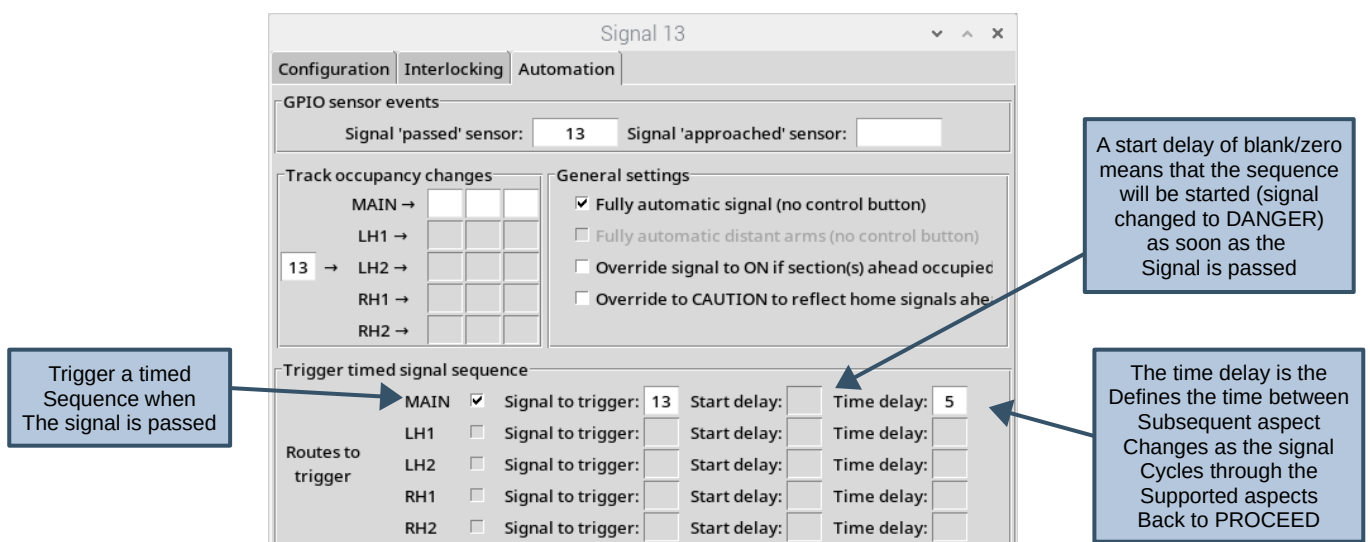Signal 3 (immediately Behind this track section) Will be overridden to 'ON' When section 4 is occupied
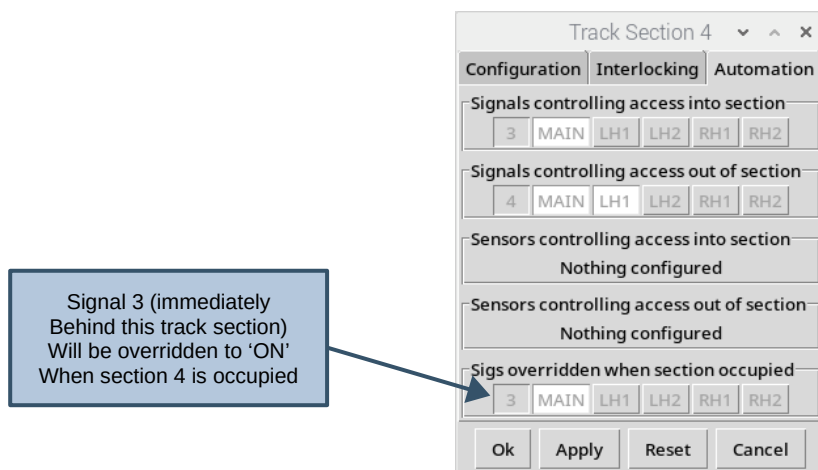
The configuration can then be tested (in RUN mode with Automation ENABLED) by left clicking on each track section in turn to change from CLEAR to OCCUPIED and then back to CLEAR. When the track section is OCCUPIED, the signal behind the track section will display DANGER.

Configuring timed signals

You will note that Signals 5, 6 and 13 do not have any track sections 'ahead of' the signal as these go out to the 'rest of the world' . To add realism, we still want these signals to change to DANGER when passed and then cycle back through the aspects to PROCEED as the train supposedly travels further down the track. This is achieved, by configuring them as 'timed signals' via the **automation** tab of the signal configuration dialog.

Timed sequences can be configured for each route supported by the signal. In this case, Signals 5, 6 and 13 control a single route and so we only need to configure a sequence for the MAIN route.

Trigger a timed Sequence when The signal is passed

A start delay of blank/zero means that the sequence will be started (signal changed to DANGER) as soon as the Signal is passed

The time delay is the Defines the time between Subsequent aspect Changes as the signal Cycles through the Supported aspects Back to PROCEED

To test the timed signal, click on the 'signal passed' button at the base of the signal (in RUN mode with Automation ENABLED). The signal will initially change to DANGER and then cycle through the aspects (CAUTION, PRELIMINARY CAUTION) back to PROCEED.

Configuring approach control

Approach control is normally used when a diverging route has a lower speed restriction. Even though the route ahead may be clear, the signal controlling the diverging route will display a more restrictive aspect (either DANGER or CAUTION) to slow down the train. As the train approaches, the signal will then be 'released' to display its normal aspect (PROCEED).

Note that if you are going to use approach control for your layout, this will require an additional track sensor located on the approach to the signal, to trigger the 'signal approached' event.

The application supports both 'release on red' and 'release on yellow' approach control modes. For 'release on red', the signal will display a DANGER aspect and the signals behind will display the expected aspects (CAUTION, PRELIMINARY CAUTION). For 'release on yellow', the signal will display a CAUTION aspect and the signals behind will display special aspects to provide the driver with pre-warning of the diverging route (FLASHING-CAUTION for the previous signal and FLASHING-PRELIMINARY-CAUTION for the signal behind that).

We'll configure Signal 4 to apply 'release on yellow' approach control for the diverging route, and configure an additional track sensor to trigger the 'signal approached' event. This is achieved via the **automation** tab of the signal configuration dialog.

Signal 4 will now be displayed with a second button positioned on the track (on the approach to the signal). This is the button to simulate the 'signal approached' event to 'release' the signal.

This can be tested (in RUN mode with Automation ON) by setting up the diverging route (Point 1 SWITCHED) and clearing Signal 4. Signals 4, 3 and 2 should display CAUTION, FLASH-CAUTION and FLASH-PRELIMINARY-CAUTION respectively. Clicking on the 'signal approached' button should then 'release' the signal (to PROCEED).

Note that the DCC signalling application does not actually flash the aspects of the signals out on your layout – it just sends the required DCC commands telling the signal/decoder to select the flashing aspect (the signal/decoder actually flashes them). To use approach control on your layout you will therefore need DCC signals or DCC accessory decoders that support the flashing signal aspects - such as the Train Tech DS5HS DCC-enabled 4 aspect signals.

DCC commands /
Command sequences
For flashing signal
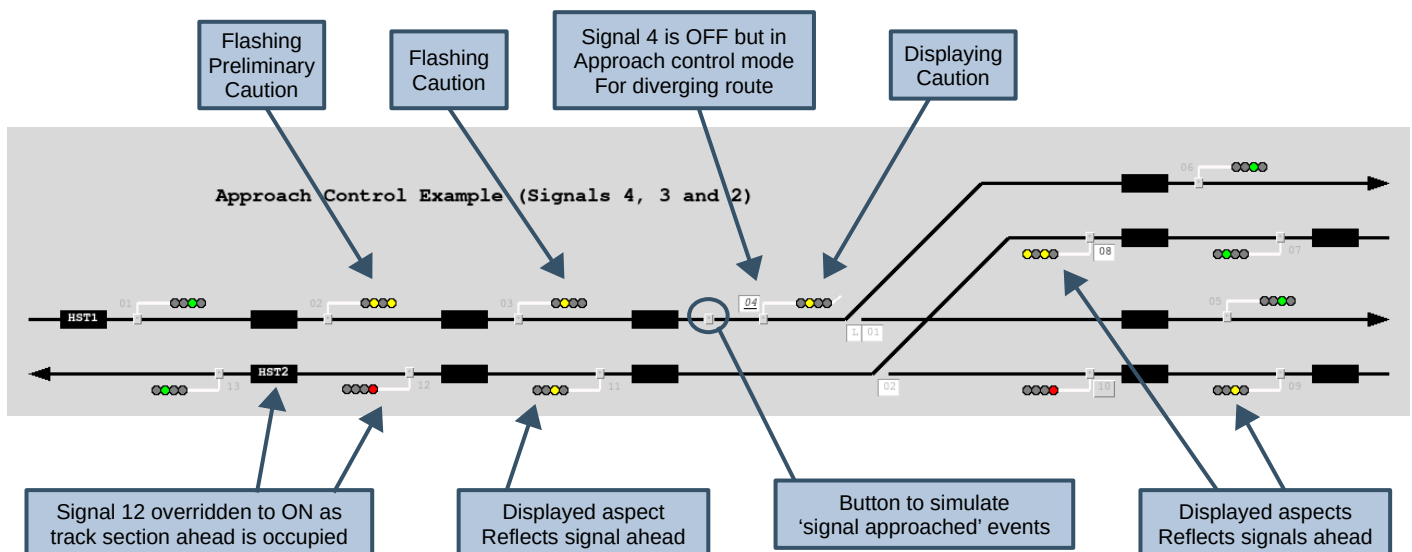aspects

Testing the competed layout
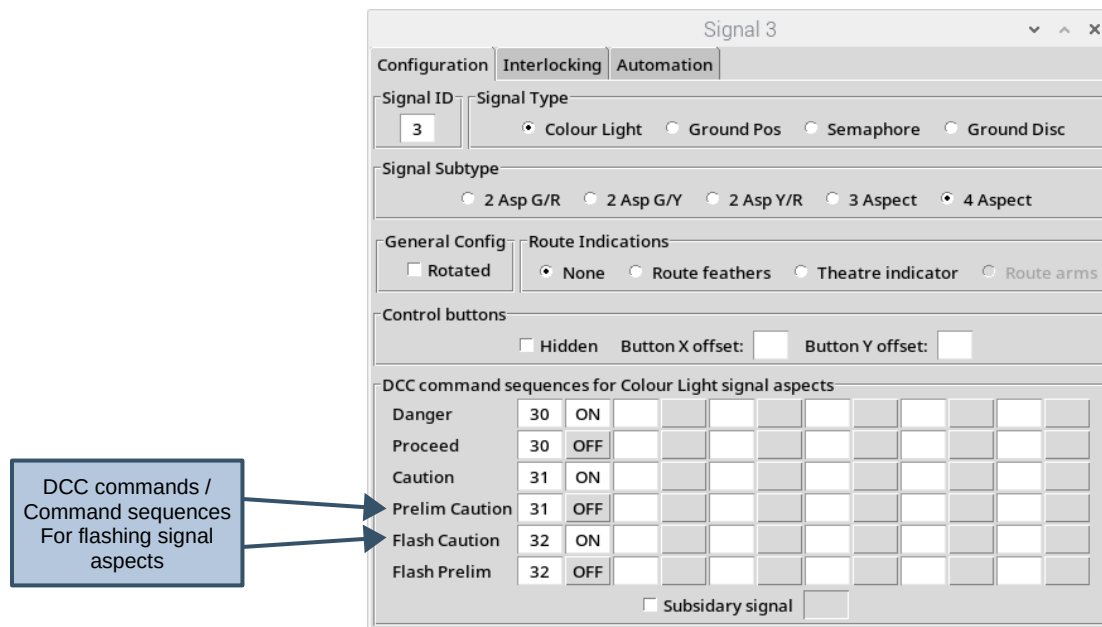
The completed layout can now be tested in much the same way as the first quick-start example layout by feeding trains into the layout (via Track Sections 1, 8 or 9) and then progressing them through the layout from one track section to the next by clicking on the 'signal passed' buttons (and the 'signal approached' button) along the route.

Flashing
Preliminary
Caution

Flashing
Caution

Signal 4 is OFF but in
Approach control mode
For diverging route

Displaying
Caution

Signal 12 overridden to ON as
track section ahead is occupied

Displayed aspect
Reflects signal ahead

Button to simulate
'signal approached' events

Displayed aspects
Reflects signals ahead

Approach Control Example (Signals 4, 3 and 2)

Quick start example 5 – intermediate track sensors

In quick-start example 1 we briefly touched on the use of Track Sensors for the clearing down of routes (once the train was 'clear' of the last signal on the route).

You can also use Track sensors for finer granularity track occupancy between signals and for extending track occupancy into non-signaled areas of your layout such as storage yards:
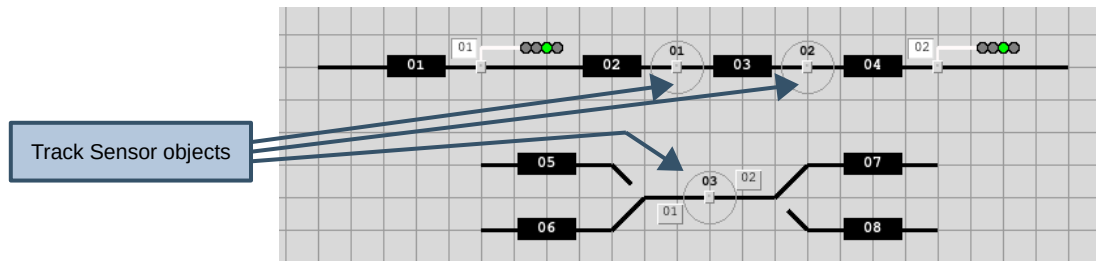
The configuration of Signal 1 is straight forward. The signal ahead for the MAIN route (configured via the Interlocking Tab) is Signal 2, and Signal 1 is configured to be OVERRIDDEN if any of the Track Sections ahead are OCCUPIED (configured via the Automation Tab).

To keep things simple, Signal 2 is configured to clear the Track Section immediately behind the signal when the signal is passed by the train:

The intermediate Track Sensors can then be configured to pass Trains from one track Section to the next when a 'sensor passed' event is triggered (either by clicking on the small button to simulate an event, or mapping Track Sensors to GPIO sensors for real layout events):

GPIO Sensors can be mapped to Track Sensors to trigger 'passed' Events from external train sensors

When a Track Sensor 1 'passed' event is triggered, Trains will be passed From Track Section 2 to Track Section 3

When a Track Sensor 2 'passed' event is triggered, Trains will be passed From Track Section 3 to Track Section 4

Track Sensor 3 is configured to pass trains from one Track Section to another depending on the setting of the points either side. Configuration is similar to that of the Signal Interlocking tab but in this case we can define possible routes both 'before' and 'after' the Track Section:

Possible Movements when triggered:

Point 1 Normal, Point 2 Normal : 6 <--> 7
Point 1 Normal, Point 2 Switched : 6 <--> 8
Point 1 Switched, Point 2 Normal : 5 <--> 7
Point 1 Switched, Point 2 Switched : 5 <--> 8

Quick start example 6 – DCC accessory switches

The signalling application also provides 'DCC switches' that can be added to the schematic and used to operate any other DCC accessory out on your layout such as track isolating sections (if you are still using analogue for train control), level crossings, light/sound effects etc.

Switches can either be configured as LATCHING or MOMENTARY:

- LATCHING switches can be configured to transmit different DCC command sequences (via the DCC accessory bus) when selected and deselected.

- MOMENTARY Switches can be configured to transmit different DCC command sequences when 'pressed' and 'released'. They can also be configured with a delay to 'release' (and transmit the 'released' command sequence), a fixed time after the switch was pressed.

For quick-start example 6 we're going to modify the first example layout to include DCC Switches for track power switching and a horn sound effect.
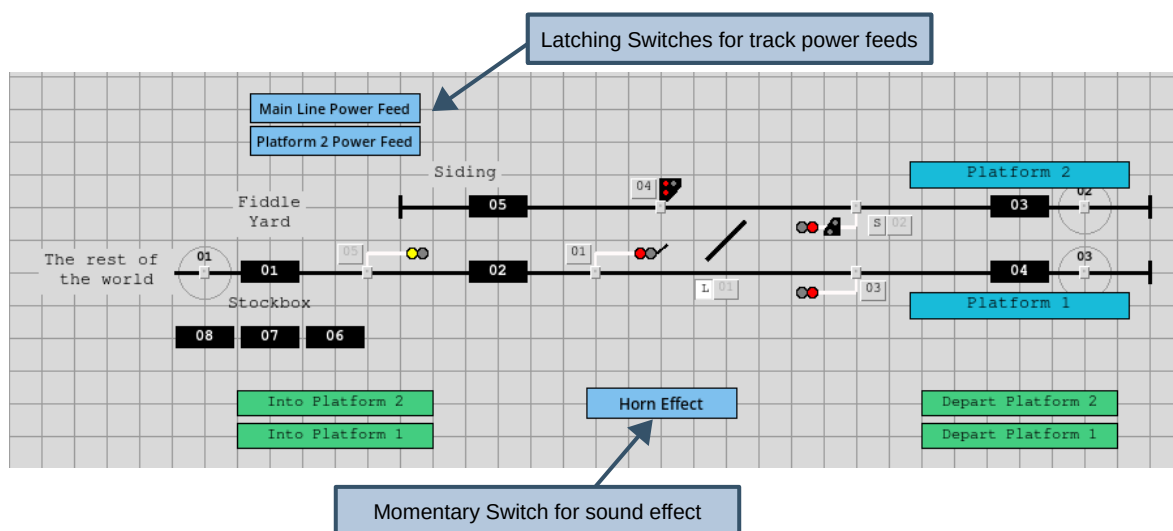
The Switch objects are added to the schematic and positioned in the normal way. However, you should note that these use the same underlying 'button' object as the Schematic Route Buttons and and so the 'one-up' numbering will reflect this.

To configure the Switches, double click on each one in turn to bring up its configuration dialog:
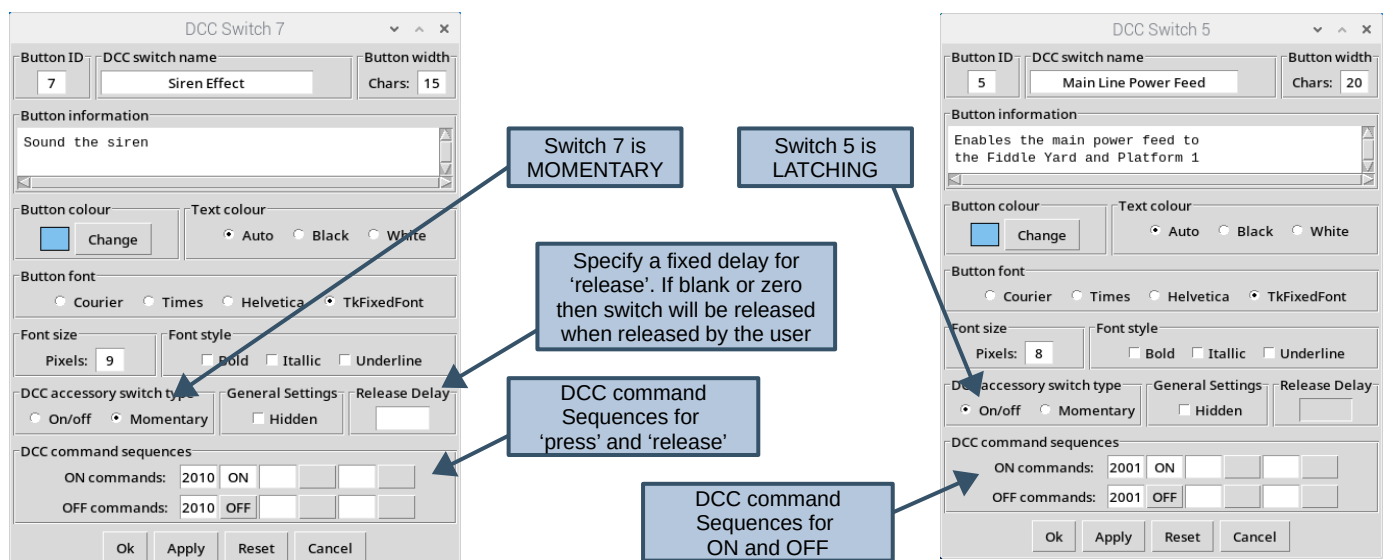
Once the DCC Switches have been configured, they will send out the specified DCC command sequences when operated (in RUN Mode, with SPROG connected and DCC Power ON).

DCC Switches can also be specified as part of a Route configuration. In this example, we have configured each Route to apply power to the required track feeds on route setup, and switch off the track feeds on route clear down:

Route 1 will set Switch 5 To ON during route set-up and revert to OFF during Route clear-down

Route 3 will set Switches 5 And 6 to ON during route Set-up and revert them to OFF during Route clear-down

Route 1 has enabled the main line Power feed as part of the route set-up
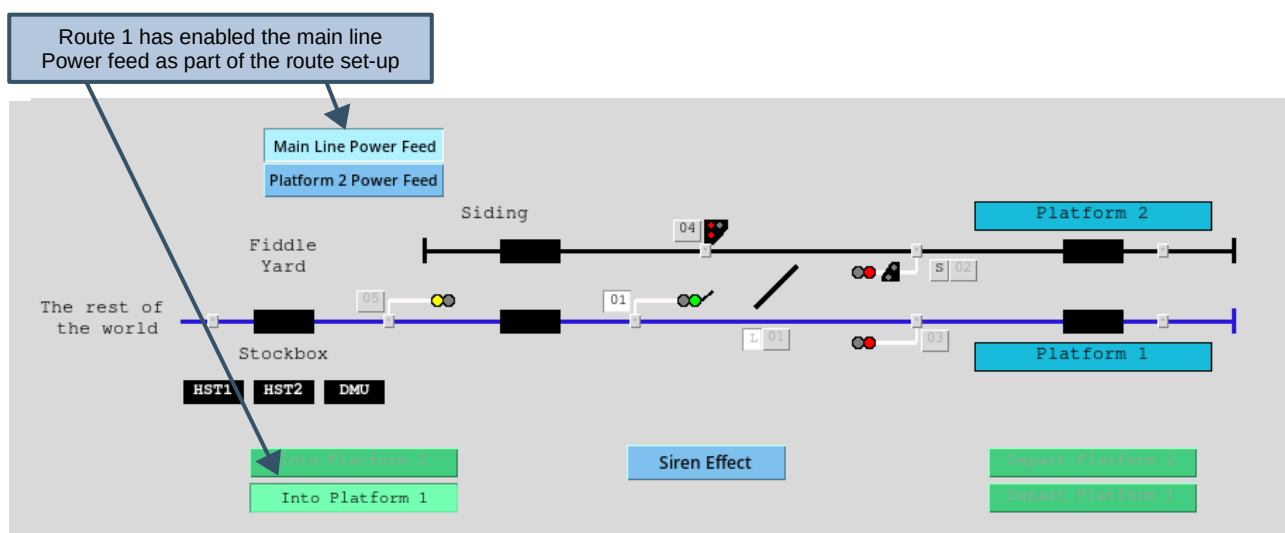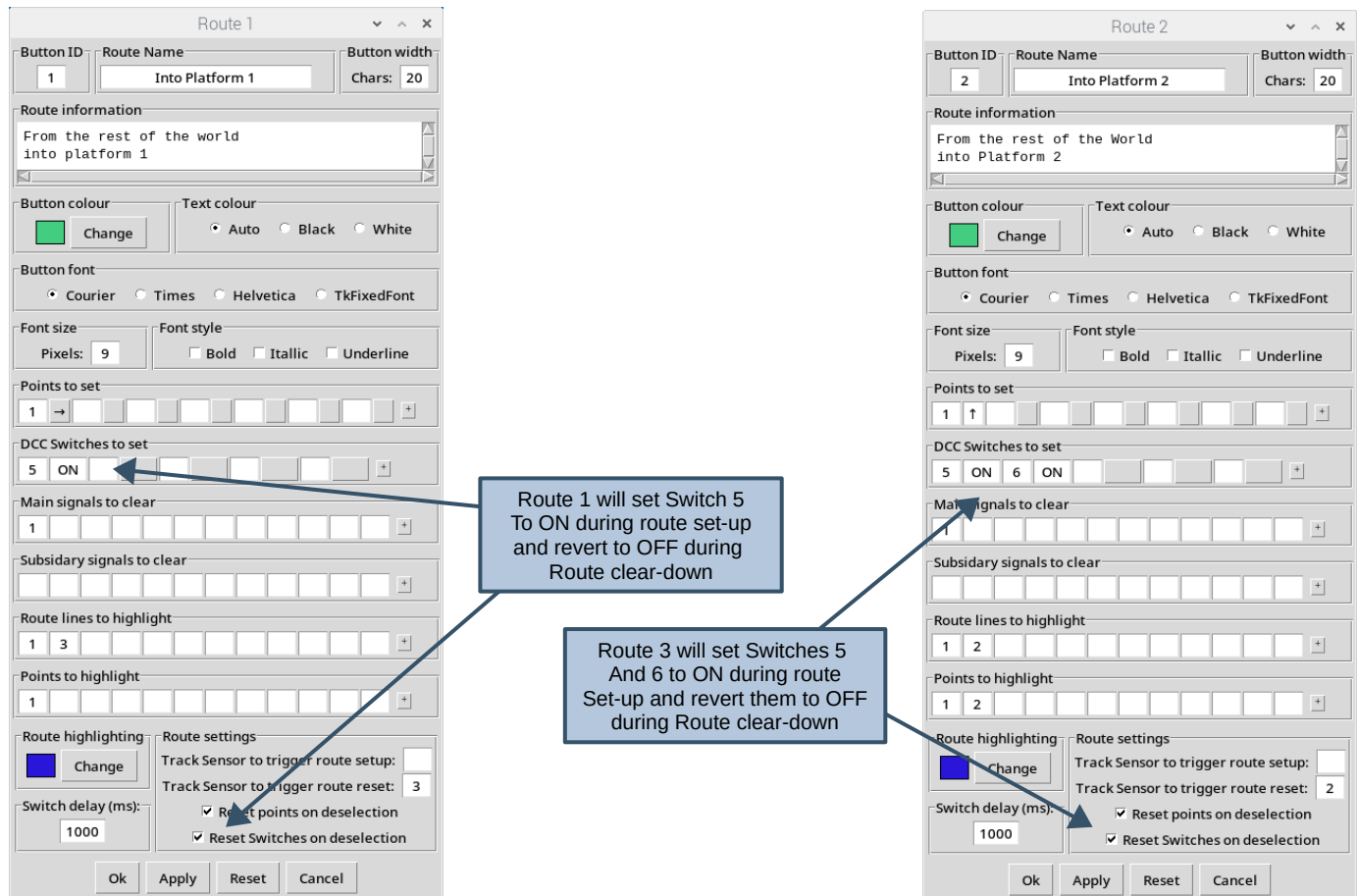
Still to discover

There are still several features of the application that have not been covered in this quick-start guide, but once you are familiar with the features above, you should be able to experiment and figure them out for yourself:

- Theater Route Indications – supported by main semaphore and main colour light signals (use instead of feathers or route arms). Provides the ability to display a single 'character' for the selected signal route.

- Block Instruments – intended for layouts split into separate 'block sections' . This is a particularly useful feature when using multiple application instances networked together, where each instance represents a different signal box, as the block instruments can be used to communicate (via bell codes) between the signal boxes and control the movement of trains between the two block sections. A basic example including Block Instruments is included in the application networking guide (which can be downloaded from the website).

- Automation - Approach control 'release on red (signals ahead)' for automation of Home signals in a block section. In this case, signals are overridden to 'ON' if any Home signals ahead are still at DANGER and only 'released' to 'OFF' as the train approaches them (reverting to 'ON' as soon as the train has passed).

- Automation - Override to caution to reflect home signals ahead – For distant signals (semaphore or colour light) – Will override the distant signal to display CAUTION if any home signals ahead (within the block section) are at DANGER.

- Interlocking - Interlock on home signals ahead – For distant signals (semaphore or colour light) - To interlock the distant signal (prevent it being cleared) unless all home signals ahead (within the block section) are also clear.

- MQTT Networking – Publishing and subscribing to Signals, Track Sections, Track sensors and Block Instruments. The 'remote' items can then be used within the signalling scheme to provide seamless integration of different signalling areas. This is the subject of a separate networking guide (which can be downloaded from the website).

- The DCC Mappings utility (select **Utilities** and then **DCC Mappings** from the main menubar) – Allows you to view all DCC addresses used in your layout configuration an what signals and points they are associated with.

- The Bulk Renumbering utility (select **Utilities** and then **Item Renumbering** from the main menubar) – Allows you to re-number all schematic objects to your requirements (e.g. for aligning Signal/Point IDs with their signal box Lever numbers.

- Changing schematic object styles (select **Styles** and then the **required menu)** – Allows you to change the styles of ALL or SELECTED objects to "prettify" your schematic as required.

Appendix 1 – How track occupancy works

As the Signalling application primarily uses momentary 'passed' events to detect train movements, it has to make certain assumptions about train movements based on the 'route' that has been set, the state of the signals and the state of the track Sections ahead and behind the signal.

The basic concepts are:

- Normal operation - Trains will be passed from the OCCUPIED to the CLEAR section
- If both sections are OCCUPIED or both sections are CLEAR then no train will be passed
- If there is 'no route' either ahead of or behind the signal then no trains will be passed

The following diagram explains what will happen for 'signal passed' events under both 'normal operation', where the driver is respecting the state of the signals, and some of the 'edge cases' that may occur if the driver inadvertently passes a signal at DANGER:
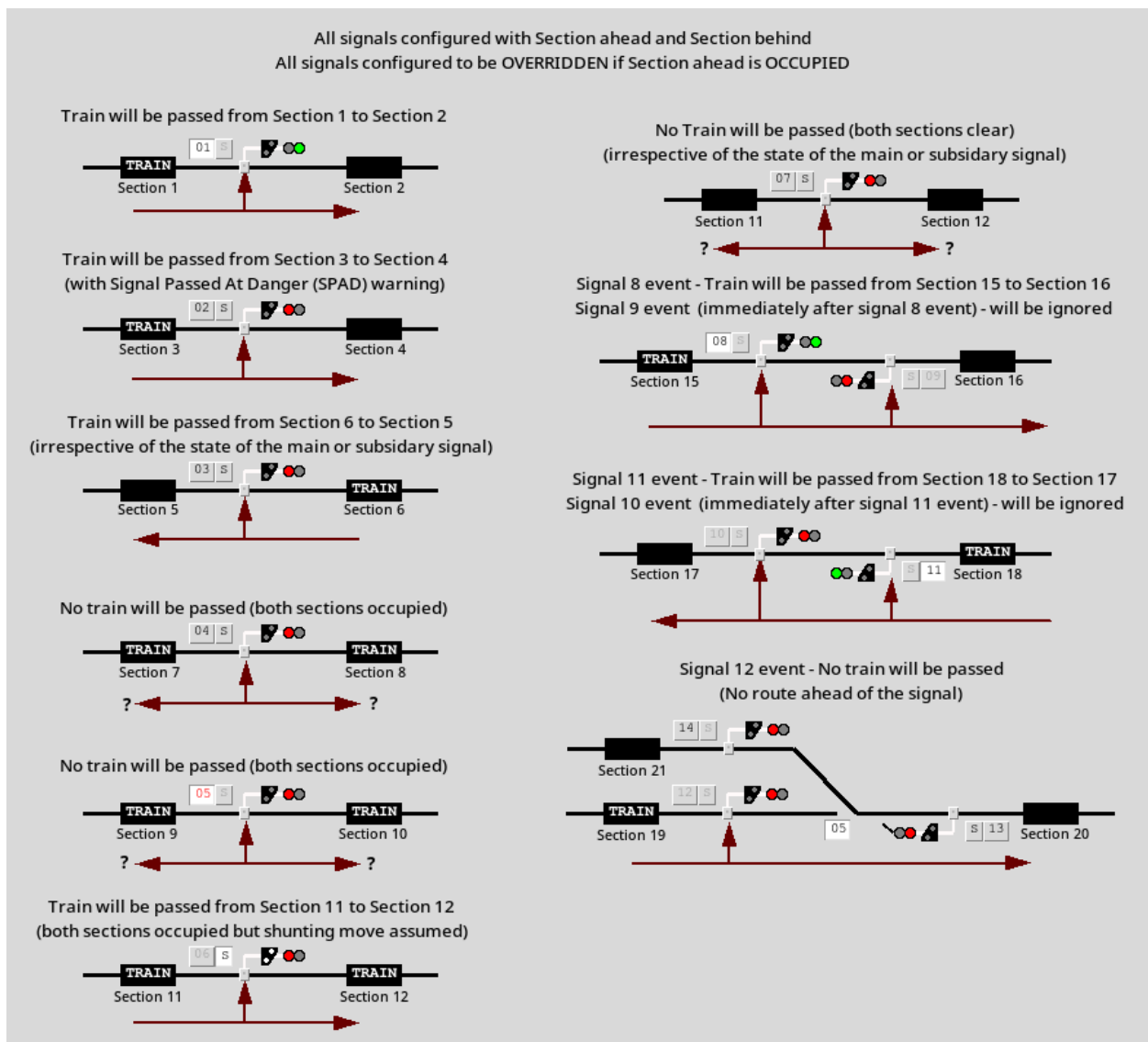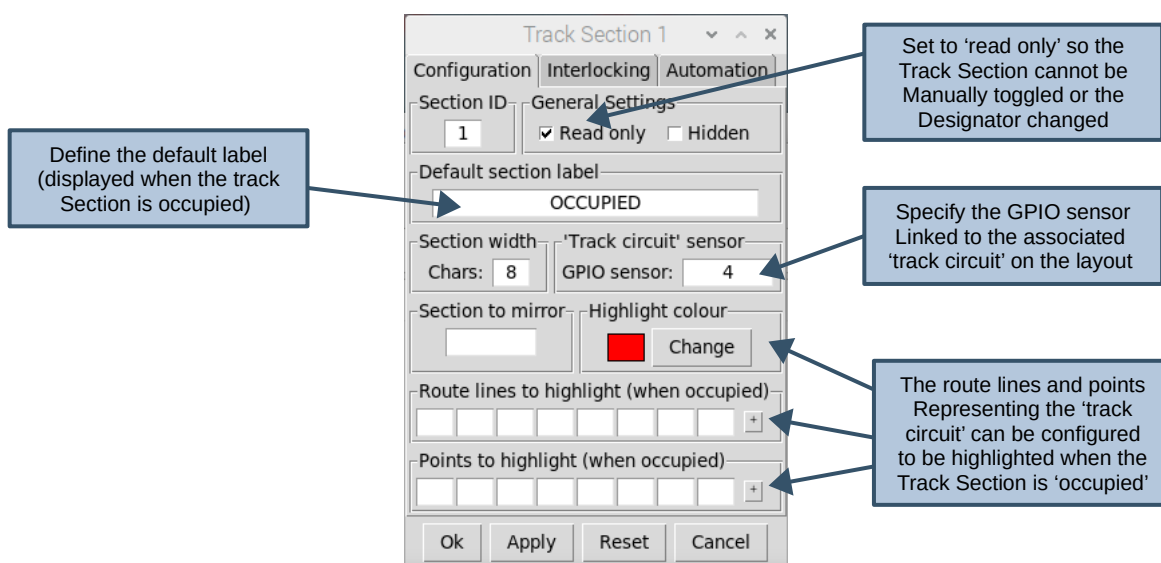

The configuration and operation of Track Sensors is identical, but with the added ability to specify multiple routes (and hence multiple Track Sections) both 'ahead of' and 'behind' the Track Sensor.

Appendix 2 – Using 'track circuit' / 'block' sensors

The DCC Signalling system is primarily designed to use 'event-based' track sensors to drive track occupancy and 'pass' train designators around the layout.

As an alternative, the system can also be configured to use 'track circuit' / 'block' type sensors (e.g. current sensing or optical sensors) to provide an absolute indication of occupancy. In this case the Track Section (on the schematic) will always represent the state of the external sensor (either 'occupied' or 'clear'), but then will not be able to track train designators around the layout.

Double click on a Track Section (in Edit Mode) to bring up the configuration window and then specify the GPIO sensor linked to the 'track circuit'. The Track Section should also be set to 'read only' to ensure the state always reflects the state of the external 'track circuit'.

Define the default label (displayed when the track Section is occupied)

Set to 'read only' so the Track Section cannot be Manually toggled or the Designator changed

Specify the GPIO sensor Linked to the associated 'track circuit' on the layout

The route lines and points Representing the 'track circuit' can be configured to be highlighted when the Track Section is 'occupied'

Note that the highlighting of route lines and points can also be configured for Track Sections driven by 'event based' GPIO sensors if required.

Appendix 3 – The DCC programming utility

A basic DCC programming utility is provided to enable 'one touch' programming (suitable for the majority of DCC point & signal decoders on the market) and Configuration Variable (CV) programming (suitable for more complex decoders such as the Harman Signallist SC1).

If 'One touch' programming is supported by the device, this is always the preferred method as this can be done 'on layout' without disconnecting all other devices from the DCC accessory bus.

The utility can be opened by selecting **Utilities = > DCC Programming** from the Main Menubar. Note that The Pi-SPROG needs to be CONNECTED with DCC Power ON to program devices (refer to the 'Operating your layout' section for further information).

One-touch DCC programming

Ensure the device it is connected to the DCC bus and has been put into 'one touch' Programming mode (refer to the device documentation for specific instructions).

Enter the required DCC address and click the required command (On or Off) to program. Once programmed, the device should respond to all subsequent DCC commands.
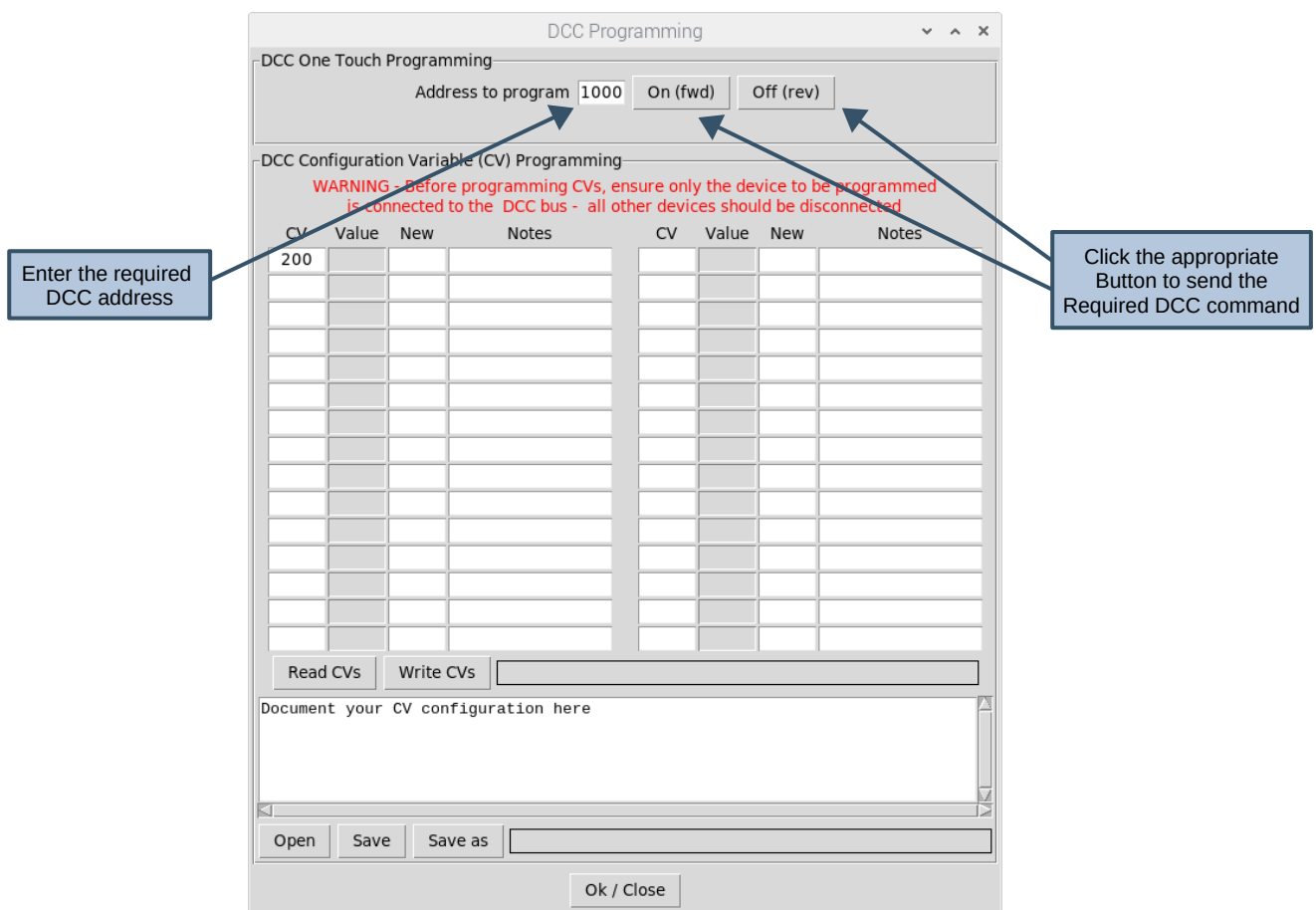
DCC CV programming

Warning – before using the CV programming utility the DCC accessory bus should only be connected to the device you want to program (all other devices should be disconnected).

Enter the addresses of the CVs you want to inspect / program and click **Read CVs** to retrieve the current values. New values can then be entered and programmed by clicking on **Write Cvs**.

A partial configuration (for the Harman Signallist SC1 decoder) is shown below:
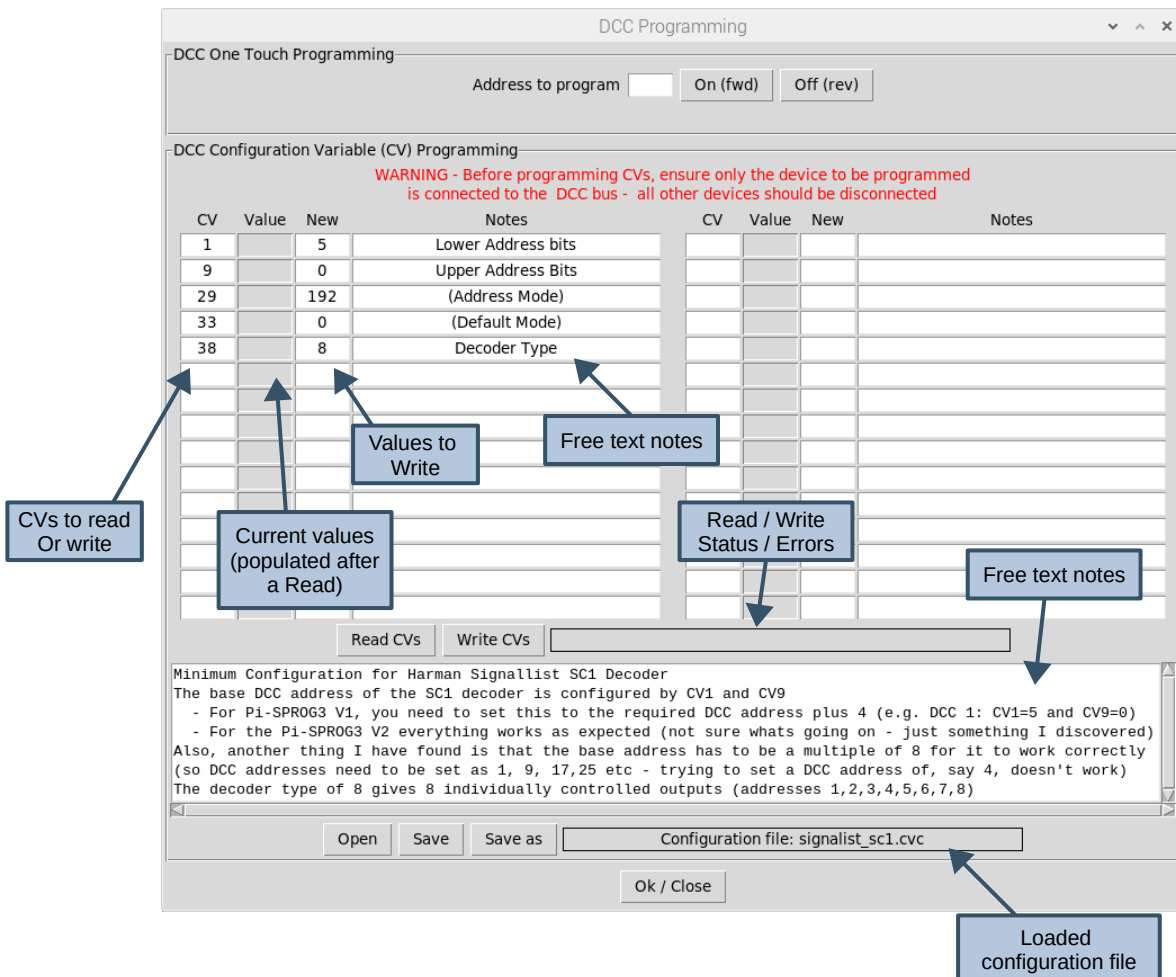
During a 'read' operation, either the retrieved values will be displayed or "**---**" signifying a particular CV could not be read.

During a 'write' operation, the displayed values will either turn Green (if the write was successful) or Red (if the write operation failed).