# Model Railway Signalling

## Quick-start Guide

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

However, as always, Rule 1 of Railway Modeling applies – it's your layout and its therefore entirely up to you how you use the application to signal your layout. Hopefully the features provided by the application will enable you to achieve the level of realism you want to achieve.

The example layouts used in this guide are all available in the 'user_guide' folder within the Github repository (https://github.com/johnrm174/model-railway-signalling).

## Table of Contents


Introduction....2
The importance of research....3
The Schematic Editor....4
Drawing your layout schematic....5
Planning your signalling scheme....6
Adding signals to the layout schematic....7
Configuring the basic interlocking....9
- Signal 1....9
- Signal 2....10
- Signal 3....10
- Signal 4....11
- Signal 5....11

Testing the basic interlocking....12
Operating your Layout....14
Adding and configuring track occupancy....15
Testing track occupancy changes....17
Configuring external track sensors....19
Interlocking with Track Sections....20
Saving and loading your layout....22
An automation example....23
- Configure interlocking....23
- Configure track occupancy....24
- Configure basic automation....24
- Configure timed signals....25
- Configure approach control....26
- Testing the competed layout....27

Still to discover....28
Appendix 1 - Using semaphore signals....29
Appendix 2 - DCC programming of signals and points....30

# Introduction

This 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:

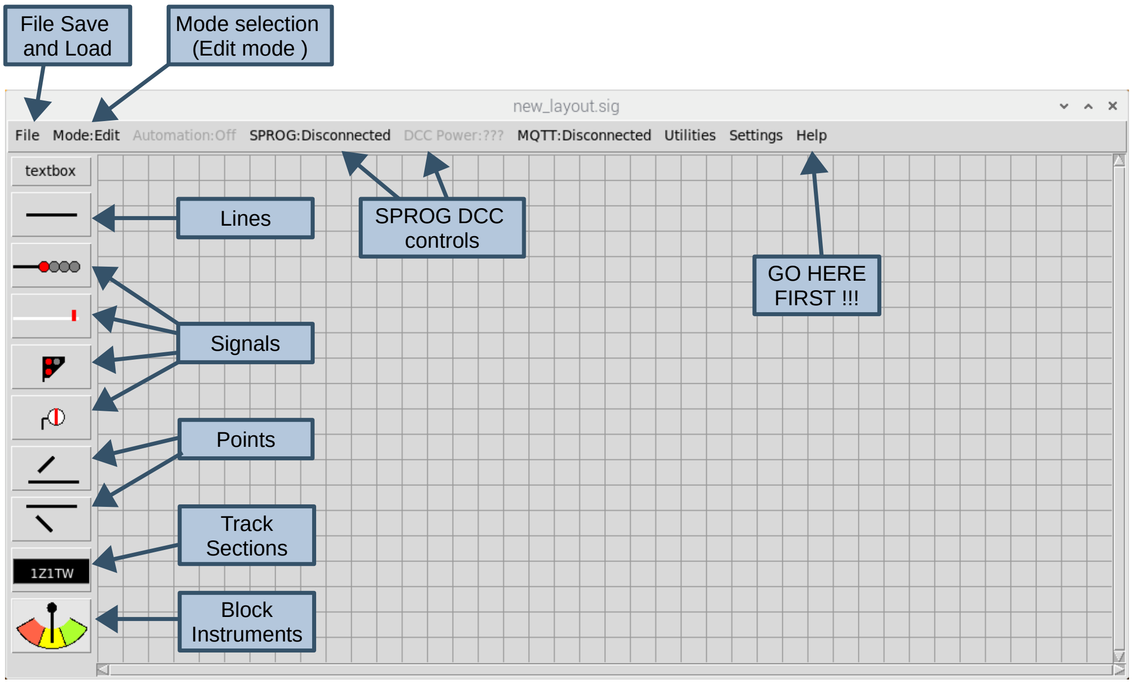
- https://signalbox.org/ - Comprehensive information on signal types and the 'Block System' and a vast library of signal box diagrams for you to draw inspiration.
- https://en.wikipedia.org/wiki/UK_railway_signalling – Its Wikipedia (enough said).
- http://www.railway-technical.com/signalling/ - A section 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'.

# The Schematic Editor

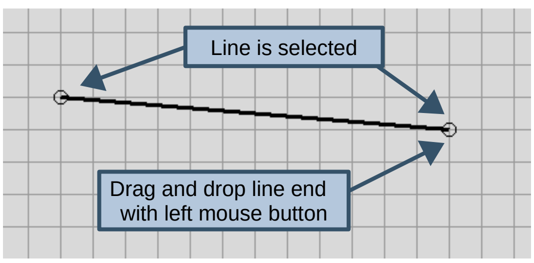
The application opens in ‘Edit’ mode with a blank drawing canvas. The panel on the left contains the buttons to add schematic drawing objects to the canvas, whilst the Menubar across the top of the window contains various controls and options for configuring the application.

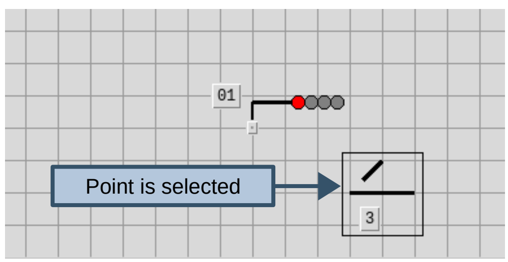
For the time being we’ll focus on the basic controls and options needed to get you up and running:




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

- Add objects to the canvas by **left-clicking** the buttons on the left hand side.
- 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)
- Add/Remove objects to/from the current selection by **shift-left-clicking** on them
- 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

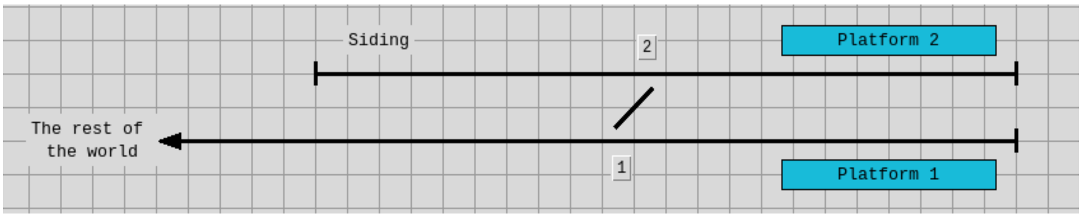



Other Schematic Editor functions summarised in the Help window, but once you are comfortable with the basics as described above, we’ll continue with the first quick-start example by clearing down the schematic to start afresh (from the Main Menubar select **File** => **New**).

# Drawing your layout schematic

For the first exercise, 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):

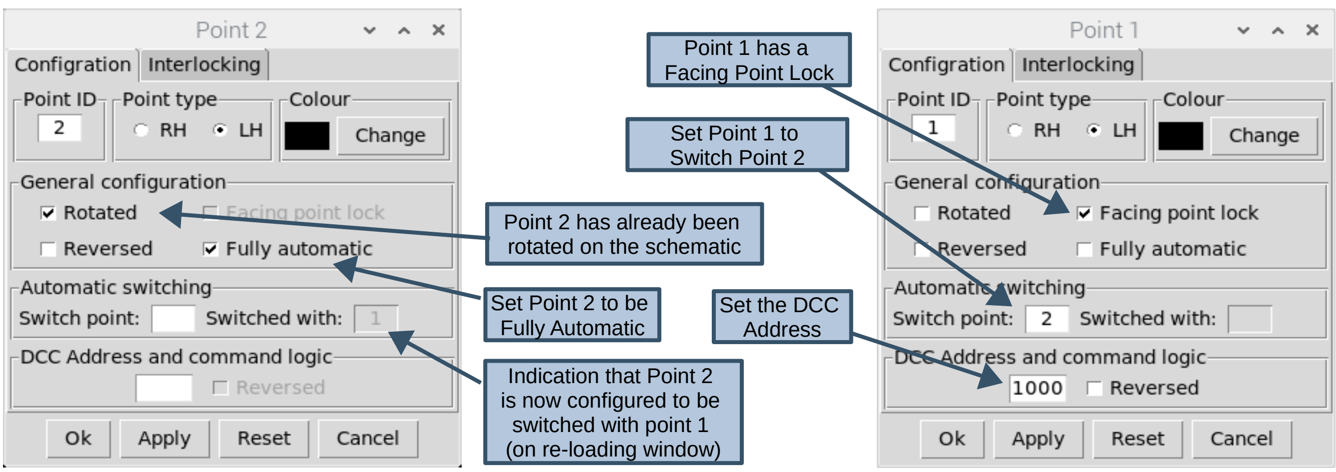



Point 1 (left hand point) can be added straight to the layout and moved into the appropriate position. To create point 2, either add another left hand point to the layout or, alternatively, select point 1 (**Left Click** on the point) and then copy/paste (**Cntl-c** / **Cntl-v**). When items are created, they are assigned the next available 'one-up' identifier so in this case the added / pasted point will be created as Point 2. Whilst it 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 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.

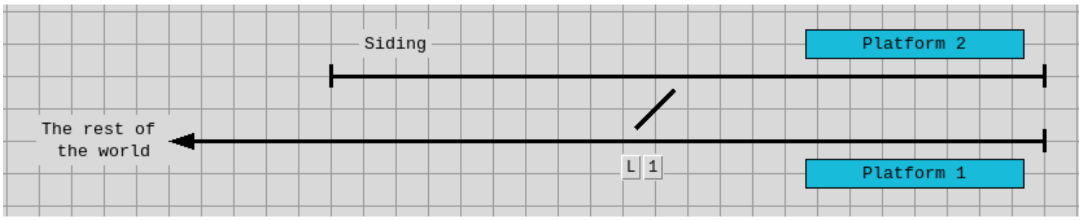
Text boxes can be added to annotate the schematic. Once created, double-left click on the 'Text' to bring up the configuration dialog and edit the contents. Note that textboxes will always be sized to fit the contents on the schematic, but padding (extra lines or spaces) can be used if required. In this 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[1]. Firstly, **double-left-click** on point 2 to bring up the configuration dialog, set it to be fully automatic and then **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) before **Applying** the changes.




1 Facing Point locks (FPLs) physically lock points in position to prevent them accidentally changing as a train passes over them. In the UK it is not permitted for passenger trains to pass over points from a facing direction (diverging direction) without them being locked into place.

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 is ‘greyed out’ and unresponsive (the point is locked). To switch the point, first release the FPL to enable the point control, 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 or Platform 2
- Platform 1 => Rest of the world
- Platform 2 => Rest of the world

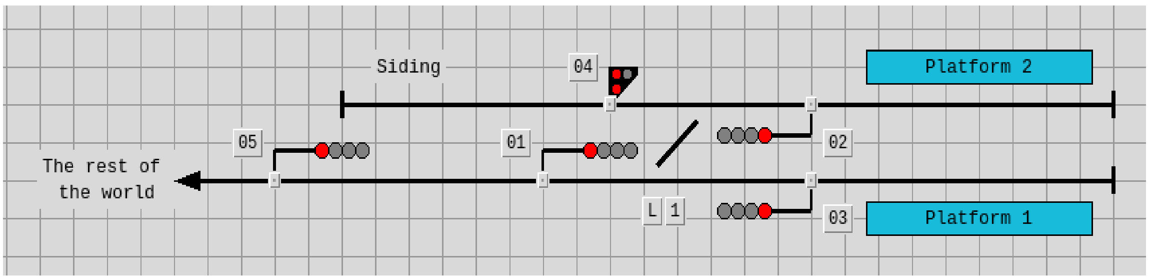
We also want to signal the following shunting movements:

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

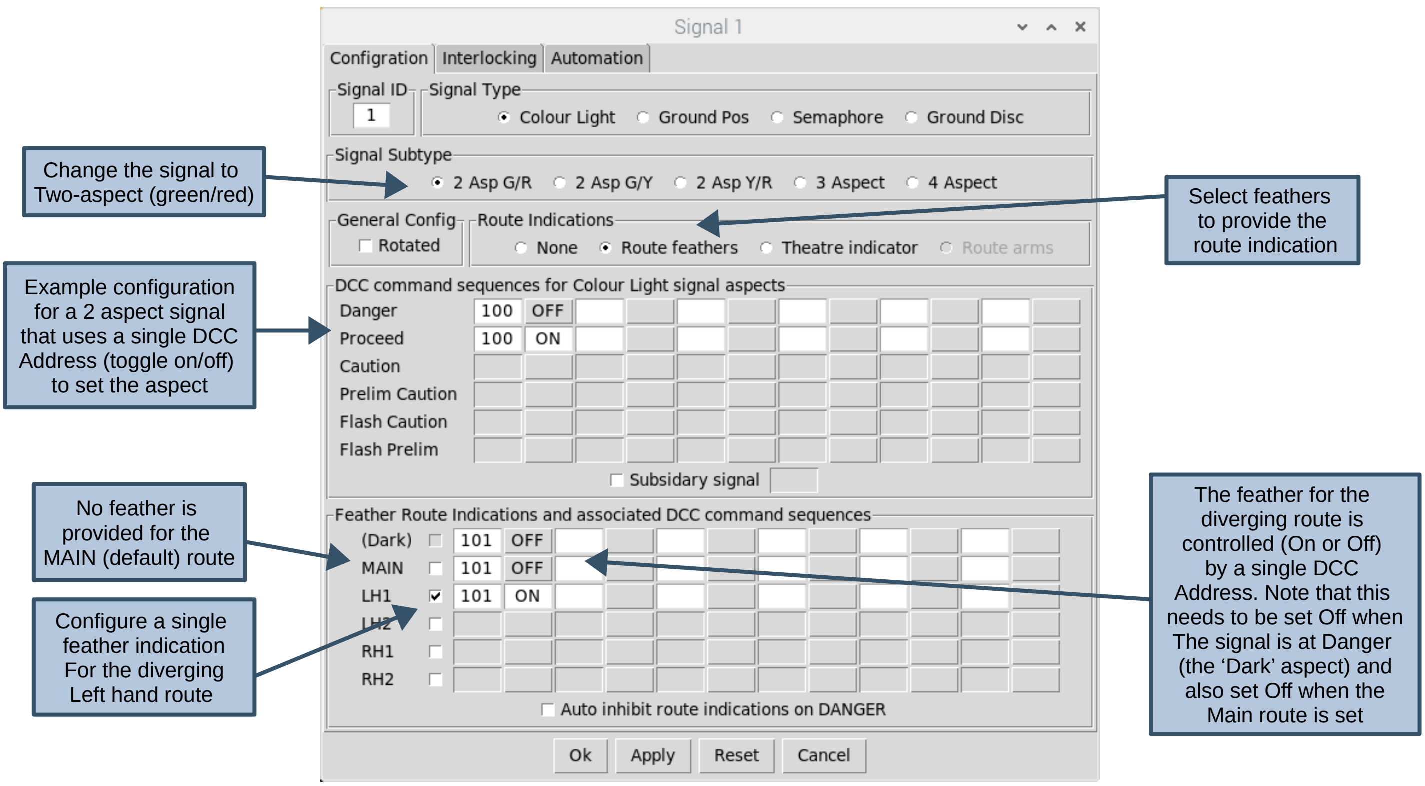
This (together with the knowledge gleaned from our research in the previous section) allows us to define the signals we want to add to the layout and where they should be positioned.

# Adding signals to the layout 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). To keep things simple, we'll use two aspect colour light signals for the main signals. 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.

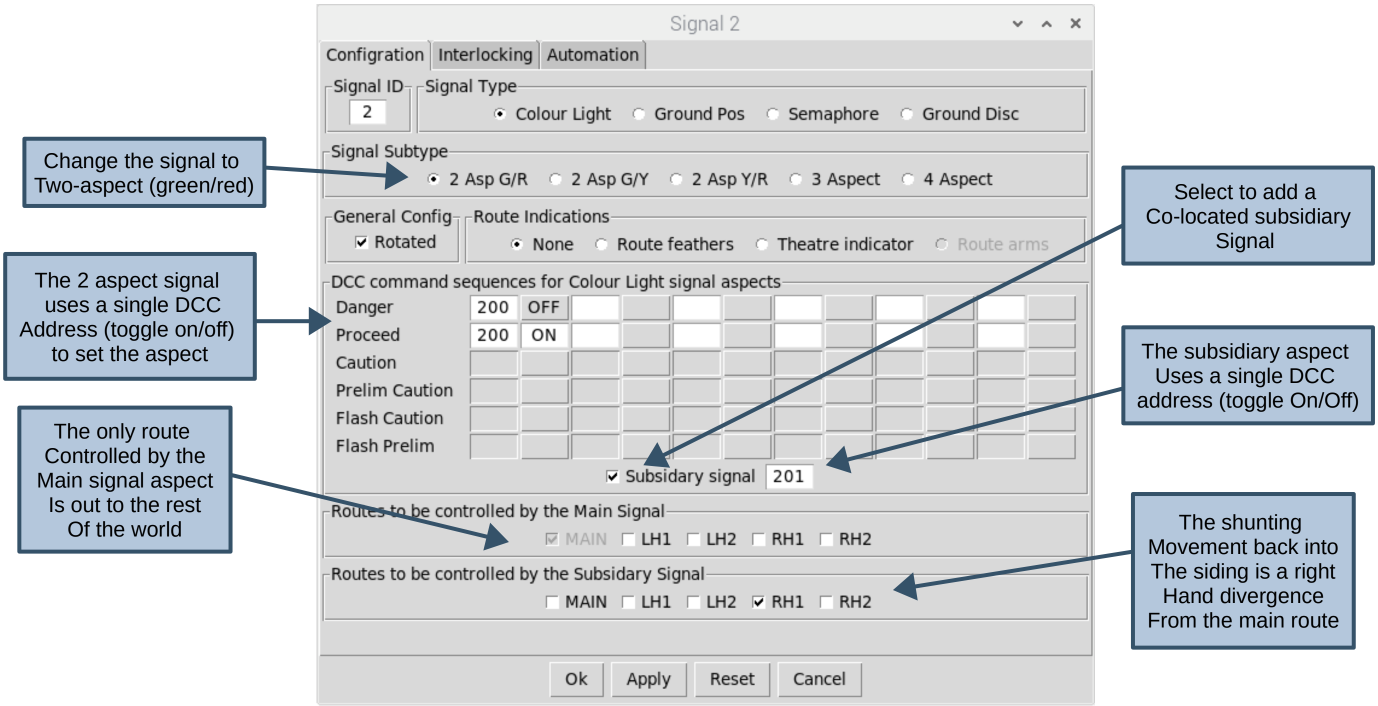



To signal the train movements we have defined, we need to edit Signal 1 to provide a route indication (to differentiate between the routes into Platform 1 and Platform2). At the same time, we'll change it to a two-aspect signal and configure the DCC addresses. As before, **double-left-click** to bring 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).

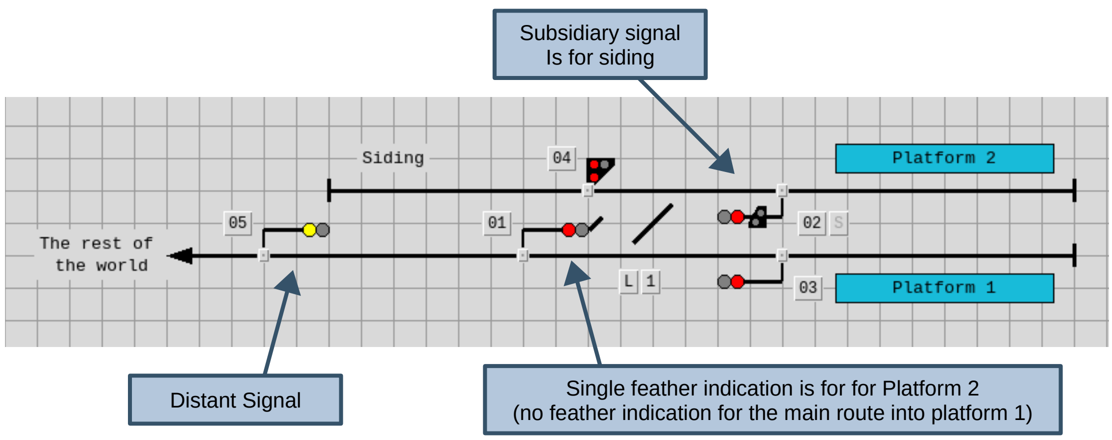
Note that we don't need route indications for Signal 2 as the only route controlled by the main aspect is the departure out to the rest of the world (the MAIN route for this signal). The subsidiary aspect controls the shunting move back into the siding, which is effectively a right hand divergence from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




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 configure the interlocking later in this example).

Finally, configure the DCC addresses for signal 4 (the ground position signal).

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



# Configuring the 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.

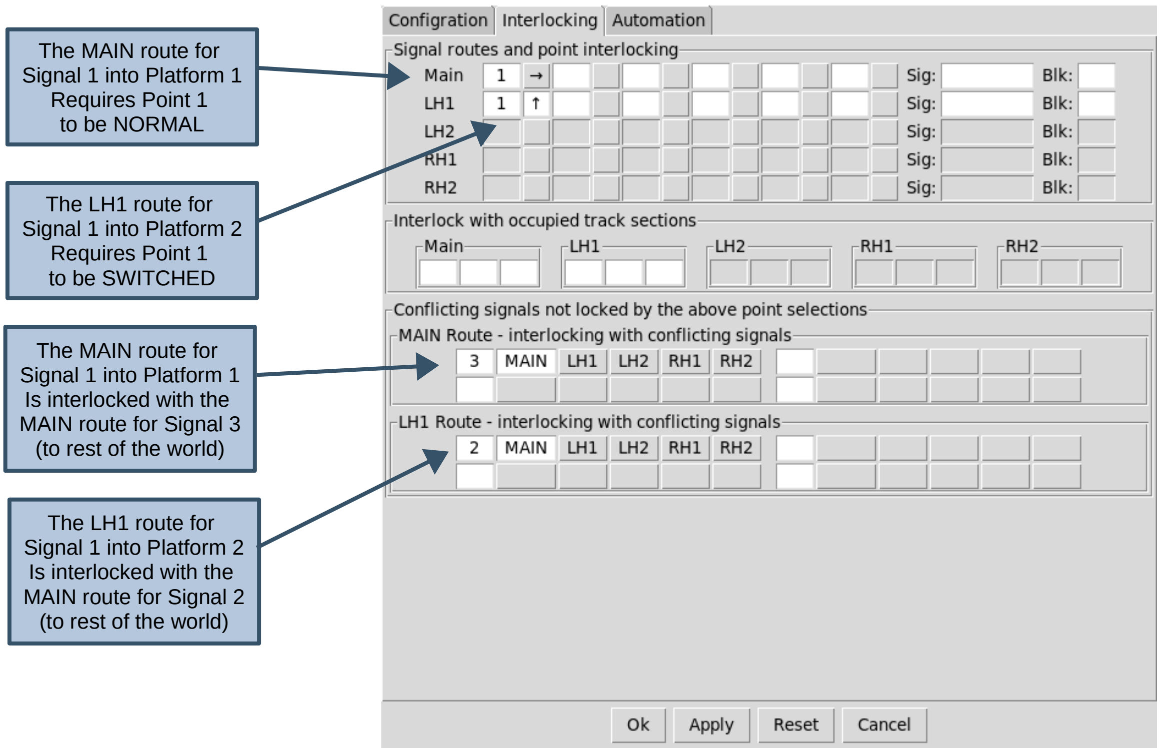
All interlocking is defined via the configuration dialog of the appropriate signals. **Double-left-click** on a signal to bring up the configuration dialog and select the **Interlocking** tab.

## 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.

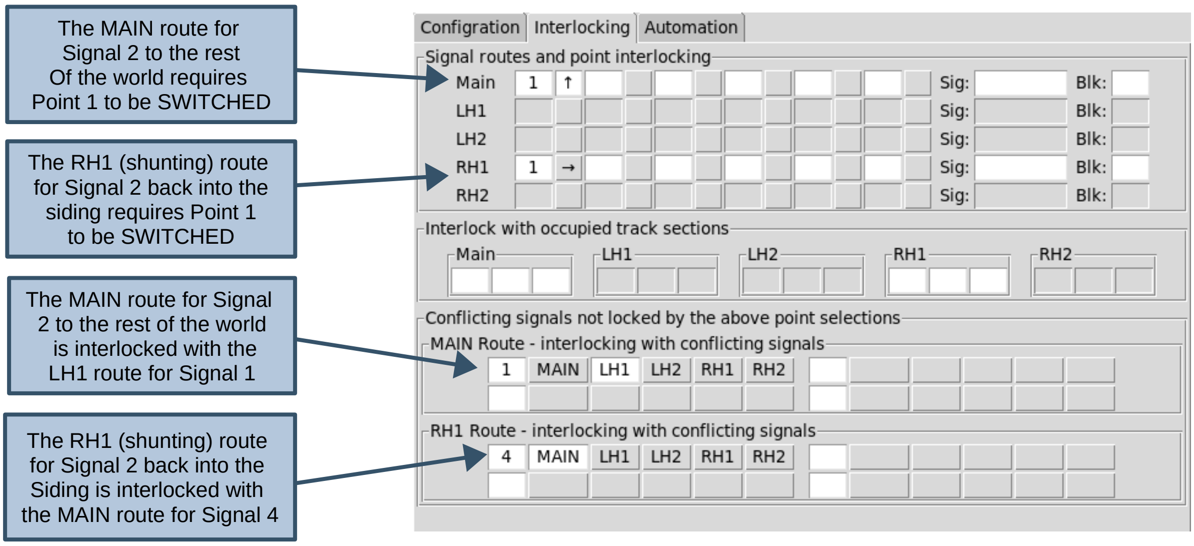


## Signal 2

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.

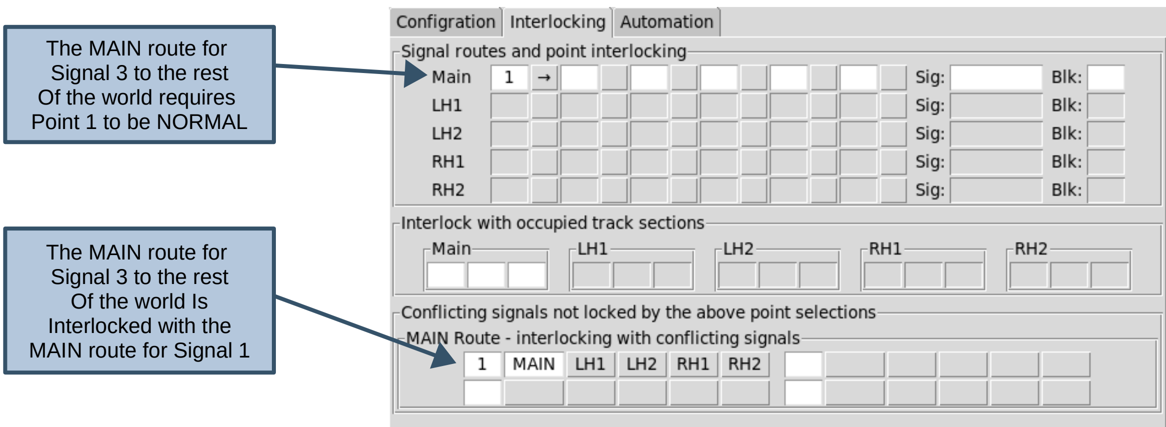
When Signal 2 was initially configured (see earlier), the subsidiary aspect was configured to allow a shunting move is back into the siding (which is a right-hand diverging route). Point 1 therefore needs to be NORMAL to enable the subsidiary signal to be cleared for the RH1 route.

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.



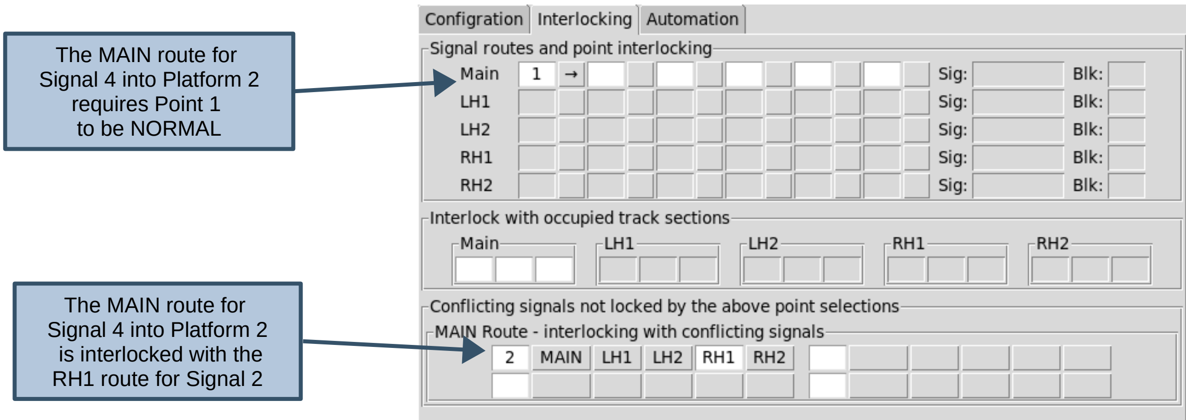

## Signal 3

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 (which needs to be SWITCHED to allow Signal 3 to be cleared). The signal also needs to be interlocked with the MAIN route of Signal 1.



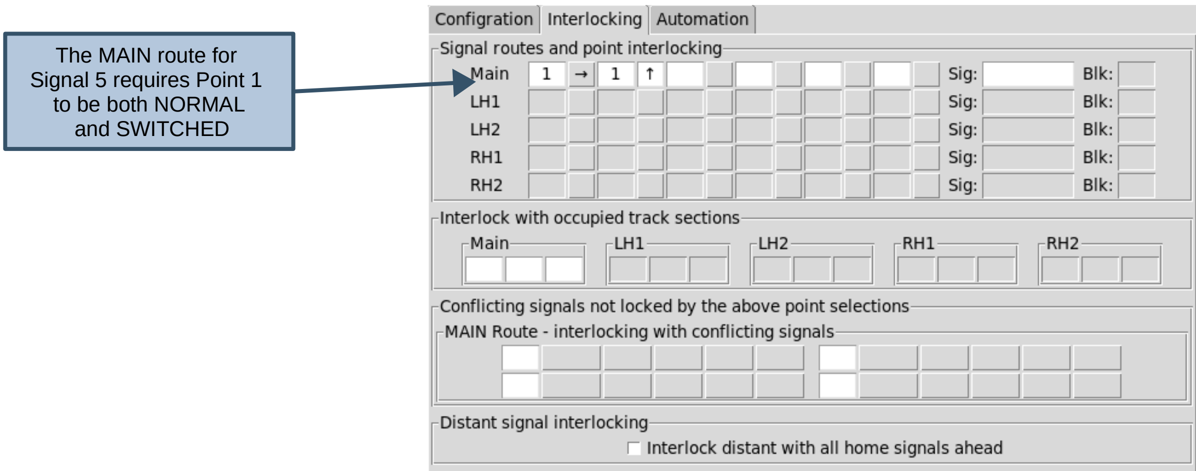
# Signal 4

Signal 4 controls a single shunting route (from the siding to Platform 2) which needs to be interlocked with Point 1 (which 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).




# 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).

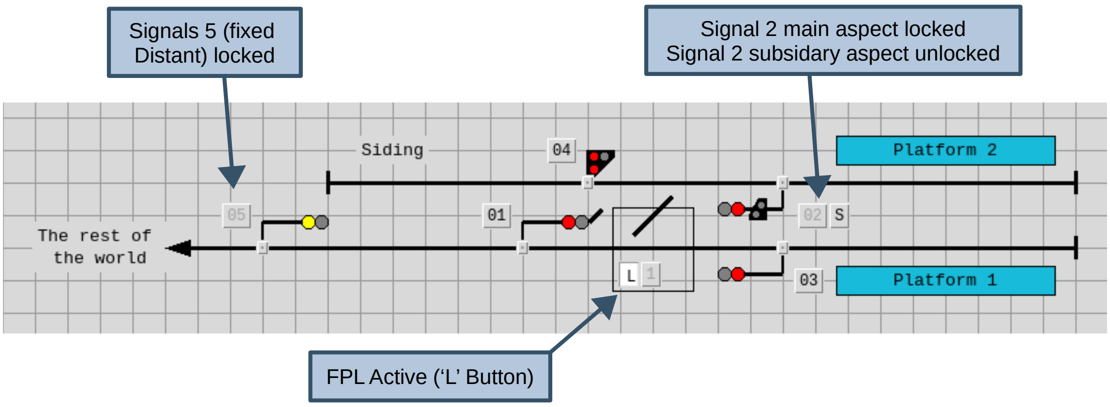


# Testing the basic interlocking

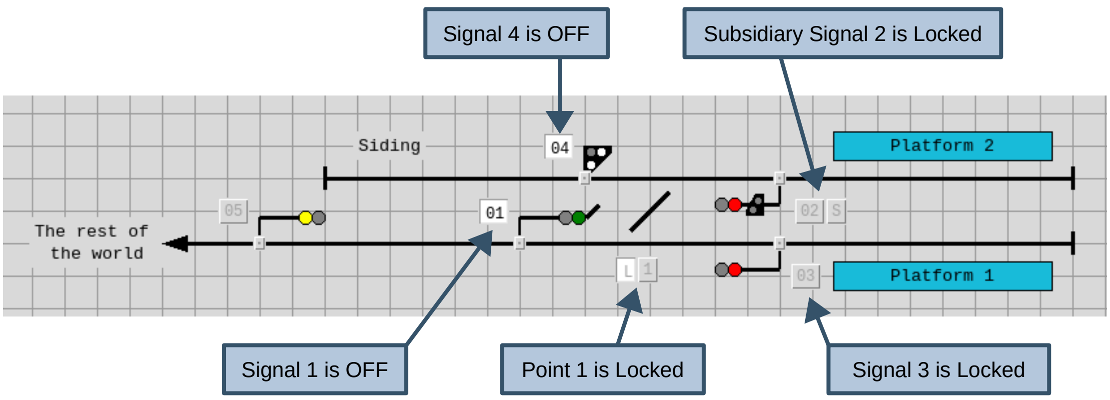
Once configured, the basic interlocking can be tested. If a signal or point is ‘locked’ then the associated control buttons will be ‘greyed out’ and unresponsive.

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[2]’ 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.

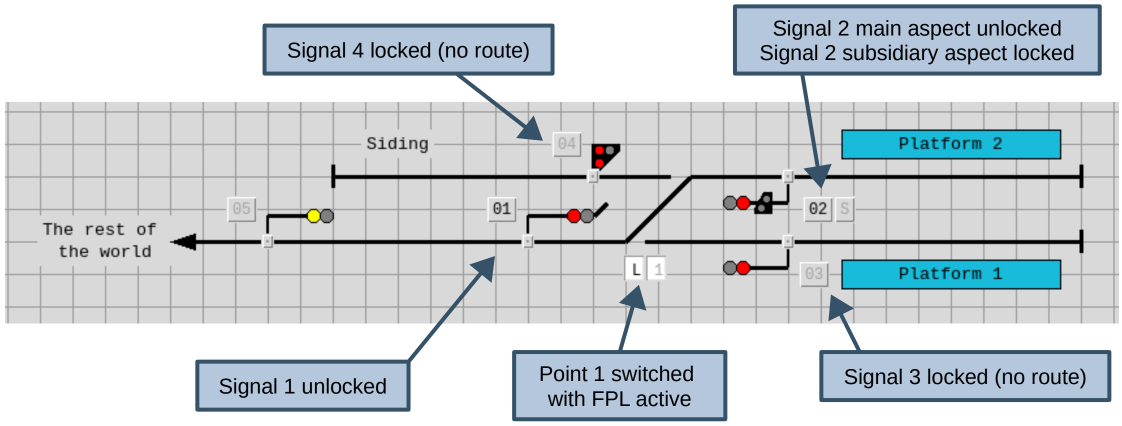



In this state, the interlocking with conflicting signals can be tested. If Signal 4 is switched ‘OFF’ then the subsidiary aspect of Signal 2 will be locked, and vice versa. Similarly, if signal 1 is switched ‘OFF’ then Signal 3 will be locked, and vice versa. Note that as soon as a route is cleared the points are locked and cannot be changed until the signals are returned to ‘ON’

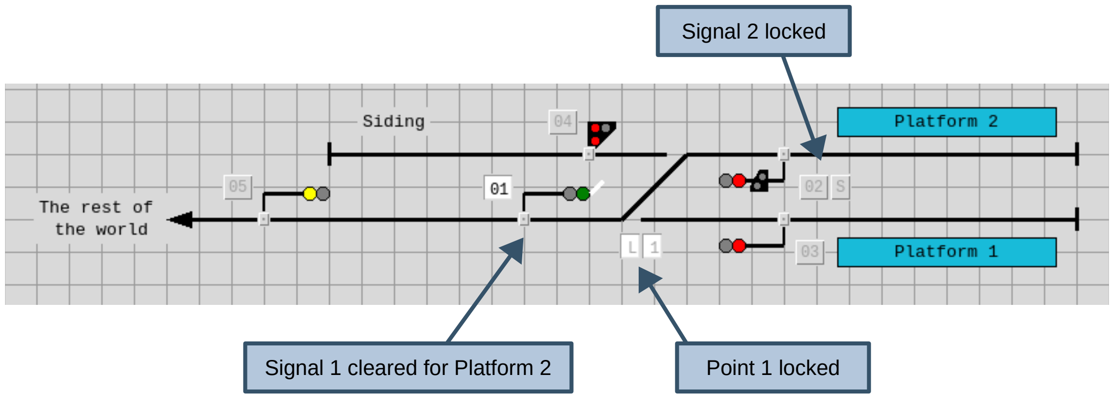



Once all signals have been returned to ‘ON’ then the points can be changed to test the other signal routes. Click on the ‘**L**’ button to ‘release’ the Facing Point Lock (FPL) and enable the main point control button. The point can then be switched . Note that as soon as the FPL is ‘released’ then all signals will be locked to prevent a route being cleared across the points and will remain locked until the FPL is ‘reactivated’ (after the points have been switched).

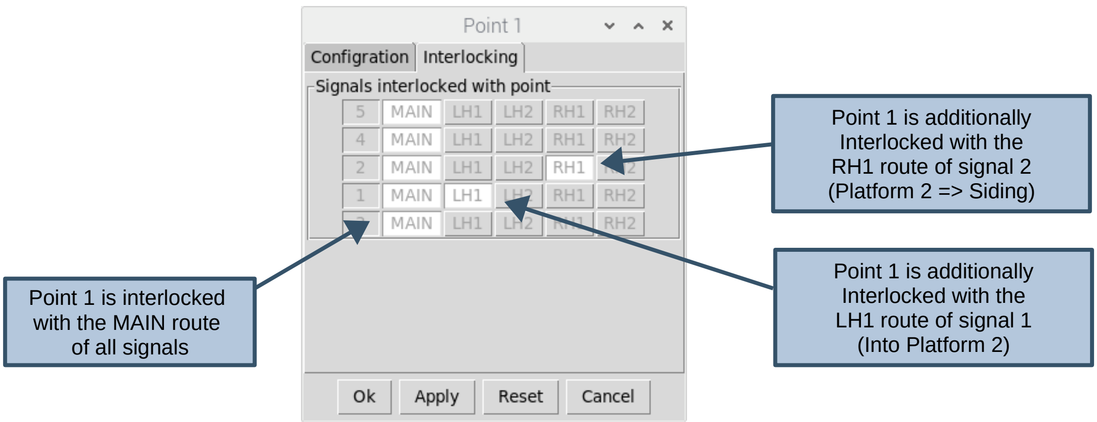
2 When a signal is ‘ON’, it is displaying its most restrictive aspect (DANGER for Home signals or CAUTION for distant signals). When a signal is ‘OFF’ it is displaying its least restrictive aspect (PROCEED for most signal types)




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).




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.



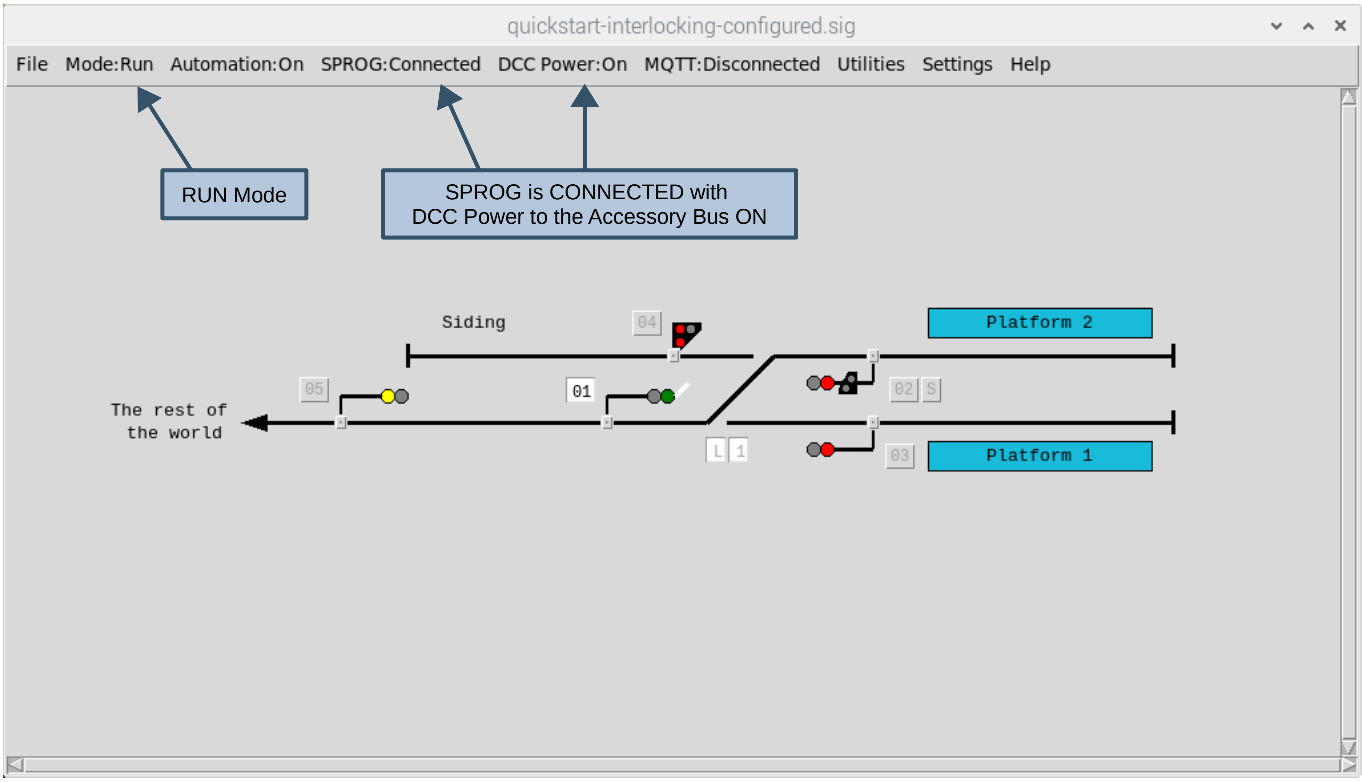
# Operating your Layout

Once you are happy with your layout 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 add 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' as long as the Pi-SPROG has been installed correctly following the instructions provided with the Pi-SPROG. If not then the SPROG connection settings can be checked and tested via **Menubar => Settings => SPROG** )*



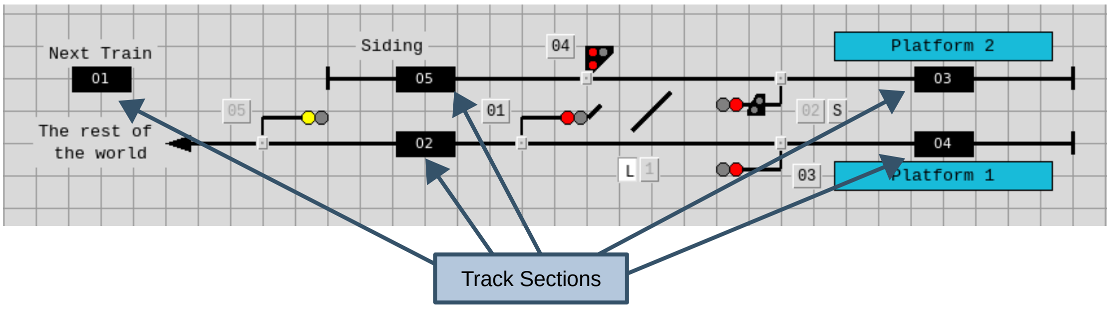

Note that the actual states of the signals and points out on the layout may not reflect the state of the points and signals shown in the application (as we haven't sent out any DCC commands yet). To synchronise everything, set the layout back to its default state via the Menubar (**Mode => Reset**). This will send out the required DCC commands to all signals and points on the layout.

All subsequent changes to signals or points in the application will send out the required DCC commands to change the signals and points on the layout accordingly.

# Adding and configuring track occupancy

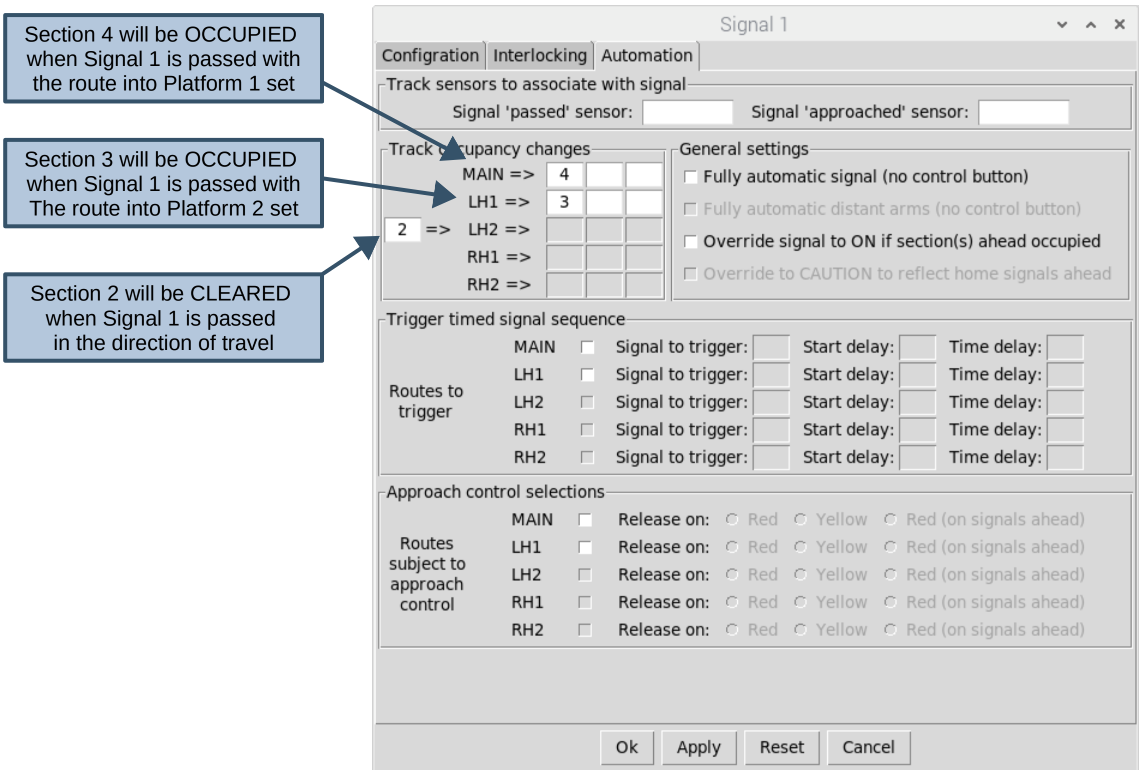
Once you have completed all of the steps above, you will have have created a layout configuration that provides basic interlocking of signals / points, and DCC control of the signals and points out on the layout. The next step is to add track occupancy indications to provide a ‘mimic’ diagram showing the position of trains out on the layout.

Firstly (in EDIT mode), add ‘track sections’ to the schematic and position them on the layout 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 ‘rest of the world’.

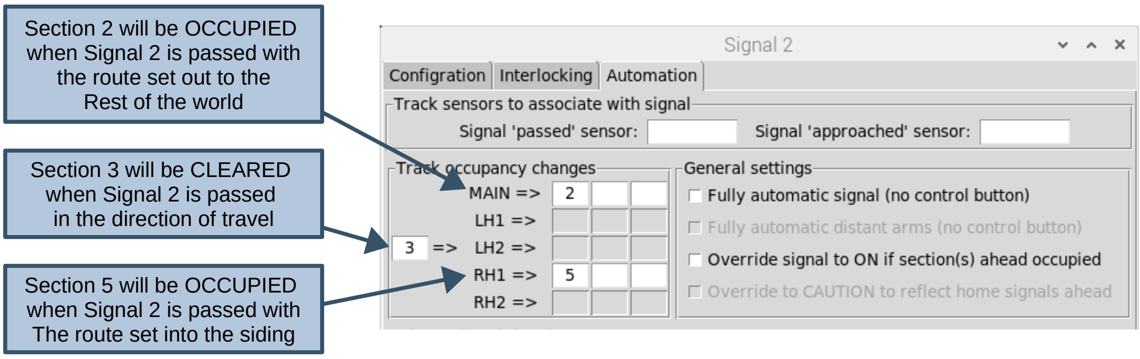



Each signal then needs to be edited to define the track section ‘behind’ the signal (i.e. the track section that would be ‘cleared’ when the signal is passed) and the track section ‘ahead of’ the signal (i.e. the track section that would be set to ‘occupied’ when the signal is passed). For signals controlling 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 route from platform 2 back into the siding) so the configuration would be:

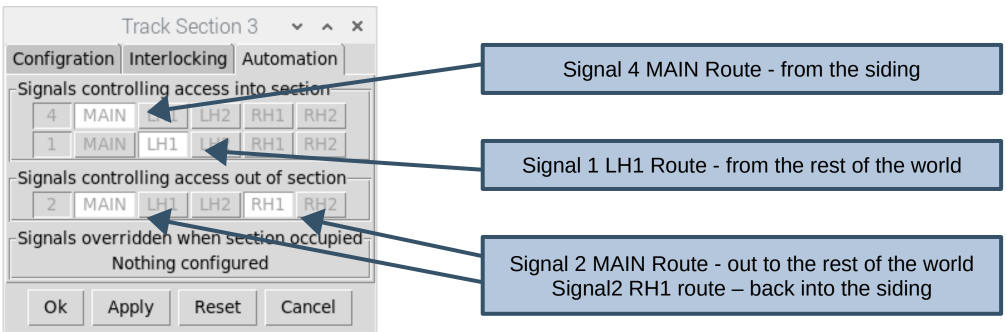



All other signals control a single (MAIN) route, so the configuration is straightforward:

- 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.

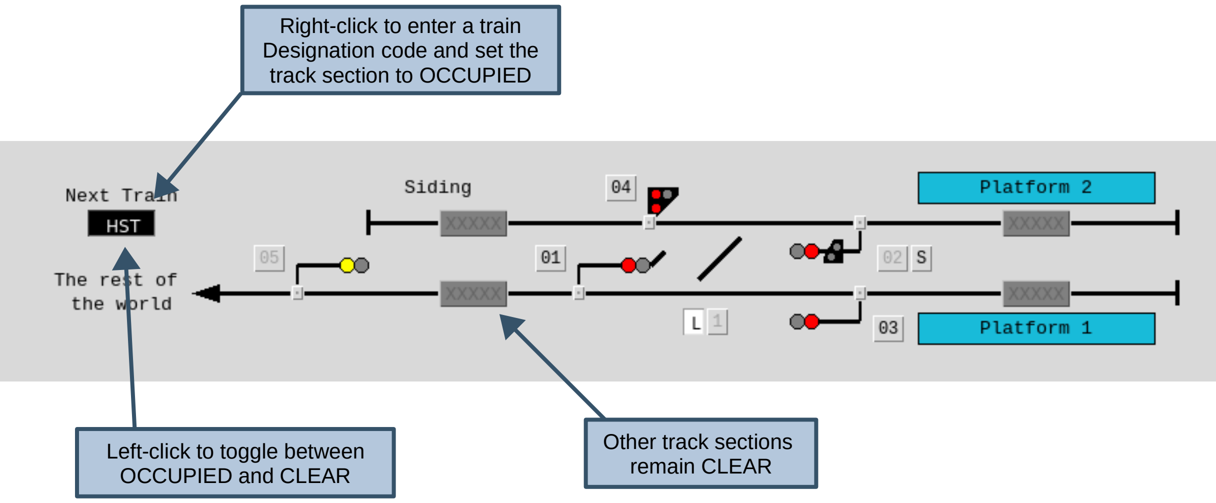
As an example, for Section 3 (Platform 2),  there are 2 signals that provide access into the section, the MAIN route from signal 4 (from the siding into Platform 2) and the LH1 route from Signal 1 (from the rest of the world into Platform 2). There are also two routes out of the section (into the siding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



# Testing track occupancy changes

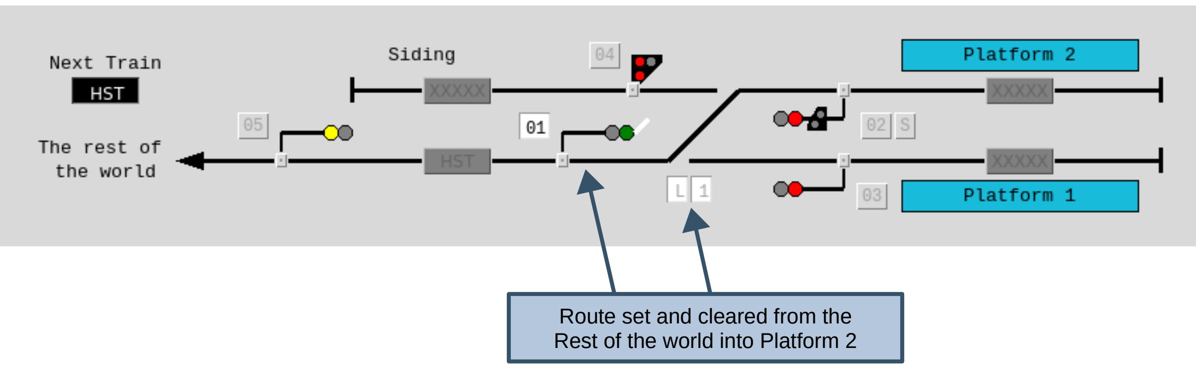
Once all signals have been configured, track occupancy changes can be tested (in RUN mode). Although we haven't configured any external track sensors, the 'signal passed' events to trigger track occupancy changes can be simulated by clicking on the buttons at the base of each signal.

On entering RUN mode, all track sections will initially show as CLEAR (greyed out). To set up the 'next train' to run into the layout, **right-click** on the track section and enter an appropriate train designation code (for this example we'll use 'HST' for simplicity). Once entered, this will set the track section to OCCUPIED (signified by white text on a black background). Track sections can also be toggled between OCCUPIED and CLEAR by left-clicking on them.

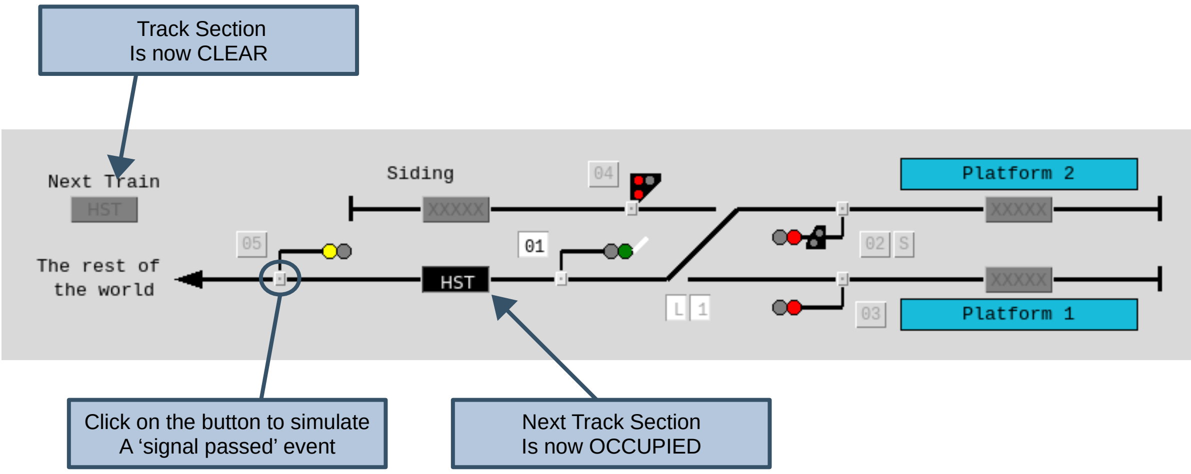



To simulate a train movement from the rest of the world into Platform 2, first configure the route into platform 2 (ensure all signals are 'ON', release the FPL for point 1, switch the point and then re-activate the FPL) and then clear the route by setting Signal 1 to 'OFF'.

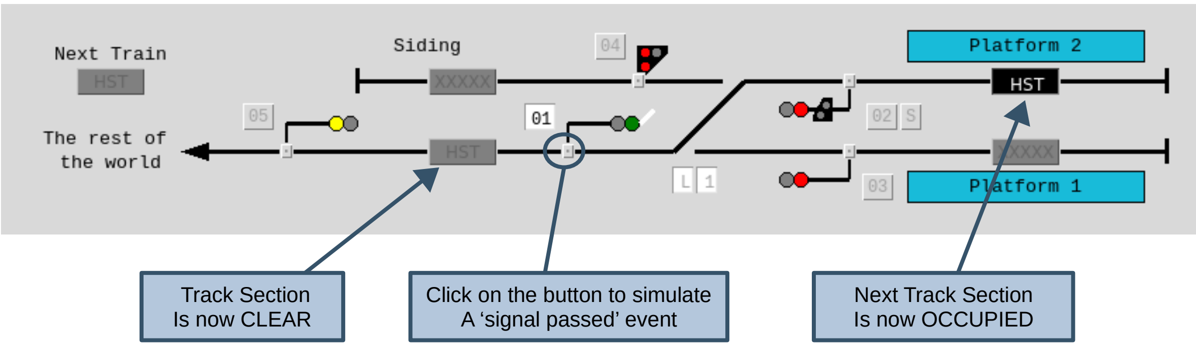
**Note that, apart from distant and shunt-ahead signals, the signal must be off 'OFF' for 'signal passed' events to trigger track occupancy changes. As per the 'real thing', it is the driver's responsibility not to pass a signal at danger!**



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.




Finally, trigger a ‘signal passed’ event for Signal 1 to ‘pass’ the train into Platform 2.




At completion of the train movement, all signals should be returned to ‘ON’, ready for the next one. Other movements you could test would be:

- Platform 2 back into the siding
- From the siding into Platform 2
- Platform 2 to the rest of the world
- The rest of the world to Platform 1
- Platform 1 to the rest of the world

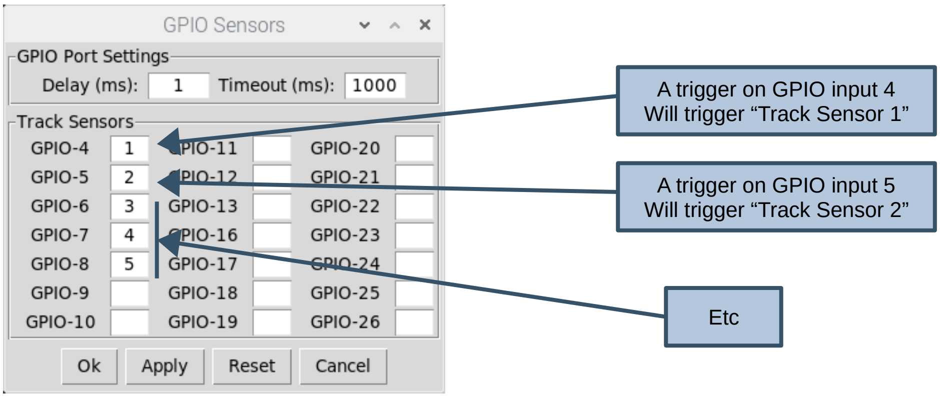
**But Remember, signals need to be ‘OFF ‘to pass trains onto the next track section so all signals need to be set and cleared for the route you are going to test.**

# Configuring external track sensors

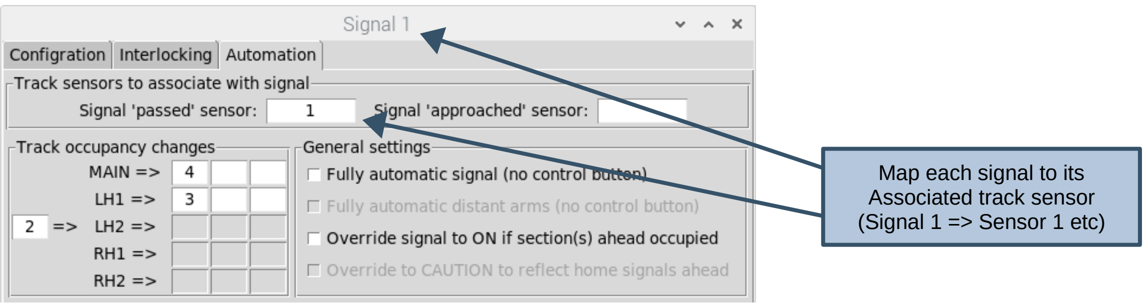
The benefit of using the Raspberry-Pi is that we can connect external track sensors into the GPIO ports of the Raspberry-Pi  and use them to trigger the 'signal passed' events.

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

For this example, we'll allocate 'track sensors' to each signal, matching the numbering of each signal. To do this, just enter a unique identifier for the 'track 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 Track Sensor to generate the 'signal passed' event. This configuration is defined via the **Automation** tab of the signal configuration dialog.



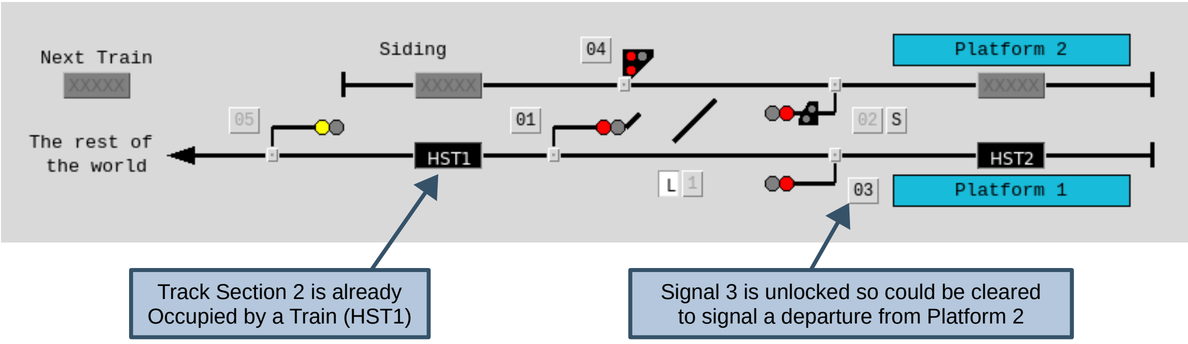

Once all signals have been mapped to their associated track sensors, 'signal passed' events will be triggered whenever the associated GPIO input is taken down to 0V.

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

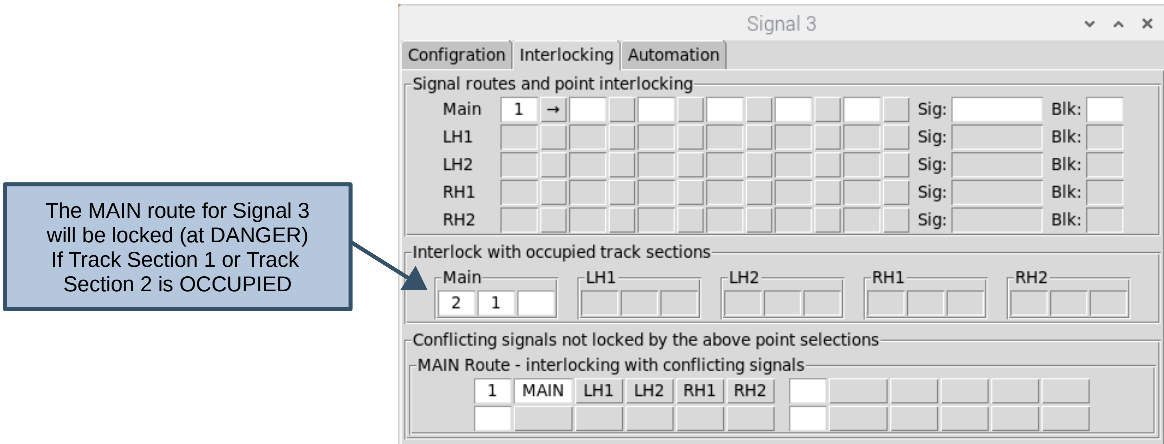
**Note that other sensor types (providing a switched voltage) should <u>never</u> be connected directly to the GPIO pins as this could damage the Raspberry-Pi. In these cases, external opto-isolators are recommended to protect the GPIO input pins.**

# Interlocking with Track Sections

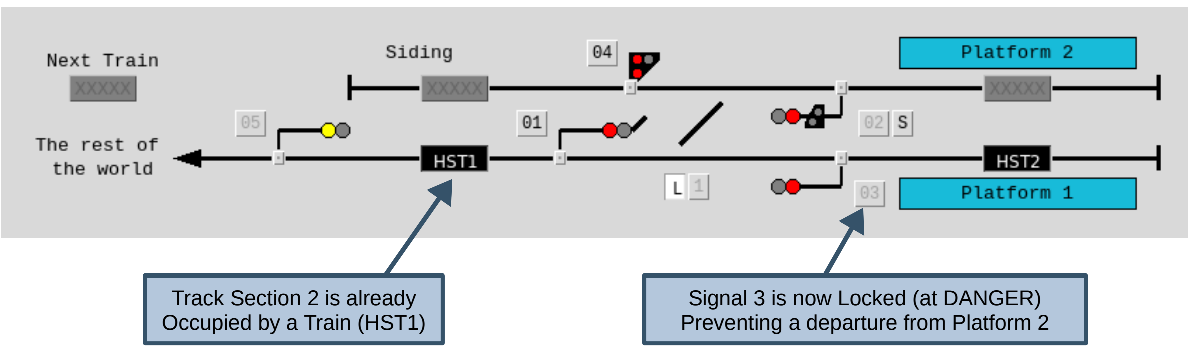
So far, we have configured interlocking for signals and points, but it would still be possible to signal a train onto a track section that was already occupied.



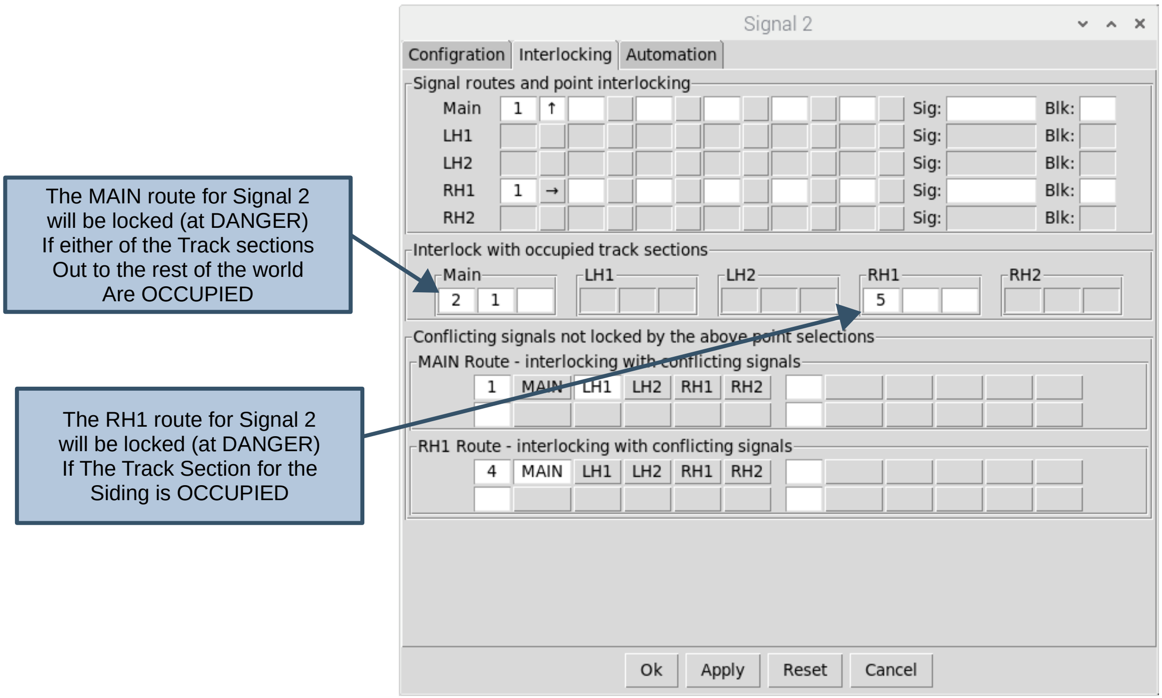

To overcome this, we can also interlock signals with Track sections. In this case we want to Interlock Signal 3 with Track Section 2 and also Track Section 1 (representing a train about to enter the layout from the rest of the world). To configure this, open the signal configuration dialog (in EDIT mode) and select the **Interlocking** Tab.



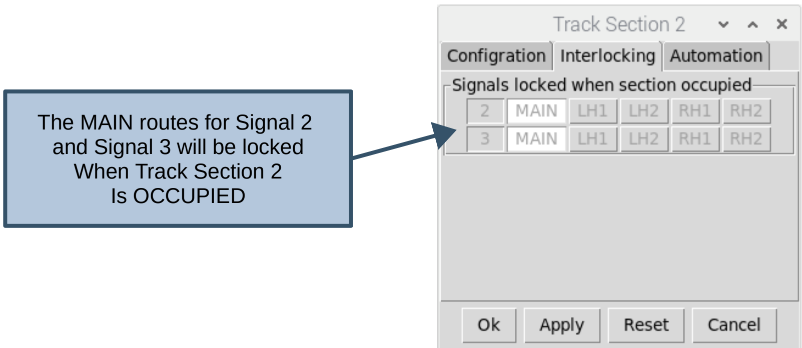

Signal 1 will now be locked (at 'ON') whenever Track Section 1 or 2 is OCCUPIED. Note that if a signal is 'OFF' when a Track Section becomes OCCUPIED then it will only be locked when returned to 'ON' (signals must always be capable of being returned to DANGER).



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).




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).



# 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.

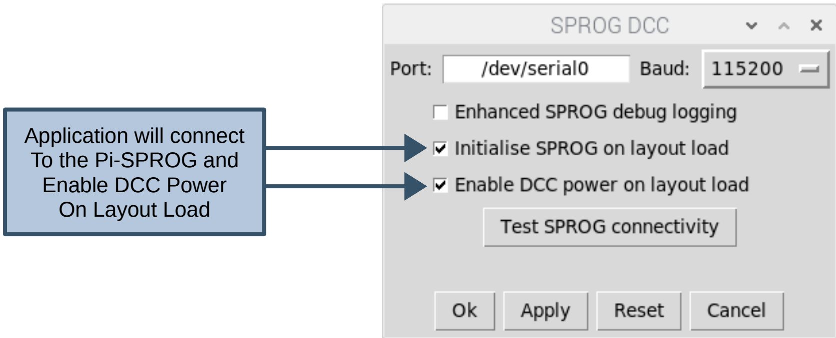
Note that the current state of the layout (in terms of signal and point settings) is saved with the configuration, allowing you to pick up a running session from where you left off.

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 checkboxes and click on **OK** or **APPLY** to save the changes).

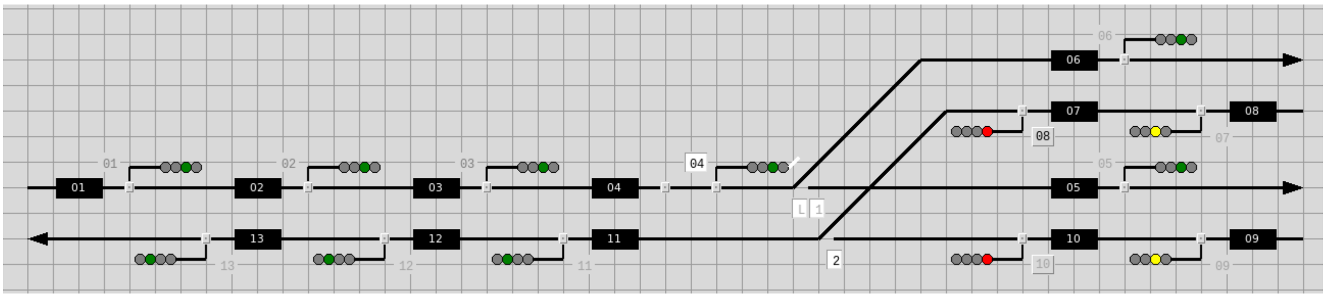



Note that these SPROG settings are specific to the layout configuration (i.e. saved and loaded as part of the layout file) and will not apply to other layouts you load into the application.

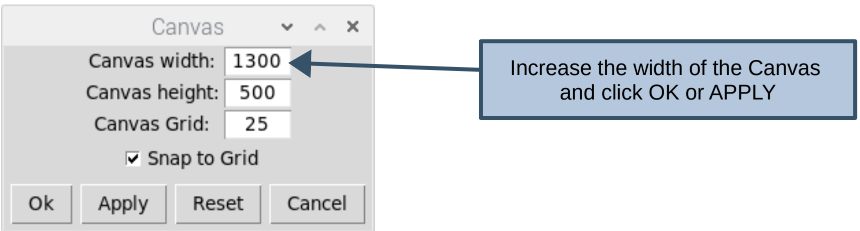
Similarly the Track Sensor configuration (in terms of GPIO port mappings to 'Track Sensors') is specific to the layout and will be saved / loaded with the rest of the layout configuration.

# An automation example

Now we've mastered the basics with the first example, 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.




# Configure 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.

# Configure track occupancy

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

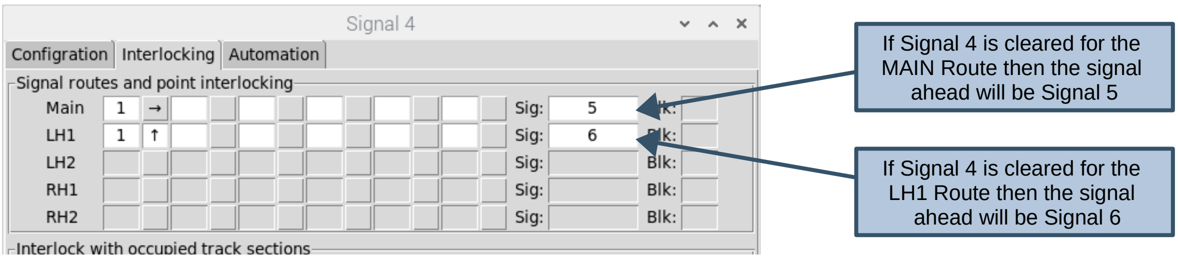
- Signal 4 needs to be configured with a ‘section ahead’ for both routes (Section 5 for the MAIN route and Section 6 for the LH1 route) and a ‘section behind’ (Section 4).
- Signals 6, 5 and 13 only have a ‘section behind’ configured (no ‘section ahead’) .
- All other signals have a ‘section behind’ and a ‘section ahead’ for the MAIN route.

# Configure basic automation

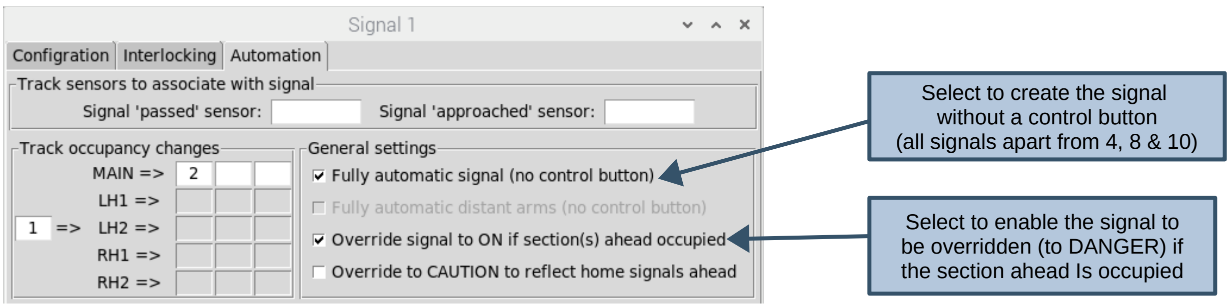
For this layout, we want all signals to:

- 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).

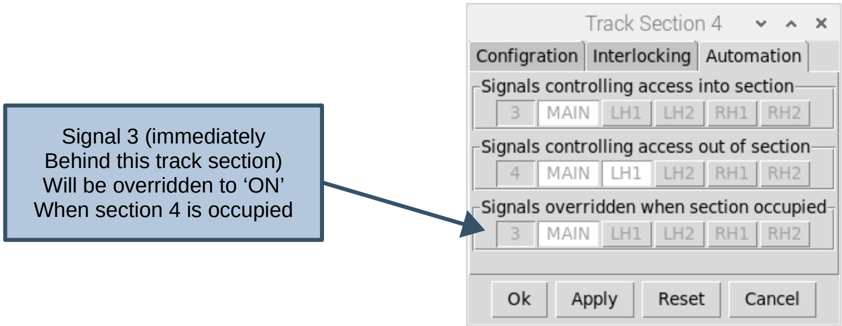
Firstly, configure each signal with details of the ‘signal ahead’ This is achieved via the **Interlocking** tab of the signal configuration dialog. Note that Signal 4 supports two routes, so we have to specify the signal ahead for each route:




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).

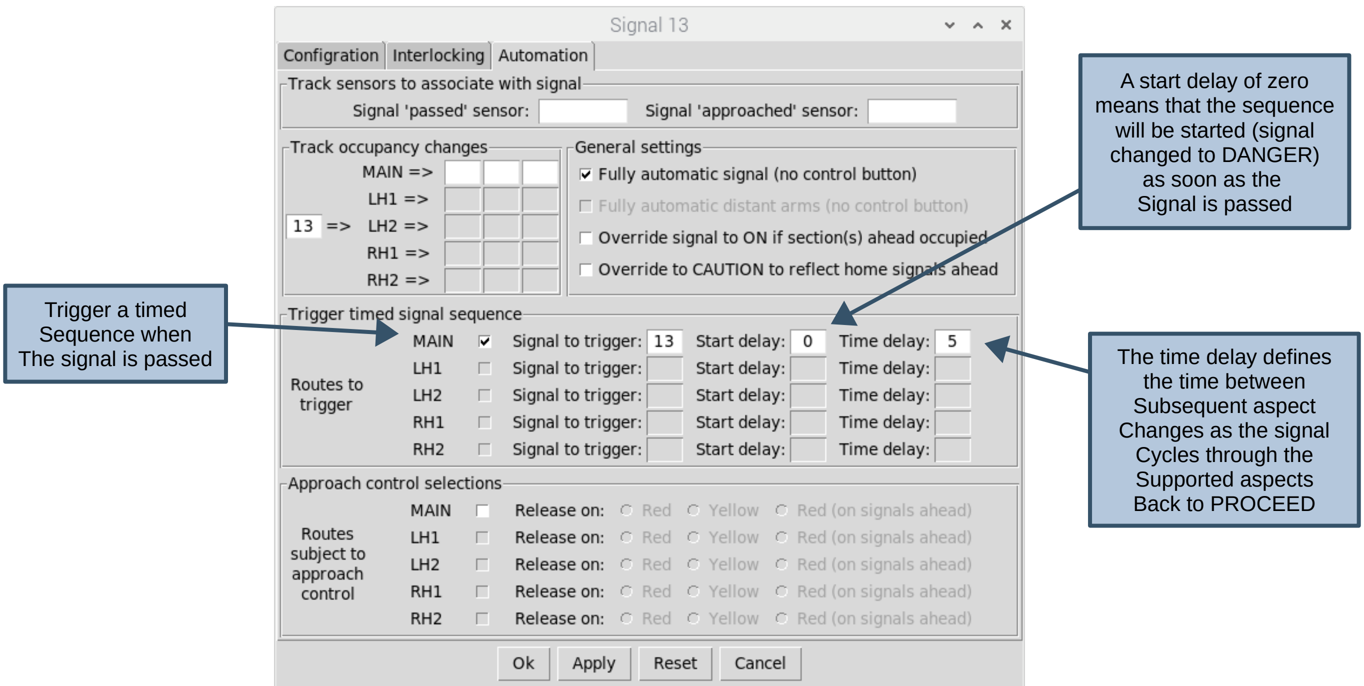



The configuration can then be tested (in RUN mode with Automation ENABLED) by left glicking on each track section in turn to change from CLEAR to OCCUPIED and then back to CLEAR. When the track section is OCCUPIED, the signal behind the track section will display DANGER.

## Configure 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.




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.

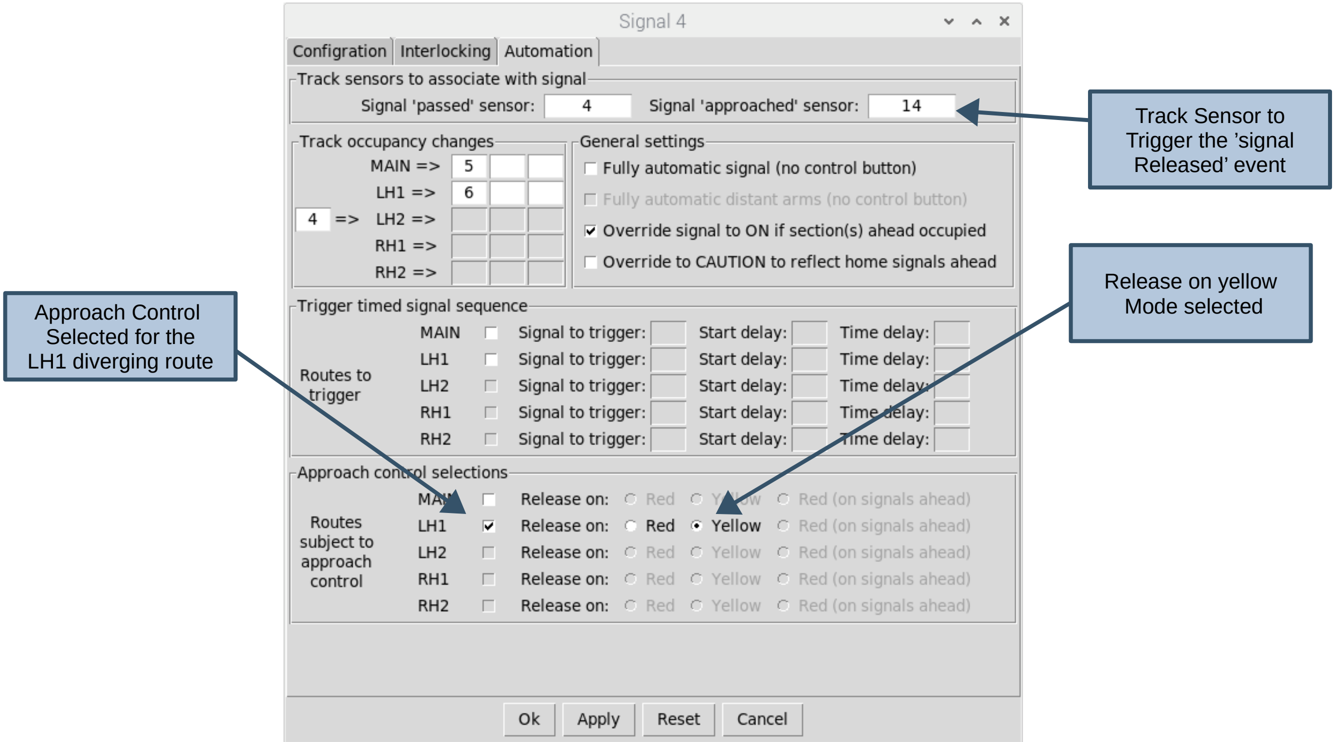
# Configure approach control

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.

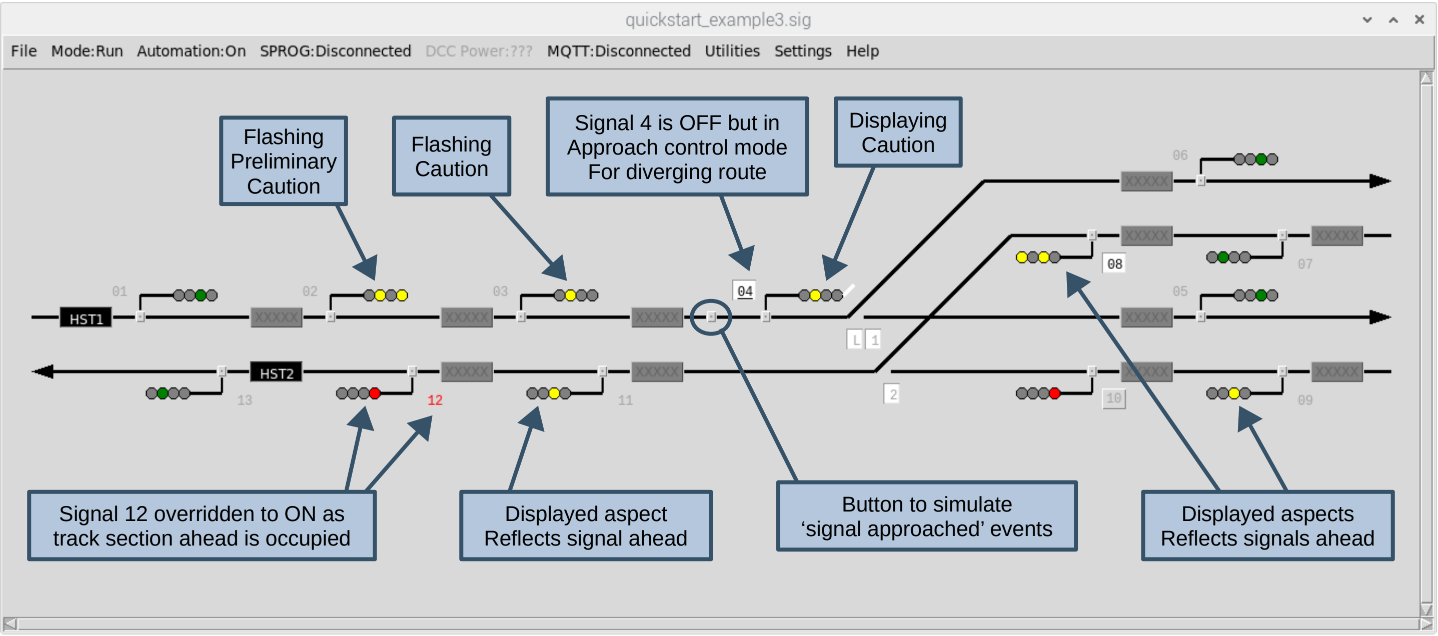



Signal 4 will now be displayed with a second button positioned on the track (on the approach to the signal). This is the button to simulate the ‘signal approached’ event to ‘release’ the signal.

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).

## 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.



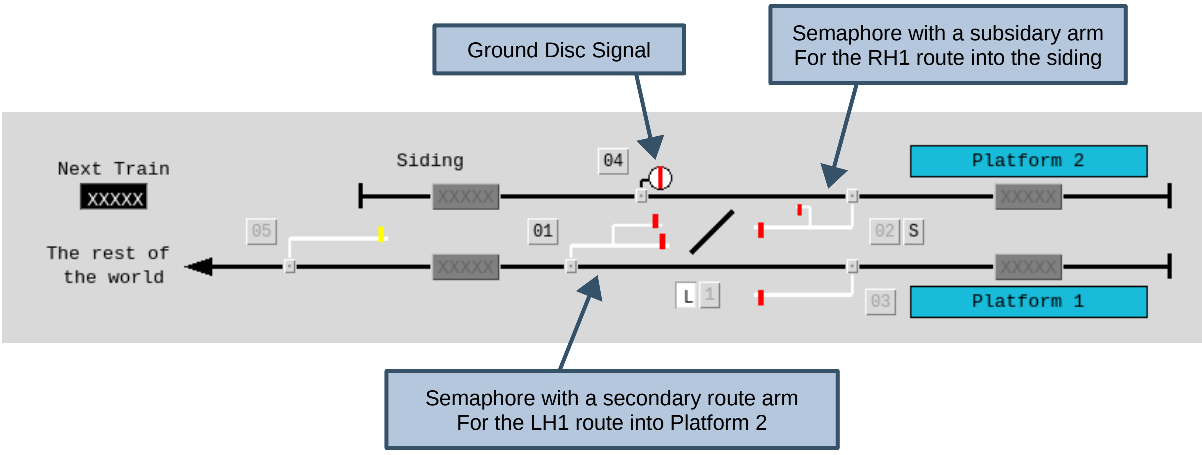
# Still to discover

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.

- Signal Theater Indications – supported by main semaphore and main colour light signals. Provides the ability to display a single ‘character’ for the selected signal route
- Block Instruments – mainly intended for layouts split into separate ‘block sections’ where the instruments can be used to control the movement of trains between the two block sections. This is a particularly useful feature when using multiple application instances networked together, where each instance represents a different block section, as the block instruments can be used to communicate (via bell codes) between the signal boxes
- 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 – To ensure a distant signal will always display CAUTION if any if any home signals ahead (within the block section) are at DANGER.
- Interlocking - Interlock on home signals ahead – To prevent the distant signal for a block section 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. Note that this requires all application instances to be connected to a MQTT broker, either on the local network (for example, installed on the Raspberry-Pi running the application) or out on the internet. In either case, t’s not too difficult with a basic level of IT knowledge as long as you follow the instructions – if you can do that, configuring the application to use networking will be easy.

# Appendix 1 - Using semaphore signals

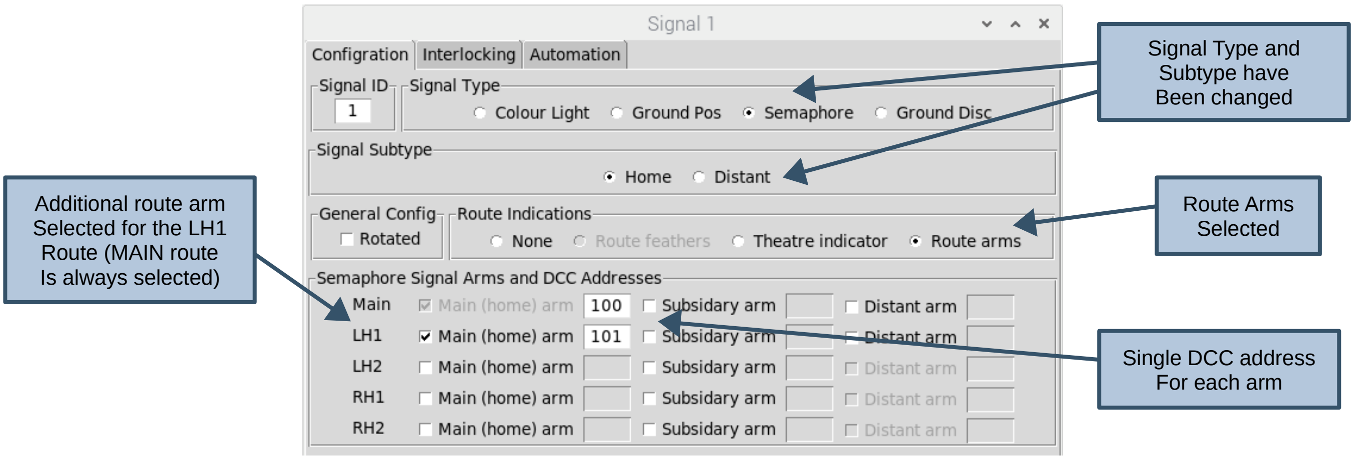
In the first quick-start example, we configured a layout using colour light signals. Depending on the period modeled, you might prefer the use of semaphore signals on your layout.




Once you have your layout configured, its easy to swap between the two signal types as the only major differences in the configuration are:

- The route indications – route feathers vs additional semaphore route arms
- DCC addressing – DCC command sequence per colour light aspect vs a single DCC address for each semaphore arm (which is either OFF or ON)

In this example, Signal 1 has been changed 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 has been allocated its own DCC address. The remainder of the configuration (on the Interlocking and Automation tabs) remains identical to Signal 1 in the colour light signalling example.




The other signals have also been 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.

As per Signal 1, only the signal types, route indications and DCC addressing needs to be changed. The rest of the configuration remains identical to the colour light signalling example.

# Appendix 2 - DCC programming of signals and points

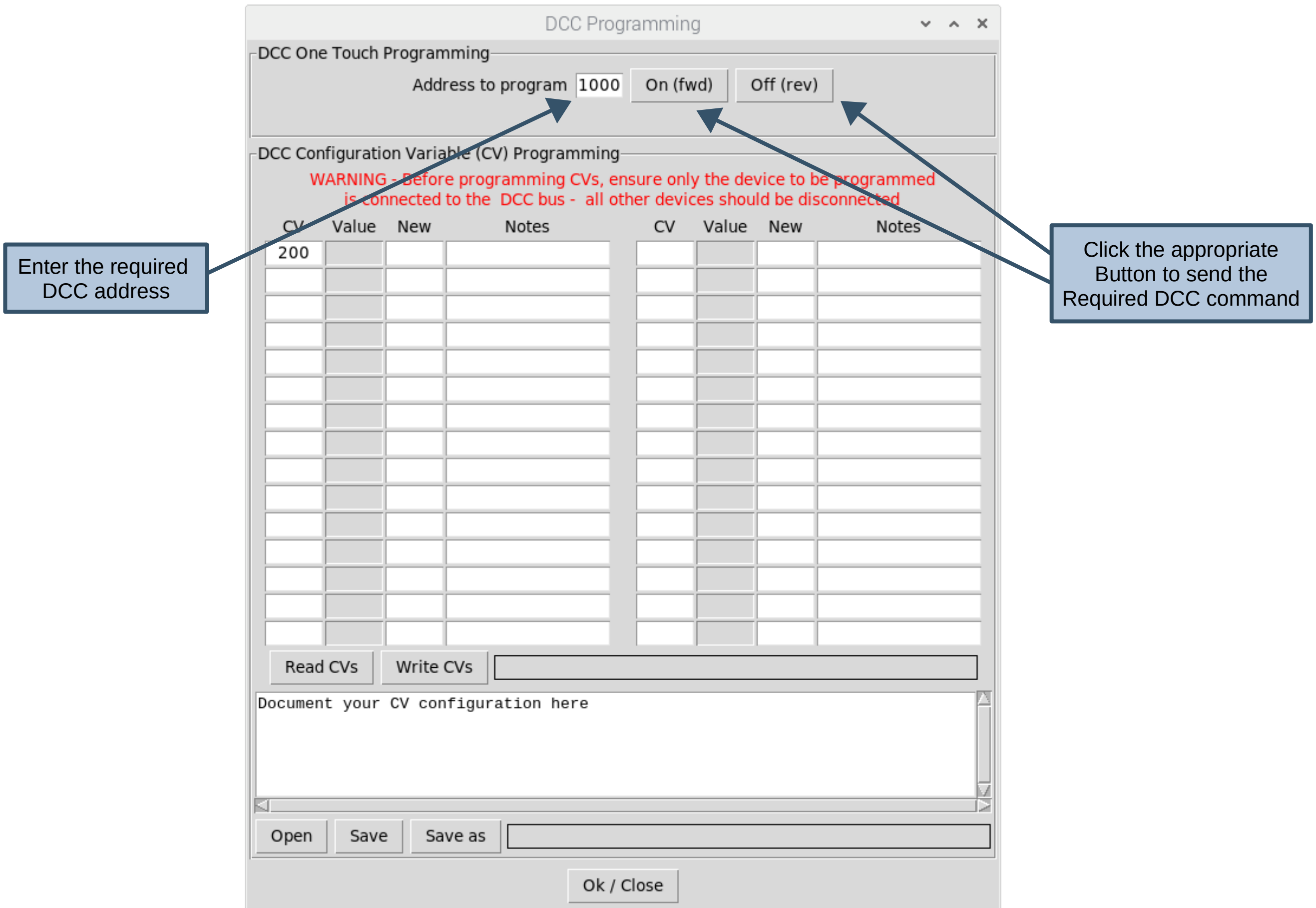
A basic DCC programming utility is provided to enable ‘one touch’ programming (suitable for the majority of DCC point & signal decoders on the market) and Configuration Variable (CV) programming (suitable for more complex decoders such as the Signallist SC1).

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).

For this quick-start guide, we’ll focus on ‘one touch’ 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.




**To program CVs, 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**.