NEWSLETTER

Vol. 37 – No. 3
May - Jume 2025

Current Central Station 3 Version – 2.5.2 (6)
Current Central Station 2 Version – 4.3.0 (11)
Current Mobile Station Version – 4.15
Current Wireless MS Version – 4.27.0.13

Digital Consultants
Rick Sinclair
Curtis Jeung

We just got back from the NMRA National Train Show in Novi Michigan where the ETE Great Lakes Chapter won first place at the show for their module layout. Curtis and I have contributed countless hours on that module, and it feels great for us to share this recognition and accomplishment.

Our first article deals with the newly released 30601 D&RGW F-7 A locomotive while in the second article, the conversion of a complex layout from a CS2 to a CS3 is nearing completion.

New Locomotive Received

I do have to admit that I like Märklin US prototype items, and I try to get as many as I can. I was really excited to receive the 30601 D&RGW locomotive. For some reason this has been one of my favorite paint schemes (Fig. 1). I love the retro box and the fact that it has a plastic sleeve, it should be easy to keep the box looking good.

Fig. 1 – Marklin 30601

I opened the box to look at the locomotive and it did not disappoint. The paint is nice and sharp, and it differs from the original with four bars of pinstriping and the silver roof.

Naturally the first thing I did was to put it on the track, test the motor, and then the lights. The headlights are nice and bright. I noticed that this only had a headlight function, and I knew it had no sound. I believe the reason behind this is because the original Rio Grande locomotive (3062) was analog and only had headlights since it was produced from 1976-1984, and being a re-release, in a way, Märklin chose to make it the same as the original.

I took the body off and noticed that there was a modern 60972 decoder and LED circuit board for the headlight. What struck me as odd was that there were LEDs on the circuit board for both the upper and lower headlights and number board lights (Fig. 2), and not only that, they were wired also to the decoder mounting board (Fig. 3). This gave me the idea that maybe they could be powered by the decoder.

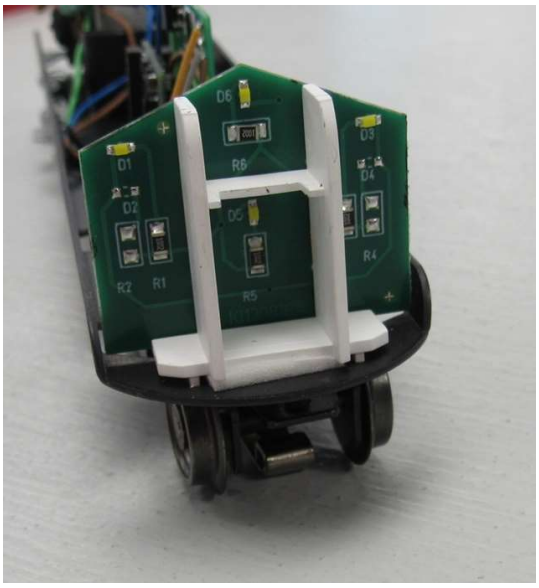

Fig. 2 – LED Light board

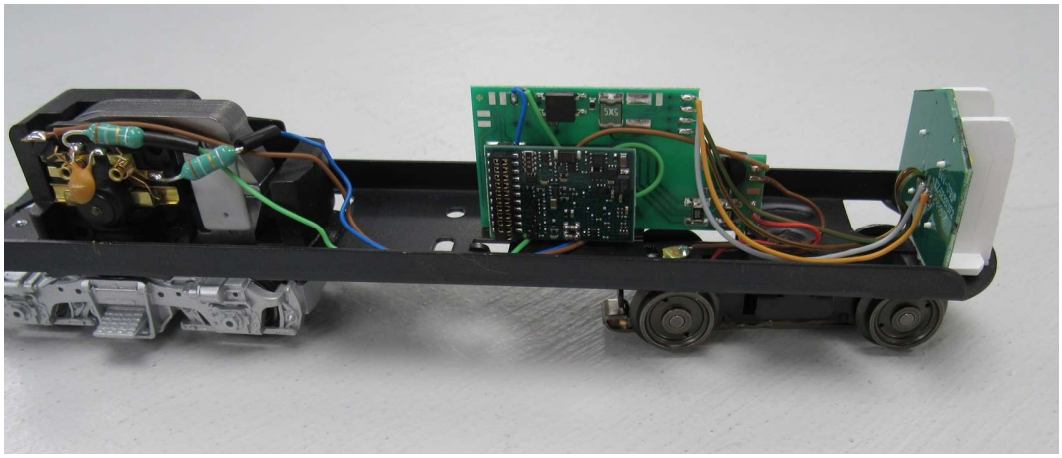
Fig. 3 – LED Circuit board wired to decoder mounting board

I quickly confirmed this with an Ohm meter that the contact was wired all the way to the decoder, and with a little bit of function mapping, I was able to figure out how to control them.

Re-Mapping

I just needed to figure out which Auxiliary output controlled the lights. After a little investigation, it turns out the Mars light is “Front Light”, the number board lights are Aux 2, and the headlight is Aux 3.

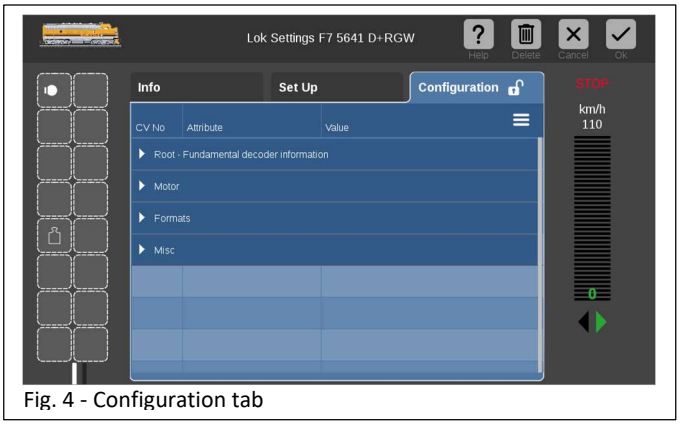

Fig. 4 - Configuration tab

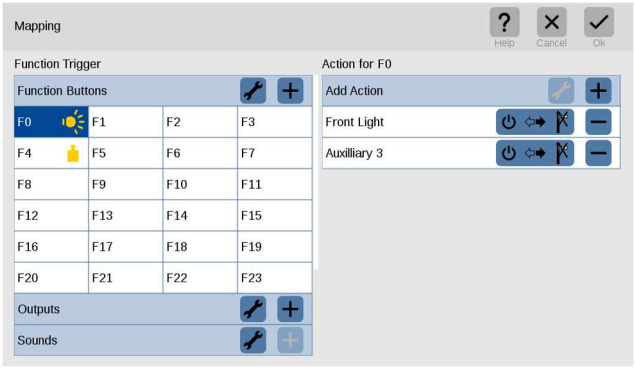

Fig. 5 - Function Mapping Screen

The only thing left to do is a little bit of re-mapping and icon assignment to make all the lights controllable and configurable.

What needs to be done is to get into Edit Mode of the locomotive. Then touch the “Configuration” tab. Once there, the decoder CV’s are read (Fig. 4). After the CS3 is finished reading the CV’s, touch any function icon to get into the function mapping (Fig. 5).

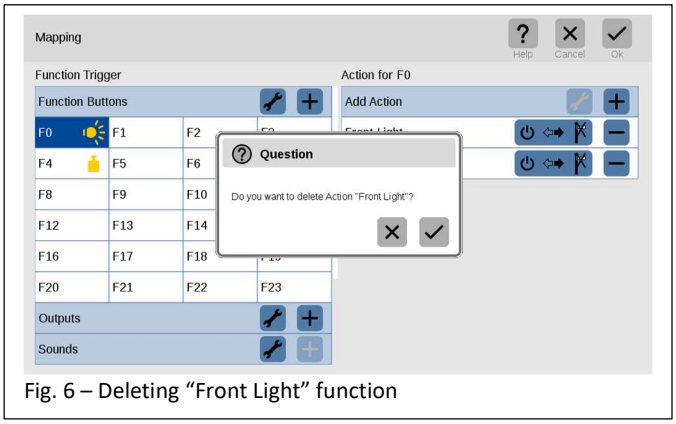

Fig. 6 – Deleting “Front Light” function

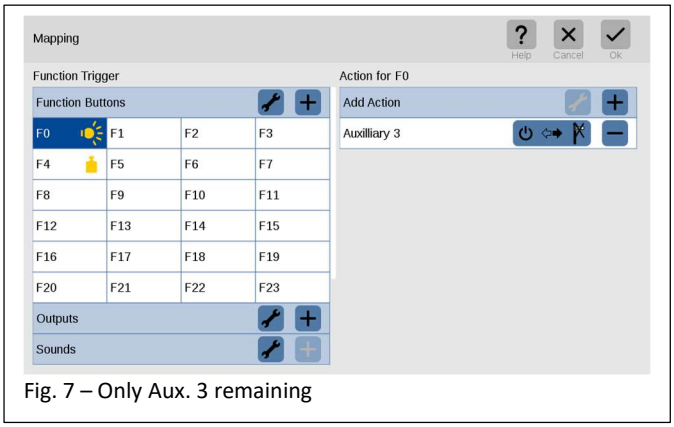

Fig. 7 – Only Aux. 3 remaining

In the function mapping for F0, you can see that this button activates the “Front Light” and “Aux 3”. I want to split these two outputs up. I tap on the minus (-) of the “Front light” on the right (Fig 6). This will remove the Mars Light from that function button (F0) and leave Auxiliary 3 as the headlight (Fig 7).

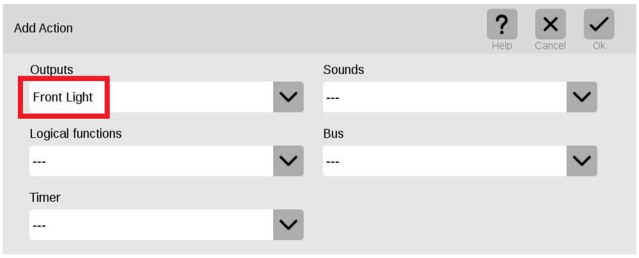

Fig. 8 Choose “Front Light” for F1

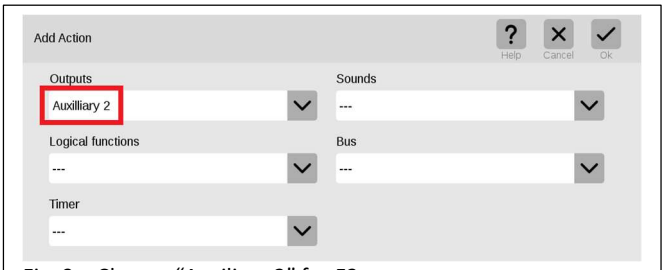

Fig. 9 – Choose “Auxiliary 2” for F2

After this, touch the F1 box then touch “Add Action” on the right and select “Front Light” for the “Outputs” tab (Fig. 8). Then touch the “F2” button and again select “Add Action” except choose “AUX 2” for the action (Fig. 9). If all was done correctly, when F0 is touched the action should be “Auxiliary 3”, then F1 should have the action “Front Light” and the action for F2 should be “Auxiliary 2”.

Selecting Function Icons

Now I can proceed to selecting the correct icons for the function buttons. I will select an icon by touching the wrench of the “Function Buttons” at the top of the function buttons. From this one screen I can set all the icons (Fig. 10).

I selected the “Front Light” icon for the Mars Light at F1, and the “Number Board” icon for F2 (Fig 11). I left them both a “Switching Function”.

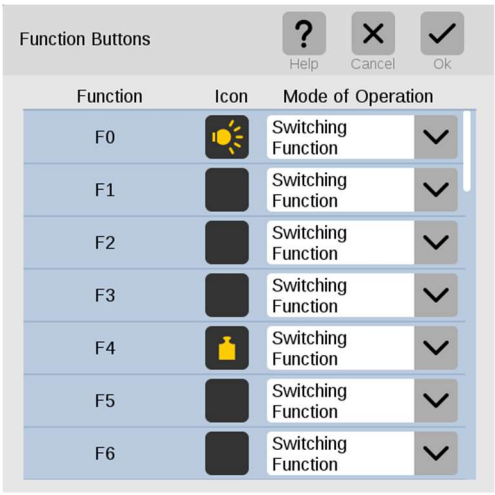

Fig. 10 – Selecting Function Icons

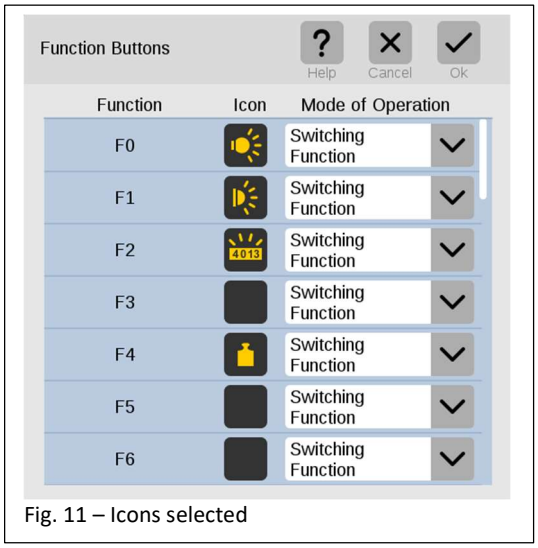

Fig. 11 – Icons selected

Output Configuration

The next thing I had to do was to configure the output of the function. I want the headlight and the number board light to fade in and out when they are turned on and off and the Mars Light needs to pulse.

I can get into the “Edit Mode” by touching the wrench in the “Outputs” line at the bottom left (Fig 12).

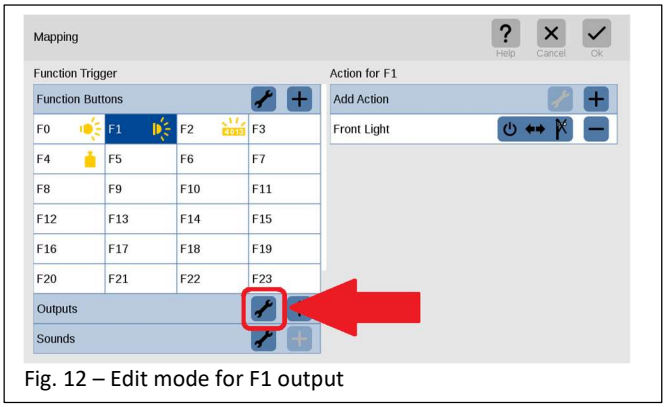

Fig. 12 – Edit mode for F1 output

From here I select “Mars Light” for Auxiliary 1 and select “Fade in / Fade out Light” for Auxiliary 2 and 3 and set the “Period” to 25 and 23 respectively.

In Fig. 13, it shows the output screen where I selected “Mars Light” for “Front Light,” “Fade in / Fade out Light” for Auxiliary 2 and 3 and I set the “Period” to 25 and 23 respectively because I want it to be slightly different fade time.

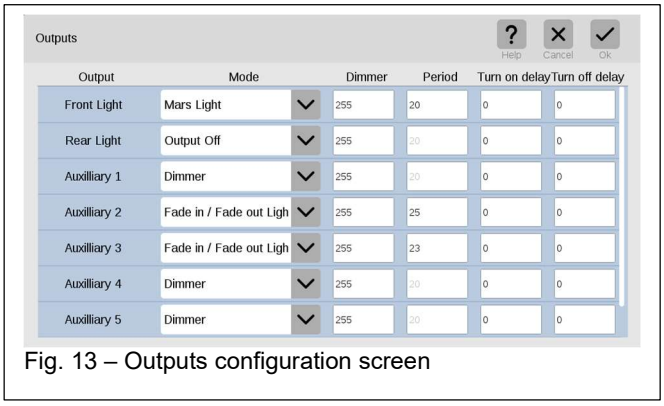

Fig. 13 – Outputs configuration screen

After these changes are made, the locomotive should be ready to run with the new light settings.

One thing to note is that if a decoder “re-set” is performed, these changes will be lost and need to be re-done.

Thoughts

After thinking for a while as to “why” the factory programmed the decoder the way they did, it occurred to me that since this is a retro model that is modernized, the factory made this model so it could be customized with the lighting that the customer might want. There are many different configurations for these lights. So, I think this was the best option that they could have made and then let the customer pick what they like.

Another nice feature is that since this has a 21-pin mounting board in it, it would be very easy to install a sound decoder into it. It would literally be a plug-and-play conversion with only the speaker wires needing to be soldered to the decoder mounting board.

This was a fun dive into a new model with so many lighting possibilities, I hope people customize their lighting to their liking.

Enjoy your hobbies!

Rick Sinclair

Two Train Single Line Shuttle Rework

In this article I will show the process of reprogramming a two-train single line shuttle. Originally this system was developed on a CS2. The system needed to be reworked as it was difficult for the user to switch trains without technical assistance. This was partially caused by the programming limits on the older CS2 architecture. The track plan that I will use is shown in Fig. 1 (below).

The first step was to eliminate a status sensor (contact '24 (18)') and remove all the events written involving the Shuttle routing. I did this because I find it easier to start with a clean slate rather than try to overcome any unforeseen compatibility issues with older code. I will keep the 'Lok...' events as they are simply locomotive commands and not related to track routing. Fig. 2 shows most of the events used previously.

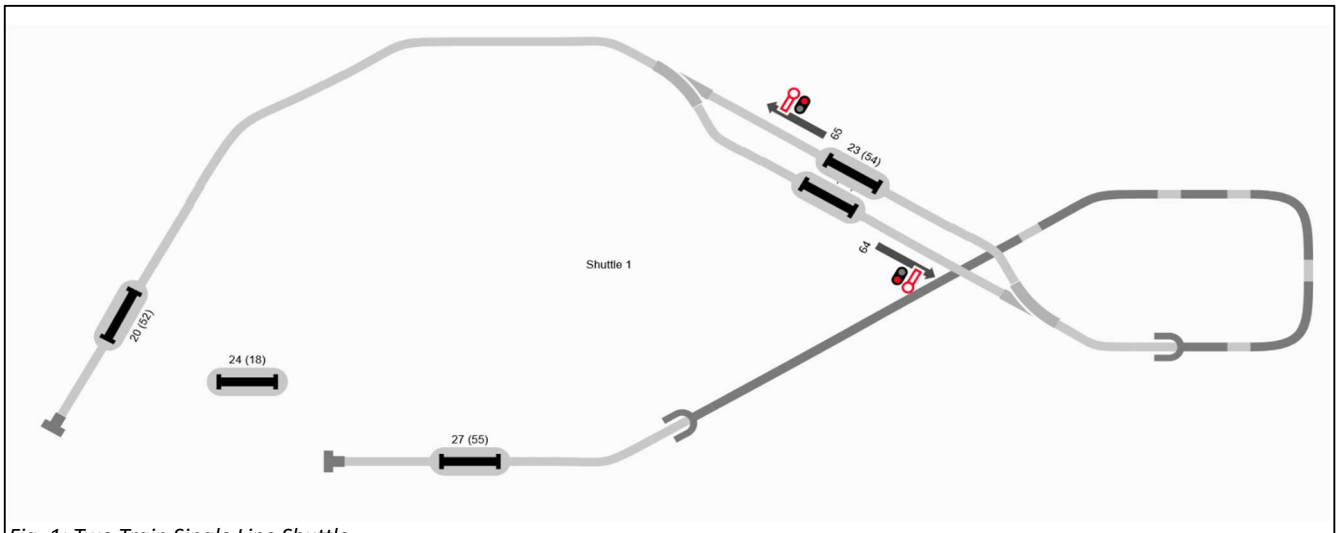

Fig. 1: Two Train Single Line Shuttle

The 'Lok x go' and the 'Lok x stop' events were simple constructs that set the speed for each Lok in use. Fig. 3 shows both events for a single Lok. How I use these events in shuttle routing will become evident later. They also simplify the process of changing locomotives that are part of a routing program, like the shuttle.

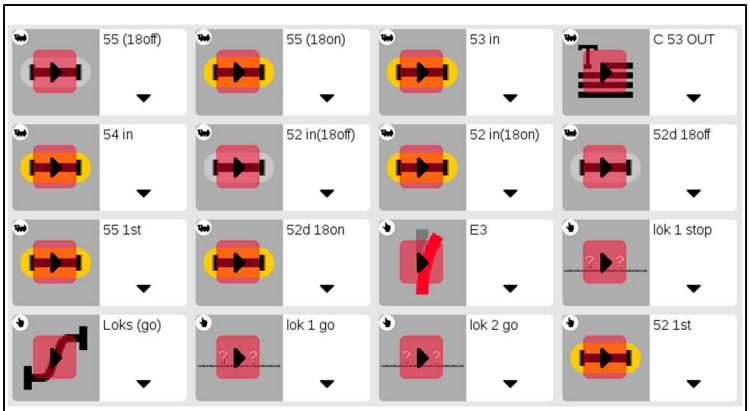

Fig. 2: Shuttle Events

Track Routing and A Two Train Conundrum

My first thought was that I had considered two solutions for the operation of the terminal ends. One used stop sections, while the other required more involved event coding. I found, and later recalled, the fault in using stop sections (i.e. signals or M84 shunts) and deleted the instructions. I think it is important to let you know why this approach is incorrect and that the simplest point to start the routing explanation is at the Mid-point Depot. Inevitably, this should also illustrate that some event programming routines can often result in a 'restart' or re-consideration of events.

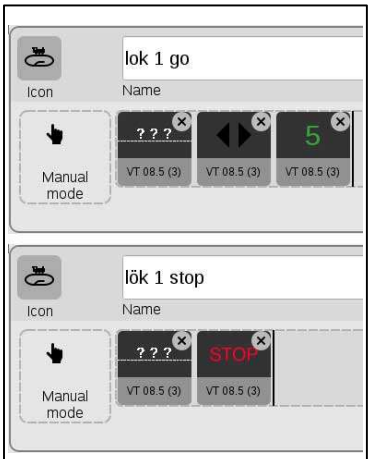

Fig. 3: Lok Events

The reason why a stop section at a terminal station is not the ideal solution lies in the stop section itself. When a train is sitting in a stop area, no commands can be sent or received to the Loks' decoder. Since a shuttle line requires a reverse command being sent to the Loks, there is a deficiency regarding when we can send the reverse command and when the track restores power (a signal set to 'go'). The train may invariably lurch in one direction while trying to send the reverse command.

Approach to the Mid-Line Depot (MLD)

The Mid-Line Depot is a two-lane station with two 'dead' switches set for right hand travel direction. If the switches were not 'dead', then I would be sure to add a turnout switching step in the previous events (above) to make sure the switches are set to right-handed direction.

Any train entering the depot is required to wait until the opposing train arrives. We cannot set the order of which train arrives first, but as a precondition, both lines stop trains at this station. This precondition has not yet been programmed, and I will return to this in the section 'Departure Event for Mid-line Depot'.

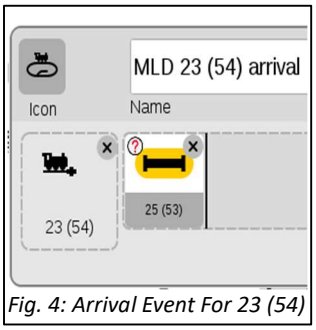

Fig. 4: Arrival Event For 23 (54)

My initial event process for the MLD is to identify when both lines of the station are occupied. The two triggers (occupation sensors) are 23(54) and 25(53) and both need to be occupied before their identical events can take place. The conditional programming is displayed in Fig. 4. When the event is triggered, it checks the opposing line to see if that line is already occupied with a waiting train.

The two numbers for each contact are needed due to the transfer of data from our CS2. The first number indicates the auto ID of the CS3, the number in parenthesis is the Contact ID reserved in the CS2. Numbering

protocols between machines is vastly different. I use both for reference making back tracing of events easier to accomplish.

To explain, when line 1 is triggered [contact 23(54)] the event actions will not take place until line 2 is occupied thus triggering contact 25(53). An opposite of this event is created for contact '25(53)' where the trigger and contacts are swapped. When both contacts become occupied, only the last event that is triggered (either contact 23 or contact 25) will have its actions processed. Do not be concerned though as both events will run.

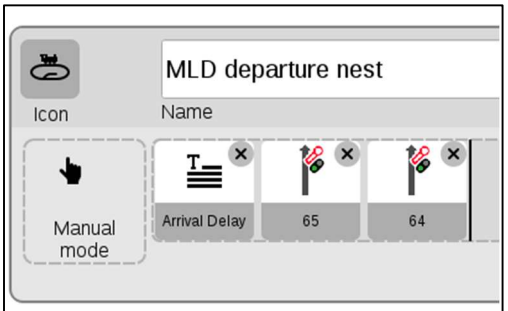

Fig. 5: Event actions to be nested in a triggered event

Mid-Line Depot Event (a nesting event)
Once both trains have arrived at the depot, the simplest action would be to wait an appropriate amount of time then release both 'stop' signals by changing the aspect to 'go'. The process I will show is an example of how to 'nest' an event. A nested event can sometime reduce or simplify both the creation and editing of action(s) that are often repeated.

I will create the event that was just described (a delay of 5 seconds,) then release both depot signals. In Fig. 5, The Event steps show the Text action, which is set for a 5 sec delay and is counted down before it sets the two semaphore signals to 'go'. This is a standalone event with no trigger assigned (the trigger is still in Manual mode).

To nest this event, I simply put it within the two event triggers discussed in the prior section MLD 25 & MLD 23. Fig. 6 displays the changes to the events, compared with that shown in Fig. 4.

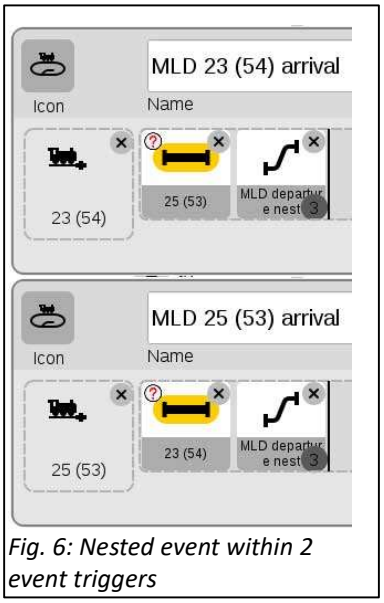

Fig. 6: Nested event within 2 event triggers

Departure Event for Mid-line Depot
The layout does display a shortcoming in the lack of block sensors where the trains have exited the mid-line depot. I will have to add a step to reset the signals to 'stop' mode. Since I was not sure how long the contact area is in the Depot, I will include a delay to allow enough time for the station to clear. Fig. 7 shows the Departure event added for the contact '25 (53)'. I have added a similar action to the companion contact '23 (54)', only the delay time has a different value.
I made these in two separate events, because it allows better use of different length shuttle consists.

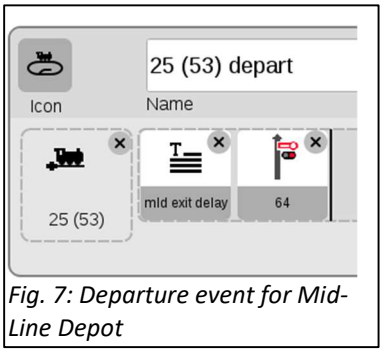

Fig. 7: Departure event for Mid-Line Depot

Learning How to Stop a Specific Train
I have created a situation where I have two trains heading towards their endpoints on a shuttle line. The biggest problem is knowing which train needs to stop, because we do not have a method of knowing which direction each of the trains is heading. For example, I cannot just tell station '2' to stop train 'A', if 'A' is heading towards Station '1' (and it has not arrived yet).

First, Control Contacts to 'Track' Lok Location

The solution is to employ 'Control contacts' to use as control marker for the location of each Lok in the shuttle. I have generated four Controlling Contacts and labeled them for each combination of Lok and destination Station: 'Lok1 to Stn1', 'Lok2 to Stn1', 'Lok1 to Stn2', 'Lok2 to Stn2'.

Controlling contacts are software indicators and have no physical element to put on a layout. They also cannot be used to trigger events. They are used in ways that are like the conditional contacts used in the previously described events with the '?' marking. Their displays can also be changed as a step action in an event.

I have set these up in another event group to reduce the event list clutter. Once they are set, there should be no reason to edit them further. On a layout display page, I have displayed these four CCs that are used as indicators in the following events. I have displayed them as trackers, shown truncated in Fig. 8.

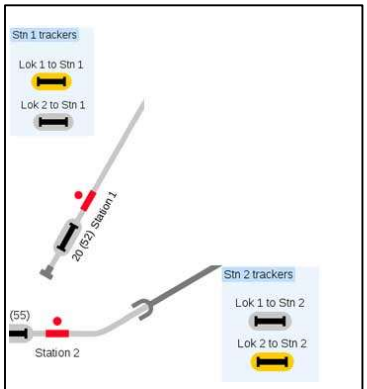

Fig. 8: Controlling Contacts on display

The events that use controlling contacts are simple in build, but more complex in implementation. The example in Fig. 8 shows the event steps that use controlling contacts.
The event basically checks its own status. Then, if true (or matches the '?' status), activates the status of the companion CC, and disables its own status. Each Lok will only have two destinations: stn1 or stn2. This program only allows one of the two destination contacts to be lit for each Lok.

Notice that these are Manual mode events. They are events that are triggered by another event, and this is where it gets challenging to explain.

Applying Controlling Contacts

We are going to return to the nested event displayed in Fig. 5 where the Loks at the Mid-Line Depot allowed to proceed to the terminal stations. I am going to add a Macro IF to check the presumed destination of each Lok. This is where I use the Controlling Contacts as a guide to track each Loks' destination. I mentioned 'presumed' because it is important to remember that CC's have no physical property, and for this method to work, we MUST make sure the physical conditions match the presumed conditions. For example, if the CC shows that Lok1 is headed towards Stn1, then I need to set the Lok1 engine to move in the Station 1 direction.

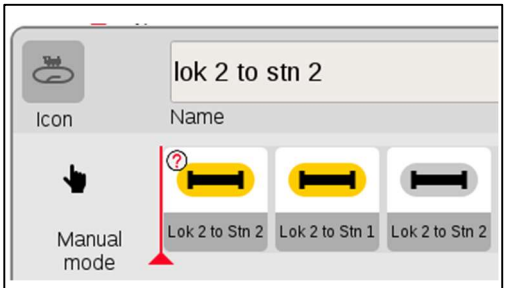

Fig. 9: Controlling Contact event

Macro IF also requires its own level of nested containers. This is evident by the empty containers displayed when you first create the macro. In Fig. 9, I show the empty Macro IF event and the changes made to enable the shuttle. I will explain the contents of each container in Macro and how they are constructed.

The first container is strictly the conditions that must be met for the M-IF to act. If the conditions are met, then the first action will occur. If not, then the second action will take place.

In the two-train shuttle, I have two distinct conditions:
1. Lok1 heading to Stn1 & Lok2 heading to Stn2, or
2. Lok1 heading to Stn2 & Lok2 heading to Stn1
(which is each train heading in the opposite direction)

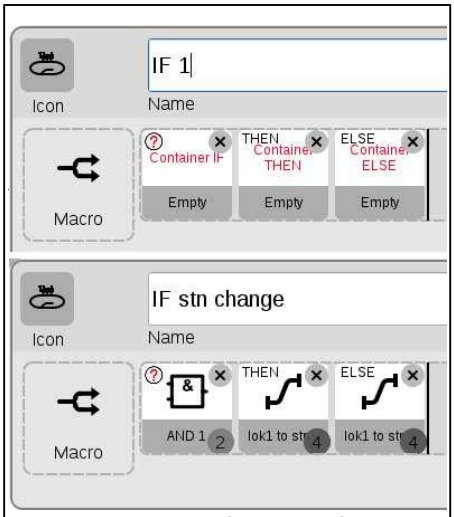

Fig. 10: Macro IF Before and After

For Macro IF, only one of these conditions needs to be described. This can confuse some users, so I will try to explain as simple as I can. If the one condition is good, then the corresponding action can take place. If tone condition is bad, then an opposite action can take place. There may be no need to describe actions where opposite conditions are described as good.

Now, on to Container IF content. Macro Containers can only hold one item. I have two conditions (Lok1 to Stn1 & Lok2 to Stn2) that will not fit into the container. The solution is to create a 'Macro AND'. The 'Macro AND' creates a single container where you can place multiple conditions internally. The 'Macro AND' will only be valid if all conditions are met. I have placed the two Controlling Contacts into the 'Macro AND' that describes #1 of the distinct conditions (Fig. 10). Once I have added the conditions into the macro, I can place the 'Macro AND' into the IF container within the 'Macro IF'.

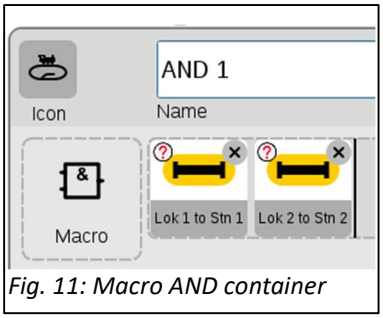

Fig. 11: Macro AND container

What Happens at the "?" Block, aka 'Container IF'
If the conditions in the 'Container IF' pass, then the actions within the 'ELSE' block (aka 'Container THEN') will take place. In this block, I have created an additional Event titled, "lok1 to stn2". Fig. 12 shows that I have set the four tracking contacts to indicate my two shuttle Loks to be traveling opposite of the condition asked in the '?' block. This Event will be placed in the 'THEN' block.

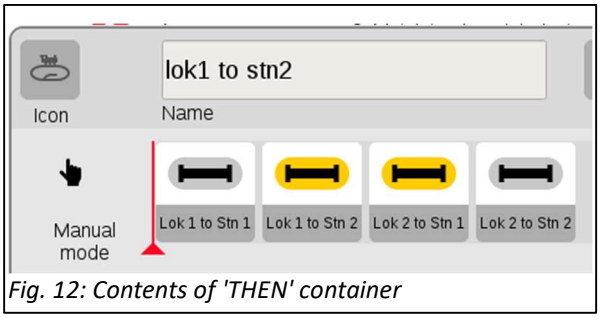

Fig. 12: Contents of 'THEN' container

If the conditions in the 'Container IF' does NOT pass, then the actions in the 'ELSE' Block will take place. For the 'Container ELSE', I have created the event container titled, "lok1 to stn1". I have set the action to match the tracking icons according to the conditions asked in the '?' Block ('container IF'). Remember, the 'IF' condition did NOT pass, so the conditions show that both contacts in the 'container IF' are 'off'. Therefore, the 'ELSE' block must switch back on'. The actions are displayed in Fig. 13.

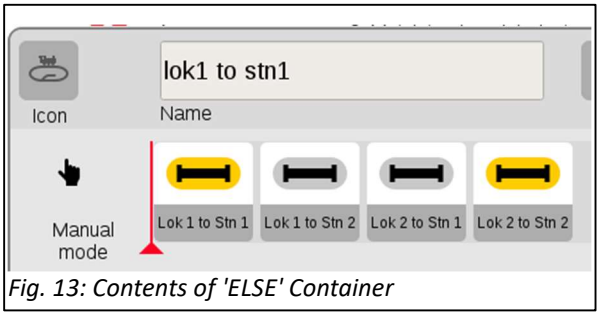

Fig. 13: Contents of 'ELSE' Container

To explain what is happening in the Shuttle cycle, I am using each train's exit from the Mid-Line Depot to set the tracking icons that indicate their destinations. The train's entry into the depot will not be accurate. I did consider programming the destination icons at the terminal points, but this added complexities beyond the scope of the article. The depot created a common point for evaluation and more definitive tracking as it would better relay the destination of travel.

Up to this point, I have shown how to use some event nesting within CS3s Extended Macro events. These are used to create pseudo tracking of locomotives. In the next article, I will complete the 2 train 1 line shuttle programming that will focus on how the tracking is used at the terminal stations at each end.

Curtis Jeung

Upcoming appearances:

- **Milwaukee Trainfest 2025**
  November 1-2, 2025

- **ONLINE Webinar – <u>Delayed one week due to vacation schedule.</u>**
  August 20, 2025

To contact Rick and Curtis for help with your Digital, technical and product related questions:

Phone: 650-569-1318 Hours: 8:00am – 5:00pm CT. Monday through Friday.
E-mail: digital@marklin.com