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.

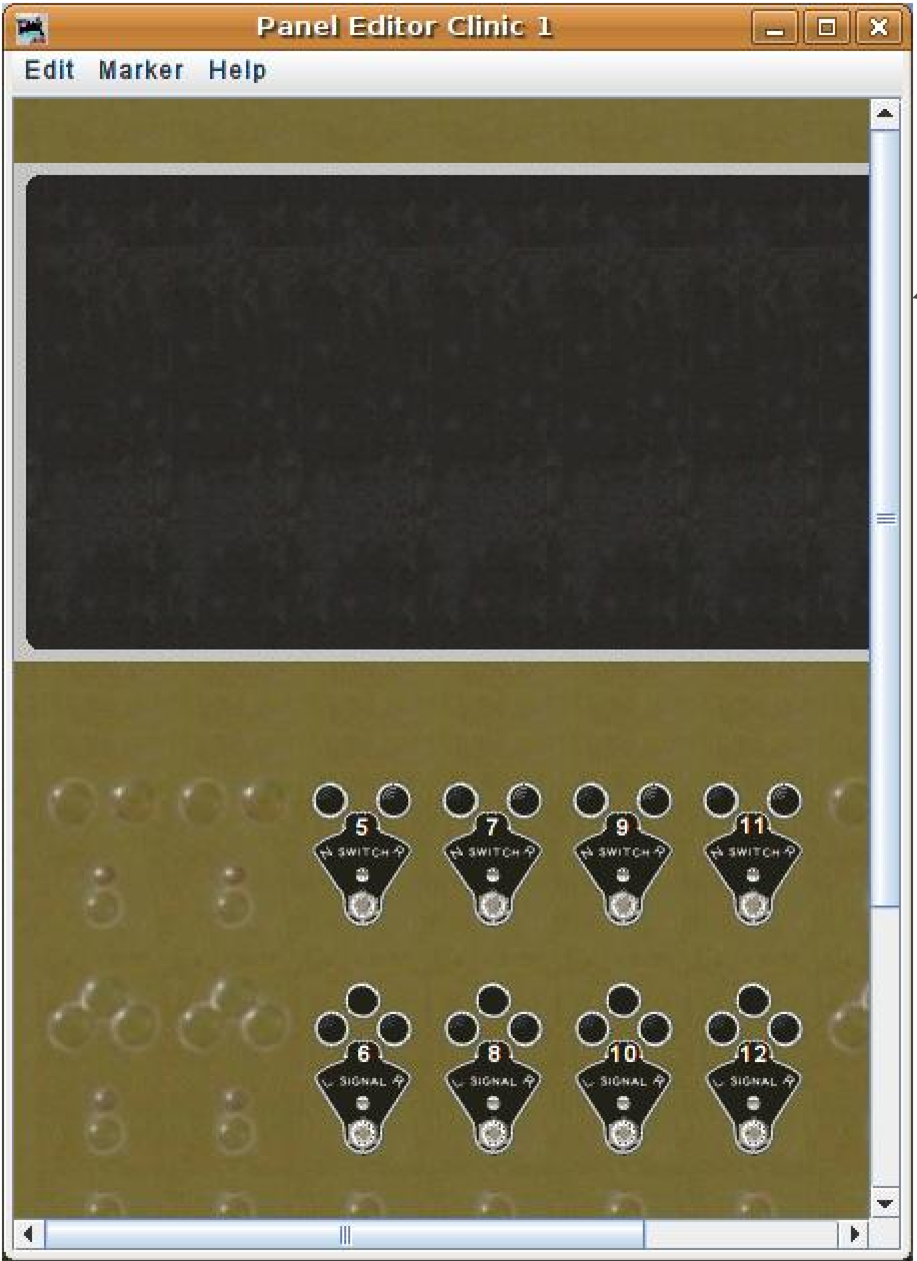
Logix are functionally similar to industrial ladder logic in that they do not have any parenthetical structure. Therefore there is no logical 'OR' function. To do an OR you simply create multiple different Logix for each different conditional, or else invert the sense of items and use the 'NOT AND' instead.

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 that 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.

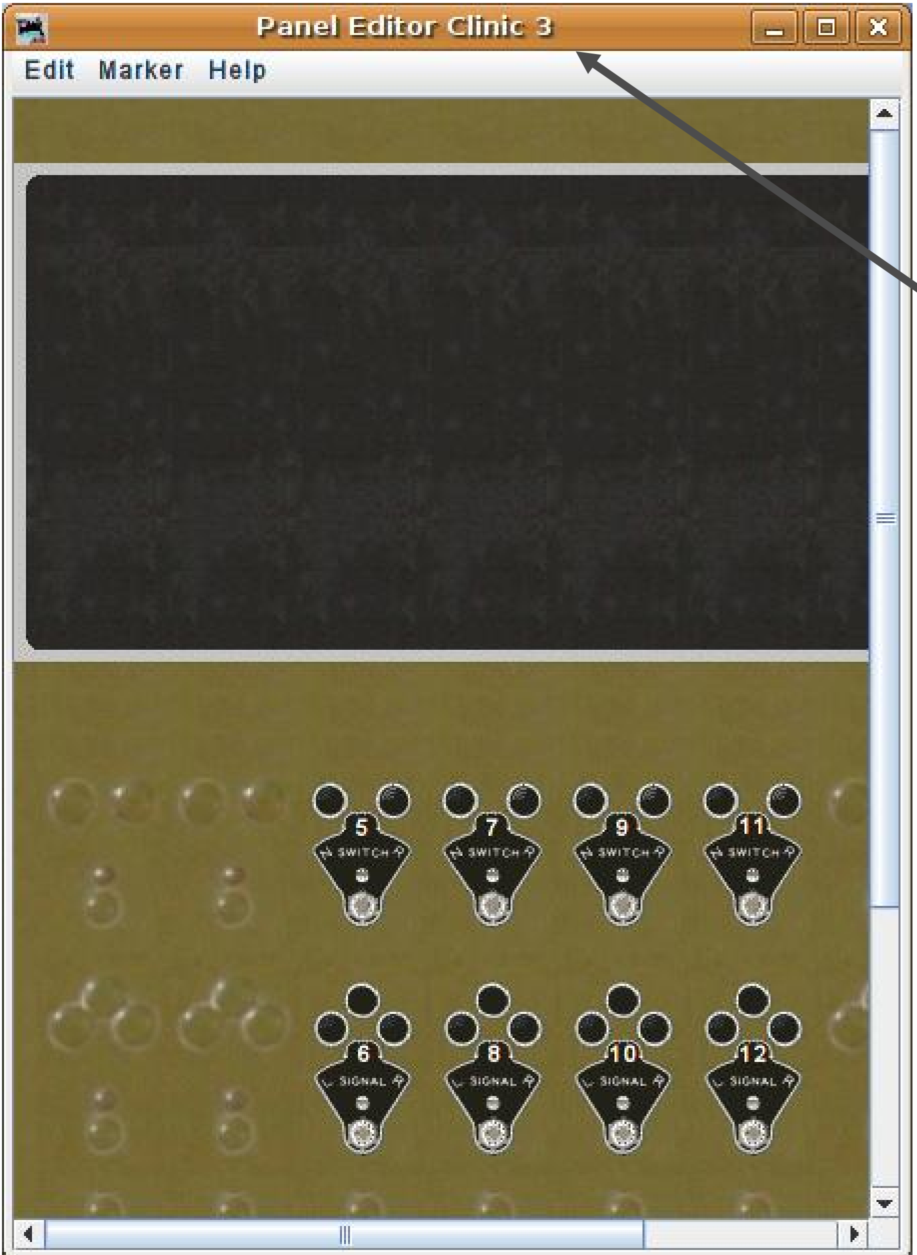
On the prototype railroads it is not allowable to have remote control of turnouts without some fairly reliable method of knowing the position of the points and preventing them from ever being changed while a train is crossing them. (or about to) Now that we mention it, these are pretty good things to 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 place with no scenery)

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.



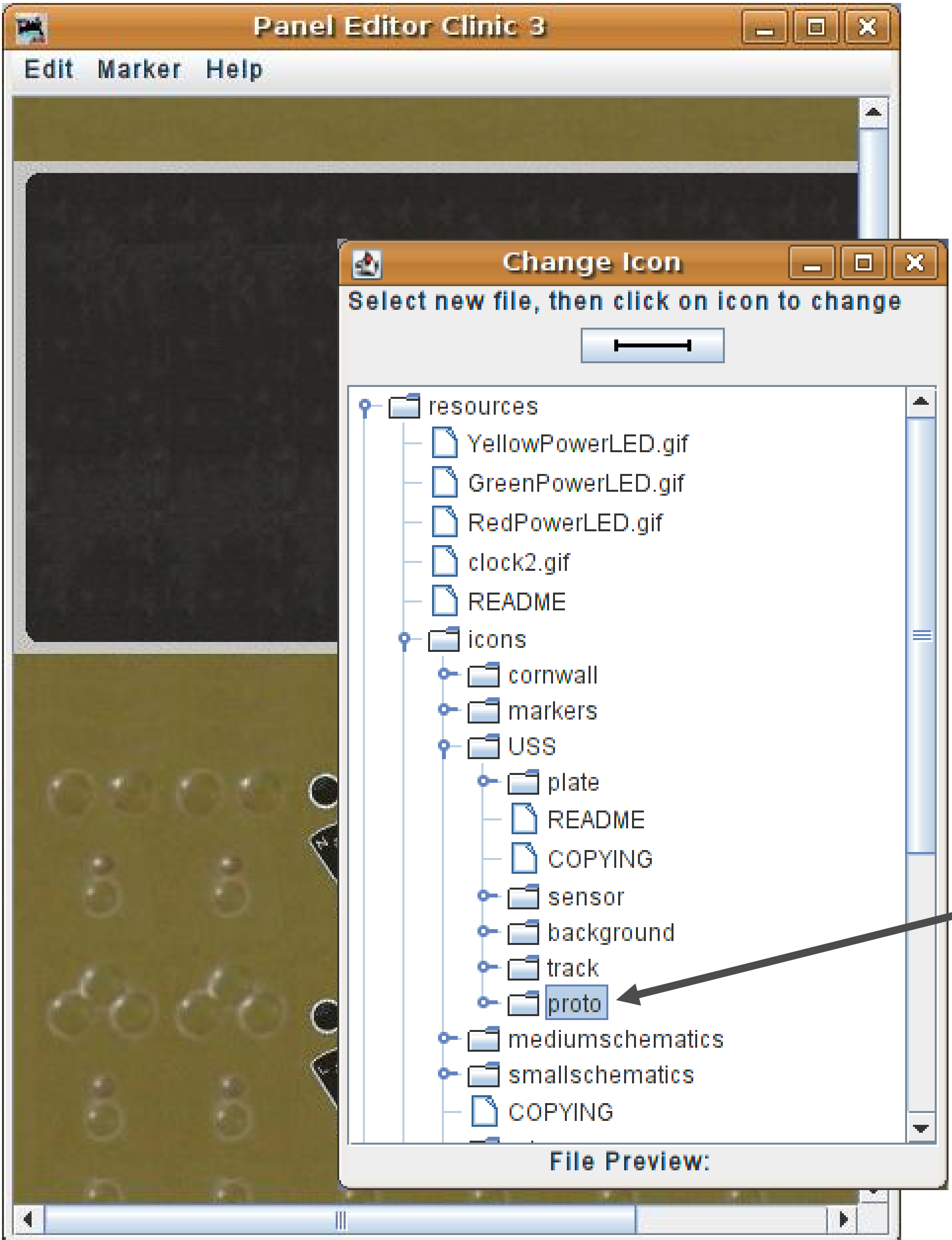
Indirect Layout Control

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



Indirect Layout Control

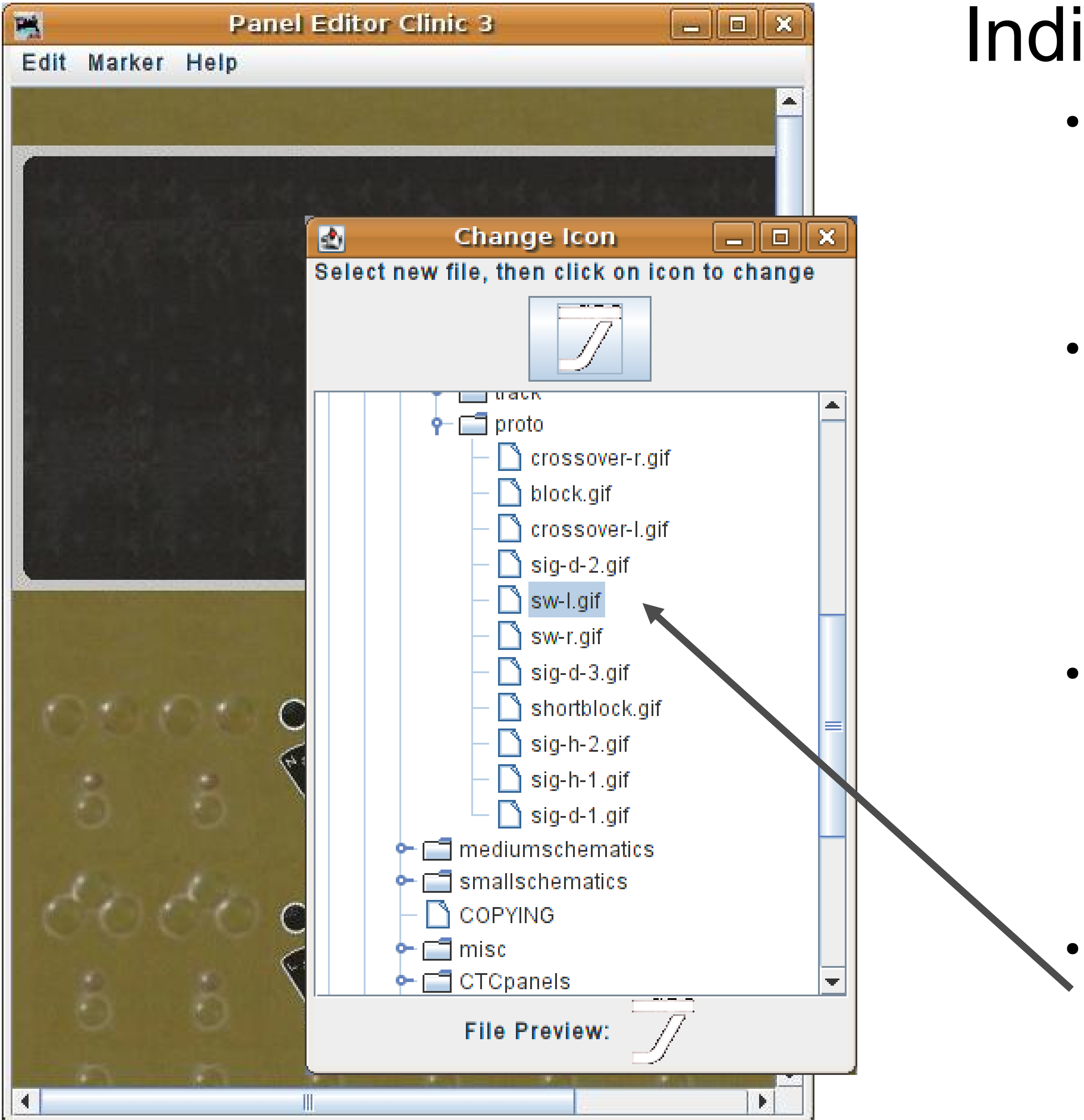
- First lets load in the basic panel background we made in clinic #1 then rename it and save it as clinic #3
- You will be expected to know how to do the basic operations already covered in previous sessions, so I am not going to repeat the detail of each operation as we move along.



Select new file, then click on icon to change

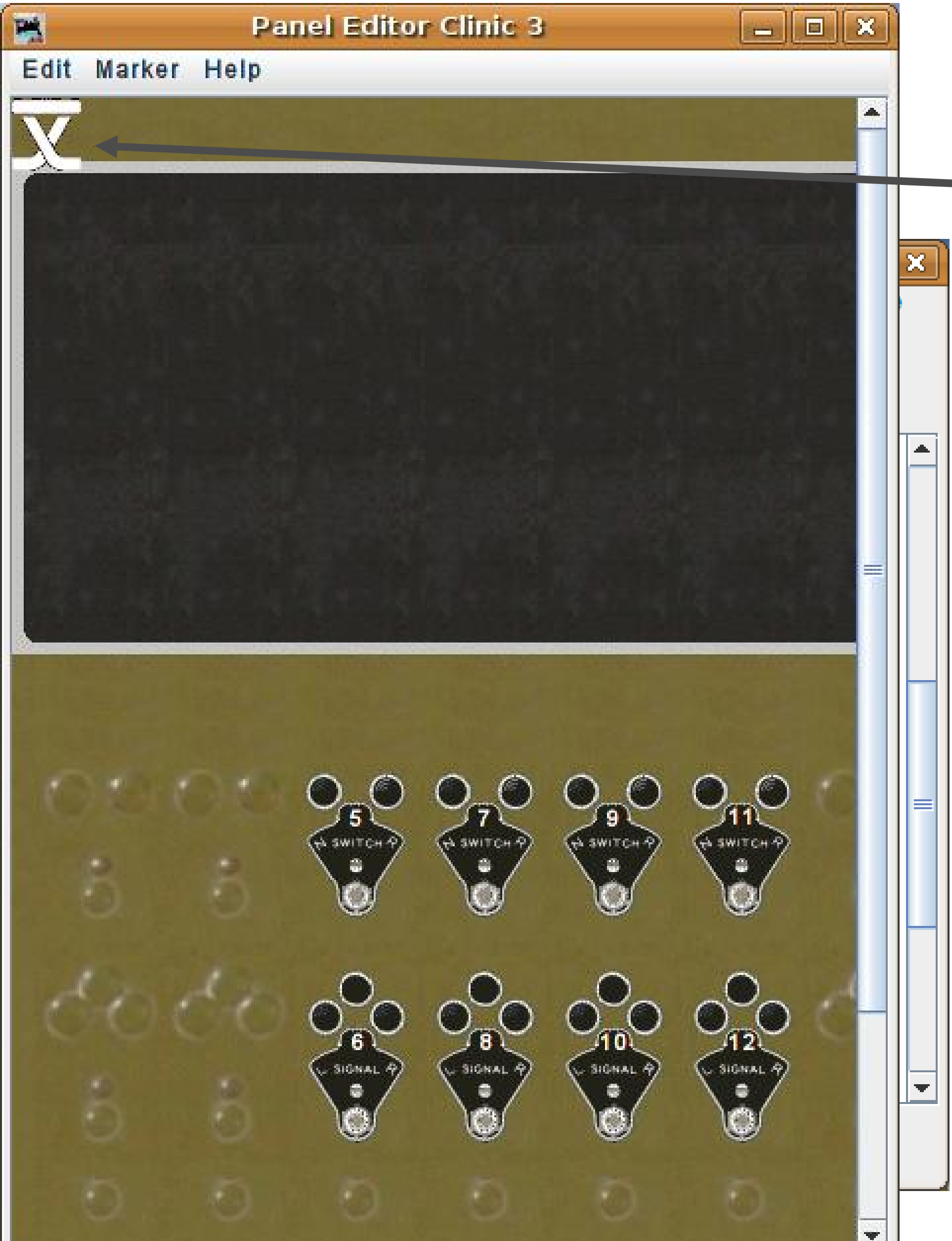
Indirect Layout Control

- First lets load in the basic panel background we made in clinic #1 then rename it and save it as clinic #3
- You will be expected to know how to do 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 (unrestored) classic era prototype US&S CTC machine.



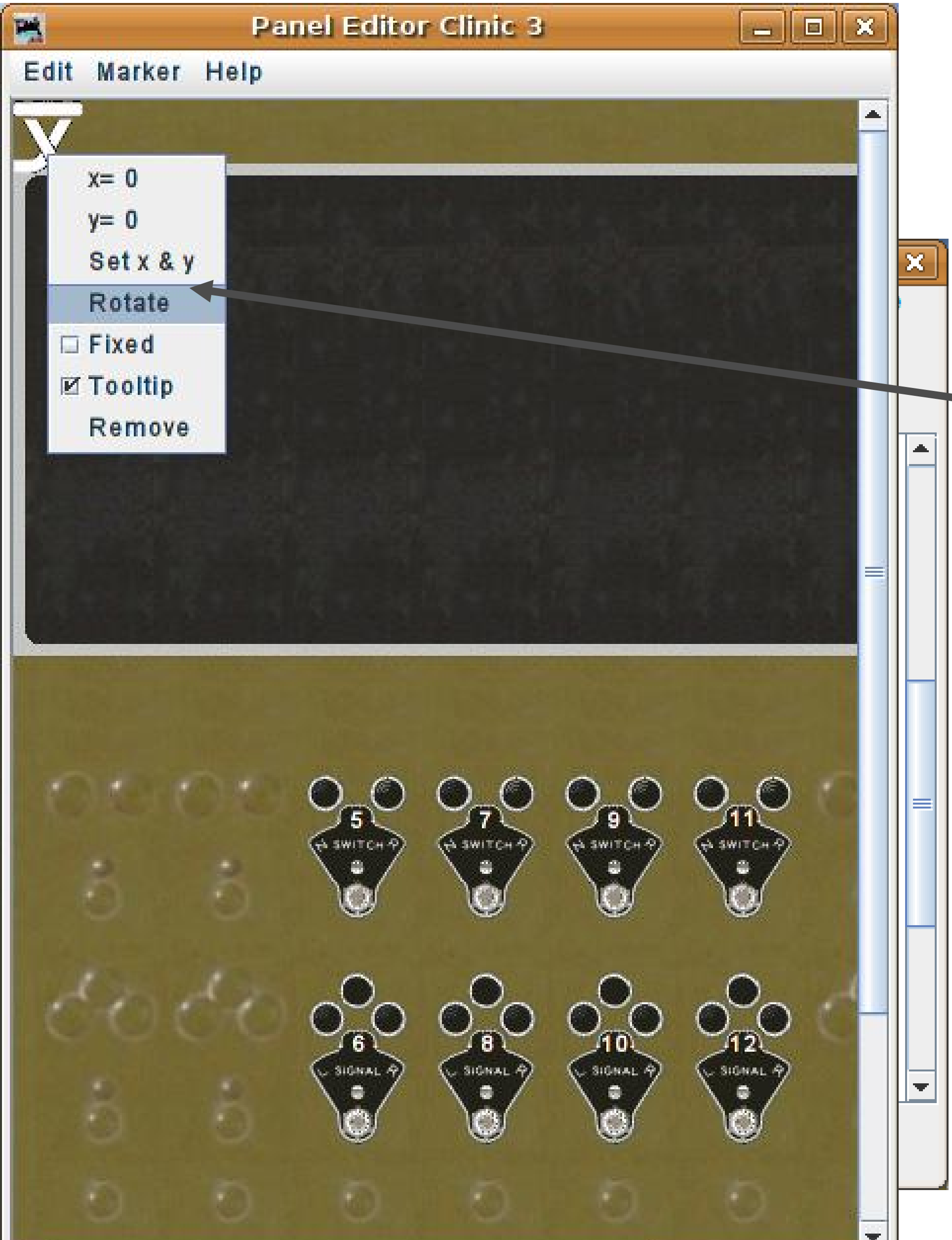
Indirect Layout Control

- First lets load in the basic panel background we made in clinic #1 then rename it and save it as clinic #3
- You will be expected to know how to do 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 (unrestored) classic era prototype US&S CTC machine.
- These few images are not designed for animation, but for making a more realistic panel.



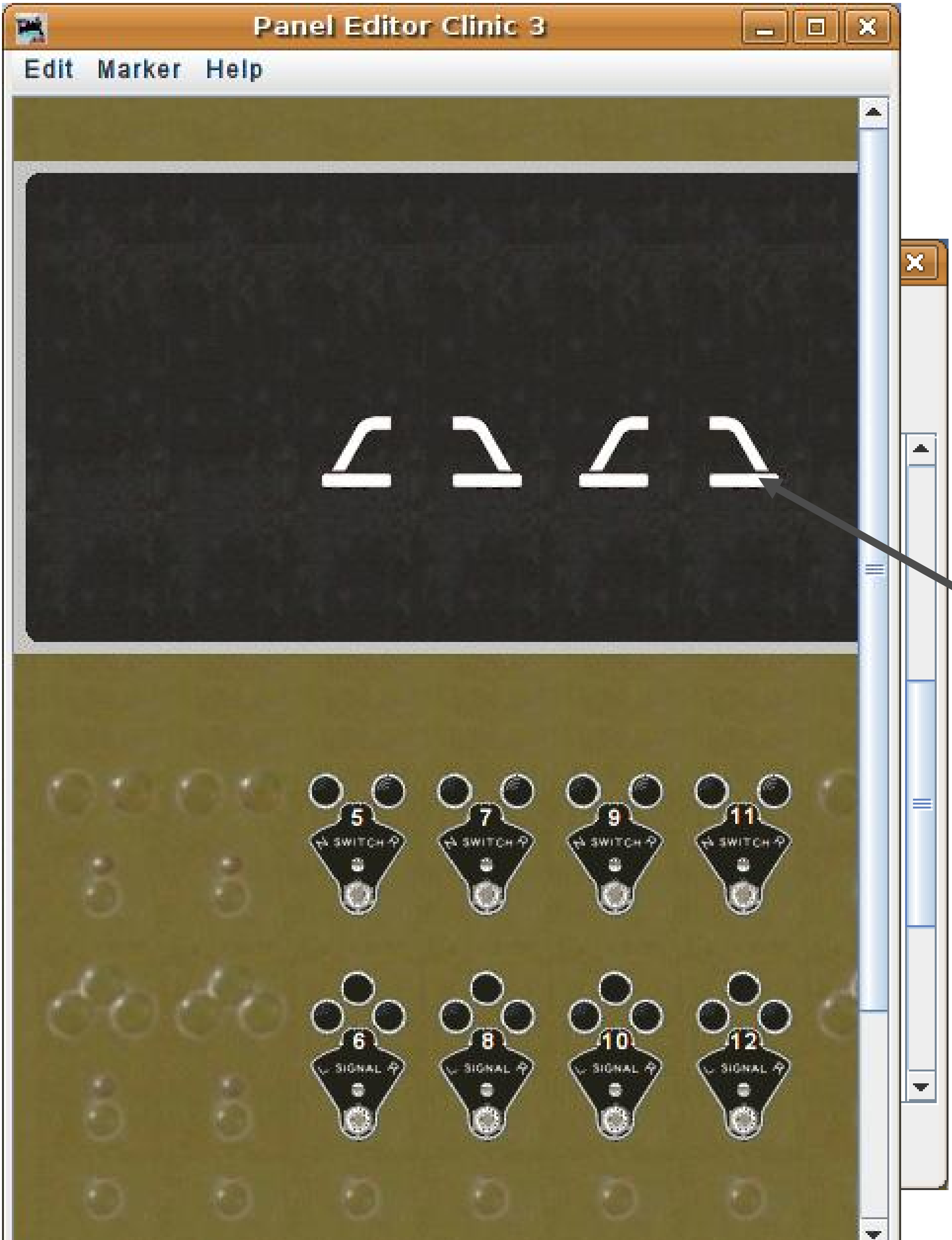
Indirect Layout Control

- Use the 'Add icon:' button to add two left (sw-l.gif) and two right (sw-r.gif) turnout icons to our panel.



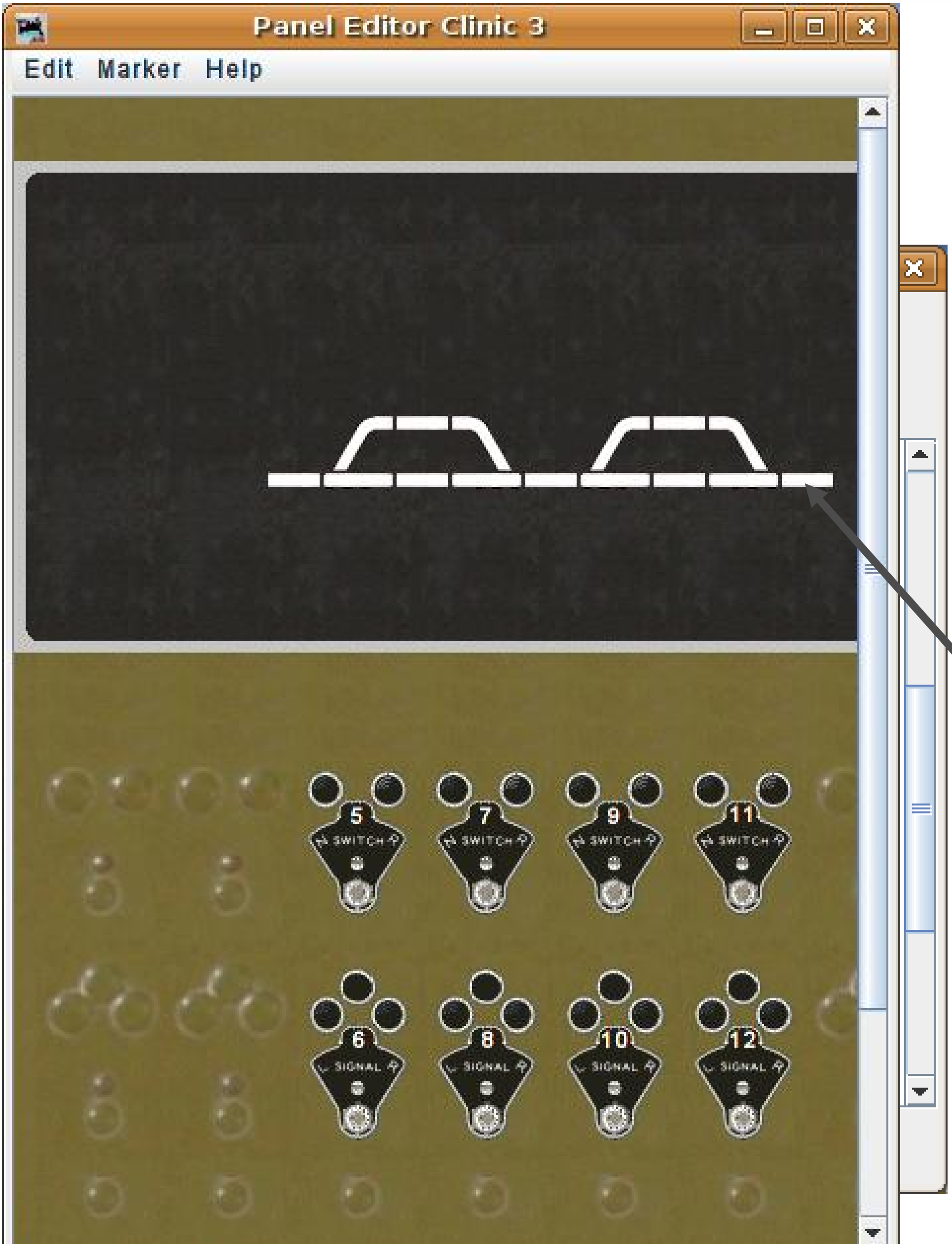
Indirect Layout Control

- Use the 'Add icon:' button to add two left (sw-l.gif) and two right (sw-r.gif) 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°.



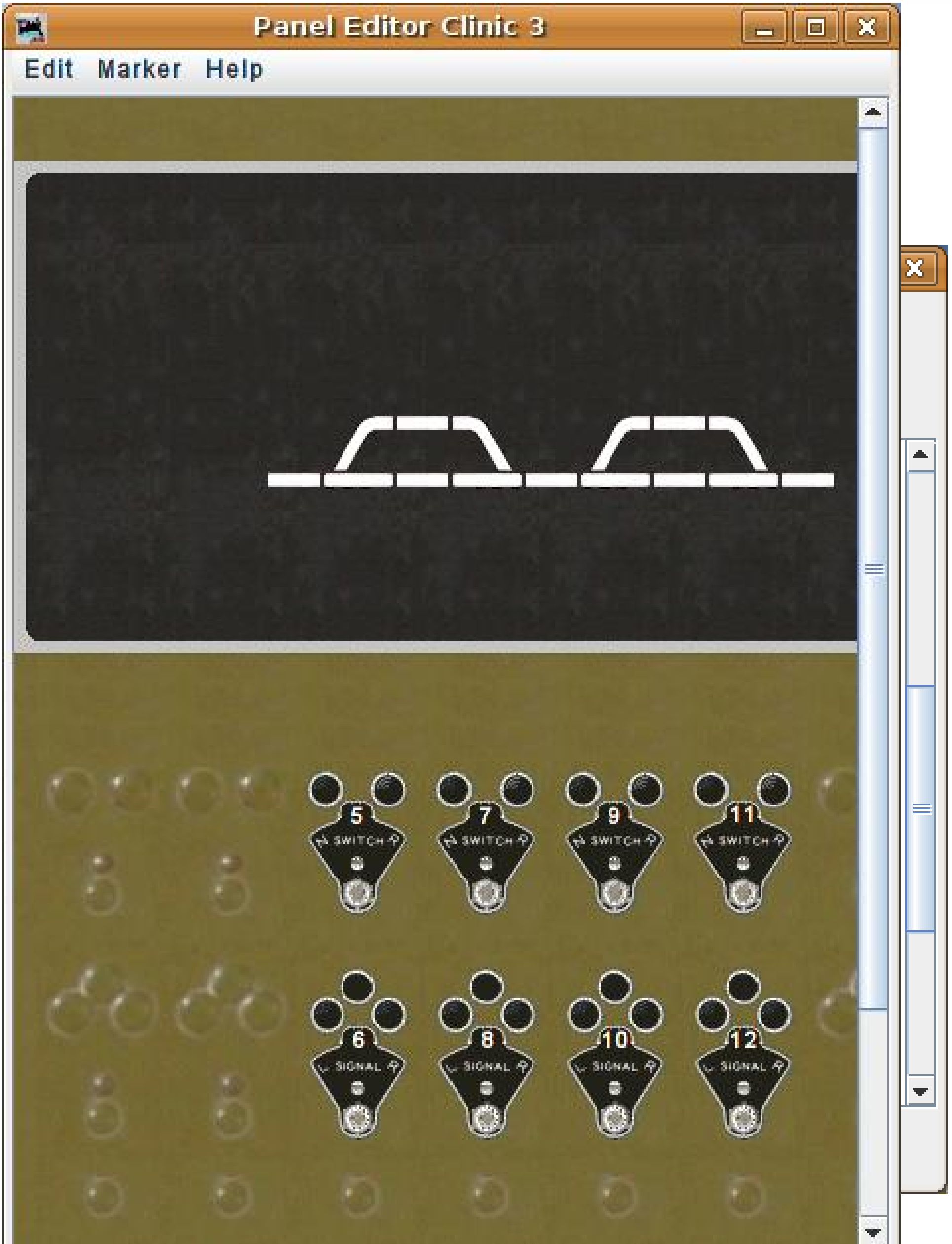
Indirect Layout Control

- Use the 'Add icon:' button to add two left (sw-l.gif) and two right (sw-r.gif) 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°.
- Each icon is 'rotated' twice to face up, and then positioned on the panel.



Indirect Layout Control

- Use the 'Add icon:' button to add two left (sw-l.gif) and two right (sw-r.gif) 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°.
- Each icon is 'rotated' twice to face up, and then positioned on the panel.
- Add in some straight track images (block.gif) and we have our basic track plan drawn.

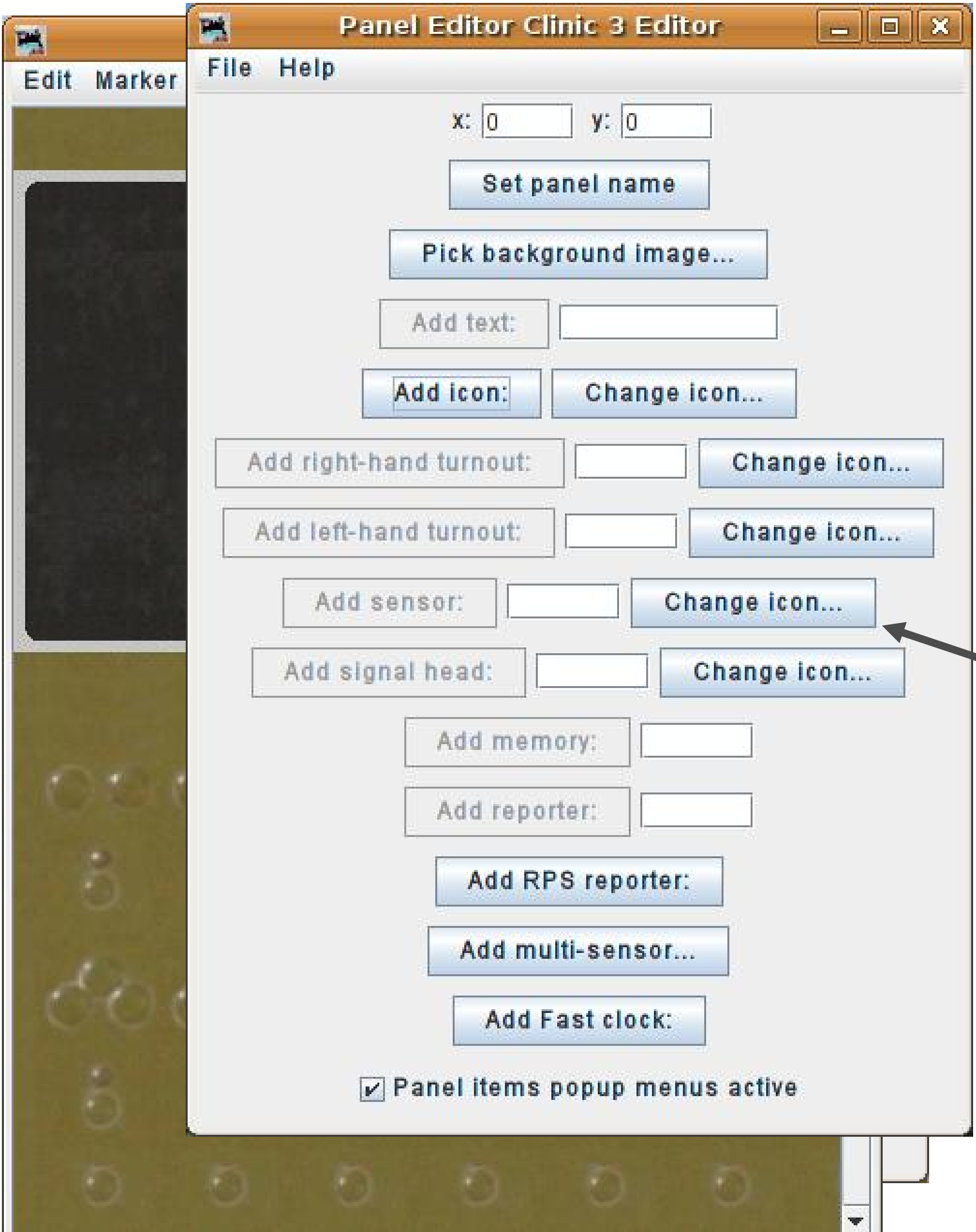


Occupancy sensors

- 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')

Indirect Layout Control

Sensor images



x: 0 y: 0

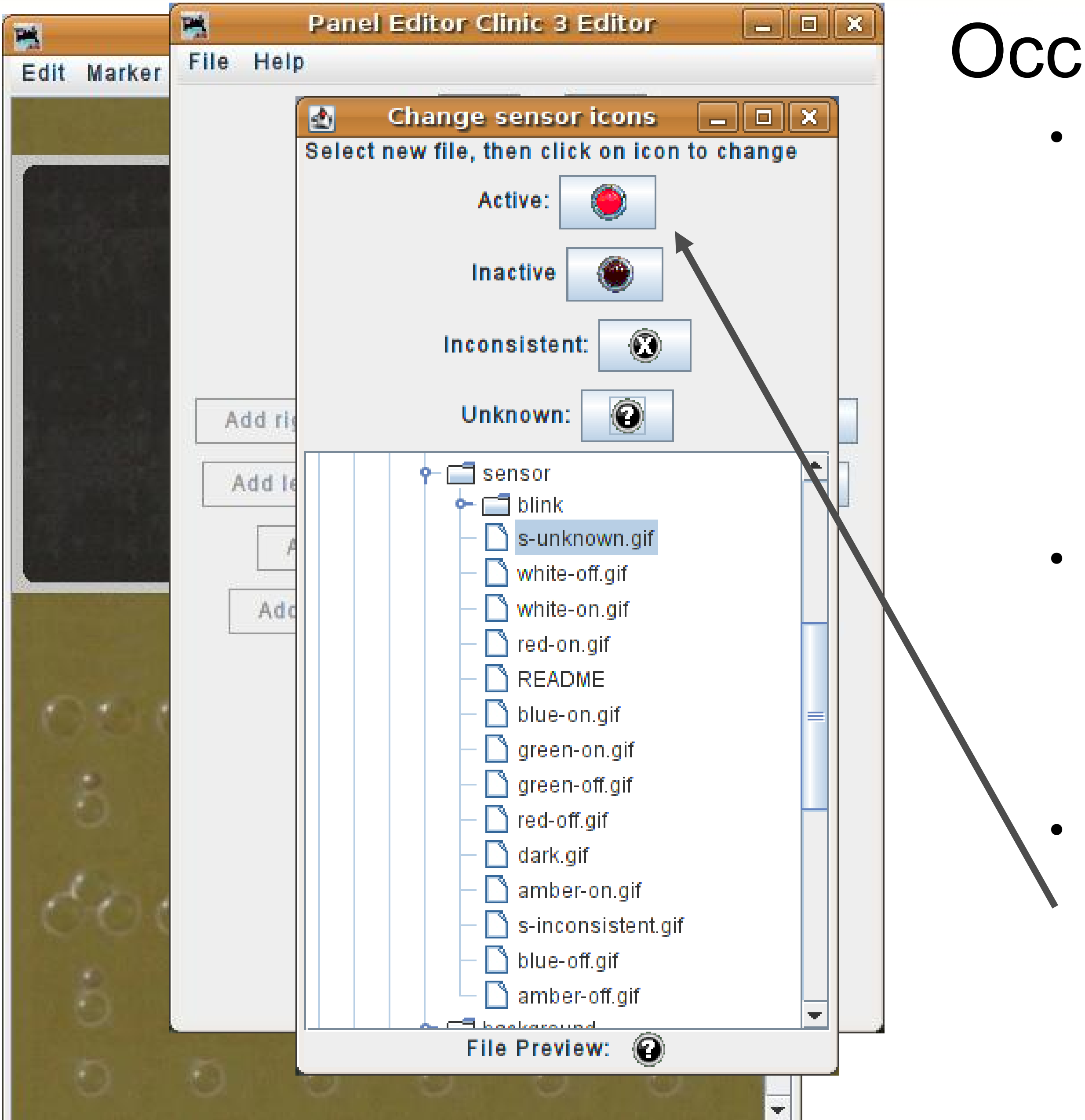
☑ Panel items popup menus active

Occupancy sensors

- 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')
- In clinic 2 we added active icons for our turnouts. Now we will do the same for our occupancy sensors. Start by changing the icons.

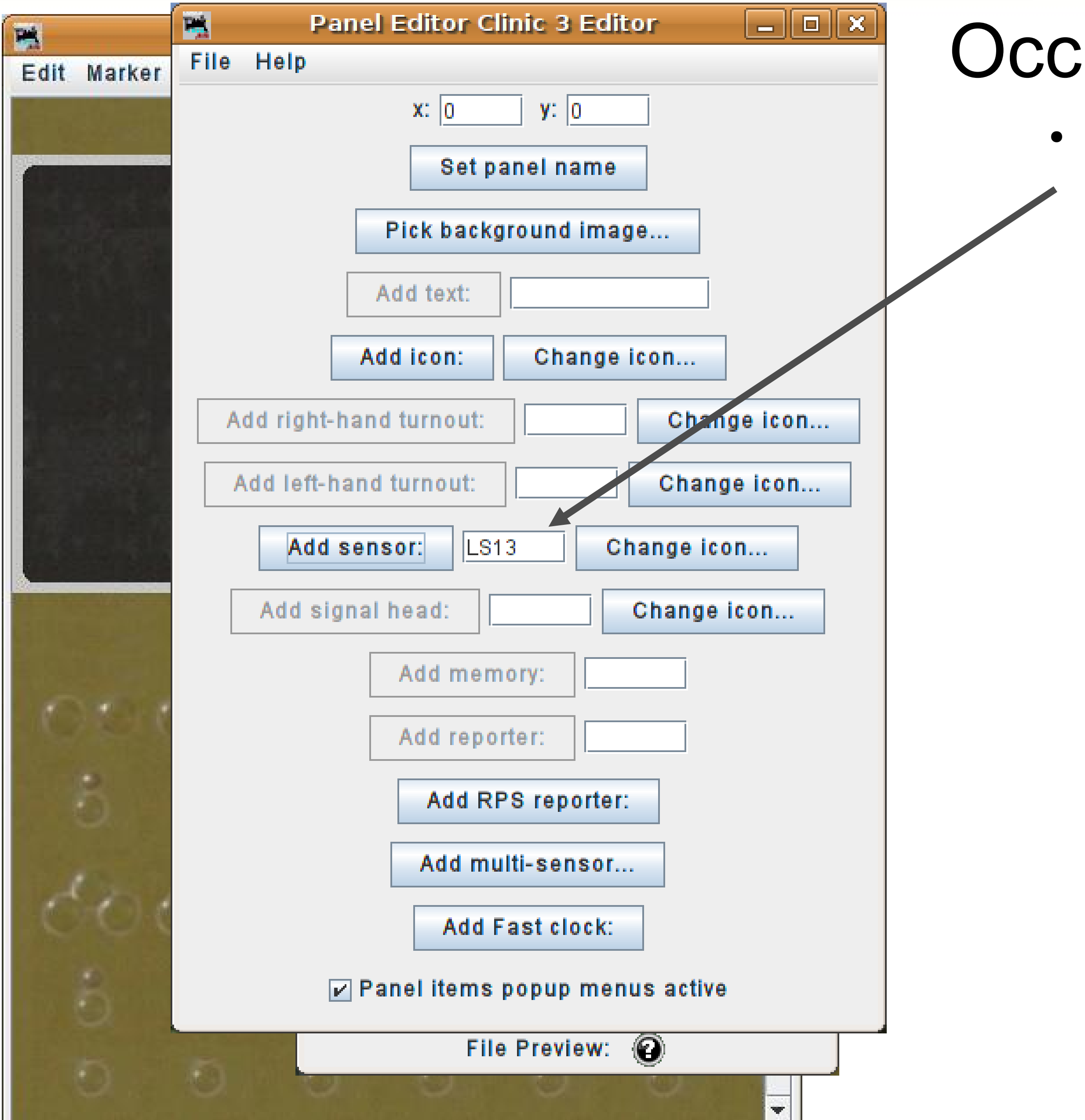
Indirect Layout Control

Sensor images



Occupancy sensors

- 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')
- In clinic 2 we added active icons for our turnouts. Now we will do the same for our occupancy sensors. Start by changing the icon images.
- Navigate to 'resources' – 'icons' – 'USS' – 'sensor'. Many railroads used red indicator lamp jewels for the OS sections. We will do the same.

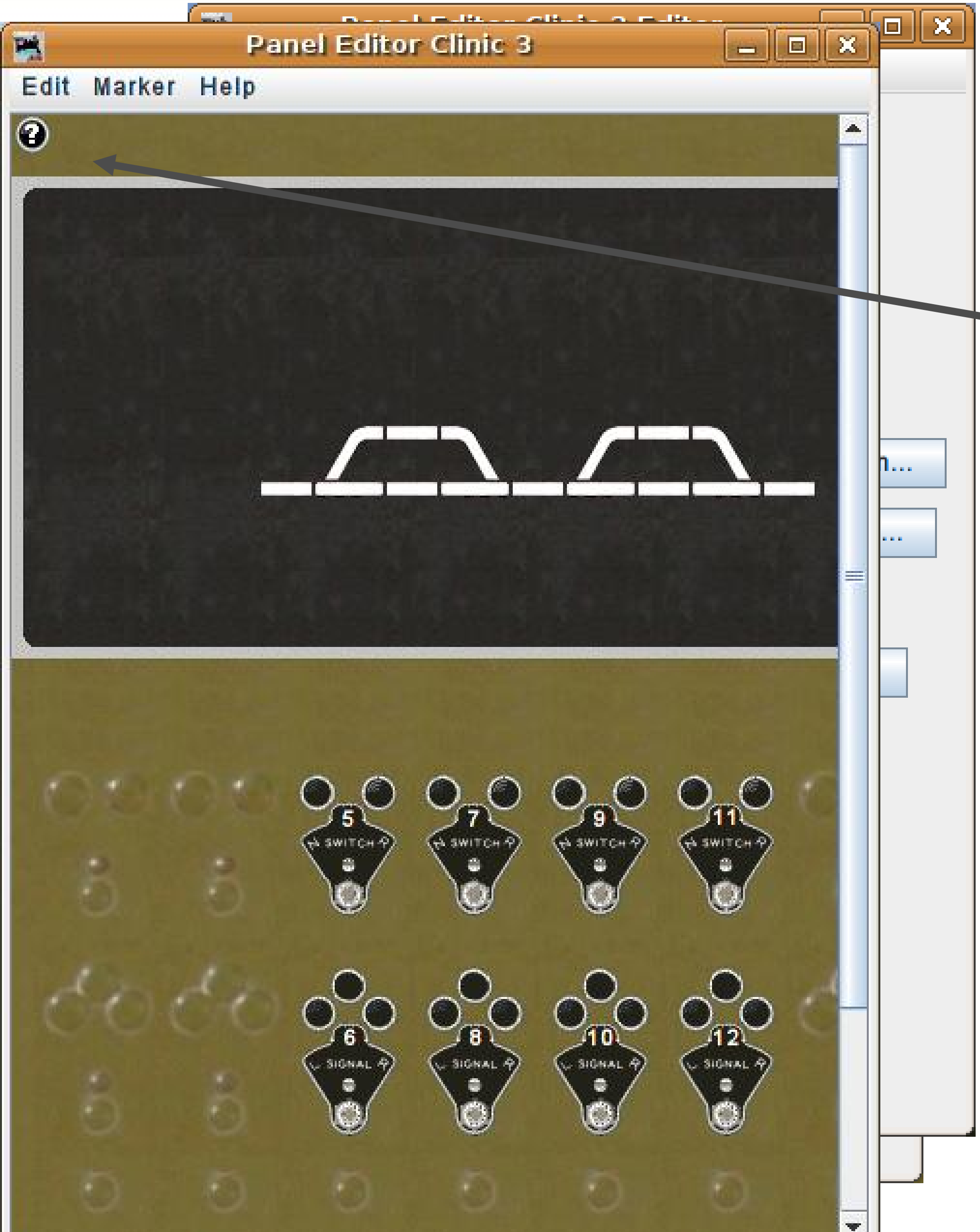


Add sensor: LS13

☑ Panel items popup menus active

Occupancy sensors

- Add sensors LS2, LS6, LS9, and LS13. (LS = LocoNet Sensor)

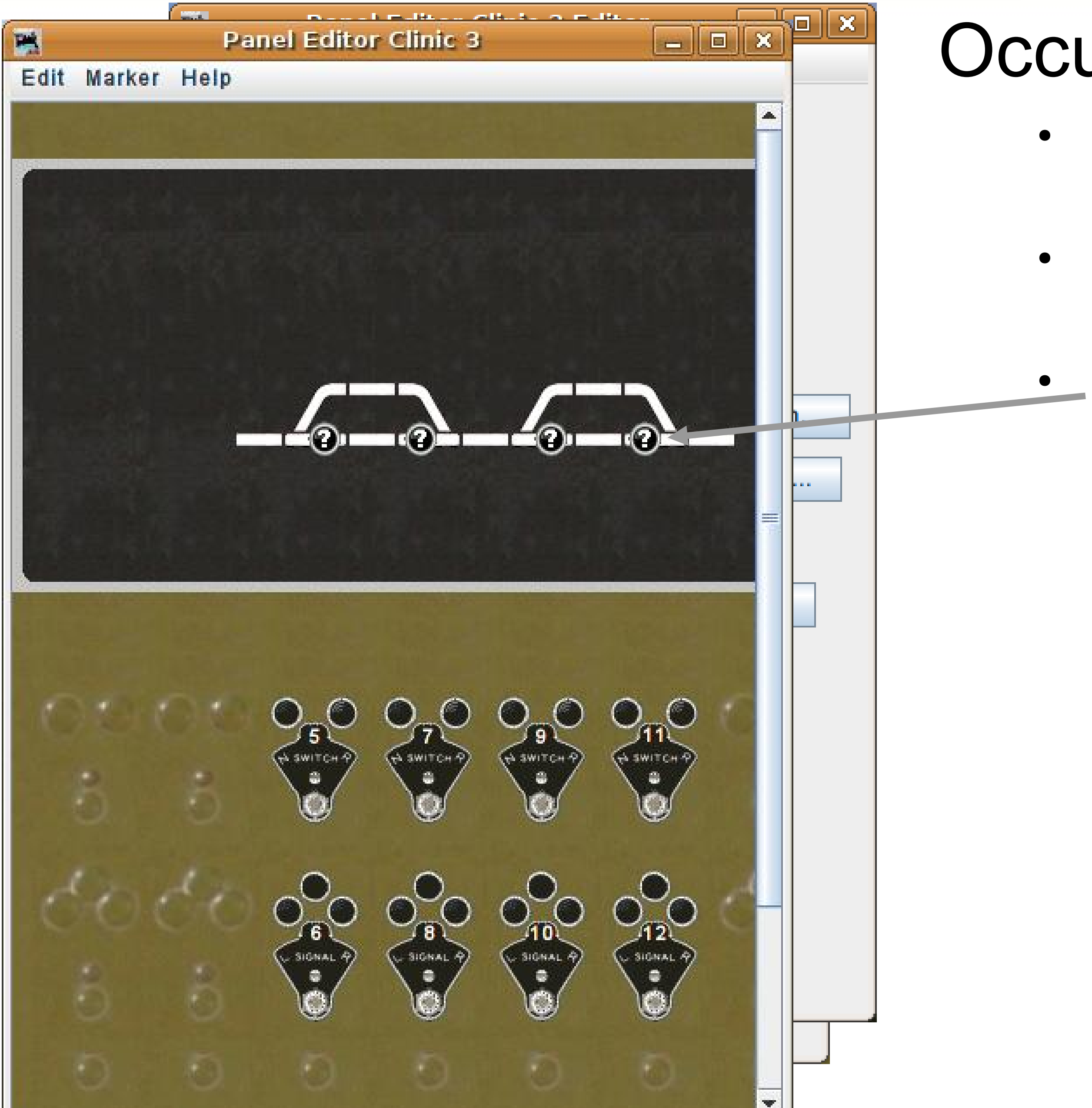


Occupancy sensors

- Add sensors LS2, LS6, LS9, and LS13. (LS = LocoNet Sensor)
- This piles our images in the usual place.

Indirect Layout Control

Sensor images

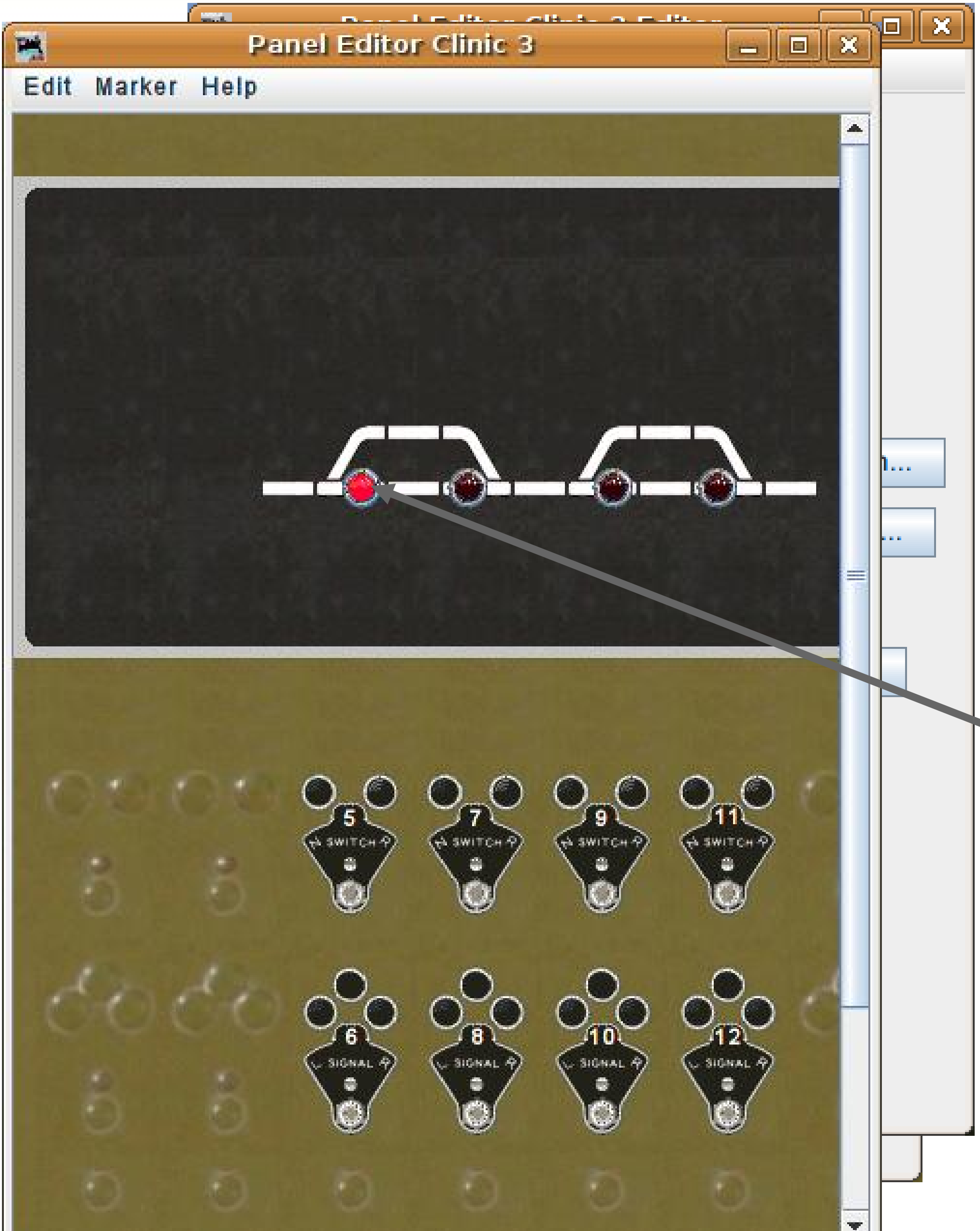


Occupancy sensors

- Add sensors LS2, LS6, LS9, and LS13. (LS = LocoNet Sensor)
- This piles our images in the usual place.
- Move them down into their proper track locations.

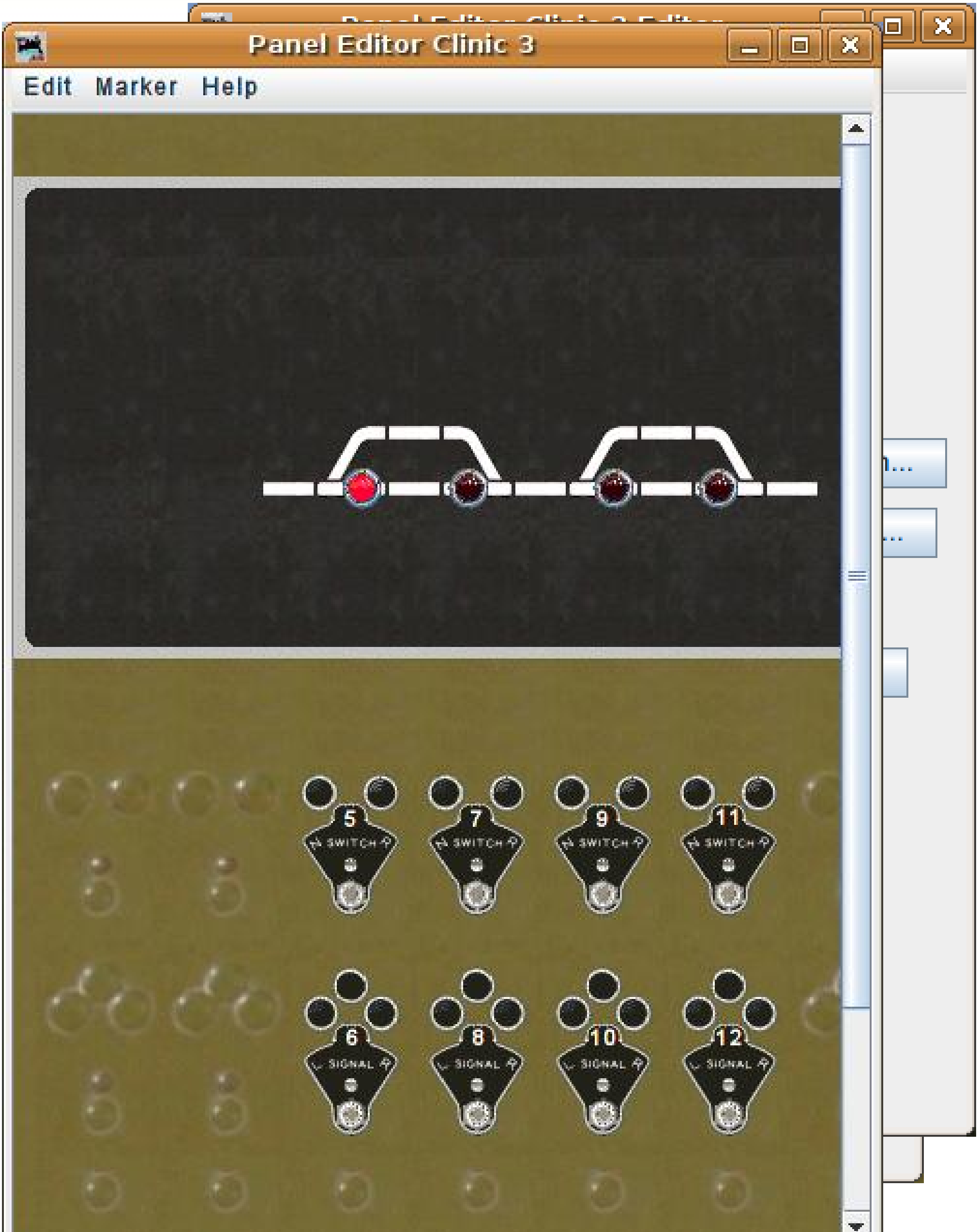
Indirect Layout Control

Sensor images



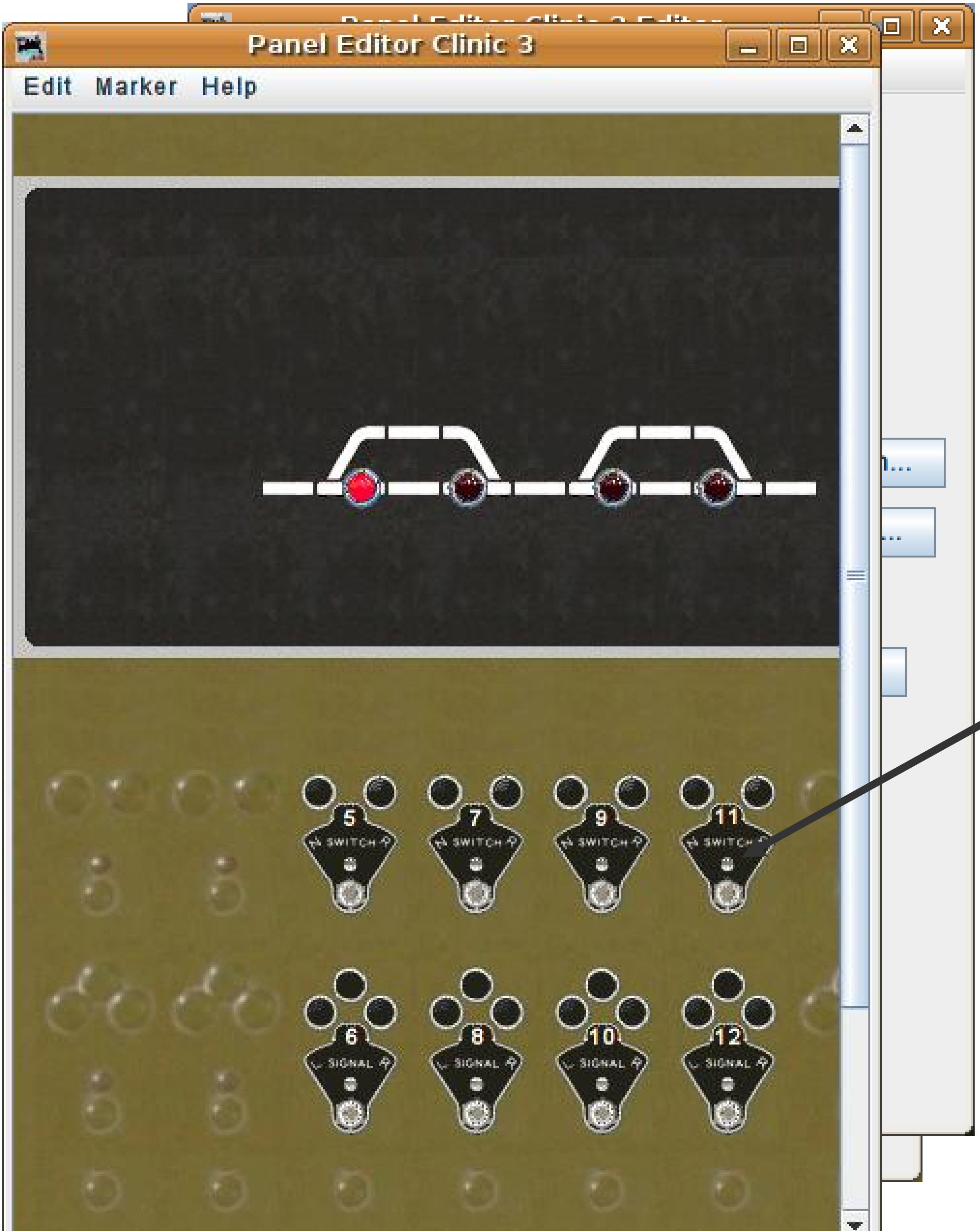
Occupancy sensors

- Add sensors LS2, LS6, LS9, and LS13. (LS = LocoNet Sensor)
- This piles our images in the usual place.
- Move them down into their proper track locations.
- 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.



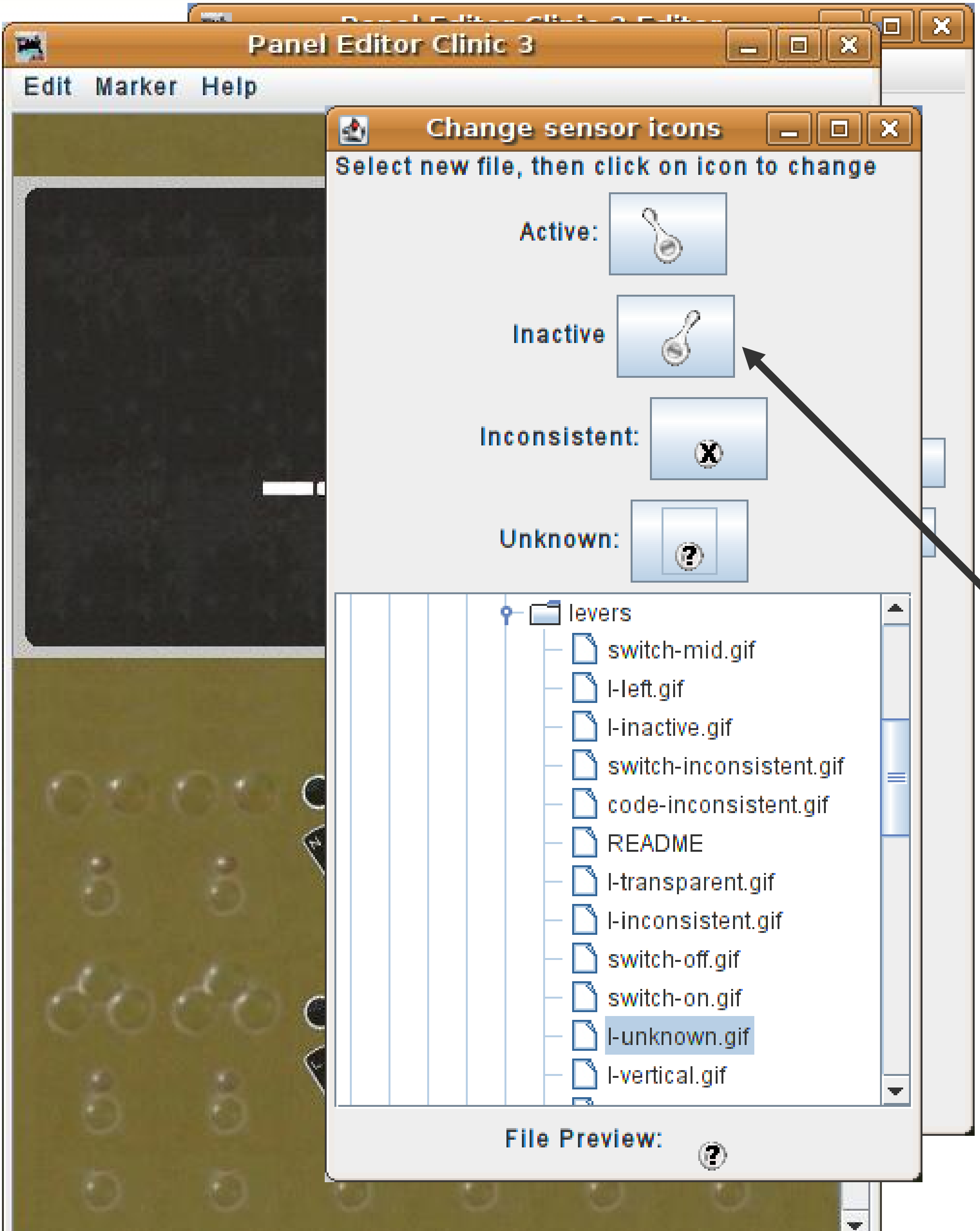
Internal sensors

- 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 levers that are not directly attached to the turnouts like we did originally.



Panel Editor Clinic 3

Change sensor icons

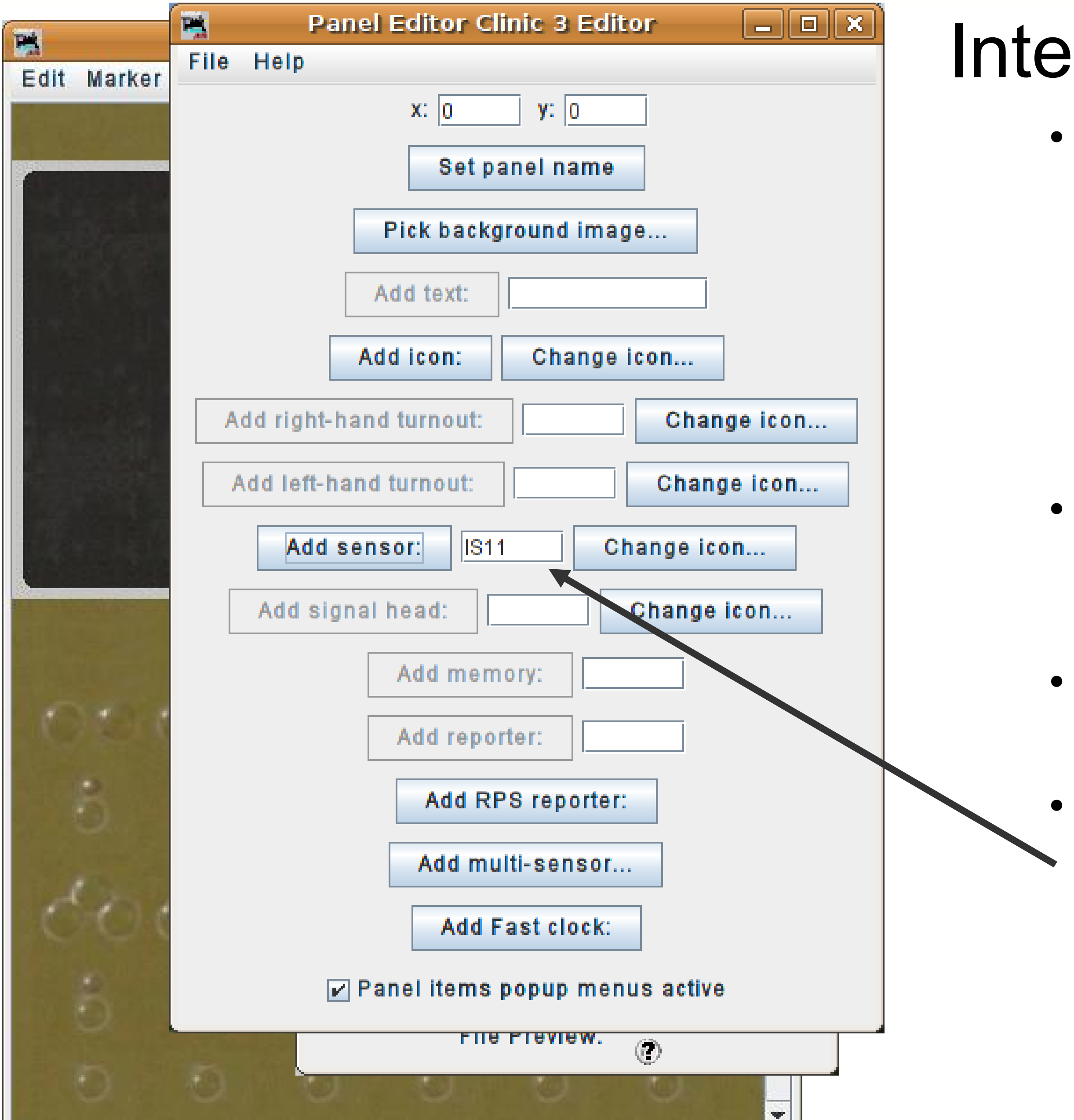
Select new file, then click on icon to change

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 levers that are not directly attached to the turnouts like we did originally.
- This time we will use the lever images for our sensors.

Indirect Layout Control

Sensor images



Panel Editor Clinic 3 Editor

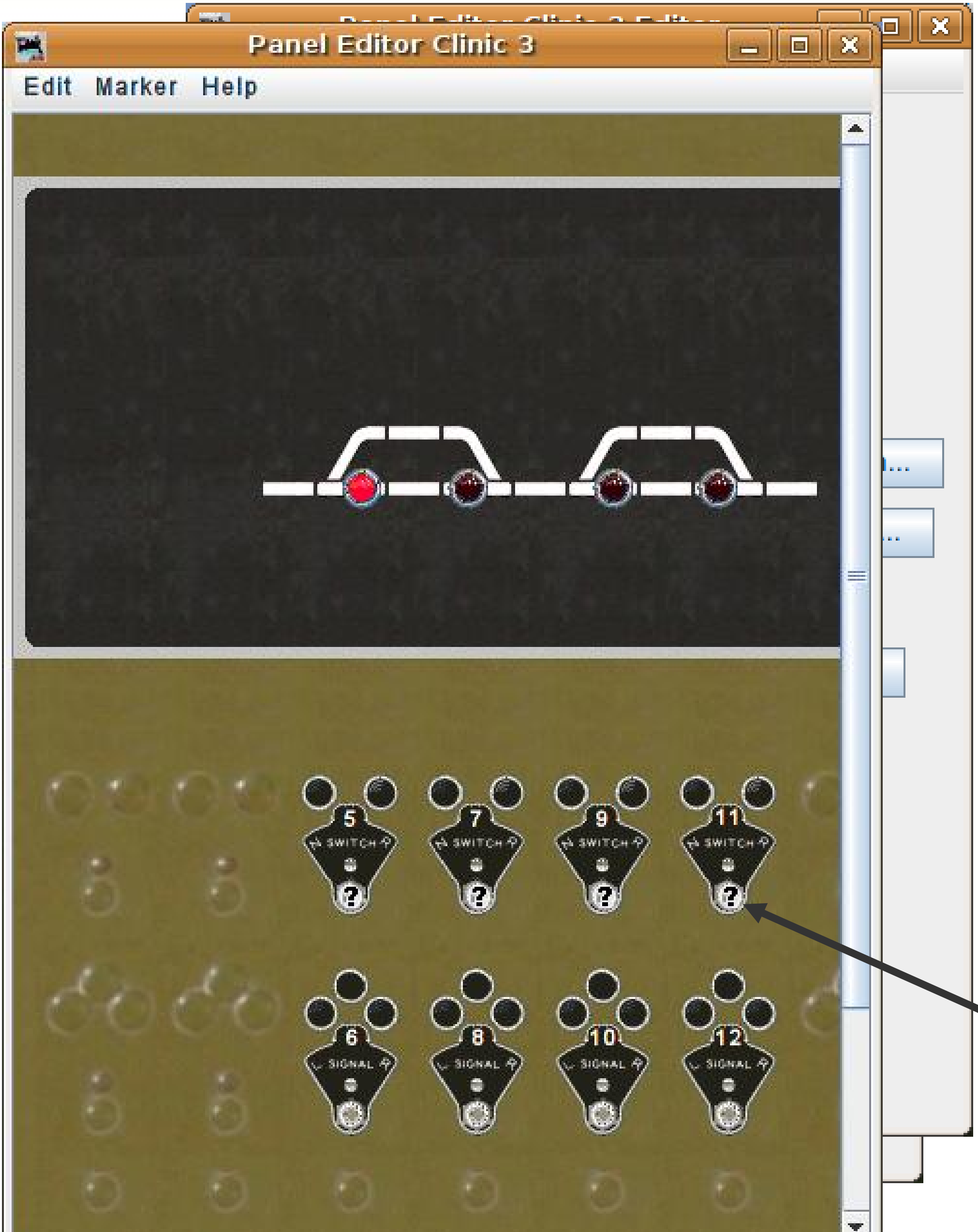
x: 0 y: 0

Add sensor: IS11

☑ Panel items popup menus active

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 levers that are not directly attached to the turnouts like we did originally.
- This time we will use the lever images for our sensors.
- Add sensors named IS5, IS7, IS9, and IS11 to match our plate numbers.

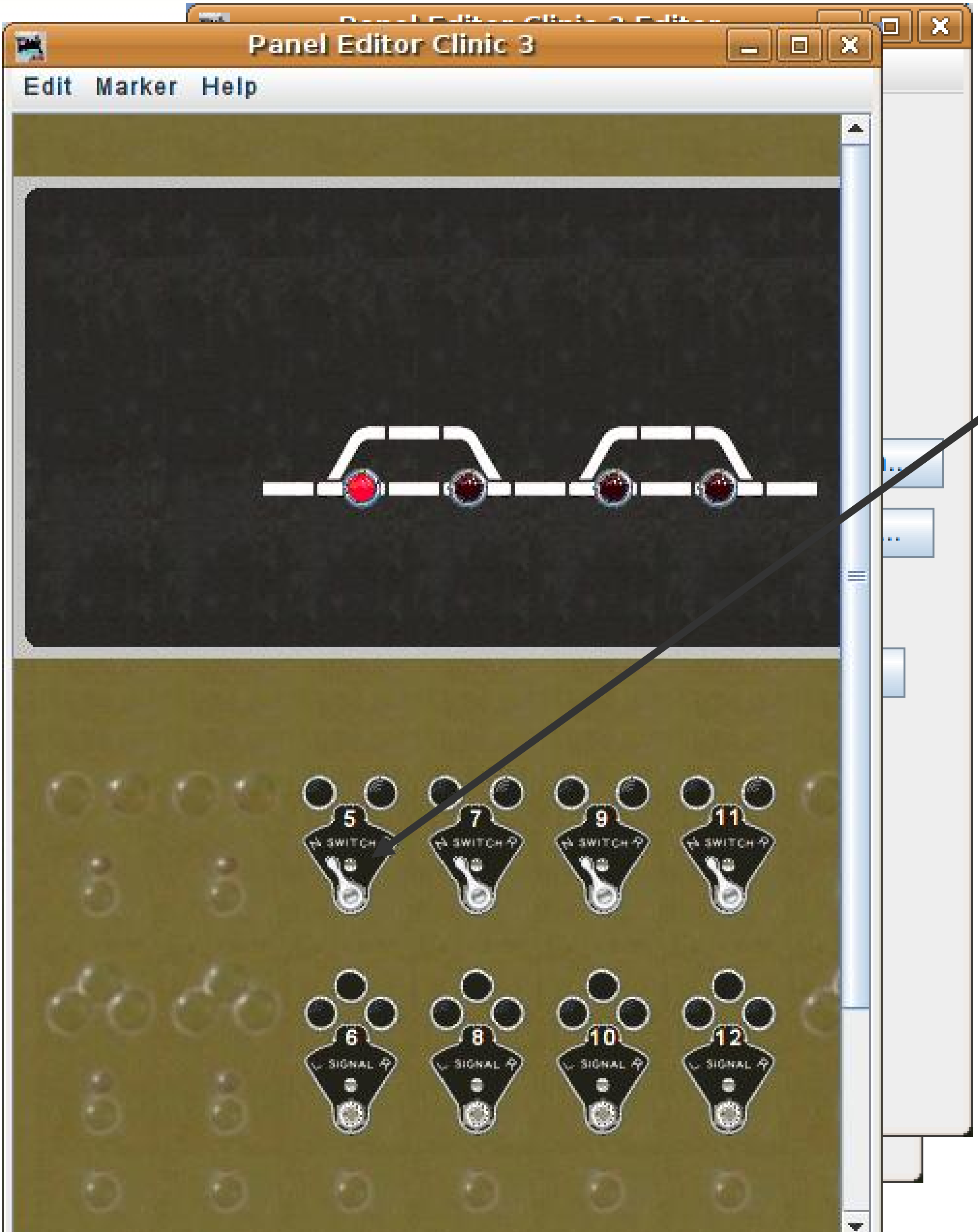


Internal sensors

- 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 levers that are not directly attached to the turnouts like we did originally.
- This time we will use the lever images for our sensors.
- Add sensors named IS5, IS7, IS9, and IS11 to match our plate numbers.
- As before, move the new icons into their proper locations.

Indirect Layout Control

Sensor images

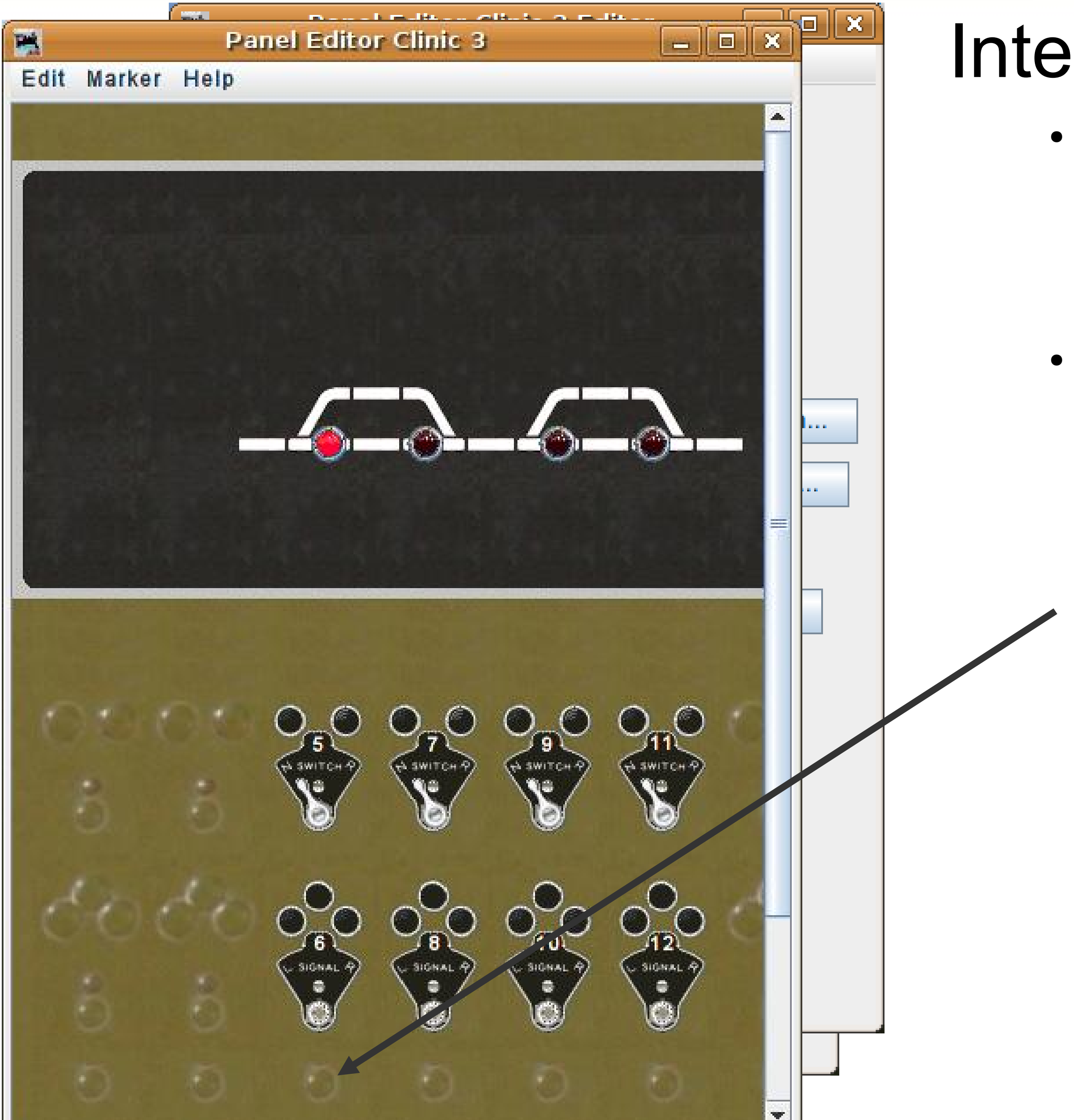


Internal sensors

- Now we have some levers that are not directly connected to the layout. We can flip them by simply by clicking on them.

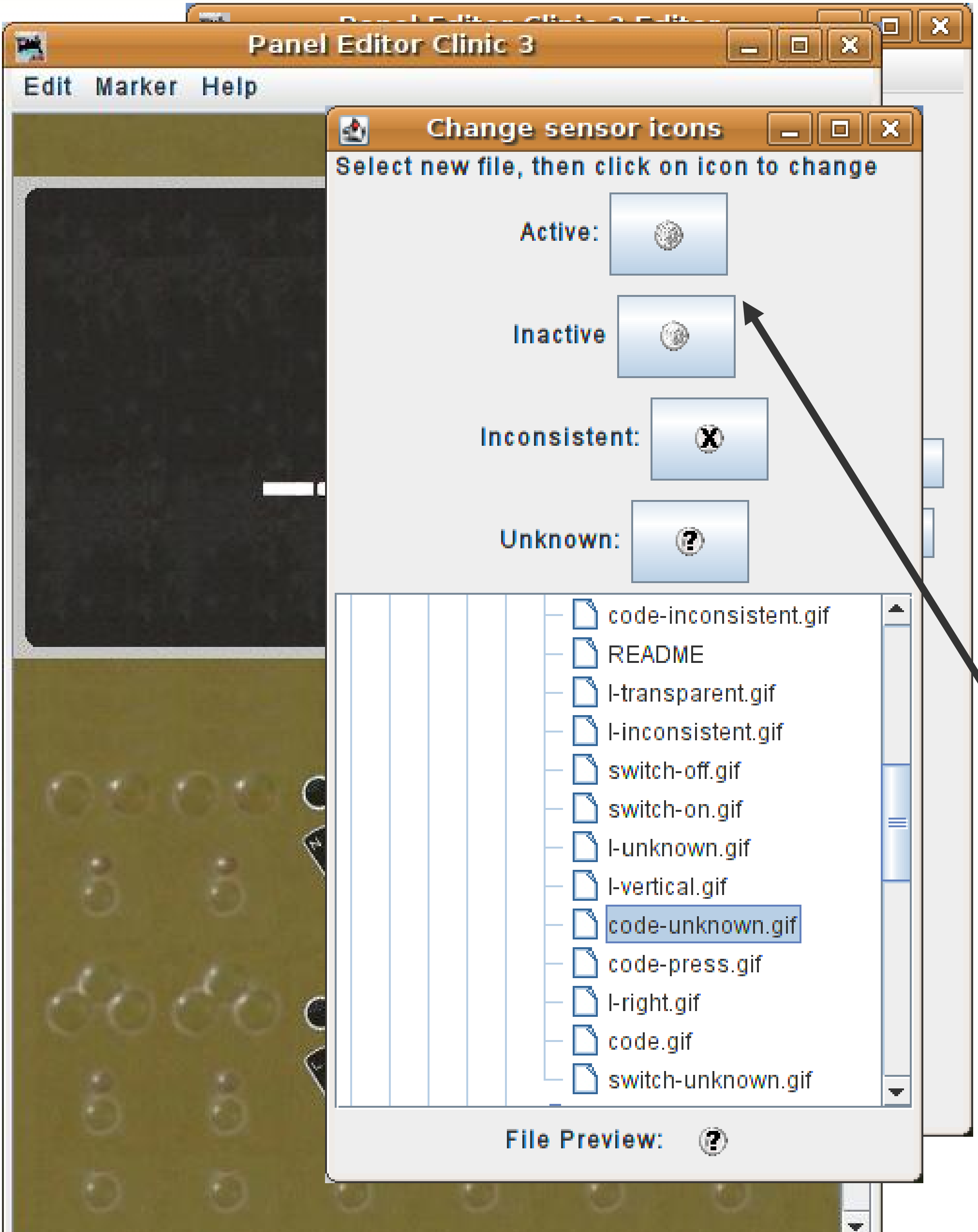
Indirect Layout Control

Sensor images



Internal sensors

- Now we have some levers that are not directly connected to the layout. We can flip them by simply by clicking on them.
- The prototype CTC panel did not directly connect the lever to the switch machine. 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 using short and long pulses. (I bet you thought DCC was a new concept)

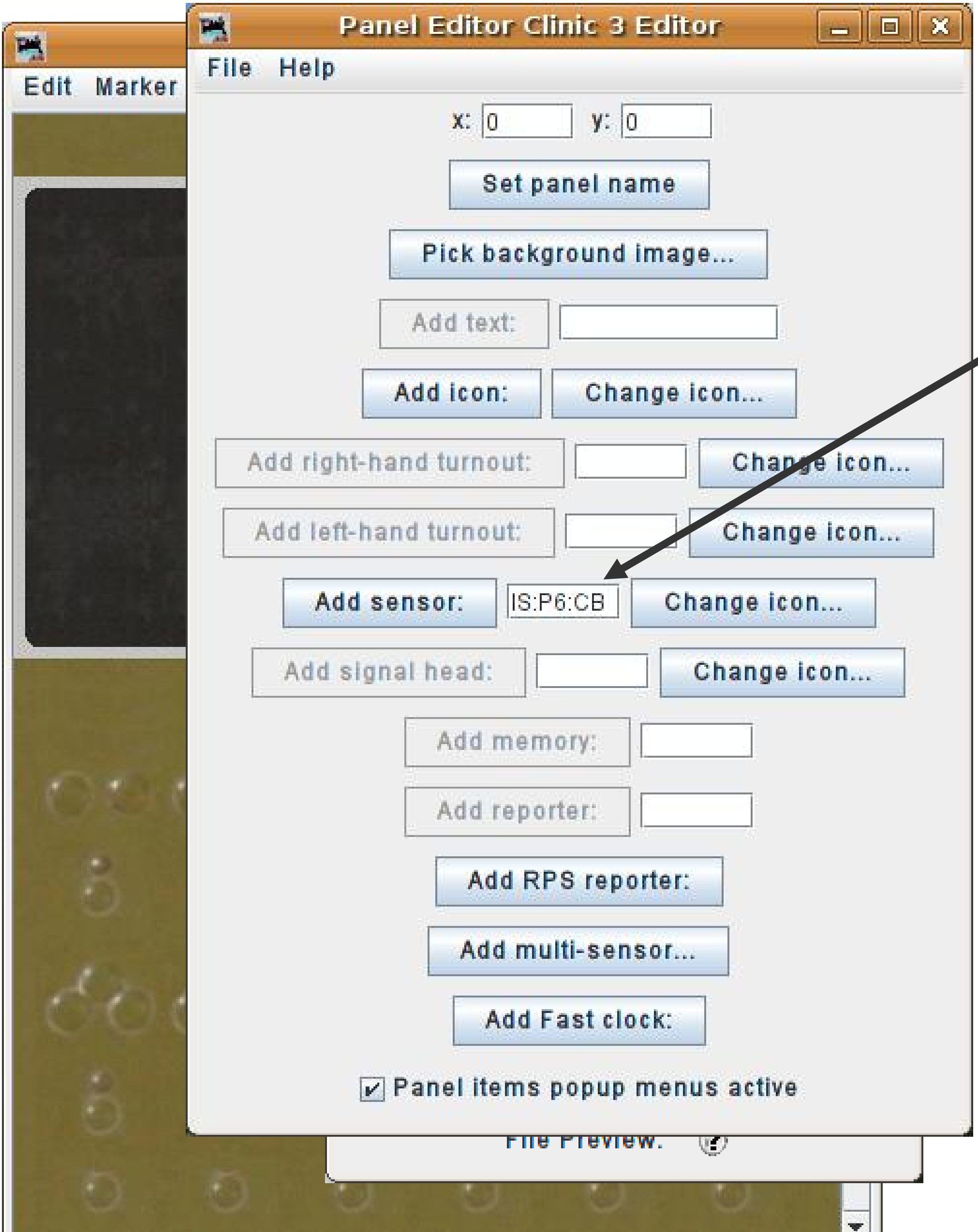


Select new file, then click on icon to change

code-inconsistent.gif
README
l-transparent.gif
l-inconsistent.gif
switch-off.gif
switch-on.gif
l-unknown.gif
l-vertical.gif
code-unknown.gif
code-press.gif
l-right.gif
code.gif
switch-unknown.gif

Internal sensors

- Now we have some levers that are not directly connected to the layout. We can flip them by simply by clicking on them.
- The prototype CTC panel did not directly connect the lever to the switch machine. 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 using short and long pulses. (I bet you thought DCC was a new concept)
- Change our icons to the 'code' button images.



x: 0 y: 0

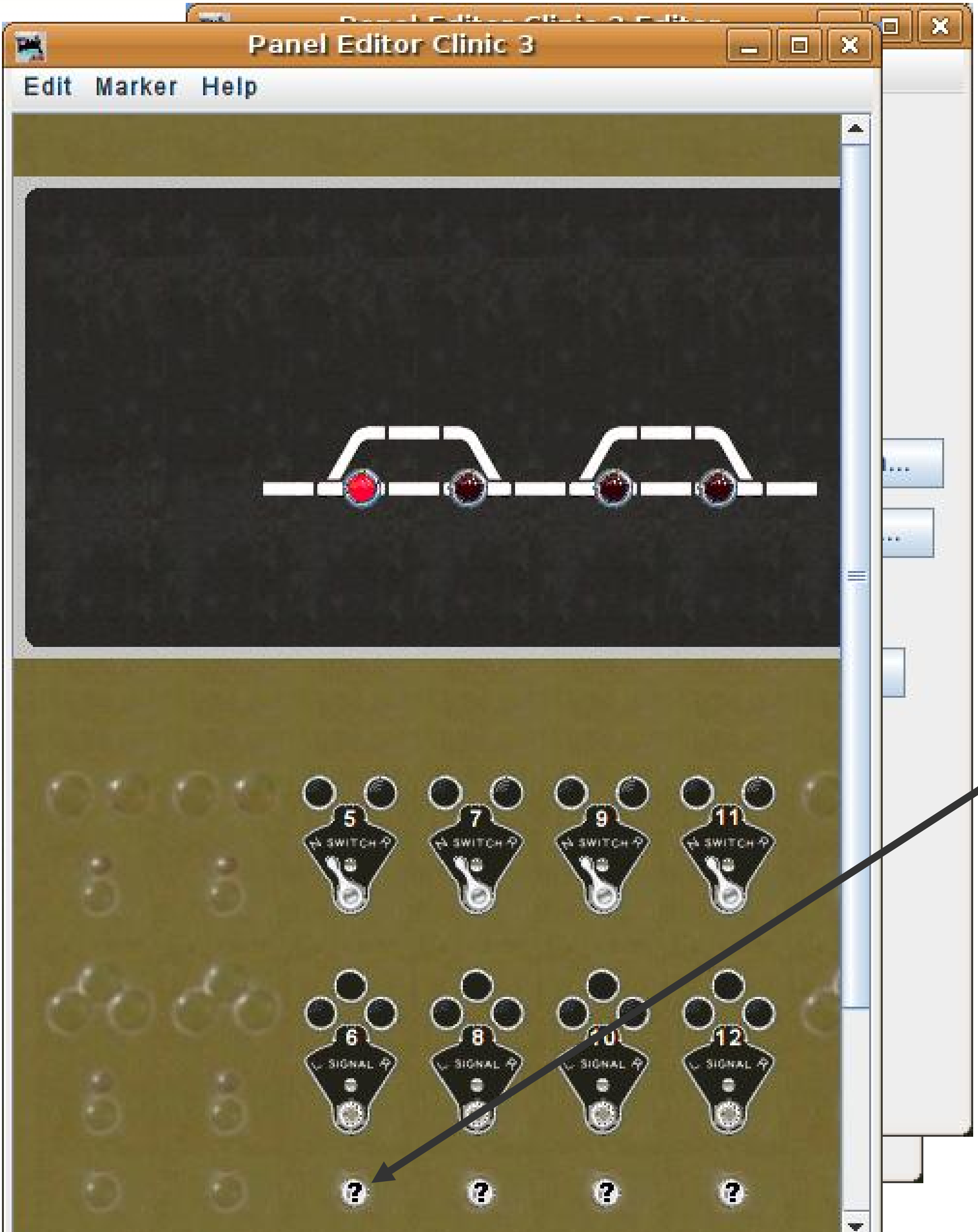
Add sensor: IS:P6:CB

☑ Panel items popup menus active

Internal sensor names

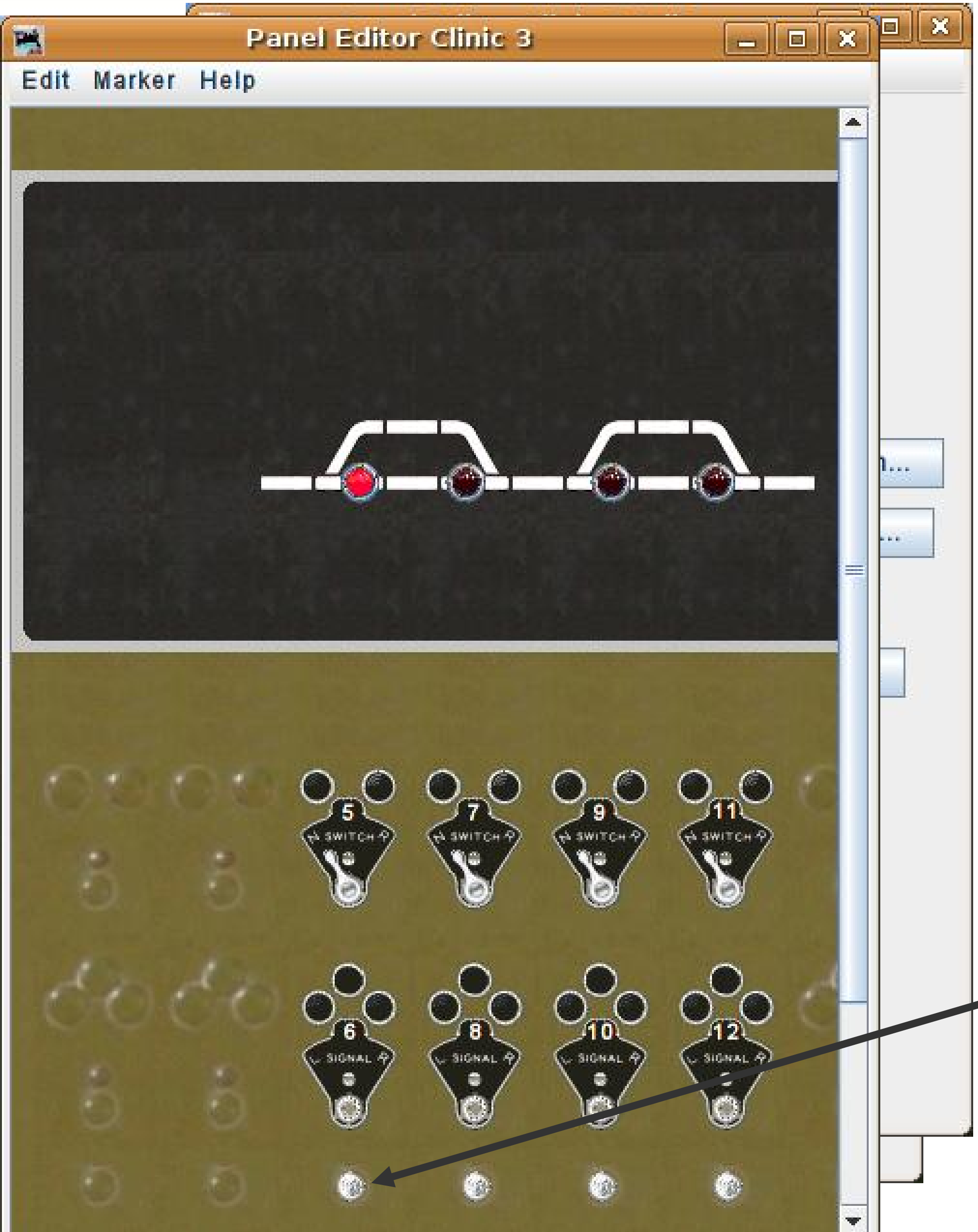
- When it comes to naming our new buttons we hit a snag. We already used IS5. We could use IS6, but what about the lever for the signals in the next clinic? Maybe we should use a more descriptive name. We are not attaching to real hardware, so any name is allowed. I chose 'IS:P6:CB'. Normally system generated names use the “:” and user names should not. The plan is that a tool will generate these names, so I use that as an excuse for including the “:”.

  IS = **I**nternal **S**ensor,
  :P6 = **P**lant **6**,
  :CB = **C**ode **B**utton.



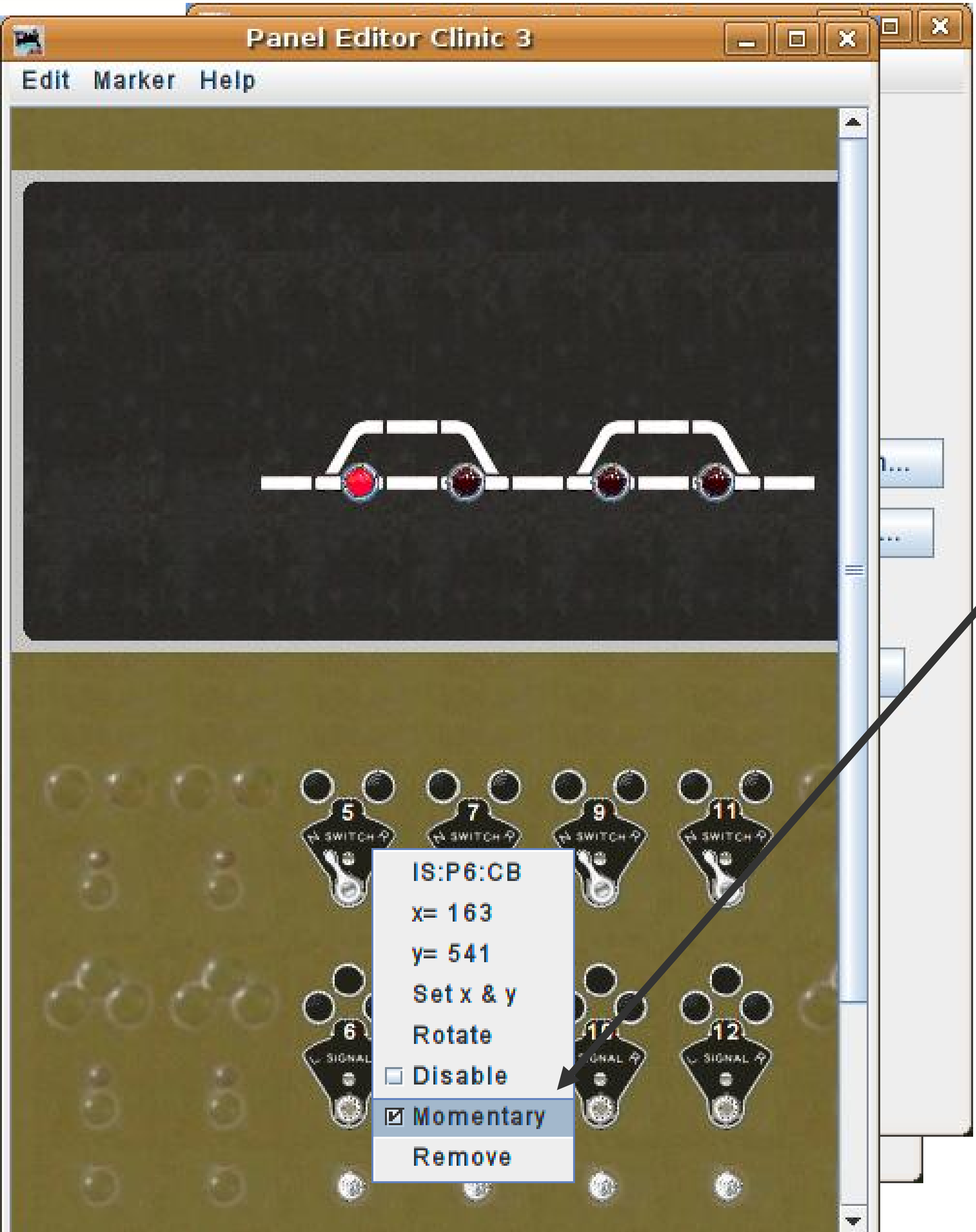
Internal sensor names

- Add 'IS:P6:CB', 'IS:P8:CB', 'IS:P10:CB', and 'IS:P12:CB' for our new buttons.
- Actually the code button was normally at the bottom of each column of switches and levers. However that is off the bottom of this portion of our image, so I cheated and placed the buttons here. You should do it right and reserve these for 'maintenance call' or 'call on' switches or lamps.



Internal sensor names

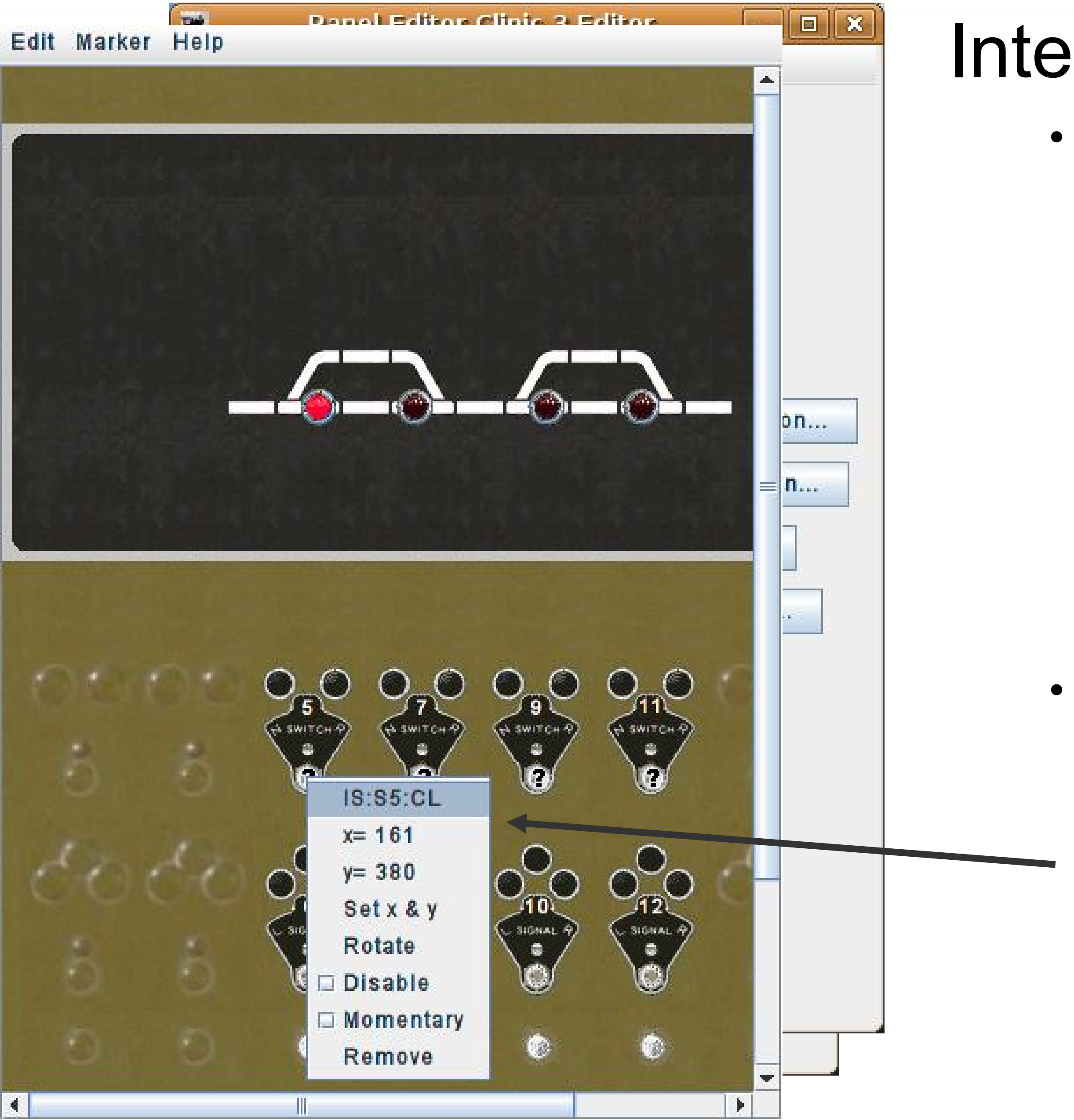
- Add 'IS:P6:CB', 'IS:P8:CB', 'IS:P10:CB', and 'IS:P12:CB' for our new buttons.
- Actually the code button was normally at the bottom of each column of switches and levers. However that is off the bottom of this portion of our image, so I cheated and placed the buttons here. You should do it right and reserve these for 'maintenance call' or 'call on' switches or lamps.
- Clicking on these images reveals that the first click pushes the button and the next click releases it. The actual code button is a spring return.



Internal sensor names

- To solve the 'momentary' problem by simply right clicking on each icon to bring up its tools, then check the box called 'Momentary'. Further testing will show that the buttons will now directly follow your mouse clicks. Note, if the button pops up when you click on it, you have simply reversed the images. To correct them change the image icons and re-enter the buttons.

Internal sensor names

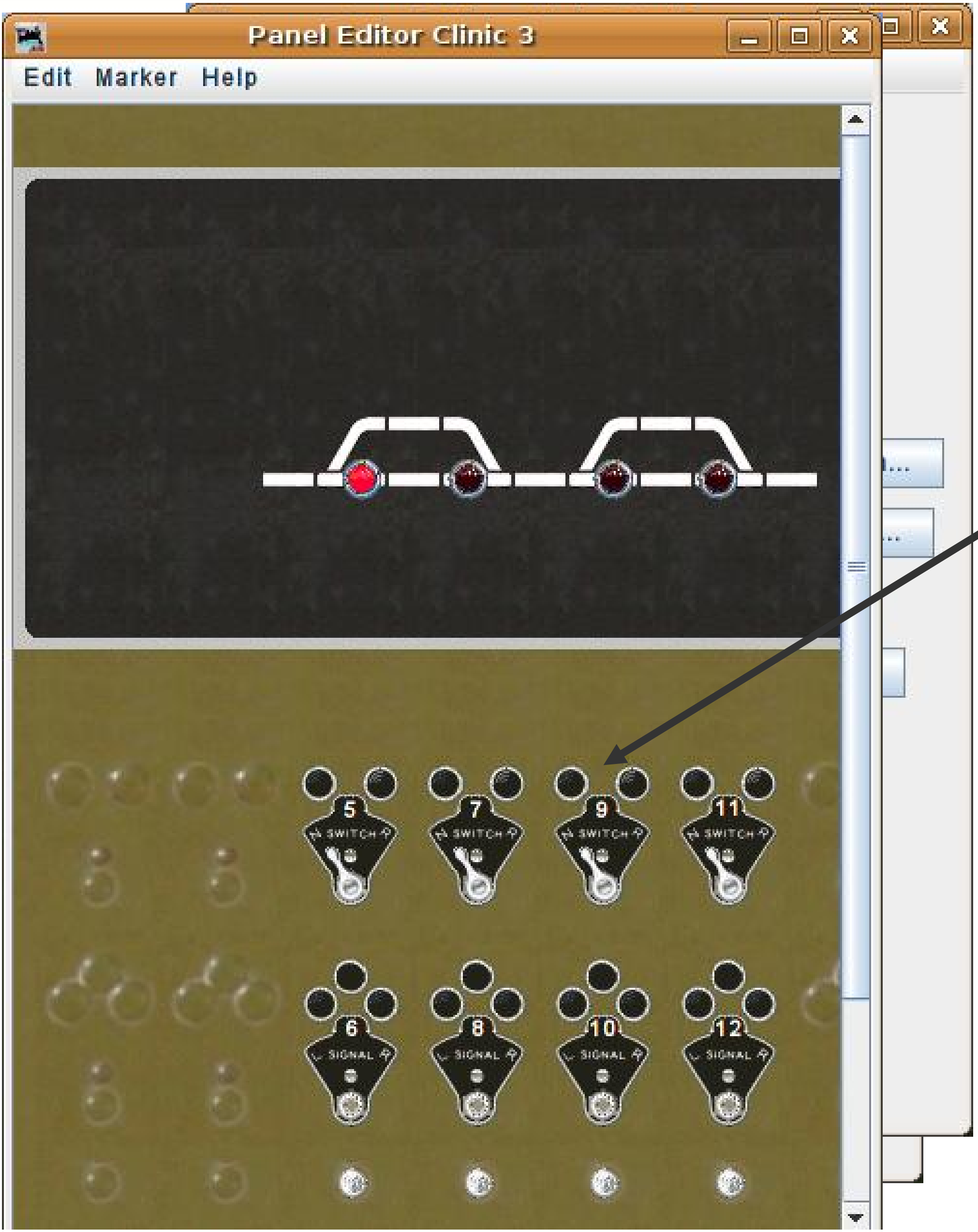


- To solve the 'momentary' problem by simply right clicking on each icon to bring up its tools, then check the box called 'Momentary'. Further testing will show that the buttons will now directly follow your mouse clicks. Note, if the button pops up when you click on it, you have simply reversed the images.

- While we are correcting things lets also use some better names for the levers. I used **IS:S5:CL**.

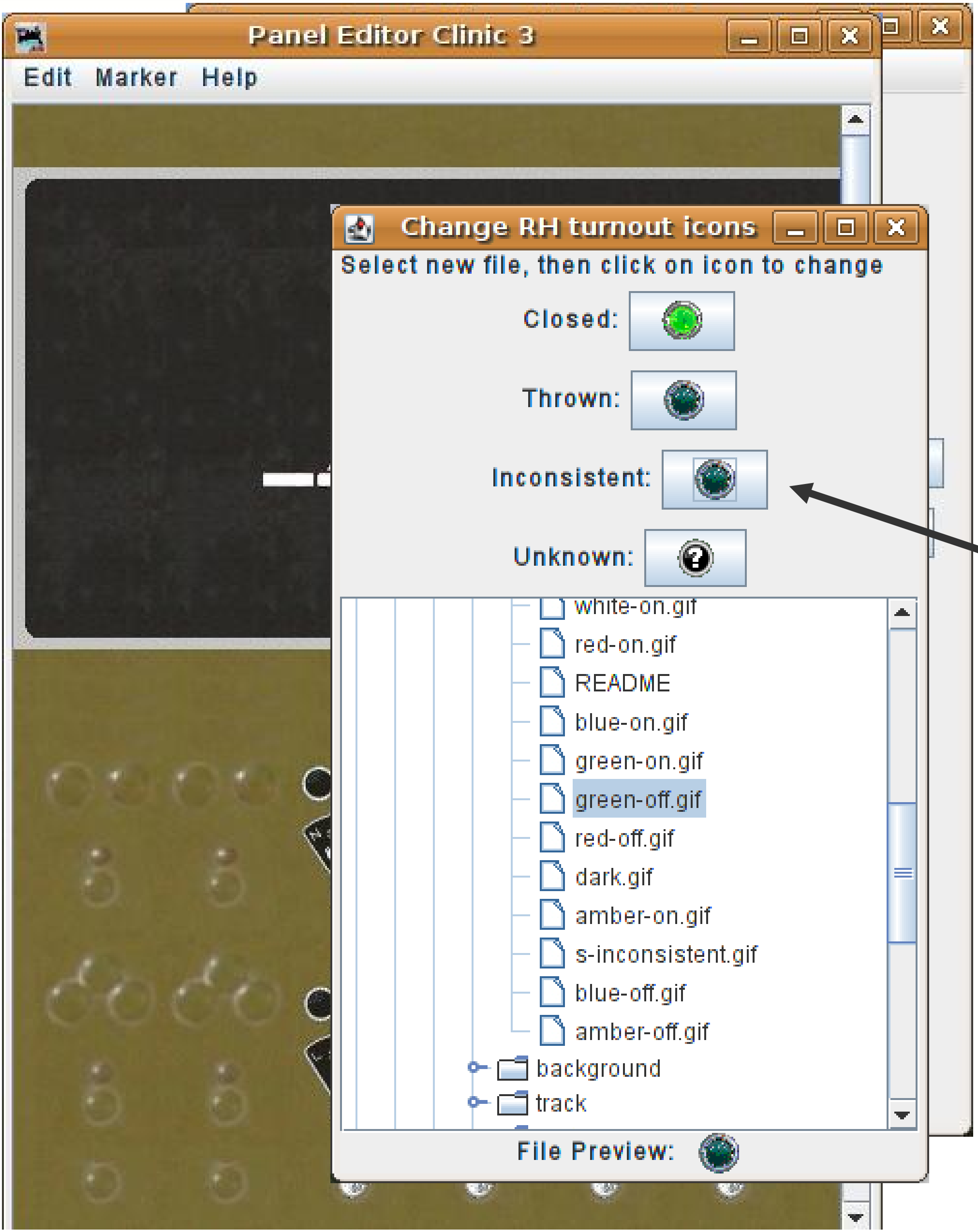
  IS = **I**nternal **S**ensor,
  :S5 = **S**witch **5**,
  :CL = **C**ontrol **L**ever.

  'Remove' the originals and add in the new replacements.



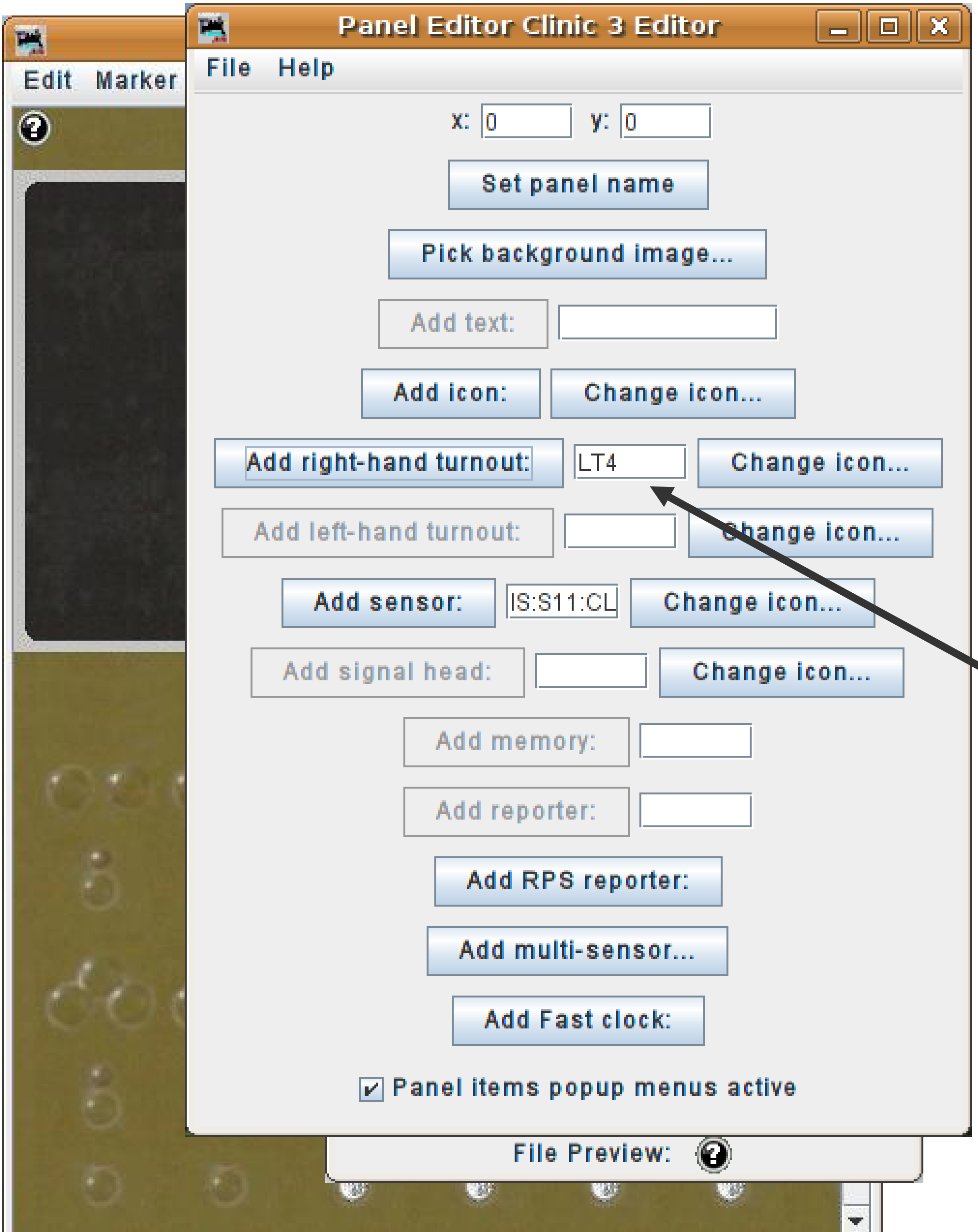
Turnout Feedback

- 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 as they were intended)



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 as they were intended)
- Change our turnout icons to be green jewels. Note: Use the 'green-off.gif' for the 'Inconsistent:' position.



x: 0 y: 0

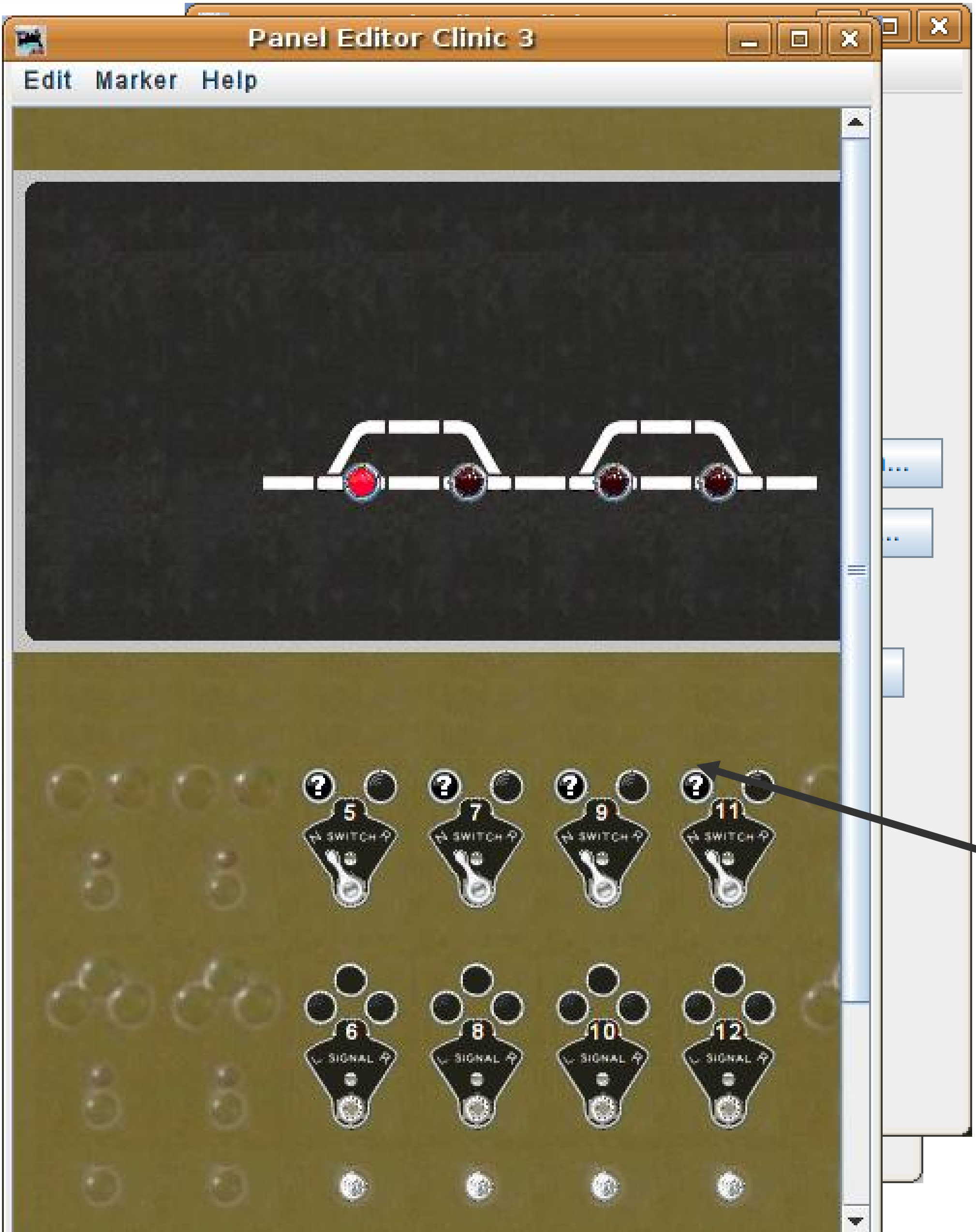
Add right-hand turnout: LT4

Add sensor: IS:S11:CL

☑ Panel items popup menus active

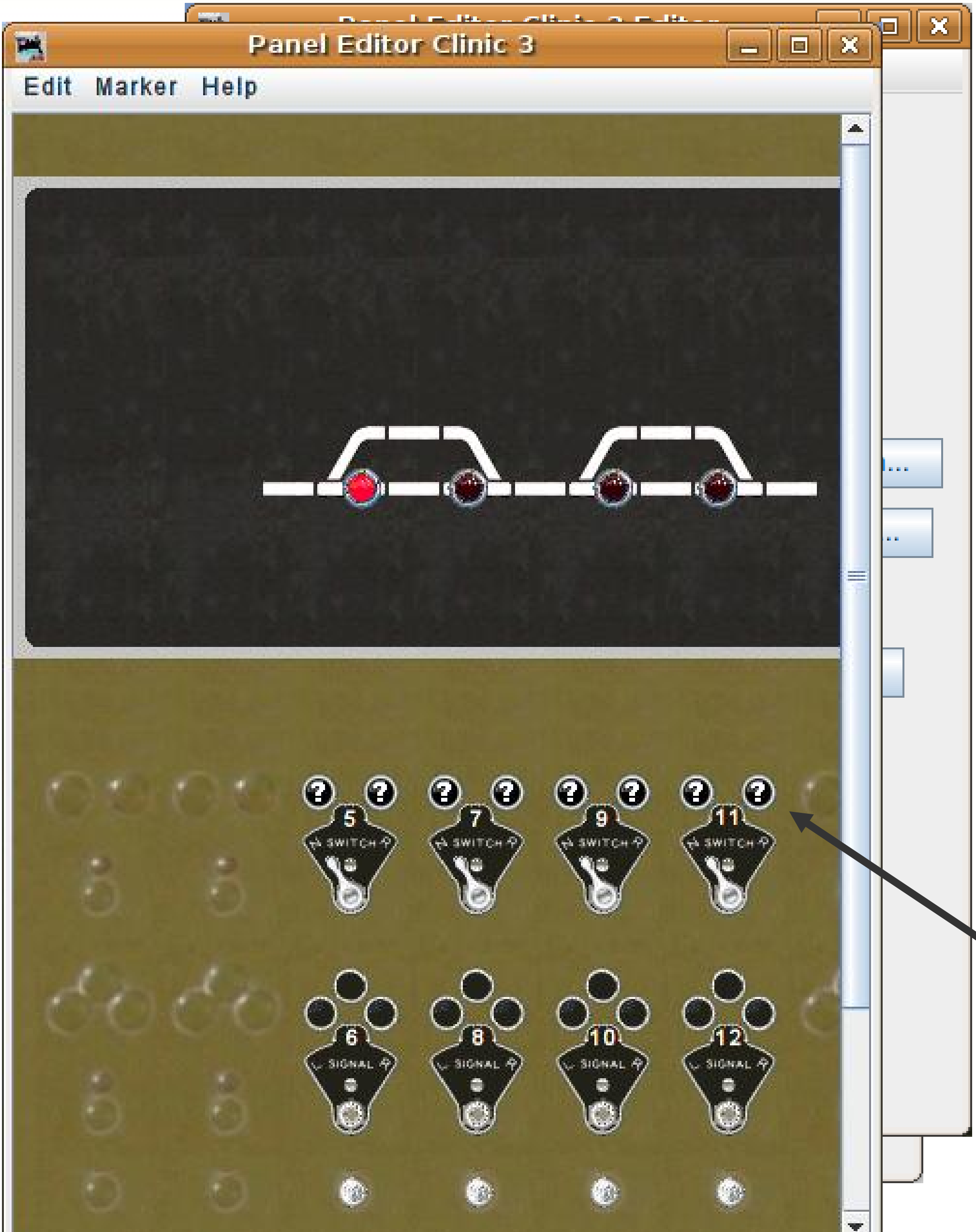
Turnout Feedback

- 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 as they were intended)
- Change our turnout icons to be green jewels. Note: Use the 'green-off.gif' for the 'Inconsistent:' position.
- Add LT1, LT2, LT3, and LT4.



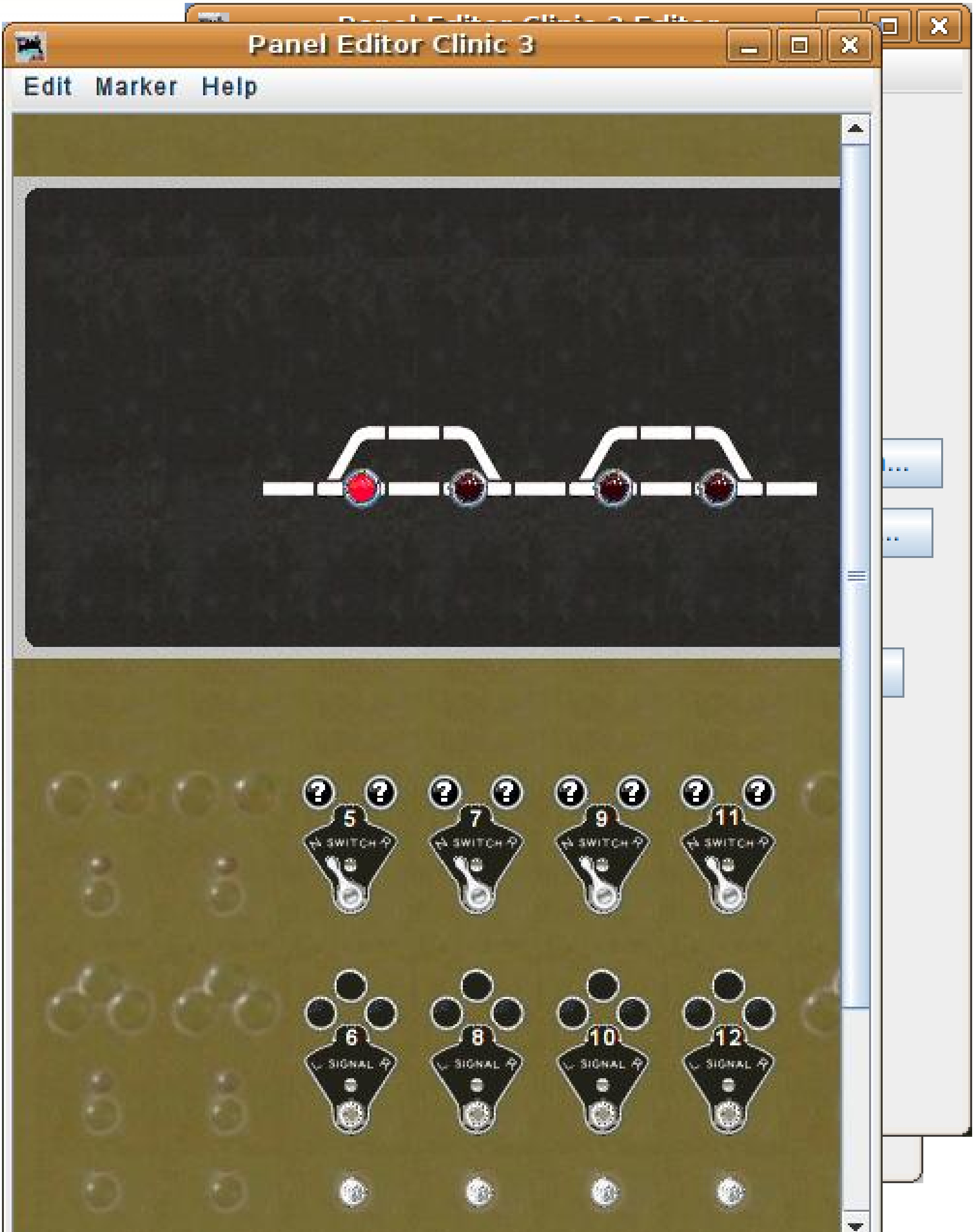
Turnout Feedback

- 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 as they were intended)
- Change our turnout icons to be green jewels. Note: Use the 'green-off.gif' for the 'Inconsistent:' position.
- Add LT1, LT2, LT3, and LT4.
- Move them into place on the 'N' (Normal) side of the plates.



Turnout Feedback

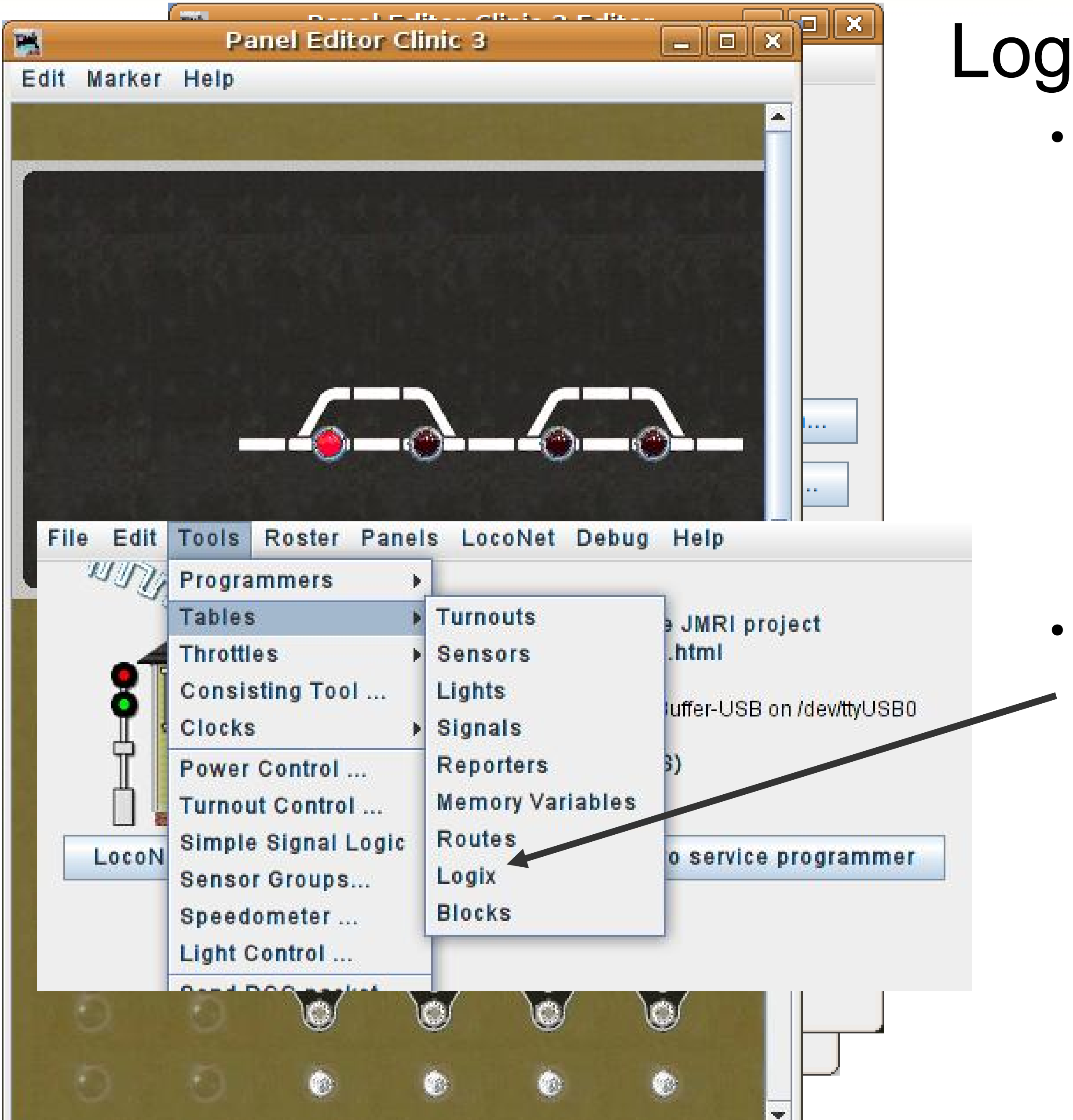
- We still need some way to tell which position the layout track switches are aligned. The levers and track image are not available, so we will use the indicator lamps. (just as they were intended)
- Change our turnout icons to be green jewels. Note: Use the 'green-off.gif' for the 'Inconsistent:' position.
- Add LT1, LT2, LT3, and LT4.
- Move them into place on the 'N' (Normal) side of the plates.
- Change the jewel colors to amber and do the same for the 'Reverse' lamps. (remember 'Thrown' is lit for them)



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.

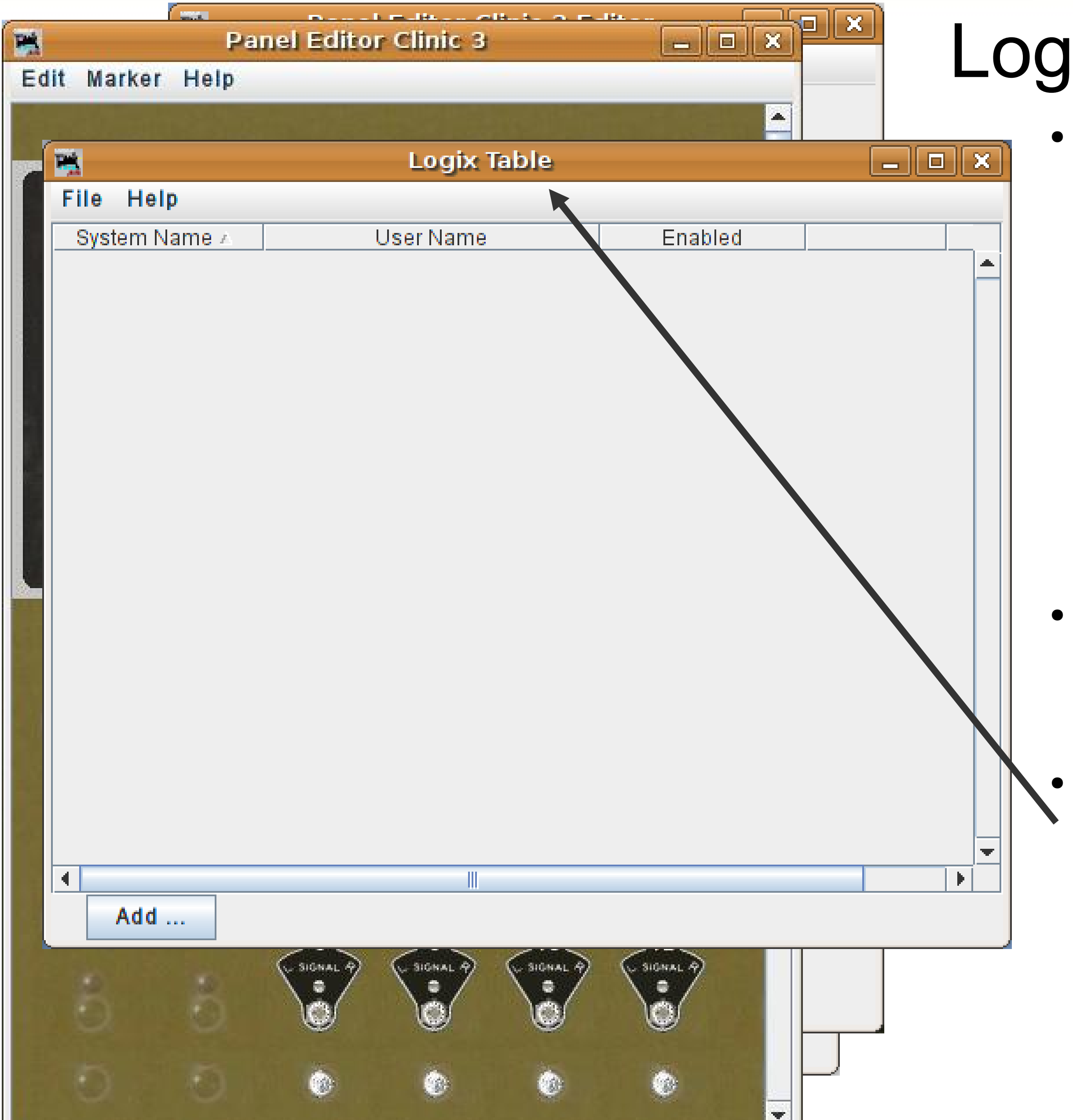


Logix

- 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.

- To open the Logix tools navigate from the main window via 'Tools' – 'Tables' – 'Logix'.



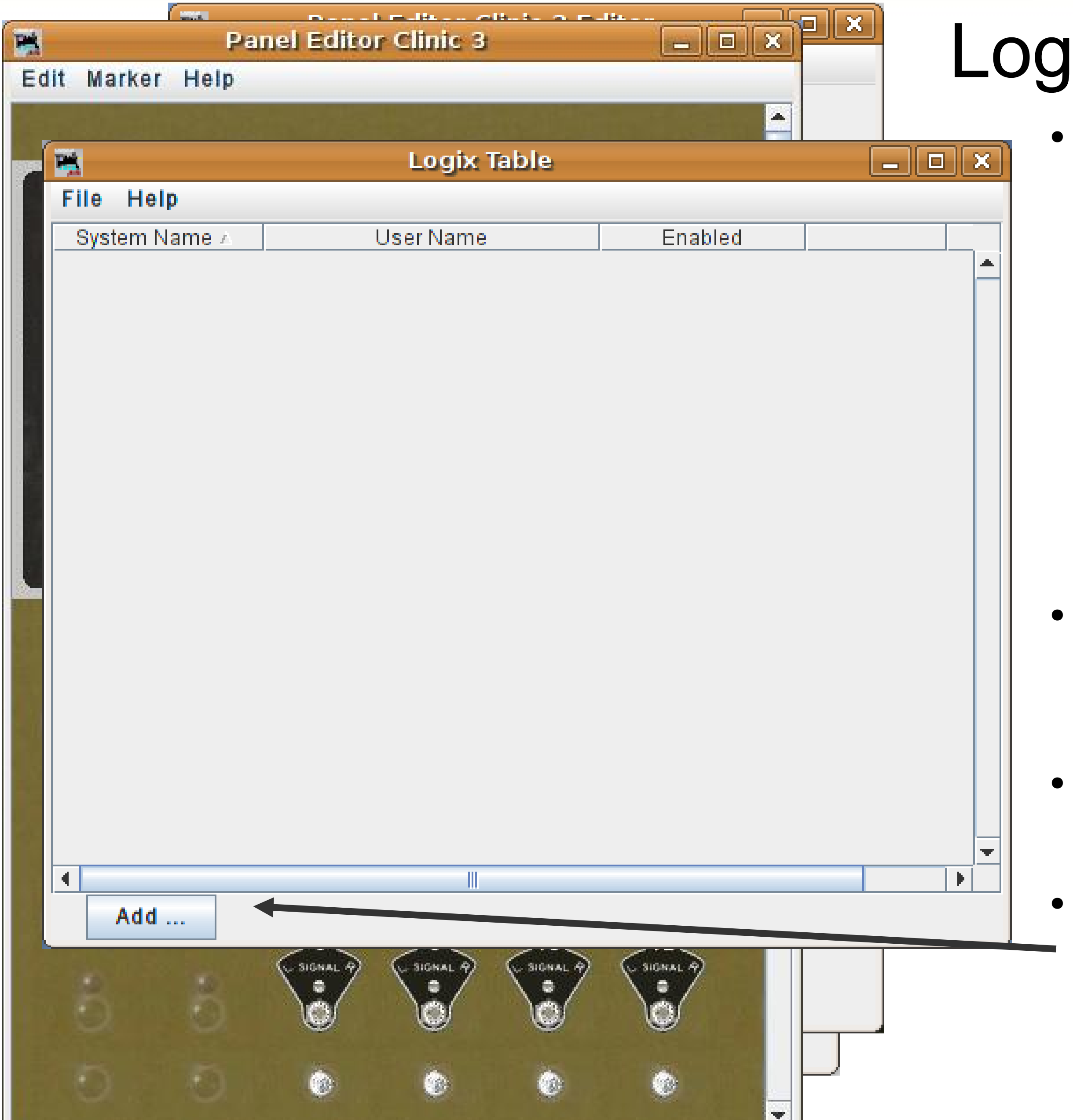
Logix

- 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.

- To open the Logix tools navigate from the main window via 'Tools' – 'Tables' – 'Logix'.
- This brings up an empty Logix table.

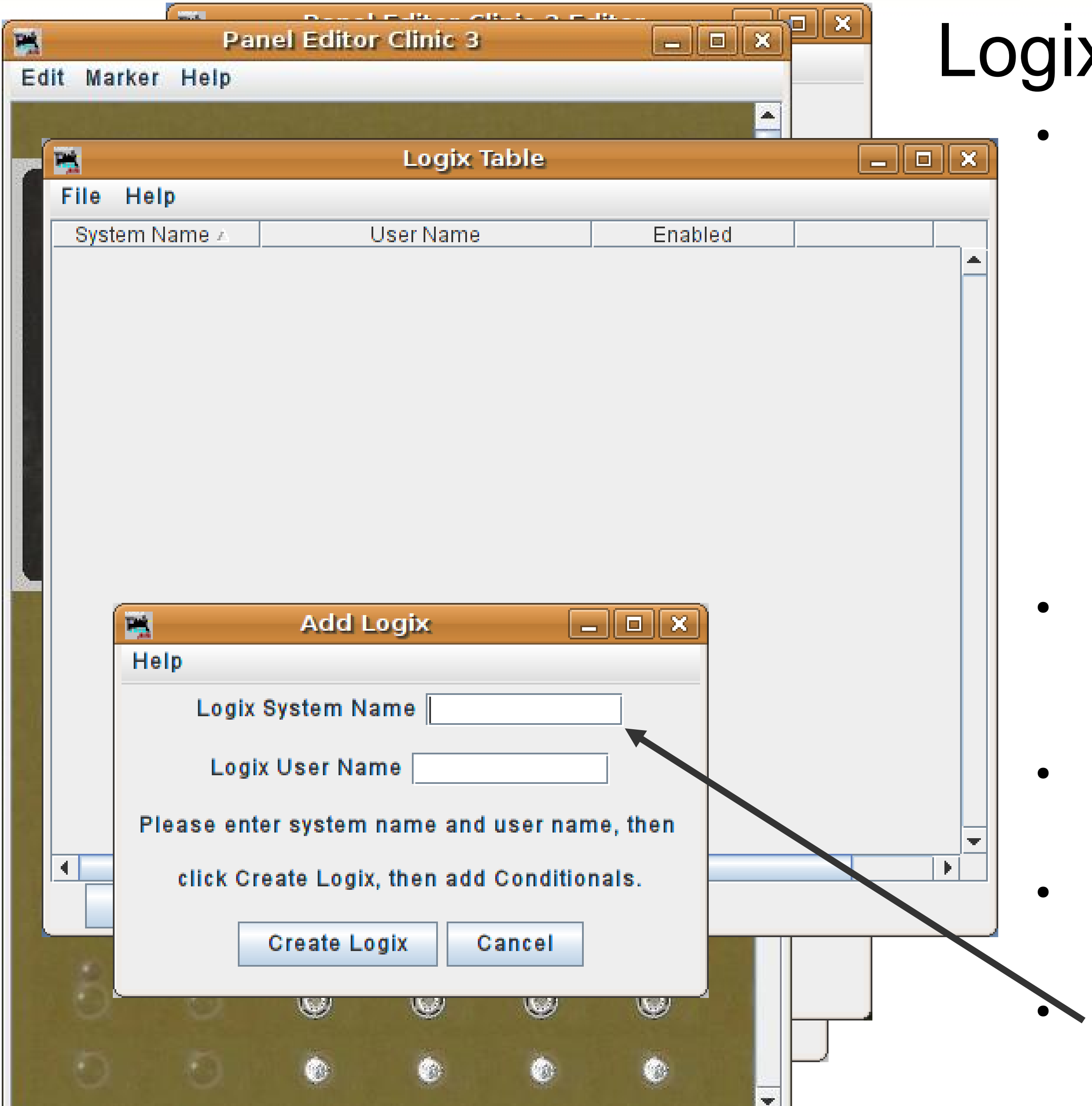
Logix



- 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.

- To open the Logix tools navigate from the main window via 'Tools' – 'Tables' – 'Logix'.
- This brings up an empty Logix table.
- Click on 'Add ...' to create a new Logix entry.



Logix System Name

Logix User Name

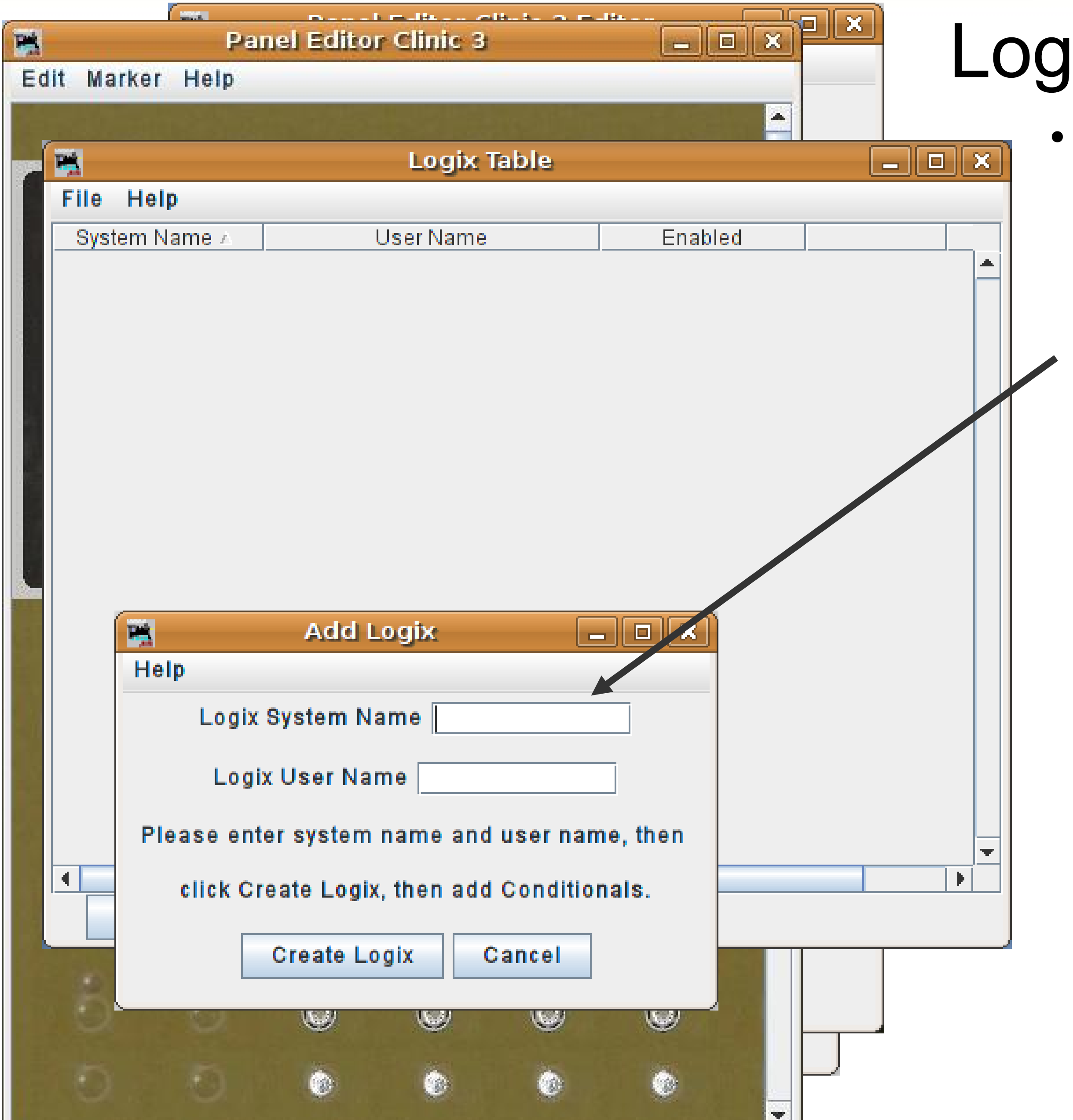
Please enter system name and user name, then click Create Logix, then add Conditionals.

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.

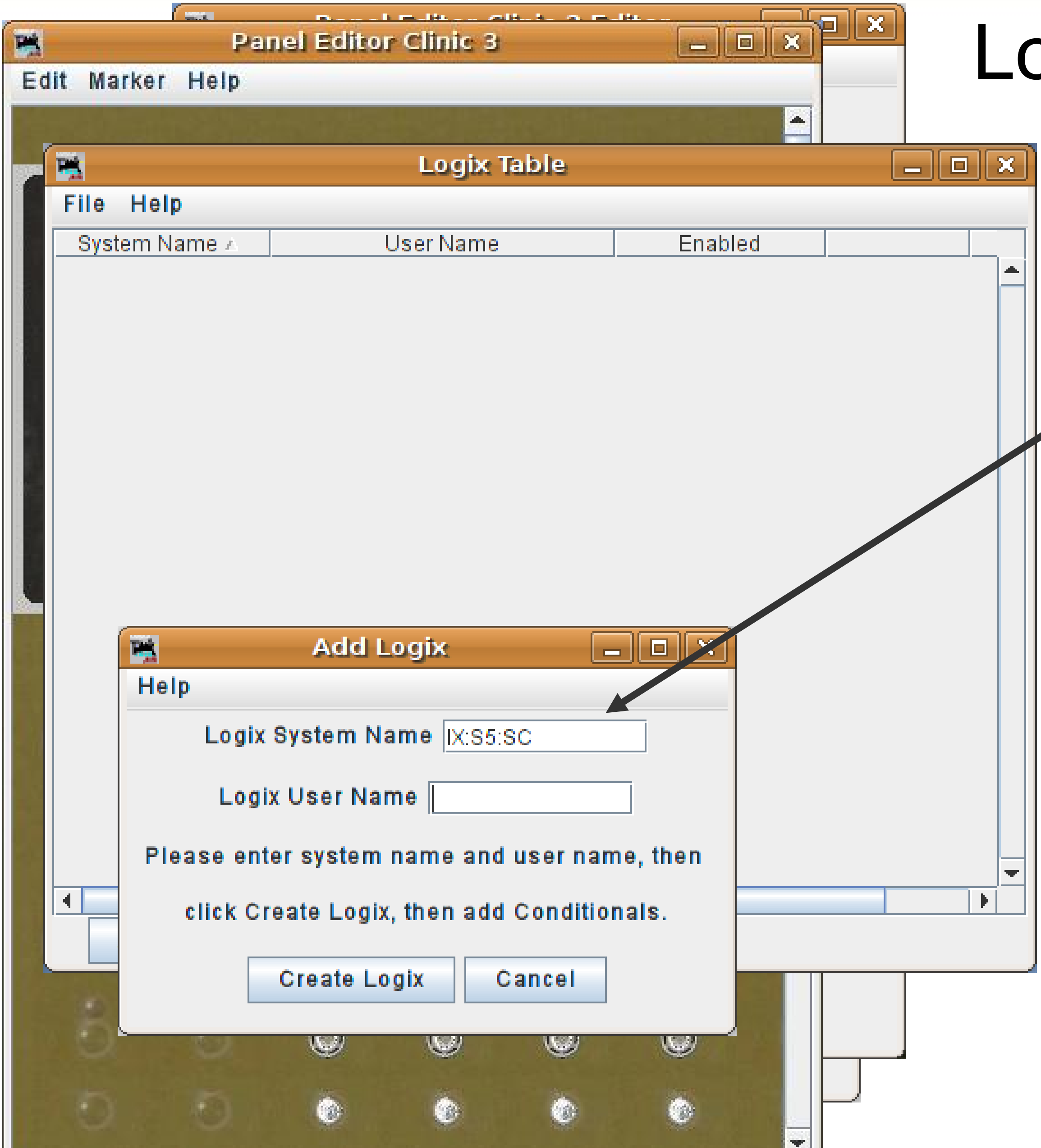
- To open the Logix tools navigate from the main window via 'Tools' – 'Tables' – 'Logix'.
- This brings up an empty Logix table.
- Click on 'Add ...' to create a new Logix entry.
- Fill in the required information.



Please enter system name and user name, then click Create Logix, then add Conditionals.

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 System Name IX:S5:SC

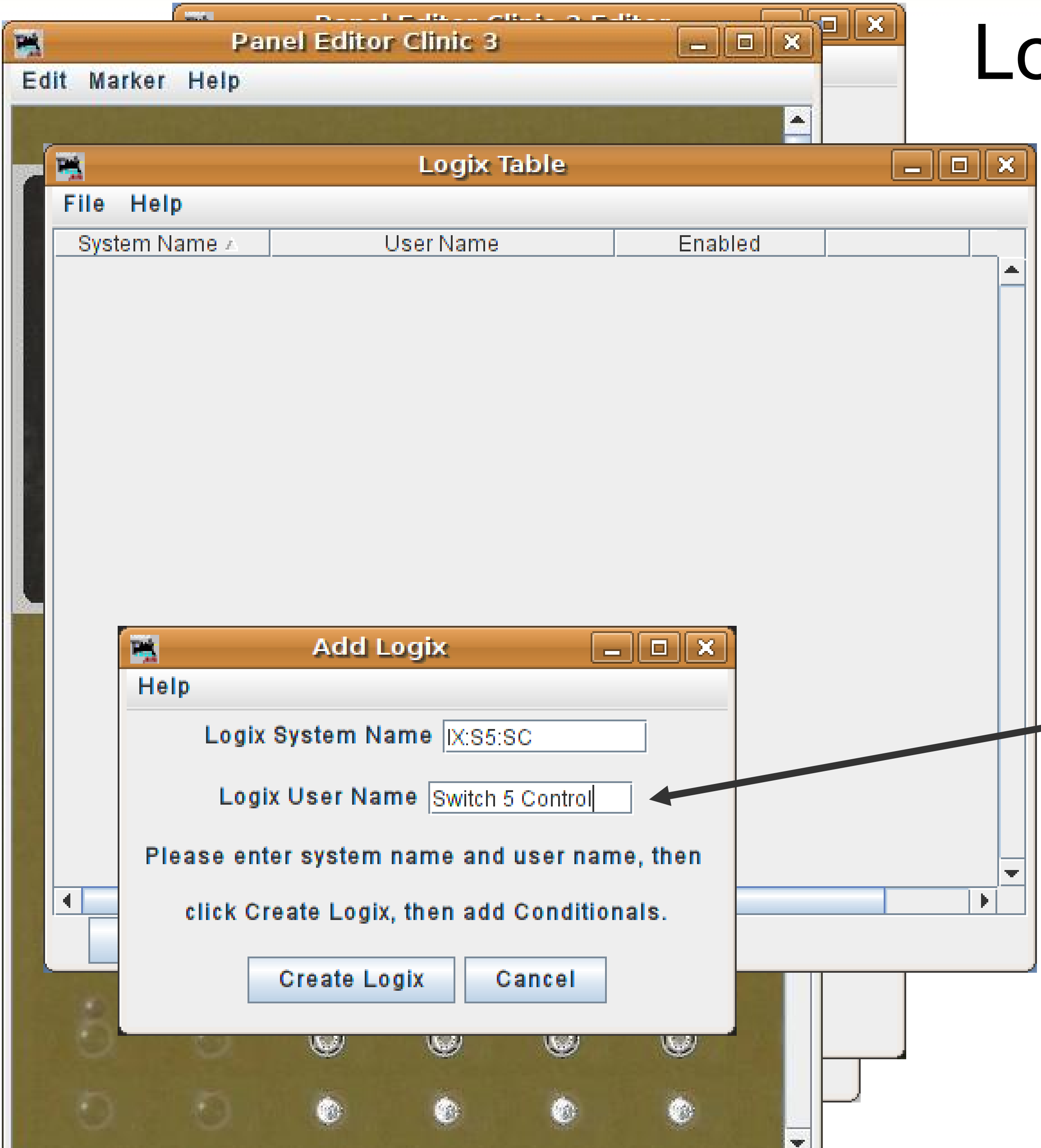
Logix User Name

Please enter system name and user name, then click Create Logix, then add Conditionals.

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'.
- I will call it IX:S5:SC

IX = **I**nternal Logi**X**,
:S5 = **S**witch **5**,
:SC = **S**witch **C**ontrol.



Logix System Name IX:S5:SC

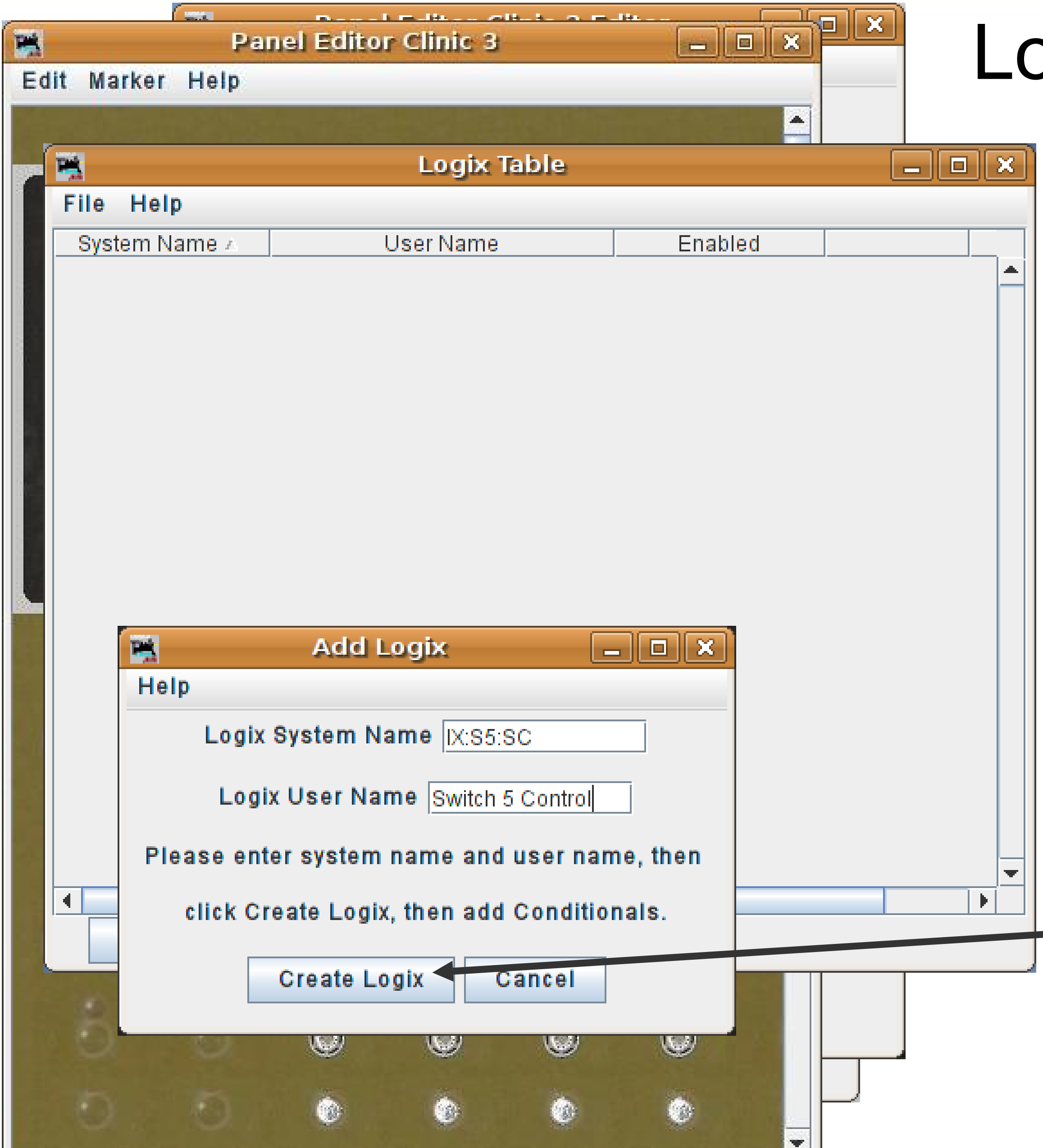
Logix User Name Switch 5 Control

Please enter system name and user name, then click Create Logix, then add Conditionals.

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 always start with 'IX'.
- Lets call it IX:S5:SC

  IX = **I**nternal Logi**X**,
  :S5 = **S**witch **5**,
  :SC = **S**witch **C**ontrol.
- The 'Logix User Name' may be any description we choose to use. 'Switch 5 Control' for example.



Logix System Name: IX:S5:SC

Logix User Name: Switch 5 Control

Please enter system name and user name, then click Create Logix, then add Conditionals.

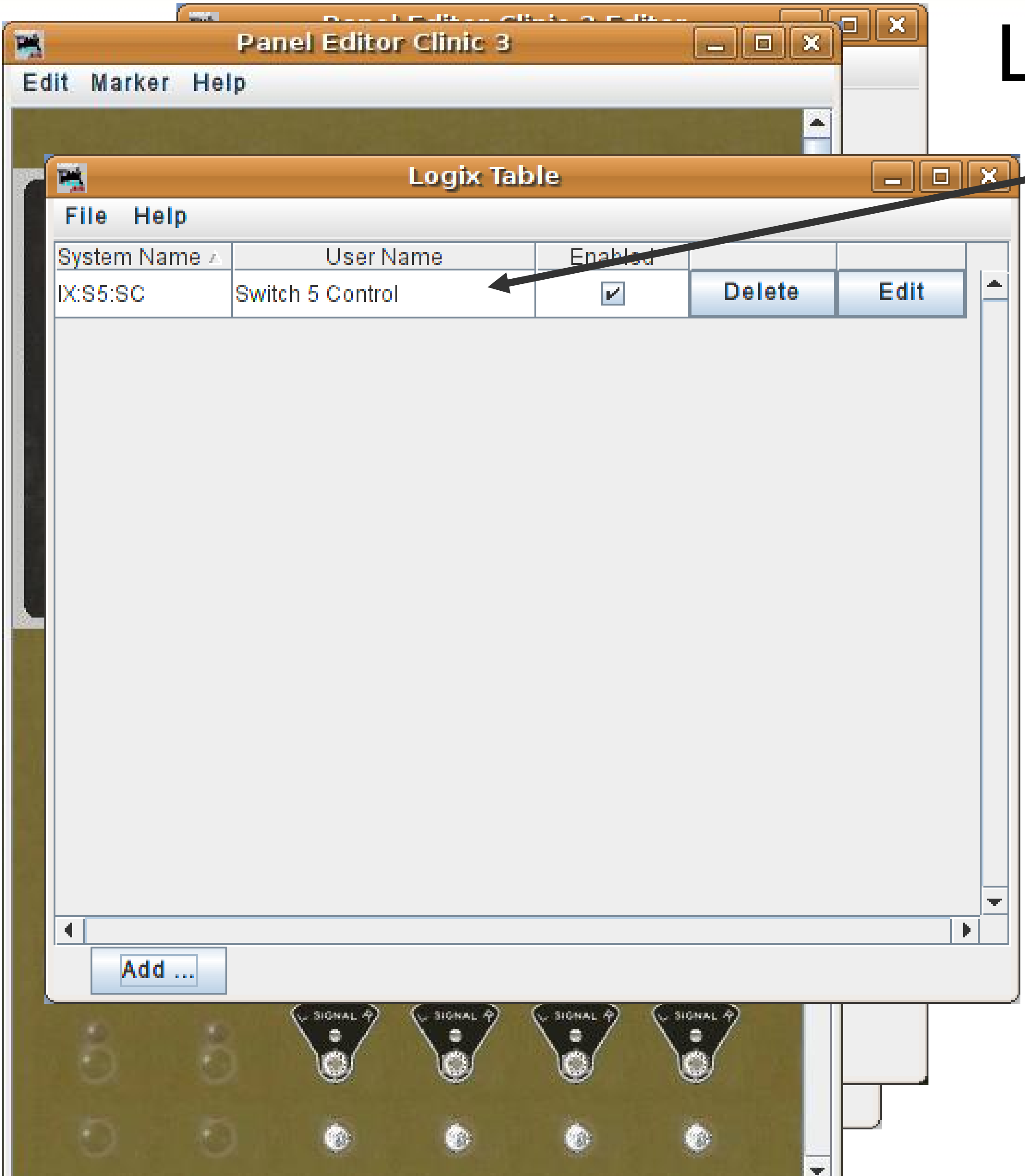
Logix naming

- The first information will be the ID. Logix are internal so the system name is 'I'. The item name is 'X', so they will always start with 'IX'.
- Lets call it IX:S5:SC

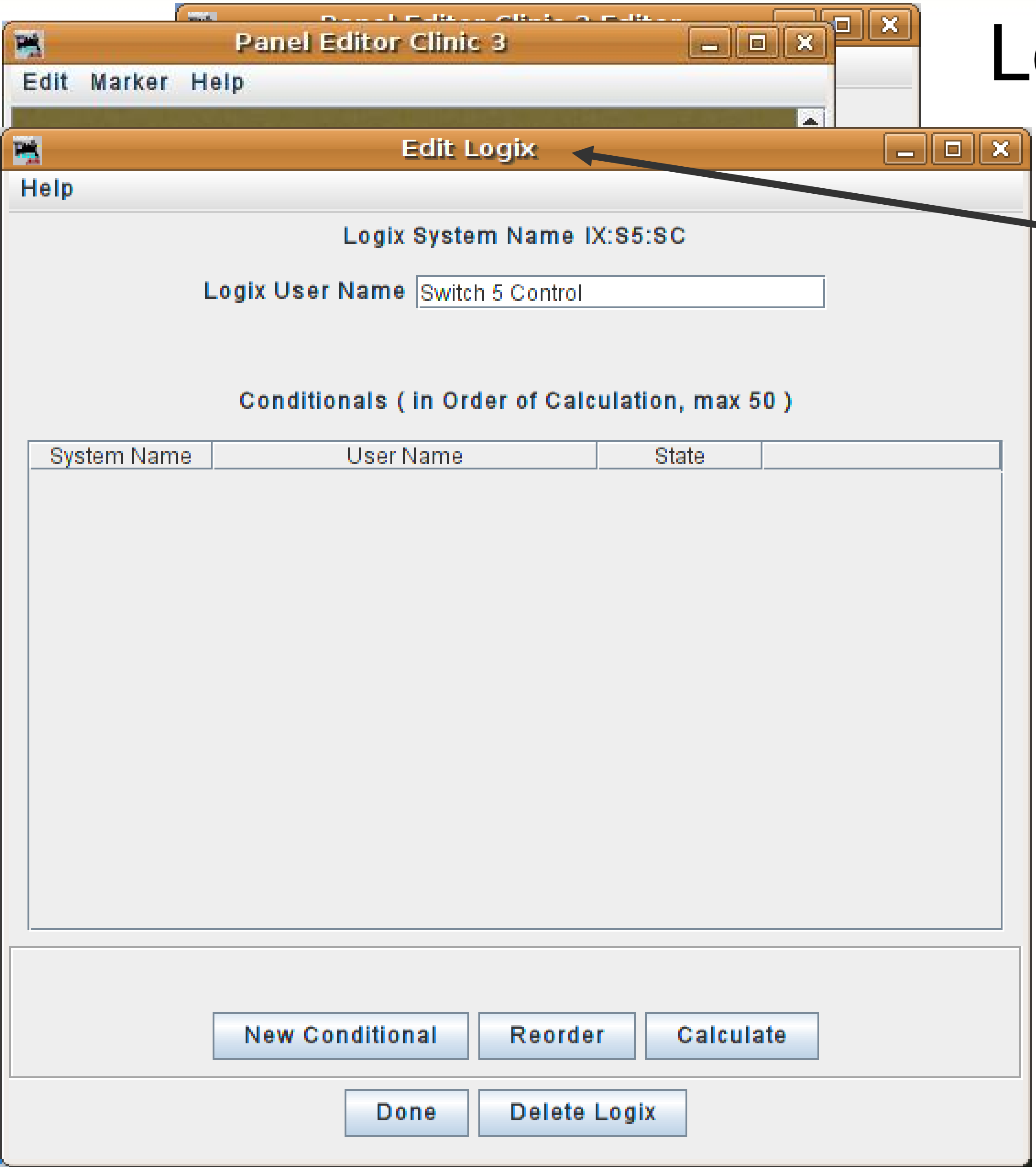
  IX = **I**nternal Logi**X**,
  :S5 = **S**witch **5**,
  :SC = **S**witch **C**ontrol.
- The 'Logix User Name' may be any description we choose to use. 'Switch 5 Control' for example.
- Once we have named our new creation click on 'Create Logix' to add it to the table and bring up a blank entry window.

Logix entry

- New table entry.



System Name	User Name	Enabled		
IX:S5:SC	Switch 5 Control	☑	Delete	Edit



Logix System Name IX:S5:SC

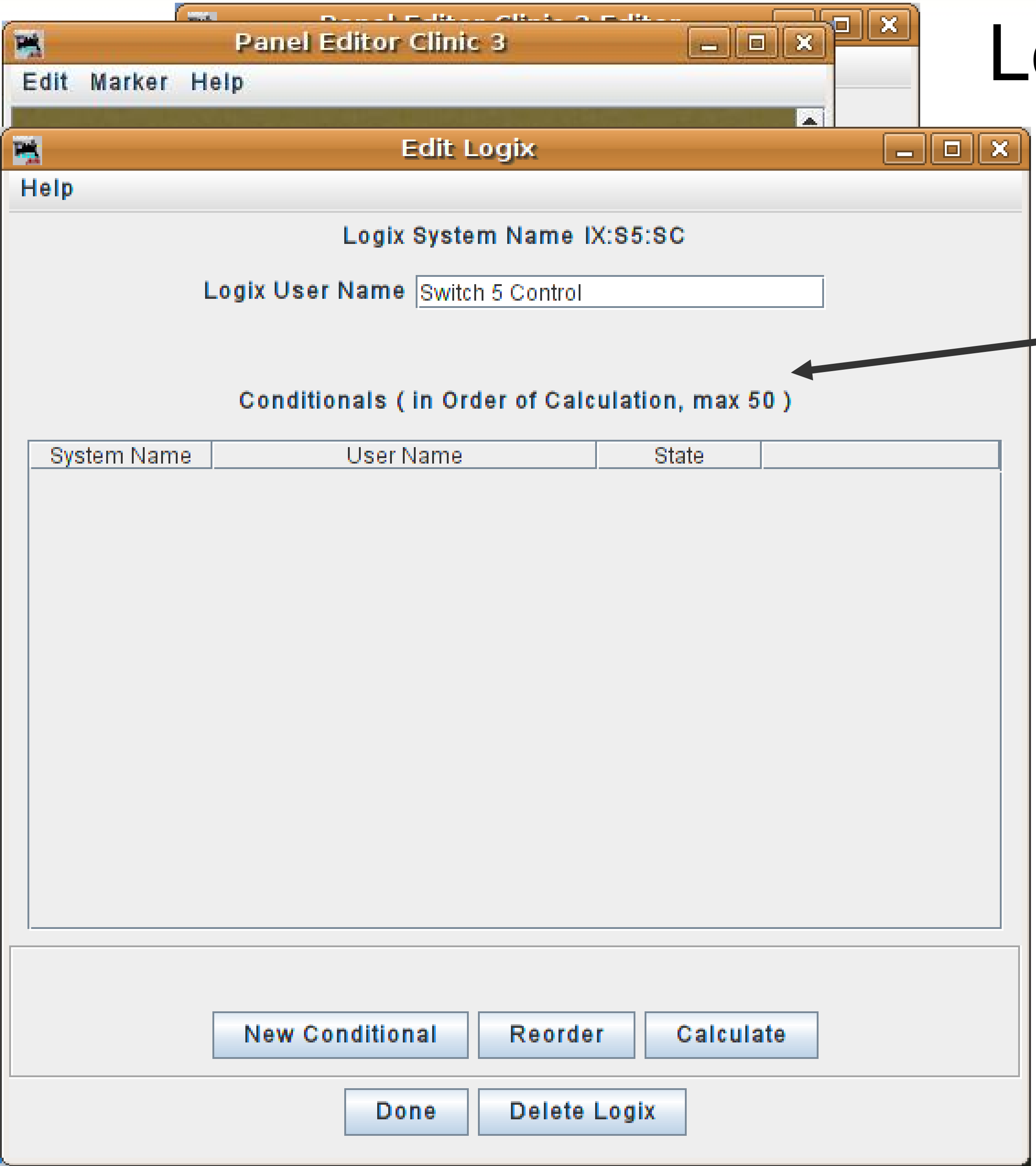
Logix User Name Switch 5 Control

Conditionals (in Order of Calculation, max 50)

System Name	User Name	State	

Logix entry

- New table entry.
- Edit Logix window.



Panel Editor Clinic 3

Logix System Name IX:S5:SC

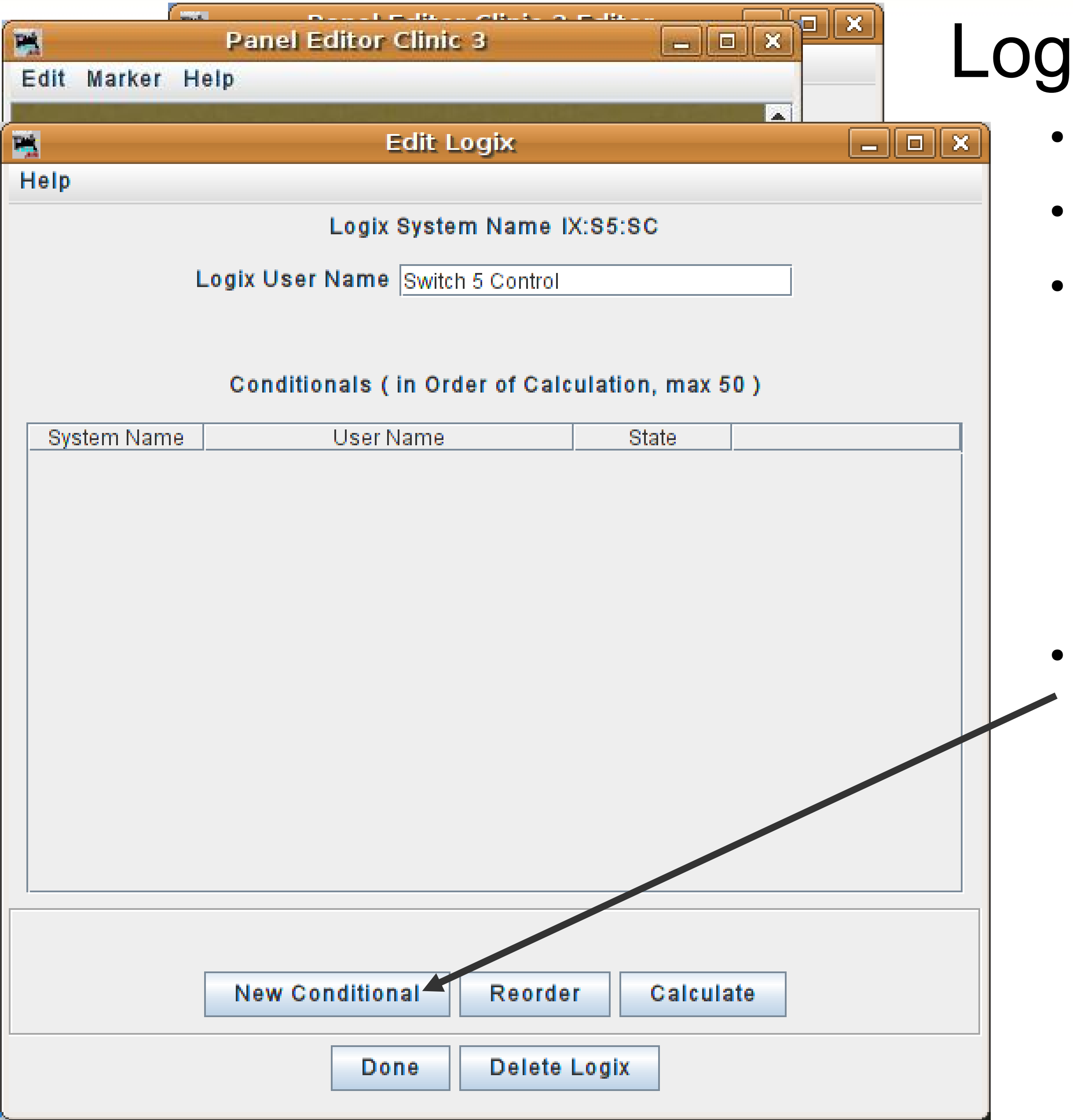
Logix User Name: Switch 5 Control

Conditionals (in Order of Calculation, max 50)

System Name	User Name	State	

Logix entry

- New table entry.
- Edit Logix window.
- Each Logix will contain one or more 'Conditionals' or things that may be true or false. A 'Conditional' may optionally do one or two actions when it becomes true or becomes false or simply changes state.



Logix System Name IX:S5:SC

Logix User Name: Switch 5 Control

Conditionals (in Order of Calculation, max 50)

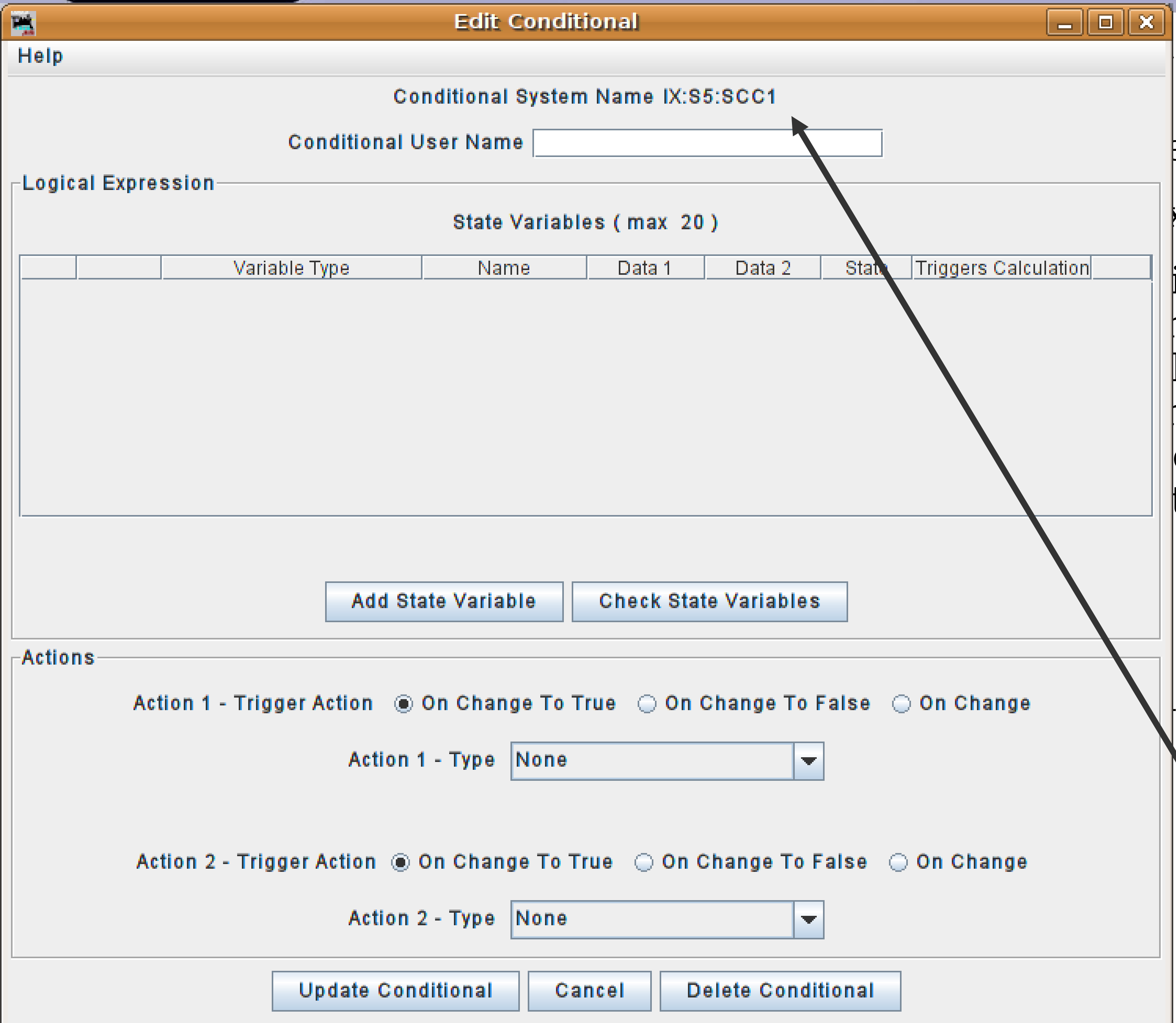
System Name	User Name	State	

Logix entry

- New table entry.
- Edit Logix window.
- Each Logix will contain one or more 'Conditionals' or things that may be true or false. A 'Conditional' may optionally do one or two 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



Conditional System Name IX:S5:SCC1

Conditional User Name

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation

Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 1 - Type None

Action 2 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type None

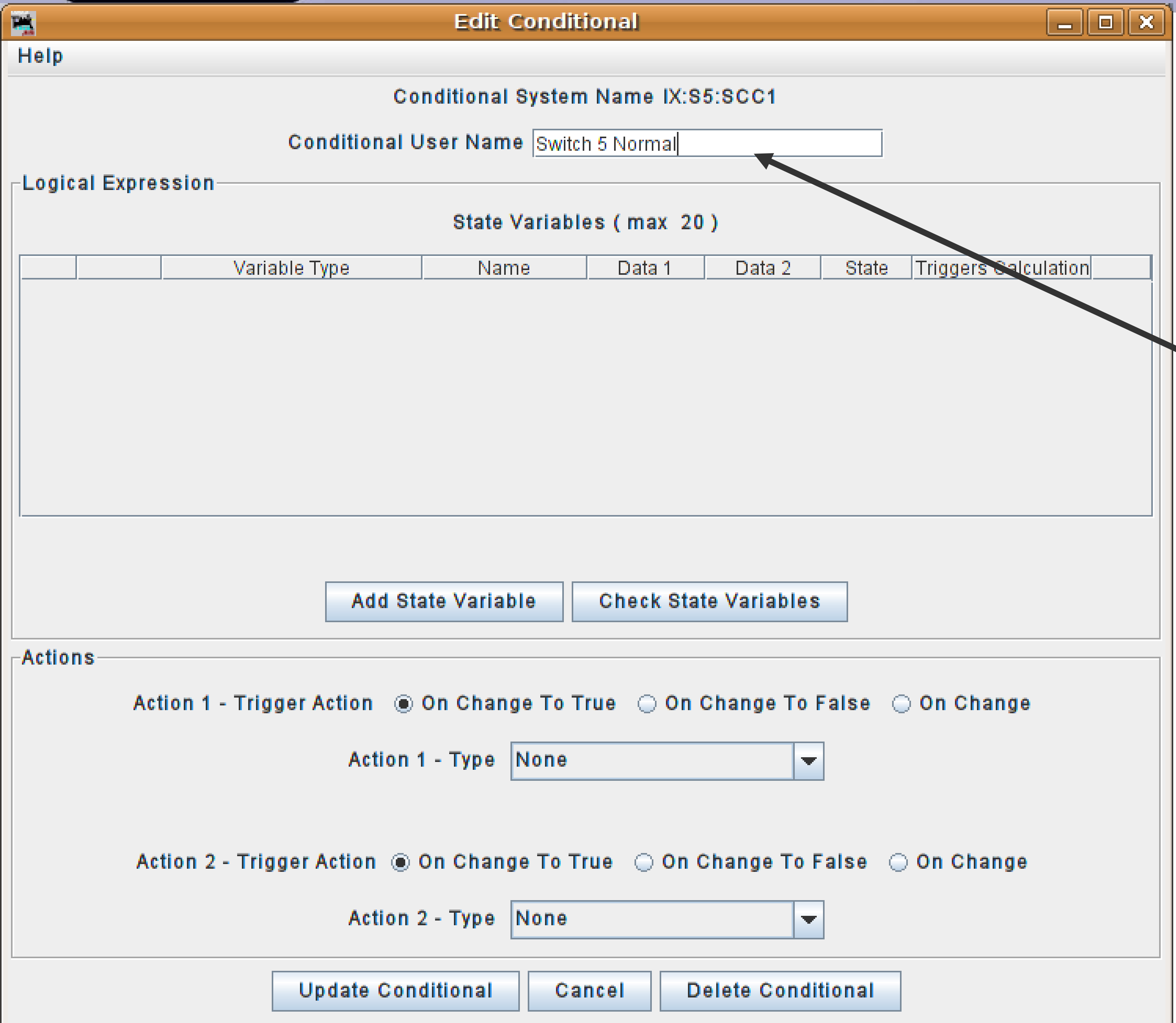
e entry.

x window.

ix will contain one or
nditionals' or things
be true or false. A
nal' may optionally do
o actions when it
true or becomes false
changes state.

'New Conditional'
bring up the 'Edit
al' window.

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



Edit Conditional

Help

Conditional System Name IX:S5:SCC1

Conditional User Name: Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	

Add State Variable | Check State Variables

Actions

Action 1 - Trigger Action: ● On Change To True ○ On Change To False ○ On Change

Action 1 - Type: None

Action 2 - Trigger Action: ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type: None

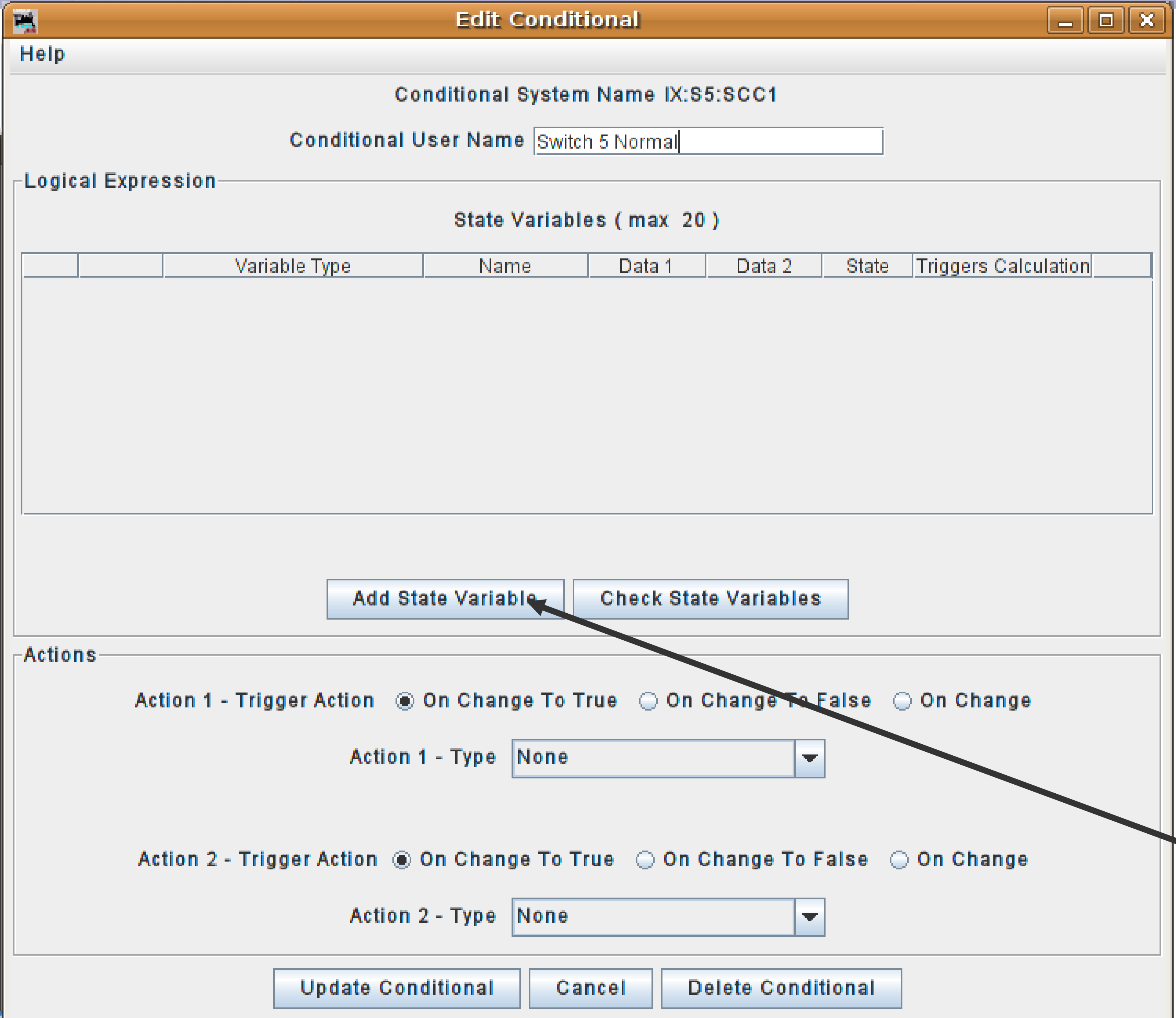
Update Conditional | Cancel | Delete Conditional

Logix entry

- We will create two conditionals, one to throw switch 5 and one to close it. Name this first one “Switch 5 Normal”.

Indirect Layout Control

Logix entry



Edit Conditional

Conditional System Name IX:S5:SCC1

Conditional User Name: Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation

Actions

Action 1 - Trigger Action: ● On Change To True ○ On Change To False ○ On Change

Action 1 - Type: None

Action 2 - Trigger Action: ● On Change To True ○ On Change To False ○ On Change

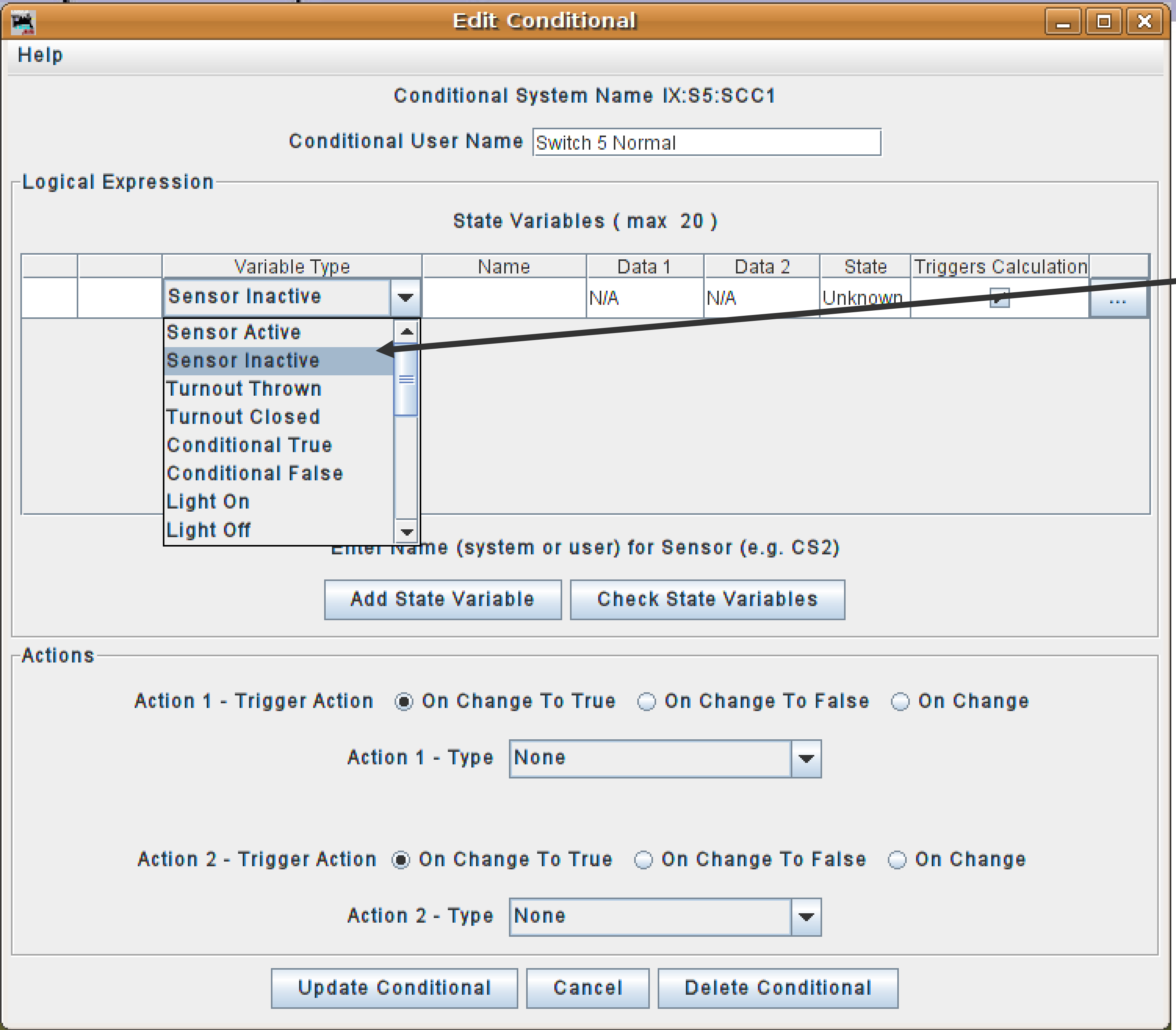
Action 2 - Type: None

Logix entry

- We will create two conditionals, one to throw switch 5 and one to close it. Name this first one “Switch 5 Normal”.
- We call the various items that will be checked 'Variables' because they vary between one value and another. In this case between being true and being false. Click here to add our first one.

Logix entry

- Click in the variable box to call up a list of possible options. Choose 'Sensor Inactive' because we don't want this to happen if a train is on the OS.



Conditional System Name IX:S5:SCC1

Conditional User Name: Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive		N/A	N/A	Unknown	☑	...

Enter Name (system or user) for Sensor (e.g. CS2)

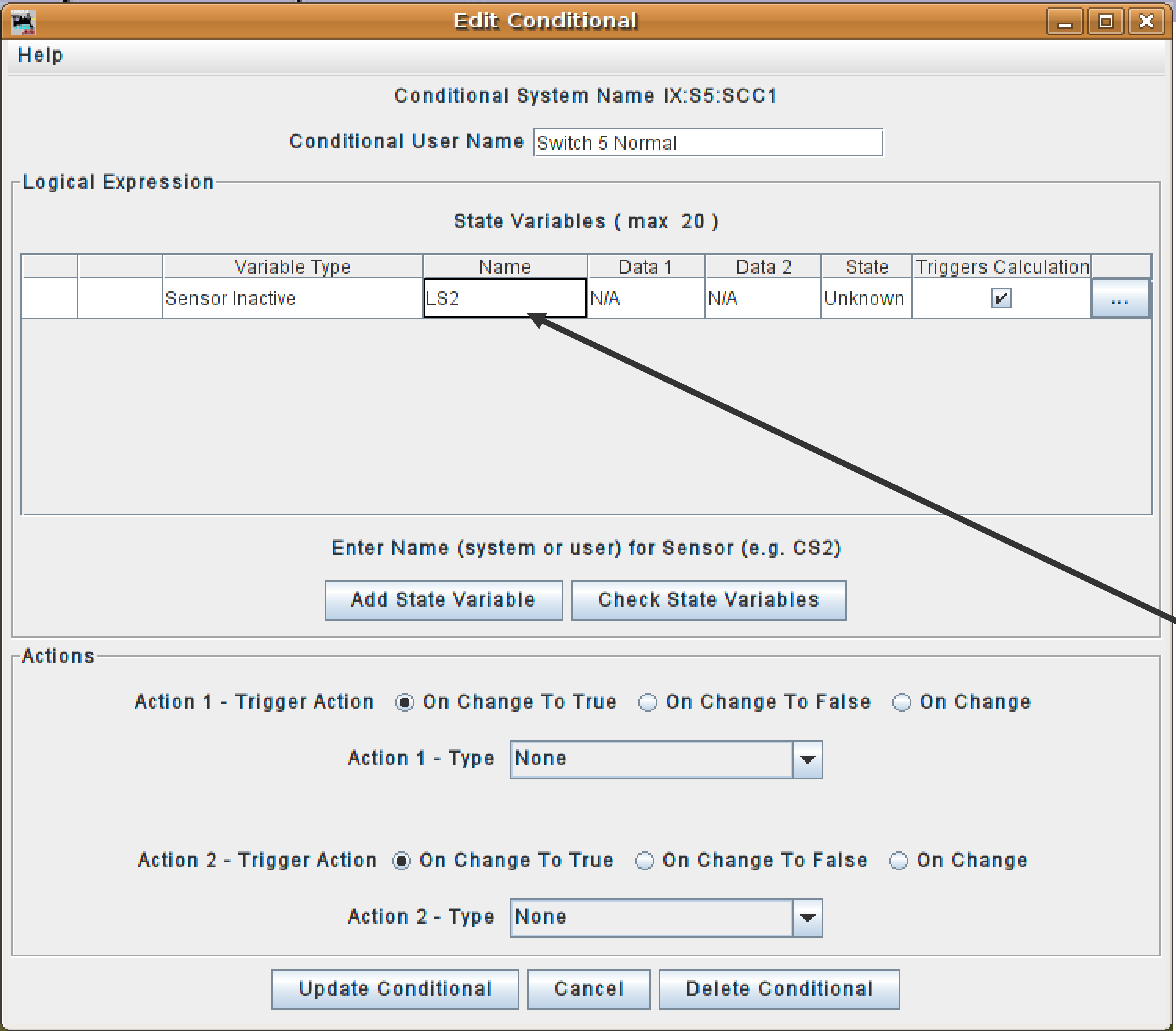
Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 1 - Type: None

Action 2 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type: None



Conditional System Name IX:S5:SCC1

Conditional User Name Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	Unknown	☑	...

Enter Name (system or user) for Sensor (e.g. CS2)

Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

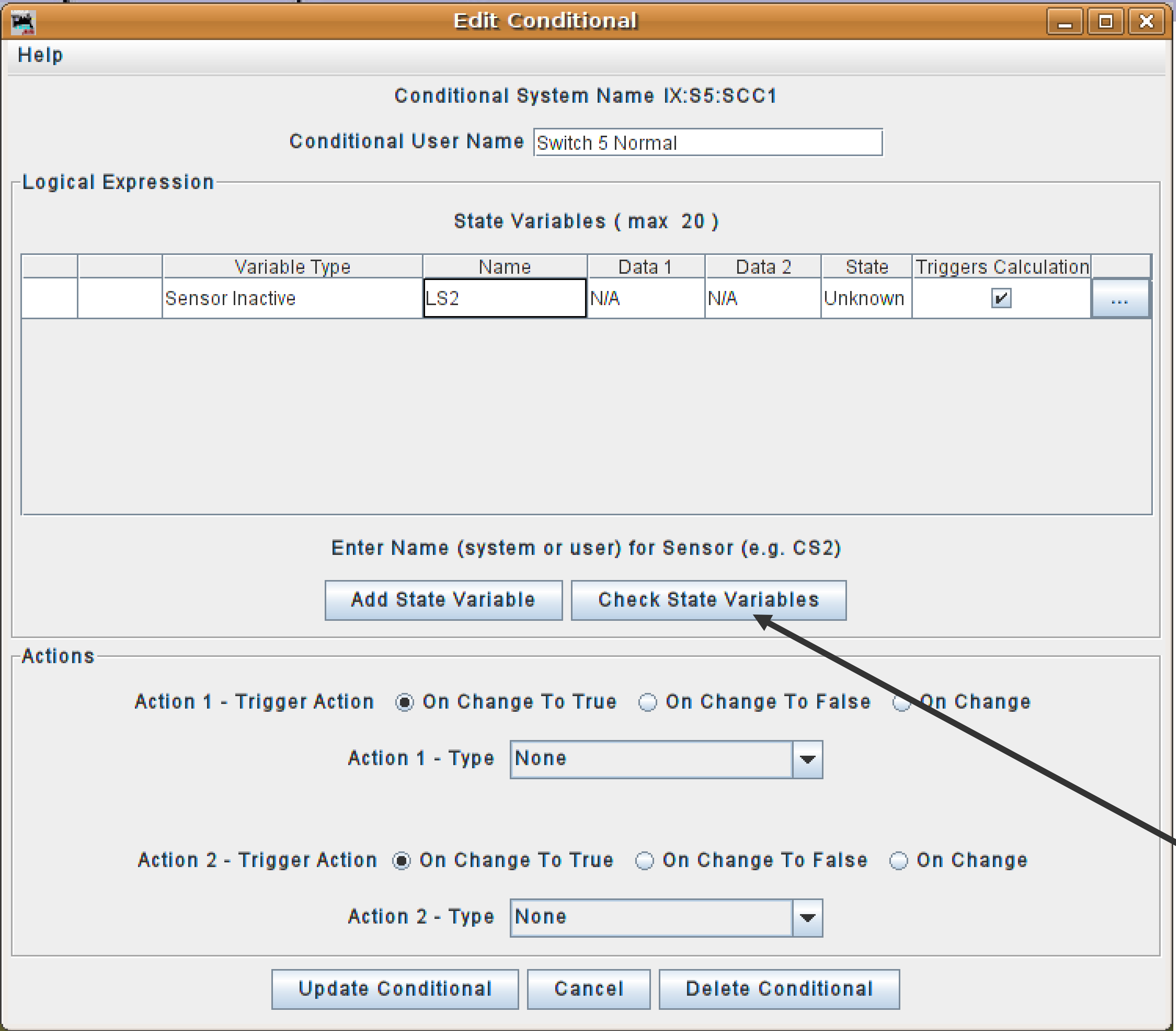
Action 1 - Type None

Action 2 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type None

Logix entry

- Click in the variable box to call up a list of possible options. Choose 'Sensor Inactive' because we don't want this action to happen if a train is on the OS.
- Enter the OS sensor ID. In this case it is 'LS2'.



Edit Conditional

Help

Conditional System Name IX:S5:SCC1

Conditional User Name: Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	Unknown	☑	...

Enter Name (system or user) for Sensor (e.g. CS2)

Actions

Action 1 - Trigger Action: ● On Change To True ○ On Change To False ○ On Change

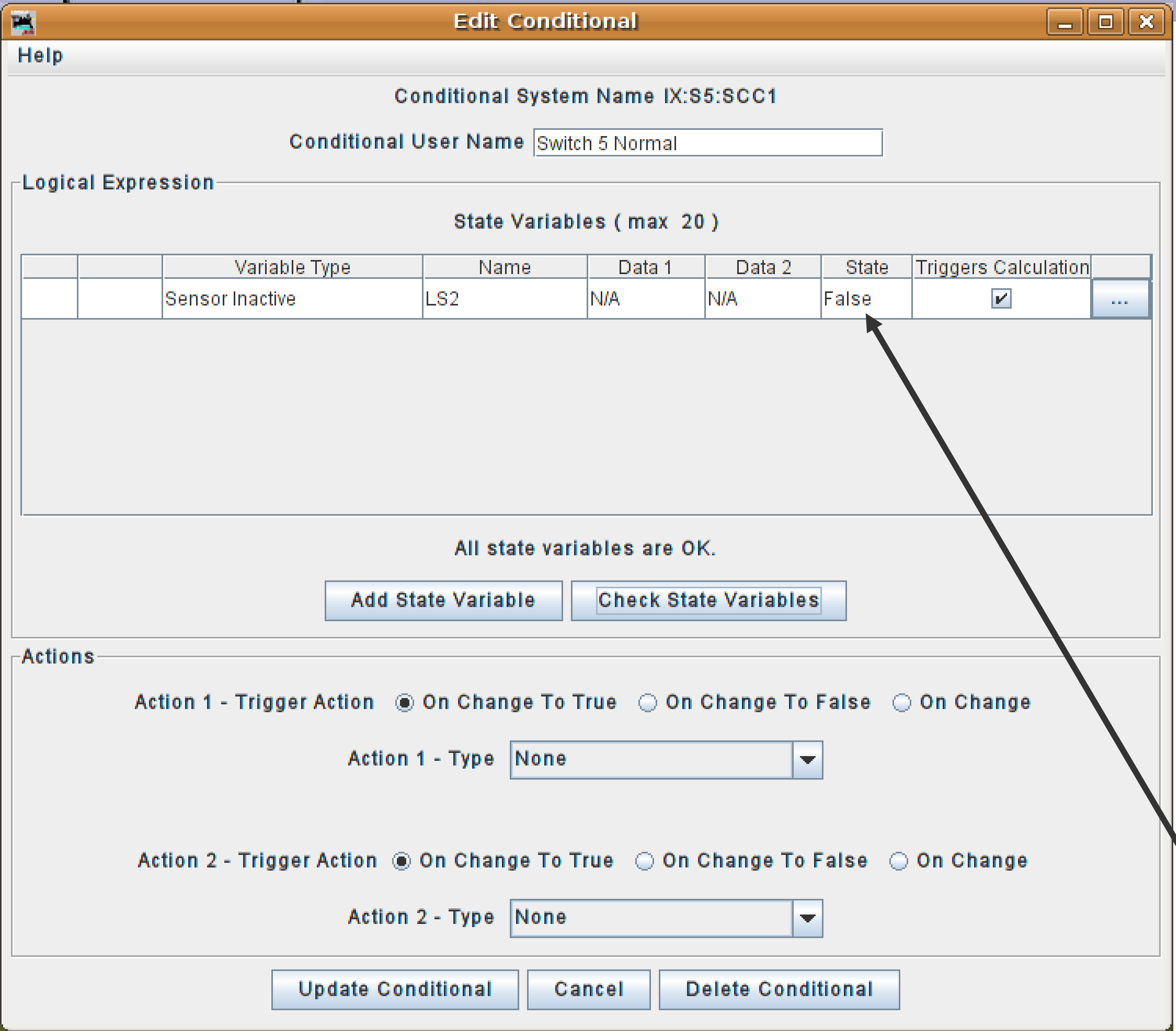
Action 1 - Type: None

Action 2 - Trigger Action: ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type: None

Logix entry

- Click in the variable box to call up a list of possible options. Choose 'Sensor Inactive' because we don't want this action to happen if a train is on the OS.
- Enter the OS sensor ID. In this case it is 'LS2'.
- Click on 'Check State Variables' to let the system read the newly added sensor item.



Conditional System Name IX:S5:SCC1

Conditional User Name Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	False	☑	...

All state variables are OK.

Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 1 - Type None

Action 2 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

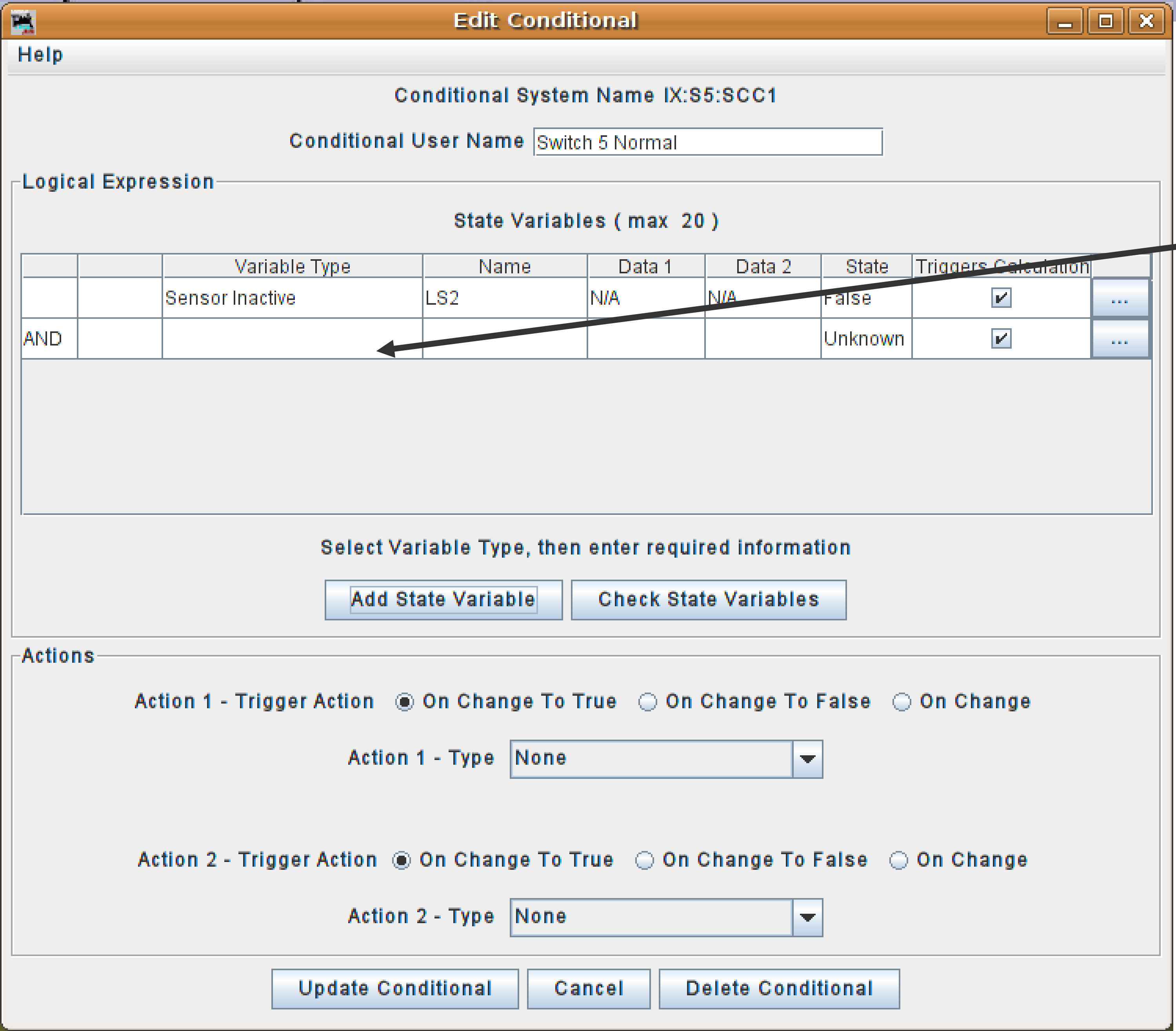
Action 2 - Type None

Logix entry

- Click in the variable box to call up a list of possible options. Choose 'Sensor Inactive' because we don't want this action to happen if a train is on the OS.
- Enter the OS sensor ID. In this case it is 'LS2'.
- Click on 'Check State Variables' to let the system read the newly added sensor item.
- Note that it is False.

Logix entry

- Add the second variable for the lever. (IS:S5:CL)



Conditional System Name IX:S5:SCC1

Conditional User Name: Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	False	☑	...
AND						Unknown	☑	...

Select Variable Type, then enter required information

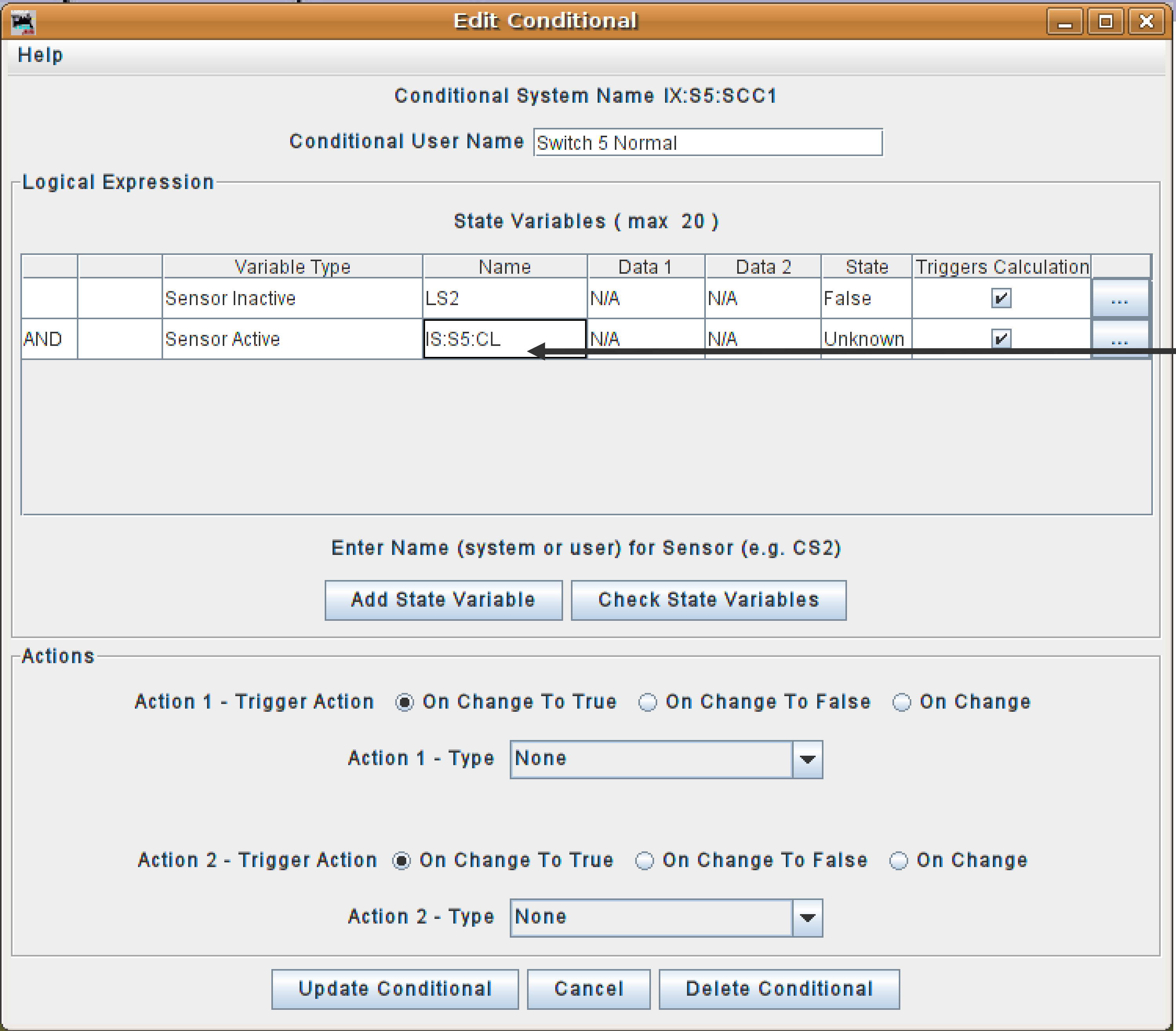
Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 1 - Type: None

Action 2 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type: None



Conditional System Name IX:S5:SCC1

Conditional User Name: Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	False	☑	...
AND		Sensor Active	IS:S5:CL	N/A	N/A	Unknown	☑	...

Enter Name (system or user) for Sensor (e.g. CS2)

Actions

Action 1 - Trigger Action: ● On Change To True ○ On Change To False ○ On Change

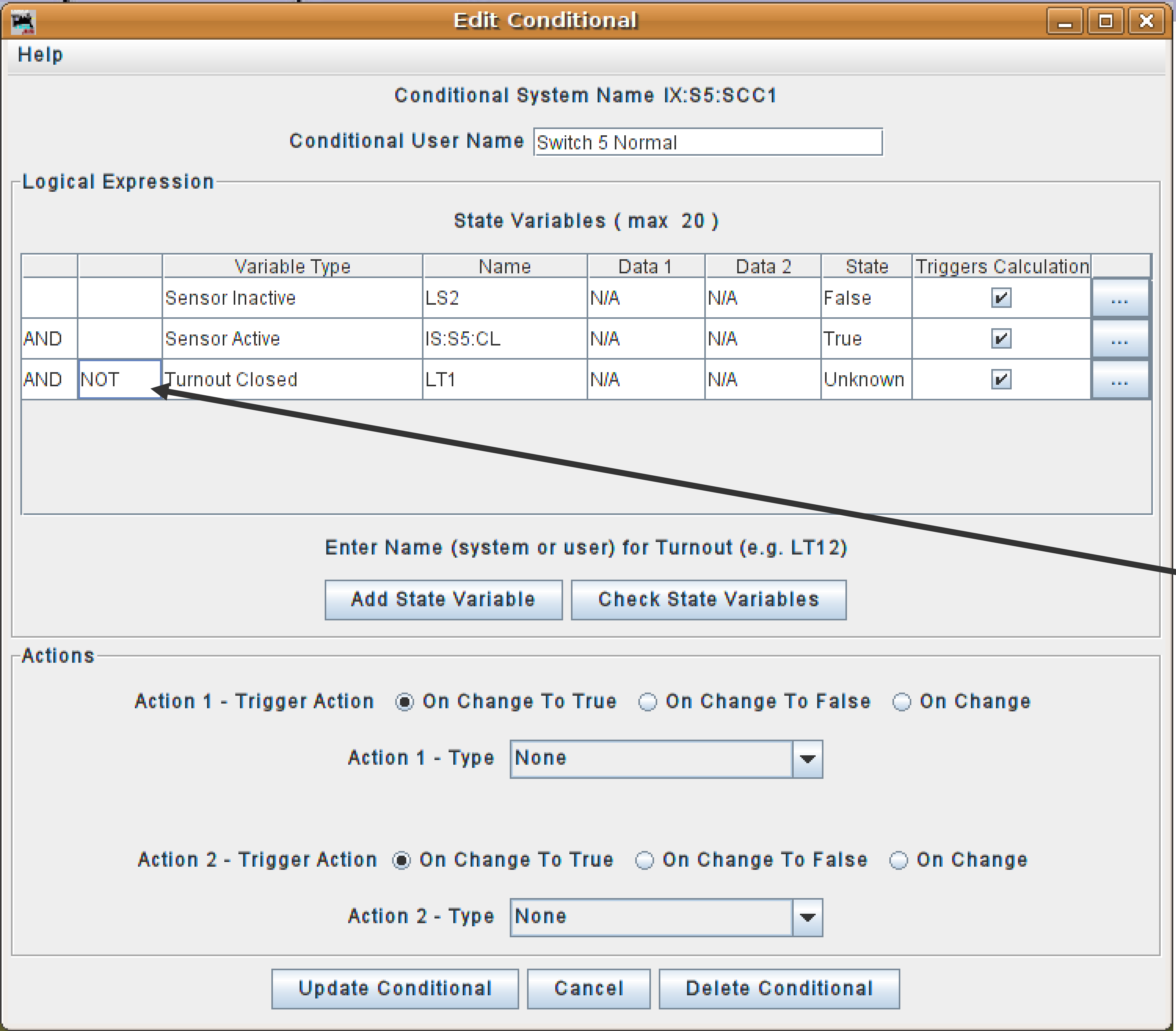
Action 1 - Type: None

Action 2 - Trigger Action: ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type: None

Logix entry

- Add the second variable for the lever. (IS:S5:CL)
- It is 'Active' when in the 'N' (Normal) position.



Edit Conditional

Conditional System Name IX:S5:SCC1

Conditional User Name Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	False	☑	...
AND		Sensor Active	IS:S5:CL	N/A	N/A	True	☑	...
AND	NOT	Turnout Closed	LT1	N/A	N/A	Unknown	☑	...

Enter Name (system or user) for Turnout (e.g. LT12)

Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

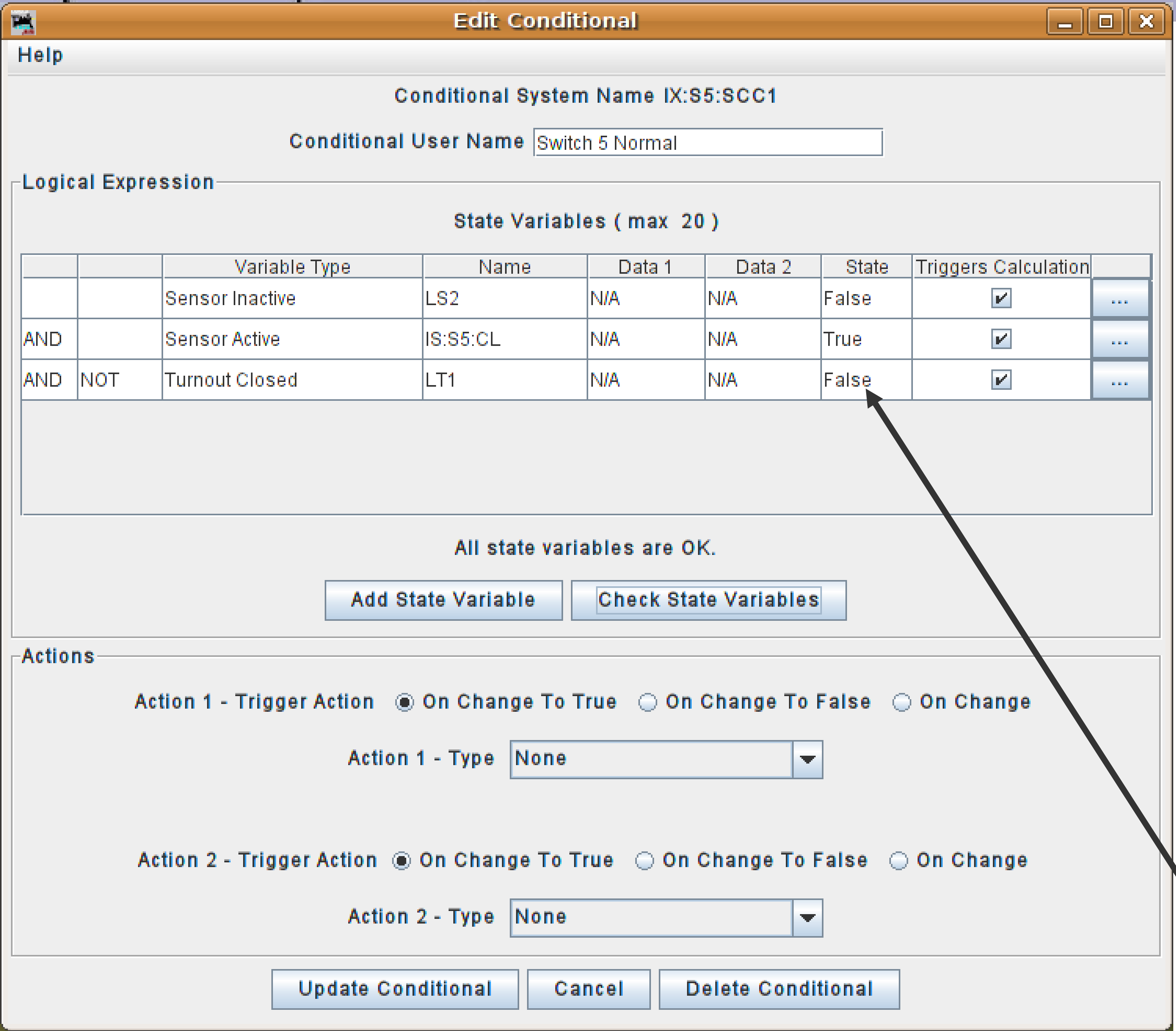
Action 1 - Type None

Action 2 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type None

Logix entry

- Add the second variable for the lever. (IS:S5:CL)
- It is 'Active' when in the 'N' (Normal) position.
- We only want to send a command if the turnout is NOT already in position. This is not the same as being 'Thrown' because it could be moving or unknown.



Edit Conditional

Conditional System Name IX:S5:SCC1

Conditional User Name: Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	False	☑	...
AND		Sensor Active	IS:S5:CL	N/A	N/A	True	☑	...
AND	NOT	Turnout Closed	LT1	N/A	N/A	False	☑	...

All state variables are OK.

Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

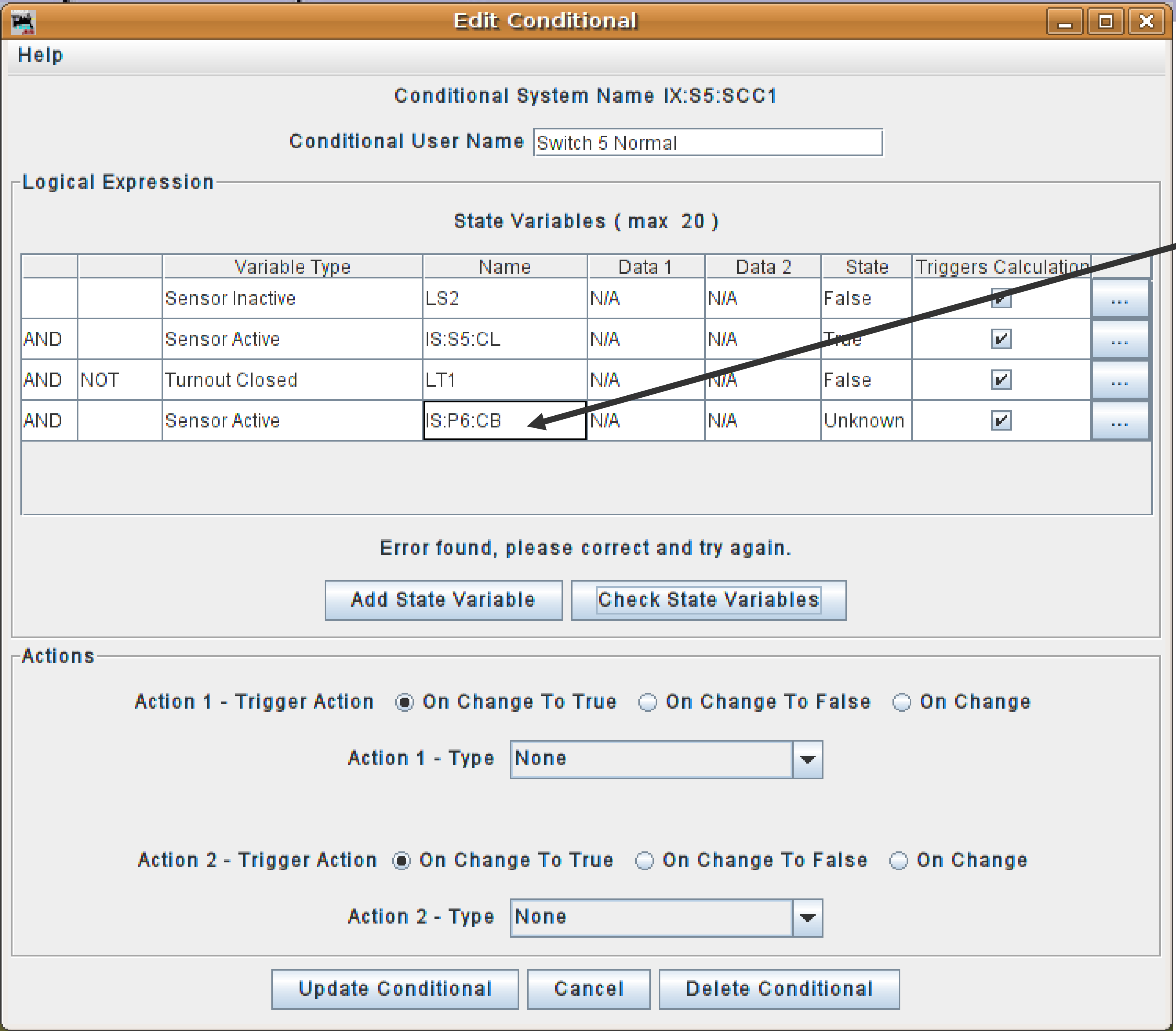
Action 1 - Type: None

Action 2 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type: None

Logix entry

- Add the second variable for the lever. (IS:S5:CL)
- It is 'Active' when in the 'N' (Normal) position.
- We only want to send a command if the turnout is NOT already in position. This is not the same as being 'Thrown' because it could be moving or unknown.
- A state check shows that 'Not closed' is 'False' because it is closed. (not not)



Edit Conditional

Conditional System Name IX:S5:SCC1

Conditional User Name: Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	False	☑	...
AND		Sensor Active	IS:S5:CL	N/A	N/A	True	☑	...
AND	NOT	Turnout Closed	LT1	N/A	N/A	False	☑	...
AND		Sensor Active	IS:P6:CB	N/A	N/A	Unknown	☑	...

Error found, please correct and try again.

Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

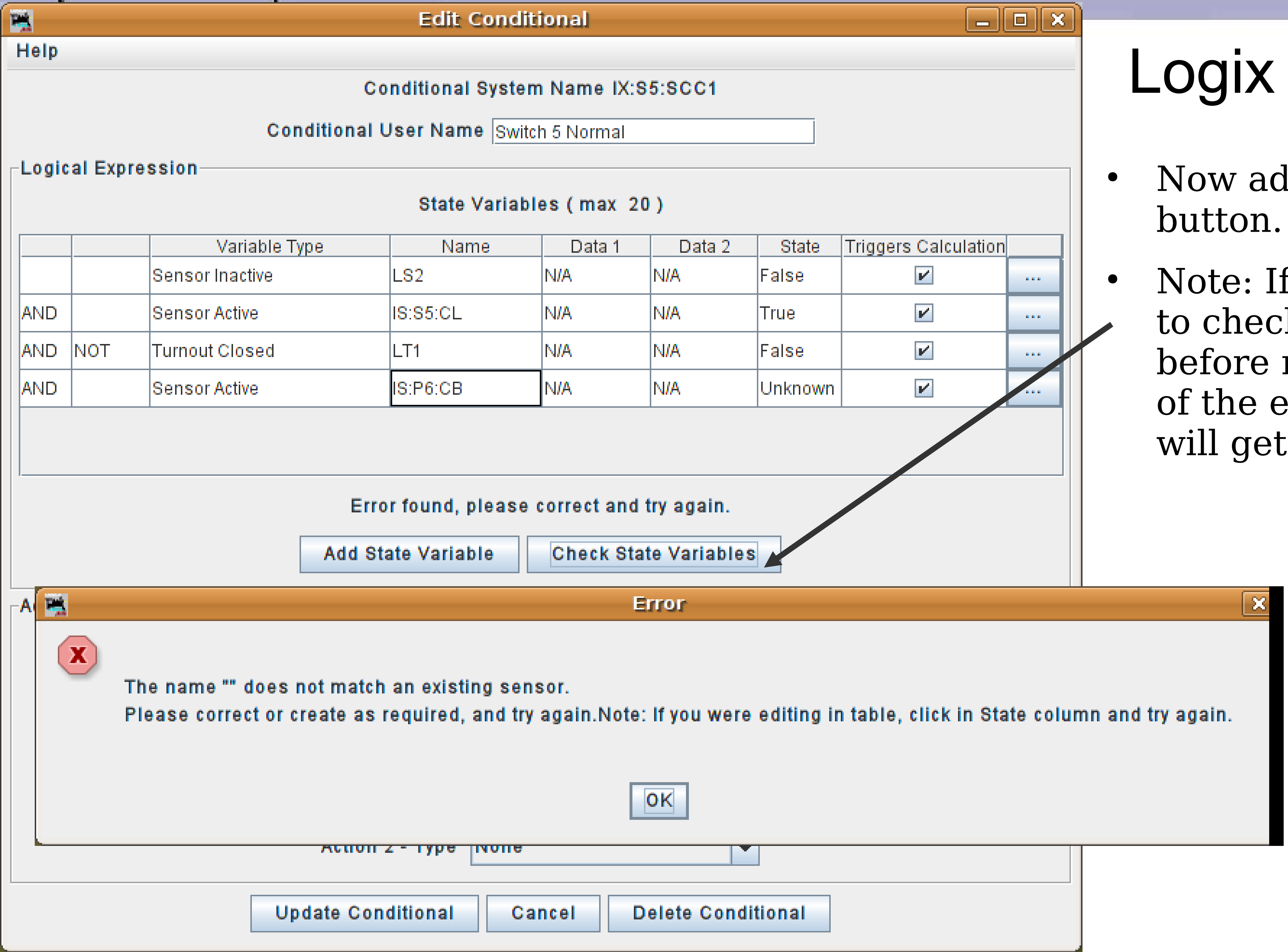
Action 1 - Type: None

Action 2 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type: None

Logix entry

- Now add our code button.



Edit Conditional

Conditional System Name IX:S5:SCC1

Conditional User Name Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	False	☑	...
AND		Sensor Active	IS:S5:CL	N/A	N/A	True	☑	...
AND	NOT	Turnout Closed	LT1	N/A	N/A	False	☑	...
AND		Sensor Active	IS:P6:CB	N/A	N/A	Unknown	☑	...

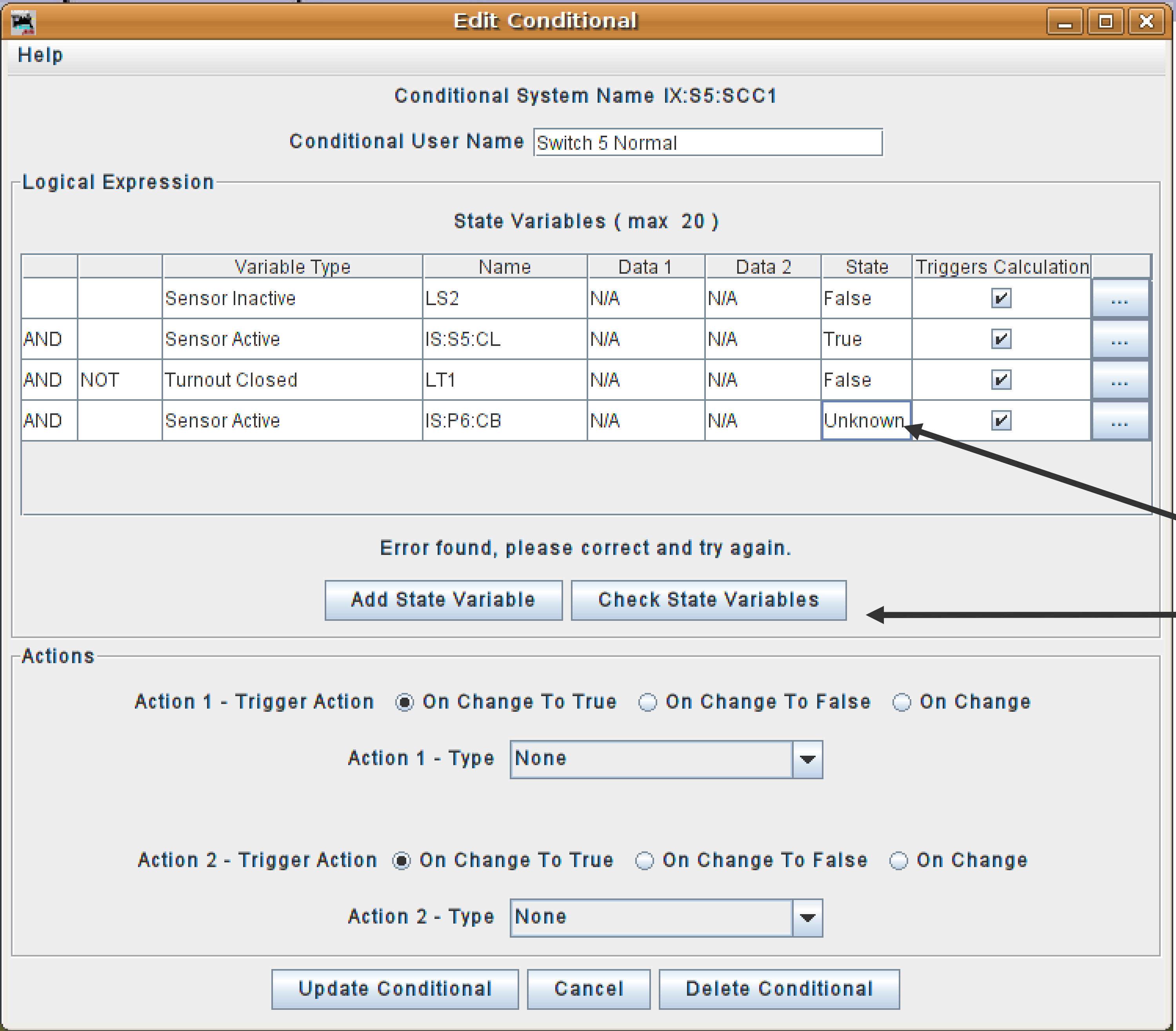
Error found, please correct and try again.

Error

The name "" does not match an existing sensor.
Please correct or create as required, and try again.Note: If you were editing in table, click in State column and try again.

Logix entry

- Now add our code button.
- Note: If we attempt to check the state before moving out of the entry box we will get an error.



Conditional System Name IX:S5:SCC1

Conditional User Name Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	False	☑	...
AND		Sensor Active	IS:S5:CL	N/A	N/A	True	☑	...
AND	NOT	Turnout Closed	LT1	N/A	N/A	False	☑	...
AND		Sensor Active	IS:P6:CB	N/A	N/A	Unknown	☑	...

Error found, please correct and try again.

Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

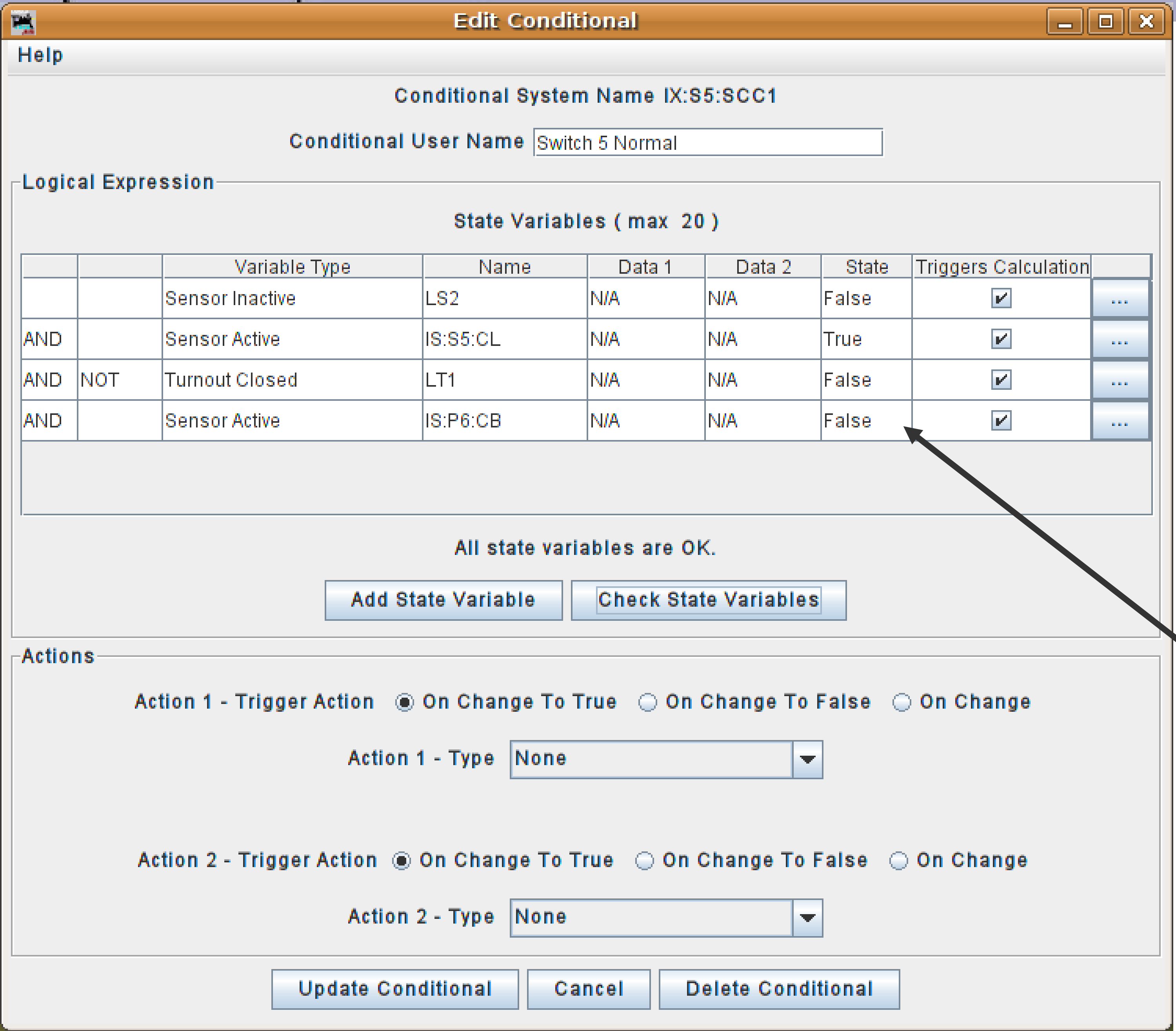
Action 1 - Type None

Action 2 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type None

Logix entry

- Now add our code button.
- Note: If we attempt to check the state before moving out of the entry box we will get an error.
- Simply click in the state column box first then check it.



Edit Conditional

Help

Conditional System Name IX:S5:SCC1

Conditional User Name: Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	False	☑	...
AND		Sensor Active	IS:S5:CL	N/A	N/A	True	☑	...
AND	NOT	Turnout Closed	LT1	N/A	N/A	False	☑	...
AND		Sensor Active	IS:P6:CB	N/A	N/A	False	☑	...

All state variables are OK.

Add State Variable | Check State Variables

Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 1 - Type: None

Action 2 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type: None

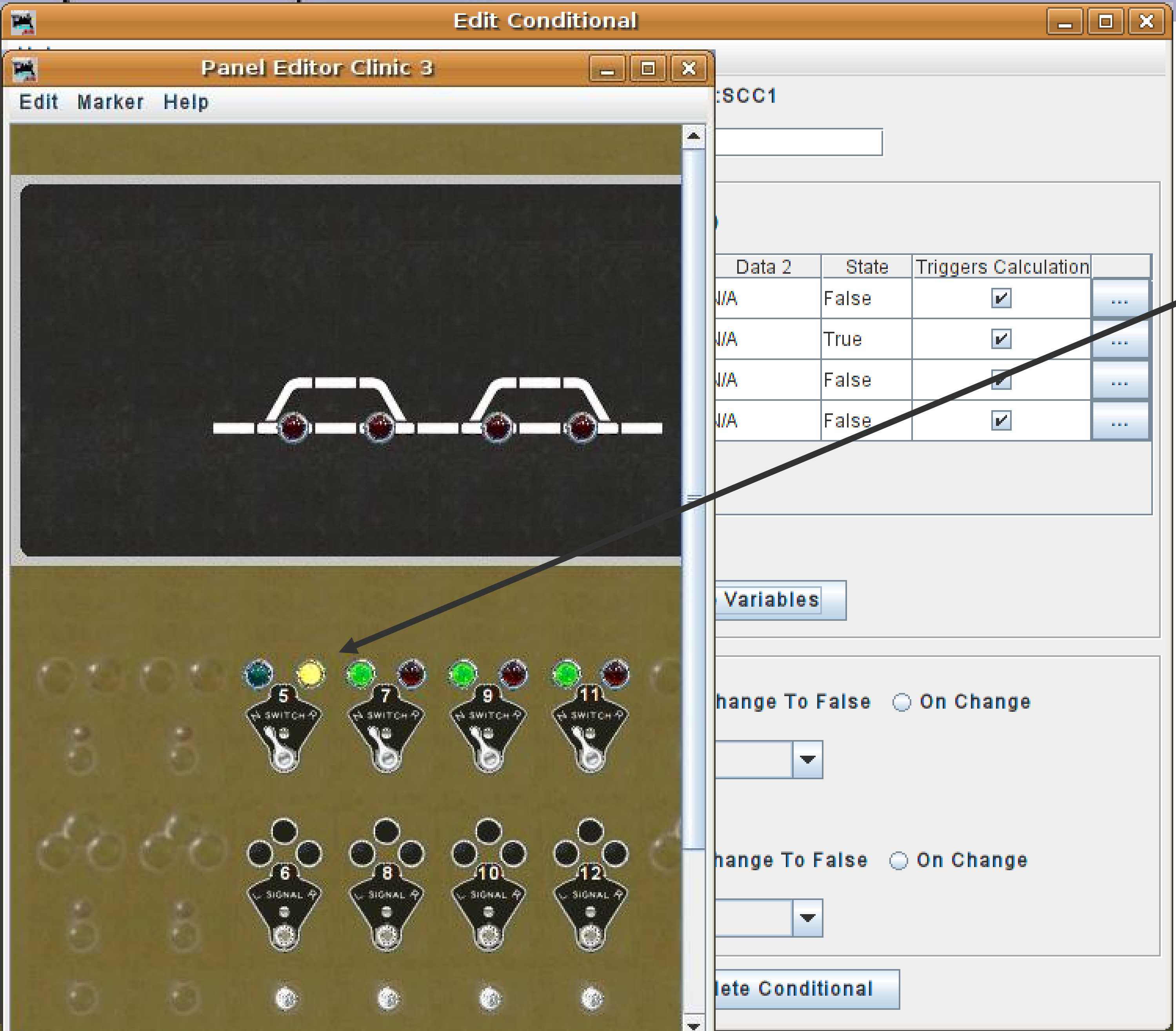
Update Conditional | Cancel | Delete Conditional

Logix entry

- Now add our code button.
- Note: If we attempt to check the state before moving out of the entry box we will get an error.
- Simply click in the state column box first then check it.
- As you can see we have piled up many reasons that we will not send a turnout command at this time, even if we do press the button.

Logix entry

- Change the turnout to reverse and move the train. (click on the icons)



Edit Conditional

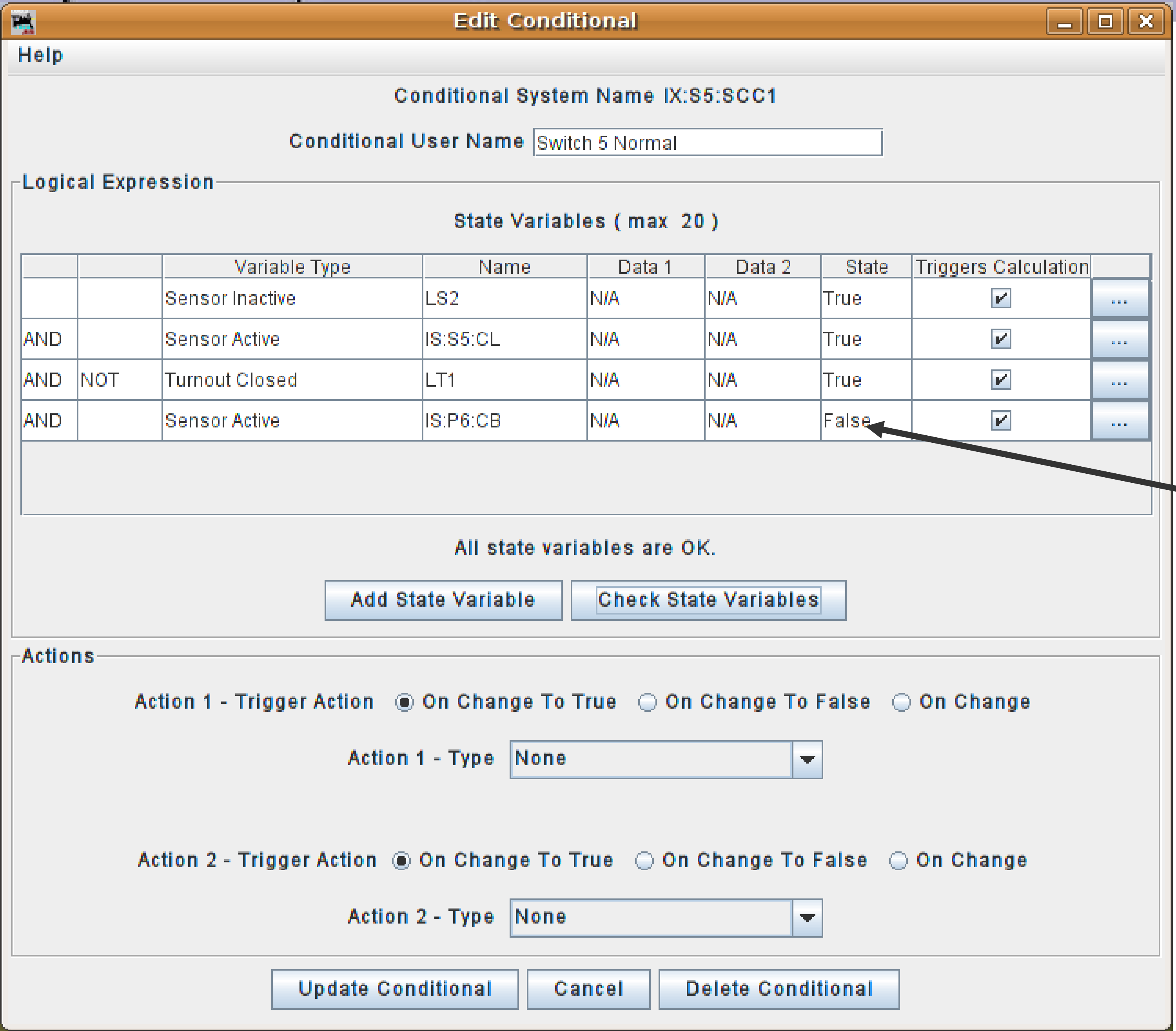
Panel Editor Clinic 3

:SCC1

Data 2	State	Triggers Calculation	
I/A	False	☑	...
I/A	True	☑	...
I/A	False	☑	...
I/A	False	☑	...

hange To False ○ On Change

hange To False ○ On Change



Conditional System Name IX:S5:SCC1

Conditional User Name: Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	True	☑	...
AND		Sensor Active	IS:S5:CL	N/A	N/A	True	☑	...
AND	NOT	Turnout Closed	LT1	N/A	N/A	True	☑	...
AND		Sensor Active	IS:P6:CB	N/A	N/A	False	☑	...

All state variables are OK.

Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

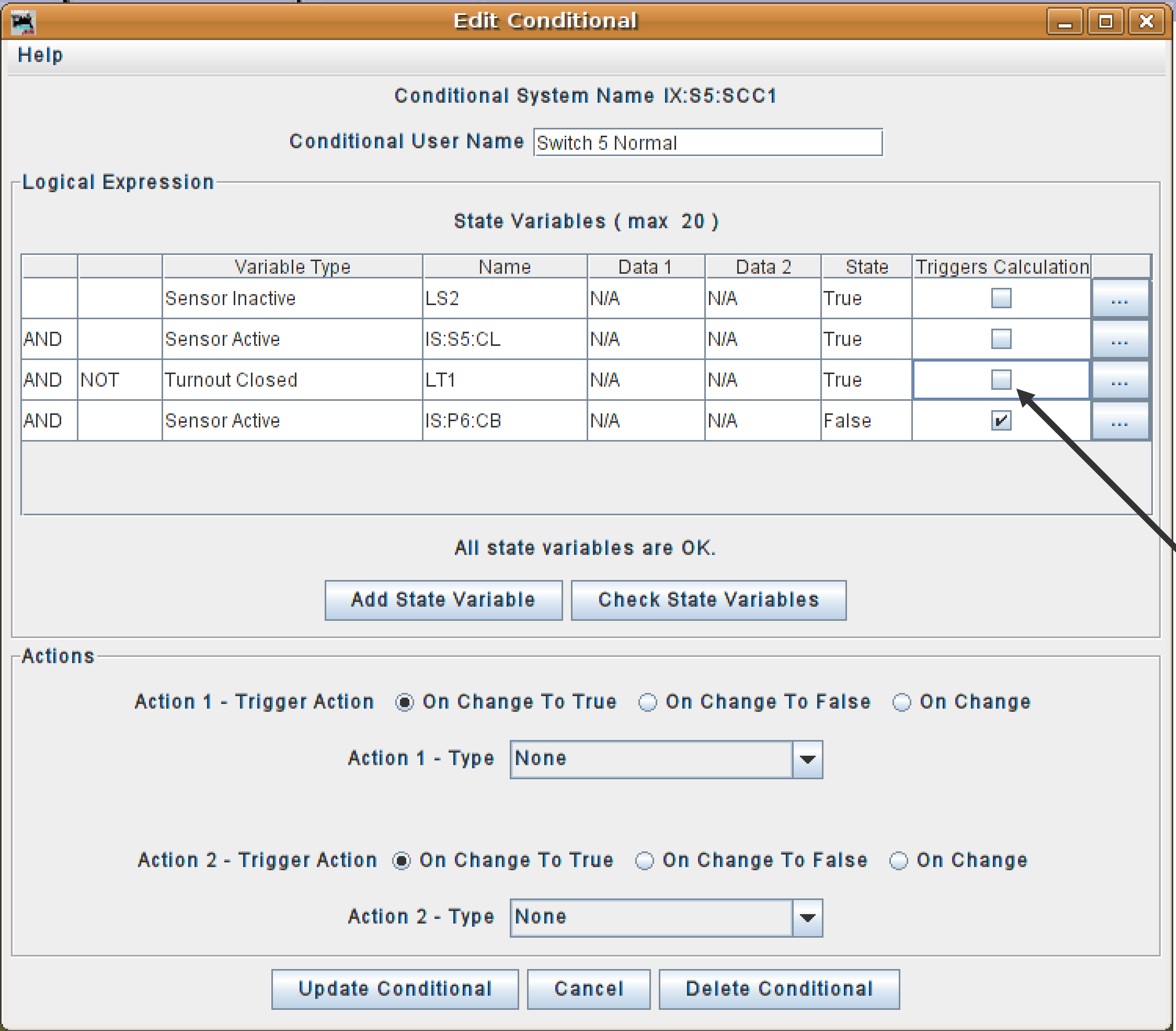
Action 1 - Type: None

Action 2 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type: None

Logix entry

- Change the turnout to reverse and move the train. (click on the icons)
- Now a check of the states shows that only the button press is False.



Edit Conditional

Help

Conditional System Name IX:S5:SCC1

Conditional User Name: Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	True	□	...
AND		Sensor Active	IS:S5:CL	N/A	N/A	True	□	...
AND	NOT	Turnout Closed	LT1	N/A	N/A	True	□	...
AND		Sensor Active	IS:P6:CB	N/A	N/A	False	☑	...

All state variables are OK.

Add State Variable | Check State Variables

Actions

Action 1 - Trigger Action: ● On Change To True ○ On Change To False ○ On Change

Action 1 - Type: None

Action 2 - Trigger Action: ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type: None

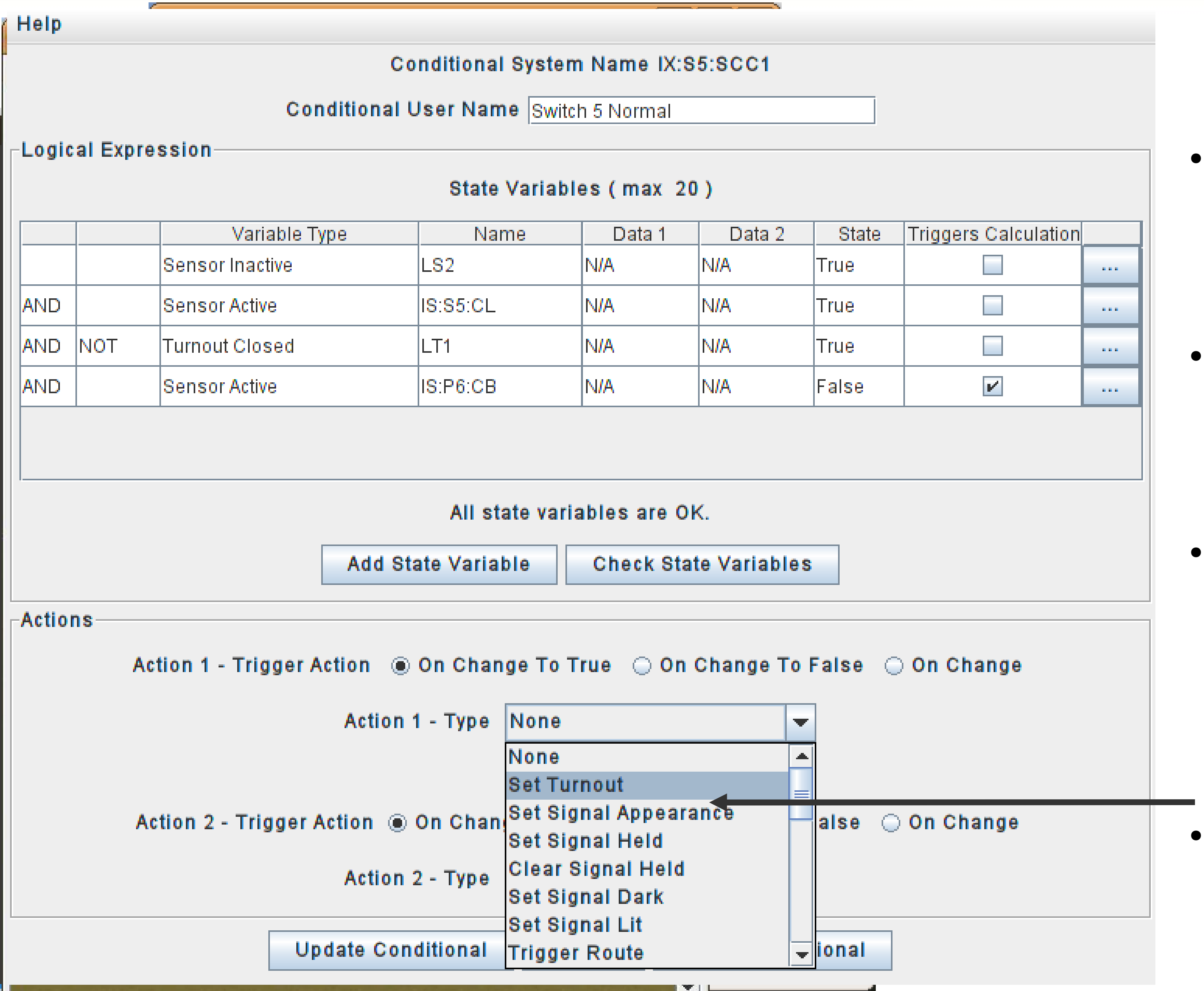
Update Conditional | Cancel | Delete Conditional

Logix entry

- Change the turnout to reverse and move the train. (click on the icons)
- Now a check of the states shows that only the button press is False.
- Actually the button press is the only item we want to attempt to trigger the conditional, so uncheck the others.

Indirect Layout Control

Logix entry



Conditional System Name IX:S5:SCC1

Conditional User Name: Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	True	□	...
AND		Sensor Active	IS:S5:CL	N/A	N/A	True	□	...
AND	NOT	Turnout Closed	LT1	N/A	N/A	True	□	...
AND		Sensor Active	IS:P6:CB	N/A	N/A	False	☑	...

All state variables are OK.

Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

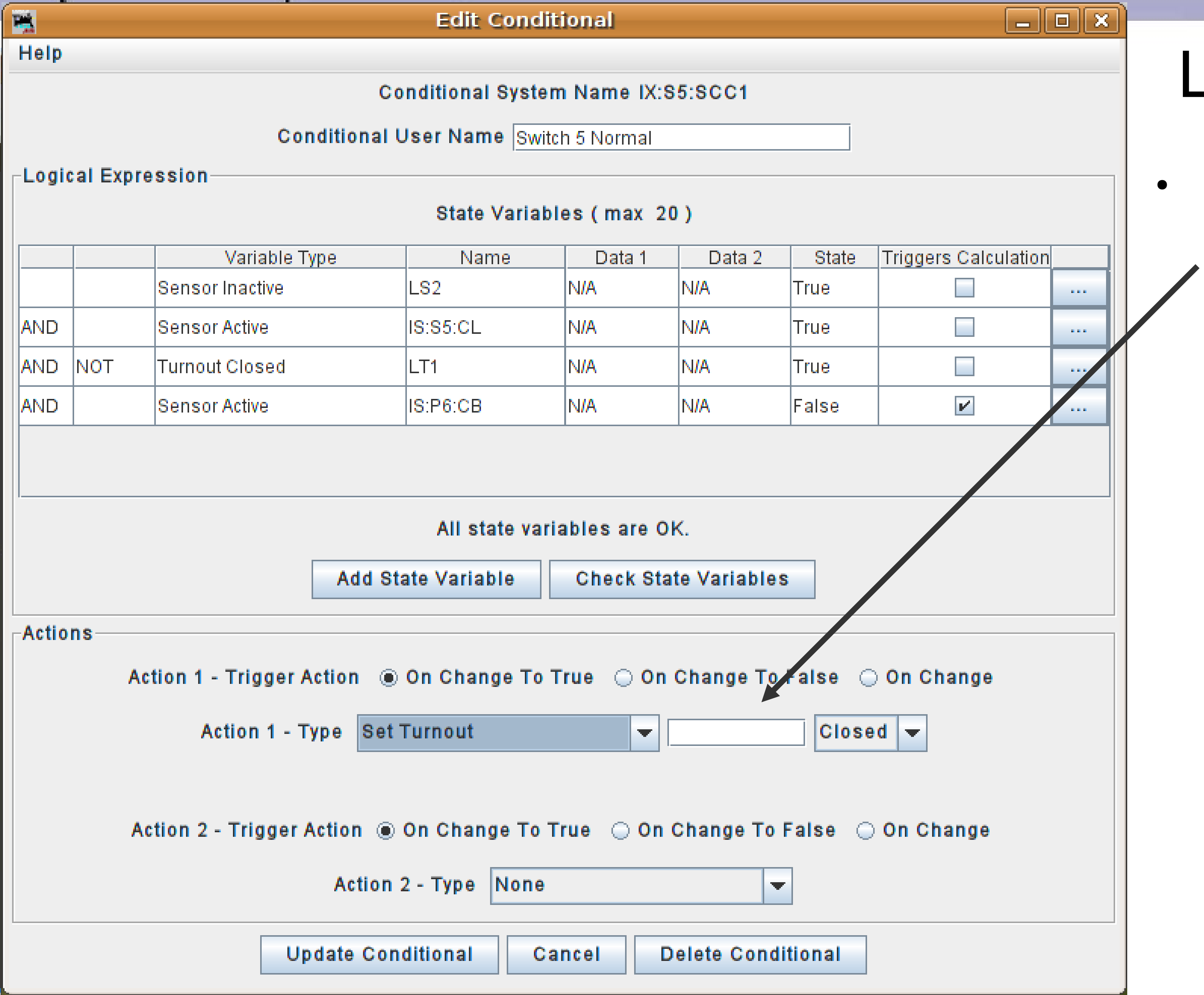
Action 1 - Type: None

Action 2 - Trigger Action ● On Chan alse ○ On Change

Action 2 - Type

Logix entry

- Change the turnout to reverse and move the train. (click on the icons)
- Now a check of the states shows that only the button press is False.
- Actually the button press is the only item we want to attempt to trigger the conditional, so uncheck the others.
- Now we choose our action. Select 'Set Turnout'.



Edit Conditional

Conditional System Name IX:S5:SCC1

Conditional User Name Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	True	□	...
AND		Sensor Active	IS:S5:CL	N/A	N/A	True	□	...
AND	NOT	Turnout Closed	LT1	N/A	N/A	True	□	...
AND		Sensor Active	IS:P6:CB	N/A	N/A	False	☑	...

All state variables are OK.

Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 1 - Type Set Turnout ▾ Closed ▾

Action 2 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

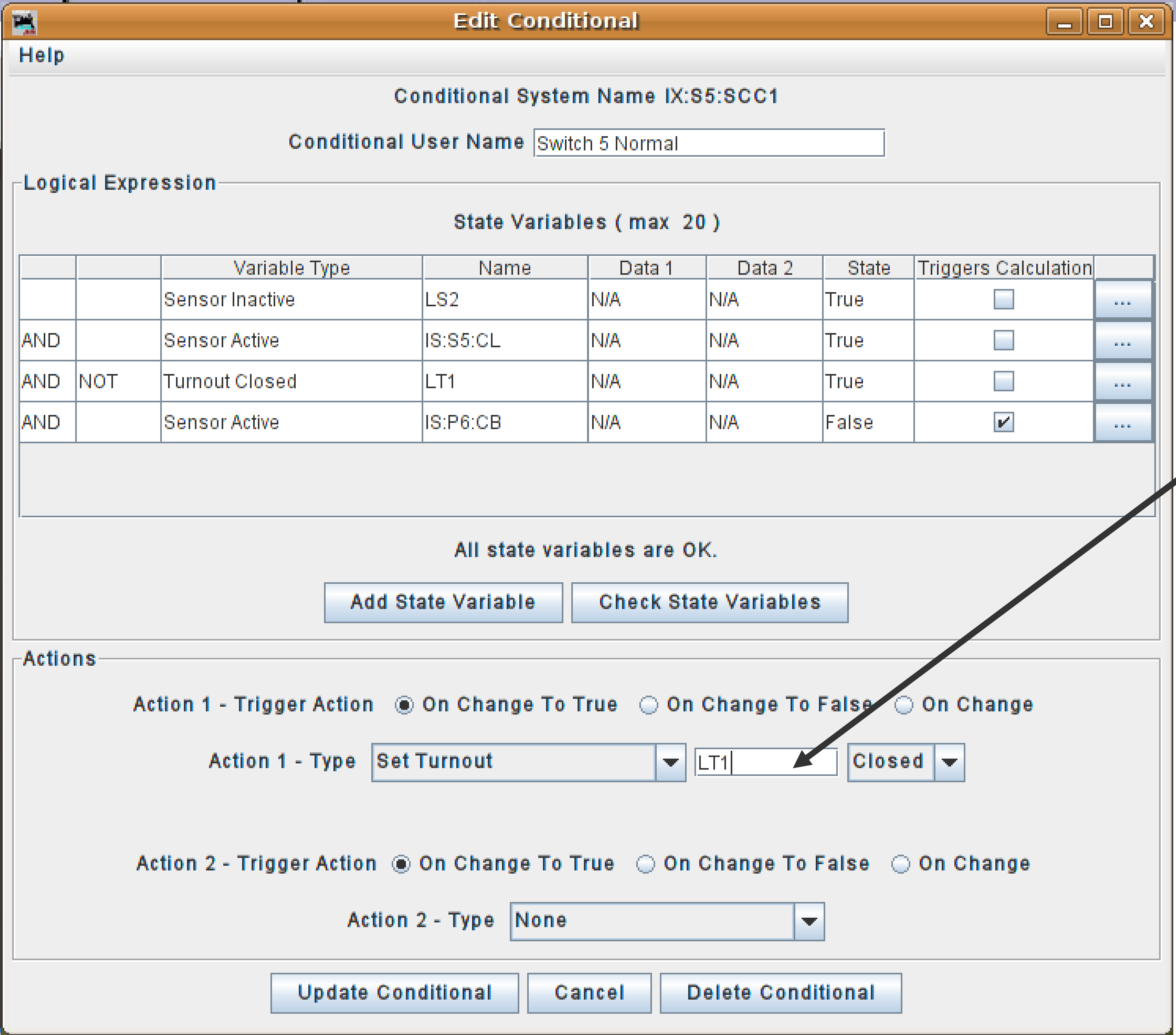
Action 2 - Type None ▾

Logix entry

- This gives us a new box and selection to enter the turnout ID and desired action.

Indirect Layout Control

Logix entry



Conditional System Name IX:S5:SCC1

Conditional User Name: Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	True	☐	...
AND		Sensor Active	IS:S5:CL	N/A	N/A	True	☐	...
AND	NOT	Turnout Closed	LT1	N/A	N/A	True	☐	...
AND		Sensor Active	IS:P6:CB	N/A	N/A	False	☑	...

All state variables are OK.

Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

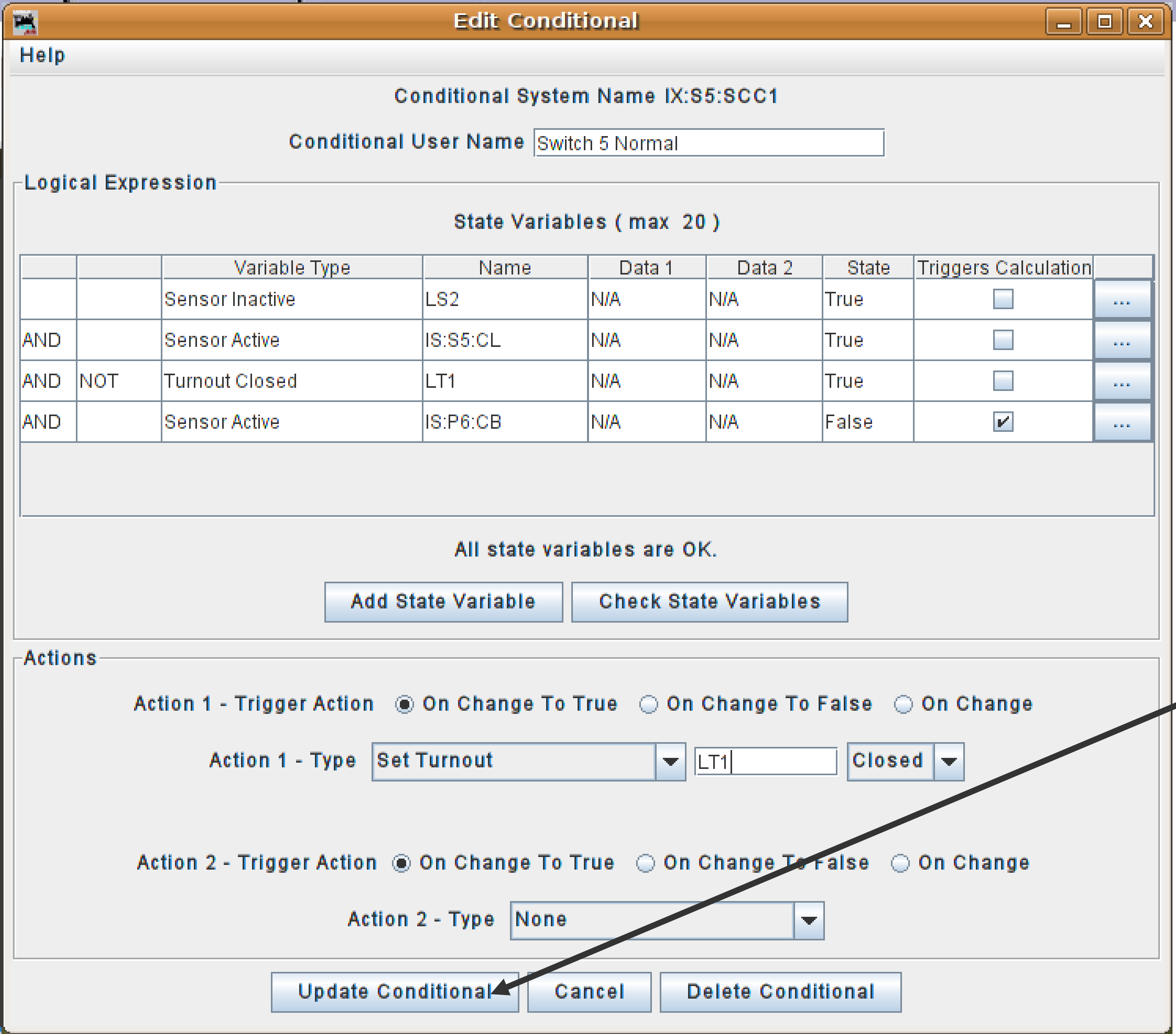
Action 1 - Type: Set Turnout, LT1, Closed

Action 2 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type: None

Logix entry

- This gives us a new box and selection to enter the turnout ID and desired action.
- Enter LT1 as the ID and the action is already defaulted to 'Closed'.



Edit Conditional

Help

Conditional System Name IX:S5:SCC1

Conditional User Name: Switch 5 Normal

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	True	□	...
AND		Sensor Active	IS:S5:CL	N/A	N/A	True	□	...
AND	NOT	Turnout Closed	LT1	N/A	N/A	True	□	...
AND		Sensor Active	IS:P6:CB	N/A	N/A	False	☑	...

All state variables are OK.

Actions

Action 1 - Trigger Action: ● On Change To True ○ On Change To False ○ On Change

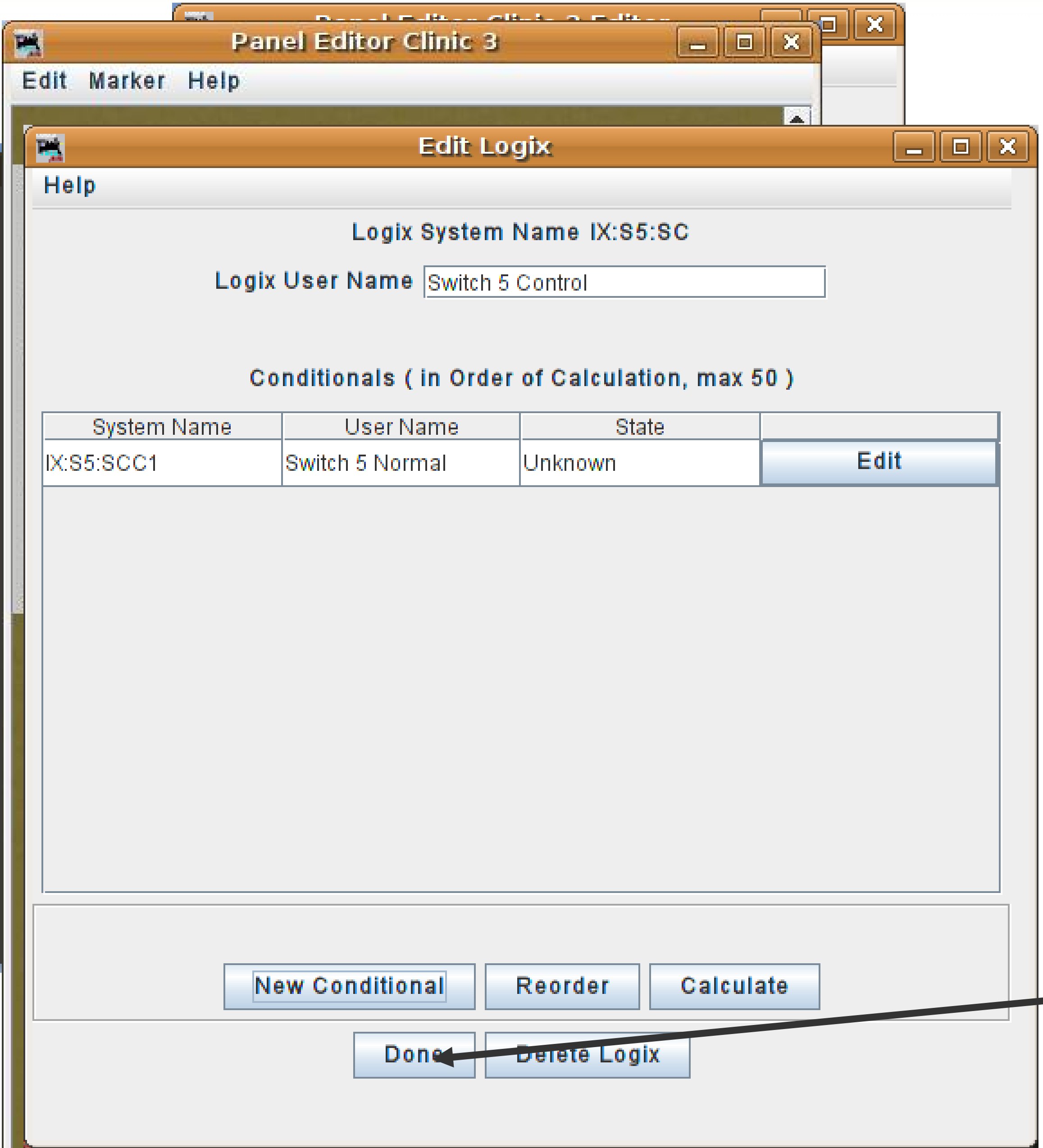
Action 1 - Type: Set Turnout; LT1; Closed

Action 2 - Trigger Action: ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type: None

Logix entry

- This gives us a new box and selection to enter the turnout ID and desired action.
- Enter LT1 as the ID and the action is already defaulted to 'Closed'.
- When you are done click on 'Update Conditional'. Unlike images, Logix will not operate until they are closed.



Logix System Name IX:S5:SC

Logix User Name Switch 5 Control

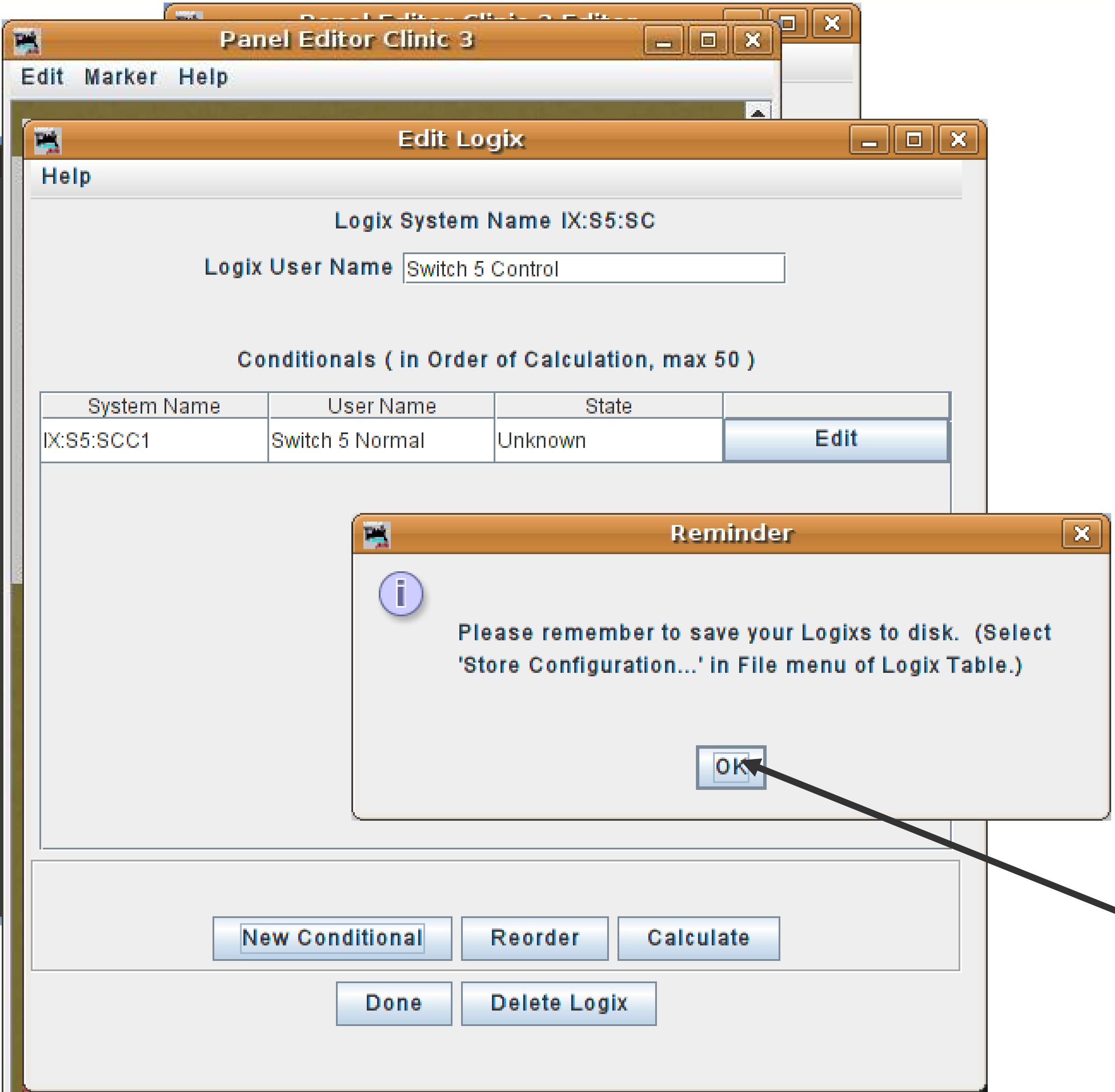
Conditionals (in Order of Calculation, max 50)

System Name	User Name	State	
IX:S5:SCC1	Switch 5 Normal	Unknown	Edit

Logix entry

- This gives us a new box and selection to enter the turnout ID and desired action.
- Enter LT1 as the ID and the action is already defaulted to 'Closed'.
- When you are done click on 'Update Conditional'. Unlike images, Logix will not operate until they are closed.
- Click 'Done' to close

Logix entry



Logix System Name IX:S5:SC

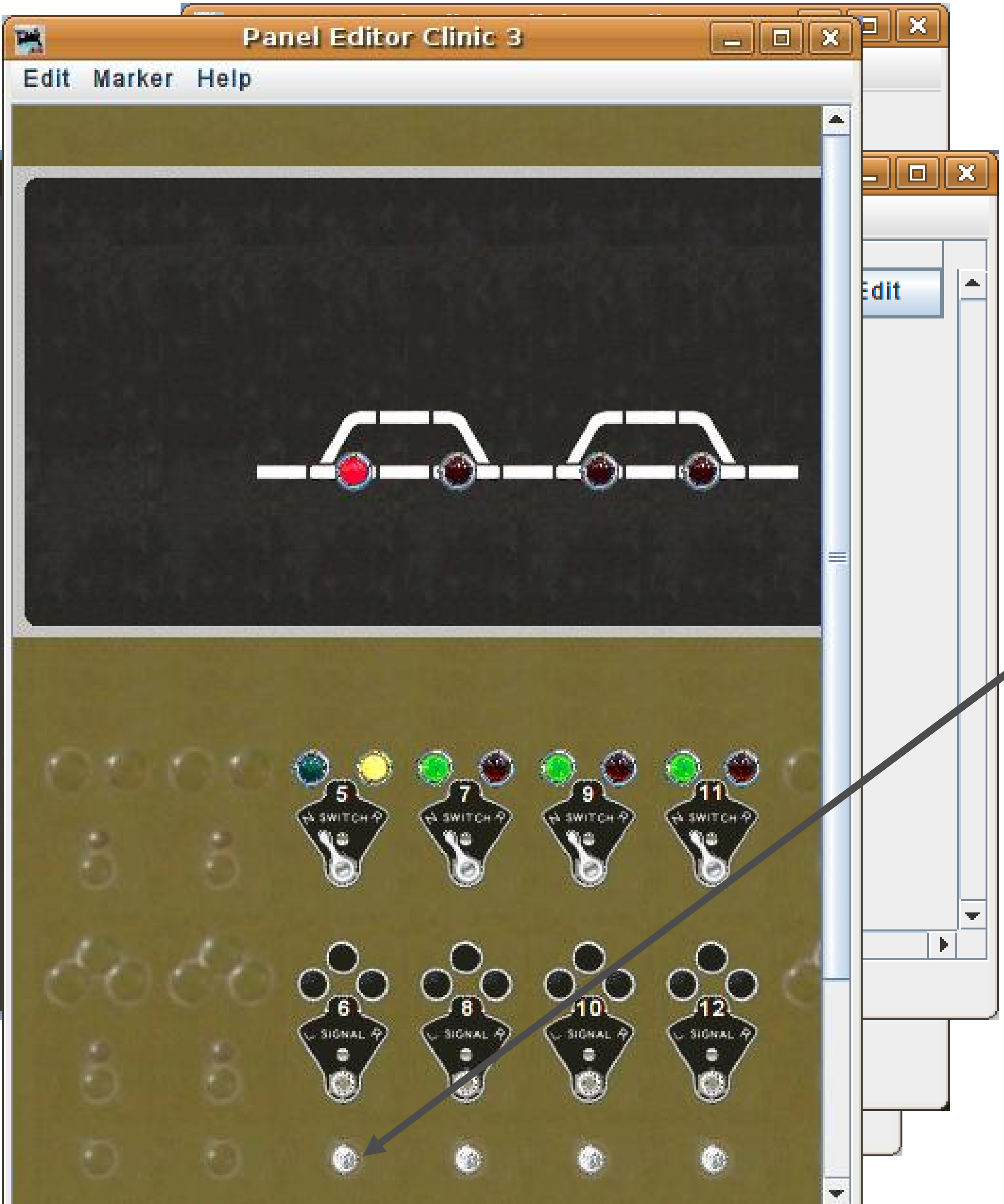
Logix User Name Switch 5 Control

Conditionals (in Order of Calculation, max 50)

System Name	User Name	State	
IX:S5:SCC1	Switch 5 Normal	Unknown	Edit

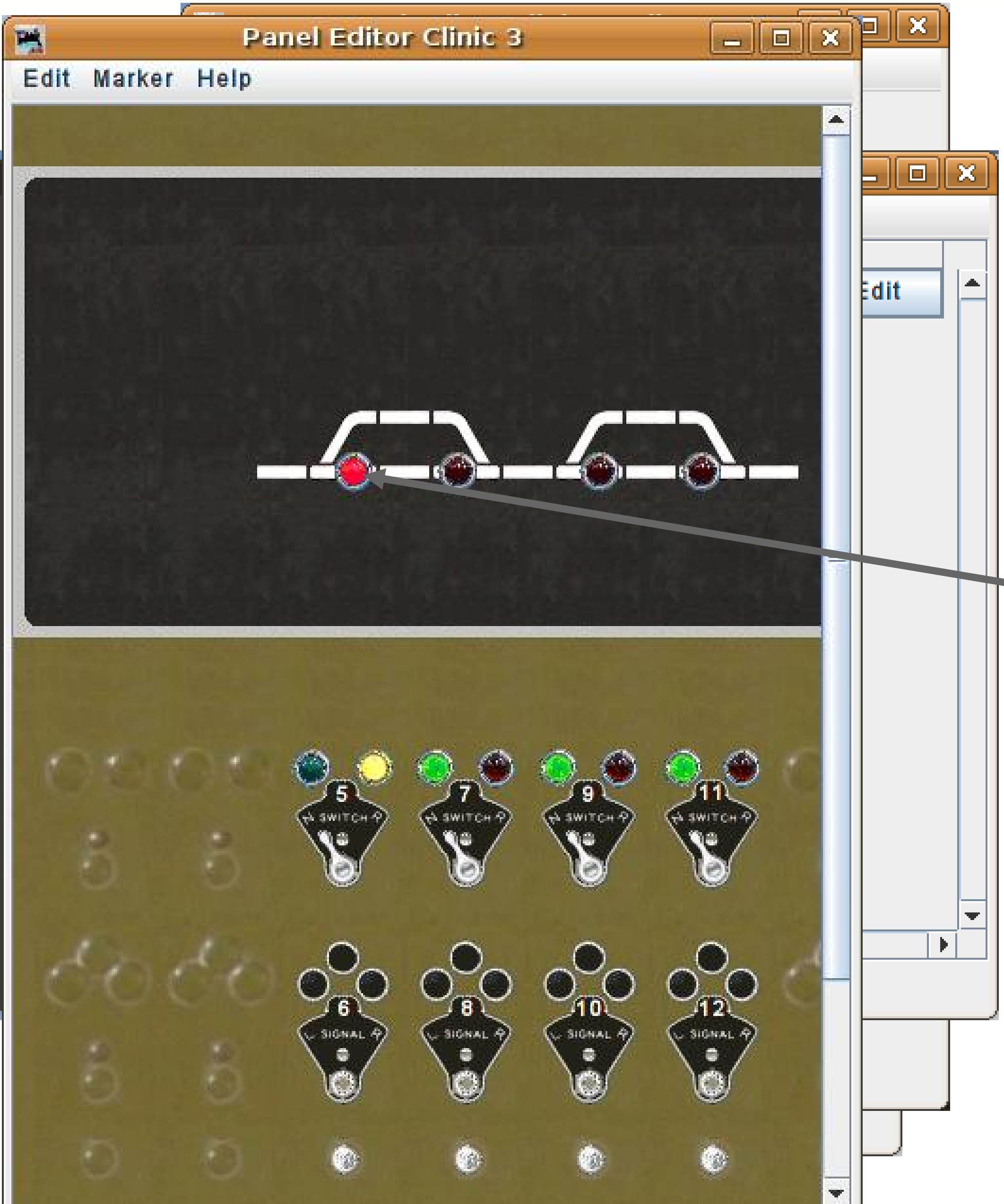
Please remember to save your Logixs to disk. (Select 'Store Configuration...' in File menu of Logix Table.)

- This gives us a new box and selection to enter the turnout ID and desired action.
- Enter LT1 as the ID and the action is already defaulted to 'Closed'.
- When you are done click on 'Update Conditional'. Unlike images, Logix will not operate until they are closed.
- Click 'Done' to close
- Saved Panels will include their Logix.



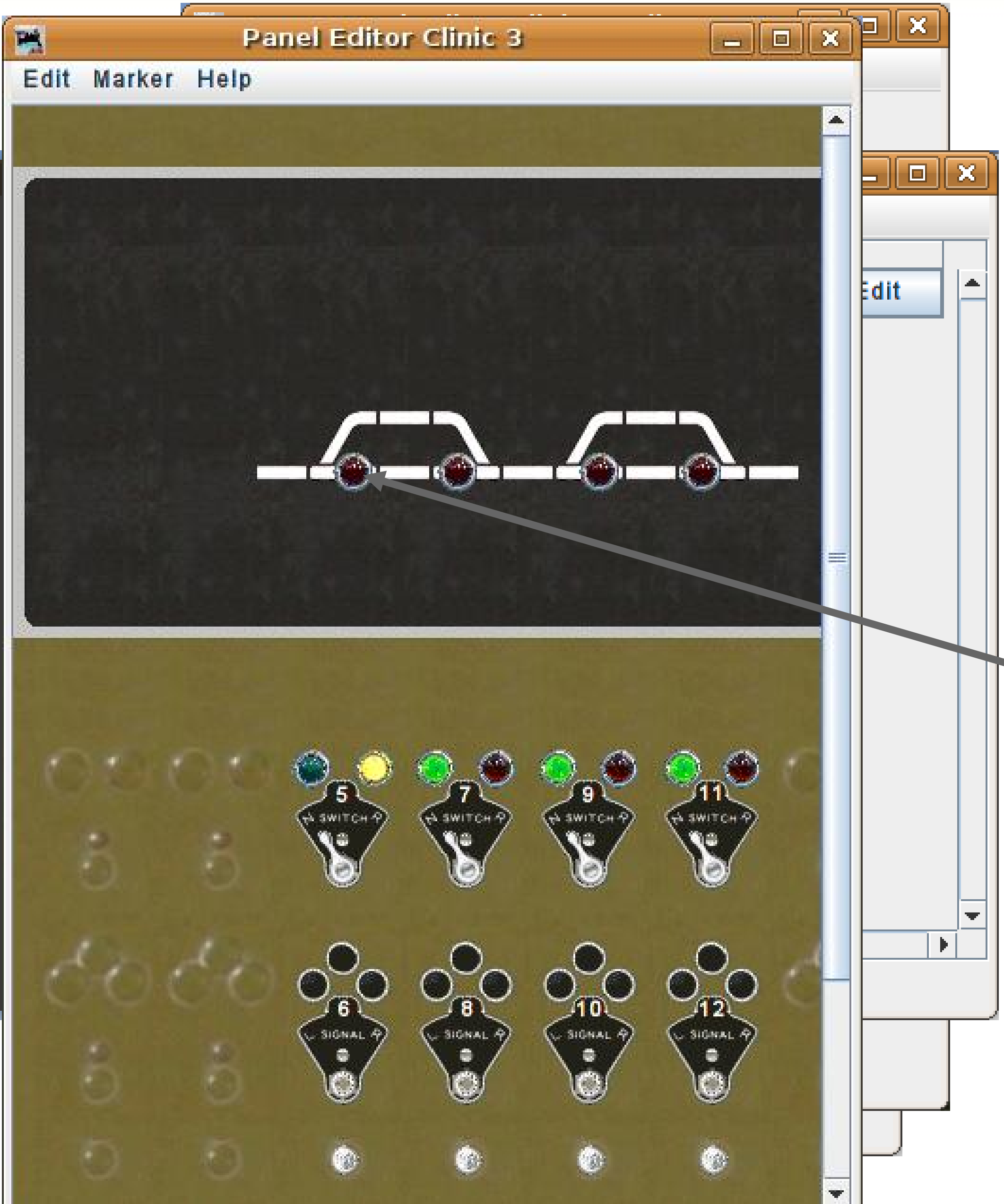
Logix entry

- Clicking on our Code Button should now throw our turnout to 'N', but only if the OS is clear.



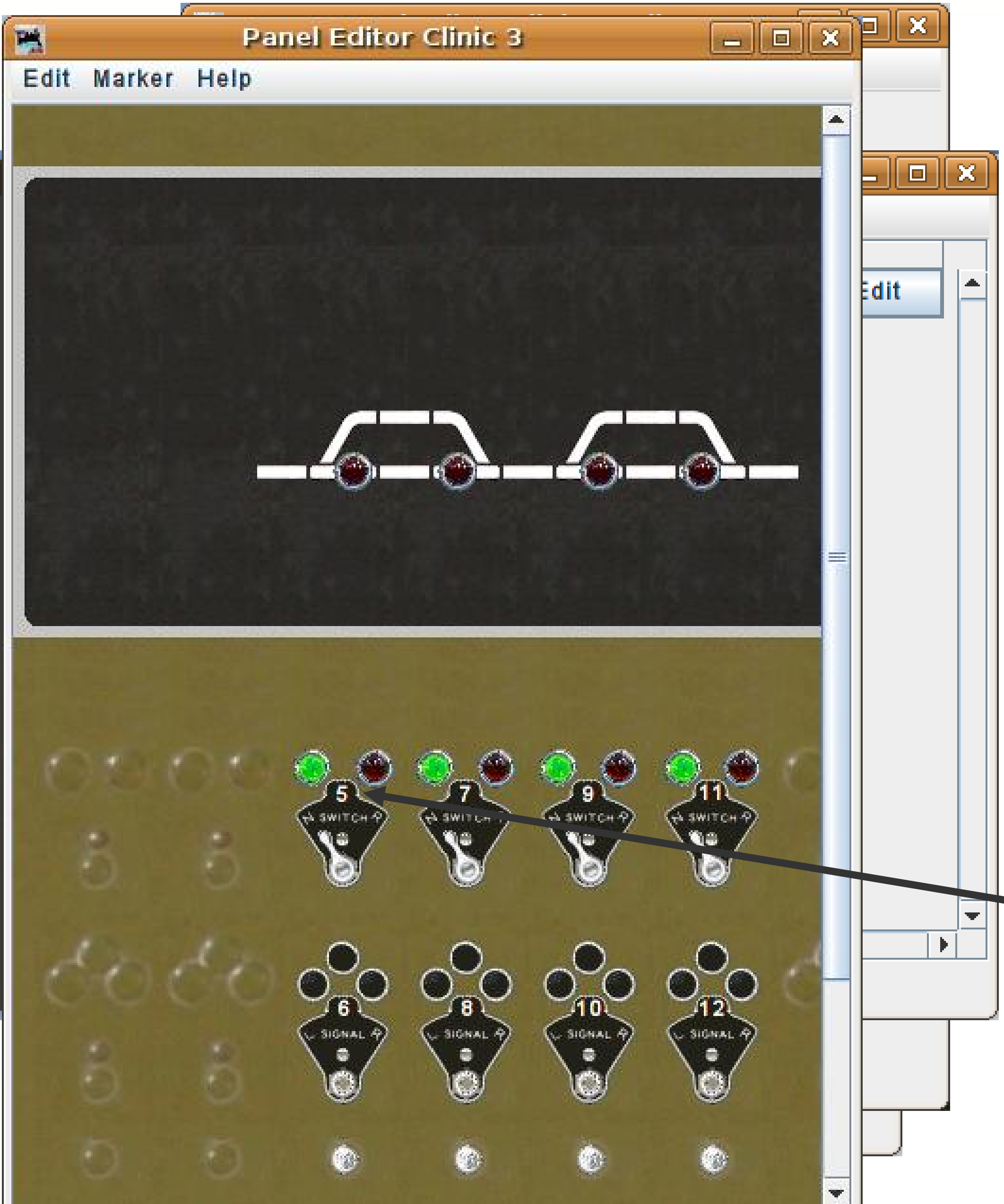
Logix entry

- Clicking on our Code Button should now throw our turnout to 'N', but only if the OS is clear.
- The first test only shows the button moving because the OS is still occupied.



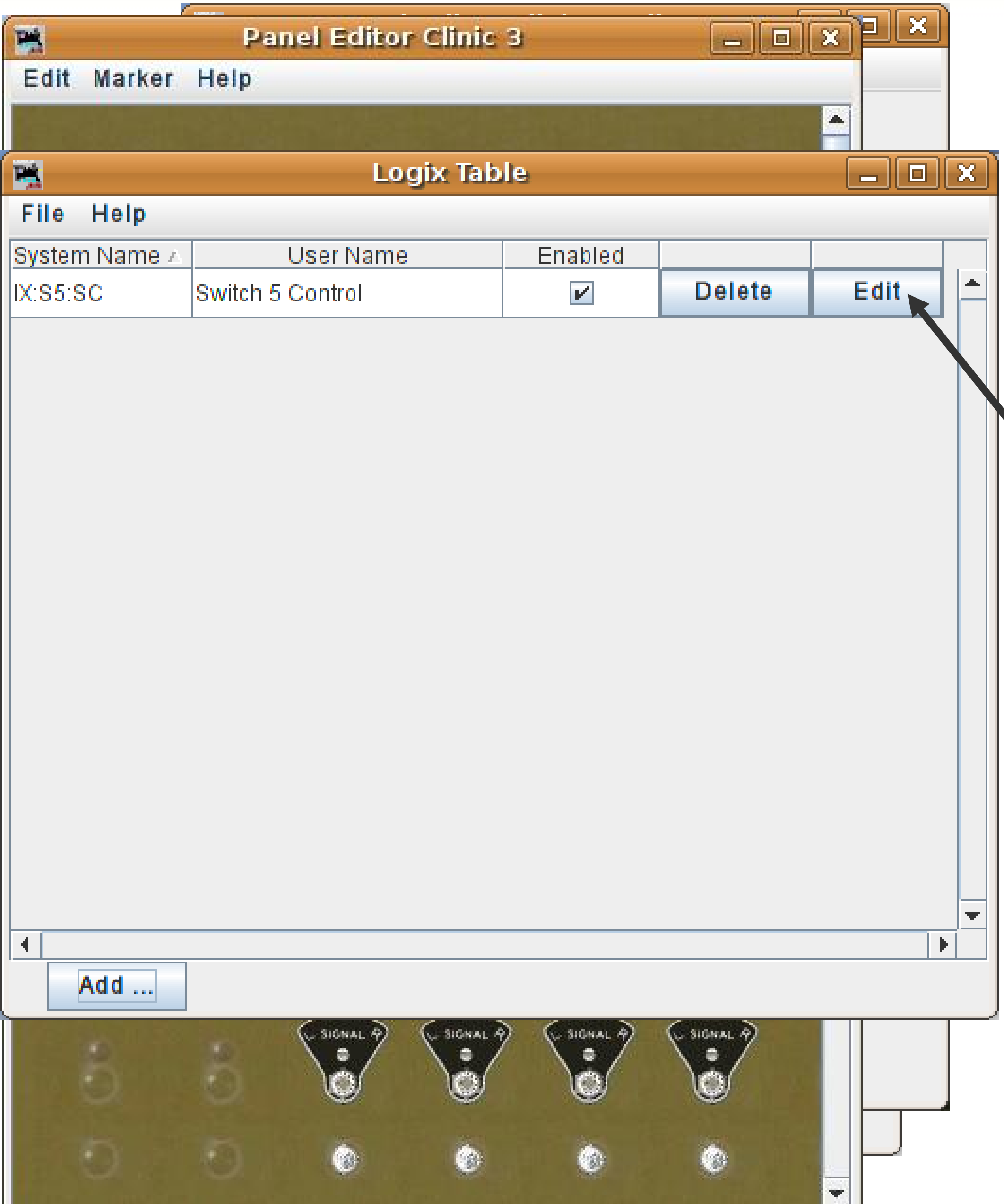
Logix entry

- Clicking on our Code Button should now throw our turnout to 'N', but only if the OS is clear.
- The first test only shows the button moving because the OS is still occupied.
- Try it again after moving the train.



Logix entry

- Clicking on our Code Button should now throw our turnout to 'N', but only if the OS is clear.
- The first test only shows the button moving because the OS is still occupied.
- Try it again after moving the train.
- This time it worked as it should.



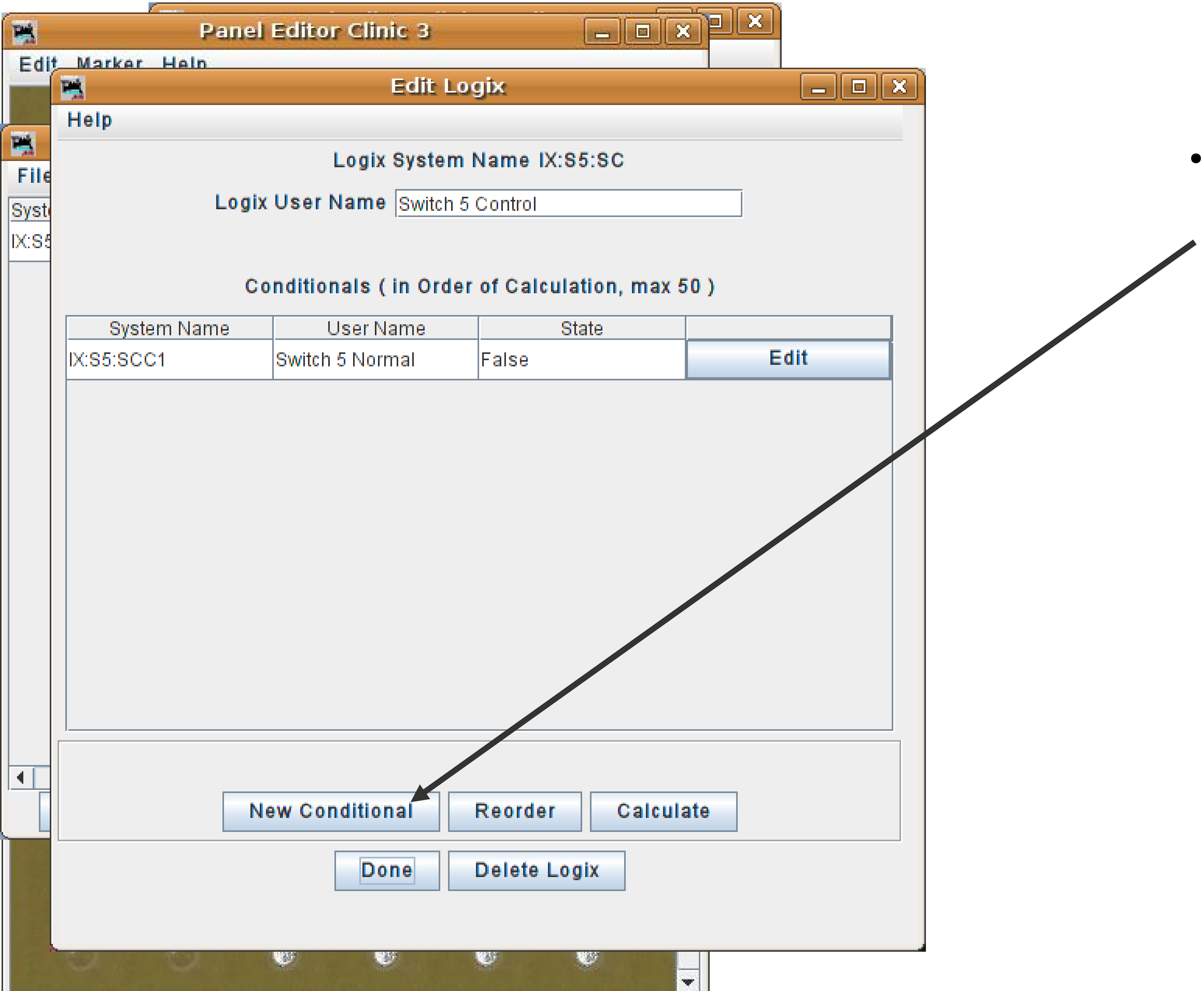
System Name	User Name	Enabled		
IX:S5:SC	Switch 5 Control	☑	Delete	Edit

Logix entry

- Clicking on our Code Button should now throw our turnout to 'N', but only if the OS is clear.
- The first test only shows the button moving because the OS is still occupied.
- Try it again after moving the train.
- This time it worked as it should.
- Now click on the 'Edit' button to open up our Logix again.

Logix entry

- Add a conditional and name it 'Switch 5 Reverse'



Panel Editor Clinic 3

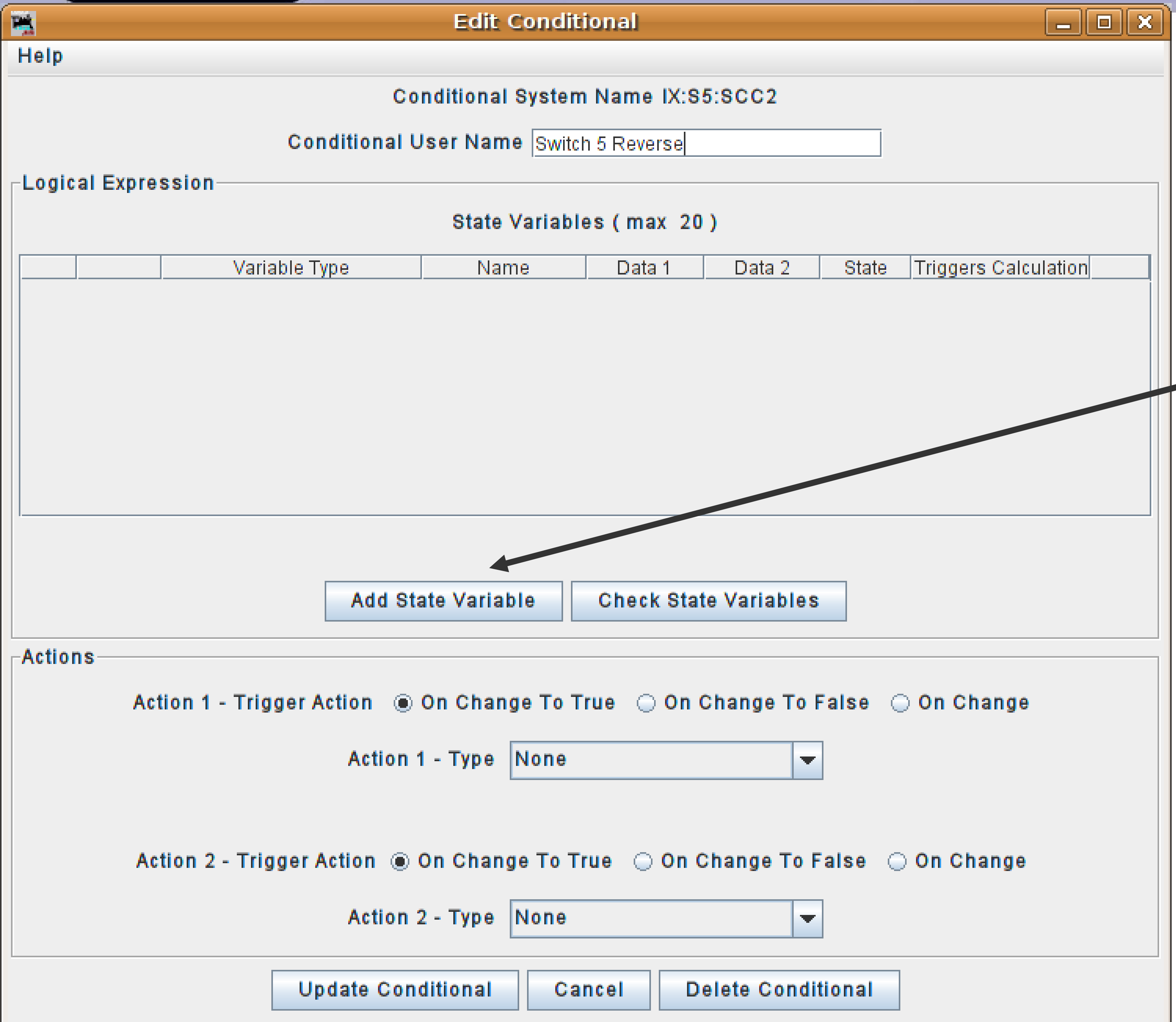
Edit Logix

Logix System Name IX:S5:SC

Logix User Name: Switch 5 Control

Conditionals (in Order of Calculation, max 50)

System Name	User Name	State	
IX:S5:SCC1	Switch 5 Normal	False	Edit



Edit Conditional

Conditional System Name IX:S5:SCC2

Conditional User Name: Switch 5 Reverse

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation

Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

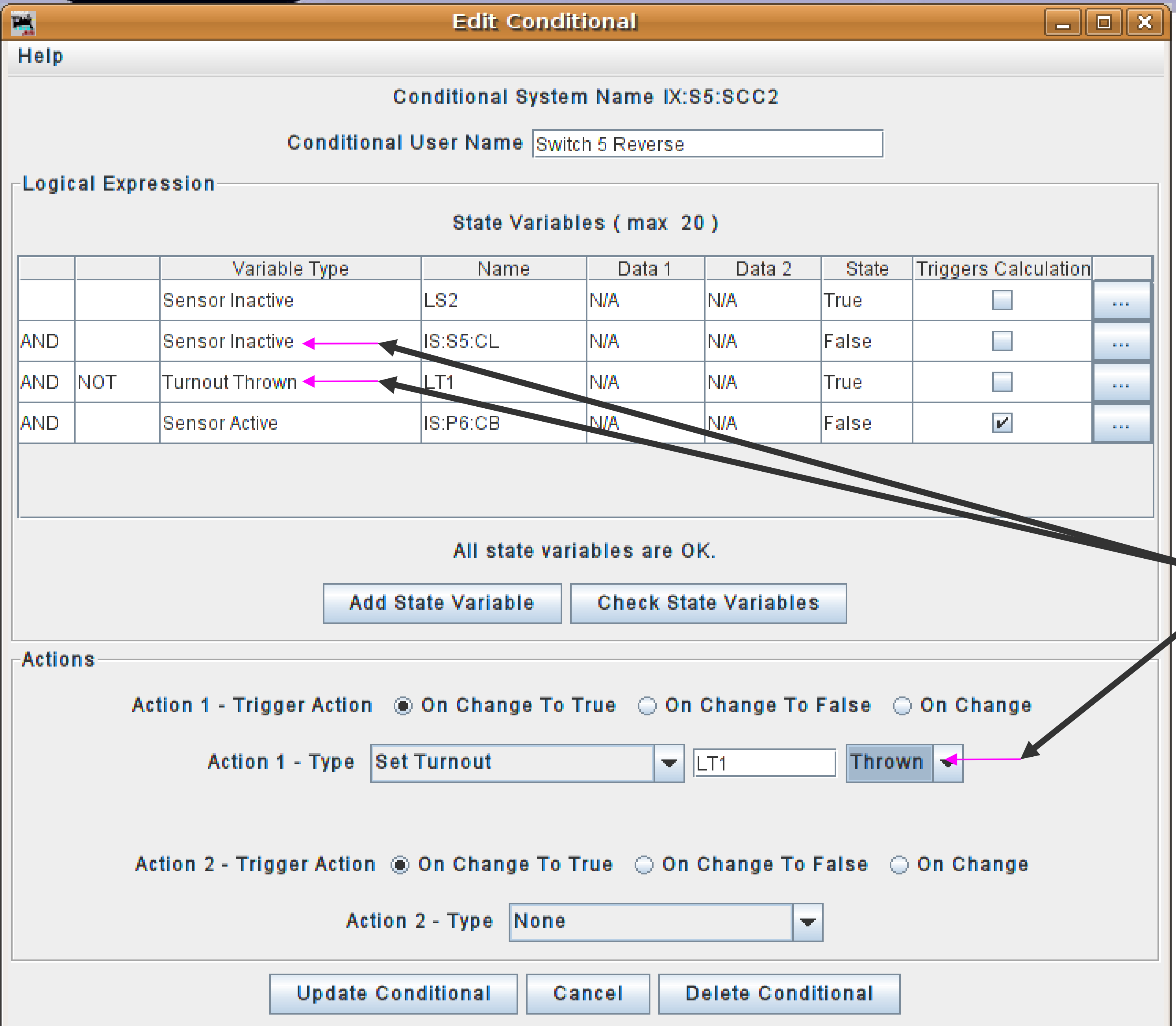
Action 1 - Type: None

Action 2 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type: None

Logix entry

- Add a conditional and name it 'Switch 5 Reverse'
- The Variables will be similar to those we used for switch 'Normal', except for the different modes.



Edit Conditional

Conditional System Name IX:S5:SCC2

Conditional User Name: Switch 5 Reverse

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	True	□	...
AND		Sensor Inactive	IS:S5:CL	N/A	N/A	False	□	...
AND	NOT	Turnout Thrown	LT1	N/A	N/A	True	□	...
AND		Sensor Active	IS:P6:CB	N/A	N/A	False	☑	...

All state variables are OK.

Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 1 - Type: Set Turnout | LT1 | Thrown

Action 2 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

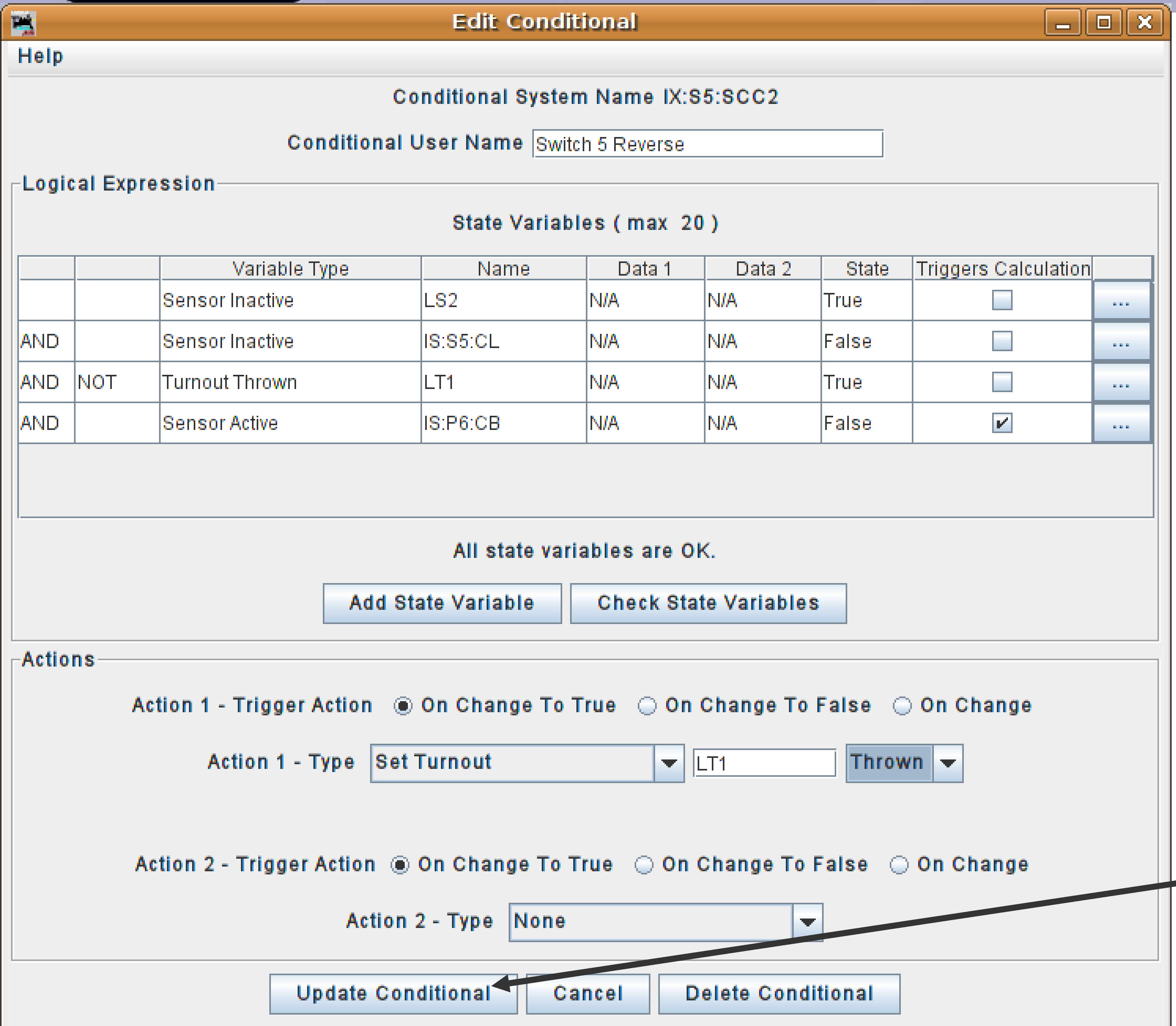
Action 2 - Type: None

Logix entry

- Add a conditional and name it 'Switch 5 Reverse'
- The Variables will be similar to those we used for switch 'Normal', except for the different modes.
- Note these three differences due to the inverted logic required to move the turnout in the opposite direction.

Indirect Layout Control

Logix entry



Conditional System Name IX:S5:SCC2

Conditional User Name: Switch 5 Reverse

Logical Expression

State Variables (max 20)

		Variable Type	Name	Data 1	Data 2	State	Triggers Calculation	
		Sensor Inactive	LS2	N/A	N/A	True	□	...
AND		Sensor Inactive	IS:S5:CL	N/A	N/A	False	□	...
AND	NOT	Turnout Thrown	LT1	N/A	N/A	True	□	...
AND		Sensor Active	IS:P6:CB	N/A	N/A	False	☑	...

All state variables are OK.

Actions

Action 1 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 1 - Type: Set Turnout, LT1, Thrown

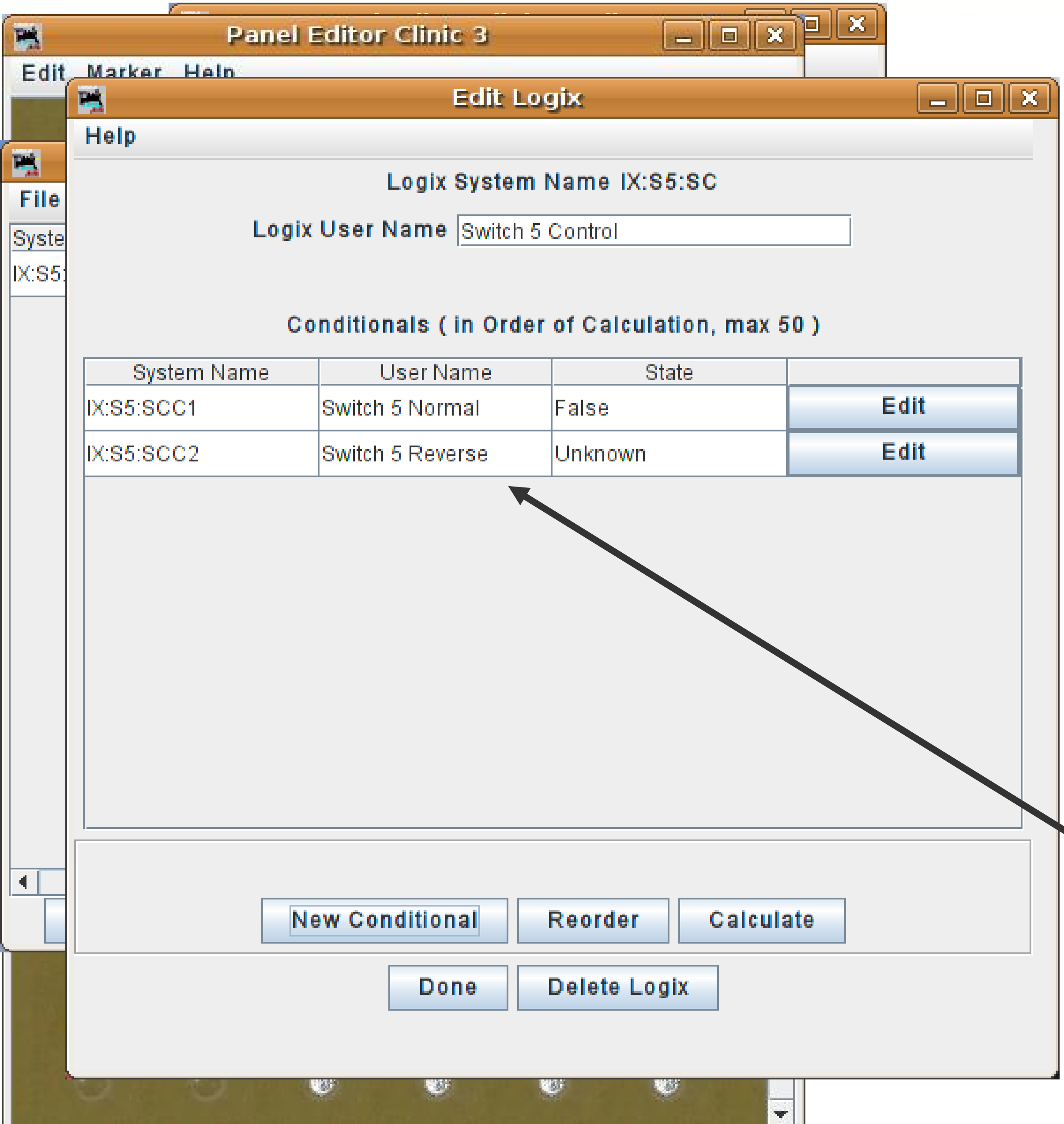
Action 2 - Trigger Action ● On Change To True ○ On Change To False ○ On Change

Action 2 - Type: None

Logix entry

- Add a conditional and name it 'Switch 5 Reverse'
- The Variables will be similar to those we used for switch 'Normal', except for the different modes.
- Note these three differences due to the inverted logic required to move the turnout in the opposite direction.
- Update the entry.

Logix entry



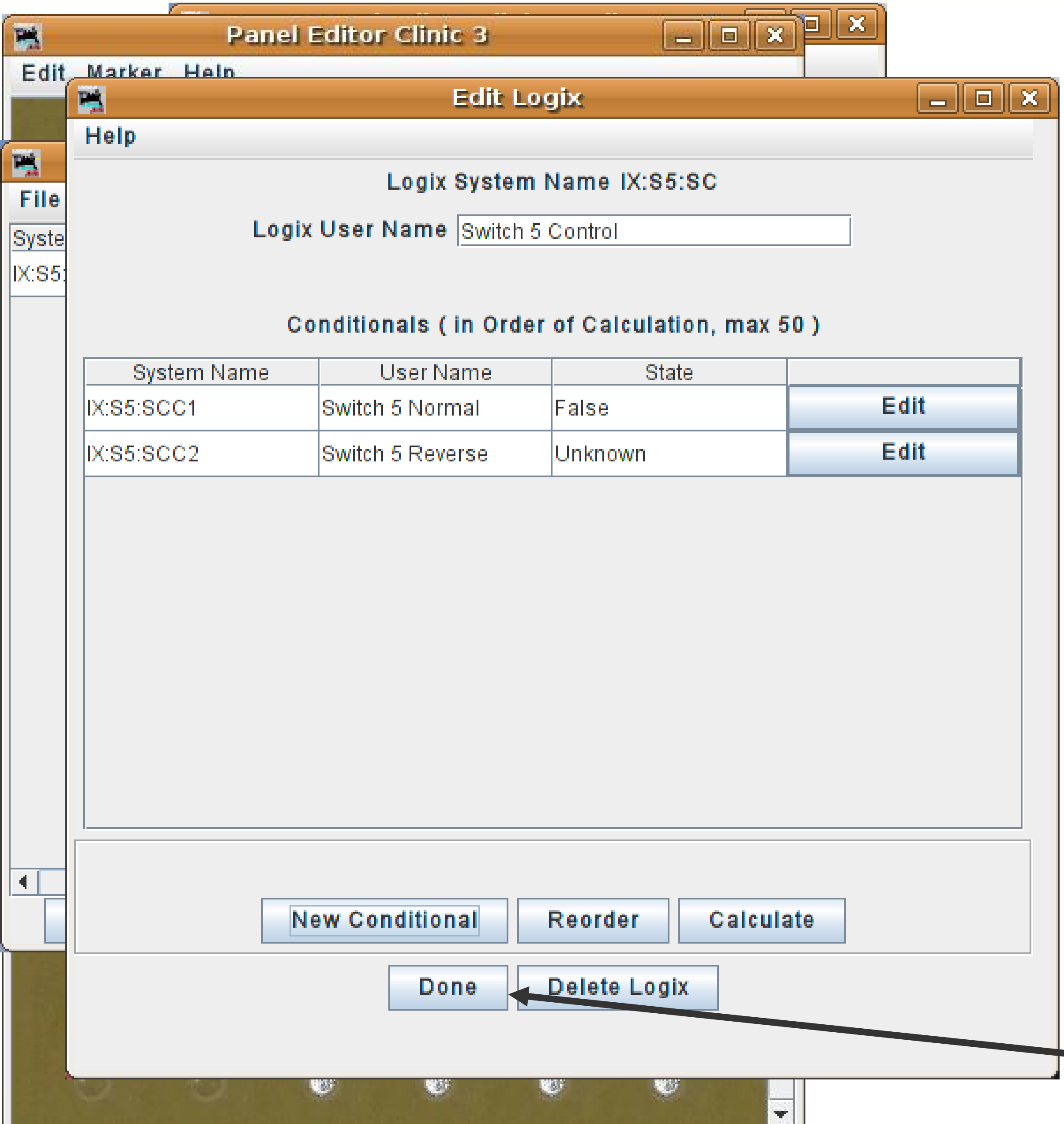
Logix System Name IX:S5:SC

Logix User Name: Switch 5 Control

Conditionals (in Order of Calculation, max 50)

System Name	User Name	State	
IX:S5:SCC1	Switch 5 Normal	False	Edit
IX:S5:SCC2	Switch 5 Reverse	Unknown	Edit

- Add a conditional and name it 'Switch 5 Reverse'
- The Variables will be similar to those we used for switch 'Normal', except for the different modes.
- Note these three differences due to the inverted logic required to move the turnout in the opposite direction.
- Update the entry.
- New item shown.



Logix System Name IX:S5:SC

Logix User Name Switch 5 Control

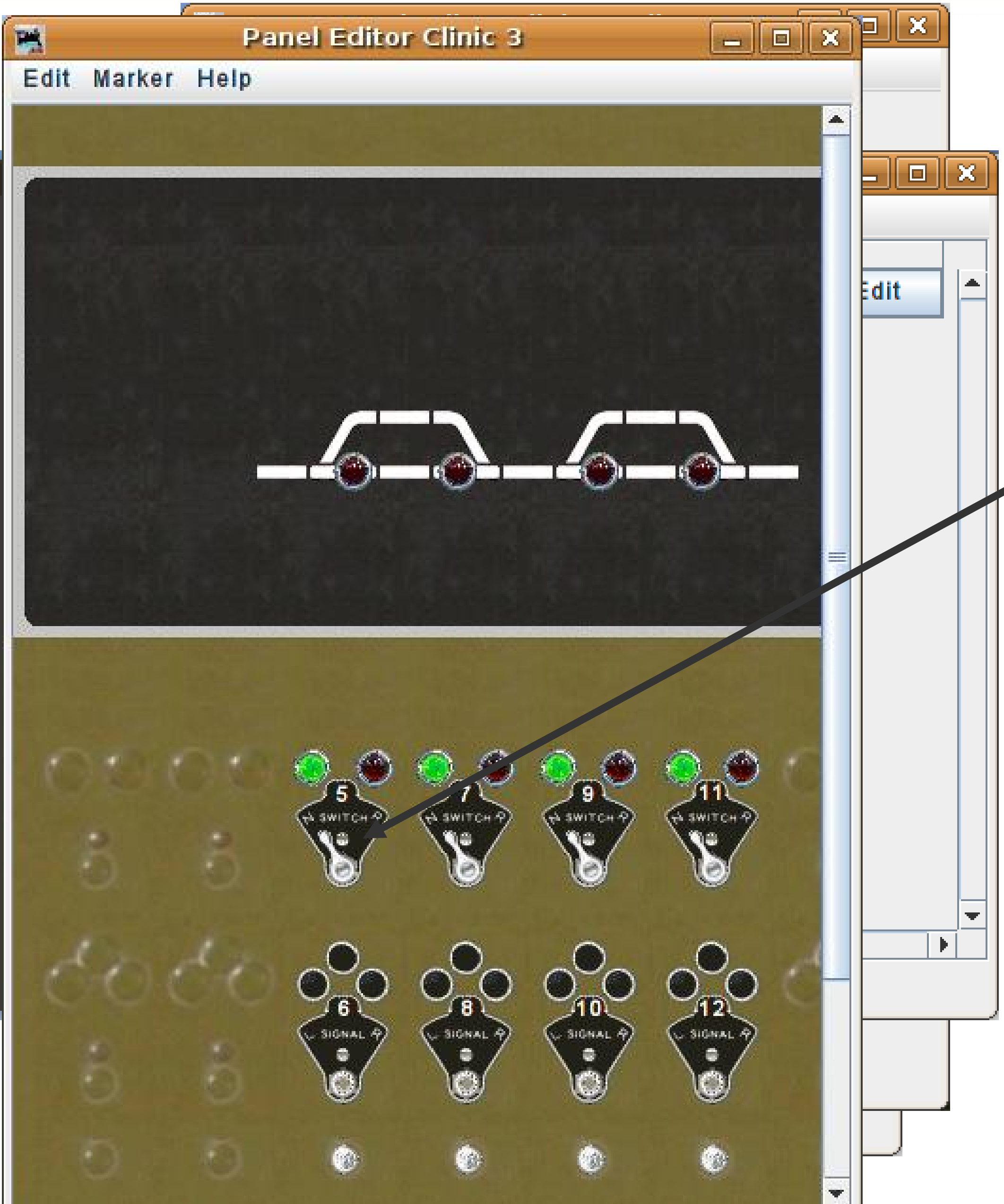
Conditionals (in Order of Calculation, max 50)

System Name	User Name	State	
IX:S5:SCC1	Switch 5 Normal	False	Edit
IX:S5:SCC2	Switch 5 Reverse	Unknown	Edit

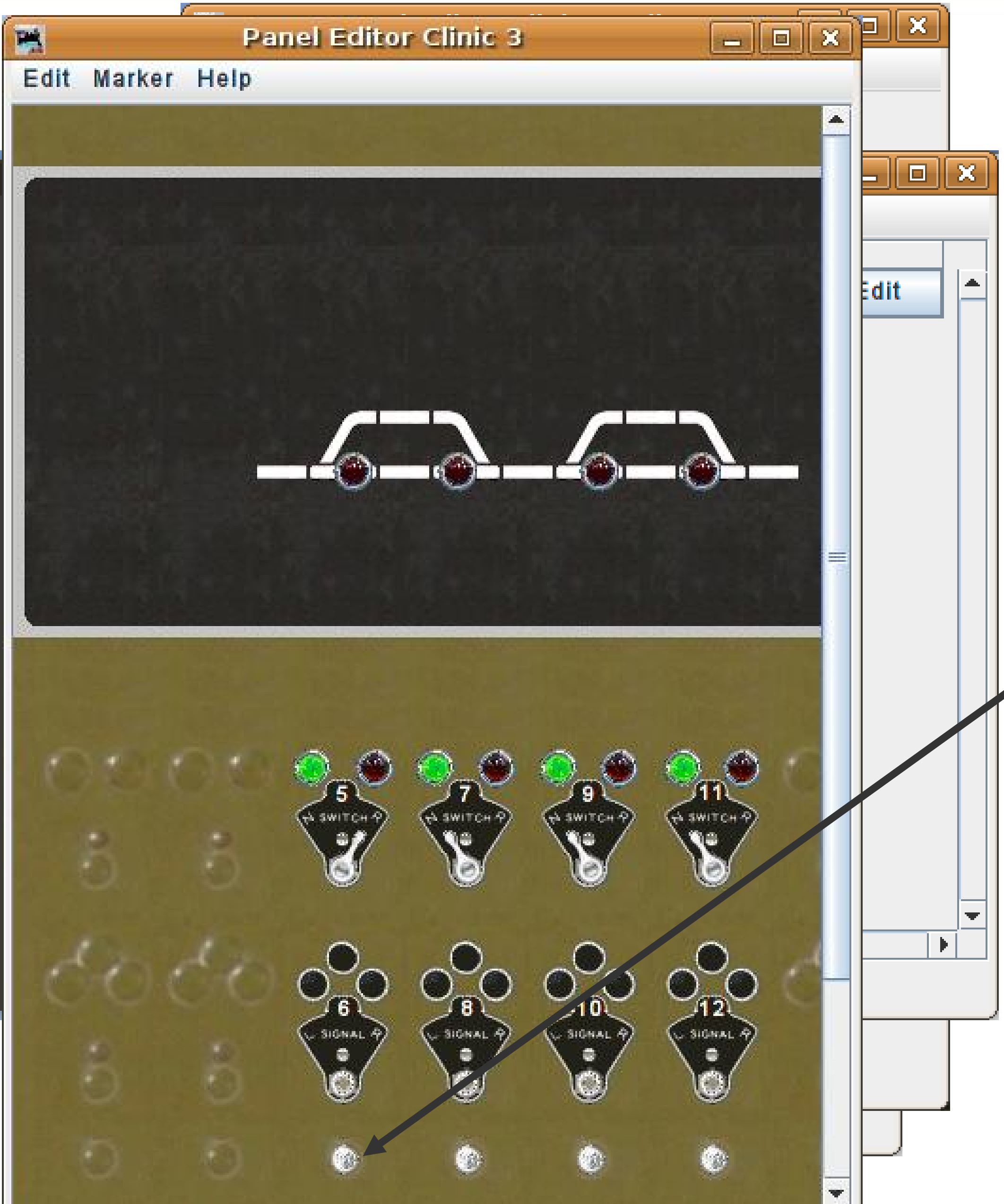
Logix entry

- Add a conditional and name it 'Switch 5 Reverse'
- The Variables will be similar to those we used for switch 'Normal', except for the different modes.
- Note these three differences due to the inverted logic required to move the turnout in the opposite direction.
- Update the entry.
- New item shown.
- Click Done.

Logix entry

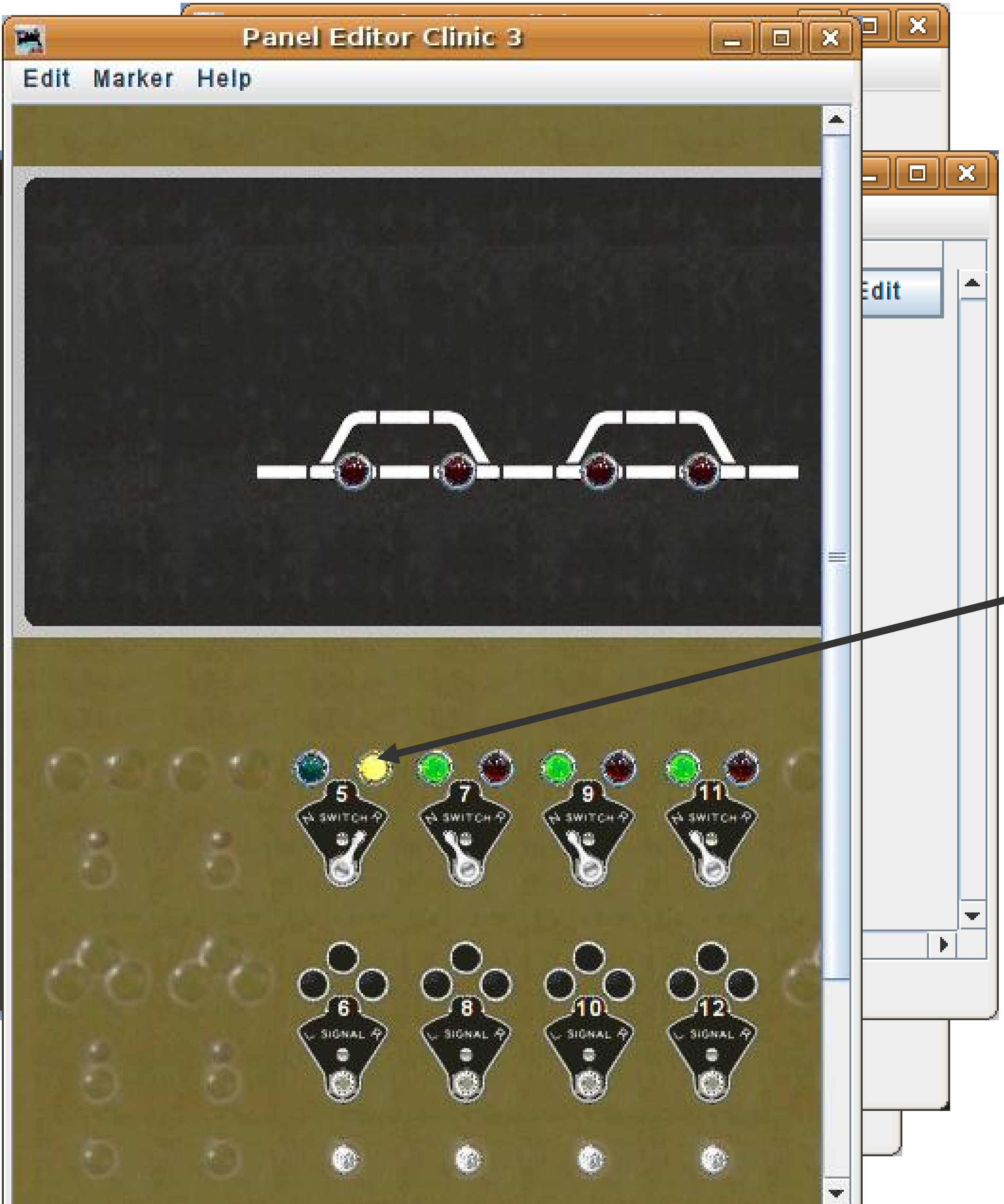


- Now change the lever.



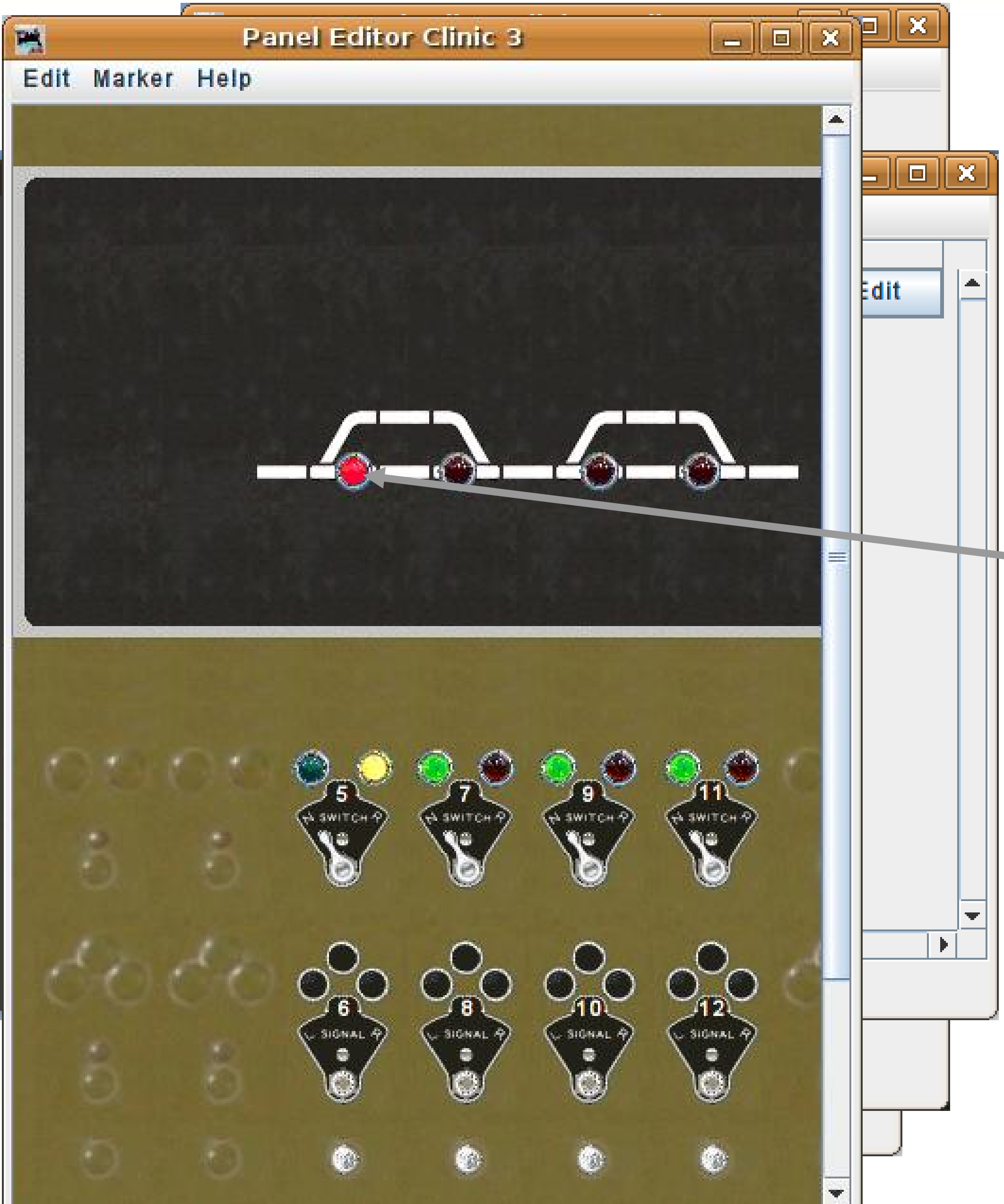
Logix entry

- Now change the lever.
- Push the Code Button.



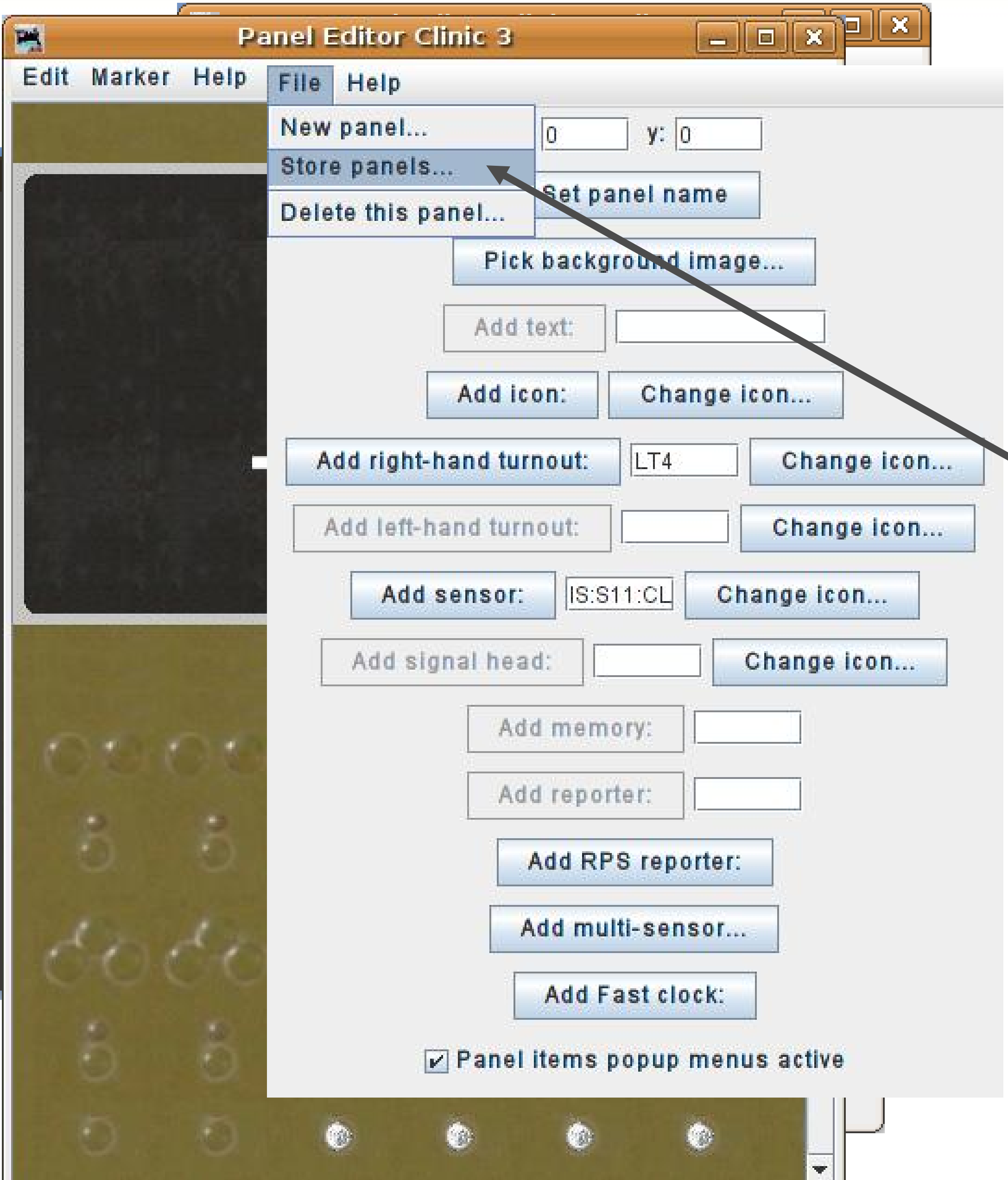
Logix entry

- Now change the lever.
- Push the Code Button.
- And Our turnout changes.



Logix entry

- Now change the lever.
- Push the Code Button.
- And Our turnout changes.
- But not if the OS shows occupied.



x: 0 y: 0

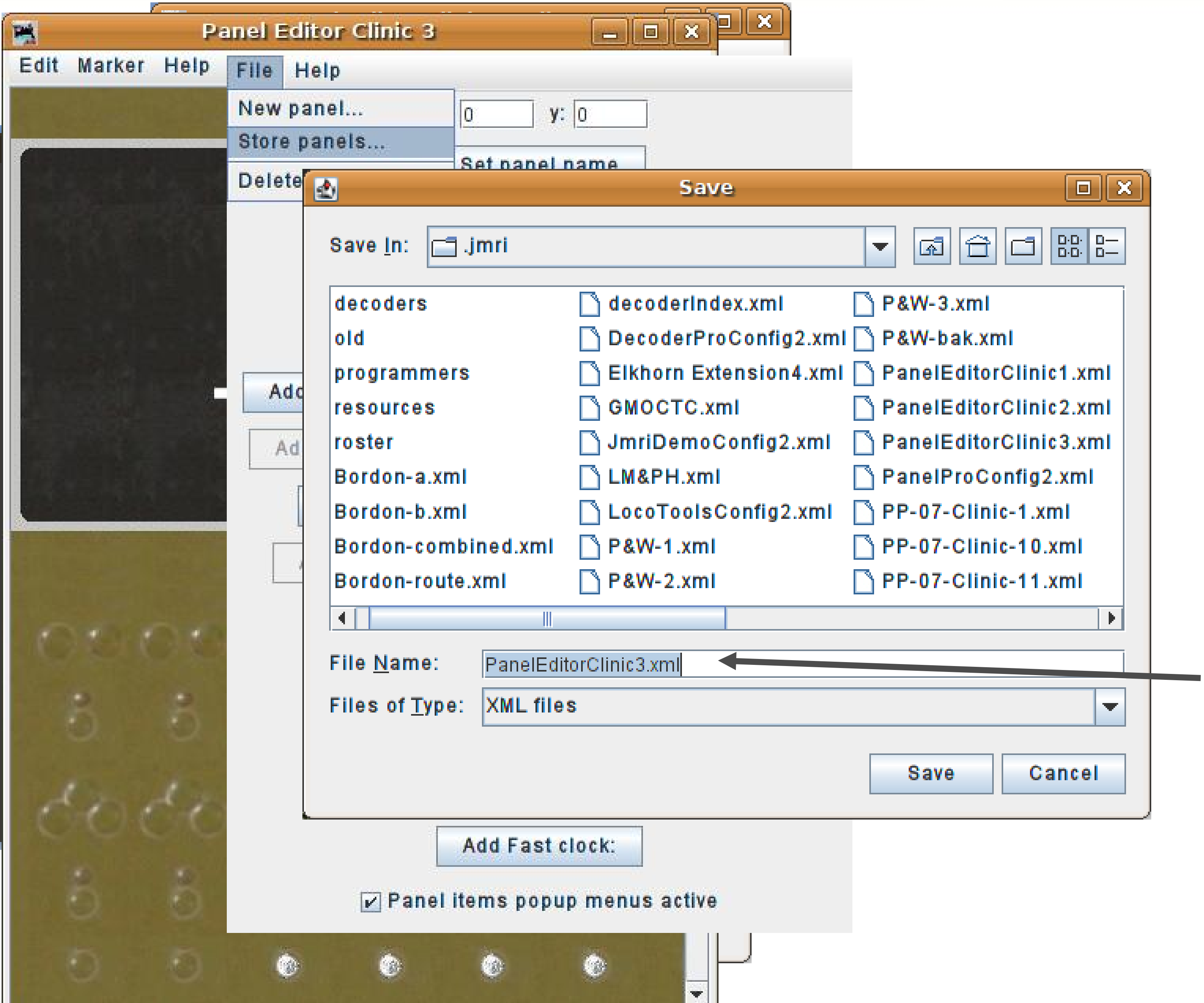
Add right-hand turnout: LT4

Add sensor: IS:S11:CL

Logix entry

- Now change the lever.
- Push the Code Button.
- And Our turnout changes.
- But not if the OS shows occupied.
- Save our work.

Logix entry



Save In: .jmri

```
decoders                 decoderIndex.xml           P&W-3.xml
old                      DecoderProConfig2.xml      P&W-bak.xml
programmers              Elkhorn Extension4.xml     PanelEditorClinic1.xml
resources                GMOCTC.xml                 PanelEditorClinic2.xml
roster                   JmriDemoConfig2.xml        PanelEditorClinic3.xml
Bordon-a.xml             LM&PH.xml                  PanelProConfig2.xml
Bordon-b.xml             LocoToolsConfig2.xml       PP-07-Clinic-1.xml
Bordon-combined.xml      P&W-1.xml                  PP-07-Clinic-10.xml
Bordon-route.xml         P&W-2.xml                  PP-07-Clinic-11.xml
```

File Name: PanelEditorClinic3.xml

Files of Type: XML files

- Now change the lever.
- Push the Code Button.
- And Our turnout changes.
- But not if the OS shows occupied.
- Save our work.
- As Clinic-3.xml

This completes Clinic 3. In the next session we will talk about how we may manually edit the Panel.xml file to easily duplicate our Switch 5 Logix to extend it to Switches 7, 9, and 11.

The remainder of the next clinic 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.