märklin DIGITAL CLUB

NEWSLETTER

Vol. 36 – No. 6
Nov. – Dec. 2024

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

Digital Consultants
Rick Sinclair
Curtis Jeung

We have just returned from the "Trainfest" show held in Milwaukee. Because the show was cancelled last year, it had been two years since we were there. Even though the show was in a new venue this year, it was great to see many of the same people again and we hope to be there next year.

Our first article is about the various ways to connect a brake module (72442) to a layout while our second topic is about understanding the Marklin Digital system.

I was helping a person on the phone who has a mix of old and new signals when it occurred to me how many ways there are to connect a brake module to a layout. One might think it is straightforward, however, there are a few different ways all ending in the same result. Please note that Marklin uses the words "Signal Module" and "Brake Module" for the same item 72242.

Brake Module Wiring

We have covered this topic in the past but in different articles. Here I will put them all together to give our readers a choice of which way works best for them.

Whether a brake module is controlled digitally or analog, the wiring to the track is the same. There needs to be three sections of track insulated from each other and the rest of the layout. It does not matter if the layout is two-rail or three-rail. The question is "how to control the brake module?".

Analog Control

The most basic way is shown in the brake module instructions. In Fig. 1 the diagram shows the brake module controlled by a 7272/72720 control box. This is a simple analog controller with a momentary push button interface, which means the brake module would be controlled manually.

While the manual control of a brake module might be desired in certain instances, often a brake module would be controlled automatically. In Fig. 2, the brake module is still being controlled analog, but it is connected to a reed switch (7555 shown) for 2-rail operation. Both a circuit track (24994) or reed switch can be used in C-Track operation. These two techniques are favorable to a contact track because a reed/circuit track is a momentary pulse and not a constant pulse like a contact track. So basically, a brake module should be activated with a controller that sends a momentary pulse.

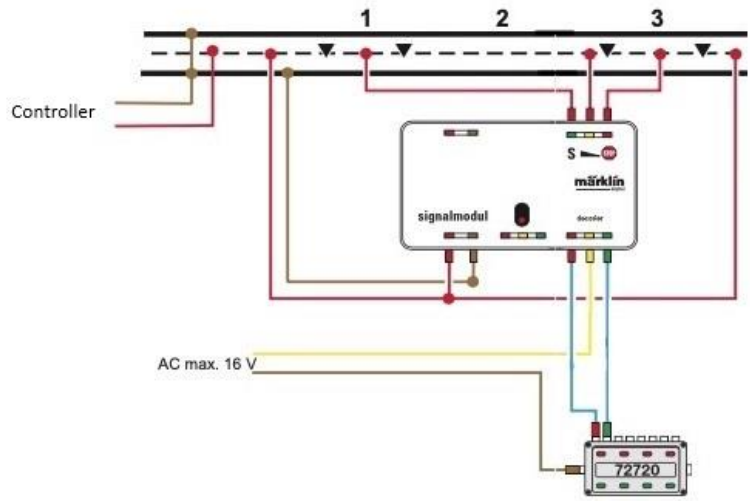

Fig. 1 - Analog/manual control of a brake module

One topic to note is there needs to be a second reed or circuit track somewhere on the layout to deactivate the brake module once it has been activated. This is explained in Digital Newsletter Vol. 31 #3 and #4.

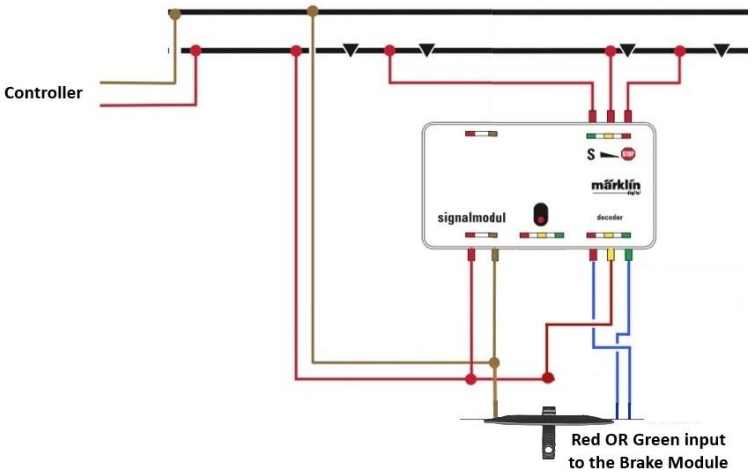

Fig. 2 – 7555 Reed contact control (analog) – 2 rail track shown

Digital Control

With digital control, there are some options to consider starting with hybrid control, up to full digital control. One thing to remember is that once an article (m83, signal, turnout, etc.) is in the article list of the CS3, it can be controlled digitally. If an item can be controlled digitally, it can be used in an event to operate automatically. So, if there is an analog signal connected to an m83, the analog signal can be controlled digitally from the m83.

Hybrid Control

If a person chooses, the brake module can be controlled digitally with an analog signal attached. This is a blend of an analog signal and a brake module being controlled digitally with an m83 (60832) (Fig 3). In this scenario, the m83 controls the brake module and an analog signal.

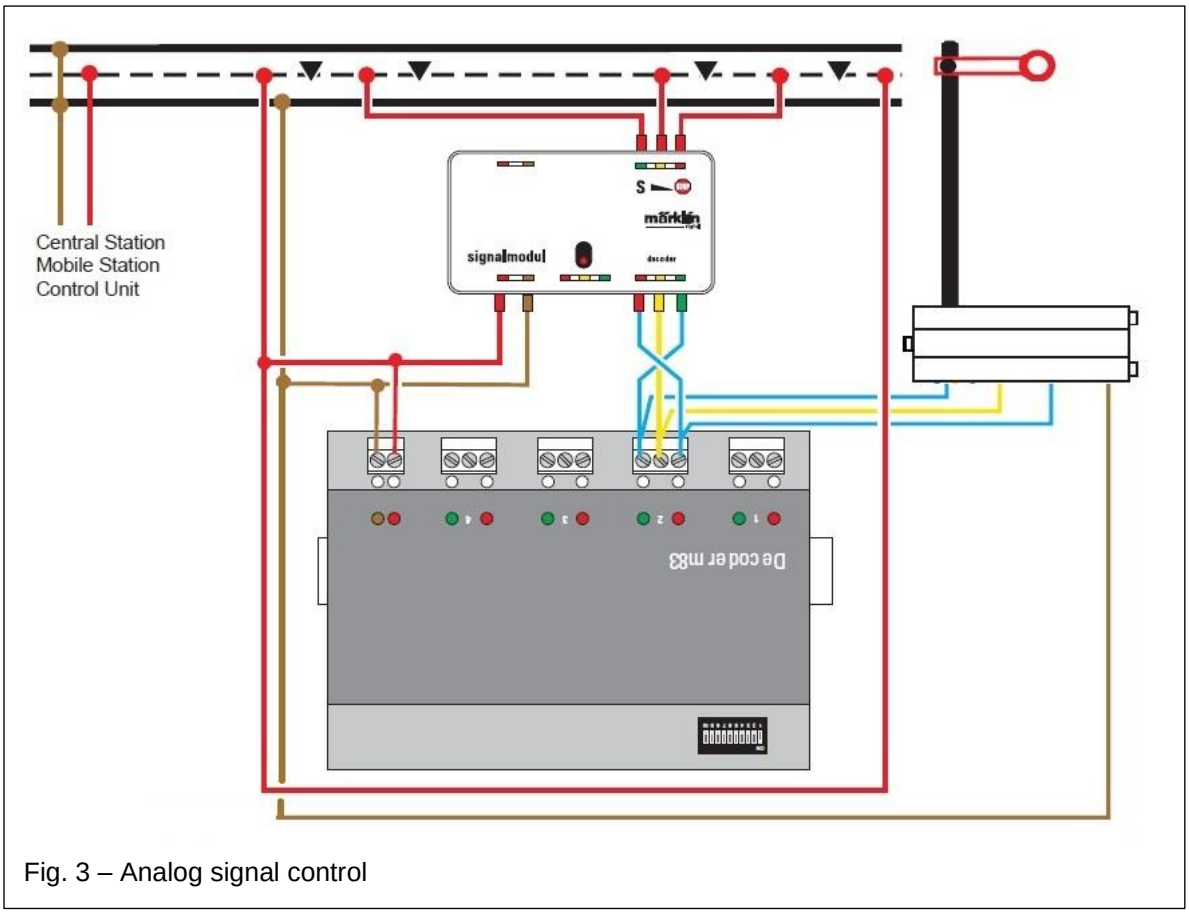

Fig. 3 – Analog signal control

Full Digital Control

With full digital control there are also options to control a brake module. The standard method is to have the signal control the brake module. This is as simple as it sounds, when the signal is green the brake module is turned off and when the signal is red the brake module is turned on. Figure 4 is from the signal manual for H0 signals. Since the signal is digital, it can control a brake module.

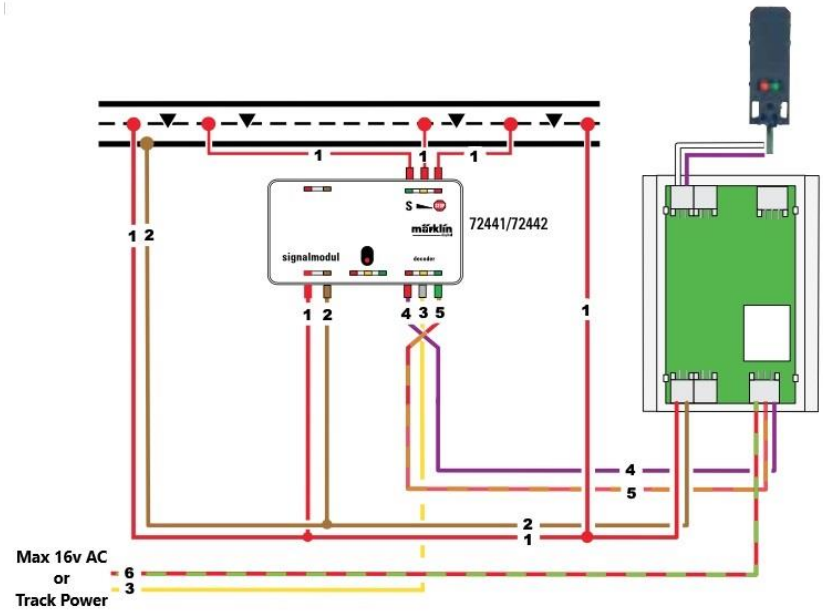

Fig. 4 – Digital signal control

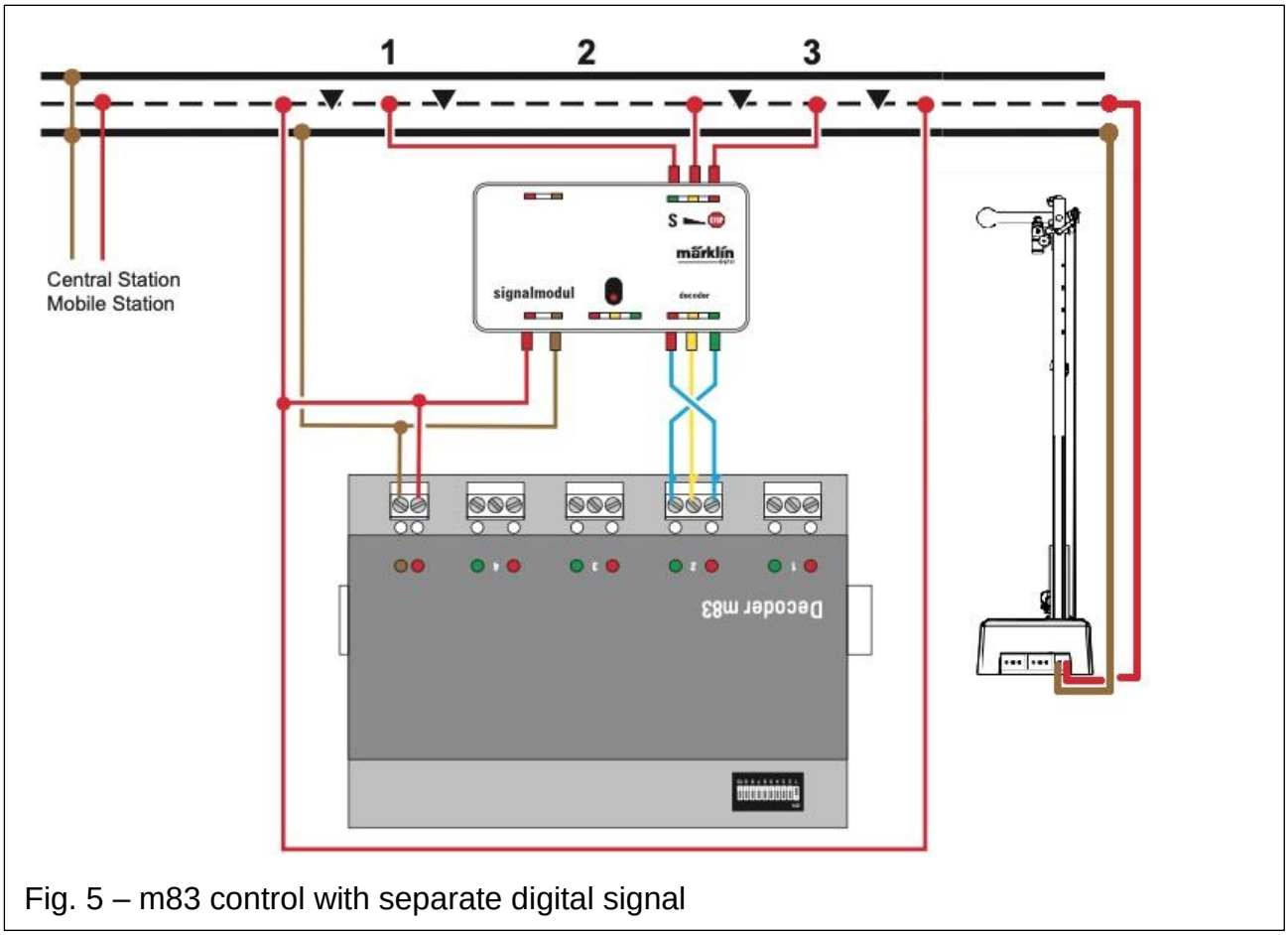

Fig. 5 – m83 control with separate digital signal

There is another technique that we like to use as well. What we do is have an m83 or k83 control a brake module like in figure 4, except we do not have a digital signal connected to the m83/k83. We separate the two for individual control of each. This gives us the advantage of adding a delay in an event script once the signal changes to green, but before the brake module turns off (Fig. 5).

The theory behind this is that even though a locomotive has a green signal, it does not mean it should go. Trains run on a schedule. The green signal means that the block is clear for the locomotive to proceed, but it still must wait for its “time” before it starts to move. This is particularly fun to model on a layout when other enthusiasts are around. They think the locomotive has stalled and then try to push it – good times!

This last technique would probably not be used but it does work. I had the thought that if there was a signal out on a layout that was not near an m83. One would just plug the signal into the brake module of course, but for separate control, as I just mentioned, it would require an m83, which would require a long set of wires. So, what if there was a single stand-alone decoder to control a brake module? – Oh wait - There is!

Since I was throwing the simplest solution out the window (running wires to an m83), I thought that a turnout decoder should be able to control a brake module because it is a short pulse just like an m83. My test worked and I was able to control a brake module with a 74462-turnout decoder (Fig 6).

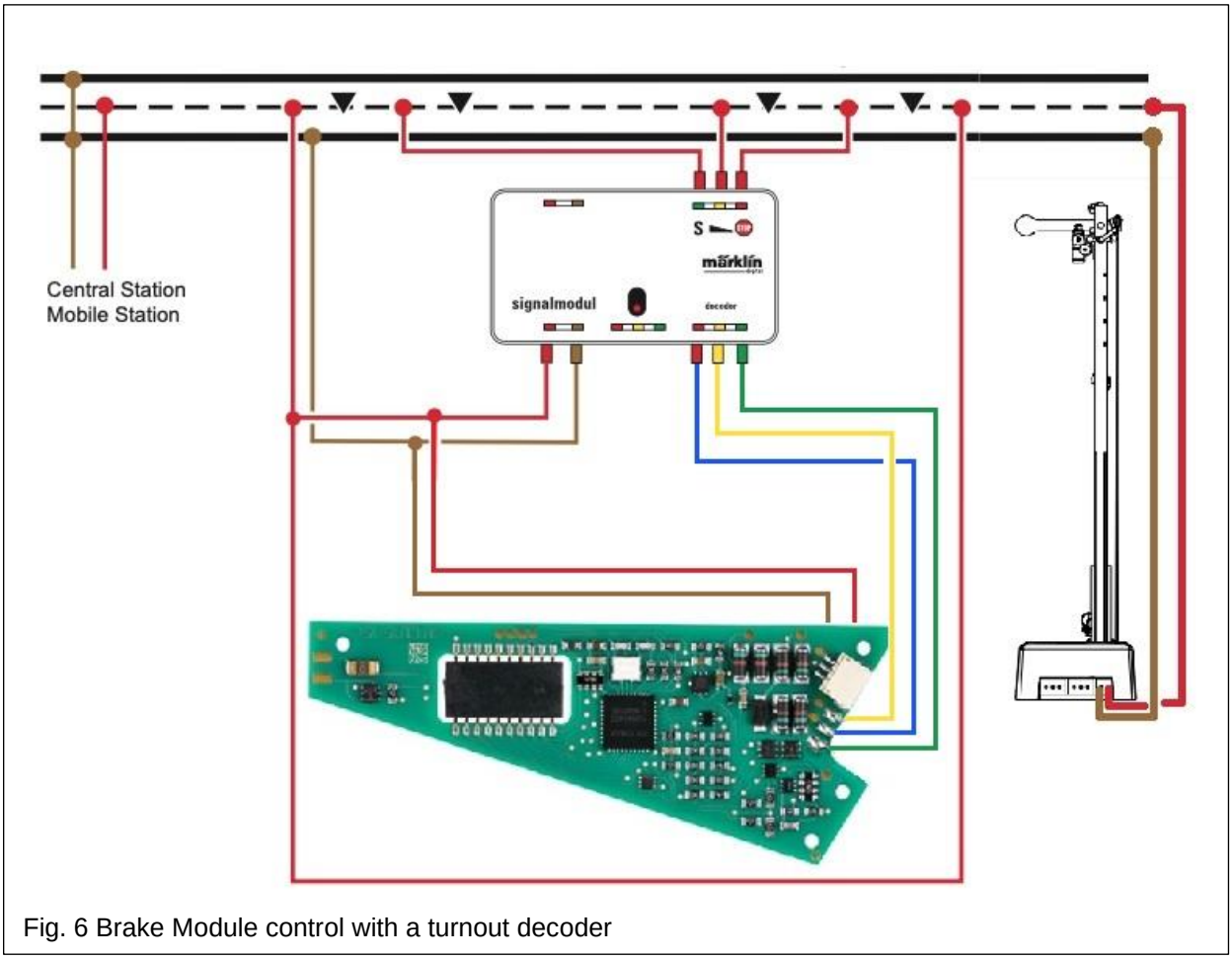

Fig. 6 Brake Module control with a turnout decoder

This method is very unusual, and I don't think anybody would have a use for it, but it is possible, so I thought I would mention it. It would be nice to hear if anyone uses this so keep us in mind and let us know "if" and "why" you did this!

This has been a fun "dive" into the different connections of a brake module, and I hope it helps people choose what technique best fits for them.

Enjoy your hobbies!

Rick Sinclair

A Little Bit of Understanding

You may not realize this, but you were just subjected to a pun... a bad pun. To understand why. The topic of this article is not directly train related, but instead, to gain some understanding of digital structure and how it is used in your Central Station. The current state of Marklin's Digital system is so well developed that we can become confused when compared to what you may have learned in Marklin's earlier digital days. However, we use some examples from the early years to help understand what the current system is now doing.

You may find that the beginning part of this article may be challenging to understand and read through. However, towards the end, there will be some simpler explanations that should help you understand how the Marklin Digital system is using some of the settings that you create.

What is a Bit

To start, the basic digital building block is known as a bit. It is a single digit bit of information that can be translated in different ways depending on how it is used. For example: yes or no, 1 or 0, on or off.

What is a Byte

If you use multiple bits together, it can have many more forms of translated materials. A group of bits is known as a byte and can have varying lengths (from 2 to many bits). One of the early forms of Marklin Digital is the Delta system. On the lok decoder you may see a 4-switch dipswitch, which you would use to set the Delta address. To set the address you would have to understand the counting system of the particular system you are using. Luckily, there are resources in the form of an address table to help you sort that out.

Explaining the math of the Delta Addressing

You can skip this section if you do not need a nap. And more importantly, this information is archaic in regard to the currently robust mFX decoder architecture.

Each dipswitch on the Delta decoder has a numerical value, but these numerical values are different than what we commonly use. For example, 12 has two digits, one for our single numbers ('x2' = 2) and one for our values of 10 ('1x' = 10). But since a dipswitch can only have two positions (on or off = 1 or 0), a dipswitch typically does not have a value of 10. This is a binary way of counting where each digit in our four-switch device increases by a multiplier of 2. Compare this information in the table below. And it is not yet an accurate representation of Delta Addressing.

Standard way of looking at number 1101 (and I mean one thousand one hundred and one)			
1000's	100's	10's	singles
1	1	0	1
one thousand	one hundred	none	one

In a Digital Binary world, the same number would be:

In Binary Land, the 1101 would be equal to 13 (8+4+1 = 13)			
8's	4's	2's	singles
1	1	0	1
one 8	one 4	none	one single (1)

Now, if you are an astute person, who is interested in such a thing, you will notice that the Delta decoder table does not match either of these numbering systems. I will try to explain this with the possibility of not being too thorough, because the archaic information will not really make a difference.

The numbering system of the Delta decoder uses a Trinary numbering, with a two-position switch. This means that the value of the switch and the switch position can really mess with your head. First, the placement of the switch (i.e. 1st,2nd, 3rd digit), is multiplied by a trinary factor (3). Second, the value of the 'singles' digit is not 1, but 2 due to the trinary setup (i.e a 3 switch values would be 0,1 or 2). I will try to illustrate this in the table below.

Delta decoder math of 1101 (54+18+2 = 74)			
54's	18's (3x the value of 6)	6's (3 x the value of 2)	2's (0 or 2)
1	1	0	1
54	18	0	2

And the last bit of information on a delta addressing scheme is the inverse nature of the switch settings. The numbers are added when the switch settings are in the OFF position, NOT the On position. Also, digit values are read into the decoder in a reverse order. The number (switch value) on the right would **NOT** be the 1's and the left would **not** be the 54's. The number on the Left is the 1's position and the Right column would be the highest number (i.e. 54's). So, if we wanted our value to be 74, the binary code may be written as 1101, but since the decoder reads dipswitch settings in a reverse fashion it would be seen as 1011. And, since the dipswitches are inverse (where 1 would actually be off), our dipswitches would be set to Off, On, Off, Off.

This information is multiplied when you consider the FX style decoders that have 8-bit dipswitch addressing. Like I mentioned, just use the address table. Like I said, just use the address table.

Marklin Motorola (MM) and Digital Control Command (DCC?) Decoder Types

The MM (Marklin Motorola) addressing scheme is based on the Trinary bit system (explained above), the DCC I am going to assume is binary in general. If you need to enter in a Decoder manually, you will also need to set the Decoder type initially. It sets up the specification for the decoder type in use so that it and all functions can be operated properly. If you get it wrong and your lok is unresponsive, you cannot change it. You will have to delete the lok and reenter it into your system, mainly because the configurations for each decoder type is quite different between the two.

The Bits in Decoder Functionality

Up to this point I have only explained how Bits are formed to create a numerical value. And now I will try to explain how this is used to create an address using the Dip Switch selectors on a decoder. For clarity, Delta and FX decoders use dipswitches along with various accessories (Turnout decoders, signal lights, etc.)

Within a decoder's programming, this bit arrangement is used in a somewhat different manner. A decoder does not just send a single numerical value in a bit stream (Byte), like the dipswitch setting. What happens is the bit stream is subdivided into various categories that are interpreted by the decoder type. And this stream can be sent twice in one packet. This is a form of error correction, due to weakened digital signal or signal interference.

Classification/Parity/Check Bit (all the same)

The first example of what has just been described can be considered a parity or a check bit. This is a bit that is added to the beginning a byte (packet) of information. This results in having a couple of purposes. One, it tells the computer that a new set of instructional (or configuration) data is being sent. In this fashion the computer knows when to start listening (as opposed to joining in a conversation midway). Two, for some devices, this can be used to tell the computer what type of device or protocol information is being sent.

For example, let's look at the manual for a Marklin mFX Digital Signal. Fig. 1 shows the statement regarding the setting of switch 10. Switch 10 becomes the parity bit for the Central Station to

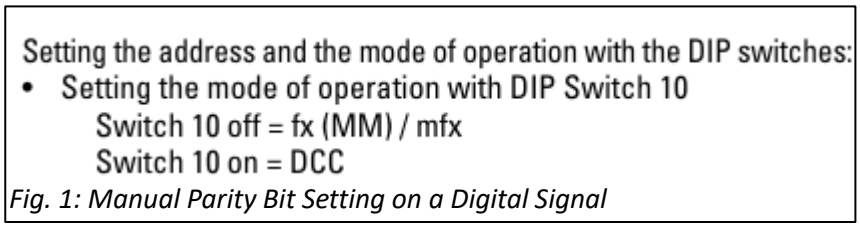
Setting the address and the mode of operation with the DIP switches:
- Setting the mode of operation with DIP Switch 10
  Switch 10 off = fx (MM) / mfx
  Switch 10 on = DCC

Fig. 1: Manual Parity Bit Setting on a Digital Signal

understand what decoder type of address it needs to receive. Remember that MM may be a trinary base number, whereas the DCC is binary. You would have a completely different address of numbers using the same dipswitch settings. This may only refer to addressing only, where the instruction code may be strictly binary. Also recall that while the 10th switch is the last on the board, it is the first data bit to be seen by the controller.

This parity system is also why you should not worry about turnouts having the same address setting as a locomotive. There is always something in the byte packet that tells the controller what type of device command is being received. (i.e. is it a Lok, a turnout, a signal, etc.?)

Subdivision of Byte Packet

Another way to see how this works is by comparing a Decoder's CV table (control values). Bear in mind that this is a conceptual interpretation, NOT an actual representation. Also, remember that this is all in code, not based on the Dipswitch settings (which is primarily for addressing). Code will look like a bunch of 1's and 0's and is commonly some multiple of 8 digits long (8,16,32, etc.).

Subdivision of Byte packet	CV 63	CV 50	CV 08	CV 05	CV 04	CV 03	CV 01	Check Bit
Values	1-255	0-15	8	1-255	1-255	1-255	1-255	0 or 1
Number of digits needed for max val	8	4	4	8	8	8	8	1

The following table is a closer representation of how the computer receives the information.

Subdivide command structure with byte packet combined below. Remember info is read from the right.							
CV 63	CV 50	CV 08	CV 05	CV 04	CV 03	CV 01	Check Bit
10001000	1111	1000	10001000	10001000	10001000	10001000	1
10001000 1111 1000 10001000 10001000 10001000 10001000 1							

Further Subdivision of Byte Packet

You can see that each CV has multiple bits and that it results in a numerical value for the respective CV setting. In many cases, the range is appropriate for its setting. For example, speed setting or braking delay would make sense to be in a range of values (0-255). However, there are some CV values that are not set to a range but contain multiple functions which result in a number value.

For example, in the most complex form, you might see a CV table that has values that look like (not an actual CV representation):

CV *	Headlights = 1 Distance light = 3 Number Boards = 5 Cabin Light = 10	Setting 0-255

You would not be able to set the single CV value to one of the displayed numbers (1, 3, 5, or 10), as this would prevent you from setting the other functions on. Instead, you would be adding the values that you want to be on (function wise). For example, if you only want the number boards and headlight on only, then you would set the CV value to 6 (1-headlight, 5-number boards, 1+5 = 6). You can also see that any combination of sub function math will not equal any single function and create an error. Example, Number boards = 5, and you cannot get a value of 5 if you only turn on the headlights and distance lights (1+3 = 4).

mFX+

The changeover from Marklin's early Delta digital systems to the current mFX architecture is highly evolved. For backwards compatibility the system had to retain its structure. In its current state, it is even more of a mystery. The primary differences are that much of the addressing and coding are developed and handled automatically.

Multiple mFX locomotives (loks with mFX decoders) that have the same factory addresses are handled as unique addresses within Marklin's Central Station or Mobile Station. In the background, when a CS finds a duplicated address, it indexes the address so that it knows it is a multiple used address. In terms of this article, the CS creates a parity bit for a duplicated address.

In a crude example, using a duplicated address 46, the CS may log multiple units as: 46.1, 46.2, 46.3, etc. Is this something to worry about? Probably not, because even a basic 8-bit binary scheme (like this article explores) you would have 256 different values. So, if you think you feel limited from having an indexed address of 46.255 (not 256 because, 0 is still considered a value), then you can consider changing a lok's address. Also, I suspect that in common code dynamics, the bit depth is larger, and you may have up to (or more) than 1024 as a max value. What would you think about an address index of 46.1024. I would say you have really, really deep pockets.

Thanks for reading (if you made it through this far)!

Curtis Jeung

Upcoming appearances:

- **ONLINE Webinar**
  December 11, 2024

- **Amherst Railway Society Railroad Hobby Show**
  January 25 and 26, 2025
  9:00 am - 5:00 pm on Saturday | 10:00 am - 4:00 pm on Sunday
  West Springfield, Massachusetts
  Learn more at: https://www.railroadhobbyshow.com/

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