märklín DIGITAL CLUB

NEWSLETTER

Vol. 34 – No. 2
March - April 2022

Digital Consultants
Rick Sinclair
Curtis Jeung

Current Central Station 3 Version – 2.4.0 (5)
Current Central Station 2 Version – 4.2.13 (14)
Current Mobile Station 2 Version – 3.148

Latest Updates: There has been an update to the CS3/3+ to version 2.4.0 (5) and a Mobile Station 2 update to 3.148.

We are heading the the Rocky Mountain Train Show this weekend in Denver, so we hope to see some club members there since we haven't been to Denver since 2019. It was our last show before everything shut down.

Our first article is about "Cleaning Wheels," and our second article is about "What it takes to Track and Animate Locomotives."

Once again, the subject has come up about track cleaning. It seems that no matter how many times track is cleaned, it gets dirty almost instantly. This is the case for me just like everyone else out there.

To me, it feels like I need to clean the track far too often. Why you might ask, because while doing repairs, I need to test a locomotive that has come into my shop and, as you might imagine, the wheels are not cleaned before I put it on the track. While I have covered track cleaning in a past newsletter article, I haven't really discussed "where" the dirt comes from.

Basically, the dirt on the track is oil and dust/dirt that sticks to it. Like many of you, I check the cleanliness of my track with my fingers. This just adds oil to the track from my skin, but it does give me an idea how dirty my track is.

As one might expect, most of the oil comes from the wheels of a locomotive and to some degree the wagons that are pulled behind it. However, getting down to the root of the problem, the oil in the locomotive comes from the person lubricating it. When I repair a locomotive for someone, much of my time is spent cleaning the old oil out, and usually it has been over oiled. The motor gets warm, the oil gets thinner, and gravity draws it down. Now if the axles are over oiled, on a locomotive or a wagon, the oil makes its way to the wheels faster. Once the oil is on the wheels it will naturally be deposited on the track while collecting dust and dirt all the way down.

Now that the oil is on the track it gets spread around by every wheel that runs it over. So basically, it flows down to, and all over, the track. There you have it, a long-winded explanation of dirty track.

Now we know where the dirt comes from, the question is, how do we control it? While there is no way to stop it because the locos need lubrication. For this reason, I like to use oil sparingly, just where I want it. Therefore, I only oil the spots that need it when I do a locomotive overhaul for someone. When it comes to wagons, I rarely oil the axles. If they squeak, I will investigate it and probably try some graphite first.

So even with this knowledge, I will still have dirty track as everyone does. This opens another problem, dirty wheels.

Now, like I mentioned above, the dirt gets spread around by every wheel that runs on the track. This means that wheels are just as dirty as the track. While it isn't feasible to clean the wheels of the wagons and locomotives in my collection, I do check them if I have not run them for a while and clean if necessary. So, with all of this said, I would like to share some tips on how to clean your rolling stock wheels.

Cleaning Wheels

While cleaning wheels is a simple topic in theory, there are a few techniques, and some are faster than others.

I like to approach the problem with a sense that all wheels are dirty. Let's face it, unless the item is coming out of the box for the first time, it has some dirt on the wheels from rolling on the rails. The longer they roll, the dirtier they get, and just cleaning the rails will only keep them clean until the next time a wheel rolls on it.

One technique I use is just a simple cotton swab and some alcohol. If the wheel is just a little dirty this will do fine. The oil and dirt come off easily and it only takes a couple of minutes. However, there will always be some dirt that comes off. I happen to have some passenger cars of Curtis'. So, I thought I would demonstrate with them. While the dirty wheels aren't a reflection of Curtis' housekeeping abilities, it just shows that every environment will have dirt.

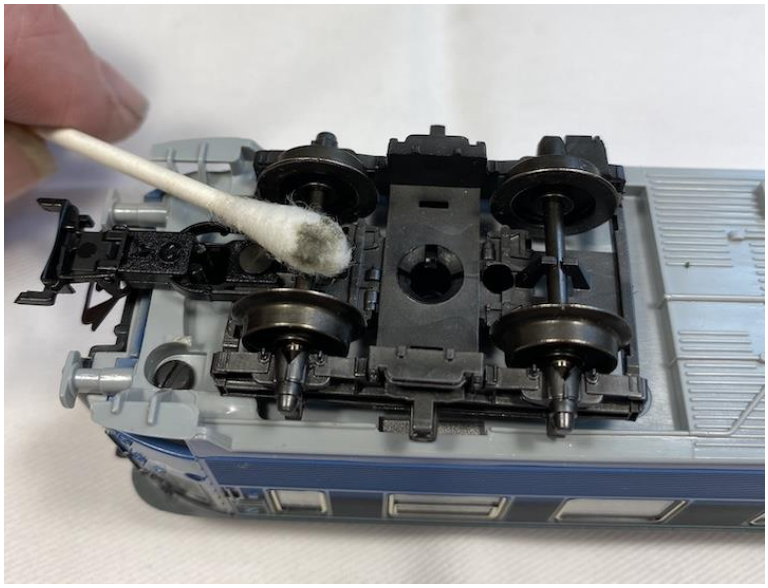
Fig 1. Small amount of dirt

The wheels look clean to the eye, but with a little alcohol on a cotton swab, the dirt can be removed (Fig. 1). This amount of dirt isn't worth cleaning, I just wanted to demonstrate that even clean wheels will have some dirt on them if they have been run. Many times, there will be small deposits on the

wheels either in a section or all the way around the wheel. These deposits will come off easily with a cotton swab also, but it will take a little more work. These small deposits can cause conductivity problems if your wagons have lighting in them.

When a deposit gets very large, it will most likely be uneven around the wheels. Figures 2 and 3 show heavier deposits. With a heavier deposit, a wagon might rock back and forth as if it has a bent axle. Usually, locomotive drive wheels don't get heavy deposits like this, but the pilot, trailing truck and the tender will get the deposits since there is usually much less weight on them and they are "coasting" rather than "driving".

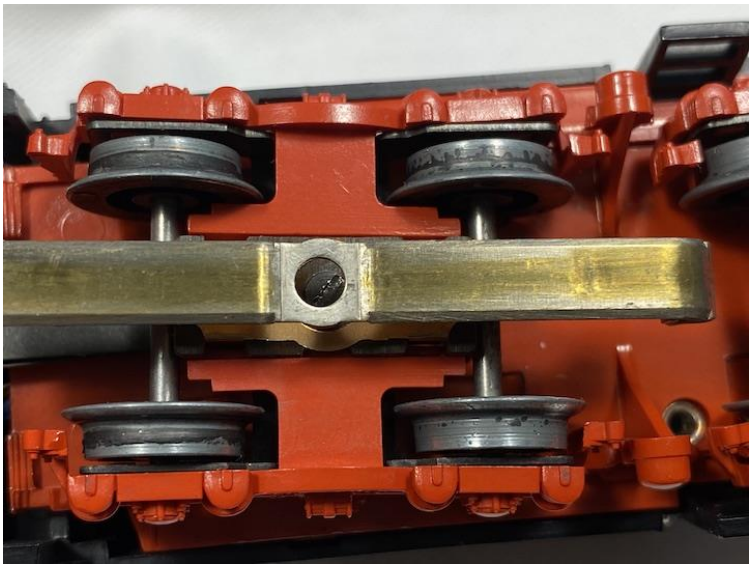
Fig. 2. Large deposits

Large deposits (Fig.4) could promote derailments on turnouts or in an area where the track has been laid in a less than ideal manner.

These large deposits can be difficult to clean off. The oil and dirt seem to stick to the older wheels. I believe they are made of zinc or aluminum. Whatever they are made of it's a soft metal. This poses a problem because scraping the wheel with a metal tool like a screwdriver might gouge the wheel.


Fig. 3. Possible running problems


Fig. 4. Large deposits

For this article I have tried my ultra-sonic cleaner to see how well it will clean. While it cleaned the lighter deposits a little bit, the heavy ones remained (Fig. 4 before, Fig. 5 after).

I do have to say, I was hoping for the ultra-sonic cleaner to do a better job. Even with heat and brushing, it didn't clean as I had hoped, not to mention the thought of taking many wheelsets out of cars, and the time involved, I would be cleaning them for days. After I used the ultra-sonic cleaner, I soaked the wheel in solvent and used a cotton swab with less than stellar results.


Fig. 5 - After ultrasonic cleaning

I have tried a fiberglass cleaning brush with mild success, but in the end, the fastest way that I have found to clean wheels is with a rotary tool.

I found that a jeweler's supply has discs of silicone with abrasives in them, very similar to the track cleaning "erasers" that are common for cleaning the track (Fig.6).

These cleaning wheels make short work of cleaning wheels (Fig 7).

One thing to note, if you choose to use this method, you do so at your own risk. The reason I mention this is that the rotary tool spins very fast even on the lowest setting. There is a high risk of losing control of the tool and the cleaning wheel may rub the bogie and cause damage. The other risk is that the wheels weren't designed to spin this fast. This means that friction can build up very quickly on the axle journals. I use my thumb to slow the wheel down while cleaning.


Fig. 6 - Silicone cleaning wheels

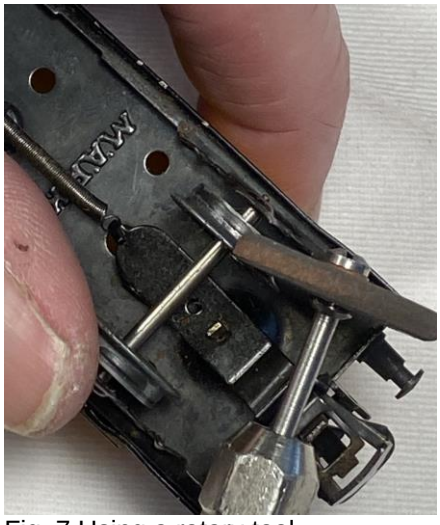
Fig. 7 Using a rotary tool

The reason I use this technique is because I have had a lot of practice and I do it often. It saves me hours of time when I am doing repairs and I don't have to take the wheels out of the journals.

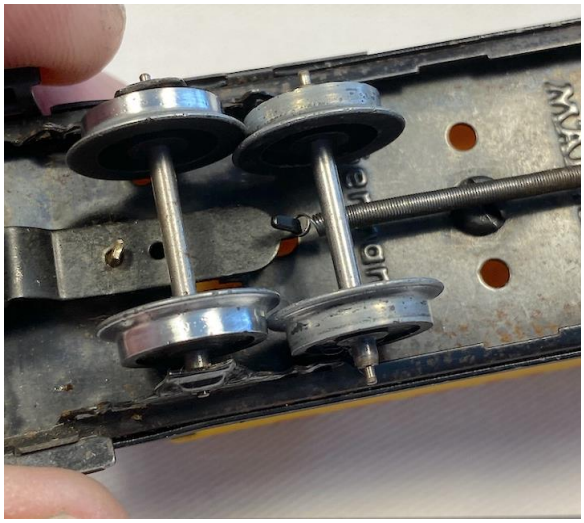

Fig. 8. Cleaning technique comparison

Figure 8 shows the wheel on the right that took me about an hour to clean with the ultra-sonic cleaner then solvent and a cotton swab. The wheels on the left took less than 10 seconds with my rotary tool.

Automatic Cleaning

There are other products to clean wheels. The company that makes track cleaning cars, Lux-Modellbau, also makes a wheel cleaner.

It is a neat device that the train rolls over and all the wheels are cleaned with an oscillation pad as they move along the cleaner (Fig. 9).

Now this isn't for cleaning the large deposits on the wheels and the pads do need to be changed from time to time, but it's part of a cleaning system to keep wheels and track clean.

The unit sets inside the layout framework and is in a central location like the entrance of a shadow station. It can be controlled by a CS2/3 by using a M84. Writing a simple routing script will activate the unit when a train passes over it and into the shadow station. This way the trains in the shadow station have clean wheels before they are sent out to the layout. They are made for N, H0, and G scales.

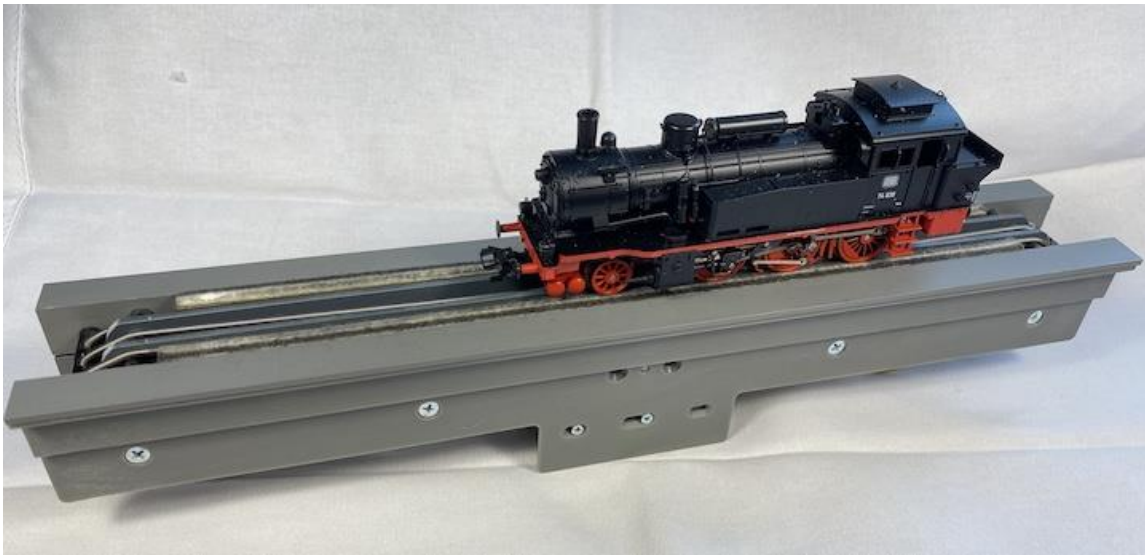

Fig. 9 – Automatic wheel cleaner

Now that I have clean wheels, I just need to clean my track again!

Enjoy your hobbies

Rick Sinclair

What it takes to Track and Animate Locomotives
(My Xmas wish to the Software Developers)

A common request that I have received since I started writing to our readers about event/memory route programming is to explain how to control locomotives within a station. More specifically, I receive comments like, "I'd like to slow down a train when it enters a yard," or "Can I have a whistle blow, before entering a crossing," i.e., typical loc/train behavior. While I may have paraphrased the various requests, it boils down to a simple concept, can I control a loc's actions while in a specific area of a layout? In general, I usually respond with either a “no,” or “not without great effort/difficulty.” In this article, I will try to explain the concepts on setting this up, followed by some challenging details in setup, and conclude with a hopeful plea to the Marklin Software Developers that it can be done. And perhaps, without much developmental effort (granted, I do not have much knowledge in the specific programming architecture of the CS3).

First, the idea of loc control is not to be confused with track articles that influence a locs actions (like a turnout or a slowdown module). Loc control revolves around being able to control loc functions specific to each engine. It’s speed, light and sound functions, for example. If you have dabbled in creating events, then you probably have already created a loc control function which is easily activated with the default manual activation (a button press). Automating these actions around the layout is a different story because the system doesn't have the benefit of what your eyes see, the ability to identify one loc from another.

Two main components to tracking Locomotives

The idea of tracking a loc, basically means having a way to locate the position of a train around your layout. This means there are two components that we need to have: Track sensors and loc identification. Track sensors are used to identify which area of the layout has loc activity. Loc identification is important because we need to identify which loc has activated which sensor. For example, we would need to know which loc entered a station depot.

A sensor can only identify activity and cannot tell one loc from another. Take a single sensor and program two events to blow the whistle on two separate locs. Whenever the sensor gets activated, it will blow both whistles, not the whistle of the loc that hit the sensor (Fig. 1). You

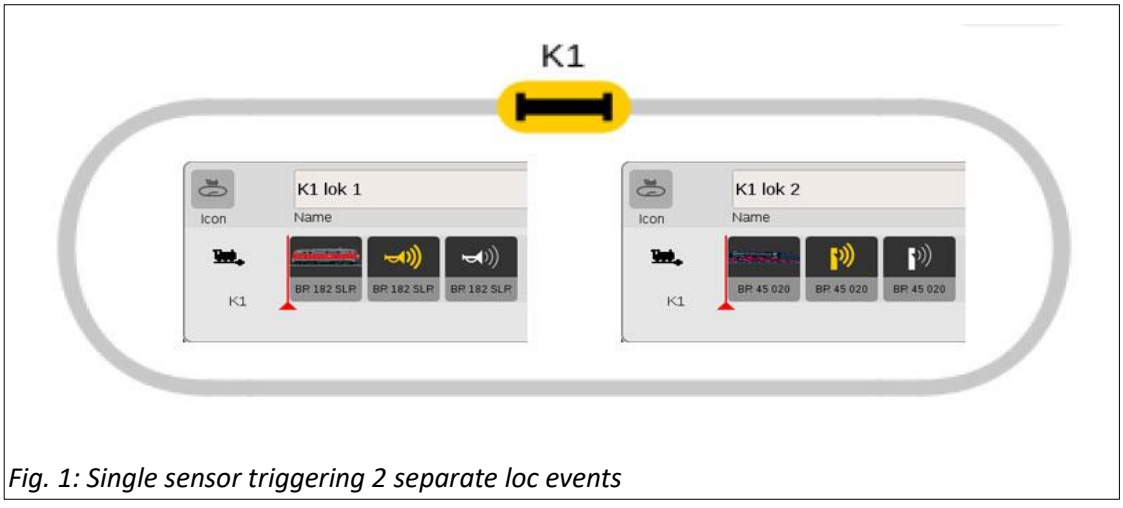

Fig. 1: Single sensor triggering 2 separate loc events

can also remove the event from one of the locs, and if you run both engines then the loc with the event will always sound off whenever EITHER of the locs activates the sensor.

To have specific locs behave predictably at specific locations requires a breakdown in actions that you may not have thought about. It is sort of like breaking down the steps to brushing your teeth, one must remember to remove the toothpaste cap to get any toothpaste. To track a loc's location, we need to first assign a specific loc to a specific block (area). Then, as the loc moves from block to block, we'll need to transfer the locs location from the departure block to the arrival block.

This is a simple concept with two problems. How do we assign a loc to a block? And how do we move the loc from block to block?

The first problem, "how can we associate a loc to a specific location?" This is necessary, because we will need to identify when a specific loc enters or exits a block. You should be familiar with the extended events functionality for programming events. Without this knowledge, you may wish to become more familiar with conditional programming. Currently, the CS3 will allow for setting event conditions which evaluates the status of turnouts, signals or lights and sensors. “Is the signal green or red” or “is a light on or off” are both examples. Locomotives cannot be entered into an event as a condition item. They can only be entered as an action item. In short, you can make a loc DO something, but you cannot check to see what it is, or what it is doing.

We need a way to identify a loc to get around this limitation. The first solution I came up with, was to use a locomotive proxy. Something I can substitute for a Loc to serve as an event condition. I chose the 4-aspect light (off, red, green, and yellow). This gave me proxies for 3 locs as well as a 'no loc' state (light off).

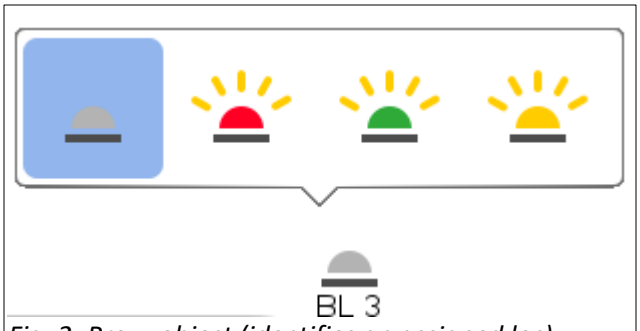

Fig. 2: Proxy object (identifies an assigned loc)

From this point, there are two areas of development that we need to explore. The first is the tracking, or the ability to change a loc's proxy from one block to the next. The second is being able to use the proxy to trigger a locs event. I will begin with tracking the proxies.

Basic Tracking Event

The concept for shifting a locational proxy is simple, I use a Macro IF structure to move one proxy from one block to the next. In Fig. 3, here are two blocks BL1 & BL2. If I assign a red light to the light in block BL1, I can use a sensor trigger (at the entry of block BL2) to identify what color block BL1's light is, then set the light in block BL2 to match.

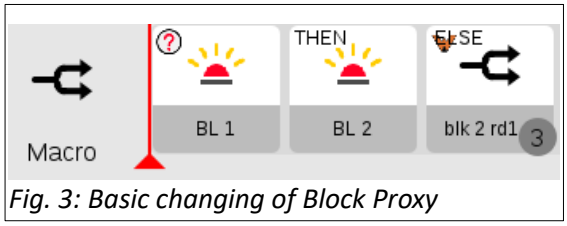

Fig. 3: Basic changing of Block Proxy

Once the train is completely past the entry trigger, I can use a departure trigger to turn off the light in block BL1 (because block BL1 has now been vacated). See the event procedure in Fig. 4.

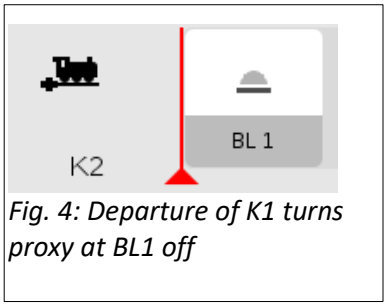

Fig. 4: Departure of K1 turns proxy at BL1 off

Revisiting the Macro IF in Fig 3. You can see the nested Macro If within the 'ELSE' container. This is necessary because there are 4 aspects to the light proxy. When scanning for the color condition of 'BL1', we need to be sure to scan for all possibilities (in this case 4). The expanded event which will display all nested macros is illustrated in Fig 5. The event steps for each Macro displayed in blue is shown in the event steps immediately in the next row.

Note that the last Macro, does not have a filled 'ELSE' container. This is because we only want each row to run once. This is indicated by red arrows and hexagon. We don't want to add a macro back to the top line, because that would make the event run in an endless cycle. This would prevent a setting of a specific proxy to the BL2 block.

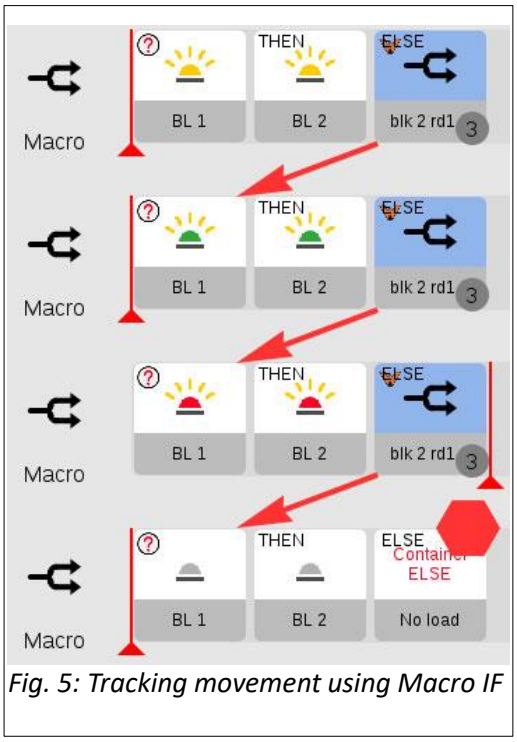

Fig. 5: Tracking movement using Macro IF

The Full Tracking Arrangement

As shifting a locs position can be easy, the implantation is cumbersome. I am not going to go into exact build details because this is not really an instructional article. You need to have a unique identifier for each block around the layout. In the following example, I only have three viable blocks, so I will need a 'BL' light icon for each block (Fig. 6). Plus, you'll need to have the 4 separate Macros for each 'BL' light icon (like Fig. 5).

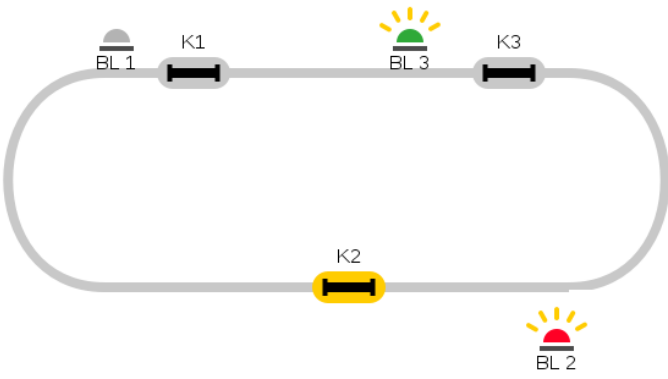

Fig. 6: 3 Proxy Identifiers on 3 block system

Setting Loc Actions

The second setup requirement must have an event for each loc that needs control in a specific block. In my example, with three viable blocks, I can only run two locs (because you must always have an empty block for a train to exit or enter). You need to do this for each block trigger.

Basic Loc Event Handling

To properly assign a loc's action event to trigger, the proxy must first be identified based on the actions of the above tracking. In Fig.7, I've set up a Macro IF, to check that BL3 is green, then allows the Lok speed event to occur. The 'ELSE' container is not used, because if the light is NOT green, then I do not want any action taken. This can also be created using a

simple event (instead of a Macro IF), but you would have to be sure to switch the light switch from an 'action' to a 'condition' (indicated by the question mark on the light's icon, in Fig. 8).

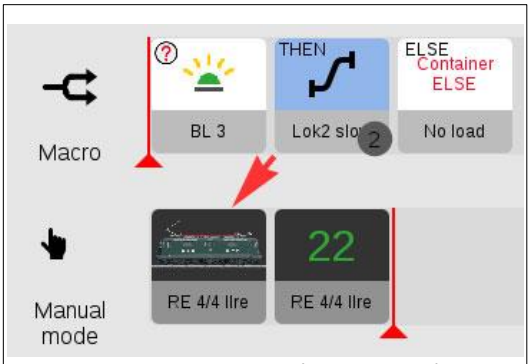

Fig. 7: Loc control event (in a Macro)

Final Assembly and How it works

So far, I have covered the components to identify different proxies, and then use those proxies to implement specific loc action events. The following explains how these work with S88 trigger points.

When a contact is triggered, it must first read in which proxy indicator is active at the triggered sensor. As an example, I use Contact 1's (K1) arrival trigger to activate the macro stream which evaluates and determines the proxy color. In other words, when a loc activates the arrival of K1, I will need to show it as having entered the block by changing the block light to an appropriate color. Fig. 9 shows the K1 arrival trigger and the first Macro IF (the beginning of the macro stream illustrated in Fig. 5).

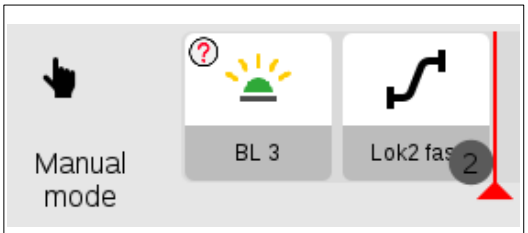

Fig. 8: Loc control event (non-Macro)

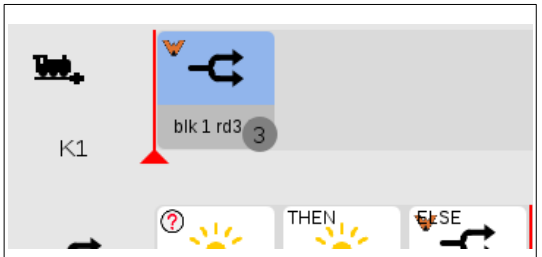

Fig. 9: Using an S88 trigger to start a tracking event

How Loc Control Events Get Activated

Activating the loc control events, like the example shown in Fig. 7, is somewhat of a mystery. Though they are manually controlled events, no actual trigger needed to be assigned. In other words, when the BL3 light was set to Green, it started the events that are displayed in Fig. 7. You'll need to add a trigger contact (replacing the "Manual Mode" with a contact) to start the event, but it will only occur if the condition (? marked icon) is correct (Fig. 8).

Issues of Tracking on a Larger Scale

After I was able to successfully operate the tracking control system I just described, my next challenge was to see if I could implement it on a more complex layout. In the interest of keeping this article to a reasonable length, I will summarize some discoveries and perhaps reserve a detailed description for a later date.

This test is on a layout which currently has 16 total blocks and is meant to have 6 trains (total) on track. To move beyond the 4-aspect light, which limited my loc selection to just three trains, I used an array of Controlling contacts. Controlling contacts look like the icons for the S88, but they are not assigned as either LinkS88 or GFP3 type articles. This meant I had to create a unique article for each loc at each block. Fig. 10 shows the display grid of a tracking sensors.

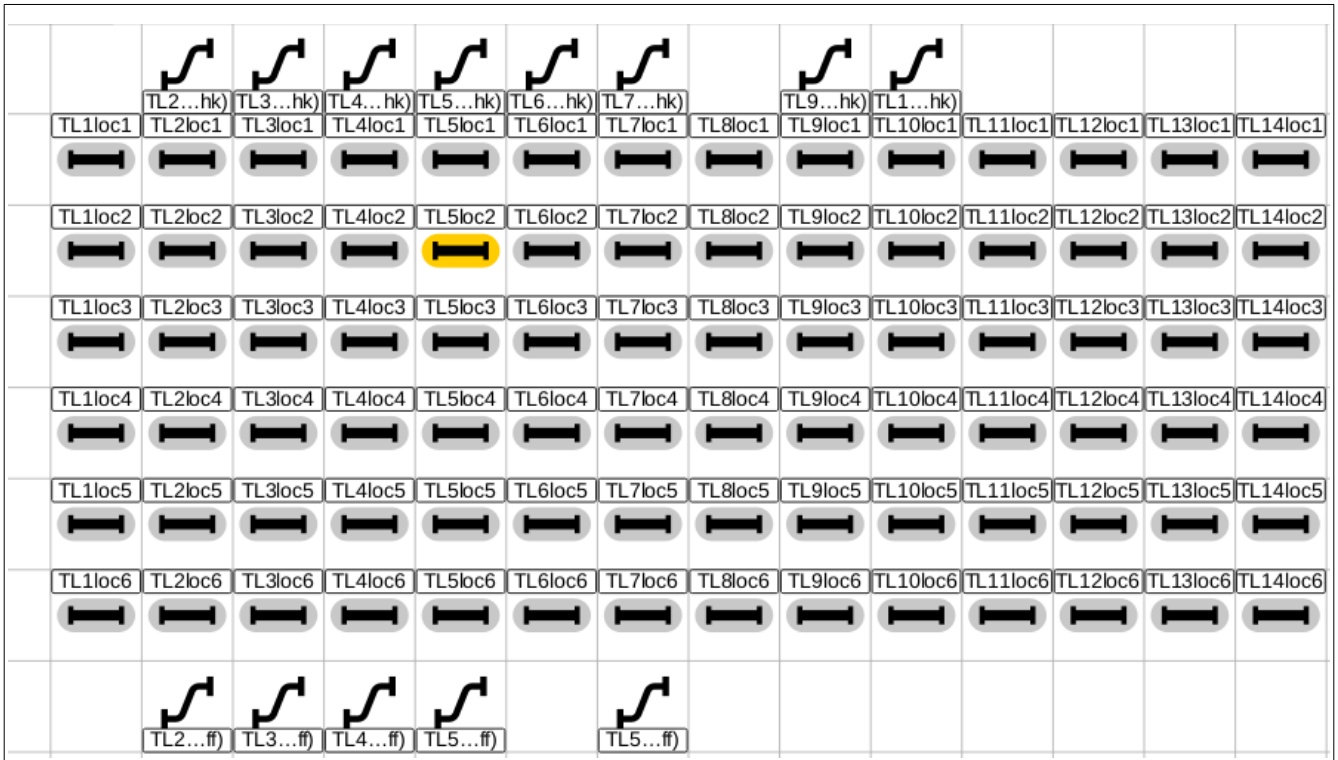

Fig. 9: Control contact articles for tracking loc location

Creating the tracking macros for each block meant that I had to write 96 Macro IFs to insure I could successfully locate where a loc is on the layout. You can see this structure is like the method associate with Fig. 5, but with a sensor 2 aspects, they would look like Fig. 11.

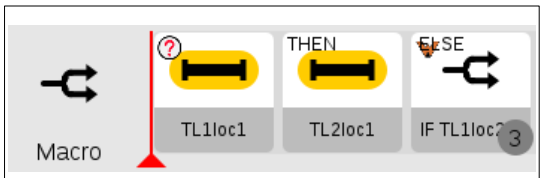

Fig. 10: Tracking Macro IF with sensors

This complexity is one of the main reasons creating localized events is not advised.

Using the Controlling contact is necessary for the above example, because using the LinkS88 or GFP3 type setting, requires pairing to a physical device. The CS3 will give an event Delay status, which causes all events to not function or run.

You will need to have departure events for each block which will turn off any sensors on previously vacated blocks (Fig. 4).

My layout also has a 2-line and 4-line hidden staging area. The tracking in these sections is not described in this article but is possible.

If you've made it this far in the article, you've likely seen how difficult it can be to do create localized loc event handling. The bulk of the difficulty is setting up the multitude of Macro IF events just to enable tracking. The quantity of events for loc control is standard for the location and number of locs that require specific control (at least using this method of localized control). Also, you may have less areas of loc control requirements than you do track blocks, which is a good thing. In the next page, I will explain a more efficient concept, and a Xmas wish to the software developers at Marklin.

My Wish for the Marklin Software Dev Team

My request for software development is to create a streamlined tracking method. In my examination of how this can be accomplished through the existing architecture, I have found evidence of an attempt to optimize tracking. I will display my discoveries and I will also show an example in how I believe this may be accomplished.

The Tracking Element

In my previous examples, I used the 4-aspect light, and the 2-aspect sensor icon. In the Article listed under the 'Switch Contacts' tab, there is an option for Train indicator (Fig. 12).

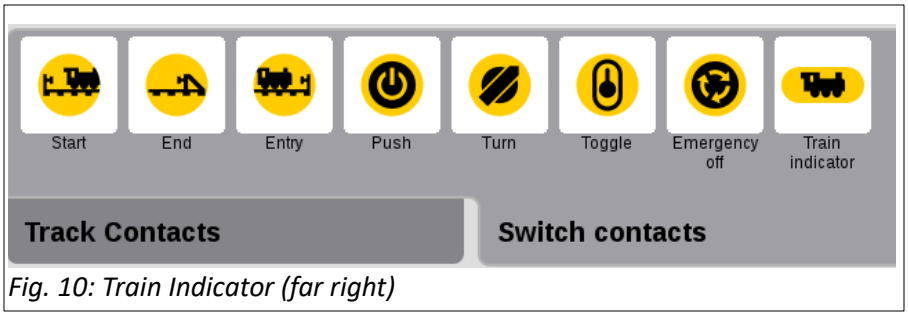

Fig. 10: Train Indicator (far right)

This icon was one of my options when setting up the large-scale tracking. However, I found display issues which prevented its use. Ironically, it is also part of the solution (which I assume may be uncompleted development). In my examples, the Layout Control Panel or the Control Desk, the sensor icons are the same. In Fig. 13, the 'Train Indicator' icon is displayed differently between the two layout displays.

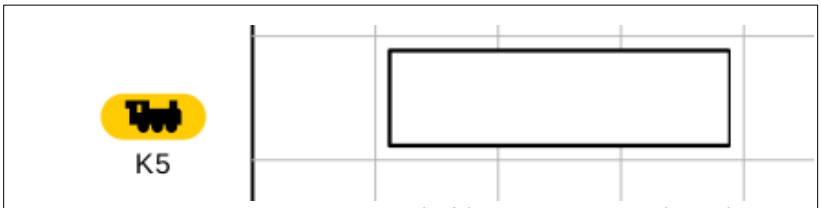

Fig. 11: Train indicator icon: Panel (left), Control Board (right)

The focus on this section is the control desk icon (right side of Fig. 13) and how to enhance its functionality. When you apply this article to an event, it only has the standard set of configurable values (on or off). The icon used appears to be set up for loading of different parameters. In the case of tracking, it would be ideal if a loc ID could be entered. This ID could either be text or object related (the latter term is something used in programming). Since we load locs into our CS3's library, loc objects are automatically created (these objects contain all the running data and identifiers for each loc). At this point in time, you cannot load or enter any information in the open space icon.

How this would look in the CS3

These fields would work in the same basic tracking events as my examples. This is illustrated fictitiously as a comparison in Fig. 14 (next page). The advantage of this proposed method would be minimizing the event count. You would not need the multiple Macros to identify the aspect of each assigned light or sensor. Instead, the CS3 software could 'READ' the input object (either a textual name, or ID number) of one block, then transfer it to the next. In the Example of Fig. 14, BL1 would be read, and the date transferred to BL2.

The way this is visually applied to a control board, is displayed in the two diagrams of Fig. 15. You could enter either a loc's icon, or a textual facsimile into the track blocks. Then, the Macros will take over the transferring of data from one block to the next.

Implementing Loc control would be an adaptation of what is used in Fig. 8, except now we could identify a specific loc as I show in Fig. 14. The only difference is we could use a standard event to create a longer event script that involves more than a single train action. I.e., lights, sound, and speed control in one event.

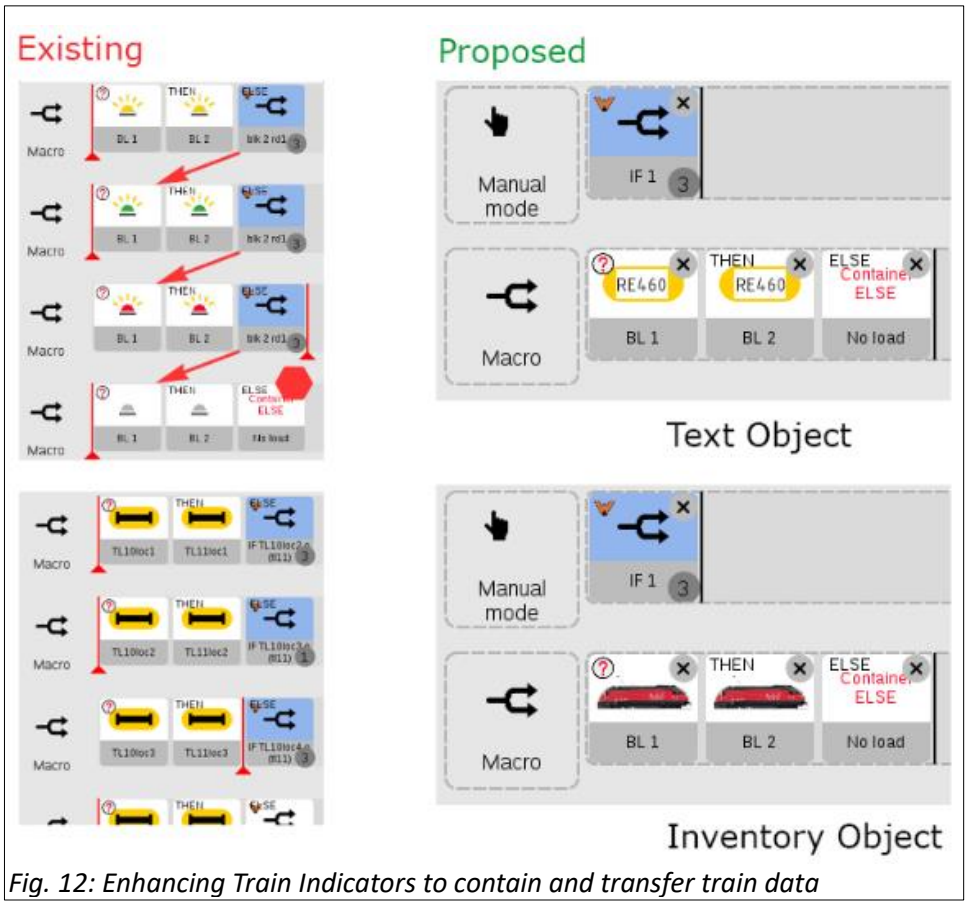

Fig. 12: Enhancing Train Indicators to contain and transfer train data

I feel that it would be easy to adapt the CS3's software to enable this tracking method, which would allow for the common desire to control a loc at a specific location. Slowing a train in a yard or station or sounding a horn as it approaches a crossing can be accomplished. Now if only I could overcome the language barriers and technical jargon then relay the concept to our tech gurus at Märklin.

Curtis Jeung

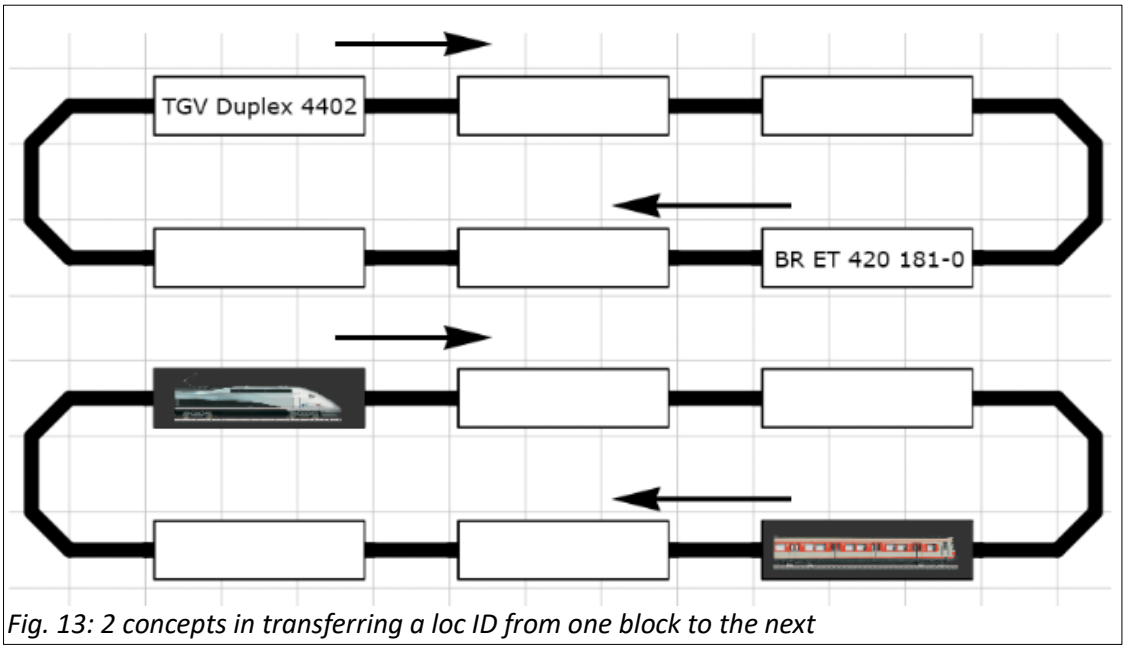

Fig. 13: 2 concepts in transferring a loc ID from one block to the next

Upcoming appearances:

Rocky Mountain Train Show
National Western Complex
4655 Humboldt St.
Denver, Colorado

Saturday, April 2nd - 10:00 am to 5:00 pm
Sunday, April 3rd - 10:00 am to 4:00 pm

Booth# D8

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

Phone: 650-569-1318 Hours: 6:00am – 9:00pm PST. Monday through Friday.

E-mail: digital@marklin.com