



Introduction to PanelPro - Logix™

Dick Bronson - *RR-CirKits, Inc.*

Intro to Logix™

Indirect layout control (PP-clinic-3)



Why LogiX?

When Dave Duchamp first started adding a graphical logic package to JMRI we wondered about what to call it. “Logic” seemed to be a logical name for logic, but Dave had already added 'Lights' as a function, therefore 'L' was no longer available as an item name, so he just used 'X' instead. The logic function was 'Internal' to JMRI, so its system name was 'I'. This means the the proper identifier for the logic function became 'IX' and we jokingly started calling them Logix in our e-mail discussions because of the 'IX'. The name has stuck.

The original Logix were functionally similar to industrial ladder logic in that they did not have any parenthetical structure. Pete Cressman, a California Java programmer, at the urging of David Parks, a modeler trying to use Logix for his extensive B&O CPL signals, figured out how to add mixed logic to Logix in a graphical way. This new capability is fully available in Release 2.6 of JMRI, and will change this presentation somewhat from last years version.



Indirect Layout Control

In our previous clinic we simply tied our active icons directly to the layout commands that we needed to send. This is no more sophisticated than drilling some holes in a piece of Masonite, spray painting some lines, mounting some switches and lamps, and then connecting them to our switch machines. Granted a computer can usually be found for not very much money, but a few switches or push buttons, a chopped up string of Christmas tree lights, and some paint would be cheaper.

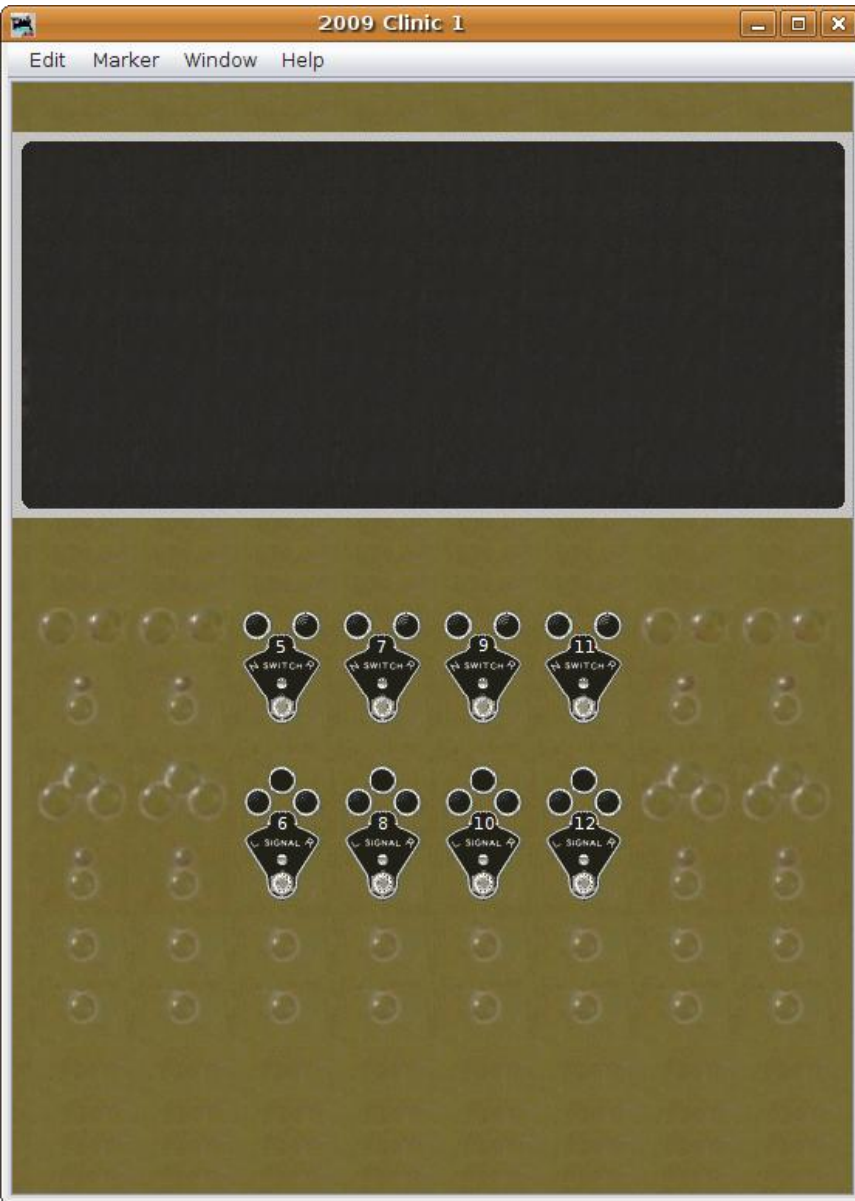
On the prototype railroads it is not allowable to have remote control of turnouts without some fairly reliable method of knowing the current position of the points and preventing them from ever being changed while a train is crossing them. (or about to) Now that we mention it, these are pretty good things to do for our models as well, even if the life hazard is less. (not counting what might happen to the dispatcher when he accidentally sends that new brass onto the floor through that spot with no scenery yet)

All this to say, maybe just flipping a turnout with a remote switch isn't the best idea after all, especially if you can't see it from the panel.



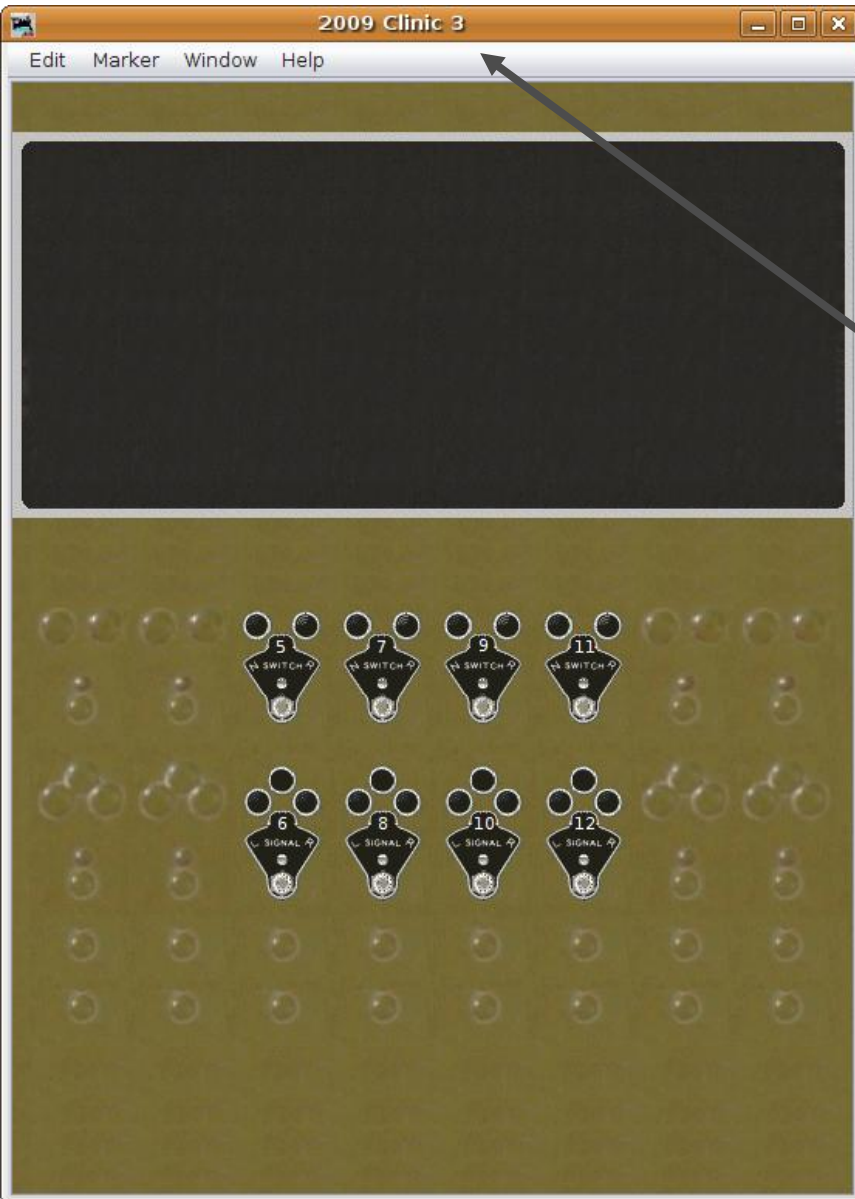
Indirect Layout Control

- First lets load in the basic panel background that we made in clinic #1 then rename it and save it as clinic #3



Indirect Layout Control

Fixed images



Indirect Layout Control

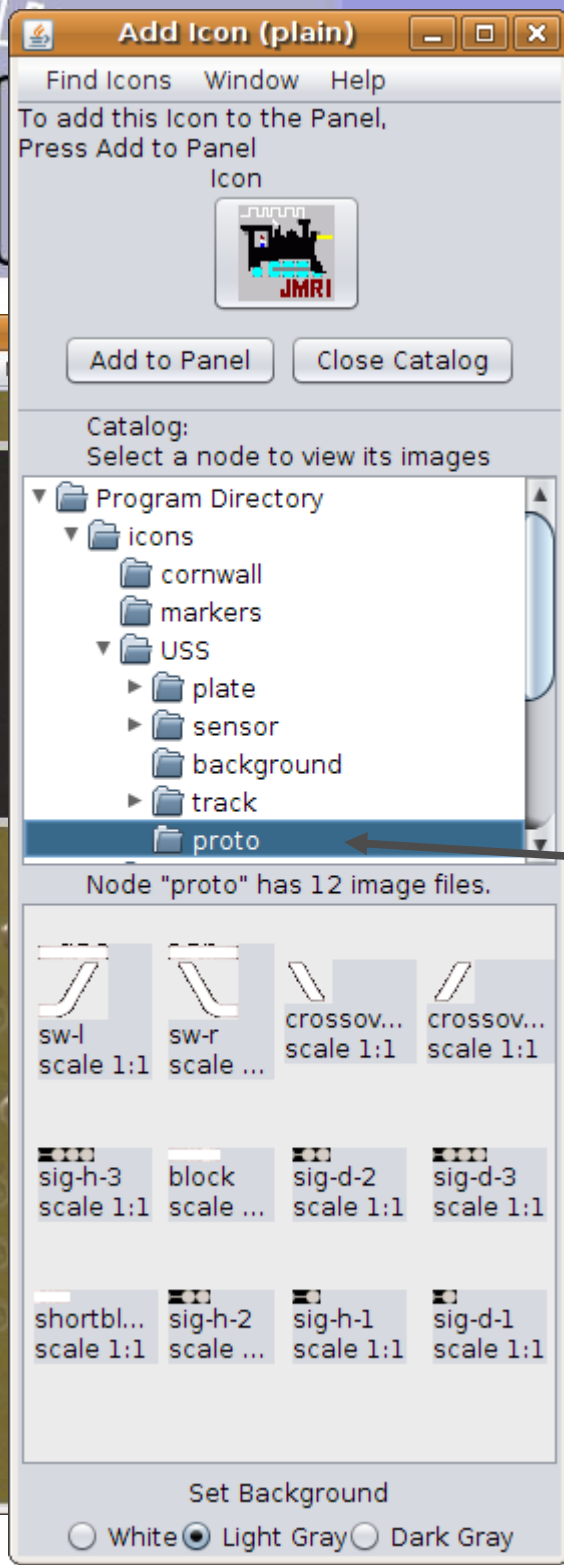
- First lets load in the basic panel background that we made in clinic #1 then rename it and save it as clinic #3
- You are expected to know how to do all the basic operations already covered in previous sessions, so I am not going to repeat the detail of each operation as we move along.

Indirect Layout Control

Fixed images

Indirect Layout Control

- First lets load in the basic panel background that we made in clinic #1 then rename it and save it as clinic #3
- You are expected to know how to do all the basic operations already covered in previous sessions, so I am not going to repeat the detail of each operation as we move along.
- Navigate to the 'proto' folder where we have a set of images created from photographs of an original classic era prototype US&S CTC machine.

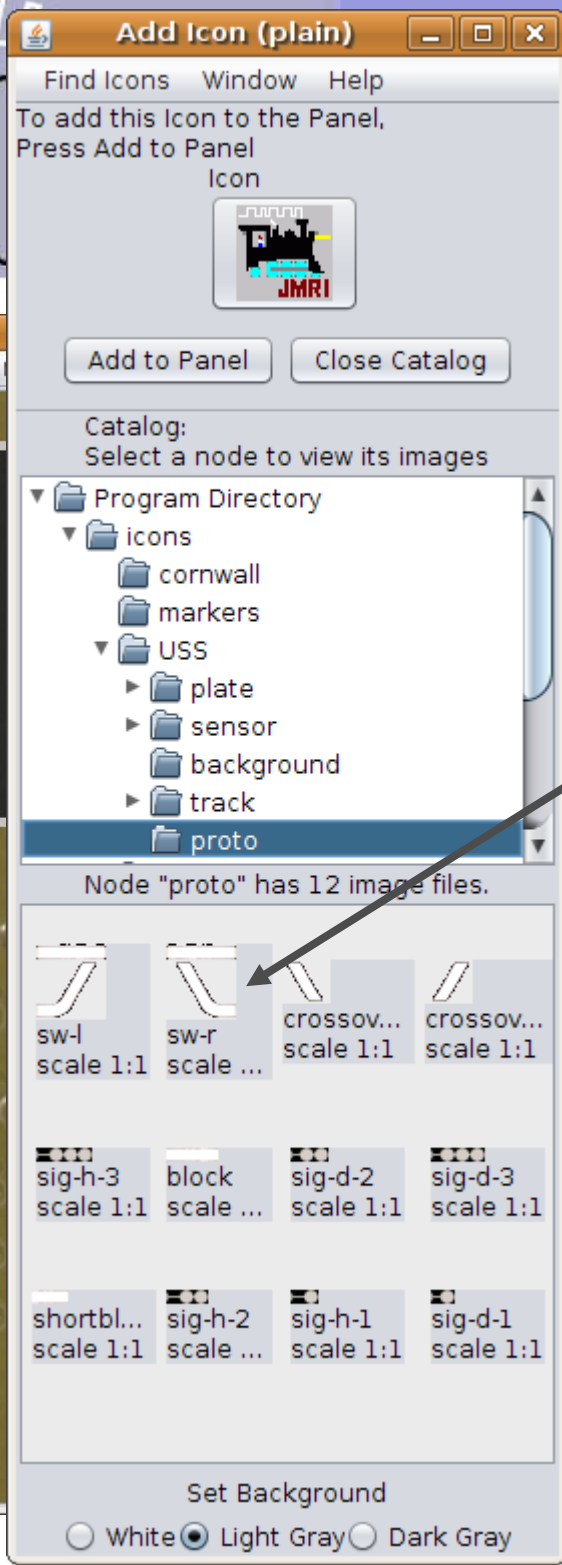


Indirect Layout Control

Fixed images

Indirect Layout Control

- These few images are not designed for animation, but for constructing a more realistic panel.

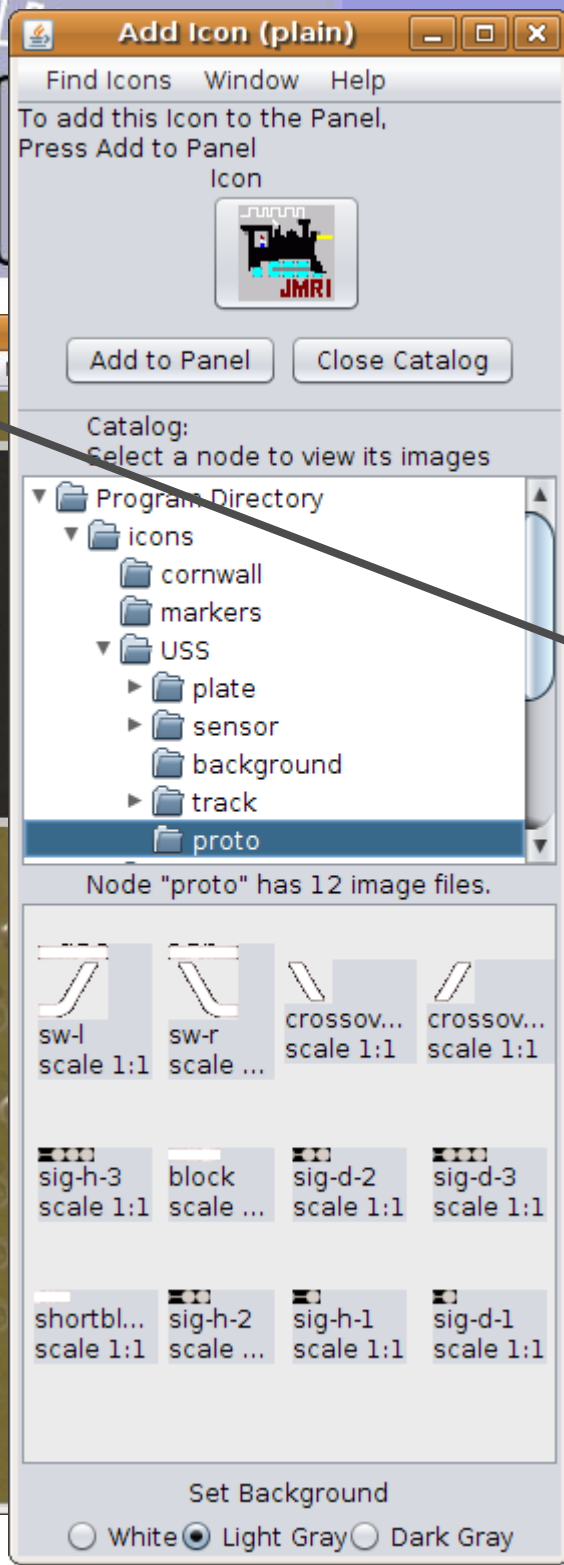


Indirect Layout Control

Fixed images

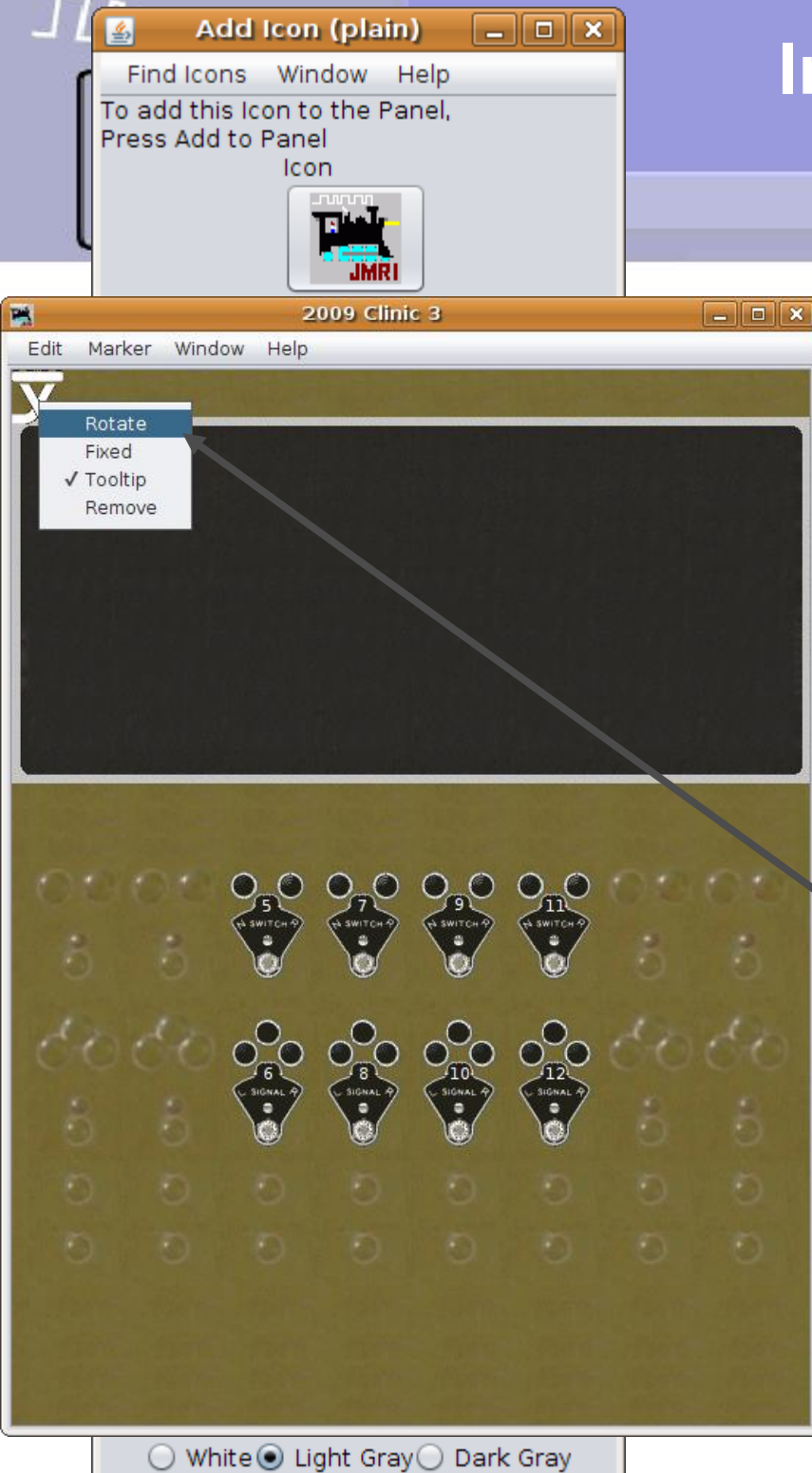
Indirect Layout Control

- These few images are not designed for animation, but for constructing a more realistic panel.
- Use the 'Add to Panel' button to add two left (sw-l) and two right (sw-r) turnout icons to our panel.



Indirect Layout Control

Fixed images

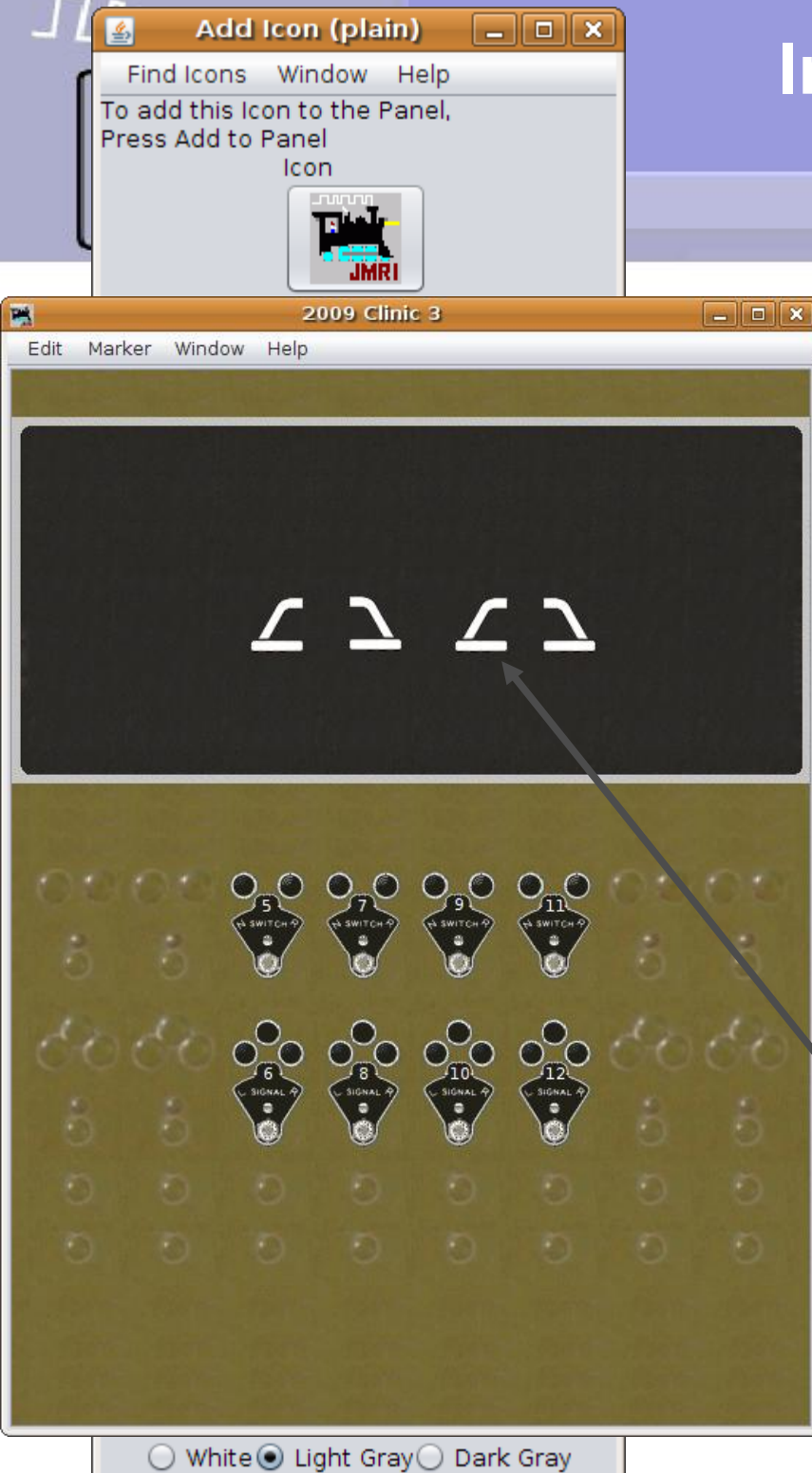


Indirect Layout Control

- These few images are not designed for animation, but for constructing a more realistic panel.
- Use the 'Add to Panel' button to add two left (sw-l) and two right (sw-r) turnout icons to our panel.
- These images only face in one direction, so they will need to be rotated for our use on this panel. Right click (meta for Mac) to bring up the tools, then click on 'Rotate' to rotate 90°.

Indirect Layout Control

Fixed images



Indirect Layout Control

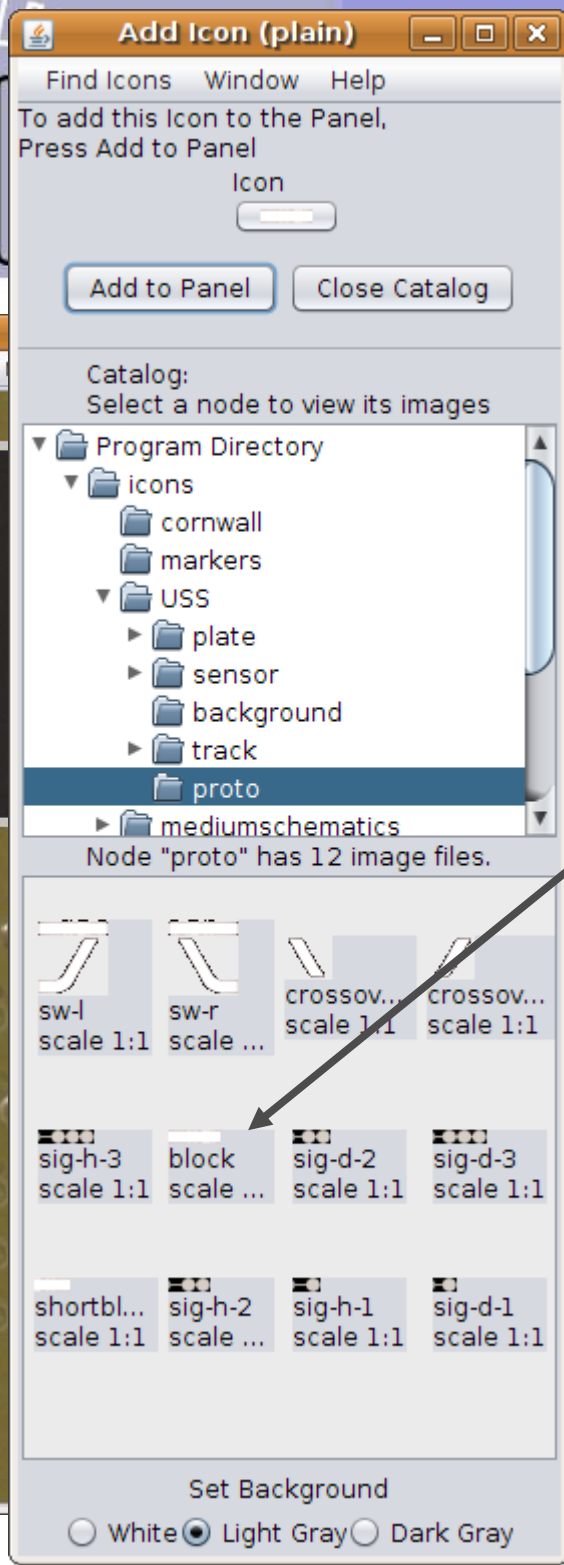
- These few images are not designed for animation, but for constructing a more realistic panel.
- Use the 'Add to Panel' button to add two left (sw-l) and two right (sw-r) turnout icons to our panel.
- These images only face in one direction, so they will need to be rotated for our use on this panel. Right click (meta for Mac) to bring up the tools, then click on 'Rotate' to rotate 90°.
- 'Rotate' each icon twice, and then position it on the panel.

Indirect Layout Control

Fixed images

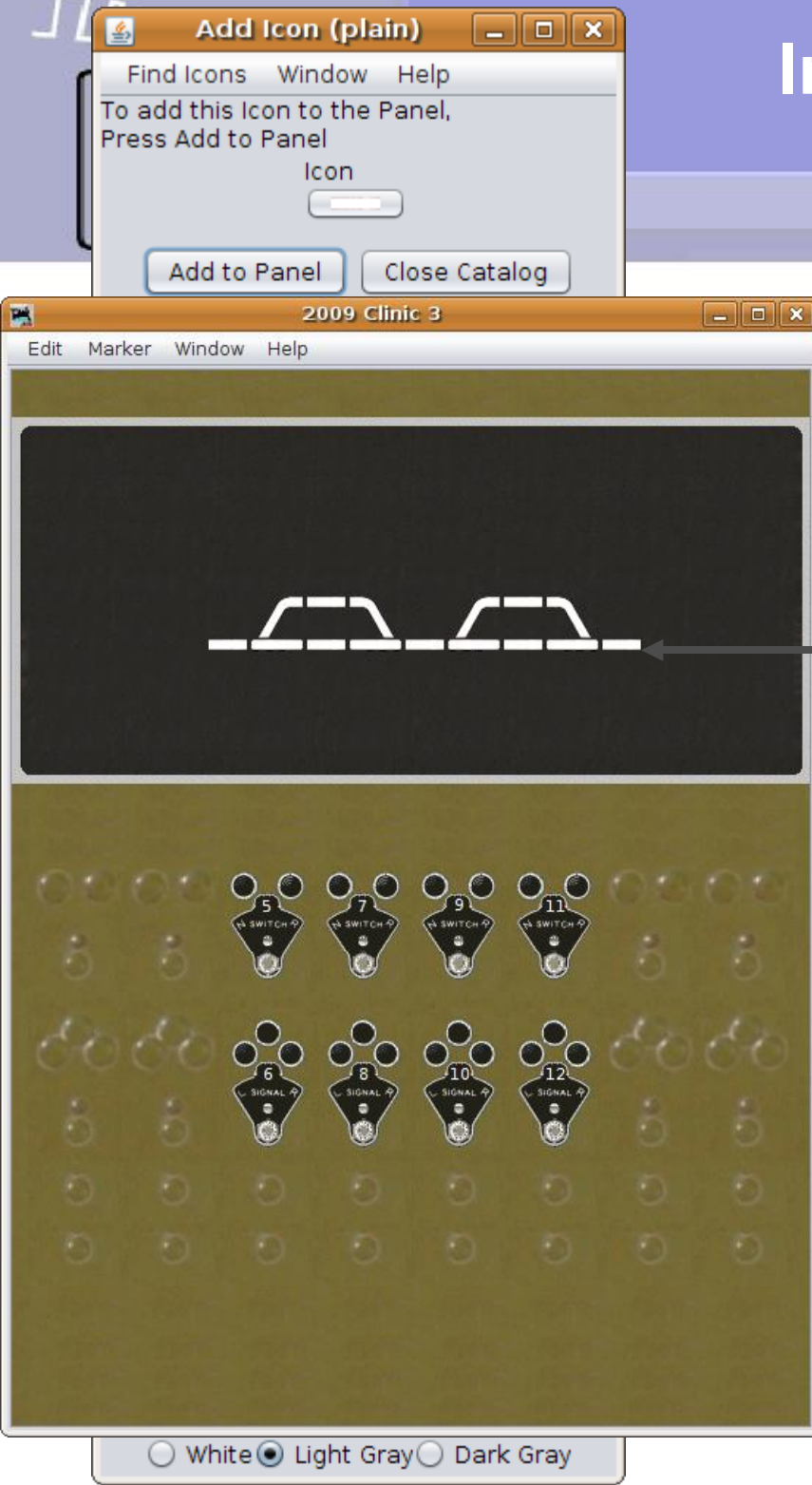
Indirect Layout Control

- Now use the 'Add to Panel' button to add seven 'block' icons to our panel.



Indirect Layout Control

Fixed images

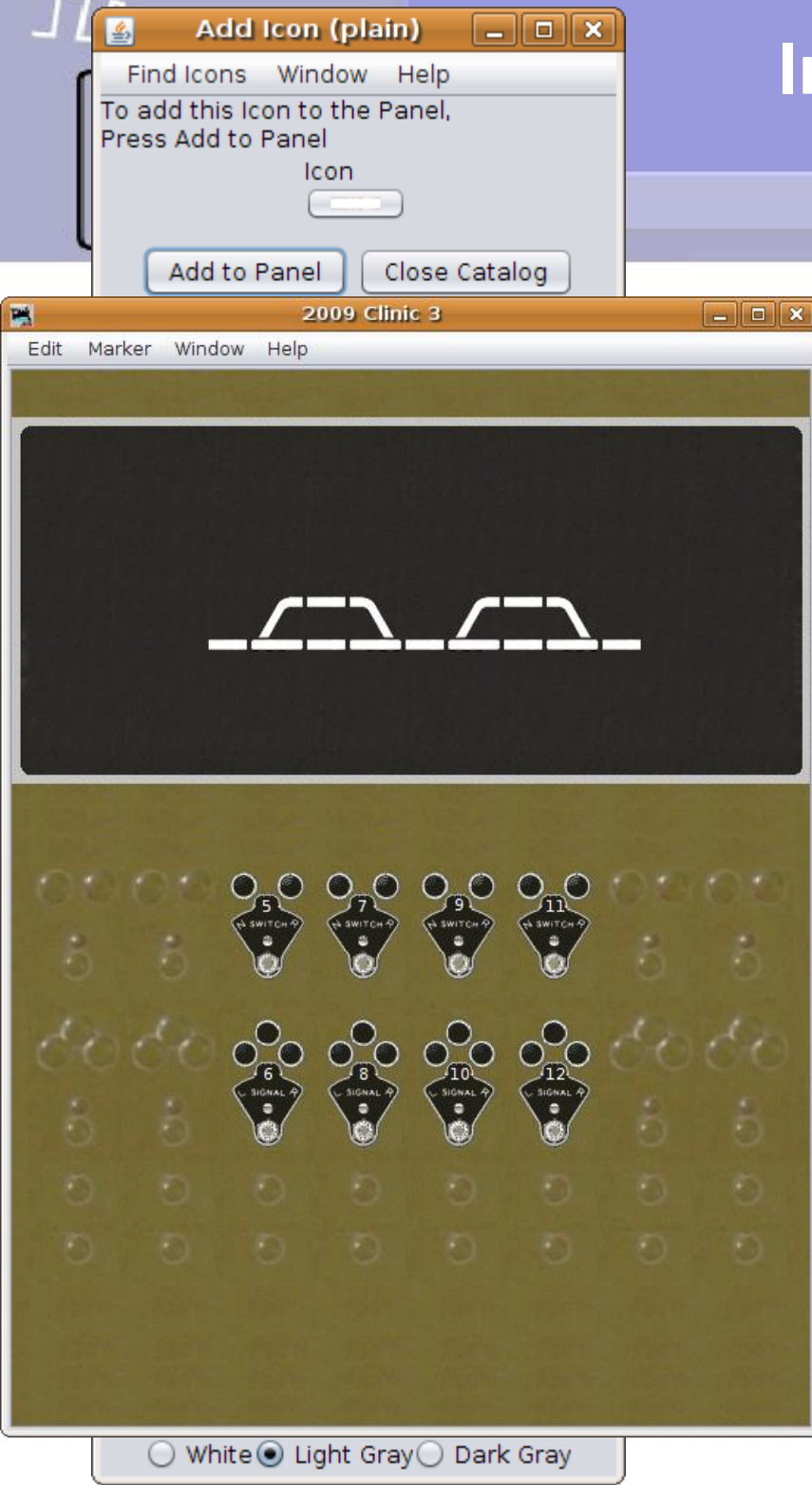


Indirect Layout Control

- Now use the 'Add to Panel' button to add seven 'block' icons to our panel.
- Place them appropriately.

Indirect Layout Control

Sensor images

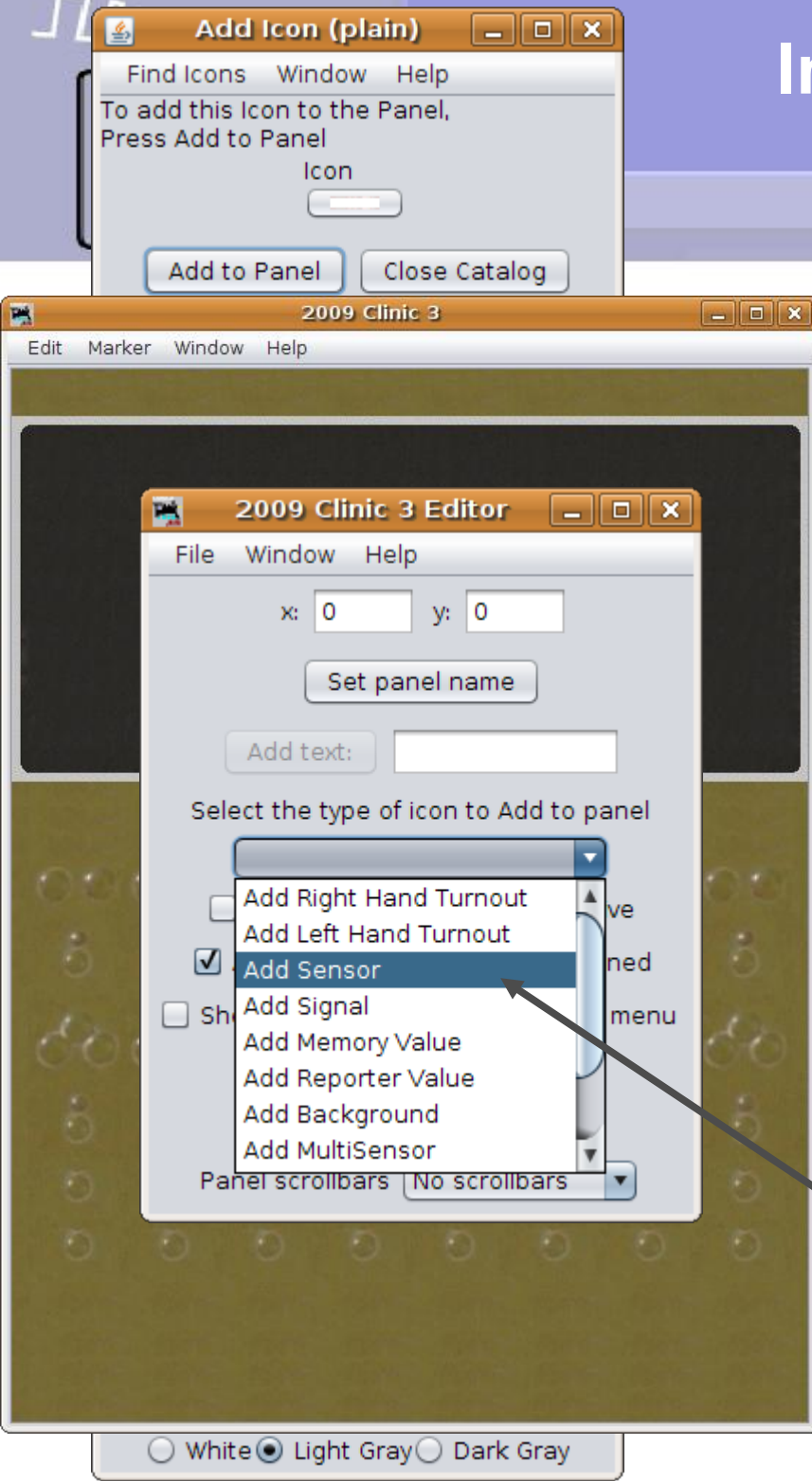


Indirect Layout Control

- Now use the 'Add to Panel' button to add seven 'block' icons to our panel.
- Place them appropriately.
- One of the 'rules' we have for remote operation is that we do not throw a switch under a train. To accomplish that we need to know when a train is on the switch or 'OS' (On Switch) section. (OS can mean other things such as 'On Sheet', Off Sheet, etc.)

Indirect Layout Control

Sensor images



Indirect Layout Control

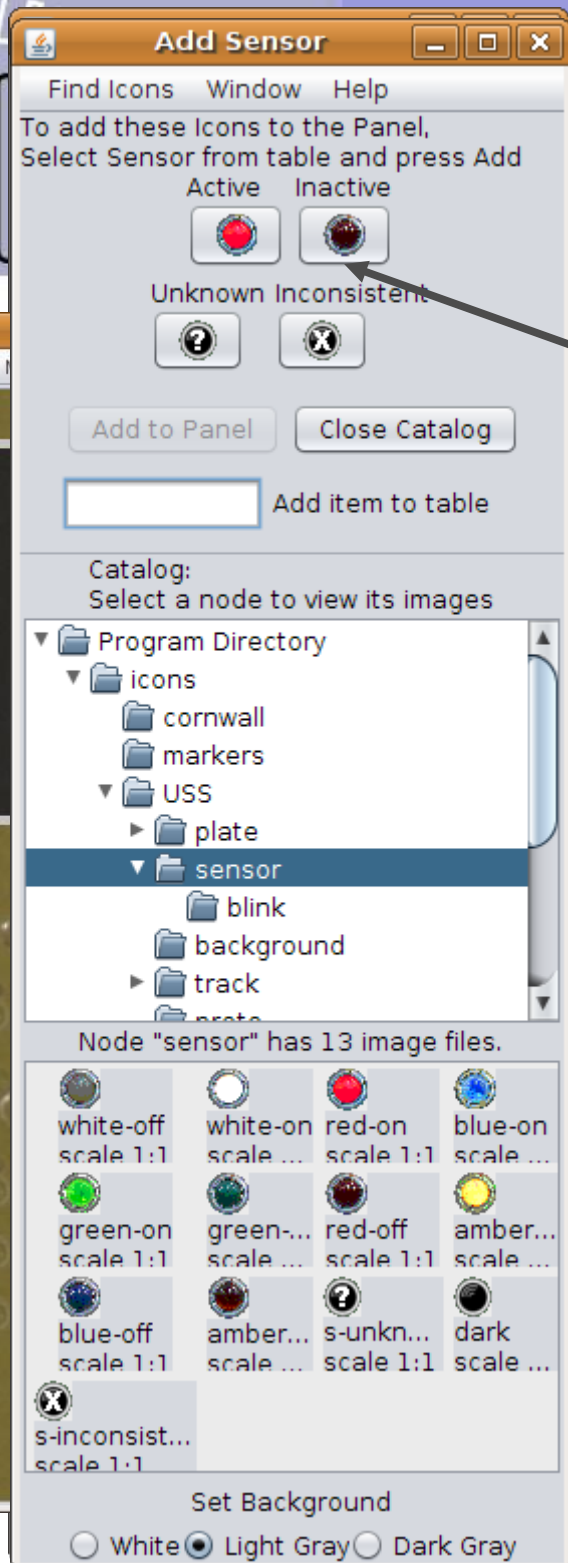
- Now use the 'Add to Panel' button to add seven 'block' icons to our panel.
- Place them appropriately.
- One of the 'rules' we have for remote operation is that we do not throw a switch under a train. To accomplish that we need to know when a train is on the switch or 'OS' (On Switch) section. (OS can mean other things such as 'On Sheet', Off Sheet, etc.)
- Choose 'Add Sensor' in the Panel Editor window.

Indirect Layout Control

Sensor images

Indirect Layout Control

- Select the images you want to use for your OS detection. The USS default for a panel was red jewels for OS, and white jewels for blocks. Many railroads had their own standards including all white, all red, all blue, etc.

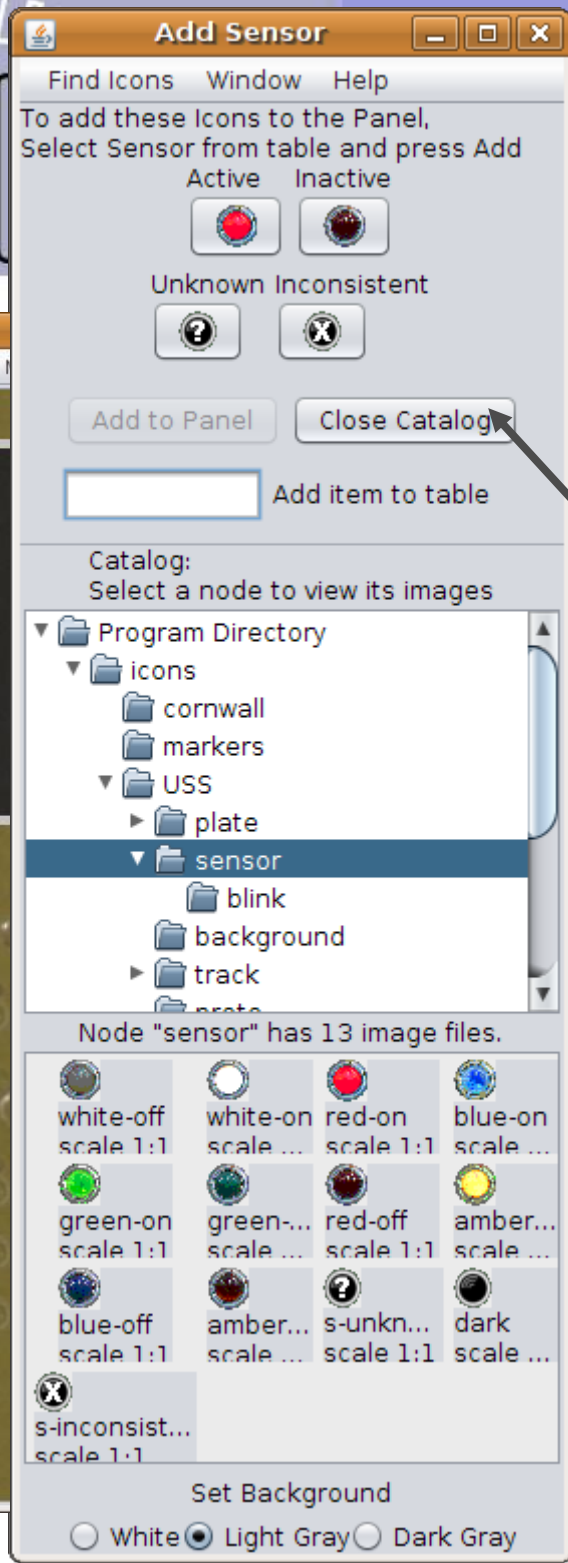


Indirect Layout Control

Sensor images

Indirect Layout Control

- Select the images you want to use for your OS detection. The USS default for a panel was red jewels for OS, and white jewels for blocks. Many railroads had their own standards including all white, all red, all blue, etc.
- To close the selection catalog, and open a list of existing sensors simply click on 'Close Catalog'.

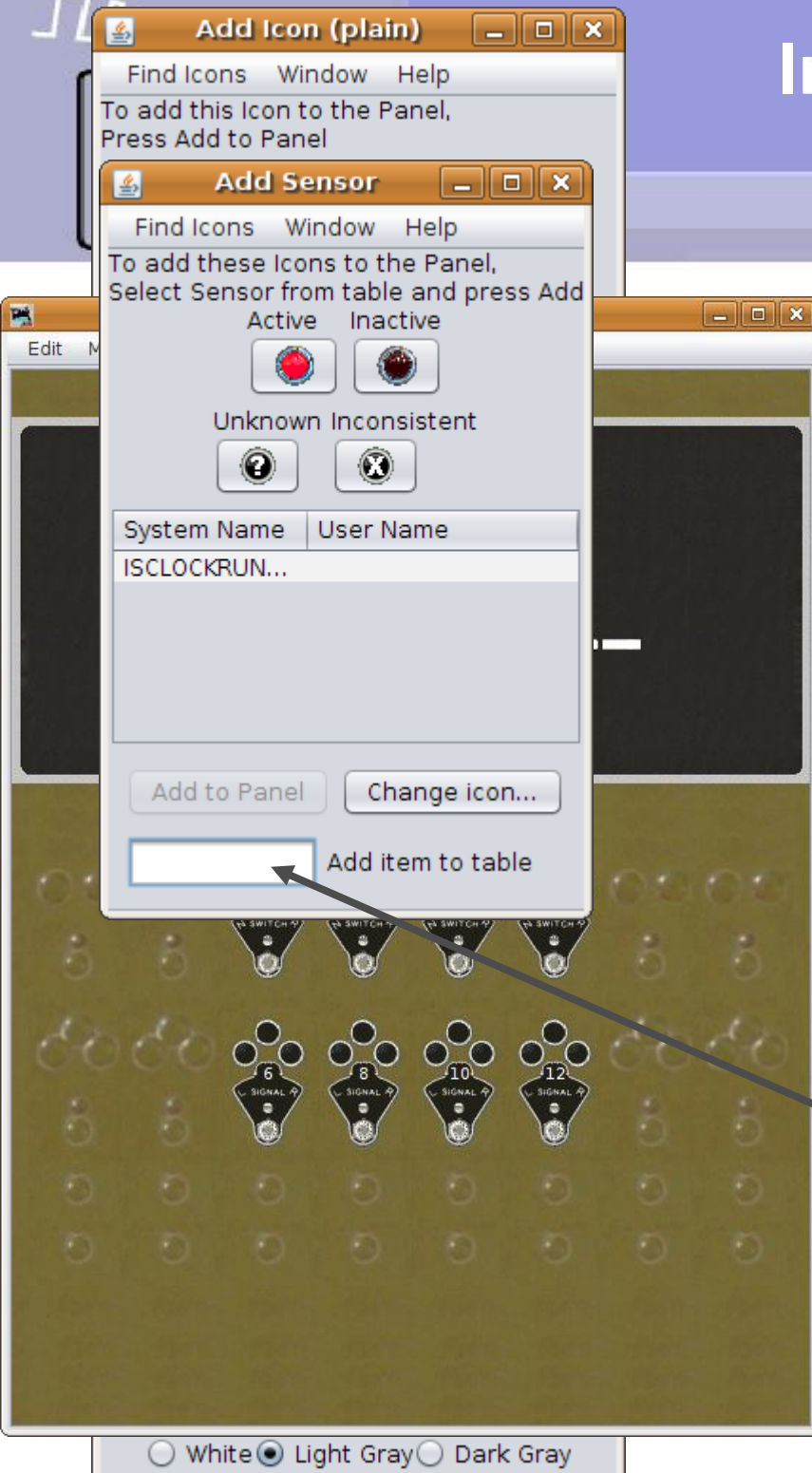


Indirect Layout Control

Sensor images

Indirect Layout Control

- Select the images you want to use for your OS detection. The USS default for a panel was red jewels for OS, and white jewels for blocks. Many railroads had their own standards including all white, all red, all blue, etc.
- To close the selection catalog, and open a list of existing sensors simply click on 'Close Catalog'.
- In clinic 2 we added active icons for our turnouts. Now we are doing the same for our occupancy sensors.

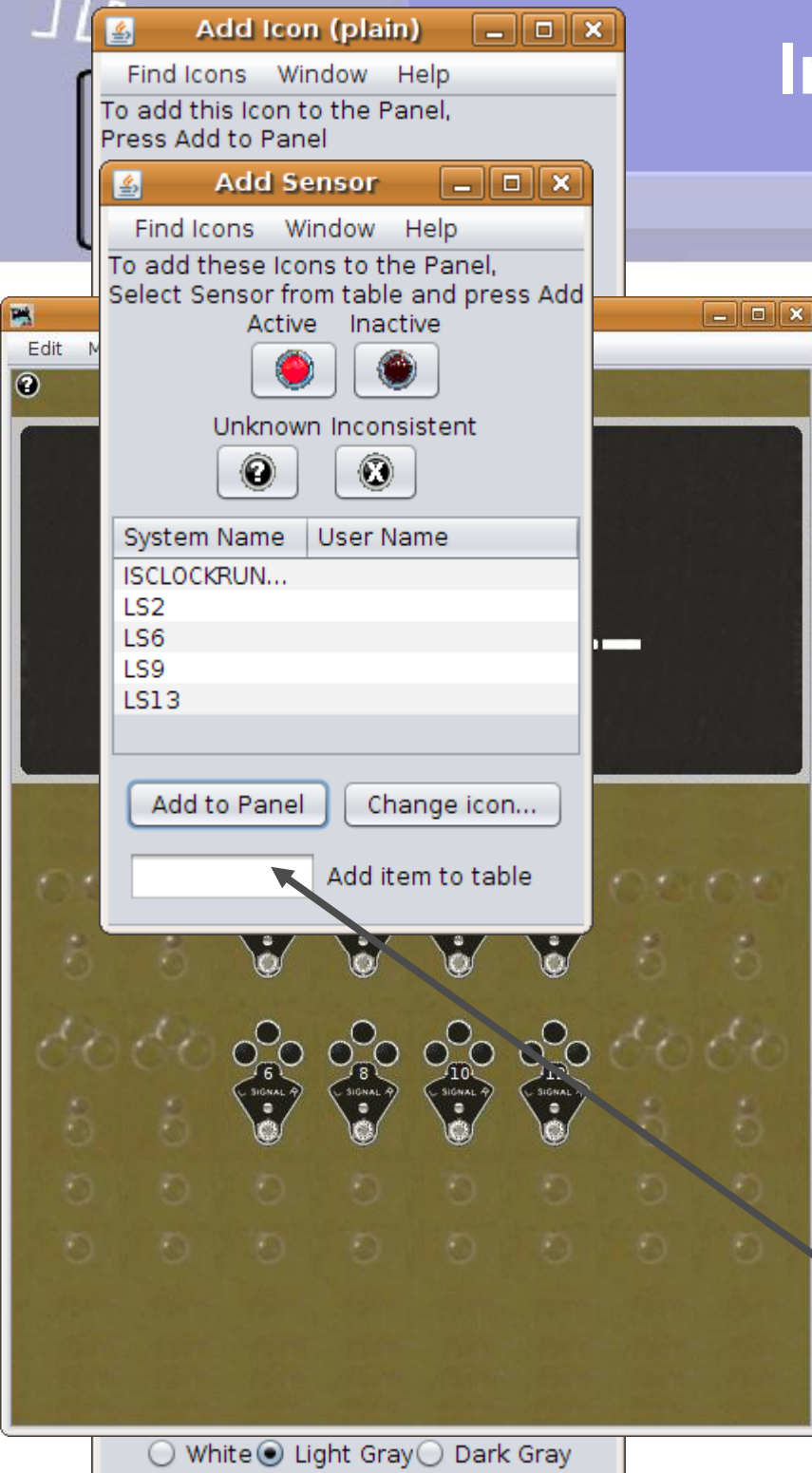


Indirect Layout Control

Sensor images

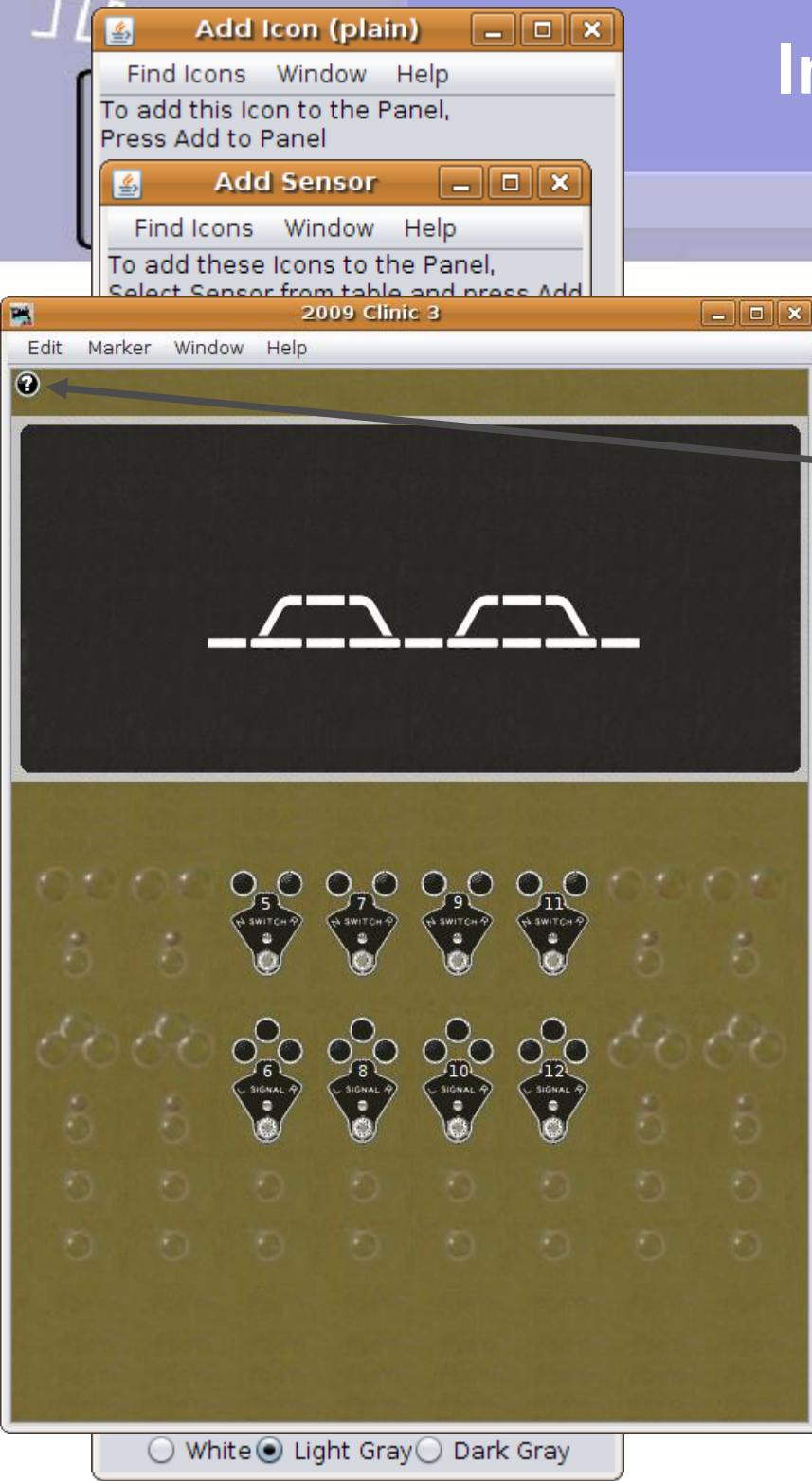
Indirect Layout Control

- Select the images you want to use for your OS detection. The USS default for a panel was red jewels for OS, and white jewels for blocks. Many railroads had their own standards including all white, all red, all blue, etc.
- To close the selection catalog, and open a list of existing sensors simply click on 'Close Catalog'.
- In clinic 2 we added active icons for our turnouts. Now we are doing the same for our occupancy sensors.
- Add sensors LS2, LS6, LS9, and LS13. (LS = LocoNet Sensor)



Indirect Layout Control

Sensor images

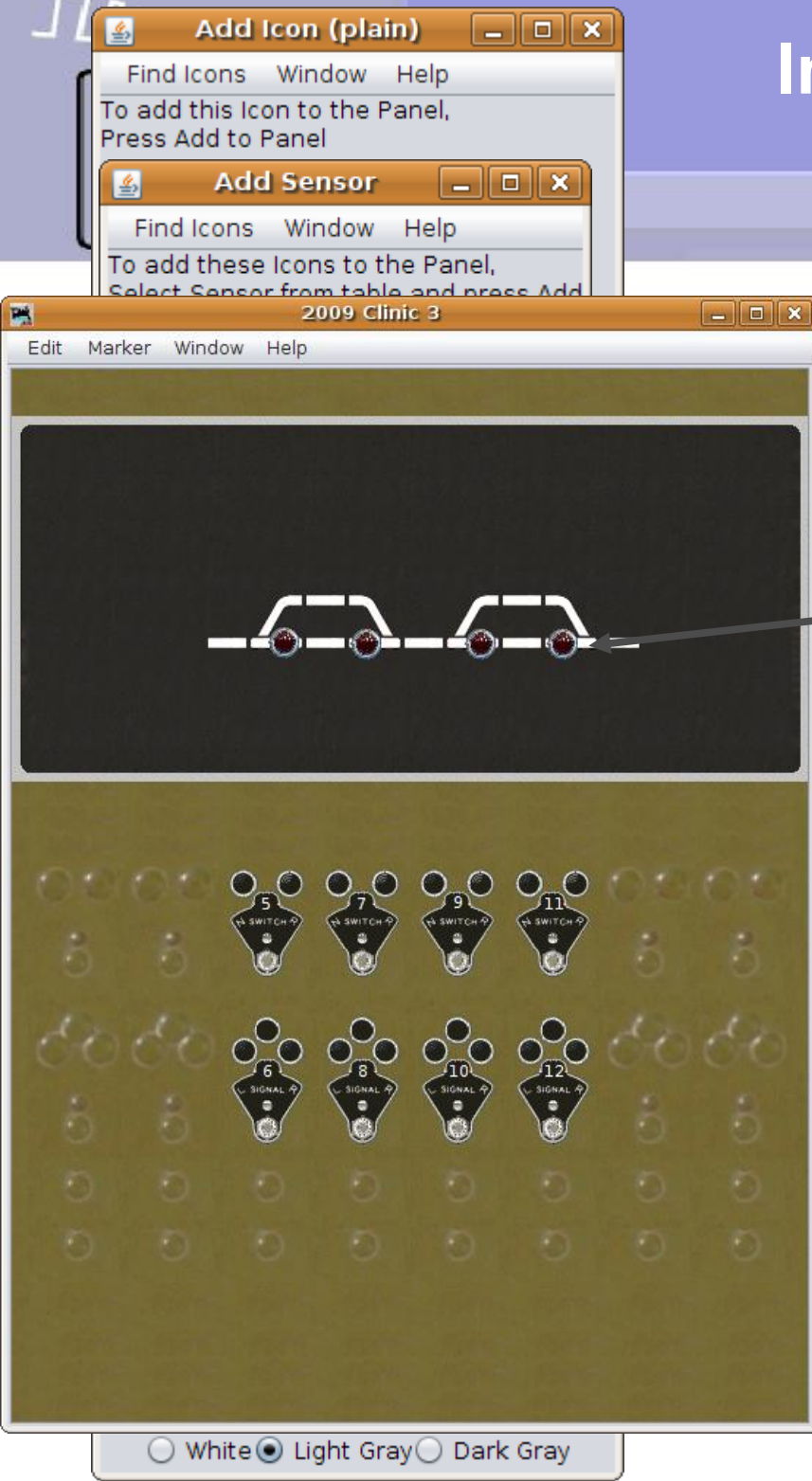


Indirect Layout Control

- We find our sensors all piled up in the usual place.

Indirect Layout Control

Sensor images

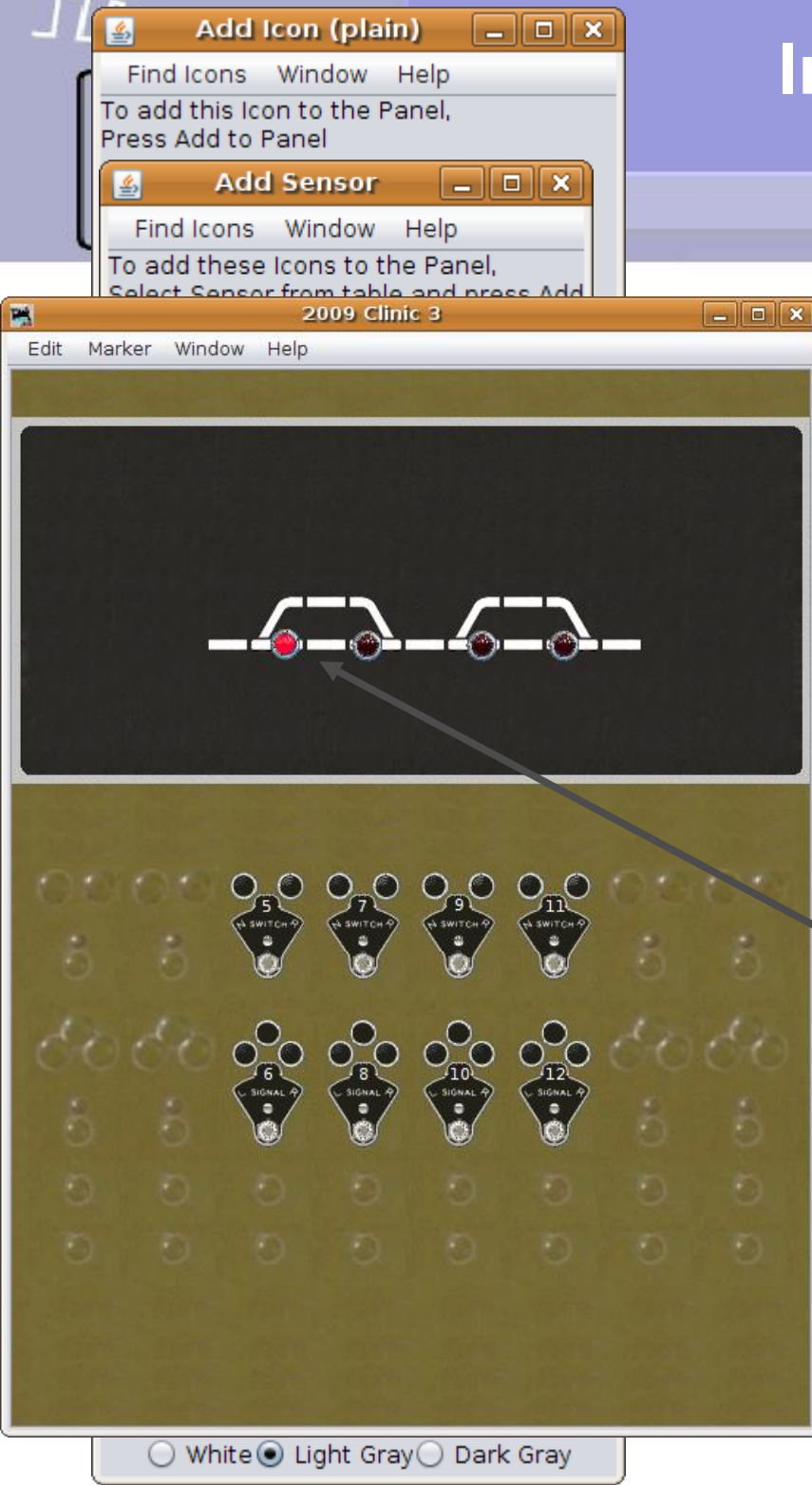


Indirect Layout Control

- We find our sensors all piled up in the usual place.
- Move them onto the panel.

Indirect Layout Control

Sensor images



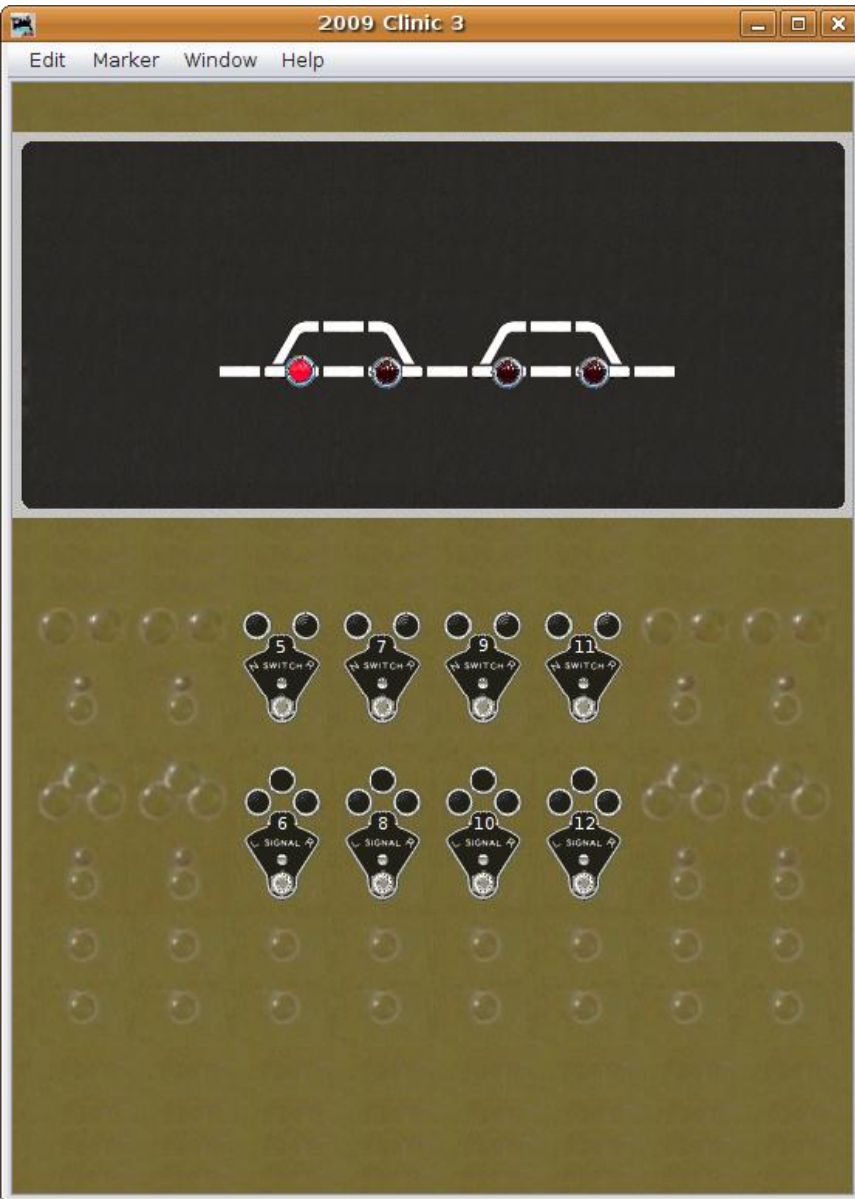
Indirect Layout Control

- We find our sensors all piled up in the usual place.
- Move them onto the panel.
- Normally we would 'disable' the sensor images so that they would only respond to our occupancy detectors. However we don't actually have any sensors attached, so we will simulate detection by clicking on our images to activate them.



Internal sensors

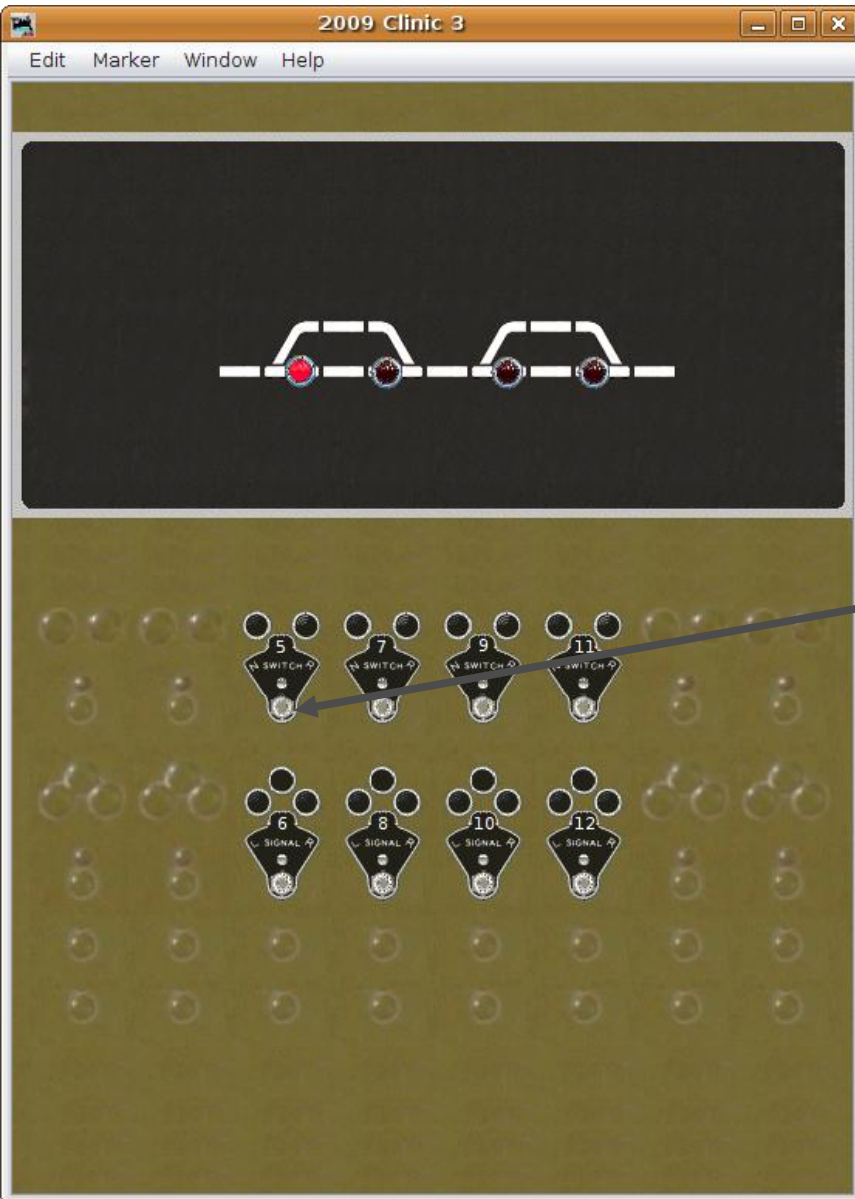
- Our next concept is that of 'Internal' sensors. These are really just single bit memory devices. They react with the images just as if they were hardware, but only exist internally to JMRI.






Internal sensors

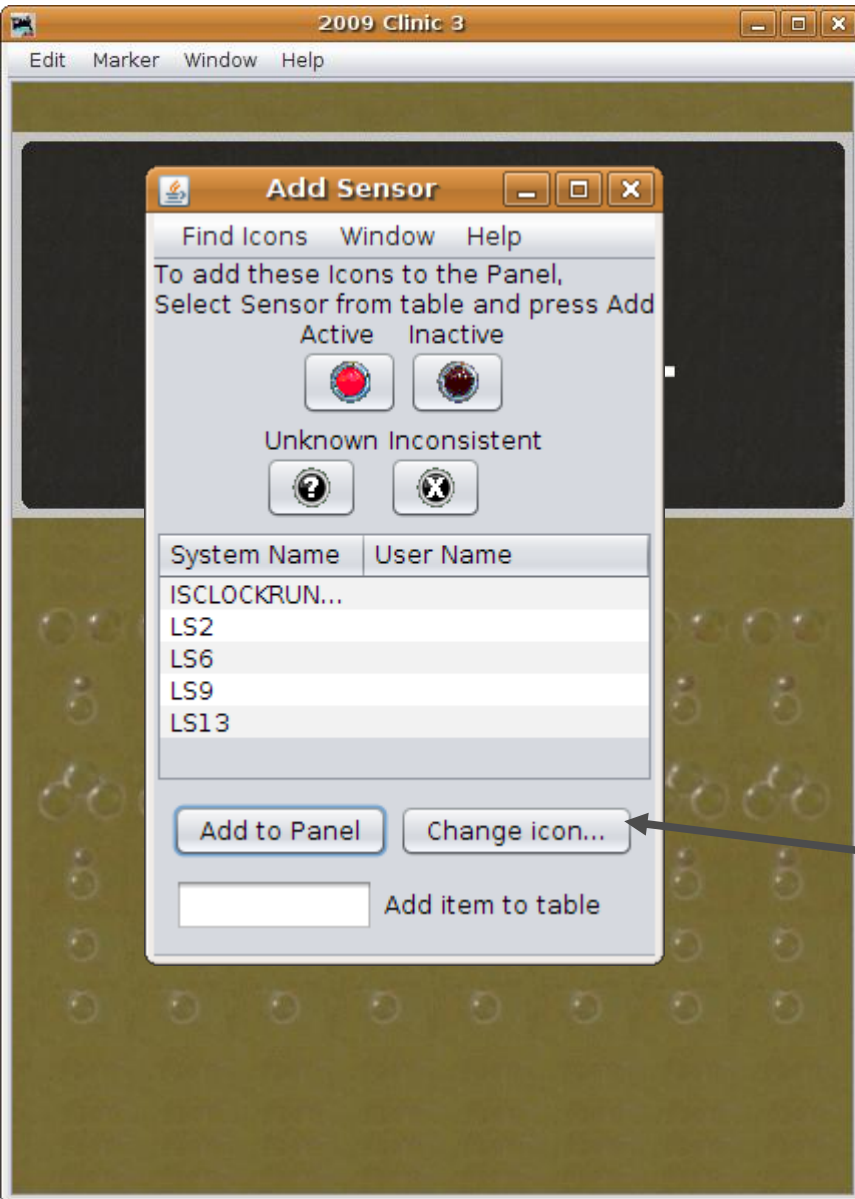
- Our next concept is that of 'Internal' sensors. These are really just single bit memory devices. They react with the images just as if they were hardware, but only exist internally to JMRI.
- We need some new levers that are not directly attached to the turnouts like we had them in the second clinic.





Internal sensors

- Our next concept is that of 'Internal' sensors. These are really just single bit memory devices. They react with the images just as if they were hardware, but only exist internally to JMRI.
- We need some new levers that are not directly attached to the turnouts like we had them in the second clinic.
- Pull up the 'Add Sensor' window again and open the 'Change icon..' 

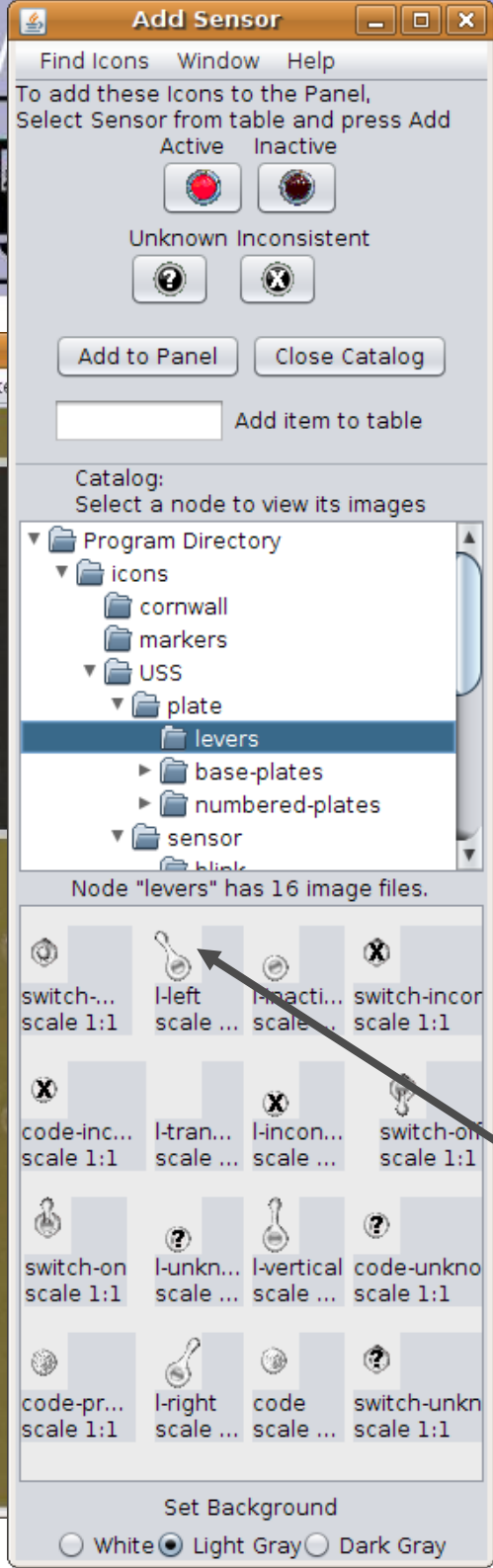


Indirect Layout Control

Sensor images

Internal sensors

- Our next concept is that of 'Internal' sensors. These are really just single bit memory devices. They react with the images just as if they were hardware, but only exist internally to JMRI.
- We need some new levers that are not directly attached to the turnouts like we had them in the second clinic.
- Pull up the 'Add Sensor' window again and open the 'Change icon..''
- Navigate back to the 'levers', but this time it will be sensors that have the lever images.

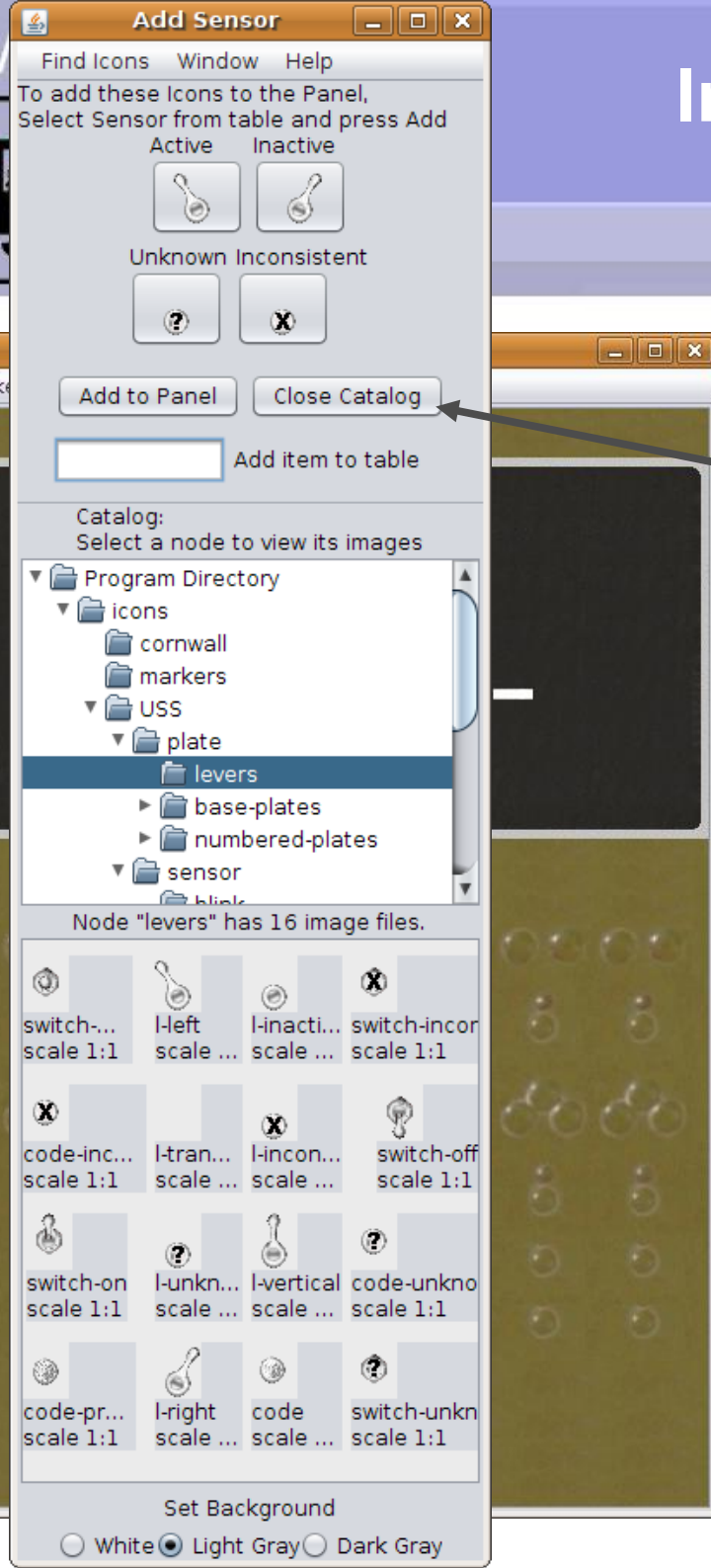


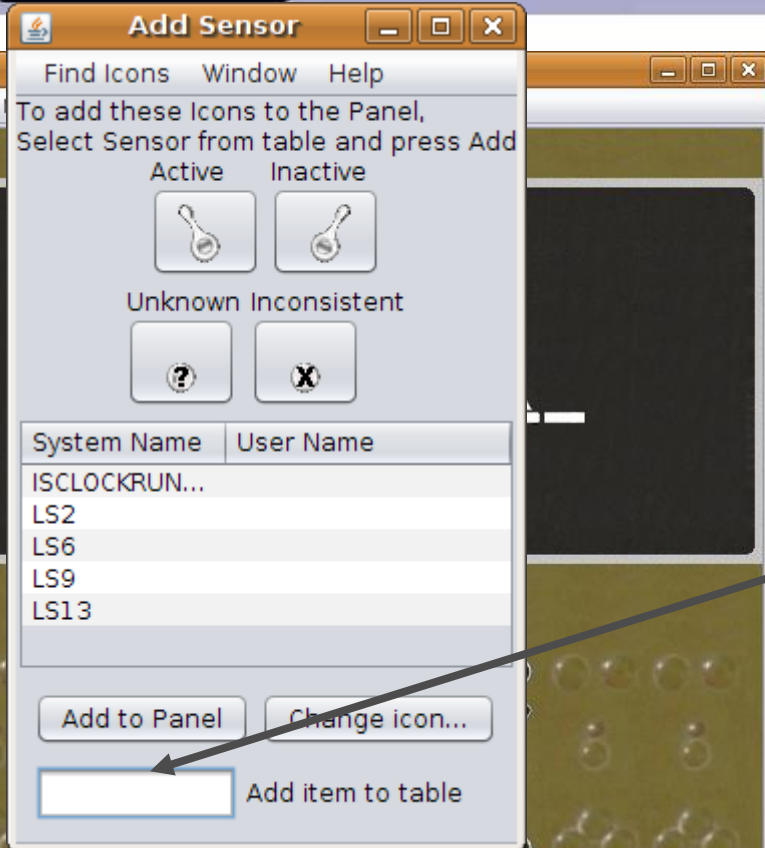
Indirect Layout Control

Sensor images

Internal sensors

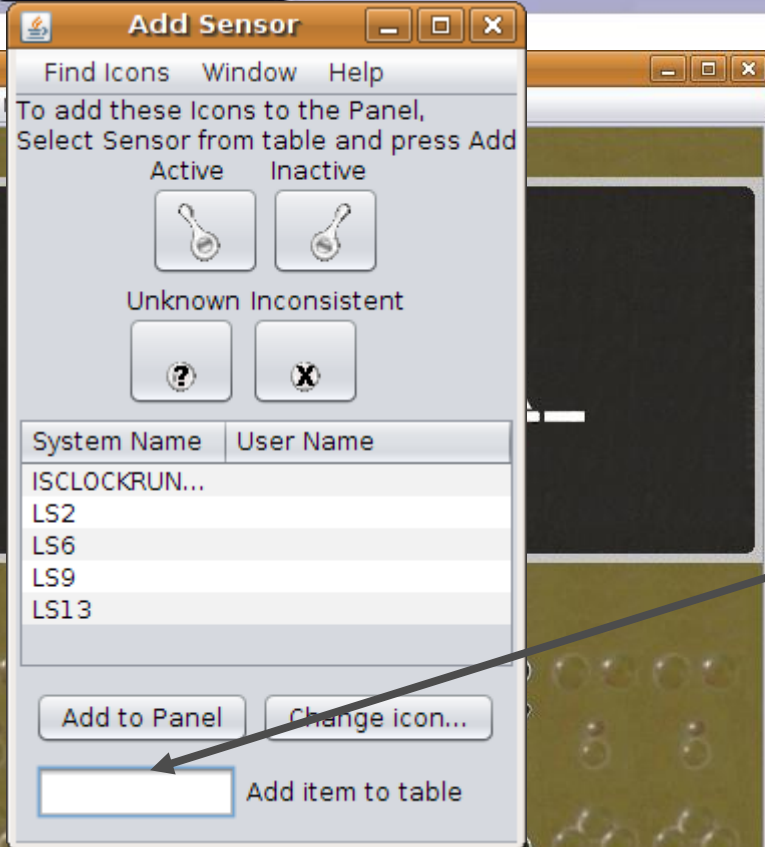
- Once the lever images are selected click on 'Close Catalog' to reduce the window size and get our sensor list.





Internal sensors

- Once the lever images are selected click on 'Close Catalog' the reduce the window size and get our sensor list.
- At this point we could call our internal sensor just about anything, even 'IS-late-to-lunch'. (IS denotes **I**nternal **S**ensor).



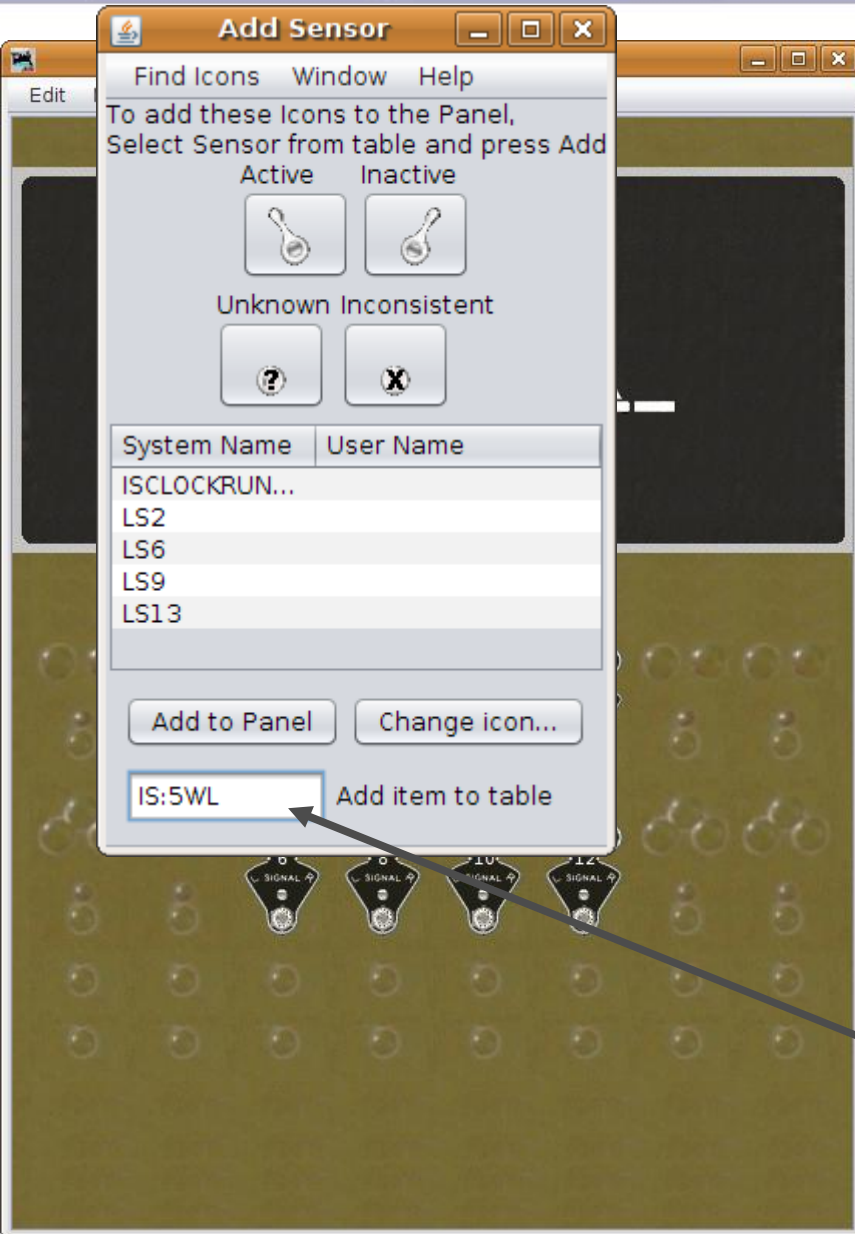
Internal sensors

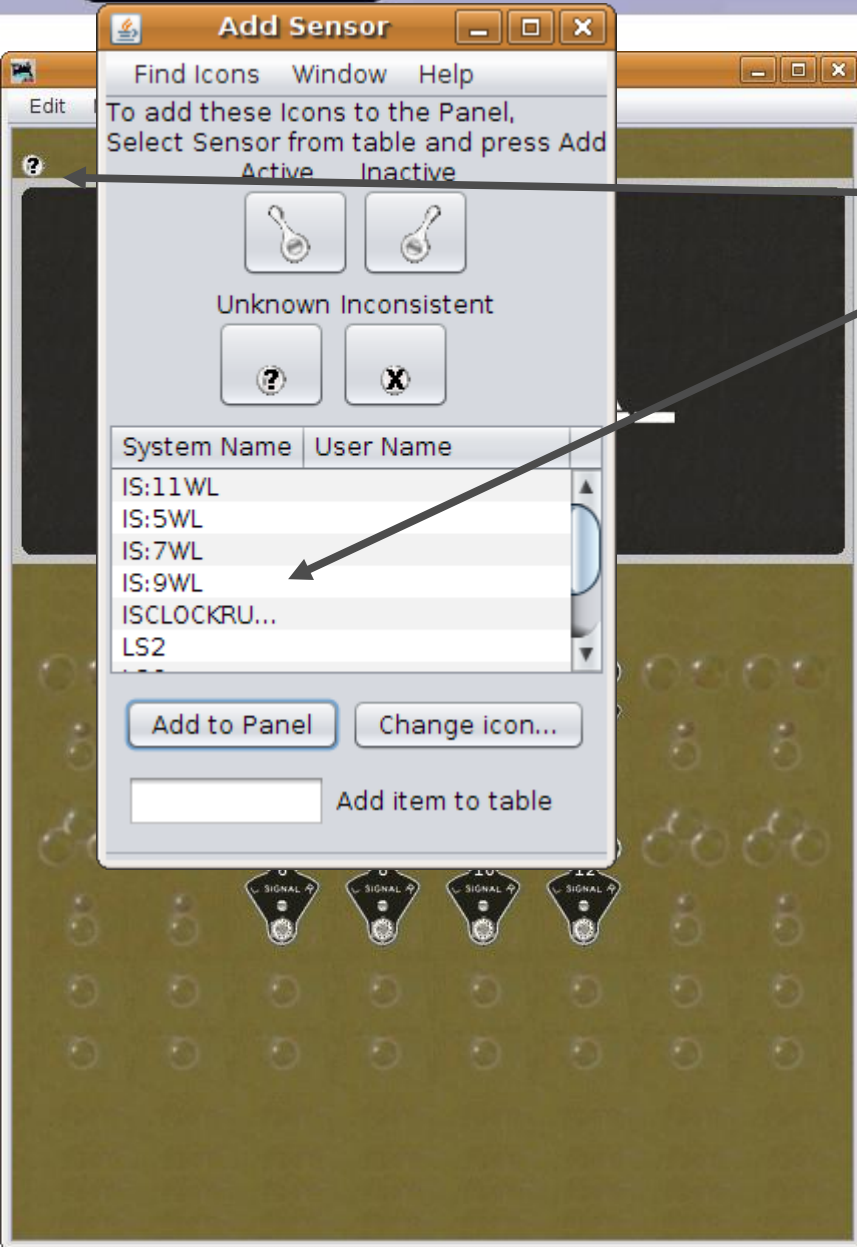
- Once the lever images are selected click on 'Close Catalog' the reduce the window size and get our sensor list.
- At this point we could call our internal sensor just about anything, even 'IS-late-to-lunch'. (IS denotes **I**nternal **S**ensor).
- The prototype railroads have a similar naming problem, so hopefully we can learn something from them.
- AREMA = American Railway Engineering & MOW Association.



Internal Sensor Names

- AREMA - (Numerical Prefix)(First Letter)..(Last Letter)
 - Numerical Prefix: The number of the lever, signal, track circuit, etc.
 - First Letter: General kind of unit.
 - Last Letter: Specific unit
- For Logix
 - ISn: = Internal Sensor n
 - W = sWitch
 - L = Lever
- Result = IS5:WL (plus IS7:WL, IS9:WL, and IS11:WL)





Internal Sensors

- Our new internal sensors have been added to both the sensor table and the panel.

Indirect Layout Control

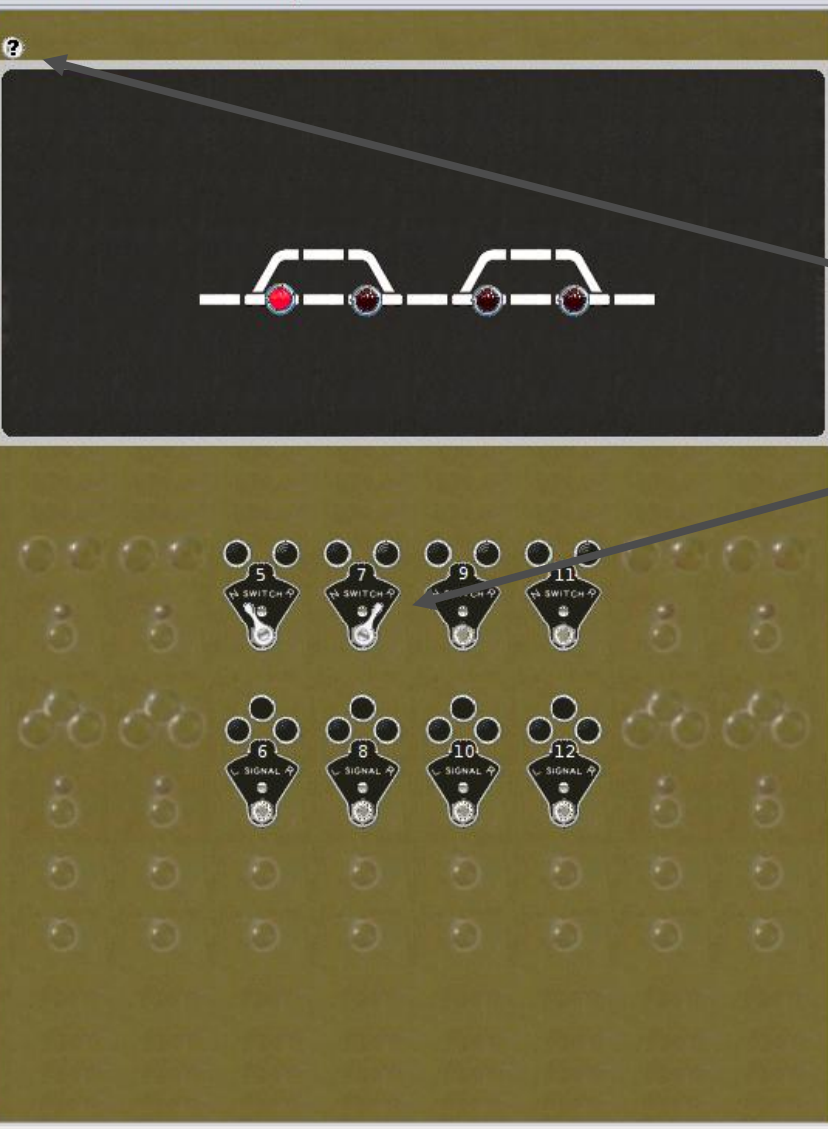
Sensor images



Add Sensor

2009 Clinic 3

Edit Marker Window Help

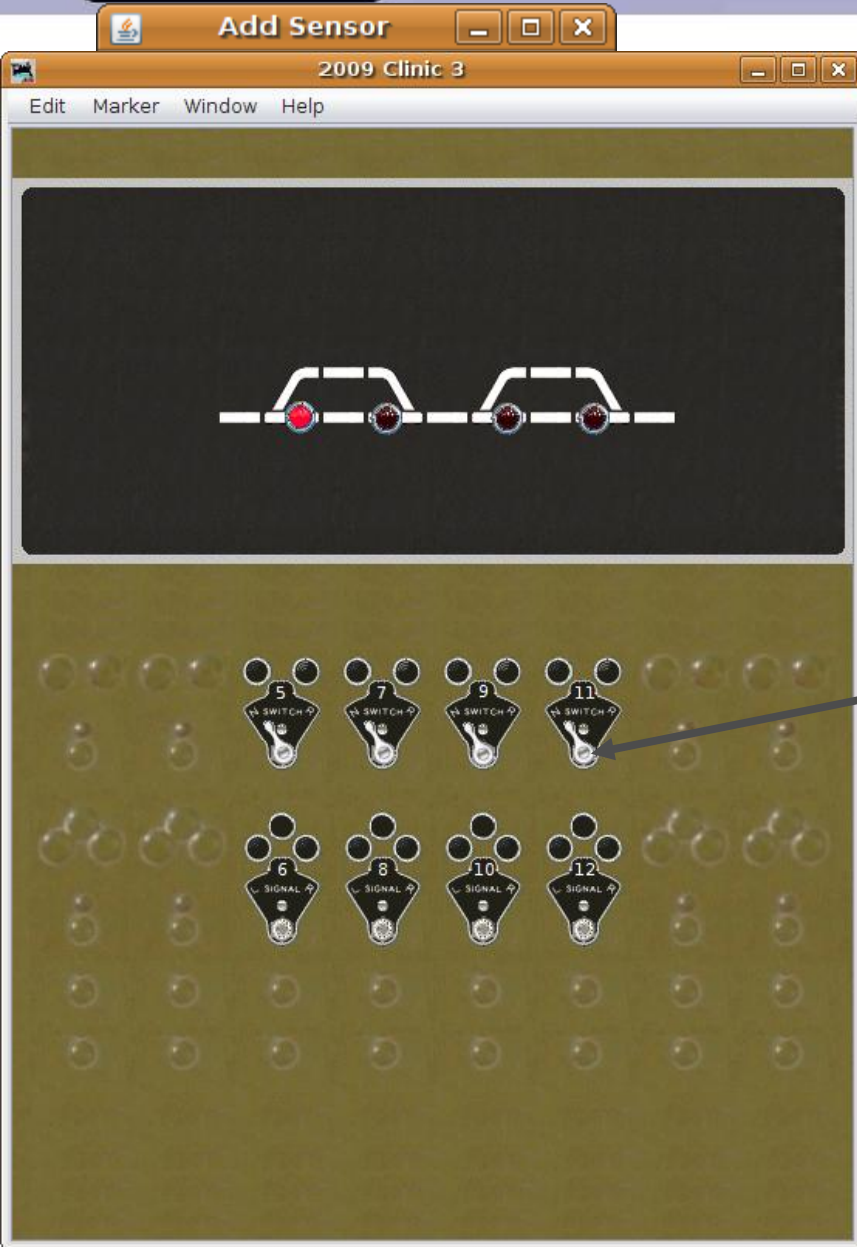


Internal Sensors

- Our new internal sensors have been added to both the sensor table and the panel.
- Move them into place.

Indirect Layout Control

Sensor images



Internal Sensors

- Our new internal sensors have been added to both the sensor table and the panel.
- Move them into place.
- Now we have some levers that are not directly connected to the layout. We can flip them simply by clicking on them.

Indirect Layout Control

Sensor images



Add Sensor

2009 Clinic 3

Edit Marker Window Help



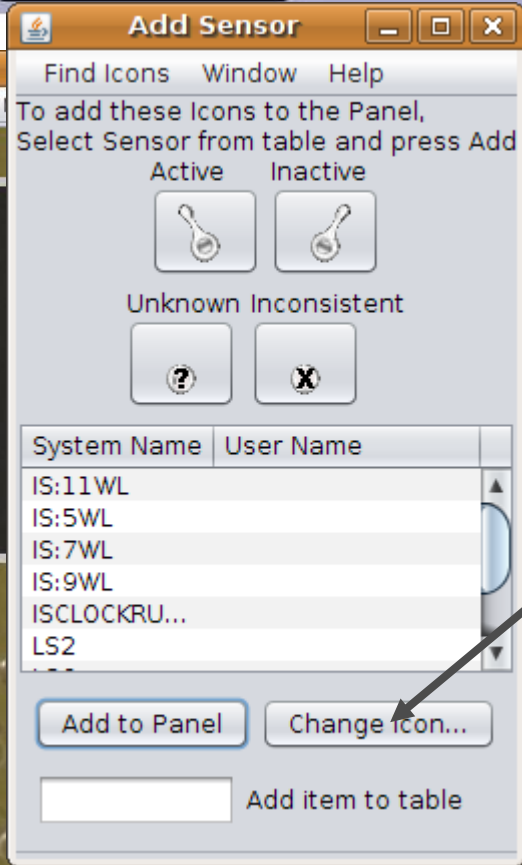
Internal Sensors

- Our new internal sensors have been added to both the sensor table and the panel.
- Move them into place.
- Now we have some levers that are not directly connected to the layout. We can flip them simply by clicking on them.
- A prototype CTC panel also did not directly connect its levers to the switch motors. The operator moved a lever and then pressed a 'Send Code' button that encoded and sent the commands over the track side wires in a serial format that used short and long pulses. (similar to DCC)



Internal Sensors

- To add code buttons re-call our 'Change Icon' list in the 'Add Sensor' window.

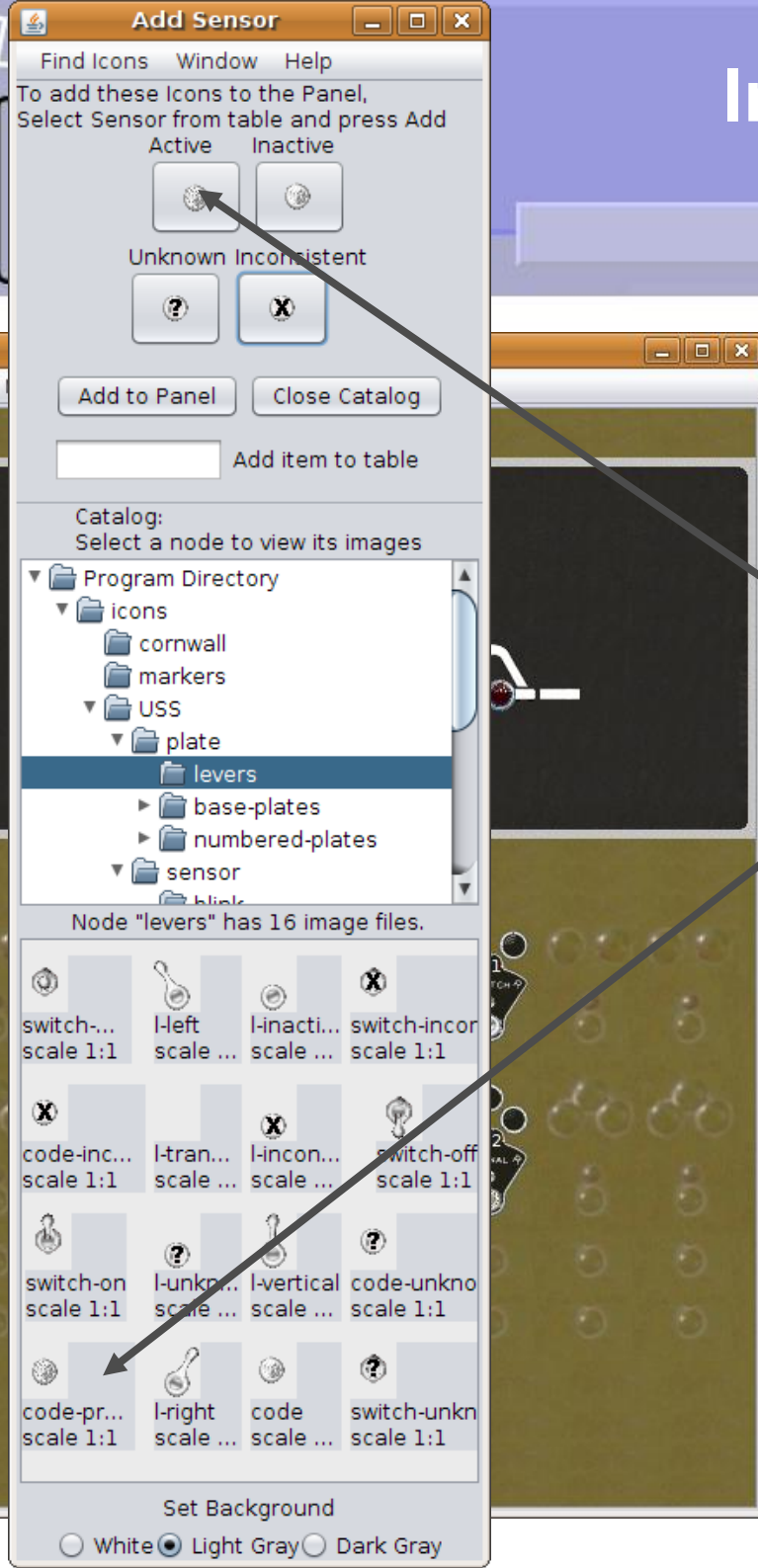


Indirect Layout Control

Sensor images

Internal Sensors

- To add code buttons re-call our 'Change Icon' list in the 'Add Sensor' window.
- The 'code-press' icon is the 'Active' entry, the 'code' icon is the 'Inactive'.

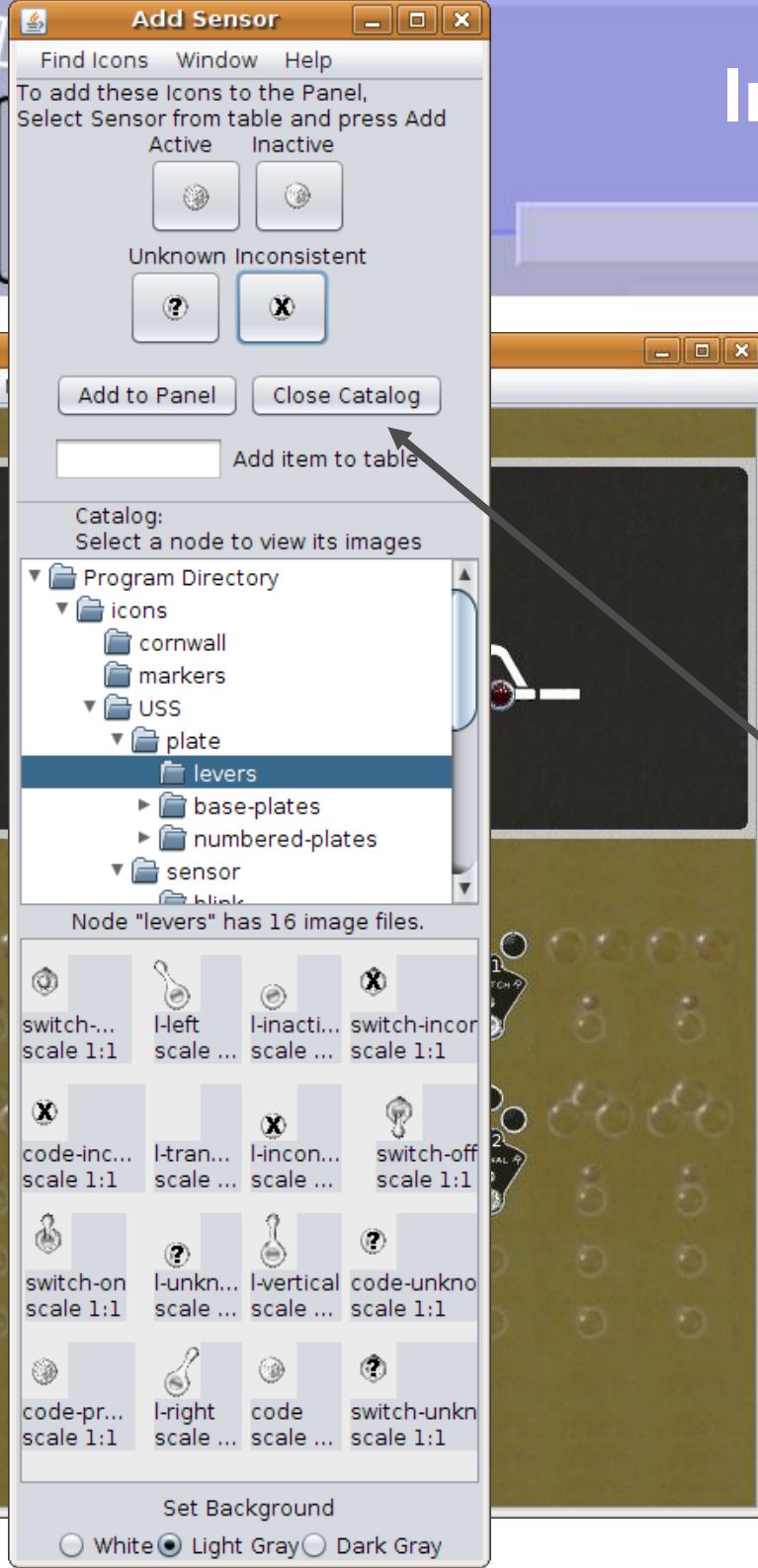


Indirect Layout Control

Sensor images

Internal Sensors

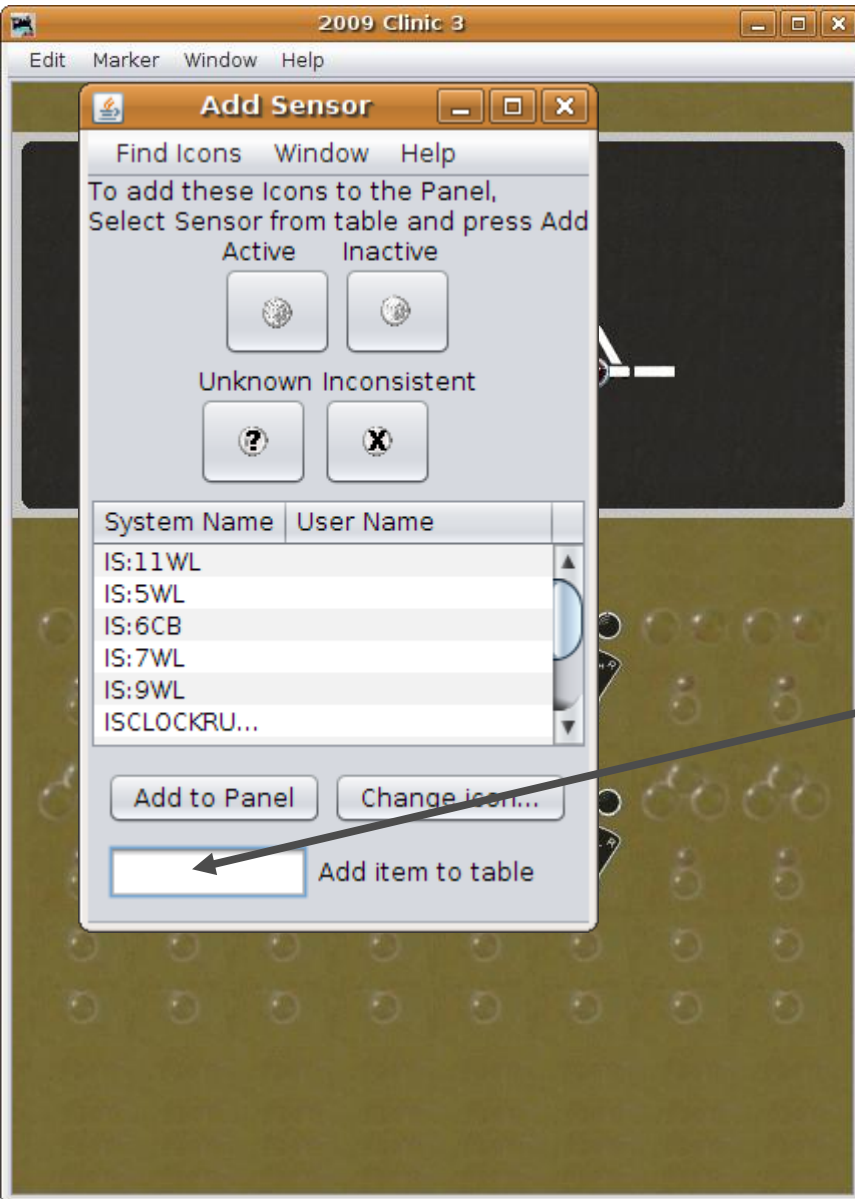
- To add code buttons re-call our 'Change Icon' list in the 'Add Sensor' window.
- The 'code-press' icon is the 'Active' entry, the 'code' icon is the 'Inactive'.
- Close Catalog brings us back to our sensor list.





Internal Sensors

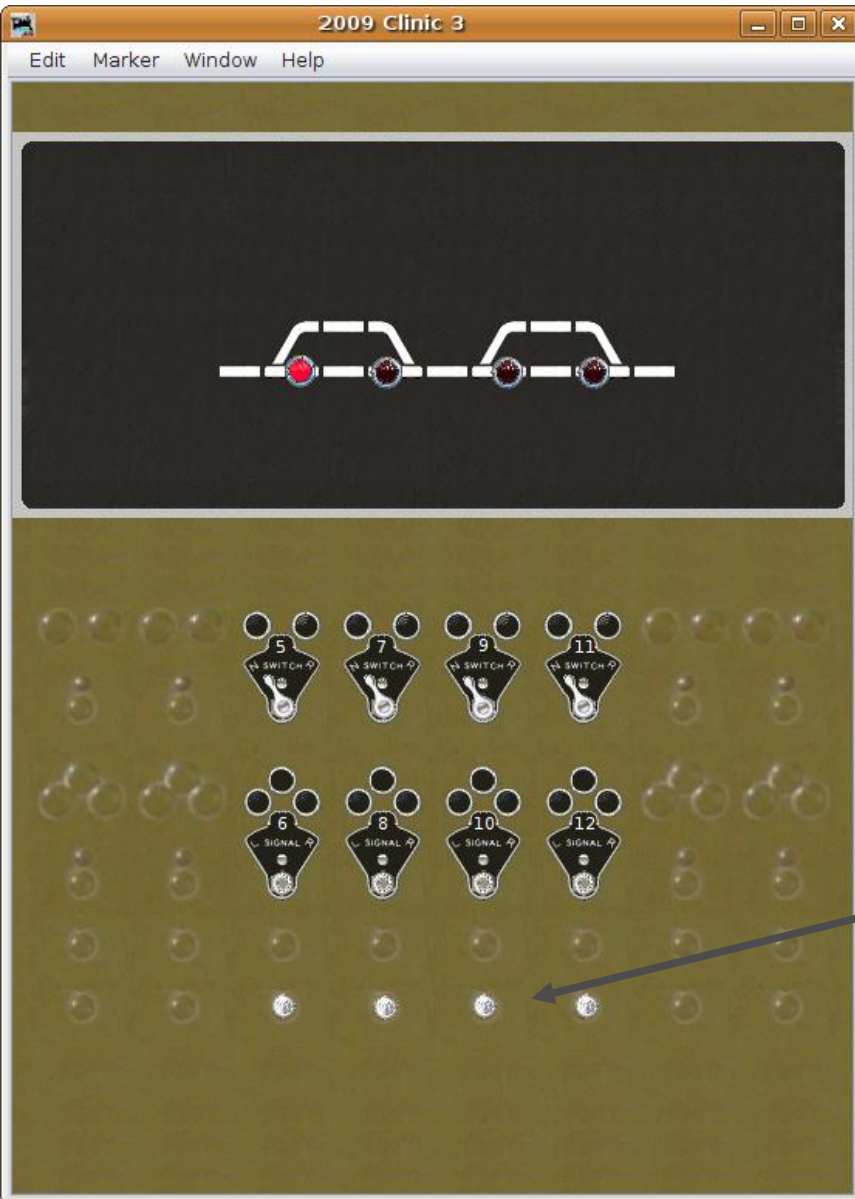
- To add code buttons re-call our 'Change Icon' list in the 'Add Sensor' window.
- The 'code-press' icon is the 'Active' entry, the 'code' icon is the 'Inactive'.
- Close Catalog brings us back to our sensor list.
- Enter IS6:CB, IS8:CB, IS10:CB, and IS12:CB for our sensors.

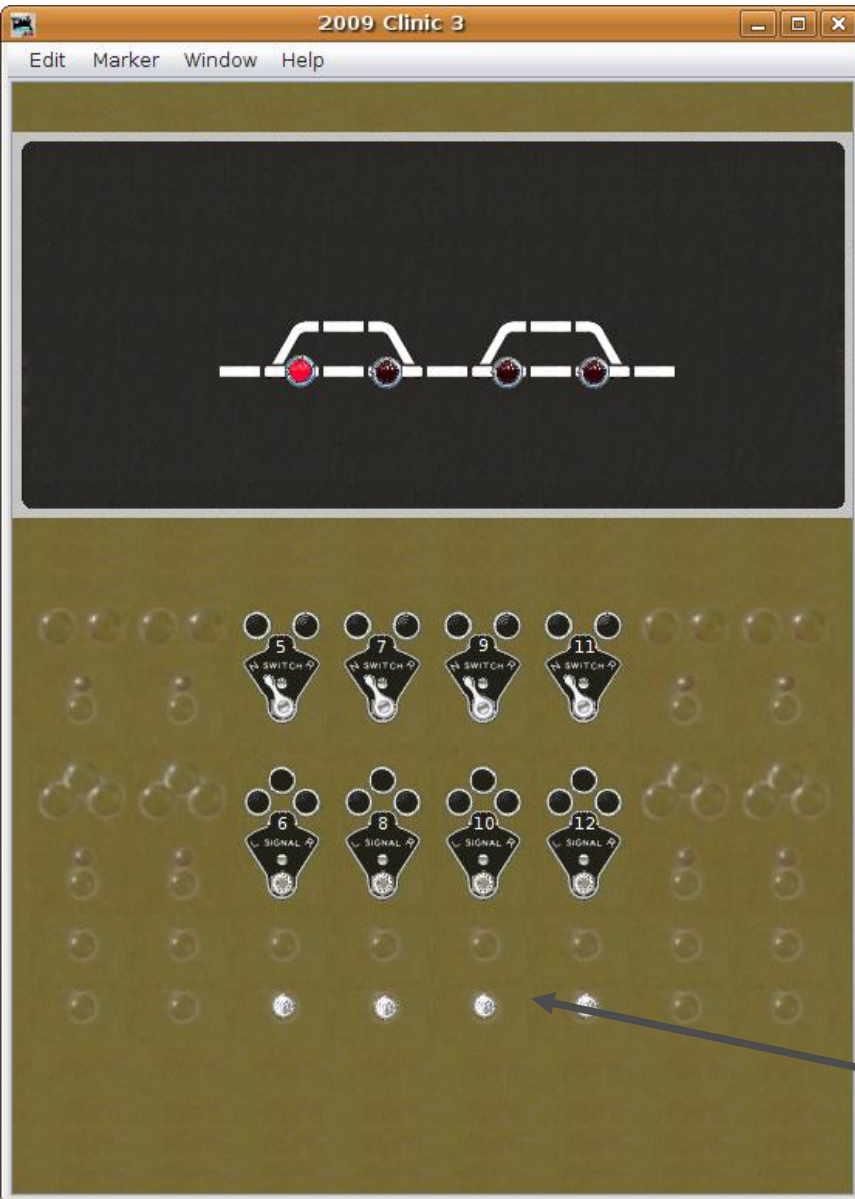




Internal Sensors

- To add code buttons re-call our 'Change Icon' list in the 'Add Sensor' window.
- The 'code-press' icon is the 'Active' entry, the 'code' icon is the 'Inactive'.
- Close Catalog brings us back to our sensor list.
- Enter IS6:CB, IS8:CB, IS10:CB, and IS12:CB for our sensors.
- Move them into place.





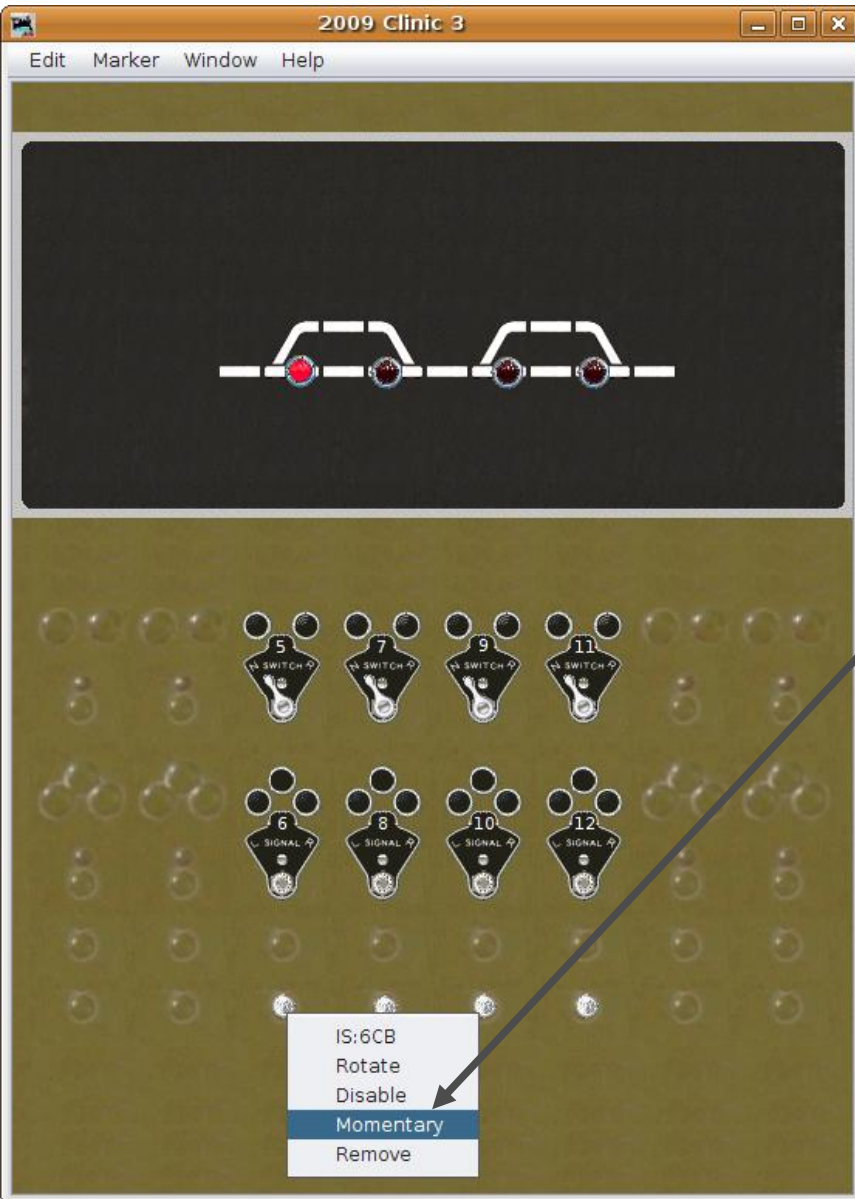
Internal Sensors

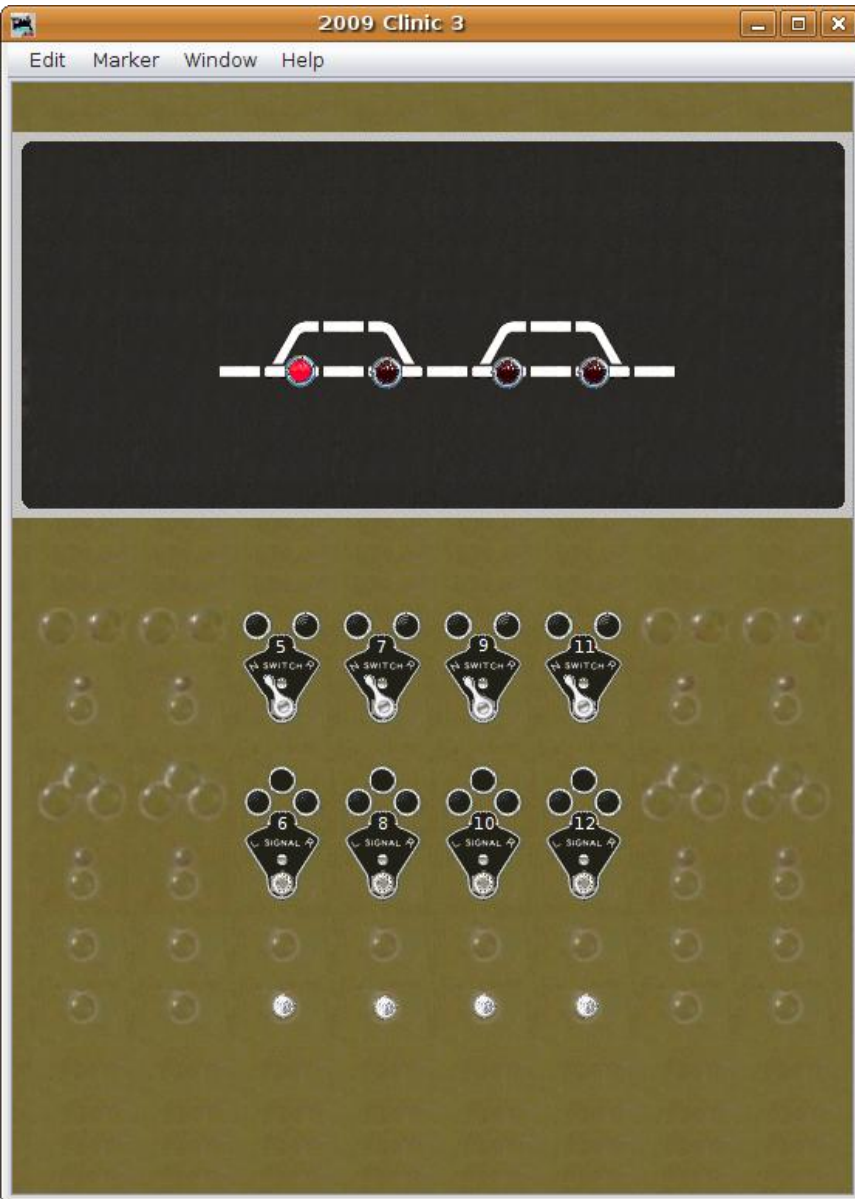
- To add code buttons re-call our 'Change Icon' list in the 'Add Sensor' window.
- The 'code-press' icon is the 'Active' entry, the 'code' icon is the 'Inactive'.
- Close Catalog brings us back to our sensor list.
- Enter IS6:CB, IS8:CB, IS10:CB, and IS12:CB for our sensors.
- Move them into place.
- Note, when we click them they go down, and when we click them again they go up. This is not very "buttonlike" behavior.



Internal Sensors

- To correct the behavior, right click on each button icon and click 'Momentary' to check it.





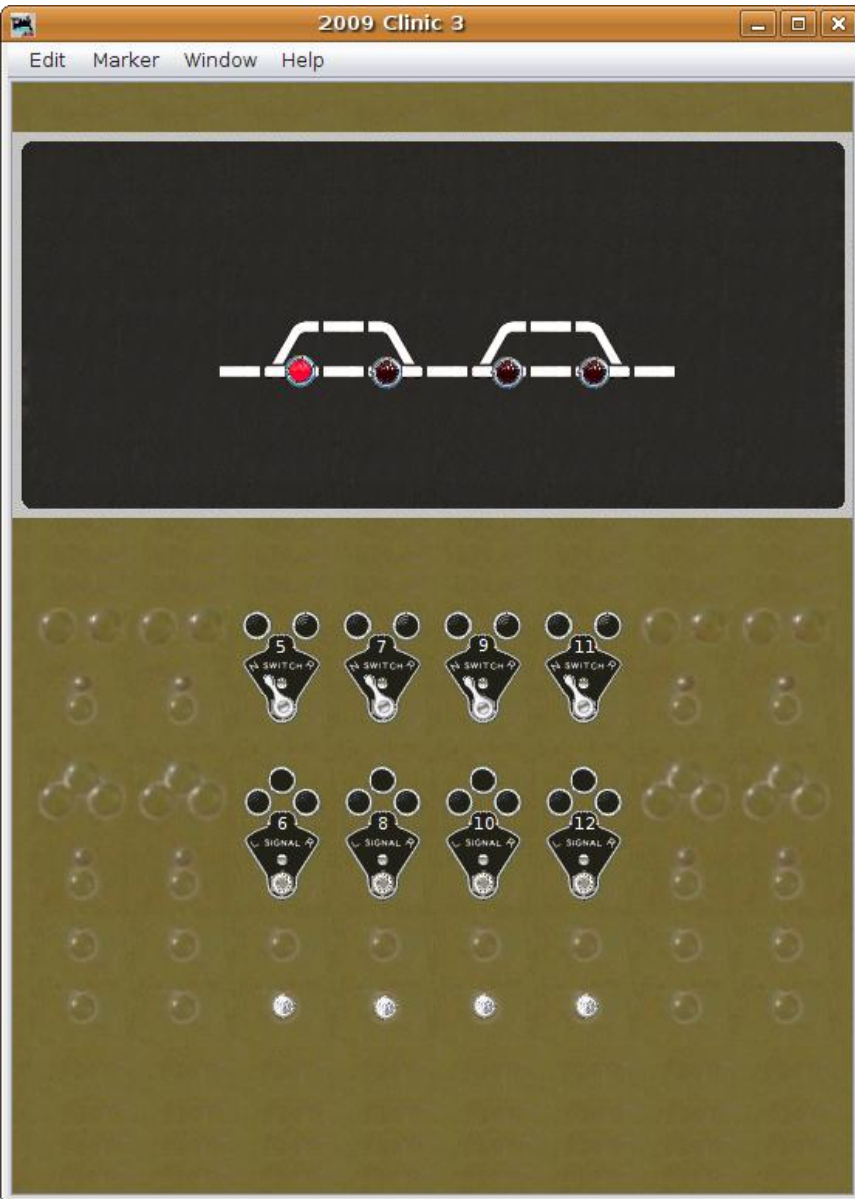
Internal sensor names

- When naming our new buttons we already mentioned that we are not attached to hardware, so any name is allowed. I am loosely basing these names on the **AREMA** nomenclature. I chose '**ISn:**' to begin them all because we need the "**IS**", and normally system generated names use the ":". The plan is that a tool will eventually generate these names, so I include the ":".

There is a fair amount of repetition in the AREMA names, but an attempt is made to not place different meanings in one position, nor use the same names in different positions. E.g R may be red, reverse, right, relay, etc.

Indirect Layout Control

Sensor names



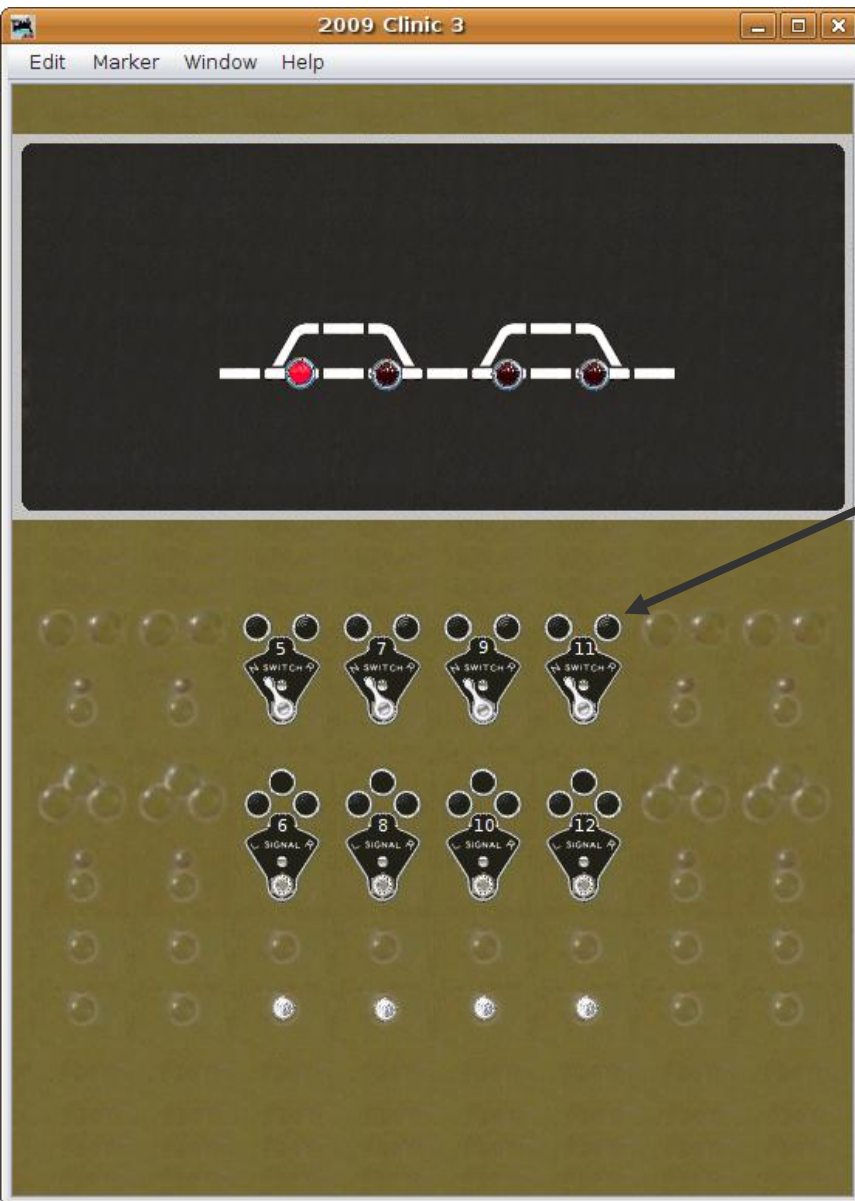
Internal sensor names

	Beginning	Middle	End
A	Alarm	Track A	
B		Track B	Button
C		Code	Controller
D		proceed	
F	Fleeting		
G		siGnal	
K			indiKtor
L	Left		Lever
N	Normal		
R	Reverse	Red	Relay
T		Track	
W		sWitch	



Turnout Feedback

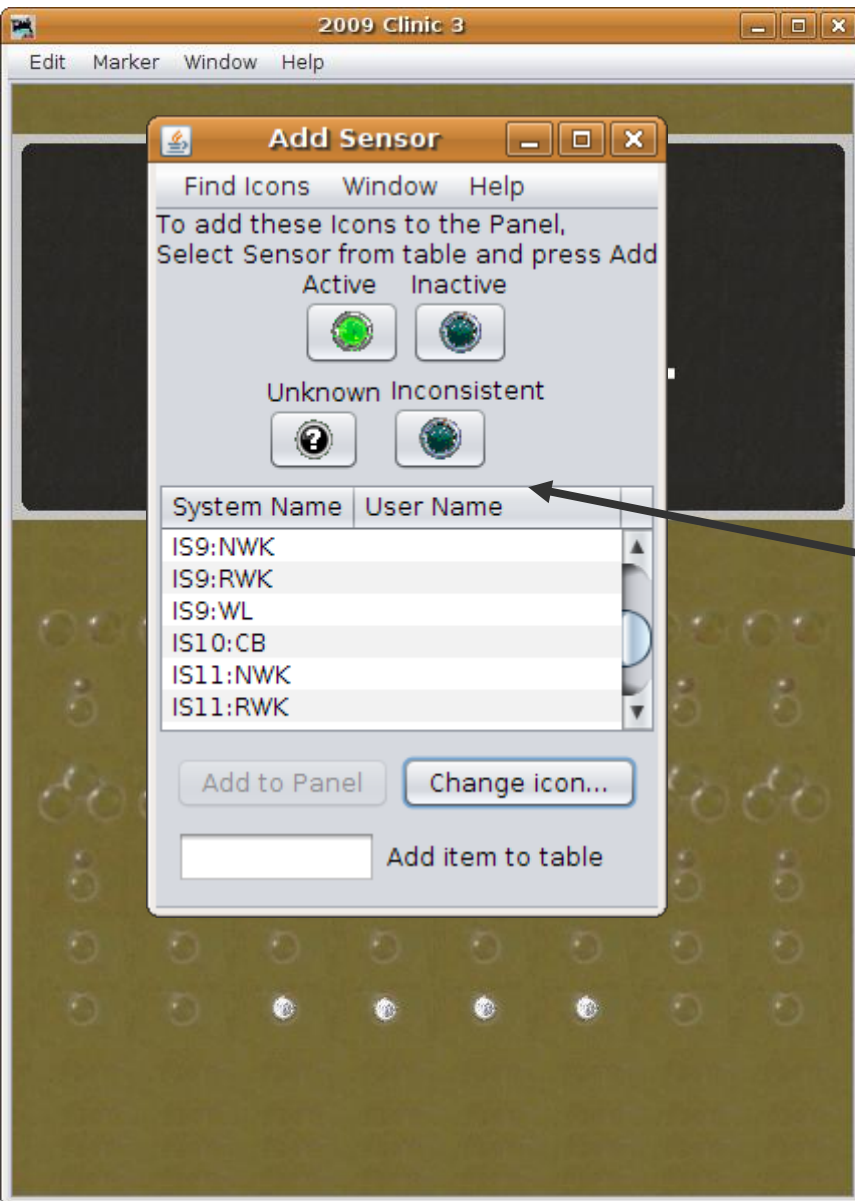
- We still need some way to tell which position the layout track switches are aligned. The levers and track image are not available, so we will use the indicator lamps. (just like the prototype)





Turnout Feedback

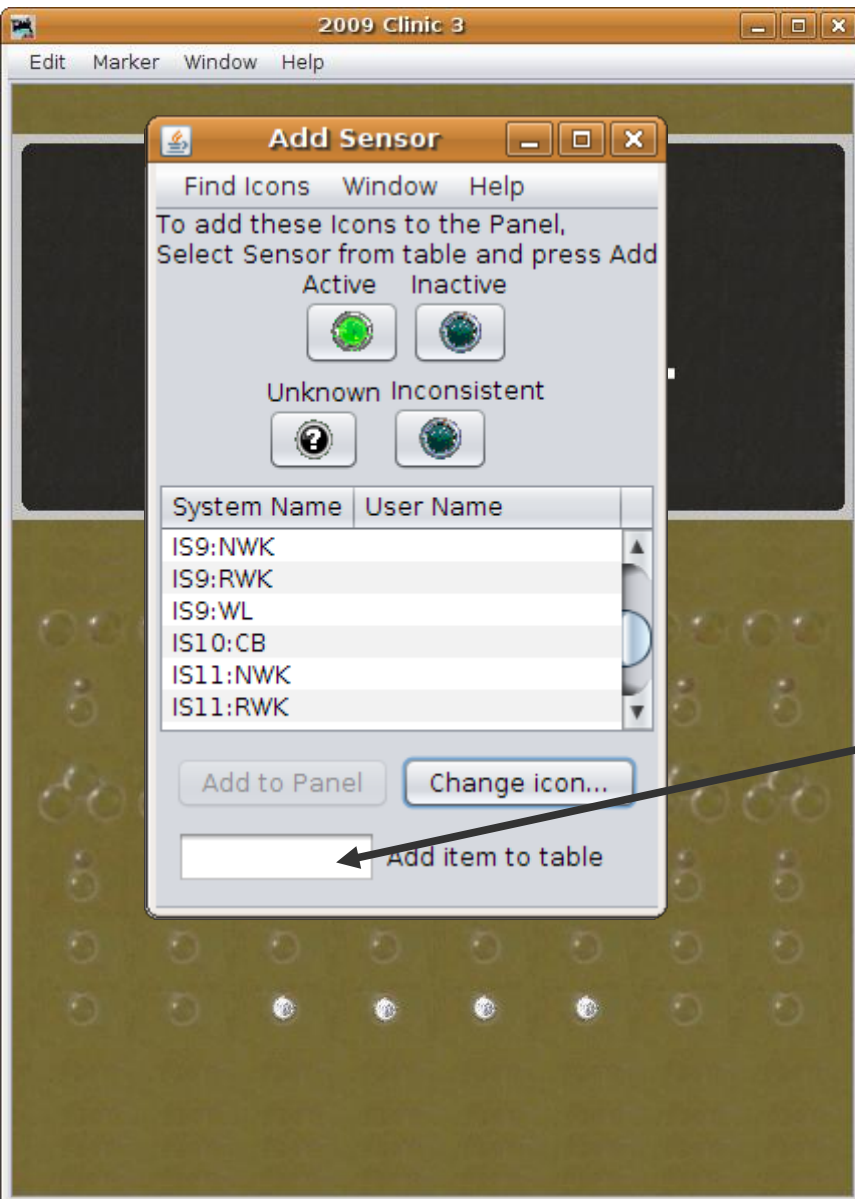
- We still need some way to tell which position the layout track switches are aligned. The levers and track image are not available, so we will use the indicator lamps. (just like the prototype)
- Pull up the 'Add Sensor' window and set it for green jewel icons. Use 'off' for 'inconsistent'.





Turnout Feedback

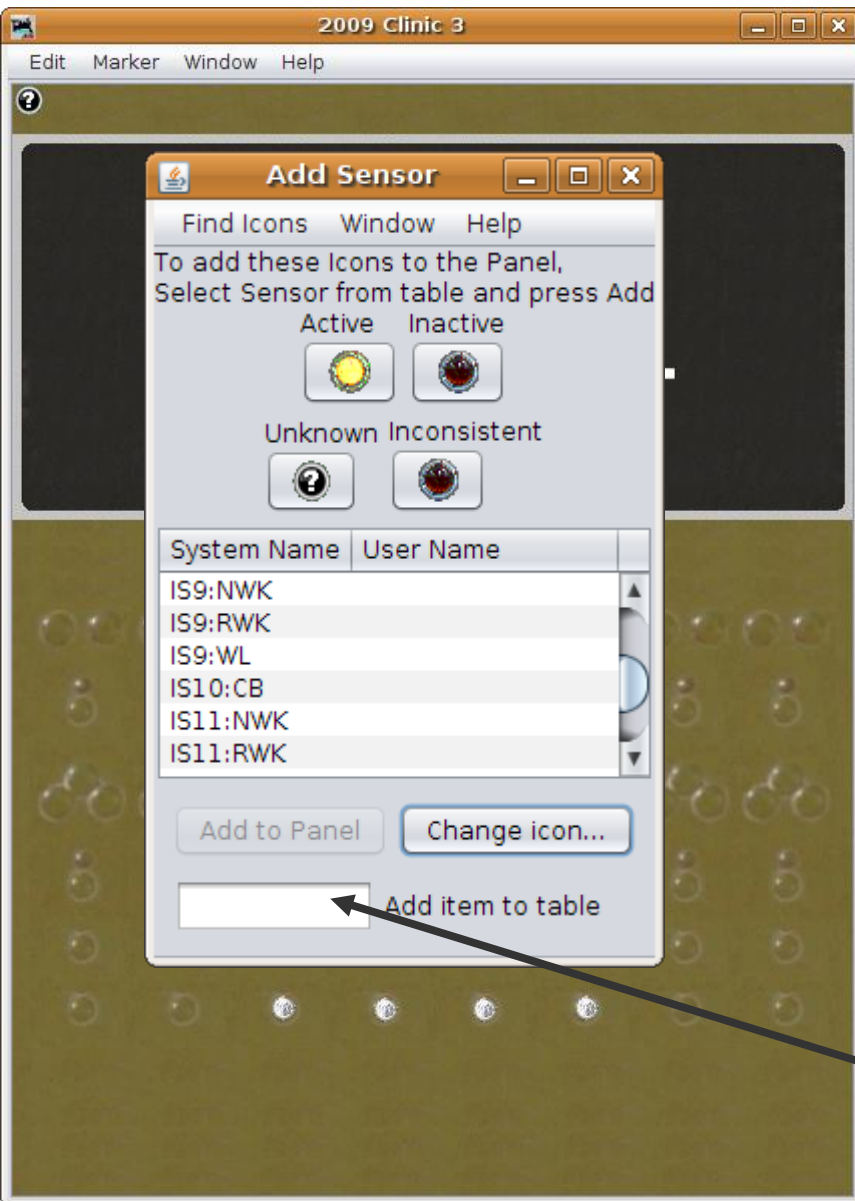
- We still need some way to tell which position the layout track switches are aligned. The levers and track image are not available, so we will use the indicator lamps. (just like the prototype)
- Pull up the 'Add Sensor' window and set it for green jewel icons. Use 'off' for 'inconsistent'.
- From our naming rules we see that the first lamp jewel name should be IS5:NWK. (N = Normal, W = sWitch, and K = indiKtor) followed by IS7:NWK, IS9:NWK, and IS11:NWK.





Turnout Feedback

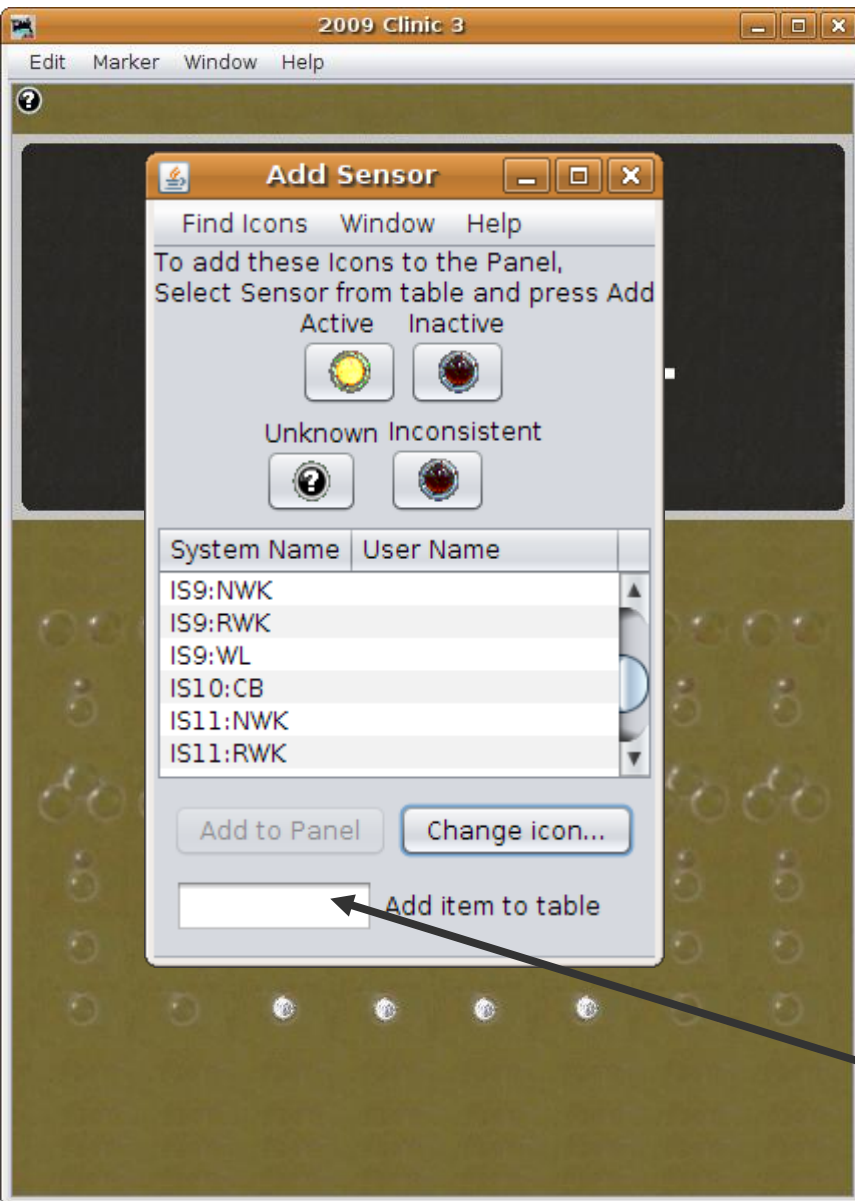
- We still need some way to tell which position the layout track switches are aligned. The levers and track image are not available, so we will use the indicator lamps. (just like the prototype)
- Pull up the 'Add Sensor' window and set it for green jewel icons.
- From our naming rules we see that the first lamp jewel name should be IS5:NWK. (N = Normal, W = sWitch, and K = indiKtor) followed by IS7:NWK, IS9:NWK, and IS11:NWK.
- Now switch to amber jewels and add IS5:RWK. (R = Reverse) followed by IS7:RWK, IS9:RWK, and IS11:RWK.





Turnout Feedback

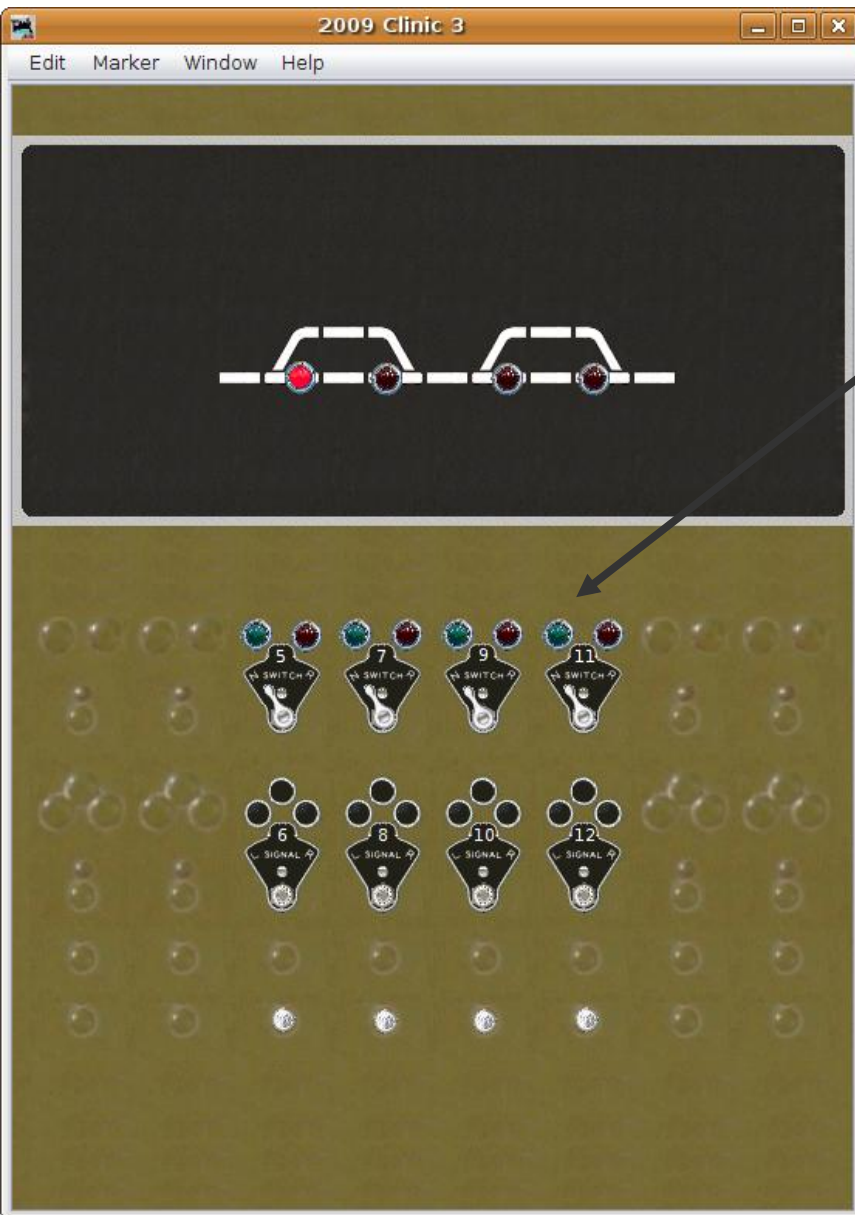
- Note: Later on, in the edit portion of these clinics, we can switch these sensors to direct feedback for testing purposes.





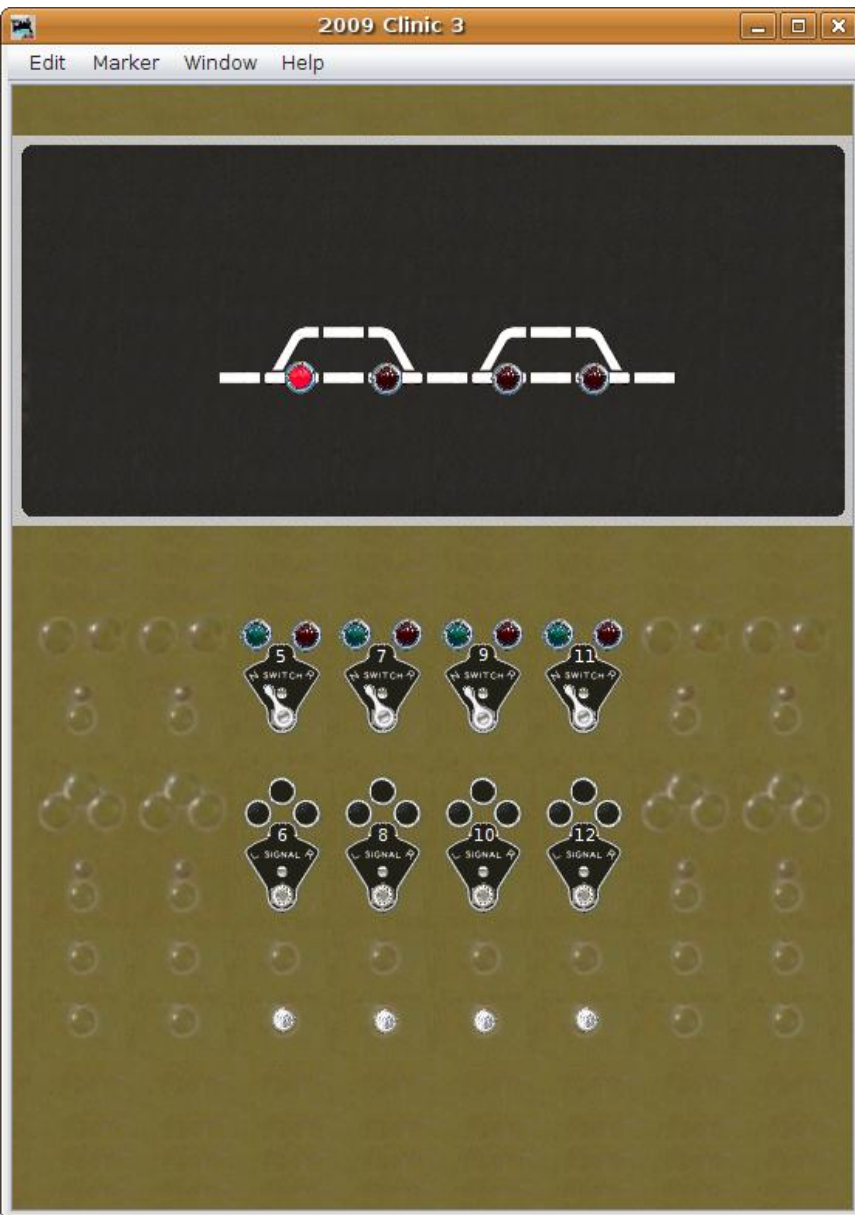
Turnout Feedback

- Move the jewels into position. We now have panel icons that we can control based on layout input. We could have used the turnout feedback contacts directly for this, but that would prevent us from adding sound effects for example. We would also need to have our layout hardware exactly line up with the panel ID numbers. This way we can also add a translation table as required. We used 'off' instead of 'inconsistent' images so that JMRI wouldn't flash a 'X' rated image if it decided that the settings were inconsistent.



Indirect Layout Control

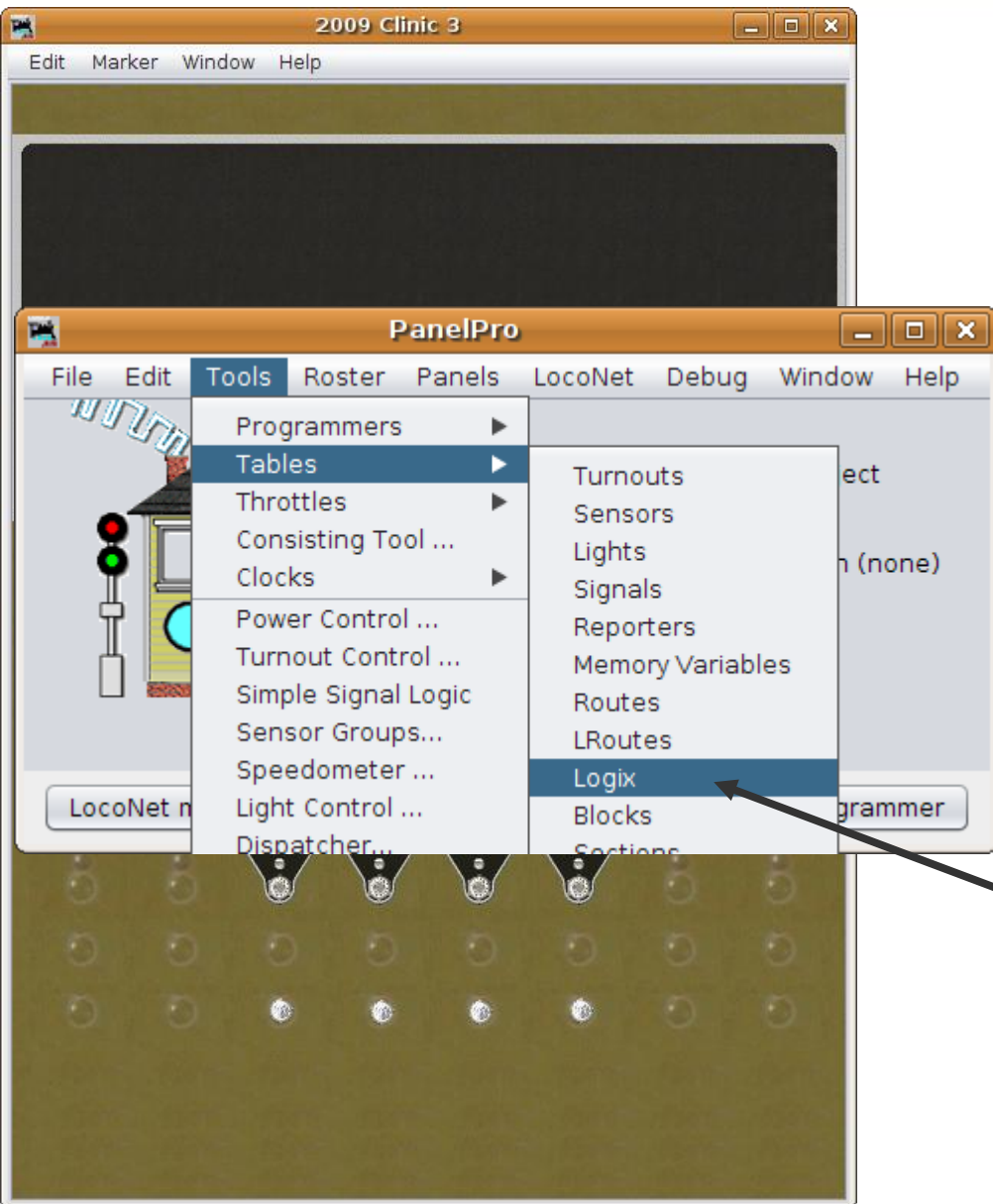
Logix



Logix

- We now have all our required inputs and outputs on the panel. All that is missing is the logic to make it work. Our first example will be simple:

If the Control Lever is changed
And the OS is NOT occupied
And the Code Button is pressed
Then send a turnout command



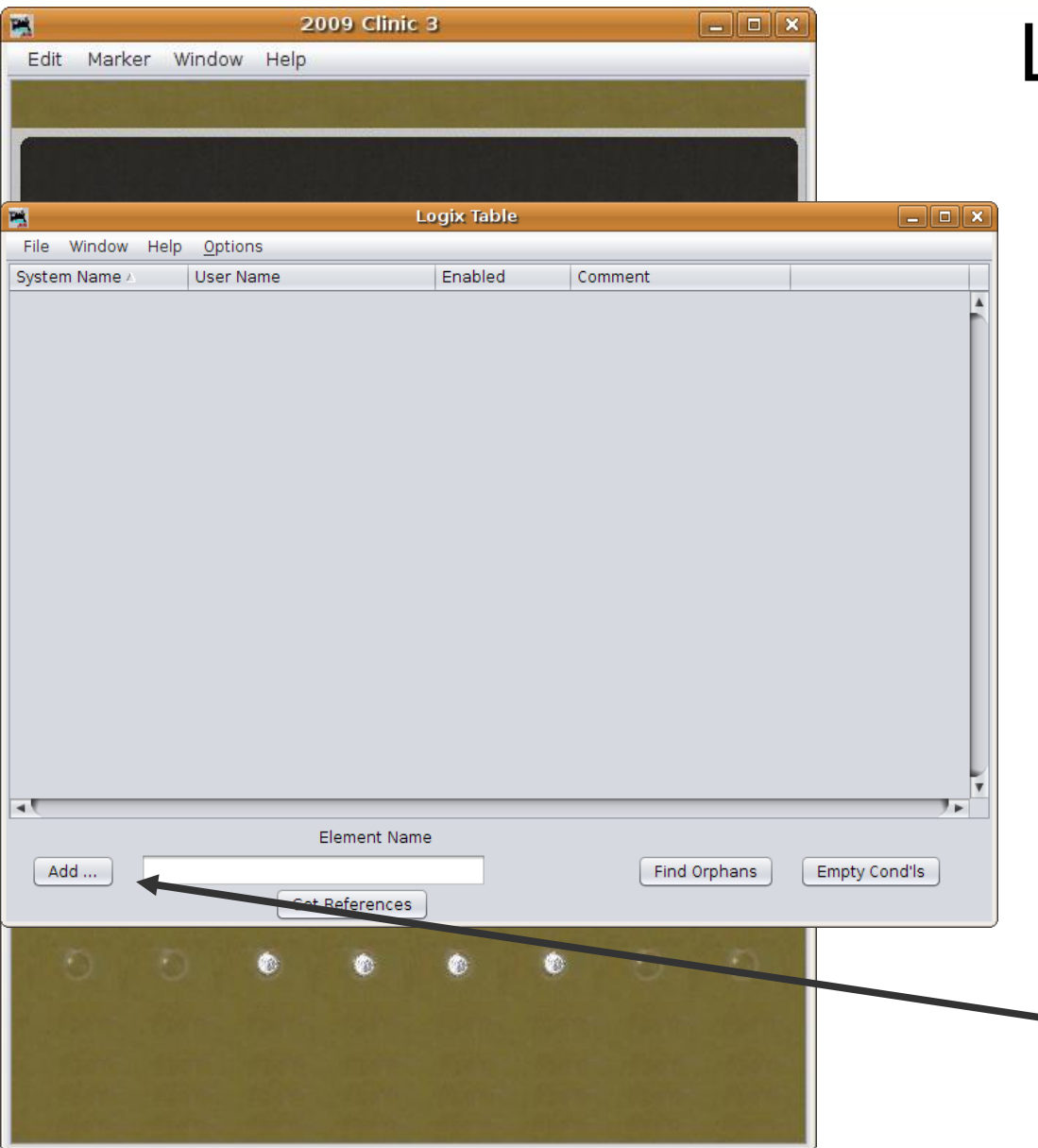
Logix

- We now have all our required inputs and outputs on the panel. All that is missing is the logic to make it work. Our first example will be simple:

If the Control Lever is changed And the OS is NOT occupied And the Code Button is pressed

Then send a turnout command

- Open Logix by selecting 'Tools - Tables - Logix'



Logix

- We now have all our required inputs and outputs on the panel. All that is missing is the logic to make it work. Our first example will be simple:

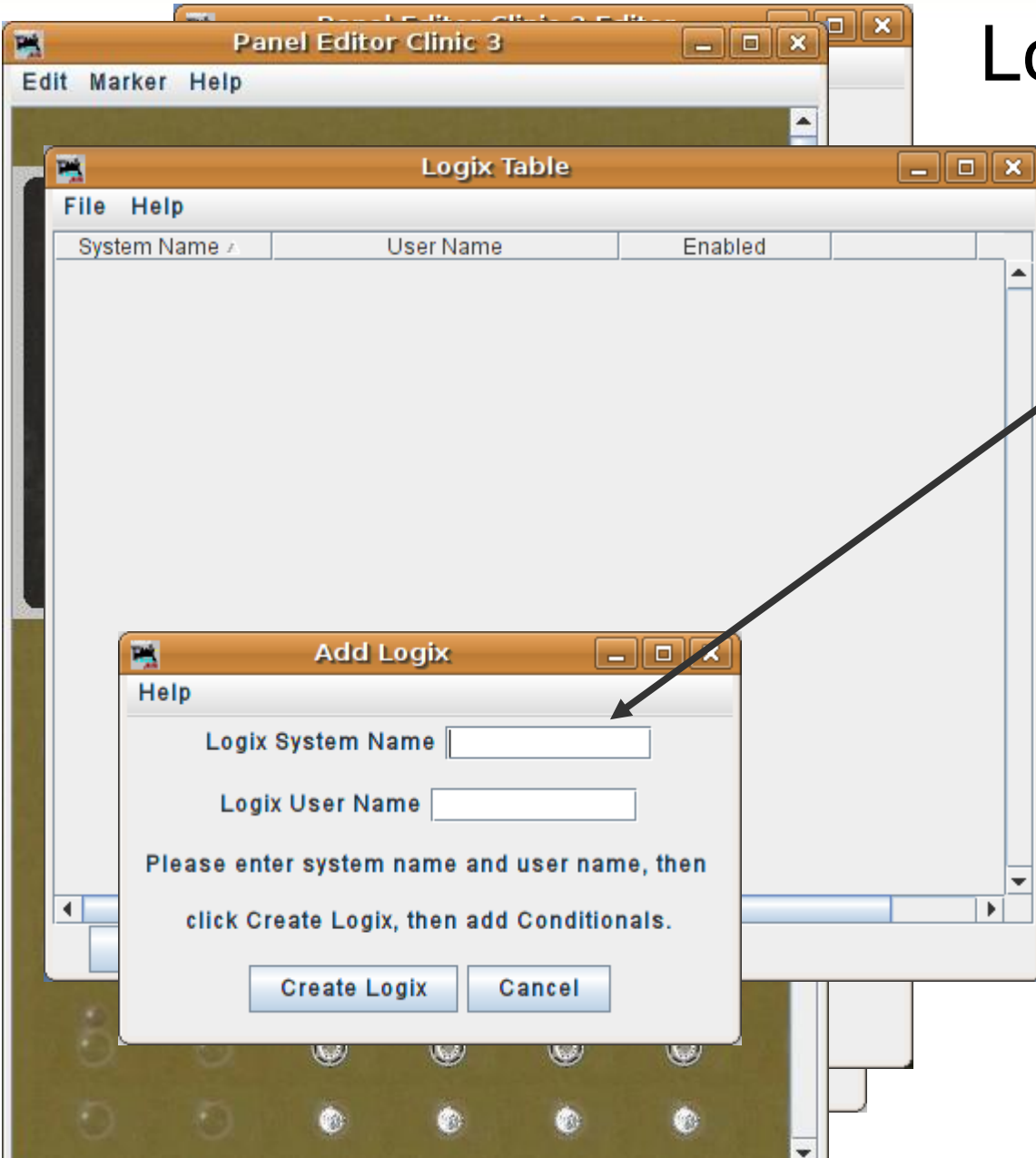
If the Control Lever is changed And the OS is NOT occupied And the Code Button is pressed
Then send a turnout command

- Open Logix by selecting 'Tools - Tables - Logix'
- This opens a new 'Logix Table' view. Click on 'Add..'



Logix naming

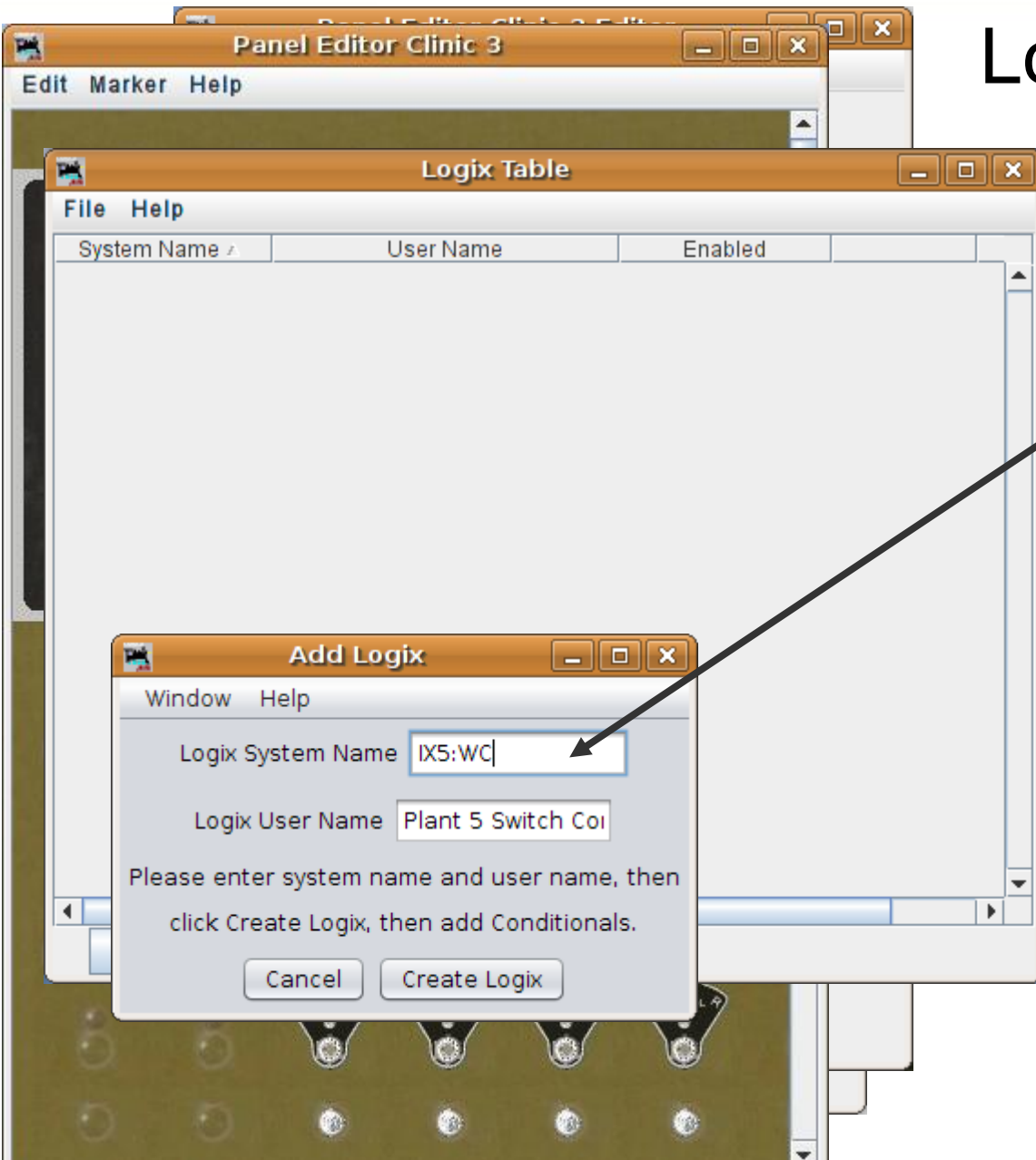
- The first information will be the ID. Logix are internal so the system name is 'I'. The item name is 'X', so they will start with 'IX'.





Logix naming

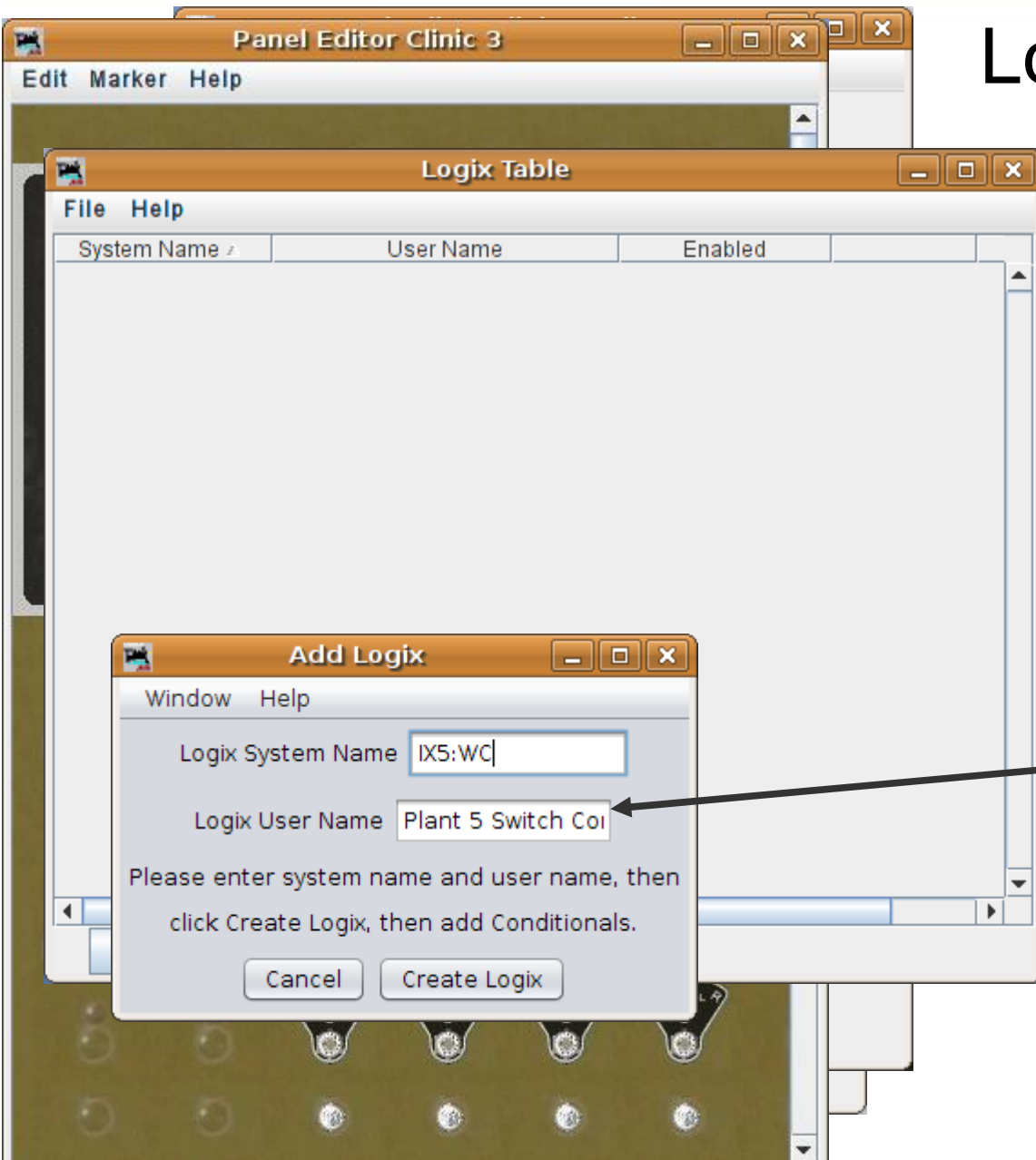
- The first information will be the ID. Logix are internal so the system name is 'I'. The item name is 'X', so they will start with 'IX'.
- We will call it IX5:WC
IX = Internal Logi**X**,
5 = Plant 5,
WC = s**W**itch **C**ontroller.





Logix naming

- The first information will be the ID. Logix are internal so the system name is 'I'. The item name is 'X', so they will start with 'IX'.
- We will call it IX5:WC
IX = Internal Logi**X**,
5: = Plant 5,
WC = s**W**itch **C**ontroller.
- Enter the user name “Plant 5 Switch Controller”. All user names must be unique so we include the plant number.



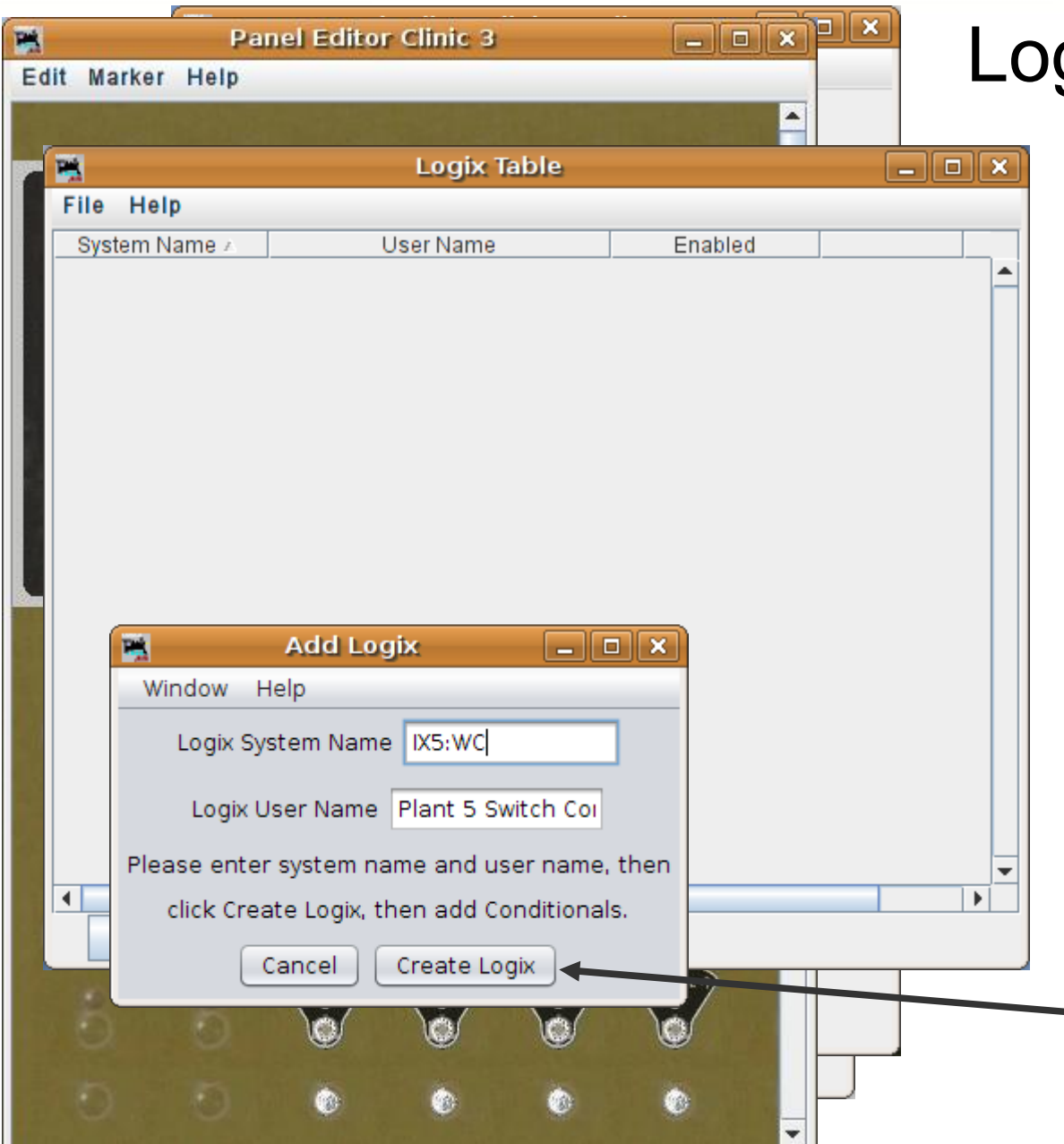
Indirect Layout Control

Logix naming



Logix naming

- The first information will be the ID. Logix are internal so the system name is 'I'. The item name is 'X', so they will start with 'IX'.
- We will call it IX5:WC
IX = Internal Logi**X**,
5: = Plant 5,
WC = s**W**itch **C**ontroller.
- Enter the user name “Plant 5 Switch Controller”. All user names must be unique so we include the plant number.
- Once we have named our new creation, click on 'Create Logix' to add it to the table window, and open the Logix Editor.



Indirect Layout Control

Logix entry



Logix entry

- The new entry shows in the table.

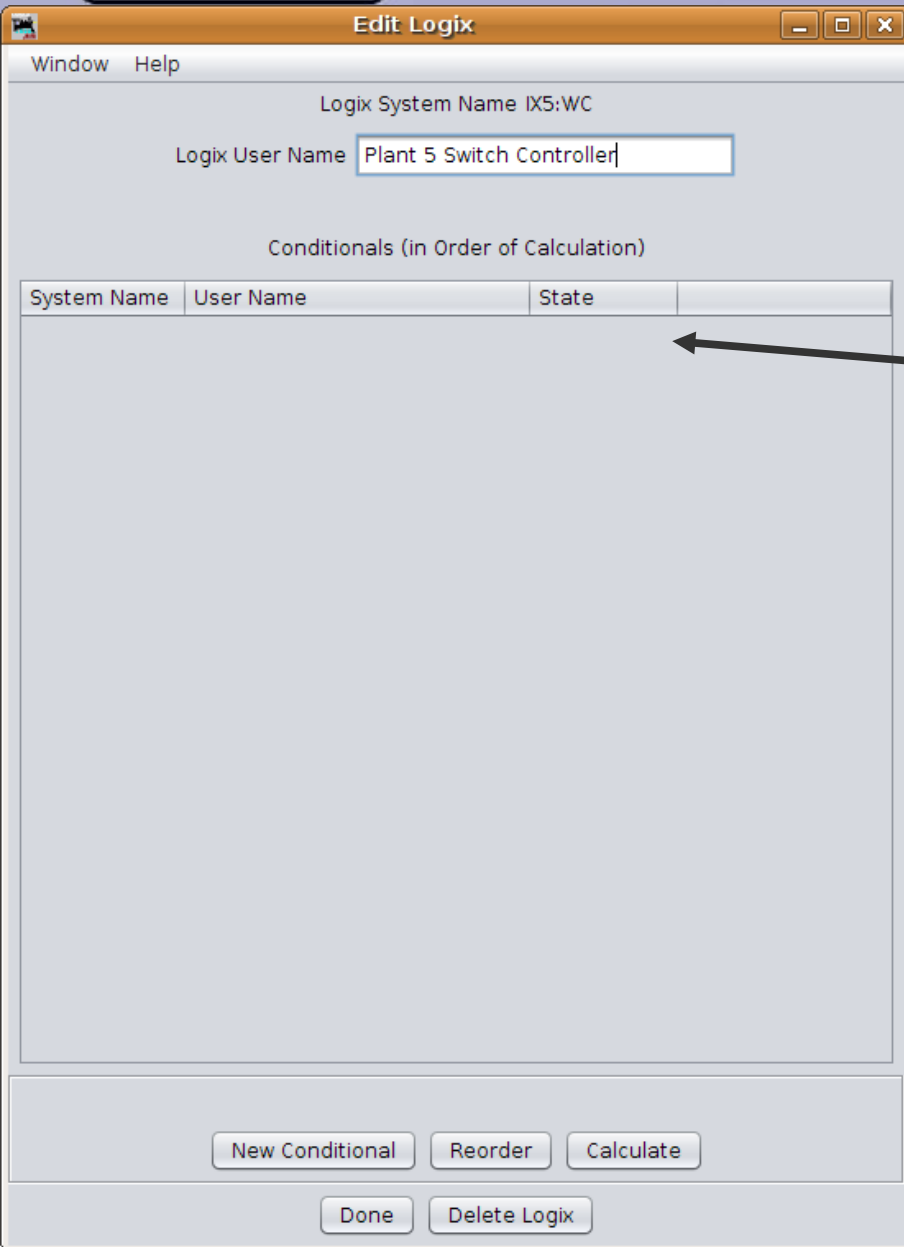
The screenshot shows the "Logix Table" window within the "Panel Editor Clinic 3" application. The window has a menu bar with "File", "Window", "Help", and "Options". Below the menu bar is a table with the following data:

System Name	User Name	Enabled	Comment
IX5:WC	Plant 5 Switch Controller	<input checked="" type="checkbox"/>	Select

Below the table, there is an "Element Name" label and a text input field. At the bottom of the window, there are four buttons: "Add ...", "Get References", "Find Orphans", and "Empty Cond'ls". The background of the application shows a partial view of a panel editor with several "SIGNAL" indicators.

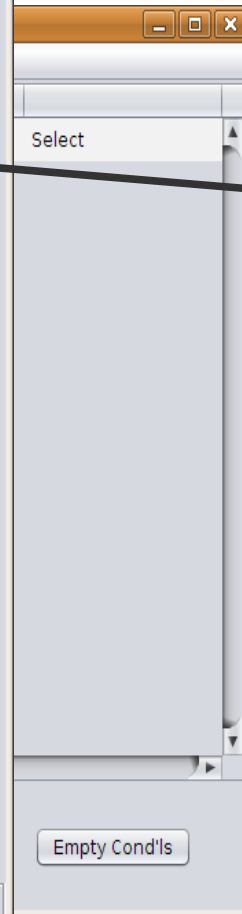
Indirect Layout Control

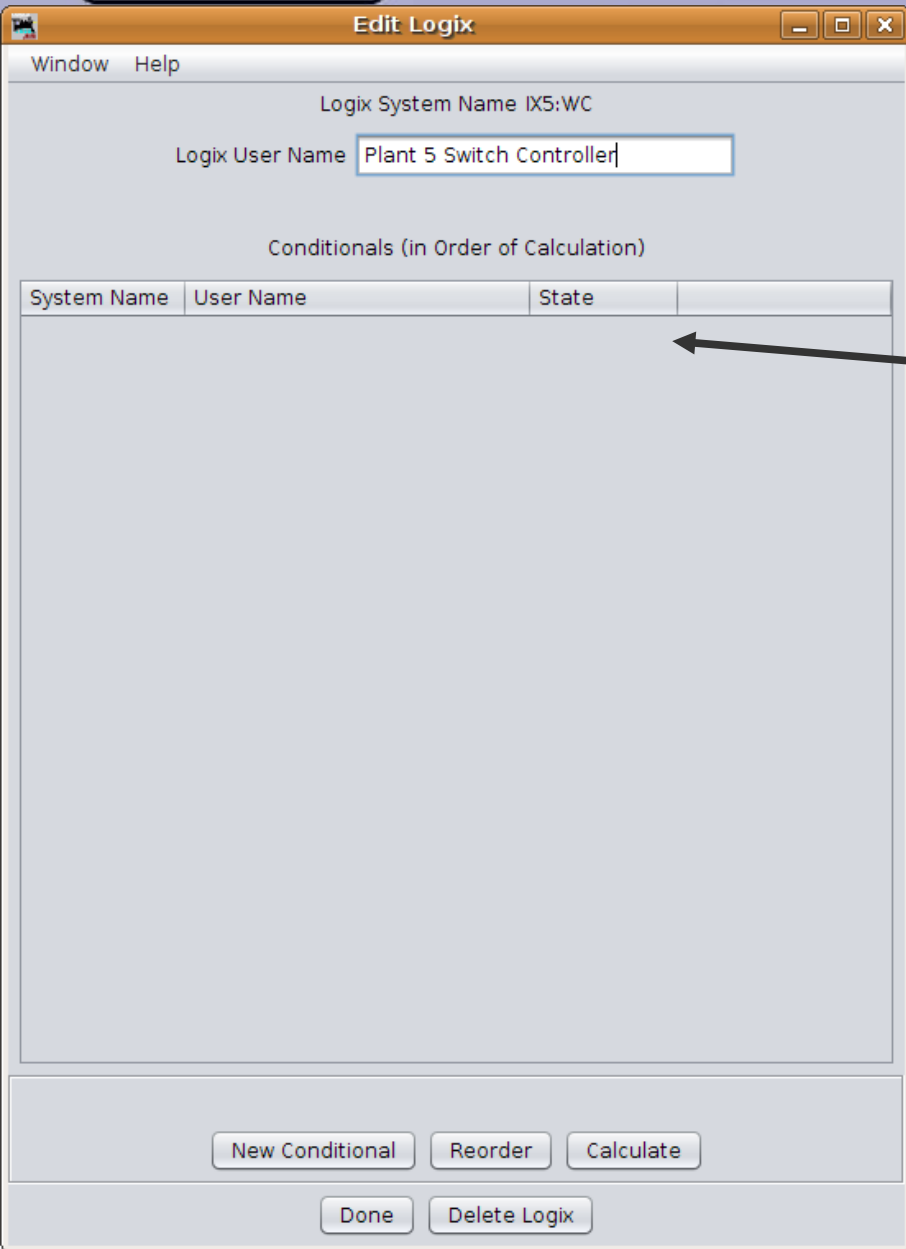
Logix entry



Logix entry

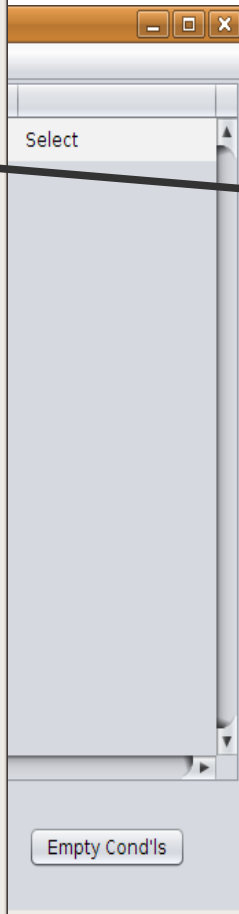
- The new entry shows in the table.
- And its 'Edit Logix' window opens.





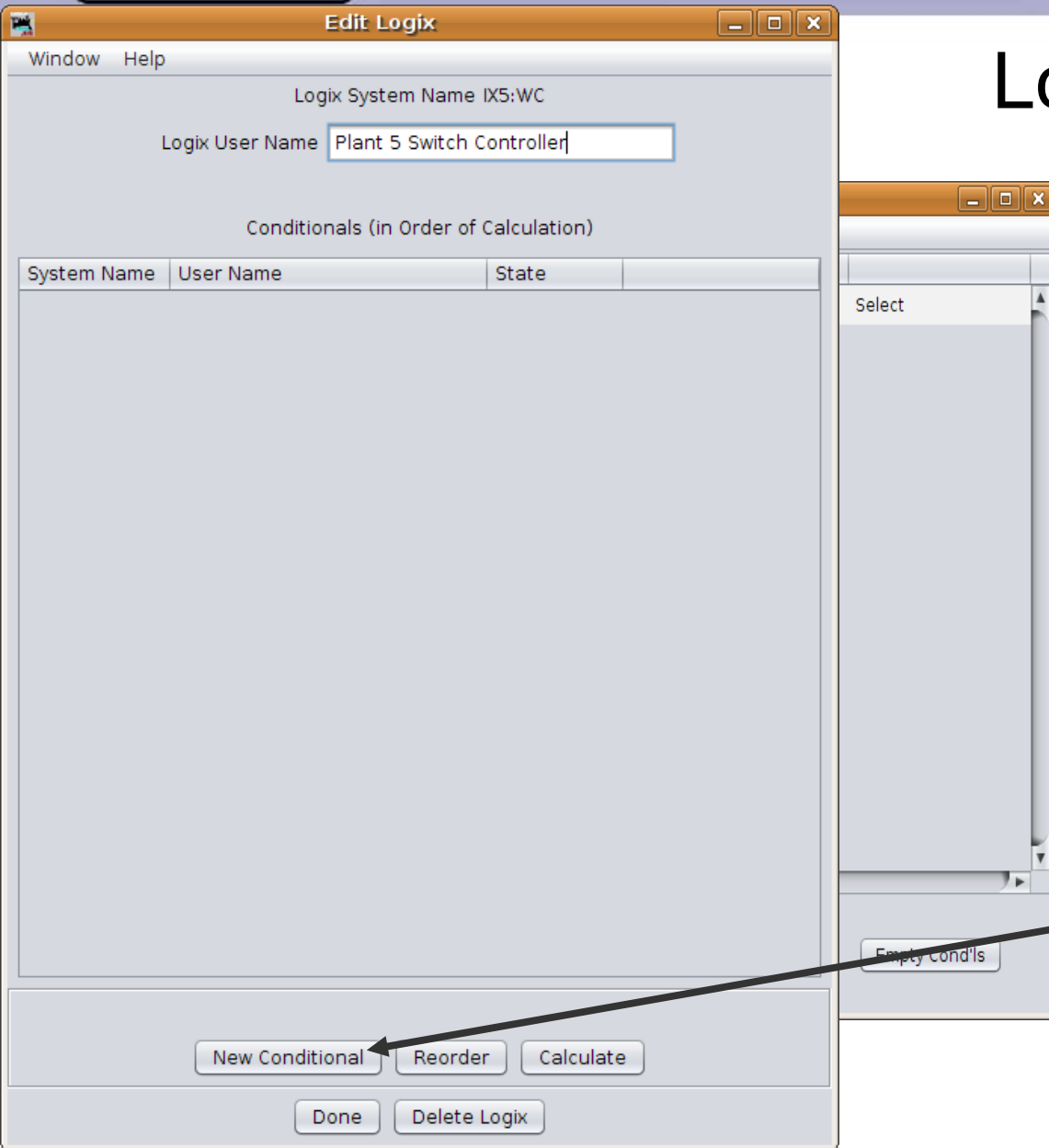
Logix entry

- The new entry shows in the table.
- And its 'Edit Logix' window opens.
- Each Logix will contain one or more 'Conditionals' or things that may be true or false. A 'Conditional' may optionally do one or more actions when it becomes true or becomes false or simply changes state.



Indirect Layout Control

Logix entry



Logix entry

- The new entry shows in the table.
- And its 'Edit Logix' window opens.
- Each Logix will contain one or more 'Conditionals' or things that may be true or false. A 'Conditional' may optionally do one or more actions when it becomes true or becomes false or simply changes state.
- Click the 'New Conditional' button to bring up the 'Edit Conditional' window.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable	Description	State	Trigg...
-----	------	-----	----------------	-------------	-------	----------

Add State Variable Check State Variables

Logic Operator

AND

Actions

Consequent Actions (the 'then' part)

Action Description

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Note that JMRI automatically added 'C1' to the name we gave this item.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable	Description	State	Trigg...
-----	------	-----	----------------	-------------	-------	----------

Add State Variable Check State Variables

Logic Operator

AND

Actions

Consequent Actions (the 'then' part)

Action Description

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Note that JMRI automatically added 'C1' to the name we gave this item.
- Name this first one “Switch 5 Normal”.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable	Description	State	Trigg...
-----	------	-----	----------------	-------------	-------	----------

Add State Variable Check State Variables

Logic Operator

AND

Actions

Consequent Actions (the 'then' part)

Action Description

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Note that JMRI automatically added 'C1' to the name we gave this item.
- Name this first one "Switch 5 Normal".
- We call the various items that will be checked by Logix 'Variables' because they 'vary' as things change on the layout. In this case between being true and being false. Click here to add our first one.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable	Description	State	Trigg...
-----	------	-----	----------------	-------------	-------	----------

Antecedent Variable

Variable Type

Update Cancel Delete

Action Description

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Note that JMRI automatically added 'C1' to the name we gave this item.
- Name this first one "Switch 5 Normal".
- We call the various items that will be checked by Logix 'Variables' because they 'vary' as things change on the layout. In this case between being true and being false. Click here to add our first one.
- Click in the 'Variable Type' box to open a list of available options.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable	Description	State	Trigg...
-----	------	-----	----------------	-------------	-------	----------

Update

Variable Type

- Sensor Active
- Sensor Inactive**
- Turnout Thrown
- Turnout Closed

Add Action Reorder

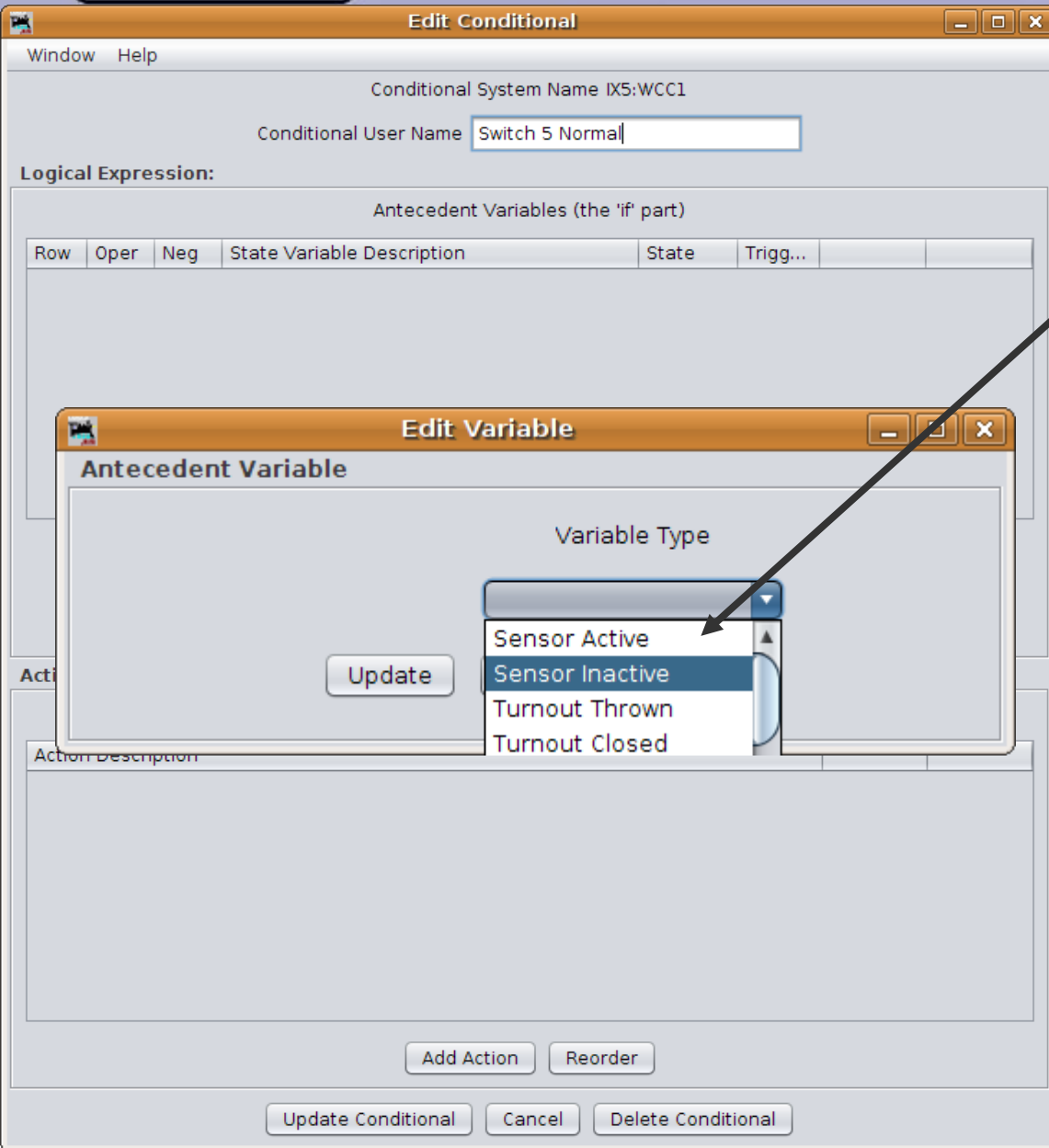
Update Conditional Cancel Delete Conditional

Logix entry

- Note that JMRI automatically added 'C1' to the name we gave this item.
- Name this first one "Switch 5 Normal".
- We call the various items that will be checked by Logix 'Variables' because they 'vary' as things change on the layout. In this case between being true and being false. Click here to add our first one.
- Click in the 'Variable Type' box to open a list of available options.
- Choose 'Sensor Inactive' so this will not happen unless a train is NOT on the OS.

Indirect Layout Control

Logix entry



Logix entry

- Note we could have also said NOT 'Sensor Active' but that would include when it was 'Unknown' or 'Inconsistent', probably not good options here.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable	Description	State	Trigg...
-----	------	-----	----------------	-------------	-------	----------

Antecedent Variable

Variable Type:

System / User Name:

Update Cancel Delete

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Note we could have also said NOT 'Sensor Active' but that would include when it was 'Unknown' or 'Inconsistent', probably not good options here.
- Enter the OS sensor ID. In this case it is 'LS2', then click 'Update' to add the variable.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	False	<input checked="" type="checkbox"/>	Edit	Delete

Add State Variable Check State Variables

Logic Operator
AND

Actions

Consequent Actions (the 'then' part)

Action Description

Add Action Reorder

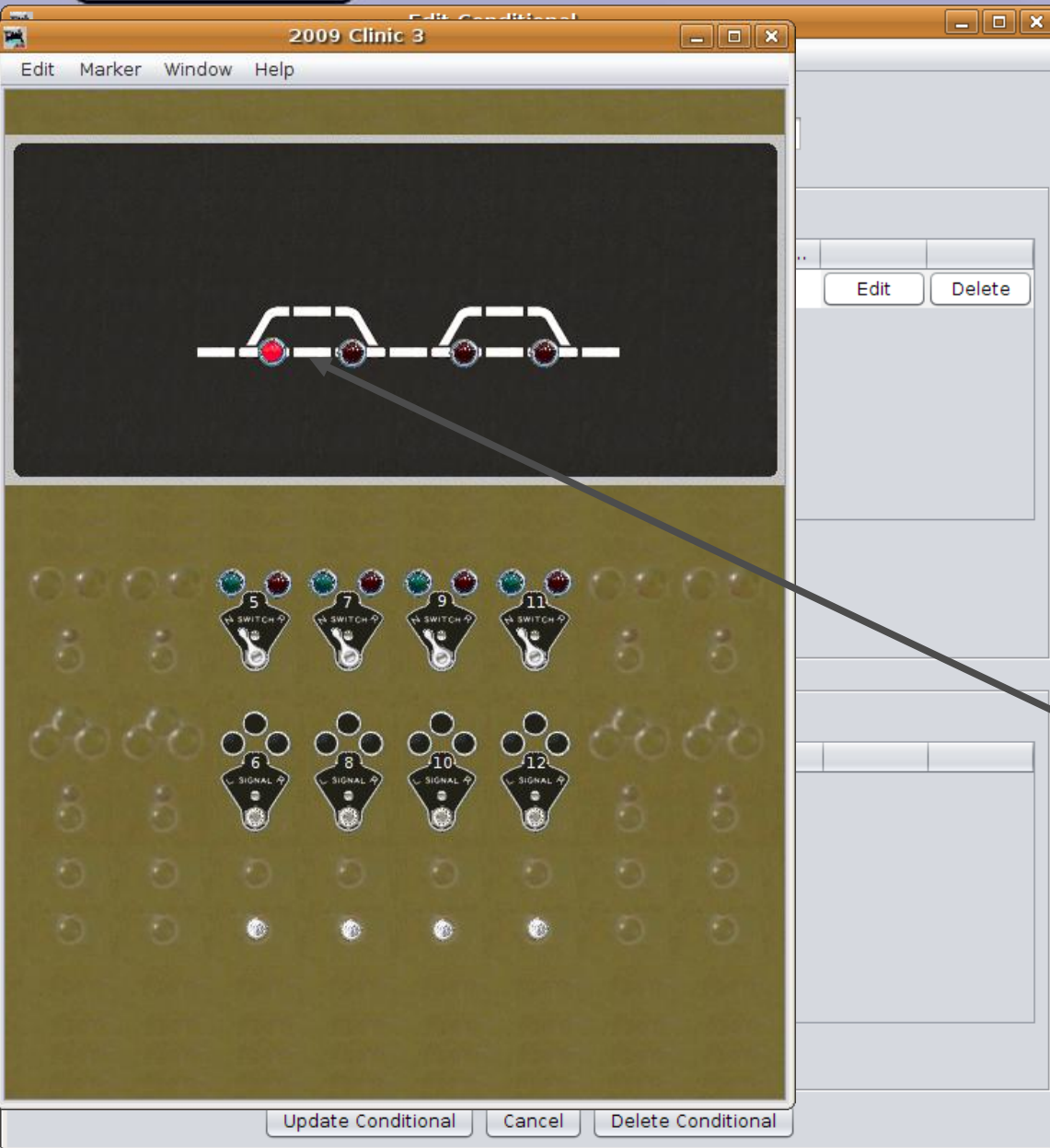
Update Conditional Cancel Delete Conditional

Logix entry

- Note we could have also said NOT 'Sensor Active' but that would include when it was 'Unknown' or 'Inconsistent', probably not good options here.
- Enter the OS sensor ID. In this case it is 'LS2', then click 'Update' to add the variable.
- The variable state is 'false'.

Indirect Layout Control

Logix entry



Logix entry

- Note we could have also said NOT 'Sensor Active' but that would include when it was 'Unknown' or 'Inconsistent', probably not good options here.
- Enter the OS sensor ID. In this case it is 'LS2', then click 'Update' to add the variable.
- The variable state is 'false'.
- Remember, we left the sensor 'Active'.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	False	<input checked="" type="checkbox"/>	Edit	Delete

Add State Variable Check State Variables

Logic Operator

AND

Actions

Consequent Actions (the 'then' part)

Action Description

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Note we could have also said NOT 'Sensor Active' but that would include when it was 'Unknown' or 'Inconsistent', probably not good options here.
- Enter the OS sensor ID. In this case it is 'LS2', then click 'Update' to add the variable.
- The variable state is 'false'.
- Remember, we left the sensor 'Active'.
- Add another variable for the code button.

Indirect Layout Control

Logix entry



The screenshot shows two overlapping windows from the JMRI software. The top window is titled 'Edit Conditional' and displays the configuration for a conditional system named 'IX5:WCC1'. The 'Conditional User Name' is set to 'Switch 5 Normal'. Under the 'Logical Expression' section, there is a table of 'Antecedent Variables (the 'if' part)'. The table has columns for 'Row', 'Oper', 'Neg', 'State Variable Description', 'State', and 'Trigg...'. A single row is visible with 'R1', an empty 'Oper' field, an empty 'Neg' field, 'Sensor, LS2, for Sensor Inactive', 'False', and a checked 'Trigg...' checkbox. Below the table are 'Edit' and 'Delete' buttons. The bottom window is titled 'Edit Variable' and shows the configuration for an 'Antecedent Variable'. The 'Variable Type' is set to 'Sensor Active' and the 'System / User Name' is 'IS6:CB'. There are 'Update', 'Cancel', and 'Delete' buttons. An arrow points from the 'Update' button in the 'Edit Variable' window to the 'Trigg...' checkbox in the 'Edit Conditional' window.

Row	Oper	Neg	State Variable Description	State	Trigg...
R1			Sensor, LS2, for Sensor Inactive	False	<input checked="" type="checkbox"/>

Logix entry

- Note we could have also said NOT 'Sensor Active' but that would include when it was 'Unknown' or 'Inconsistent', probably not good options here.
- Enter the OS sensor ID. In this case it is 'LS2', then click 'Update' to add the variable.
- The variable state is 'false'.
- Remember, we left the sensor 'Active'.
- Add another variable for the code button.
- If you forgot that it is IS6:CB simply hover the cursor over the button.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	False	<input checked="" type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete

Add State Variable Check State Variables

Logic Operator

AND

Actions

Consequent Actions (the 'then' part)

Action Description

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Both items are checked to 'trigger' the action. We probably only want it to happen when the code button is first pressed, so un-check the first one.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	False	<input checked="" type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete

Add State Variable Check State Variables

Logic Operator
AND

Actions

Consequent Actions (the 'then' part)

Action Description

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Both items are checked to 'trigger' the action. We probably only want it to happen when the code button is first pressed, so un-check the first one.
- We also only need to send the command if the turnout is **not** already in position, so let's add another variable for that.

Indirect Layout Control

Logix entry



Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	False	<input checked="" type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete

Antecedent Variable

Variable Type

- Turnout Thrown
- Turnout Closed
- Conditional True
- Conditional False

Update

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Both items are checked to 'trigger' the action. We probably only want it to happen when the code button is first pressed, so un-check the first one.
- We also only need to send the command if the turnout is **not** already in position, so let's add another variable for that.
- We check on 'Turnout Closed'

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	False	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND		Turnout, LT1, for Turnout Closed	False	<input type="checkbox"/>	Edit	Delete

Add State Variable Check State Variables

Logic Operator
AND

Actions

Consequent Actions (the 'then' part)

Action Description

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Both items are checked to 'trigger' the action. We probably only want it to happen when the code button is first pressed, so un-check the first one.
- We also only need to send the command if the turnout is **not** already in position, so lets add another variable for that.
- We check on 'Turnout Closed'
- But see that it is 'false'. We need it to be 'true' for our logic to work.

Indirect Layout Control

Logix entry



Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	False	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND		Turnout, LT1, for Turnout Closed	False	<input type="checkbox"/>	Edit	Delete

NOT

Add State Variable Check State Variables

Logic Operator AND

Actions

Consequent Actions (the 'then' part)

Action Description

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Both items are checked to 'trigger' the action. We probably only want it to happen when the code button is first pressed, so un-check the first one.
- We also only need to send the command if the turnout is **not** already in position, so lets add another variable for that.
- We check on 'Turnout Closed'
- But see that it is 'false'. We need it to be 'true' for our logic to work.
- If we click just to the left of the description we discover that we can add a 'NOT' to the variable.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	False	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete

Add State Variable Check State Variables

Logic Operator

AND

Actions

Consequent Actions (the 'then' part)

Action Description

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Now our statements seem to be what we need... Oh yes, we only want to do this if the lever tells us to.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	False	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete

Logic Operator

AND ▾

Actions

Consequent Actions (the 'then' part)

Action Description		

Logix entry

- Now our statements seem to be what we need... Oh yes, we only want to do this if the lever tells us to.
- Add another variable for the lever's position.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	False	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete

Antecedent Variable

Variable Type System / User Name

Sensor Active IS5:WL

Update Cancel Delete

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Now our statements seem to be what we need... Oh yes, we only want to do this if the lever tells us to.
- Add another variable for the lever's position.
- IS5:WL (**5**: s**W**itch **L**ever)

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	False	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Active	True	<input type="checkbox"/>	Edit	Delete

Add State Variable Check State Variables

Logic Operator
AND

Actions

Consequent Actions (the 'then' part)

Action Description

Add Action Reorder

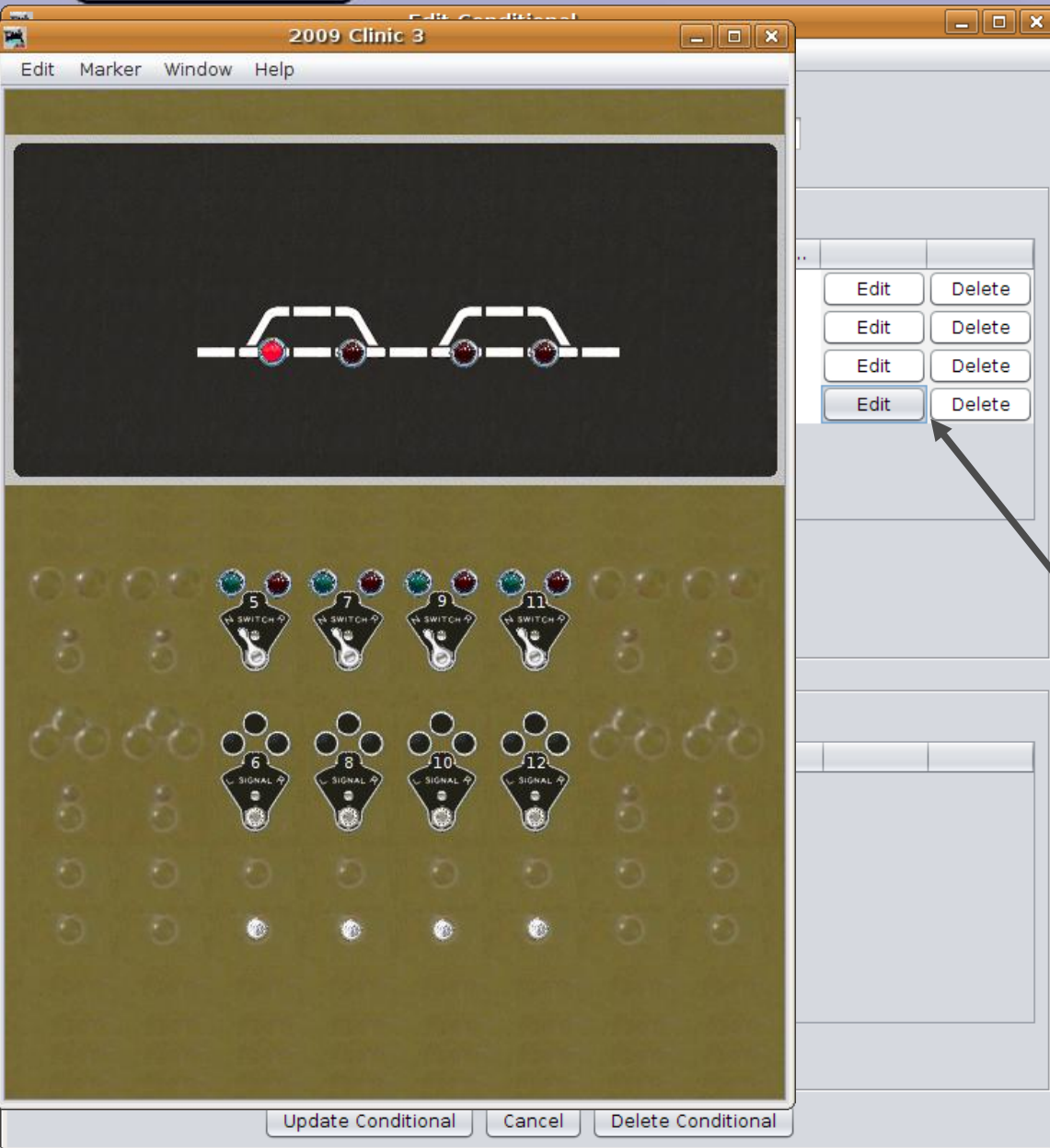
Update Conditional Cancel Delete Conditional

Logix entry

- Now our statements seem to be what we need... Oh yes, we only want to do this if the lever tells us to.
- Add another variable for the lever's position.
- IS5:WL (**5**: s**W**itch **L**ever)
- Note that you may Delete or Edit any variable at any time.

Indirect Layout Control

Logix entry



Logix entry

- Now our statements seem to be what we need... Oh yes, we only want to do this if the lever tells us to.
- Add another variable for the lever's position.
- IS5:WL (**5**: s**W**itch **L**ever)
- Note that you may Delete or Edit any variable at any time.
- Bring the panel to the front and 'move' the train off of the OS section. (by clicking the sensor)

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	False	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Active	True	<input type="checkbox"/>	Edit	Delete

Add State Variable Check State Variables

Logic Operator
AND

Actions

Consequent Actions (the 'then' part)

Action Description

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Now our statements seem to be what we need... Oh yes, we only want to do this if the lever tells us to.
- Add another variable for the lever's position.
- IS5:WL (**5**: s**W**itch **L**ever)
- Note that you may Delete or Edit any variable at any time.
- Bring the panel to the front and 'move' the train off of the OS section. (by clicking the sensor)
- Now 'Check State Variables' to re-read the layout.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Active	True	<input type="checkbox"/>	Edit	Delete

Add State Variable Check State Variables

Logic Operator
AND

Actions

Consequent Actions (the 'then' part)

Action Description

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Now our statements seem to be what we need... Oh yes, we only want to do this if the lever tells us to.
- Add another variable for the lever's position.
- IS5:WL (**5**: s**W**itch **L**ever)
- Note that you may Delete or Edit any variable at any time.
- Bring the panel to the front and 'move' the train off of the OS section. (by clicking the sensor)
- Now 'Check State Variables' to re-read the layout.
- Everything is now "true" but the button itself.

Indirect Layout Control

Logix entry



Logix entry

- Time to put some 'Action' in our Logix.

Window Help

Conditional System Name IX5:WCC1

Conditional User Name

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Active	True	<input type="checkbox"/>	Edit	Delete

Logic Operator

▼

Actions

Consequent Actions (the 'then' part)

Action Description		

Indirect Layout Control

Logix entry



Logix entry

- Time to put some 'Action' in our Logix.

Window Help

Conditional System Name IX5:WCC1

Conditional User Name

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Active	True	<input type="checkbox"/>	Edit	Delete

Consequent Action

Action Type

Indirect Layout Control

Logix entry



Logix entry

- Time to put some 'Action' in our Logix.
- Specifically 'Set Turnout'.

Edit Conditional

Window Help

Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Active	True	<input type="checkbox"/>	Edit	Delete

Edit Action

Consequent Action

Action Type

- None
- None
- Set Turnout
- Set Signal Appearance
- Set Signal Held
- Clear Signal Held

Add Action Reorder

Update Conditional Cancel Delete Conditional

Indirect Layout Control

Logix entry



Logix entry

- Time to put some 'Action' in our Logix.
- Specifically 'Set Turnout'. Enter 'LT1', then 'Update'.

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Active	True	<input type="checkbox"/>	Edit	Delete

Consequent Action

Action Type: Set Turnout

Change Option: On Change To True

System / User Name: [Empty]

Turnout Position: Closed

Update Cancel Delete

Add Action Reorder

Update Conditional Cancel Delete Conditional

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Active	True	<input type="checkbox"/>	Edit	Delete

Add State Variable Check State Variables

Logic Operator

AND

Actions

Consequent Actions (the 'then' part)

Action Description		
On Change To True, Set Turnout, LT1 to Closed	Edit	Delete

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Time to put some 'Action' in our Logix.
- Specifically 'Set Turnout'. Enter 'LT1', then 'Update'.
- Our first action.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Active	True	<input type="checkbox"/>	Edit	Delete

Add State Variable Check State Variables

Logic Operator
AND

Actions

Consequent Actions (the 'then' part)

Action Description		
On Change To True, Set Turnout, LT1 to Closed	Edit	Delete

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Time to put some 'Action' in our Logix.
- Specifically 'Set Turnout'. Enter 'LT1', then 'Update'.
- Our first action.
- Lets get fancy and send the sound of the code relays. Add another action 'Play Sound File'.

Indirect Layout Control

Logix entry



Edit Conditional

Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete

Edit Action

Consequent Action

Action Type: Play Sound File

Change Option: On Change To True

Set File: File

Action Data: [Text Field]

Click for a file selection dialog for choosing a sound file

Consequent Actions (the 'then' part)

Action Description		
On Change To True, Set Turnout, LT1 to Closed	Edit	Delete

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Time to put some 'Action' in our Logix.
- Specifically 'Set Turnout'. Enter 'LT1', then 'Update'.

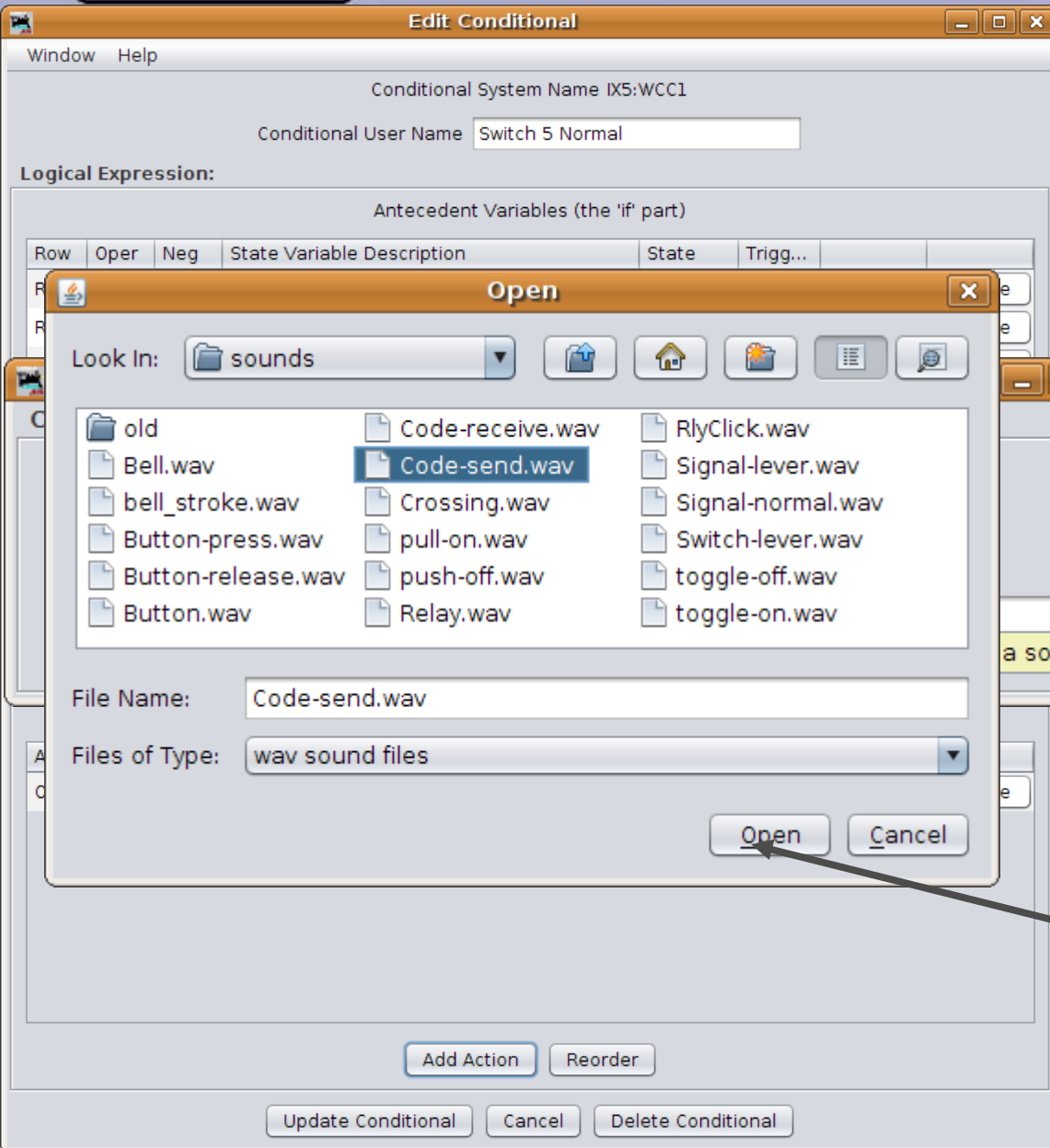
Our first action.

Lets get fancy and send the sound of the code relays. Add another action 'Play Sound File'.

- Click on 'File' to search for sounds.

Indirect Layout Control

Logix entry



Logix entry

- Time to put some 'Action' in our Logix.
- Specifically 'Set Turnout'. Enter 'LT1', then 'Update'.

Our first action.

Lets get fancy and send the sound of the code relays. Add another action 'Play Sound File'.

- Click on 'File' to search for sounds.
- 'Code-send.wav' is what we need. Select then 'Open'.

Indirect Layout Control

Logix entry



The screenshot shows two overlapping windows from the JMRI software. The top window is titled 'Edit Conditional' and displays the configuration for a conditional system named 'IX5:WCC1'. The 'Conditional User Name' is 'Switch 5 Normal'. Under 'Logical Expression', there is a table of antecedent variables:

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete

The bottom window is titled 'Edit Action' and shows the configuration for a consequent action. The 'Action Type' is 'Play Sound File' and the 'Change Option' is 'On Change To True'. The 'Set File' is '/usr/local/JMRI/resources/sounds/Code-send.wav'. The 'Update' button is highlighted with a blue border. Below this window, a table shows the list of consequent actions:

Action Description		
On Change To True, Set Turnout, LT1 to Closed	Edit	Delete

At the bottom of the 'Edit Action' window, there are buttons for 'Add Action', 'Reorder', 'Update Conditional', 'Cancel', and 'Delete Conditional'. An arrow points from the 'Update' button in the 'Edit Action' window to the 'Update Conditional' button in the 'Edit Conditional' window.

Logix entry

- Time to put some 'Action' in our Logix.
- Specifically 'Set Turnout'. Enter 'LT1', then 'Update'.

Our first action.

Lets get fancy and send the sound of the code relays. Add another action 'Play Sound File'.

- Click on 'File' to search for sounds.
- 'Code-send.wav" is what we need. Select then 'Open'.
- Click 'Update' to enter the new action.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Active	True	<input type="checkbox"/>	Edit	Delete

Add State Variable Check State Variables

Logic Operator
AND

Actions

Consequent Actions (the 'then' part)

Action Description		
On Change To True, Set Turnout, LT1 to Closed	Edit	Delete
On Change To True, Play Sound File from file, /usr/local/JMRI/resources/sounds/Code-...	Edit	Delete

Add Action Reorder

Update Conditional Cancel Delete Conditional

Logix entry

- Time to put some 'Action' in our Logix.
- Specifically 'Set Turnout'. Enter 'LT1', then 'Update'.
- Our first action.
- Lets get fancy and send the sound of the code relays. Add another action 'Play Sound File'.
- Click on 'File' to search for sounds.
- 'Code-send.wav" is what we need. Select then 'Open'.
- Click 'Update' to enter the new action.
- The sounds should be first. :(

Indirect Layout Control

Logix entry



Logix entry

- Click on Reorder.

Window Help

Conditional System Name IX5:WCC1

Conditional User Name

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Active	True	<input type="checkbox"/>	Edit	Delete

Logic Operator

▼

Actions

Consequent Actions (the 'then' part)

Action Description		
On Change To True, Set Turnout, LT1 to Closed	Edit	Delete
On Change To True, Play Sound File from file, /usr/local/JMRI/resources/sounds/Code-...	Edit	Delete

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Active	True	<input type="checkbox"/>	Edit	Delete

Logic Operator

▼

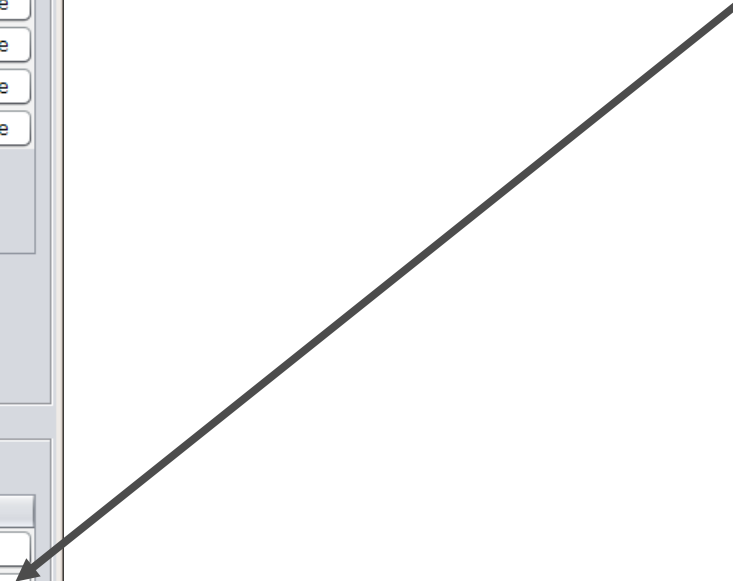
Actions

Consequent Actions (the 'then' part)

Action Description		
On Change To True, Set Turnout, LT1 to Closed	Edit	First
On Change To True, Play Sound File from file, /usr/local/JMRI/resources/sounds/Code-...	Edit	First

Logix entry

- Click on Reorder.
- Then choose who is on first.



Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Active	True	<input type="checkbox"/>	Edit	Delete

Logic Operator

▼

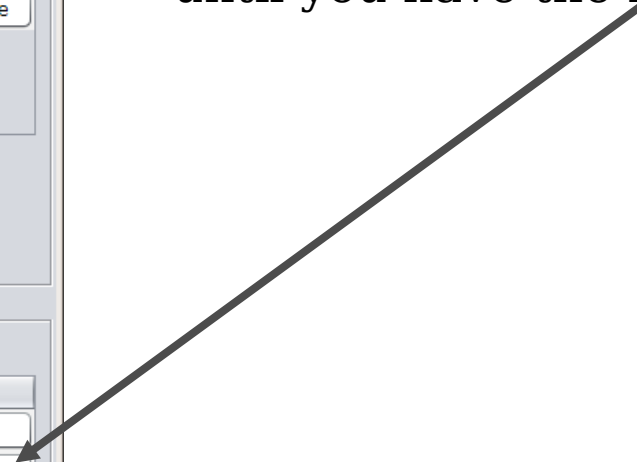
Actions

Consequent Actions (the 'then' part)

Action Description		
On Change To True, Play Sound File from file, /usr/local/JMRI/resources/sounds/Code-...	Edit	1
On Change To True, Set Turnout, LT1 to Closed	Edit	Next

Logix entry

- Click on Reorder.
- Then choose who is on first.
- Continue picking the next item until you have the new order.



Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Active	True	<input type="checkbox"/>	Edit	Delete

Logic Operator

▼

Actions

Consequent Actions (the 'then' part)

Action Description		
On Change To True, Play Sound File from file, /usr/local/JMRI/resources/sounds/Code-...	Edit	1
On Change To True, Set Turnout, LT1 to Closed	Edit	Next

Logix entry

- Click on Reorder.
- Then choose who is on first.
- Continue picking the next item until you have the new order.
- One action does not wait on another to happen, so the turnout will still activate before the sounds are finished playing. To fix that hit 'Edit' on the turnout entry.

Indirect Layout Control

Logix entry



The image shows two overlapping windows from the JMRI software. The top window is titled 'Edit Conditional' and shows the configuration for a conditional system named 'IX5:WCC1' with the user 'Switch 5 Normal'. It lists antecedent variables in a table:

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete

The bottom window is titled 'Edit Action' and shows the configuration for a consequent action. The 'Action Type' is 'Set Turnout', the 'Change Option' is 'On Change To True', and the 'Turnout Position' is 'Closed'. Below this, a table lists consequent actions:

Action Description		
On Change To True, Play Sound File from file, /usr/local/JMRI/resources/sounds/Code-...	Edit	1
On Change To True, Set Turnout, LT1 to Closed	Edit	Next

Buttons for 'Update Conditional', 'Cancel', and 'Delete Conditional' are visible at the bottom of the 'Edit Action' window.

Logix entry

- Click on Reorder.
- Then choose who is on first.
- Continue picking the next item until you have the new order.

One action does not wait on another to happen, so the turnout will still activate before the sounds are finished playing. To fix that hit 'Edit' on the turnout entry.

- Change 'Set Turnout' to 'Delayed Set Turnout'.

Indirect Layout Control

Logix entry



Logix entry

- Click on Reorder.
- Then choose who is on first.

the next item
new order.
not wait on
so the
activate before
shed playing.

TO FIX THAT HIT 'Edit' on the
turnout entry.

- Change 'Set Turnout' to 'Delayed Set Turnout'.
- Enter '5' for the number of seconds to delay, then 'Update'.

The screenshot shows two overlapping windows from the JMRI software. The top window is titled 'Edit Conditional' and contains the following information:

- Conditional System Name: IX5:WCC1
- Conditional User Name: Switch 5 Normal
- Logical Expression: (empty)
- Antecedent Variables (the 'if' part):

Row	Oper	Neg	State Variable	Description	State	Trigg...		
B1			Sensor LS2	for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete

The bottom window is titled 'Edit Action' and contains the following information:

- Consequent Action
- Action Type: Delayed Set Turnout
- Change Option: On Change To True
- System / User Name: LT1
- Turnout Position: Closed
- Action Data: 5
- Buttons: Update, Cancel, Delete

Below the 'Edit Action' window is the 'Actions' window, which shows a list of consequent actions:

Action Description		
On Change To True, Play Sound File from file, /usr/local/JMRI/resources/sounds/Cod...	Edit	1
On Change To True, Set Turnout, LT1 to Closed	Edit	Next

At the bottom of the 'Actions' window are buttons for 'Add Action', 'Reorder', 'Update Conditional', 'Cancel', and 'Delete Conditional'. An arrow points from the '5' in the 'Action Data' field of the 'Edit Action' window to the 'Reorder' button in the 'Actions' window.

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC1

Conditional User Name Switch 5 Normal

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Closed	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Active	True	<input type="checkbox"/>	Edit	Delete

Add State Variable Check State Variables

Logic Operator
AND

Actions

Consequent Actions (the 'then' part)

Action Description		
On Change To True, Play Sound File from file, /usr/local/JMRI/resources/sounds/Code-...	Edit	Delete
On Change To True, Delayed Set Turnout, LT1 to Closed, after 5 seconds.	Edit	Delete

Add Action Reorder

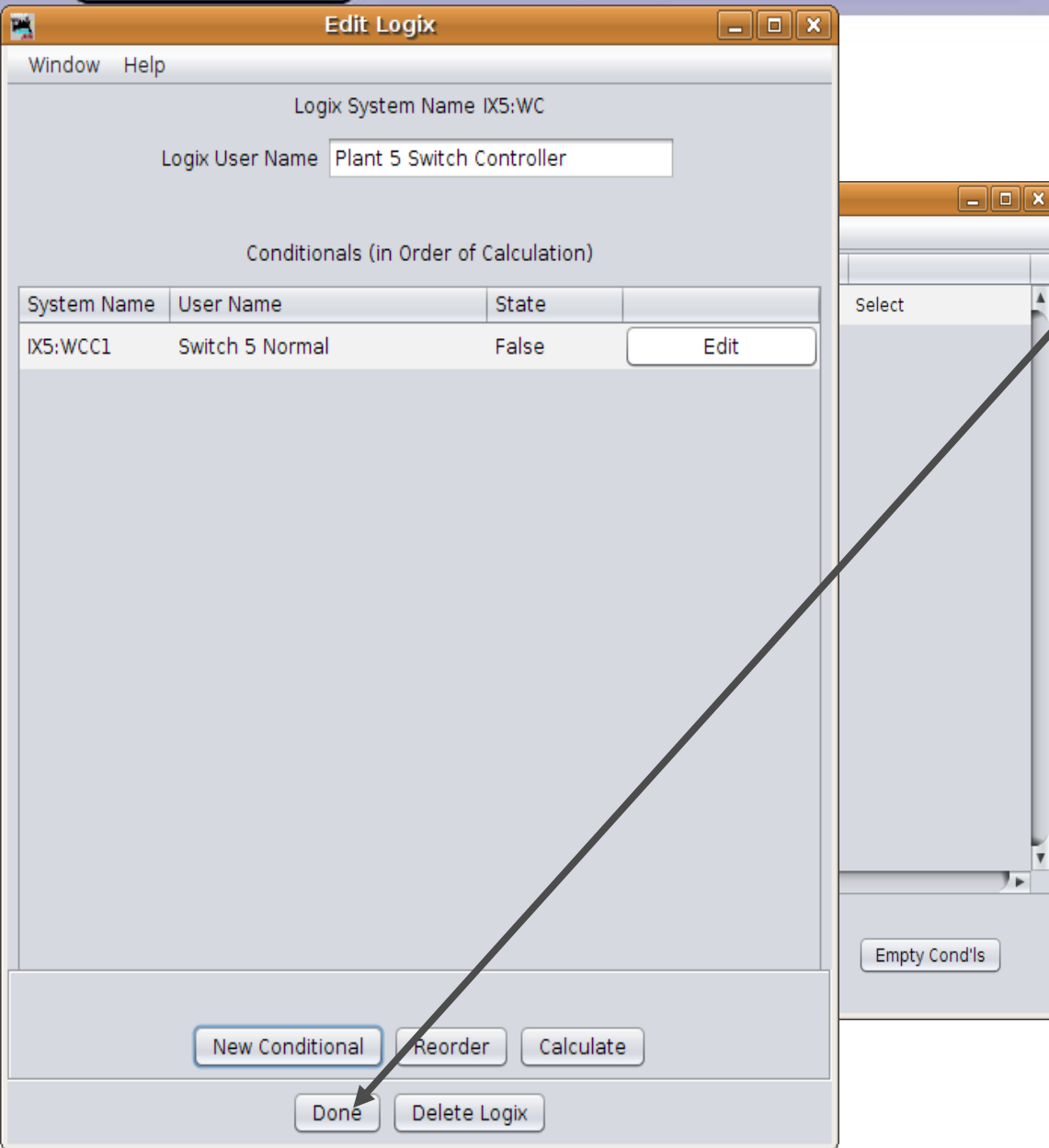
Update Conditional Cancel Delete Conditional

Logix entry

- Click on Reorder.
- Then choose who is on first.
- Continue picking the next item until you have the new order.
- One action does not wait on another to happen, so the turnout will still activate before the sounds are finished playing. To fix that hit 'Edit' on the turnout entry.
- Change 'Set Turnout' to 'Delayed Set Turnout'.
- Enter '5' for the number of seconds to delay, then 'Update'.
- This is our first Conditional completed. 'Update' it to save.

Indirect Layout Control

Logix entry

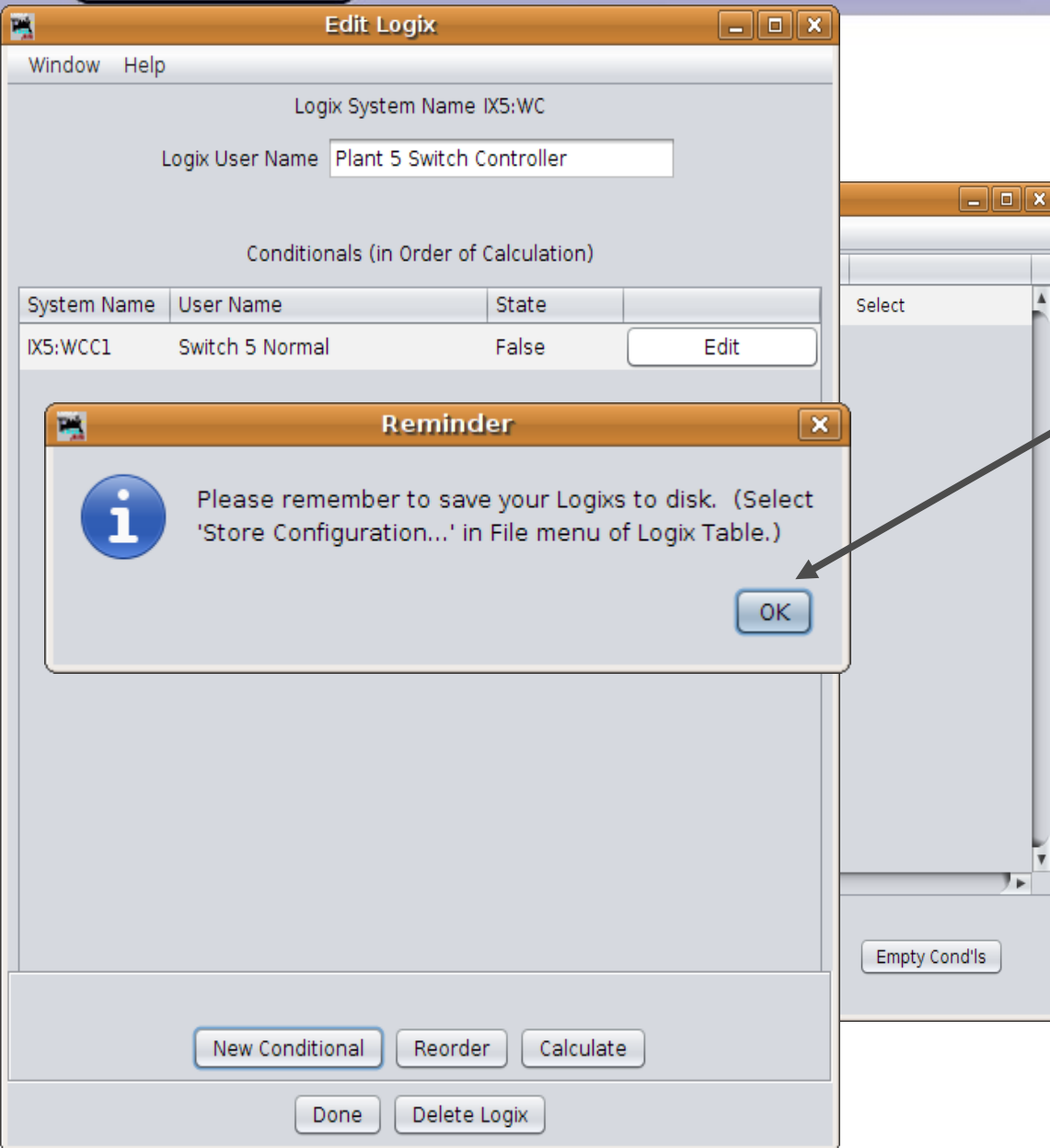


Logix entry

- Click on Done to close and enable the Logix.

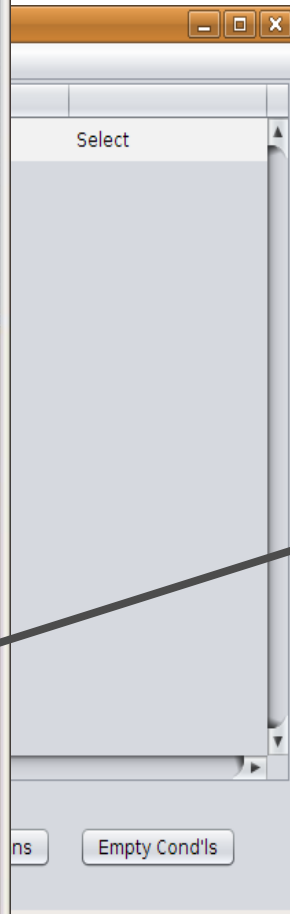
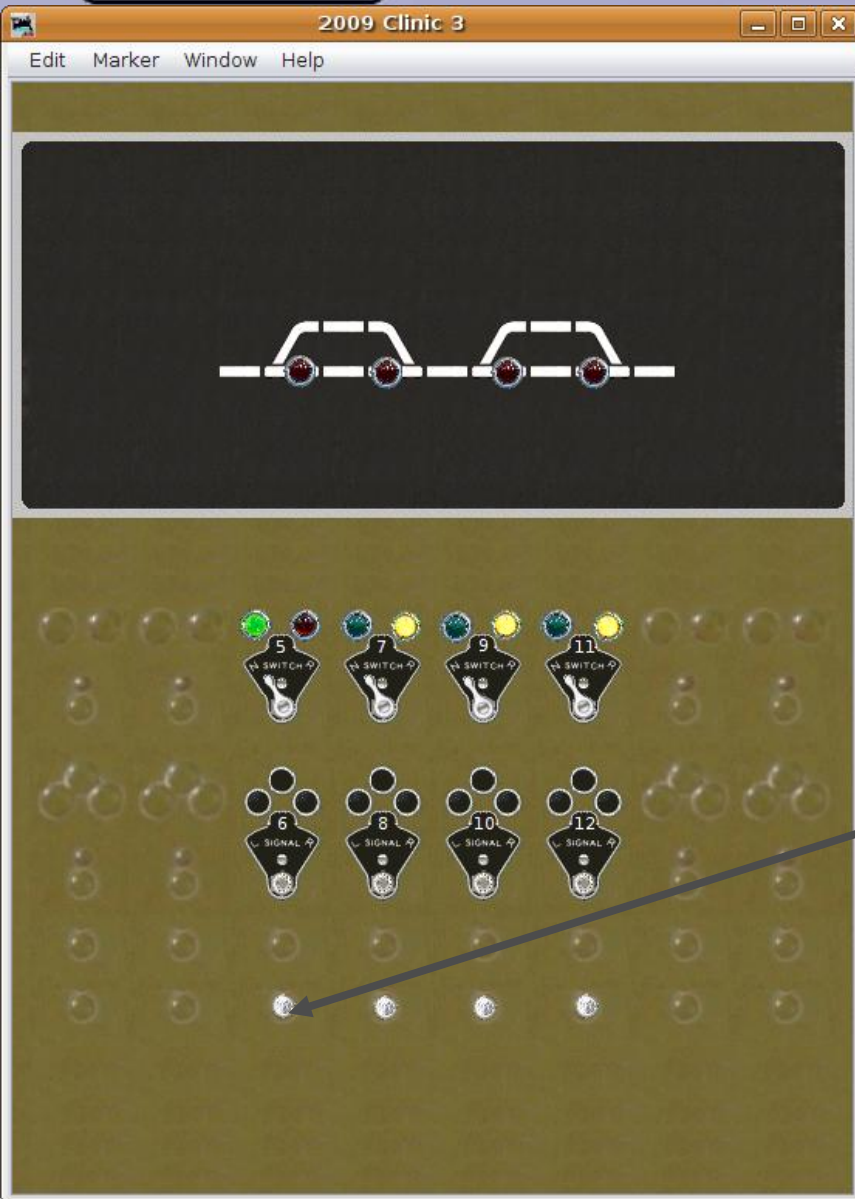
Indirect Layout Control

Logix entry



Logix entry

- Click on Done to close and enable the Logix.
- We are reminded to save our work. The Logix may be saved as stand alone files or with their panels. I prefer mine to be saved with the panel files.

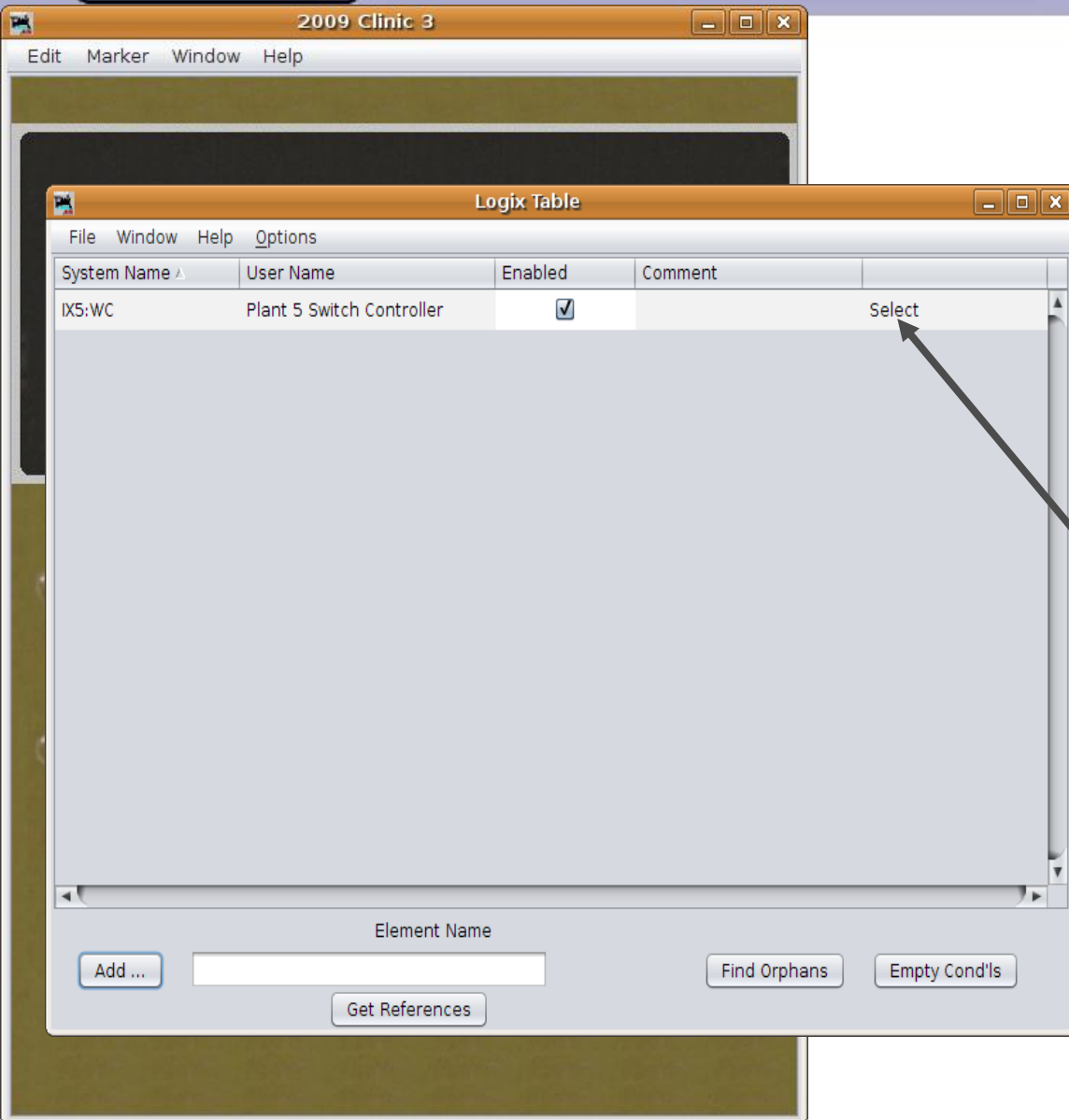


Logix entry

- Click on Done to close and enable the Logix.
- We are reminded to save our work. The Logix may be saved as stand alone files or with their panels. I prefer mine to be saved with the panel files.
- Clicking our code button should send the sound of relays, then 'close' the turnout, but only if the OS is clear and the turnout isn't already closed.

Indirect Layout Control

Logix entry

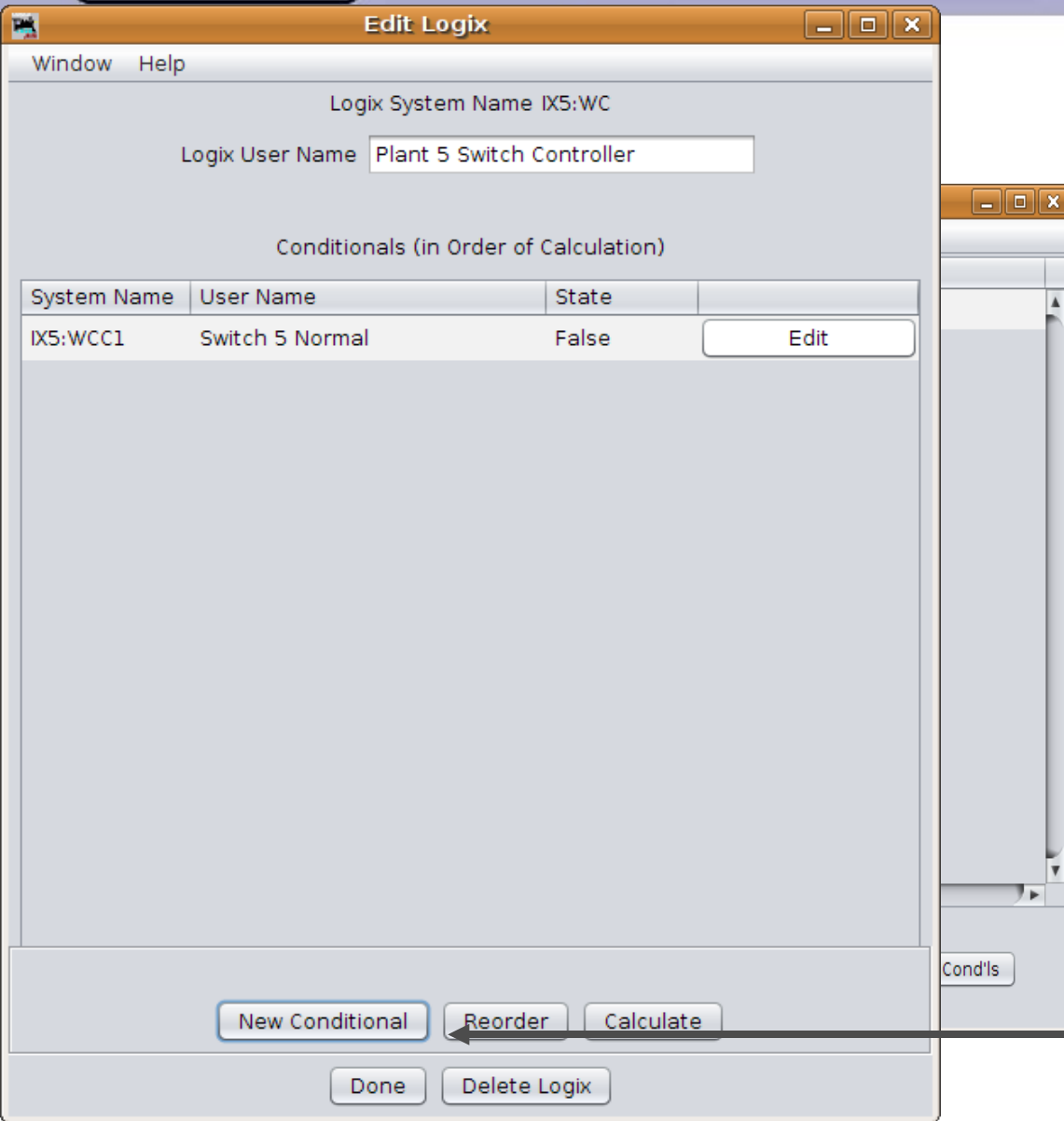


Logix entry

- Click on Done to close and enable the Logix.
- We are reminded to save our work. The Logix may be saved as stand alone files or with their panels. I prefer mine to be saved with the panel files.
- Clicking our code button should send the sound of relays, then 'close' the turnout, but only if the OS is clear and the turnout isn't already closed.
- Select Edit in order to add the 'thrown' conditional.

Indirect Layout Control

Logix entry



Logix entry

- Click on Done to close and enable the Logix.
- We are reminded to save our work. The Logix may be saved as stand alone files or with their panels. I prefer mine to be saved with the panel files.
- Clicking our code button should send the sound of relays, then 'close' the turnout, but only if the OS is clear and the turnout isn't already closed.
- Select Edit in order to add the 'thrown' conditional.
- Then 'New Conditional'.

Indirect Layout Control

Logix entry



Logix entry

- Enter 'Switch 5 Reverse' then all the variables and conditionals for the other directions actions.

Window Help

Conditional System Name IX5:WCC2

Conditional User Name

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Thrown	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Inactive	False	<input type="checkbox"/>	Edit	Delete

Add State Variable Check State Variables

Logic Operator

AND ▼

Actions

Consequent Actions (the 'then' part)

Action Description		
On Change To True, Play Sound File from file, /usr/local/JMRI/resources/sounds/Code-...	Edit	Delete
On Change To True, Delayed Set Turnout, LT1 to Thrown, after 5 seconds.	Edit	Delete

Add Action Reorder

Update Conditional Cancel Delete Conditional

Indirect Layout Control

Logix entry



Window Help

Conditional System Name IX5:WCC2

Conditional User Name

Logical Expression:

Antecedent Variables (the 'if' part)

Row	Oper	Neg	State Variable Description	State	Trigg...		
R1			Sensor, LS2, for Sensor Inactive	True	<input type="checkbox"/>	Edit	Delete
R2	AND		Sensor, IS6:CB, for Sensor Active	False	<input checked="" type="checkbox"/>	Edit	Delete
R3	AND	NOT	Turnout, LT1, for Turnout Thrown	True	<input type="checkbox"/>	Edit	Delete
R4	AND		Sensor, IS5:WL, for Sensor Inactive	False	<input type="checkbox"/>	Edit	Delete

Logic Operator

AND

Actions

Consequent Actions (the 'then' part)

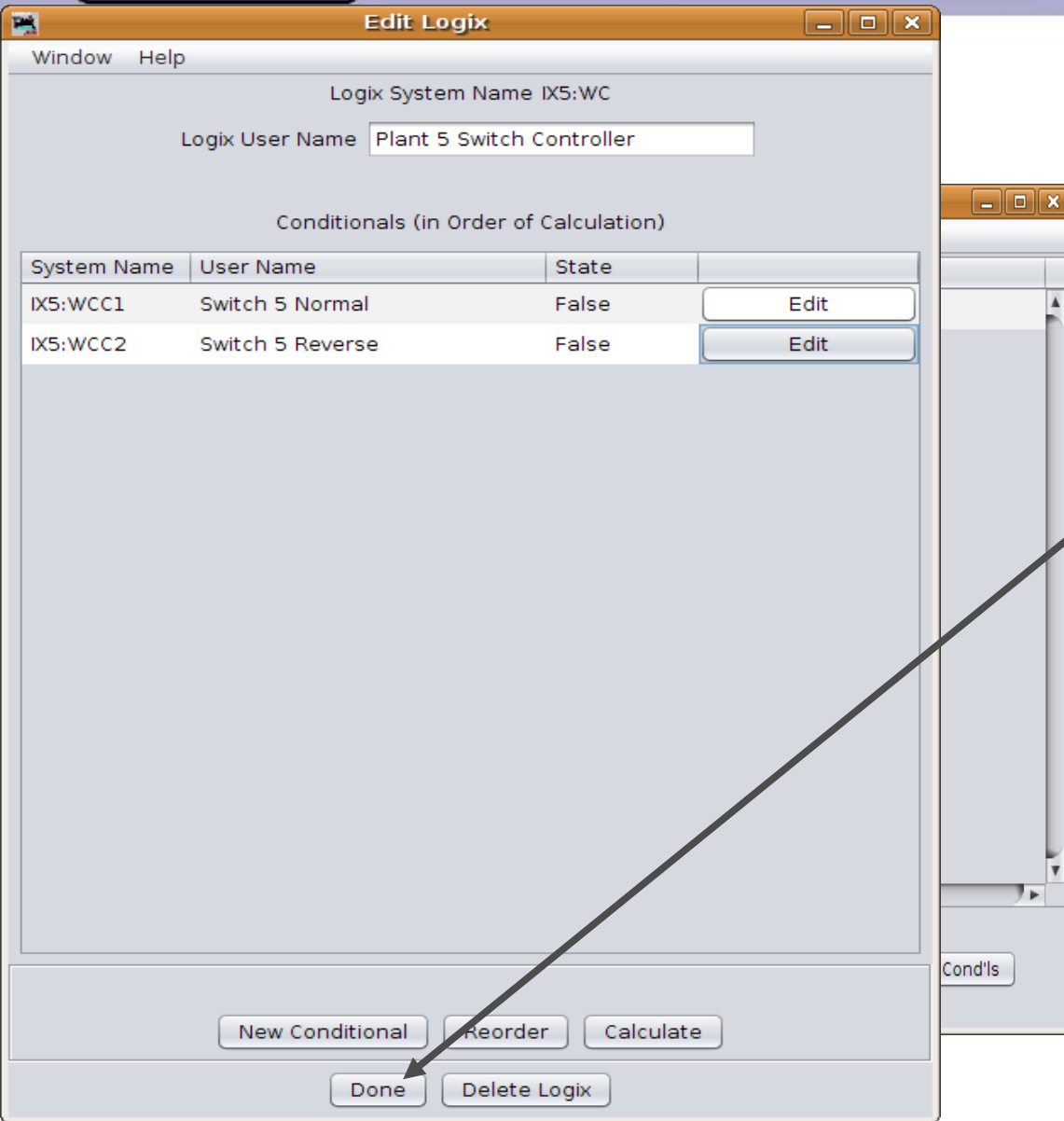
Action Description		
On Change To True, Play Sound File from file, /usr/local/JMRI/resources/sounds/Code-...	Edit	Delete
On Change To True, Delayed Set Turnout, LT1 to Thrown, after 5 seconds.	Edit	Delete

Logix entry

- Enter 'Switch 5 Reverse' then all the variables and conditionals for the other directions actions.
- Update the new conditional.

Indirect Layout Control

Logix entry

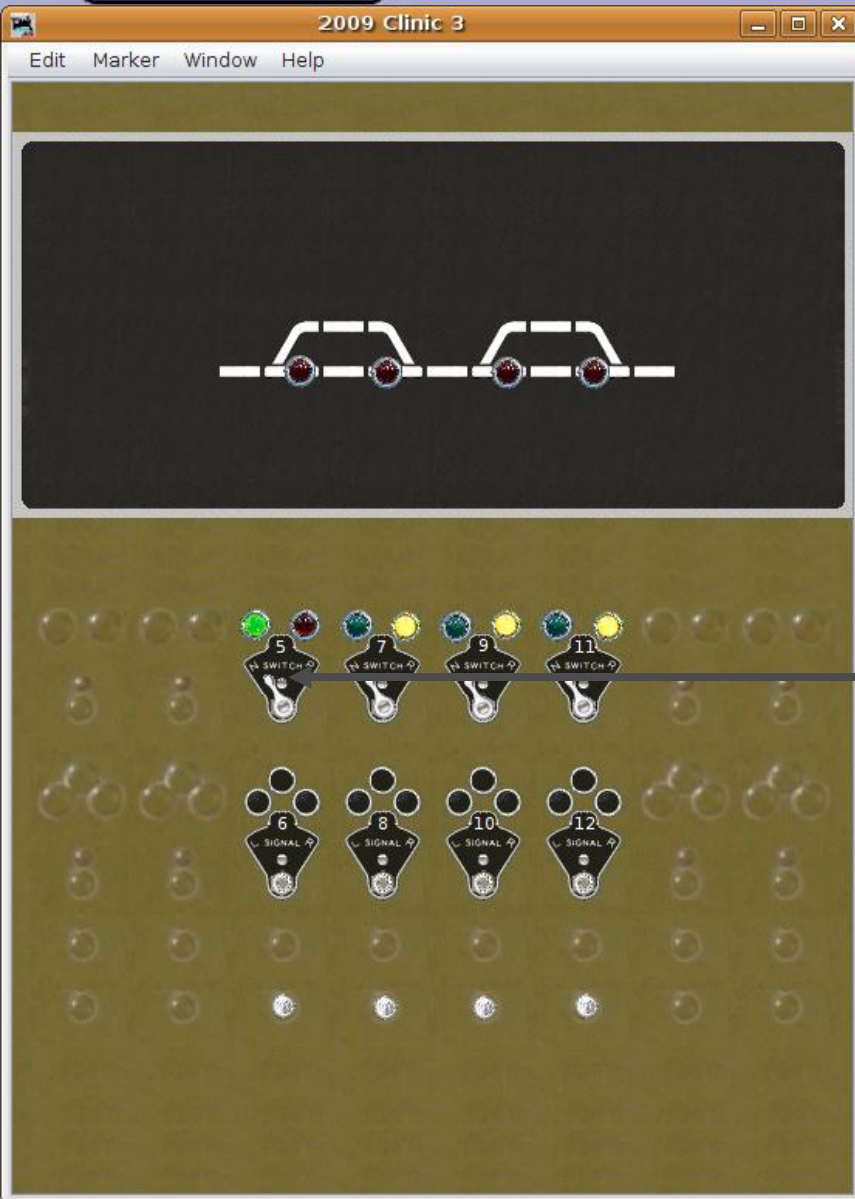


Logix entry

- Enter 'Switch 5 Reverse' then all the variables and conditionals for the other directions actions.
- Update the new conditional.
- And click 'Done'.

Indirect Layout Control

Logix entry

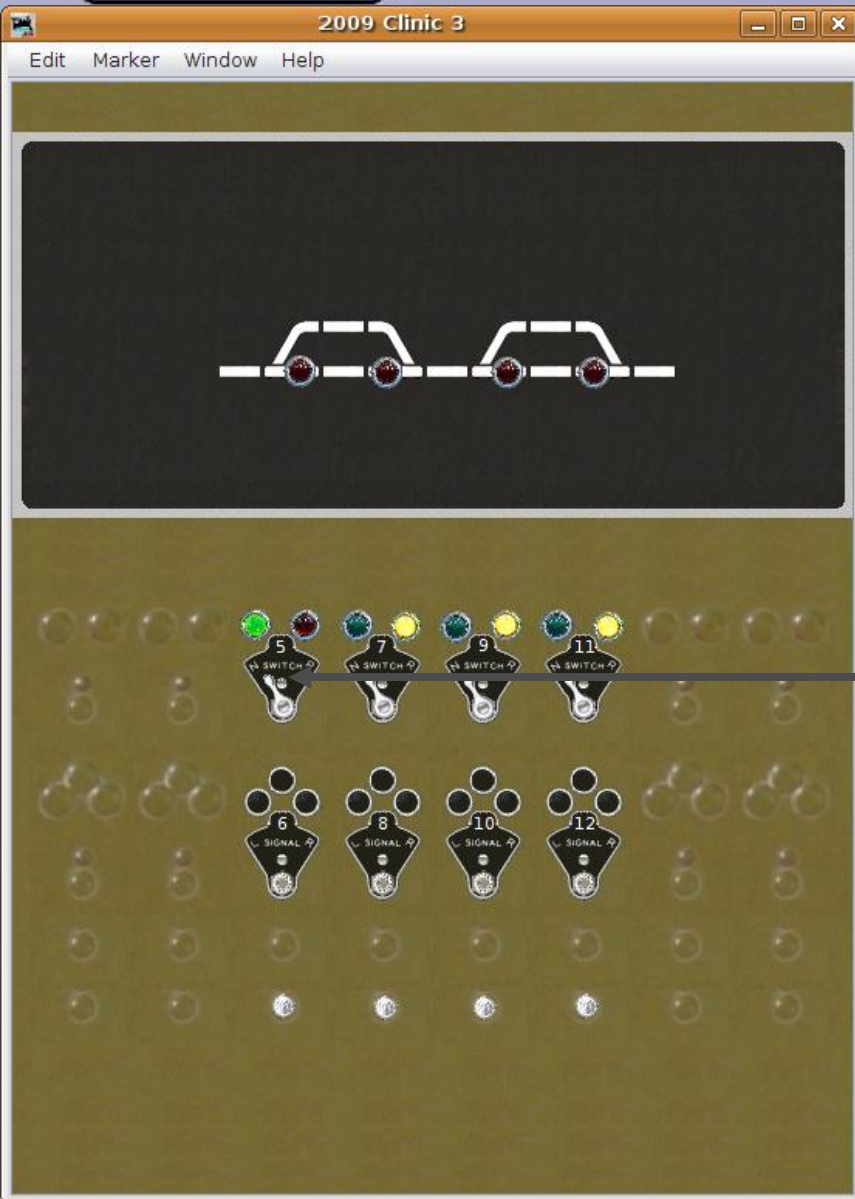


Logix entry

- Enter 'Switch 5 Reverse' then all the variables and conditionals for the other directions actions.
- Update the new conditional.
- And click 'Done'.
- Test again and we should have interlocked control of the first turnout. The problem is that there is no feedback between the turnout position and the indicator jewels.

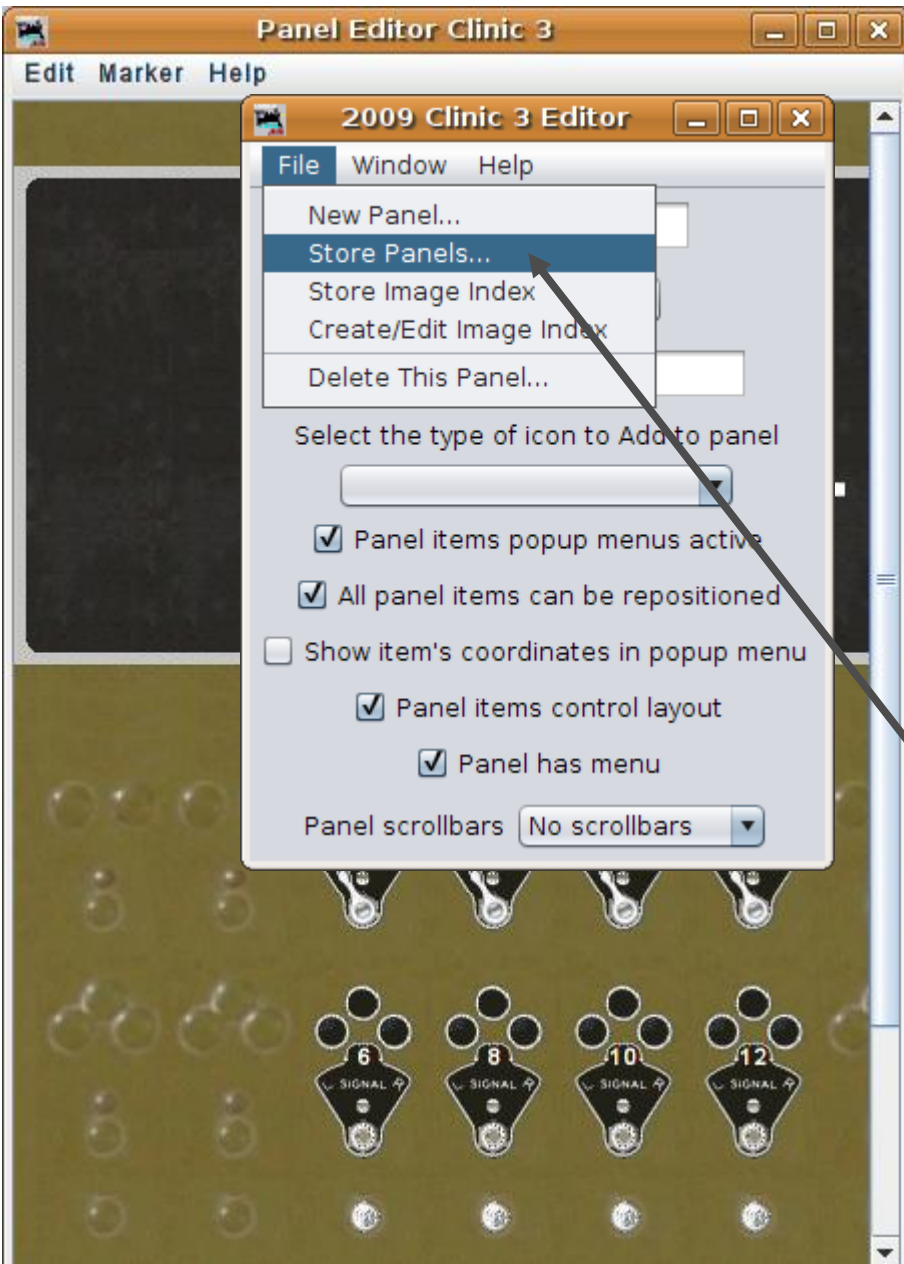
Indirect Layout Control

Logix entry



Logix entry

- Enter 'Switch 5 Reverse' then all the variables and conditionals for the other directions actions.
- Update the new conditional.
- And click 'Done'.
- Test again and we should have interlocked control of the first turnout. The problem is that there is no feedback between the turnout position and the indicator jewels.
- In our next clinic we will cover editing options and test our panel.

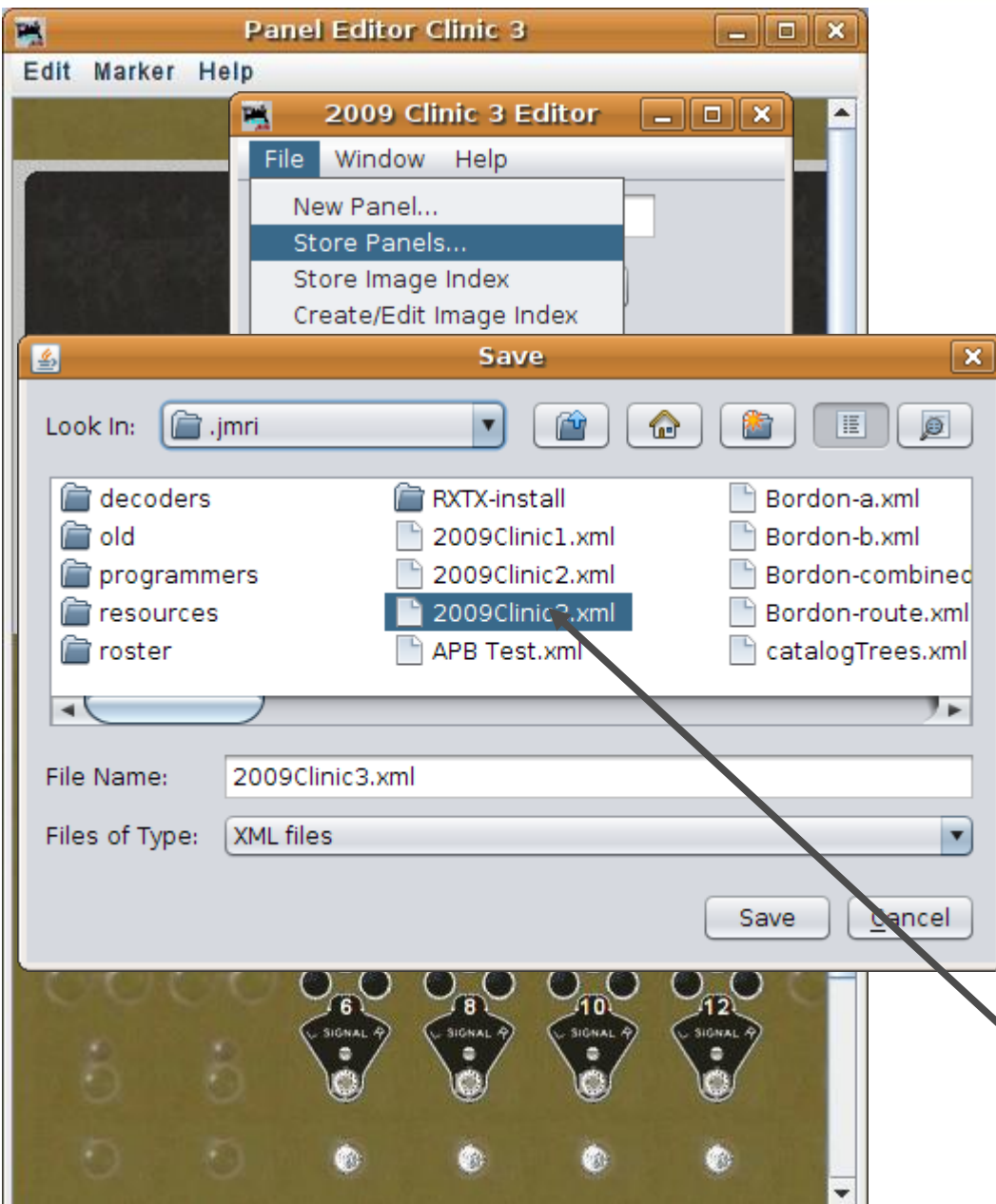


Logix entry

- Enter 'Switch 5 Reverse' then all the variables and conditionals for the other directions actions.
- Update the new conditional.
- And click 'Done'.
- Test again and we should have interlocked control of the first turnout. The problem is that there is no feedback between the turnout position and the indicator jewels.
- In our next clinic we will cover editing options and test our panel.
- Save our work.

Indirect Layout Control

Logix entry



Logix entry

- Enter 'Switch 5 Reverse' then all the variables and conditionals for the other directions actions.
- Update the new conditional.
- And click 'Done'.
- Test again and we should have interlocked control of the first turnout. The problem is that there is no feedback between the turnout position and the indicator jewels.
- In our next clinic we will cover editing options and test our panel.
- Save our work.
- As 2009Clinic3.xml

Introduction to PanelPro



This completes Clinic 3.

The next clinic will start with basic new editing options for Logix and Panels.

Next we will cover basic ABS signaling using SSL. (Simple Signal Logic)

These clinic files will be available at our web site.

<http://www.rr-cirkits.com/Clinics/Clinics.html>

Versions from previous years clinics are also available there for your convenience.