

RAILBRICKS™

BRICK RAILROADING MAGAZINE



The French Revolution

A peak inside FreeLUG
- the French Enthusiasts LEGO® User Group



INSIDE:

- ◀ The Diesels of Dara Norman (aka Swoofy)
- Building Realistic Wooden Trestles
- PBricks and Pushing the 9v Limits
- Power Functions Train Tricks
- And Much More!

Next Stop... Chicago!



Brickworld

2008TM

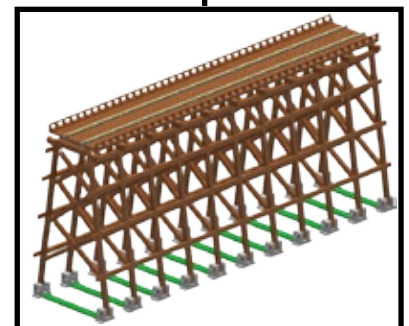
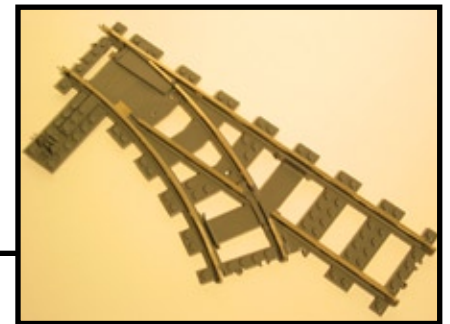
JUNE 19-22

June 19-22, 2008 • Chicago, Illinois
www.Brickworld.us

RAILBRICKS

BRICK RAILROADING MAGAZINE

The Whistle Stop.....	4
LEGO Factory Update.....	5
Blue Brick Review.....	6
Breaking the Code.....	8
Club Spotlight: FreeLUG.....	9
Flashback Review: 7760.....	10
Builder Spotlight: Swoofy.....	12
Reverse Engineering Challenge.....	16
Power Functions + 9v Train Tricks.....	19
Keeping 9v Alive.....	20
Advanced 9v Track Modification.....	22
Use the World as Your Inspiration.....	26
A Bridge Too Far.....	27
Selective Compression.....	34
Instructions: Horse Stock Car.....	36
Trainspotting.....	43
LEGO Trains and LEGO PBricks.....	44
FRED's View.....	63



All Aboard!

The RAILBRICKS Team

Senior Editor:

Jeremy Spurgeon

Staff Editors and Writers -
The "Think Tank":

Erik Amzallag

Steve Barile

Matt Bieda

Benn Coifman

Tim David

Didier Enjary

Holger Matthes

John Neal

Mark Peterson

Larry Pieniazek

Jordan Schwarz

Content Contributors:

Thorsten Benter

Jan van Dijken

Ondrew Hartigan

Alban Nanty

Xavier Viallefont

Copy Editing/Proofing:

Chris Spurgeon

Copyright © 2007-2008 RAILBRICKS

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

Welcome back! Issue 3 has been dubbed the 'Tips & Tricks' issue because we wanted to give you all as much inspiration as possible while working on your layouts. I think we have something in this issue for everyone, whether you are just getting into the hobby and looking for some ideas on how to integrate some of your 9v and RC track for interesting effects, or the advanced hobbyist looking to push the limits of your old RCXs.

Since our last issue, the RAILBRICKS website opened up a building instructions section. So far, we have had some great submissions, but we need your help in making it the one stop place for fan built train models. After finishing reading through this issue, head over to the website and either grab a few instructions for inspiration, or submit a few for others to share in your creativity.

Thanks again for reading and supporting RAILBRICKS. I hope you have as much fun perusing these pages as we had putting them together!



Jeremy and IFoL (Infant Fan of LEGO) Elora

As always, this is a community built around sharing ideas, so if you have an idea for an article, submit it to submissions@railbricks.com.

Play Well!

-Jeremy Spurgeon

Printed issues of RAILBRICKS will be available at <http://www.lulu.com>.

Instructions and Tips & Tricks articles within RAILBRICKS fall into one of three categories and are labeled with the following icons:



Beginner



Intermediate



Advanced

LEGO Factory Update

by Jordan Schwarz

If you're a regular visitor of the various LEGO-related news sites, you've probably heard some of the rumblings regarding the latest update to the LEGO Factory palette. The brick assortment governs the models that can be built in LEGO Digital Designer (LDD) software and then ordered online. It undergoes periodic revisions and the most recent, issued in April, has some implications for train fans.

The Factory palette gained some novel elements from the new LEGO Factory Space-themed sets. It saw the addition of new pieces in most brick categories and now contains a substantial assortment of pieces in light blue, dark blue, lime green, and orange. There are more minifigures to choose from, and windows are being updated to the style seen in newer sets such as Green Grocer. However, astute train fans were quick to notice what was missing, as opposed to what had been added.

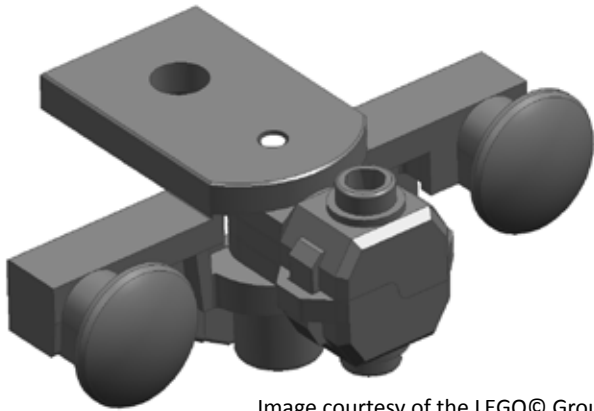


Image courtesy of the LEGO© Group

You can still purchase wheel sets for train building, but the 9V elements have been removed from LEGO Factory. This doesn't come as too great of a surprise, since these elements are slowly being discontinued from LEGO Shop At Home in advance of the 2009 release of the Power Functions-based trains. Unexpectedly, a new type of train buffer beam has been added, and the old, familiar model has disappeared. The Factory palette now contains a black plow-style buffer and a standard model in both black and (new) light grey.

These new buffers have sparked considerable de-

bate and speculation since their arrival in the palette. The swiveling magnet has been replaced with a permanently-attached magnet, fully enclosed in a plastic case. Examination in LDD shows that these magnets are articulated to allow side-to-side rotation, but the magnets themselves do not appear to spin as their predecessors did.

A recent communication from Jan Beyer of LEGO answered the wave of speculation regarding these new

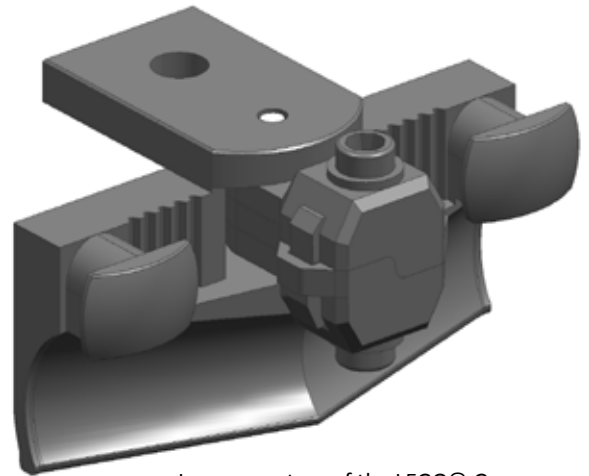



Image courtesy of the LEGO© Group

train buffers. The new buffers are designed to prevent children from swallowing magnetic parts. Swallowing two magnets at separate times can create (and, in fact, has lead to) potentially lethal complications. Surely, consuming magnets is not a habit of most adult LEGO train fans, however LEGO has made this change in consideration of the safety of children and the possibility of future laws banning the distribution of swallowable magnets, such as those found in LEGO train buffers. The communication from LEGO indicates that these new couplers will be magnetically compatible with the old ones. This is made possible via a fully enclosed magnet which spins in its enclosure to achieve the proper polarity for coupling.

One possible concern of train builders is that the new magnets, by design, cannot be detached from the buffer beams. This will make it more difficult to incorporate couplers into designs where a standard buffer is not desired; the front coupler of the 10020 Santa Fe Super Chief locomotive is just one example. As always, creative train fans will surely find a way to circumvent this issue, although it will require more effort with this new coupler design. 

BlueBrick

REVIEW

by Jeramy Spurgeon

THE HISTORY

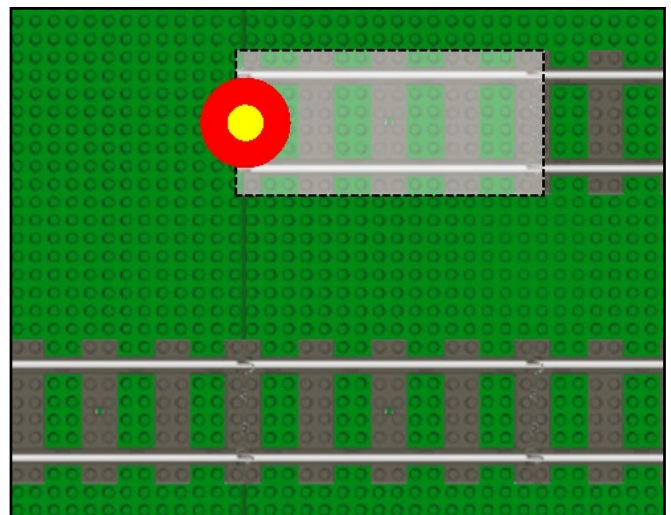
A critical tool for any train club (or individual train fan) is the ability to design layouts before implementing them in real brick. Thus far AFOLs have been primarily reliant on a layout design software called Track Designer, written by Matthew Bates. For years, Track Designer was the de facto layout designer. It is an easy to use and graphically friendly software, though support for it in recent times has been non-existent. More recently, Cary Clark developed a vector based track layout software called TrackDraw. It seemed like the software would overtake TrackDesigner in popularity, mainly because it offered scalable graphics and an open source file format for home brew piece additions, which Track Designer lacked. TrackDraw, however, never made it out of its Beta stage and further development on it has been canceled. Because neither of these popular software was being supported any longer, FreeLUG's Alban Nanty set off to write his own layout software, which he would later name BlueBrick.

THE FEATURES

To run BlueBrick, you must first have the .Net Framework 2.0 installed on your computer. This means that BlueBrick will only run on Windows based machines. Running BlueBrick is easy once this initial step is done, simply open the program folder and click the application. There is no installation and BlueBrick can be run from external hard drives without needing to install additional software. BlueBrick opens with a

grid-based layer visible, separated into 32x32 baseplate squares. Double clicking on the grid layer or going into the options for the software, allows for a high level of customization to the initial grid view, with the ability to save your preferences so that you retain your settings the next time you open BlueBrick.

Jumping into BlueBrick is fairly straight forward. A parts palette floats to the right of the area map with each piece having the ability to be dragged and dropped onto the workspace. Clicking pieces within the palette will add the piece to an activated piece on the map, either by connection point, or to the right of the activated part.



Adding track is easy in BlueBrick.

One of the many highlights of BlueBrick is the layer feature. Much like Photoshop, individual layers can be created that contain certain elements of the layout. For

example, a layer could be created that contains just the tables for your plan. Another layer, stacked on top of the previous, could be designated as the baseplate layer. Finally a layer can be designated as the track layer. The software then provides the ability to toggle the layers on and off, allowing you to quickly go back to count the number of tables needed without having to move your other components out of the way. By separating your track and baseplates into layers, it becomes very easy to select large amounts of similar items without disturbing the surrounding parts.

BlueBrick also has the ability to import Track Designer files! This backward compatibility was quite a feat for Alban (read the following section for his secrets in cracking the Track Designer code). TrackDesigner files that have pieces that aren't in BlueBrick will be imported with a red X in place of the part. These can be easily deleted or replaced with a custom part.

BlueBrick also offers easy custom part creation. For baseplate parts that do not need a defined connection, simply find a top down view of the part, adjust its size in relationship to other parts, then save it to the appropriate parts folder. In BlueBrick, 256x256 pixels is the equivalent to 32 studs x 32 studs.

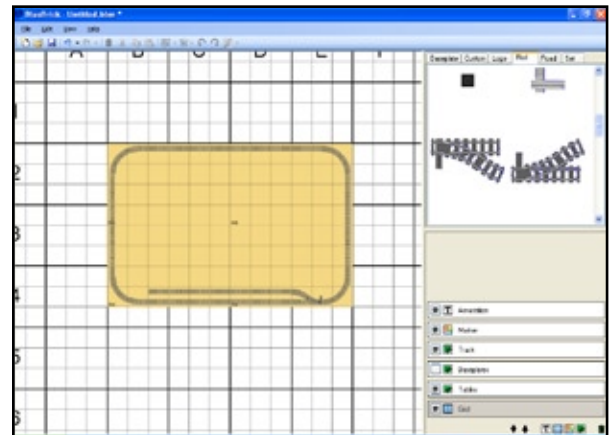
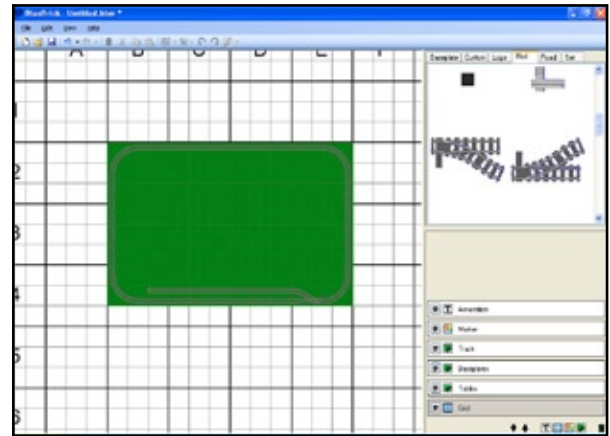
Name	Size	Type	Dimensions
3811.1.gif	27 KB	GIF Image	256 x 256
3811.2.gif	27 KB	GIF Image	256 x 256
3811.3.gif	22 KB	GIF Image	256 x 256
3811p02.1.gif	31 KB	GIF Image	256 x 256
3857.2.gif	14 KB	GIF Image	256 x 128
3857.3.gif	11 KB	GIF Image	256 x 128
3857.14.gif	30 KB	GIF Image	256 x 129
3867.2.gif	6 KB	GIF Image	128 x 128
3867p01.1.gif	9 KB	GIF Image	128 x 128
4186.7.gif	66 KB	GIF Image	384 x 384
6024.8.gif	22 KB	GIF Image	256 x 256

Other parts files, such as track, have two components: a graphical representation of the file (.gif) and a connection definition file (.xml). Because the connection definition file is an XML document, creating new track geometries is easy, as long as you can figure out the geometry and trigonometry involved. And you thought all of that math you learned in school would never prove useful!

The software offers simple image export with many options. The resulting images are very crisp and clean in appearance and print very well.

THE GOOD, THE BAD, AND THE UGLY

While BlueBrick may well prove to be the new standard track design software, it is not without its flaws, though as Alban further develops the package, these may soon be of little concern. First of all, it is a Win-



The layers feature of BlueBrick allows you to easily and quickly distinguish between different aspects of your layout.

dows only application. MAC and Linux users will not be able to use this unless they are running a Windows operating system on their machines as well. It also requires the .Net Framework, which can be a stumbling block for novice computer users.

Secondly, BlueBrick uses raster based graphics which cannot be scaled to large sizes without distortion. This is only a problem if you wish to export very large size versions of your layout.

These are only minor nitpicks of the software and once users get into building their layouts with it, these will seem almost non important.

BlueBrick can be downloaded from Alban Nanty's website which includes frequently asked questions and simple tutorials:

<http://bluebrick.lswproject.com/> 

Breaking the Code

Secrets of the Track Designer File Format Revealed!

by Alban Nanty

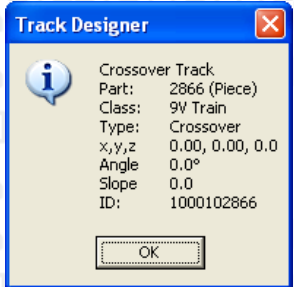
A widely used tool by the AFOL community to prepare their exhibition layouts is Train Depot Track Designer written at the end of the previous century by Matthew Bates. The NGLTC website currently hosts the latest compiled version of this software but the source code appears to be lost. At the time it was developed, it was common for the developers to use a binary format to store the data handled by the application, for two main reasons: first it was faster to load, and secondly it was a common way to protect your file format if you were writing a commercial software. Nowadays, this practice is less common because the computers are fast enough to load human-readable files, and even commercial software wants interoperability and cross-platform file format.

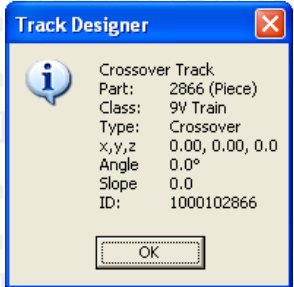
Since the source code and the author of Track Designer seems to have disappeared into cyberspace, all the chances to see compatible programs with Track Designer look totally lost. Though when I wrote my own LEGO® layout editor, I wanted to make it compatible with Track Designer. I knew that nobody would like to leave behind all their nice layouts produced with Track Designer. So I decided to dive into the TDL binary files to try to understand where and how this information is stored. The method used for this reverse engineering is quite simple but very long. I just create a simple Track Designer layout with only one part in it then I change one thing at a time (for example move the part, or change the part type, or add two parts) and save the layout in a different file. By comparing the two binary files and by knowing what I changed, I was able to locate the position of the information (like the type of the part, its position, its orientation, etc...)

I will now reveal what I have found (some parts of the format are still unknown to me but maybe some other people will continue to search) and hope that a lot of programmers will write new compatible tools. At first of course, I think of Cary Clark's TrackDraw, but maybe some-

one would like to write a stand alone converter between Track Designer and LDRAW for example (even if BlueBrick can already do this conversion in one way)?

Since Track Designer is a Windows program, the TDL files are stored in a binary format called Little Endian. The global structure of the TDL files is fortunately simple; there is a header of 124 bytes (generally used to store some global information) followed by several blocks of 106 bytes (except the last one which is only 104 bytes, but I don't know why), each block describing a part in the layout.

If you look at the properties of a part in Track Designer (by right-clicking on a part and choosing "Properties"), you will see the above window appear. All of this information will be found in the 106 bytes block. 



	Data	Type & Size	Comment	
Only once	Header	Unknown 124 bytes	I didn't search the details of the data stored in the header because I didn't need them, but I noticed that at the byte 68 there's the Instance ID (4 bytes) of the first part of the list. The header finishes with the string "CTrackPiece" which is probably the name of the class used to store the parts.	
	Part ID	int 4 bytes	This is the ID of the part (last line in the properties window). The high numeral is use to distinguish the class (2 nd line), whereas the low numeral correspond to the Lego® part number (1 st line).	
Multiplied by the number of Parts	Instance ID	int 4 bytes	Instance ID are in fact the pointer in memory that TrackDesigner save as an int, that's why you can see this number changing every time you reopen Track Designer and resave the layout. Track Designer use this instance ID to save the connection between the parts. Of course a remapping is done by Track Designer after loading a layout.	
	Angle	double 8 bytes	The angle of the part in degree (6 th line in the properties window)	
	X	double 8 bytes	X coordinates in stud unit (5 th line in the properties window).	
	Y	double 8 bytes	Y coordinates in stud unit (5 th line in the properties window).	
	Z	double 8 bytes	Z coordinates (5 th line in the properties window). I don't know exactly the unit, but the value is multiplied by 3 in the file, compared to the value displayed in the properties window.	
	Type	int 4 bytes	The type of the part (4 th line in the properties window), where: 0 = Straight 1 = Left Curve 2 = Right Curve 3 = Left Split 4 = Right Split 5 = Left Merge 6 = Right Merge 7 = Left Join 8 = Right Join 9 = Crossover 10 = T Junction 11 = Up Ramp 12 = Down Ramp 13 = Short Straight 14 = Short Left Curve In 15 = Short Right Curve In 16 = Short Left Curve Out 17 = Short Right Curve Out 18 = Left Reverse Switch 19 = Right Reverse Switch 20 = Custom	
	Geometry ID	int 4 bytes	Some part like the curve, T cross, or rail point may have several geometries (or picture) defined in the registry, this is the 0-based index of the geometry used.	
	Multiplied by 4 (1 per connection)	Instance ID	int 4 bytes	This is the instance ID (same kind as the second parameter of the part, so a pointer saved as an int) of the part connected to this part for this current connection.
		unknown	int 4 bytes	An unknown data for this connection.
		unknown	int 4 bytes	An unknown data for this connection.
	Unknown	int 4 bytes	An unknown data for this part. Seems linked with the support parts (the value is 3 for the support parts)	
	Slope	short 2 bytes	Slope of part (7 th line in the properties window). The value is multiplied by 10 in the file compared to the displayed value.	
	Unknown	short 2 bytes	An unknown data for this part.	
Unknown	short 2 bytes	An unknown data that is not present in the last part. Is it some padding data?		

FreeLUG - Enthusiasm First

Article and pictures by Didier Enjary

FreeLUG is the French LUG (LEGO Users Group). The story of the club started in February 2003 in the Paris area when early members met and decided to give official status and a website to the club in order to give it a national audience.

The club consists today of about 150 members. Approximately 50 of them are active members and participate in events and exhibitions of various sizes. The website is also pretty active as each member is encouraged to write reports, reviews and LEGO related articles. The communication medium for members consists of a chatroom and mailing lists. We do not use forums.

FreeLUG organizes two meetings on the national level each year. The first one in February is the opportunity for members to vote on how to use club expenses (about \$3000 mainly from the member fees) to pay for the website, insurance, mailings, meeting rooms and reduced price goodies (like Velux or Maersk sets). Of course this two day meeting is also the opportunity for members to exhibit their latest creations, to share ideas, and to make a collective shopping trip to the local toy store.

The second meeting in the summer is a private (members and friends only) exhibition. Each year, the location and the team organizing the event changes, allowing people from various regions to participate in the real life of the LUG. For those of you who are curious about French geography, the 2003 event took place in Rennes (Britanny), 2004 in Grenoble (the Alps),

2005 in Reims (Champagne), 2006 in Toulouse (home of Airbus company), and the 2007 event took place on our National Day (14th of July) near Cler-



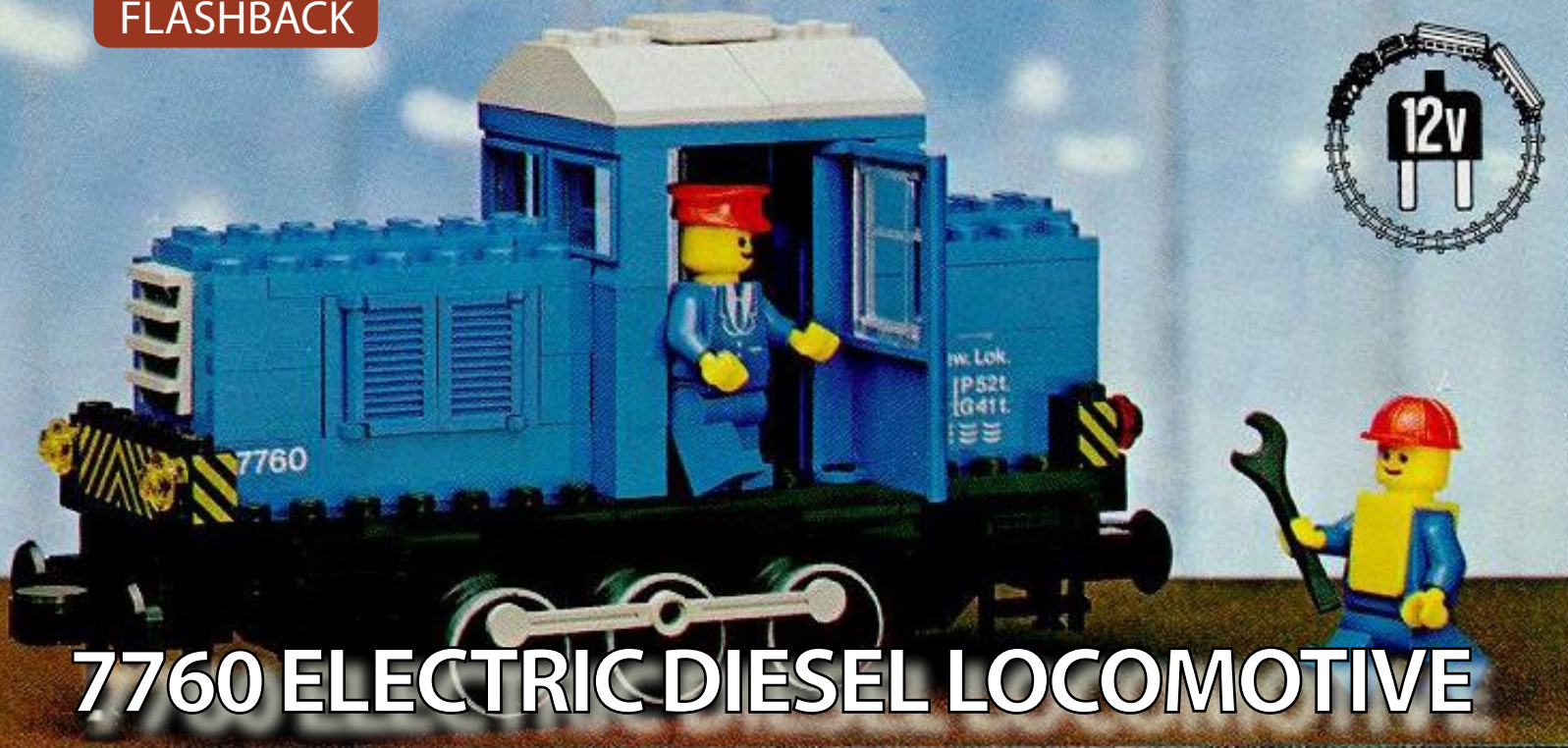
mont-Ferrand, Auvergne (home of Michelin, well known as a tire brand but also a historical producer of railroad engines called «Micheline»). The 2008 meeting will take place at Fana'briques in Alsace (which borders Germany).

During the year, on a local, national or international level (Dutch LEGO-WORLD, Italian LEGOFEST), members attend or participate in various AFOL events (about 30 for 2007). It is obvious that a lot of FreeLUG members are into LEGO Trains (FreeLUG is member of the French National Model Railroad-

ing Association), but almost all the themes are represented among the builders, from Mindstorms to mosaics. FreeLUG's mission is to gather all French speaking adult fans, no matter what themes they enjoy, and we are proud to welcome AFOLs from Belgium, Switzerland, Quebec, expatriated people and even friends from the US or the Netherlands.



<http://www.freelug.org> 



7760 ELECTRIC DIESEL LOCOMOTIVE

by Jordan Schwarz

It is ironic that one of the most exclusive and sought-after LEGO trains is a modest, smallish rendition of a shunting locomotive with a mere 153 pieces. With the introduction of the modern mini-figure in the late 1970s, LEGO ushered in the contemporary era of town and train sets. First offered in 1980, the little blue engine was among the first of the classic 12V trains.¹ With relatively few units produced, the exclusivity and elusive nature of 7760 has become its defining characteristic in the modern era.

The 7760 Electric Diesel Locomotive was officially available only in Europe, and it is perhaps the smallest LEGO rendition of a diesel locomotive. The first generation of LEGO trains was modeled after German prototypes, as Germany then constituted one of the largest markets for LEGO trains and toy trains in general, a trend still evident today. The physical inspiration for 7760 may come from the German Class 360 diesel-hydraulic shunter with a 640 hp / 478 kW engine. These units date back to the 1950s and were still operational with Deutsche Bahn in the year of this set's release.²

Accordingly, 7760 wore the marks of Deutsche Bahn (DB) as did the other LEGO trains of the era. It also featured a decal on the rear of the locomotive, reading "Gew.Lok. BR.{P52t. G41t.}", indicating the weights of the engine.

Speaking of weights, the set featured two train weights – 2 x 6 x 2 ballast bricks – in the unusual color of blue. Electrical pickups on the 12V trains differed from the system found on the 9V ones. Rather than using conducting wheels as the contacts, the 12V system used disc-shaped pickups on the bottom of the train motor which contacted the set of power-carrying metal rails. Locomotives using this system typically featured weights to ensure that the electrical contacts would firmly contact the metal rails. One advantage of the 9V system is that these weights are no longer required for proper operation, although they may offer an engine better traction.

The instructions for this set offered a piece of advice that might seem unusual given modern safety considerations: "Clean the rails regularly with a little methylated spirits." In essence, 9V rails are self-cleaning and this procedure has never been advised for those tracks.

The instructions also featured assembly procedures for a second model, a locomotive similar in design to the main model. Several track layout examples were also included, along with a count of the accessories required to construct each layout.

At first glance, 7760 may appear to be an unremarkable set. It was of a design typical of most small

shunters, with a low, long hood and a small cab at the rear. The lights on the locomotive were non-functional and did not feature the waveguides needed for use with working lights. The set featured only a few train-specific parts, though it did include a 12V train motor. However, the few parts specific to 7760 are rather elusive and expensive, making it difficult to produce an exact clone of the engine today.

The most unusual of these parts are the two front windows – with their 1 x 3 x 2 dimensions. These were found in white and red in many early LEGO sets, but 7760 is the only set to make use of them in blue. These windows give the engine its characteristic look and give builders looking to obtain a 7760 clone the greatest challenge. At the time of this writing, only two BrickLink stores offered these windows, at more than \$20 USD each!

The blue train weights are also exclusive to 7760. Other parts in the set were more common in the 1970s, such as the blue shutters over the engine compartment and the yellow safety vest worn by the mechanic, which has reappeared in recent years. The set also featured the 6-wheel train motor with working drive rods, another relatively rare item.


Why so much talk of building a clone of 7760 instead of buying the actual set? Set prices for this rare find are even higher, with sellers on BrickLink charging between \$175 and \$500 USD for this set in various degrees of used condition. The set commands similar prices on eBay. A single BrickLink seller offered the set new for \$3000 USD at the time this article was written. One may debate whether this asking price is fair for such a train set, but it is clear, in any case, that the Electric Diesel Locomotive is a very rare set indeed.

As evidenced by these figures, 7760 has become something of a legend in its own right. Many replicas have been inspired by it, some in other colors such as red and yellow. A faithful clone of 7760 can be made fairly cheaply in red due to the availability of the windows in that color. It is expensive, however, to build an exact replica in blue.

One builder, Erpelmutz, has taken a unique approach to this problem. His version of the famed blue engine,



pictured, updates the design with modern pieces and building techniques. A SNOT-built cab provides windows reminiscent of the original 1 x 3 x 2 frame windows at a fraction of the cost. Studless construction is used for the engine hood. The shutters covering the engine compartment have been replaced by modern container doors, and the locomotive includes a 9V motor. To achieve the look of the original three-axle design, the builder makes use of a set of wheels from Big Ben Bricks in-between the powered axles. Even the mini-figure on the engine has been updated to reflect the look of a modern LEGO train hustler. This approach shows a great way to update a favorite design of the past while saving money, compared to the cost of acquiring a copy of 7760.

The simple exterior of this small blue engine conceals one of the most famous LEGO train models of all time. Despite its outward simplicity, 7760 is inherently unique and notoriously elusive. It remains a favorite of many builders to this day. 

1: Inventory and set information was verified via peeron.com and the LUGNET set guide.

2: The website <http://www.einbahn.org/German/shunters.shtml> gives information, specifications, and history regarding the Class 360 and related classes of locomotives.



SWOOFY!

Interview by Benn Coifman

A casual scan of Brickshelf will turn up many great train builders. RAILBRICKS recently caught up with one of them, **Dara Norman**, also known as 'Swoofy' online.

RB: You are an accomplished LEGO builder but you also have an eye for detail on trains. Did your interest in trains start separately from LEGO or evolve from the LEGO train sets?

Swoofy: I got my first LEGO set when I was pretty young and built with LEGO all throughout my childhood. I also got a Lionel train set pretty young. I eventually had a huge HO layout and began kitbashing my own models. At some time in the 80's, my parents took a trip to Switzerland and they brought back my first LEGO train set (7710) and the infamous 7777 idea book. I was instantly hooked, but the problem was that the only train set offered in the US was the rather crude 7720 battery train. So I had an oval of track, 7 train wheels, six magnet couplers and an idea book that assured me that there was a huge LEGO train world out there that I had no access to. In 1990 I took a trip to Austria and I was

determined to bring back more trains, but that was the time of transition between 12v and 9v so the offerings were slim. After that I entered my dark years and nearly missed the 9v trains altogether. I had not thought of LEGO in years when a random internet search in 2002 revealed that the LEGO train world was as strong as ever and I was in a much better position to join it. I began collecting sets again (or amassing) and started building. I credit Ben Beneke for my re-inspiration because it was upon seeing his BR50 that I decided, "this is what I want to do." My favorite official LEGO sets are the Santa Fe sets, the BNSF GP and the 7750.

RB: With a handle like Swoofy, we just have to ask about the back-story.

Swoofy: The last 3 years of college I had become obsessed with wooden ship modeling. The last model I built in college was modeled after a racing sharpie named 'Swift'. In my part of the South, we say 'swooft' sometimes for swift, so my boat was named 'Swooft'. This just happened to coincide with the time Yahoo

started handing out free email addresses. Swoofly@yahoo.com was available and it stuck.

RB: You clearly strive for realism in your models, selecting specific railroads and equipment. How do you choose your topics?

Swoofly: Nostalgia led me originally. My childhood bedroom window looked out onto the mainline of the Louisville & Nashville railroad so my days were filled with those big gray locomotives. Most of my early models (post 2000) were L&N equipment that I remembered or had found in L&N books. I soon got tired of all black or all gray models (L&N colors) and began looking around for other equipment. Since I now live in Los Angeles, CA it seemed only fair that I should build the local lines, mostly UP and BNSF. Now I look at many sources for ideas: railpictures.net, railroad books and my own railfan pictures.

RB: What do you strive for in your design, just the trains, or the whole railroad environment?

Swoofly: I would love to get back to the 'whole railroad environment' of my HO days, but for now I model whatever I can store easily. My wife and I live on a 35' sailboat so a large layout is out of the question right now.

RB: Are you a member of a LUG or LTC? Do you ever show your works publicly? Where might someone see your work in person in the coming months?

Swoofly: Last year I joined the Southern California LEGO Train Club (SCLTC). The experience so far has been great. As a club, we have no permanent display anywhere, but from mid November to mid January we set



up at the San Diego Model Railroad Museum in Balboa Park. We also display at Fullerton Railroad Days in Fullerton, CA. Also this year the National Model Railroad Association (NMRA) convention will be in Anaheim, CA July 13 - 19 and SCLTC will be there.

RB: You have an active dialog going on with many other builders on Flickr, how has that experience gone?

Swoofly: When Brickshelf temporarily shut down last year, lots of people were exploring other options. I heard there was a group starting on Flickr so I checked it out. Turns out the LEGO trains group had been started by Tim Gould so I knew it was a good start. I like the fact that anyone can reply to your posts and you can describe your photos. Those two features alone give it a leg up on Brickshelf, but EVERYONE is a member of BS and that's an advantage too. The Flickr group is still quite small. The dialog available on Flickr is very helpful at times. We can encourage each other, make suggestions, ask for help or just poke fun at each other.

RB: Could you tell the readers a little bit about your design process? How do you start, do you sketch things out in CAD or go straight to the bricks? Where do you find inspiration?





Swoofy: I start by looking at train pictures. I visit rail-pictures.net and similar sites almost daily because hundreds of new pictures are posted every day. Something will eventually grab me and I'll look into it further. I check the wiki entry to find out its history and then try to find some good pictures of the subject. I usually have 4 or 5 models in planning at a time and eventually one will strike me and building will begin. Some models have been on the drawing board for years but I keep thinking about them. When I start a build I draw a side-on view of the model to work out the proportions. This is the most important step for me. I use the unscientific formula of $1.6 / \text{actual length (in feet)} = \text{stud length}$. This allows me to work out the proportions before I start building. From this drawing I'll try to build the model in my head as much as I can to figure out the pitfalls. Sometimes I'll build in MLCad first, sometimes with MLCad and bricks, and sometimes without the computer. I can use my laptop at work to build so it depends on free time too, but I prefer building with bricks when I can. After the first build, I make sure it works on LEGO track (points, curves and crossovers) and check it against the reference photos to make sure

it all looks right. Then comes the tweaking; trying different details to see if they work better, different colors here and there if necessary. Eventually I'm satisfied and add stickers. Creating stickers is currently my weakest suit, but it does signal completion.

RB: Where do you place the greatest weight in your designs, the specific piece of equipment or the whole train?


Swoofy: For most of my creations the specific piece of equipment is the focus. Engines are my favorites. Even back in my HO days I always seemed to have more engines than railcars. The one exception so far would be my L&N Humming Bird passenger train set. The E7 locomotive and the passenger cars needed to match so the whole train was the emphasis there. It's very satisfying to see it run (unfortunately not on the boat!), but it's quite heavy. Having a small space to display also makes me concentrate on engines, because I just can't run long trains. Once I joined SCLTC I did begin building lots of rolling stock for display, but engines are still my primary focus.



RB: Are any of your models ever “finished” or do you continually go back and rework them?

Swoofy: YES! I’m happy to say that at least 5 of my models are finished! It is hard, I agree, to let some models go, but some I do feel completely satisfied with. Some of the ‘incomplete’ status comes from the lack of LEGO in a certain color (hello dark blue, dark green and dark red?), but that also adds to the challenge and that’s the real reason I build. I’ve recently found that after running a model for sometimes days in a row, a flaw in the design will emerge and it will have to be reworked. Also sometimes I learn or figure out a new technique and want to incorporate it into an earlier model. My ‘finished’ models include: CSX SD80MAC, UP 3GS21B, BNSF/CSX B40-8W and GP60B.

RB: The theme of this issue is tips and tricks. What are one or two of your favorite building tricks that you think any LEGO train builder should know?

Swoofy: Know your Brick Math! Kim Toll’s great PDF in the ILTCO library is a wonderful resource. I like to work out problems in my head (i.e. at work) so knowing the SNOT relationships is a great boon. Also, keep a building notebook. It can be an MLCAD file, LDD file or real paper, but whenever you have a detail idea, write it down. It may not have anything to do with a current project, but it makes a great reference. It’s also useful for working out how other builders have solved a problem. 

Links for more of Dara’s work:

Personal site (coming soon):

<http://RailroadBricksmith.com>

Flickr: <http://www.flickr.com/photos/swoofy/>
Brickshelf:

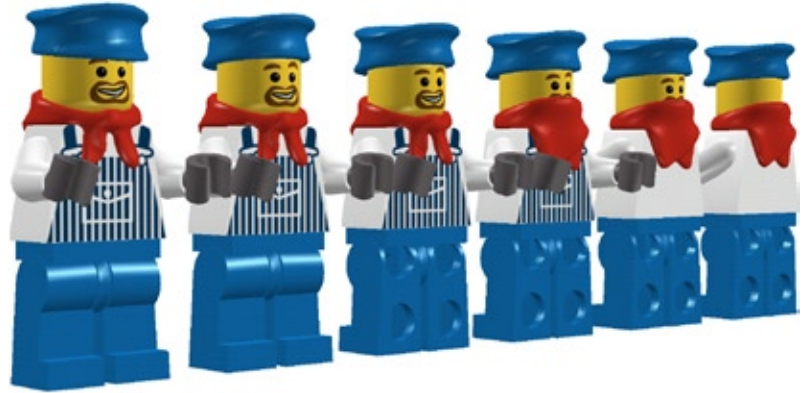
<http://www.brickshelf.com/cgi-bin/gallery.cgi?m=swoofy>

SCLTC: <http://www.SCLTC.org>



Reverse Engineering Challenge 3

by Benn Coifman



This column seeks to challenge readers to look around at other builders' work and tease out how they achieved a specific effect, an important skill as you wander off the instruction sheet and into your own creations. Continuing the recent trend focusing on windows, this issue we move to the end of the train. Here we see a blunt end observation car, designed so that it could both be run at the end of the train allowing passengers to see the track receding as the train speeds along, and also giving the railroad flexibility to put the car in the middle of a train while allowing passage. In other words, the diaphragm is flanked by two windows.

In very close quarters this car includes both side facing windows and rear facing windows, as well as the diaphragm with its own window, all in six wide. These



features could just as well be for the cab windows on a subway train or commuter train.

Submit your solution to challenge@railbricks.com with the title **THIRD REVERSE ENGINEERING CHALLENGE** in either LDraw format or provide sufficient digital photos on how to construct the car by August 1st, 2008. If you build a physical model, you can use more common colors. Be sure to include your name and contact information.

The editorial staff will select the best design from all of the buildable submissions that achieve this effect and winner will receive a "RAILBRICKS Challenge" engraved brick. We'll publish the solution in the next issue.

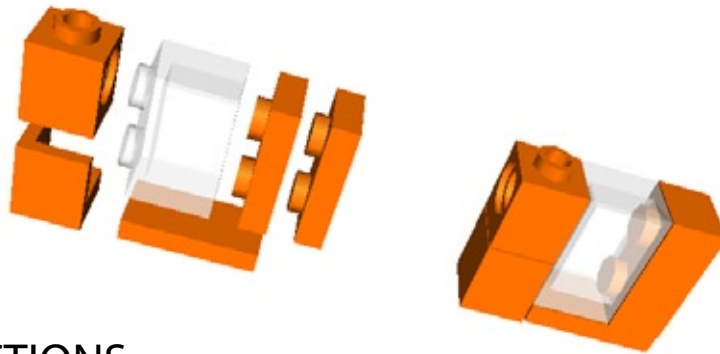
All submissions become the property of RAILBRICKS and by submitting an entry you will allow us to print your submission in whole or in part.

If you have ideas or suggestions for future challenges, contact us at submissions@railbricks.com.

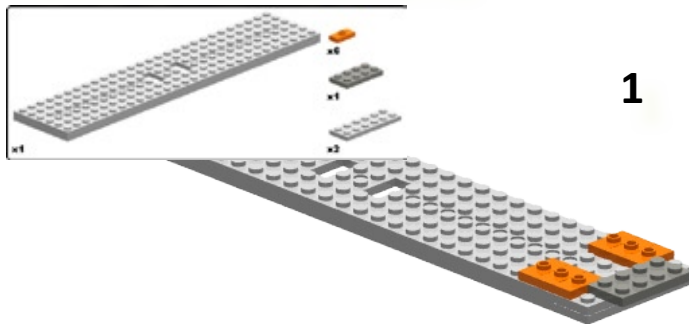
Reverse Engineering Challenge 2 REVEAL

This reveal shows a lot going on behind the scenes. At the time the design was first built, orange 1x3 plates and dark green 1x1 bricks were impossible to find, so some improvising was needed. They were replaced with two wide parts, but those ran into the conventional diaphragm on the end of the car. So instead of using dark gray 1x2 for the vertical components, this design uses 1x1. While dark green 1x3 bricks were available, they were more expensive than the wedges. Note how the side door windows are not actually attached but they are held in place by the other bricks. Another feature of this design is the use of the 1x2 panel instead of a 1x2 tile, eliminating the seam that would otherwise be present. You still cannot have every part in dark green; I used black jumper plates at the top of the door. These black pieces could be replaced with dark green tiles for the purist.

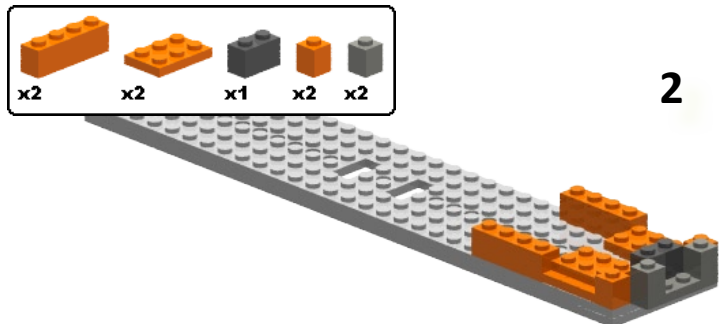
We had several great entries, and for this challenge the engraved brick goes to Karlong Chan. He came up with a great way of pinning the door windows in, shown below. Unfortunately, in reality the second stud on the 1x2 hits the bottom of the technic brick, but flipping the window around, using a technic pin to hold it in place and substituting in a tile, can achieve nearly the same effect. Congratulations Karlong!



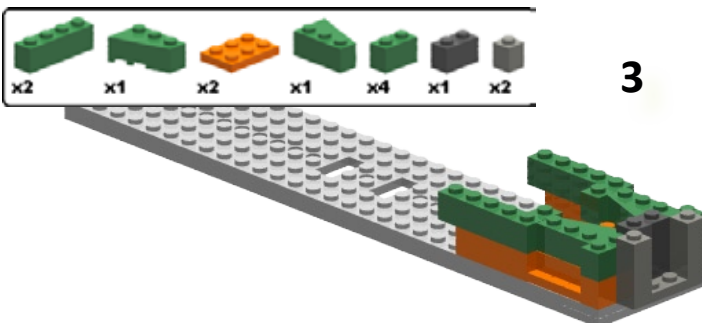
REVEAL INSTRUCTIONS



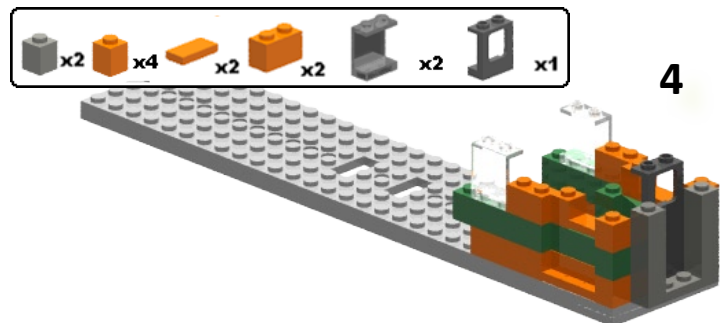
1



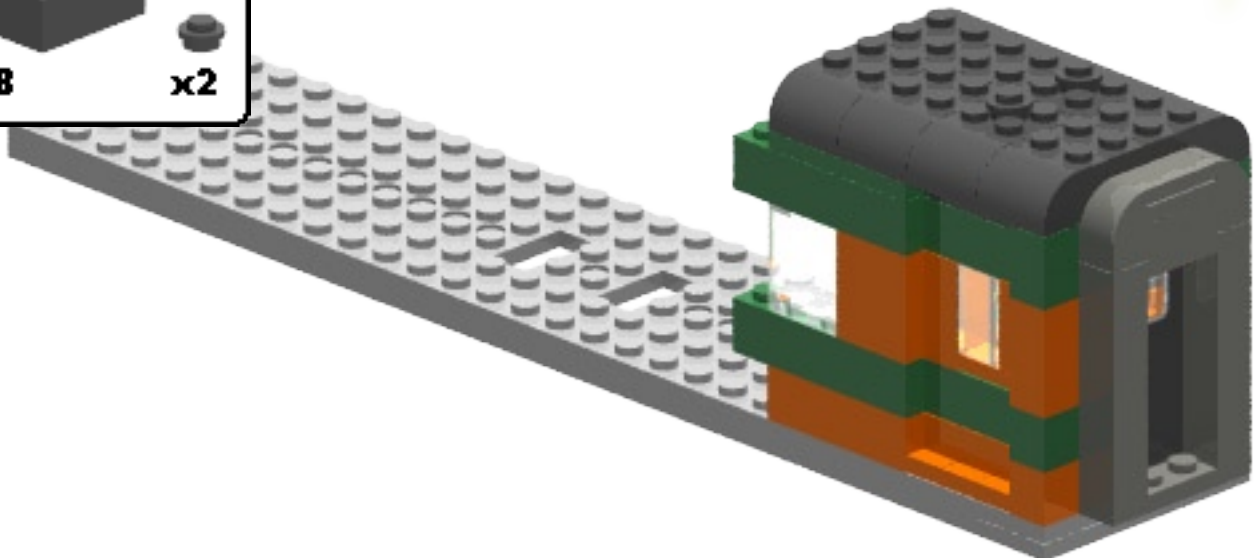
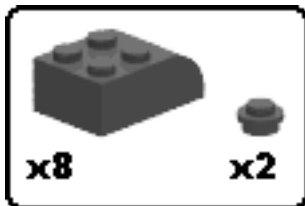
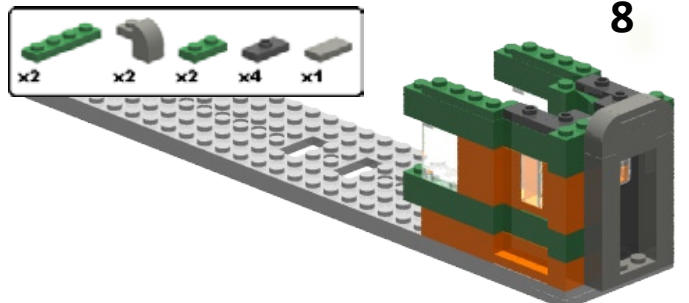
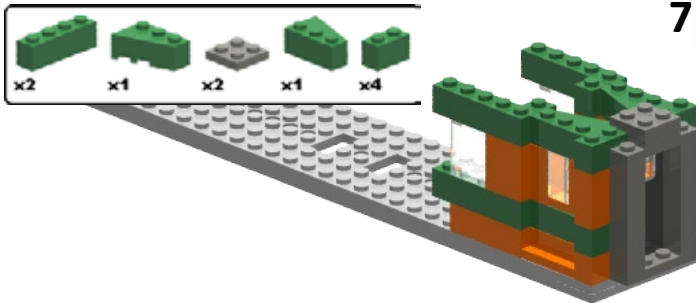
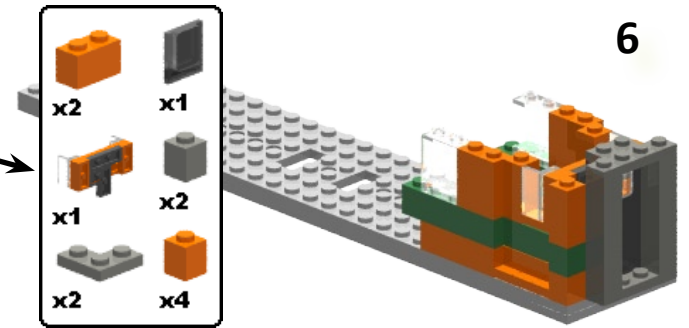
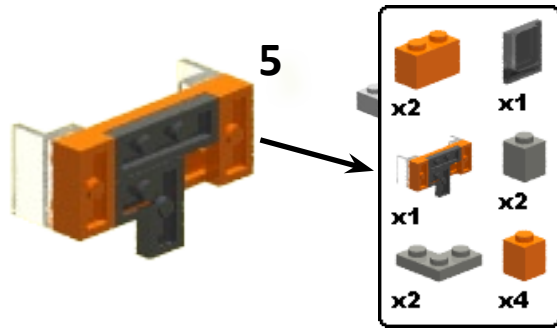
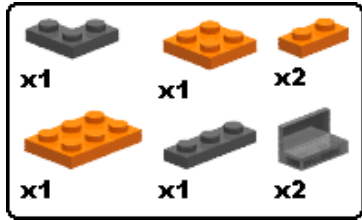
2



3



4



POWER FUNCTIONS

In the last issue, the Power Function (Pf) and Pf train elements were described. This time we'll have a look at a very interesting combination of Pf and 9v train elements to achieve some fun eye candy. In a normal Pf train configuration you have a 9v battery box, a Pf IR receiver, and Batt train train-motor; with plastic wheels (shown at right), all running on a plastic track layout. If the Batt train train-motor is replaced with a standard 9v train motor; with metal wheels, an interesting side effect happens.


Following the electrical flow; it starts at the battery box, then routed by the IR receiver to the train motor. When using the Batt train train-motor with plastic wheels the flow stops at the motor, but if a standard 9v train motor with metal wheels is used the flow stops at the wheels. If those wheels happen to be on a piece of metal track then the flow continues into the track! If metal track regions were strategically placed around the layout and power leads attached to them, when the train passes over the metal track, that particular power lead is energized by the train! These metal regions are not limited to only one section of track; they can be as long as needed!

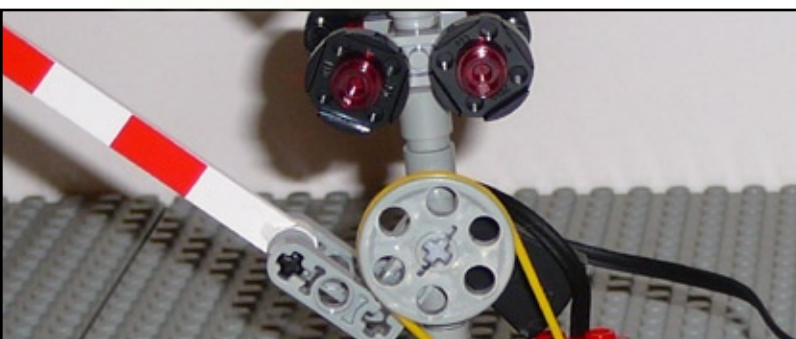
+ 9v TRAIN TRICKS

by Steve Barile



Some examples: crossing gate lights that blink (old LEGO emergency vehicle blinking lights) when the train passes or crossing gates that use motors to lower them and gravity to raise; cows moving their heads (originally achieved via Mindstorms: NGLTC) as the train goes by; an animated bridge; a pneumatic pump system... or as simple as a wind mill that spins or park swings and merry-go-round motion while the train goes through that region of the layout. One clever idea is to connect a region of metal track via two power leads to an independent metal track with a 9v train trolley on it. And so on...

With the vast number of creative train AFOLs out there I'm sure this is only the tip of the iceberg! Feel free to send a letter to the editor (with pics) about some cool ideas that you happen upon! 



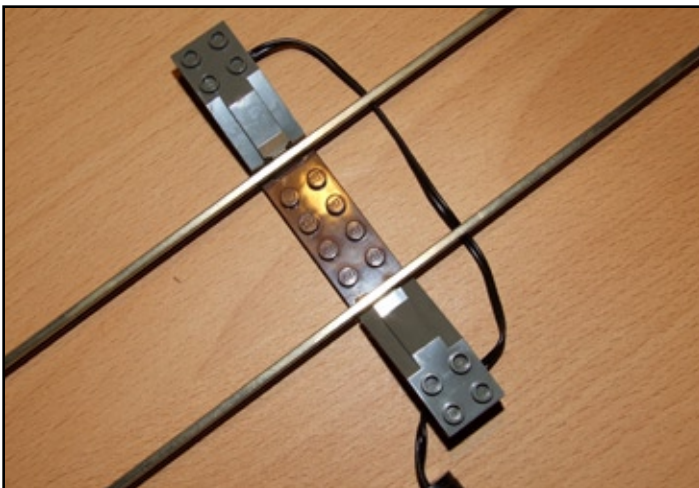
by Jan 'Grunneger' van Dijken

My name is Jan van Dijken, I'm 37 years old and living in Stadskanaal, the Netherlands.

I've been a real LEGO-fan since my youth, but really started modelling in 2006.

The reasons were the marvellous trains made by Reinhard "Ben" Beneke and Gerben "Puntcom" Zuurveld. I'm working on a small fictional German layout, situated in the 1960's. My buildings are based on H0-models from Faller, Vollmer and Kibri.

Building my trains 7-wide (to me, a more realistic scale) gave me one big problem: the track. In my opinion, the standard LEGO-curves and switches are too sharp for the 7-wide engines that I have. So I searched for an alternative and found it in Ken Rice's flexible track. Ken made his track with 0-gauge rail from Atlas, but these were too small for my taste. I prefer 1-scale track from Peco (<http://www.peco-uk.com/Products/pecoproducts.htm>). 1-scale track has the same height as standard LEGO-track and it's compatible with the standard train contact from the 9volt series as shown in the picture below.



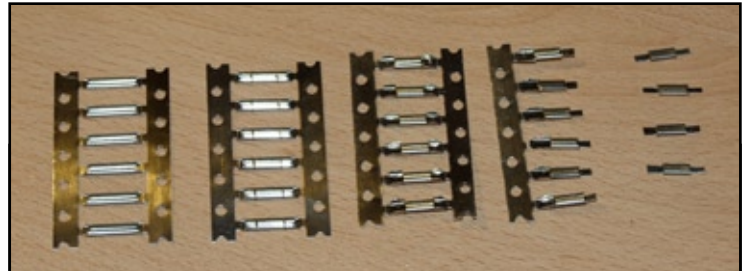
I STARTED WORKING...

The most difficult task was adapting the railjoiners. I had to make them one by one...

I use a small "Dremel" (a hand held drill) combined with the smallest cutting blades from Proxxon. These are the best suited for cutting the joiners. I also use small needle nose pliers, a small cutting plier and a knife.

It's not a very cheap way to produce tracks, at least

not in the Netherlands. I have to import the track and joiners from Britain and the cutting blades only last long enough to make 50 to 100 joiners, if you're lucky. One miss and the blade is useless. You also have to fix the joiners very tightly to a table. It's not possible to hold the joiners with one hand while sawing with the other. The end result however, is worth every penny as far as I'm concerned.



The steps to creating the sleepers are outlined below.

- 1: Cut the joiners from the metal sprue. The cuts should measure roughly 2 studs apart.
- 2: Bend the excess metal near the tabs flat and cut with a set of plier cutters. It may be enough to bend them until they break.
- 3: Bend the end tabs down at an angle of 90 degrees.
- 4: Cut the studs off of the 2nd and 7th row of a 2x8 plate. I use the reddish brown plates, because I feel that it is a more realistic look when attached to the Peco rail.
- 5: Put the joiner tabs in the holes and bend them flush with the bottom.
- 6: Repeat this as many times as your thumb can handle it.



Always tape your thumb before working because it will be very painful if you don't. I've managed to make 50 sleepers in one evening but then had to stop for a week for my thumb to recover. Once the sleepers are done, simply slide the rails into the joiners, adding additional sleepers as needed.

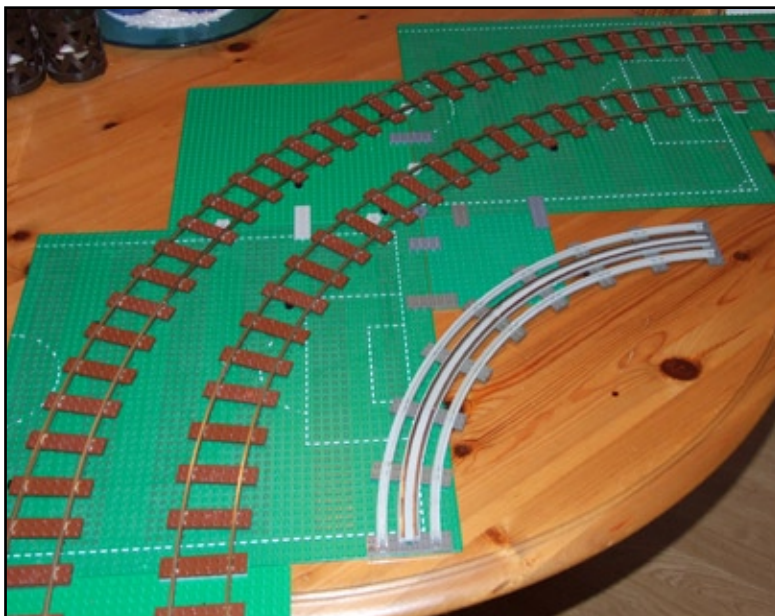


The straight track above is the same length as 7 LEGO-tracks. It contains 23 sleepers; 12 with the metal joiners and 11 without. It's not necessary to use 23 "joiner-sleepers" in a straight track and your thumb will appreciate it.

The curve track is another story...

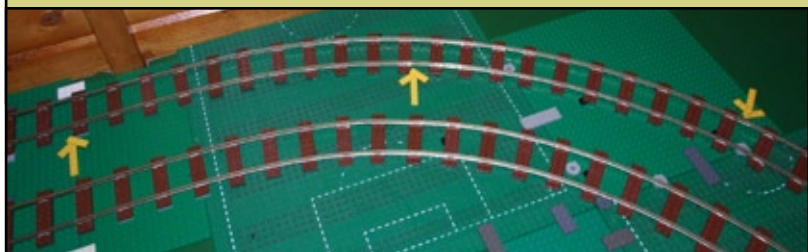
The basics with a curve are the same, but every sleeper must be connected to the rail (using the metal joiners) to ensure a nice smooth looking curve.

The picture below is of the finished curves together with a standard LEGO-curve. The difference is obvious. I use the football baseplates because of their size and stiffness. The bigger the baseplate the better the curve.



TO MAKE A CURVE....


- 1: Join the baseplates together and draw a circle with a large compass. I use the same one teachers use in class to draw circles on a chalkboard. This line will serve as your guideline for the curve.
- 2: Start with three sleepers aligned in a straight line at the end to ensure a good connection when attached to the straight track. Without this leading straight section of the curve, there is too much tension and it will be difficult to attach to a straight section. Cut one length of rail in half. Use one half on the inside and one on the outside of the curve. Slide a full length rail and the half rail through the three straight sleepers as far as possible. The rails should be situated as shown in the figure below. Slide every sleeper onto the rail until you reach the end of the short rail. Take another full length of rail, connect it with a joiner to the short rail and continue putting sleepers until you reach the end of the curve.
- 3: Pin the sleepers as much as possible to the baseplates with 1x2 jumper plates, 1x1 plates, or 2x2 turntable plates. Cut off the excess rail and you're finished!



It looks simple and it is once you get the hang of it.

That's the basic idea for making these tracks. I used reddish brown plates and put them 3 studs apart from each other. It's possible to combine these homemade tracks with standard LEGO track. View more pictures of this and other projects:

<http://www.brickshelf.com/cgi-bin/gallery.cgi?m=Grunneger>

I'm currently working on a switch, but this is still in its rough stage. I'll put the pictures in my BS-folder the moment it's ready. 

Editor note: View Ken Rice's flex track project at his website:

<http://users.erols.com/kennrice/flextrack.htm>

ADVANCED 9v TRACK MODIFICATION

by Ondrew Hartigan

Since the 9v LEGO train line was first introduced, some AFOLs have desired a set of switches (points if you prefer) that do something other than just make a pair-well siding. For well over 5 years now I have been doing just that, so today I am going to show you how to make the most useful, in my opinion, modified switch.



TOOLS

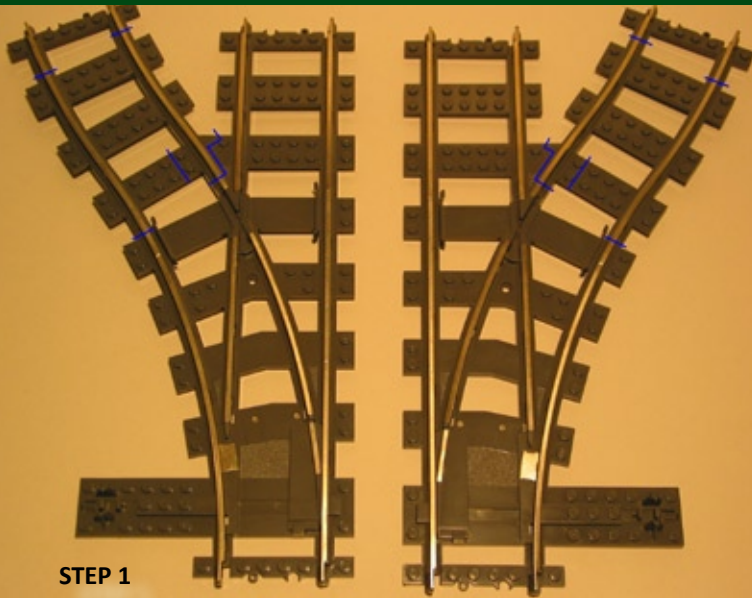
- Small flathead Screwdriver
- Small needle nose Pliers
- X-ACTO knife set
- Razor saw, superfine teeth 52 TPI,
- Ultra thin blade .008 Zona Brand <http://www.zonatool.com>
- Glue—I use ZAP-A-GAP CA+ made by Pacer Technology. In testing of 5 different glues, this is the only glue that I found that the plastic would rip before the glue joint would.
- (1) 1x4 LEGO brick to cut up
- (1) 2x2 LEGO plate to cut up

“Stubby switches”, as I have dubbed them, were not invented by me, but you could say I perfected the process for making them. As of today I have made close to 100 switches with the very same process outlined over the next few pages.

Note: This modification can be done to both 9v and RC track.

STEP 1

Select one left and one right switch track and equivalent track to use as a jig. In this article we will make a pair of stubby switches.



STEP 1

Notice the blue lines; this is only a general visual of the cut lines to be made later. However, as Mark mentioned in *Track Modification 101* (issue 2), "it is beneficial for complicated modifications to layout the cuts before you do anything."

STEP 2

Remove the Metal Rails

Using a small flat head screwdriver, gently pry back the tabs as marked, on both switches. Gently pull off all 4 rails and set aside in their matching pairs. Note that it might be wise to mark from which switch the rails came. You should not have to force the rails off the track, and be sure not to bend the rails, since you will need a portion of these later.

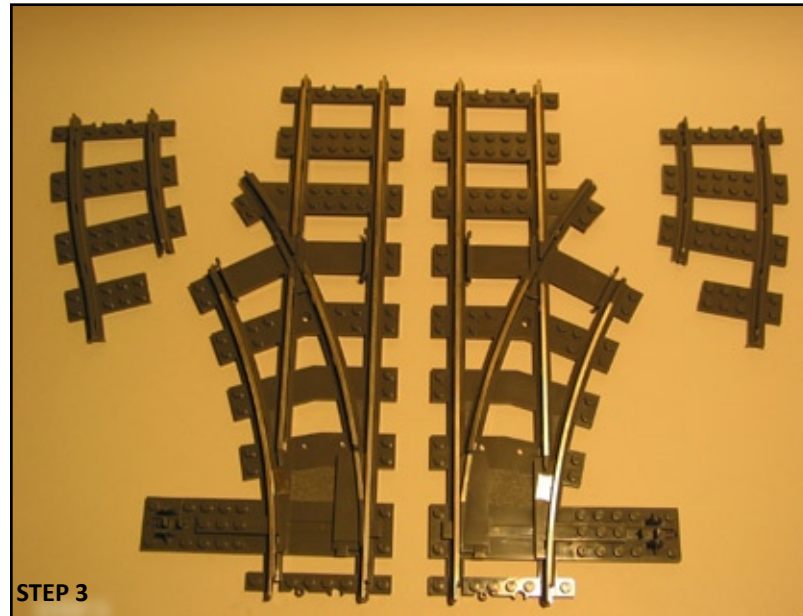


STEP 2

STEP 3

Cutting the Track

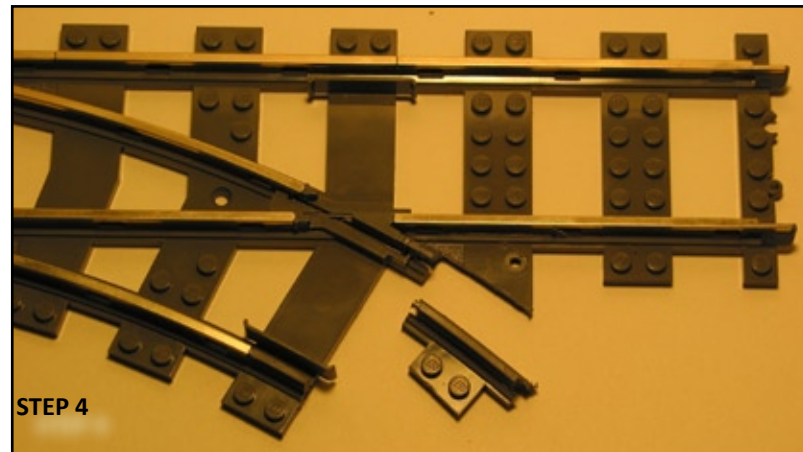
Using a saw or an X-ACTO knife, cut both switches as shown. Take your time on the outside rail and do not cut the wheel guide. Use a larger saw blade and cut it vertically to avoid damage to the wheel guide.



STEP 3

STEP 4

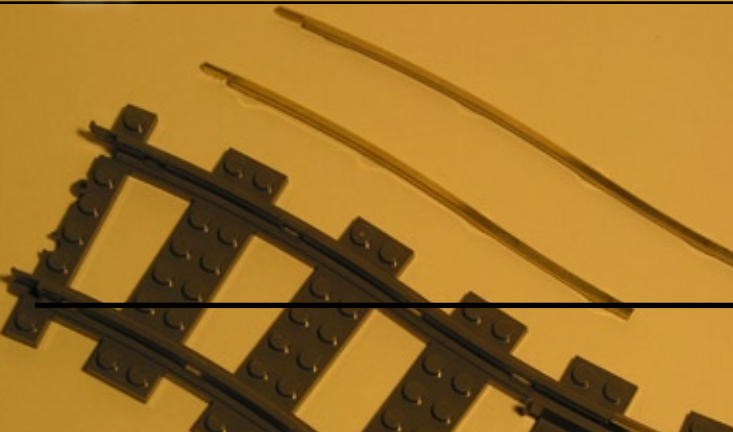
Trim the inside rails as shown. Be sure to not cut away too much as this could cause assembly issues in the remaining steps.



STEP 4

STEP 5 (Picture on the next page)

Using curve track, build an assembly jig as shown. This will work for the right hand switch. To do the left, make a mirror image of the jig.





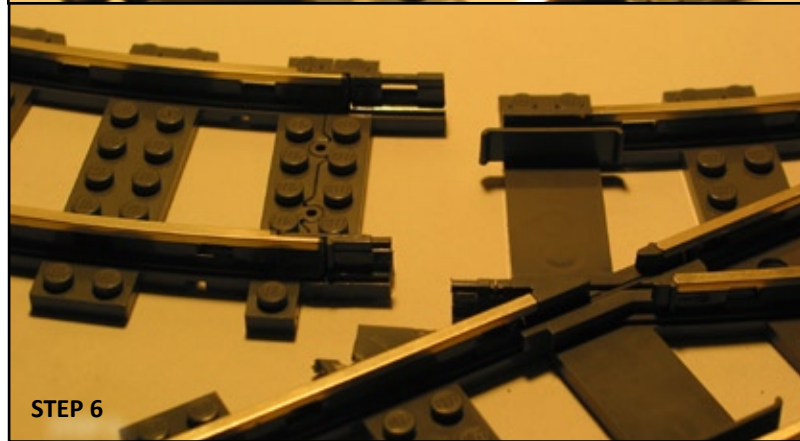
STEP 5

STEP 6

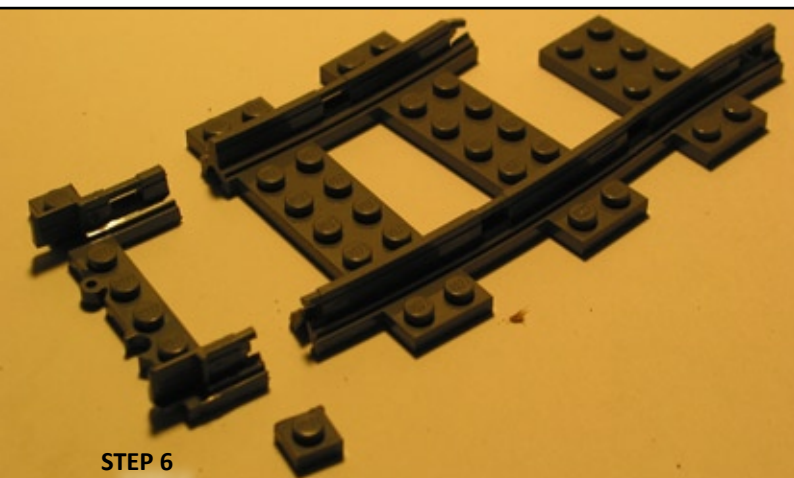
Install both the right switch and the cut off section from the left hand switch. Using an X-ACTO blade mark the cut off section where it lines up with the switch. Remove the cut off section and cut at the lines you marked. Reinstall in the jig to ensure a proper alignment. Your cut ends should line up perfectly. If they don't, clean up the edges so they meet. (A file works great but it can be done with an X-ACTO knife) When the piece fits perfectly remove it and then reinstall the piece on the jig with an extra piece of track attached. Using this will give you something to scribe against to mark the switch for a notch. Be sure to make both the left and the right switch before going on to step 7!



STEP 6



STEP 6

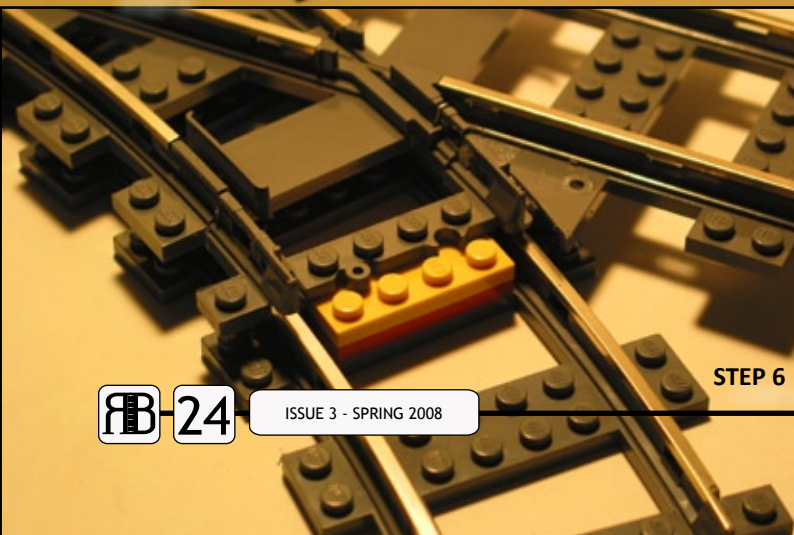


STEP 6

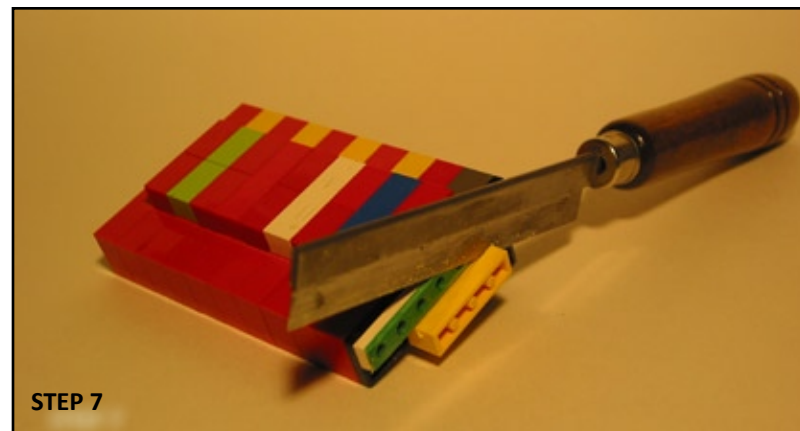
STEP 7

Cutting Shims

Using a Zona saw and a cutting jig like the one shown, cut shims from a 1x4 brick. I'm using yellow but any color will work. Make sure it's not a clone brick, as they do not bond nearly as well as LEGO to LEGO. Instructions for the cutting jig shown are available here: <http://railbricks.com/instructions/shim.pdf>. Attach a 2x2 plate to a brickplate or baseplate and cut off the studs. Remove the plate and cut it in half corner-to-corner with your X-ACTO knife.

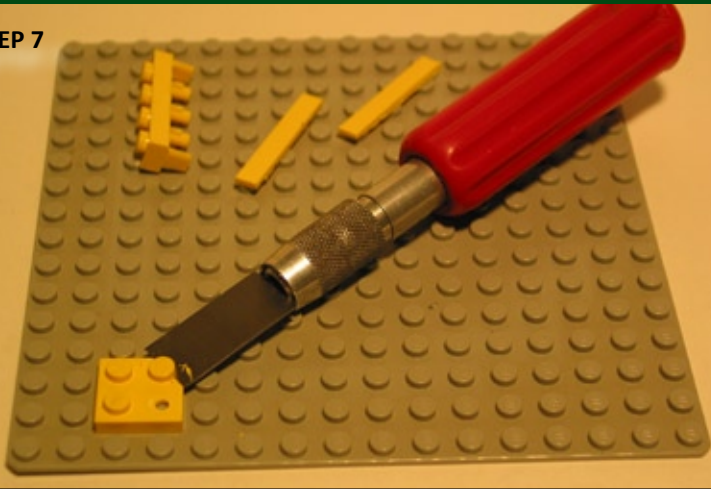


STEP 6



STEP 7

STEP 7



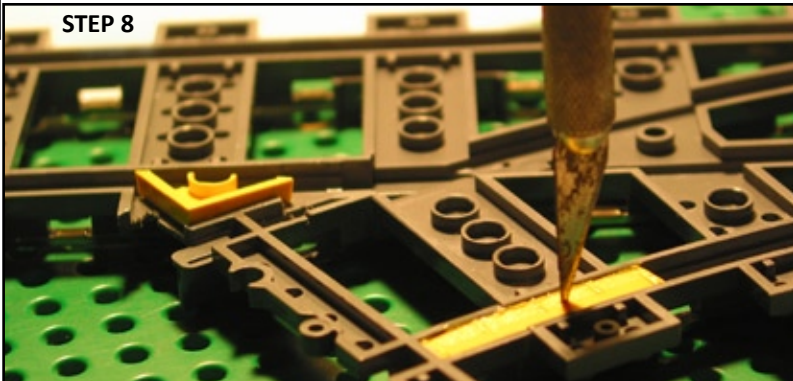
STEP 8

Gluing It All Together

Flip both switches upside down and double check fit. Using ZAP-A-GAP, glue the piece to the switch then glue in your shims as shown. Be sure not to touch the glue, as it doesn't come off easily. Typically I just poke the shim with my X-ACTO blade and use it as a handle while gluing. A small screwdriver may help you manipulate the glue-covered parts as well.

IMPORTANT: Let the glue dry for at least 2 hours before moving to step 9!

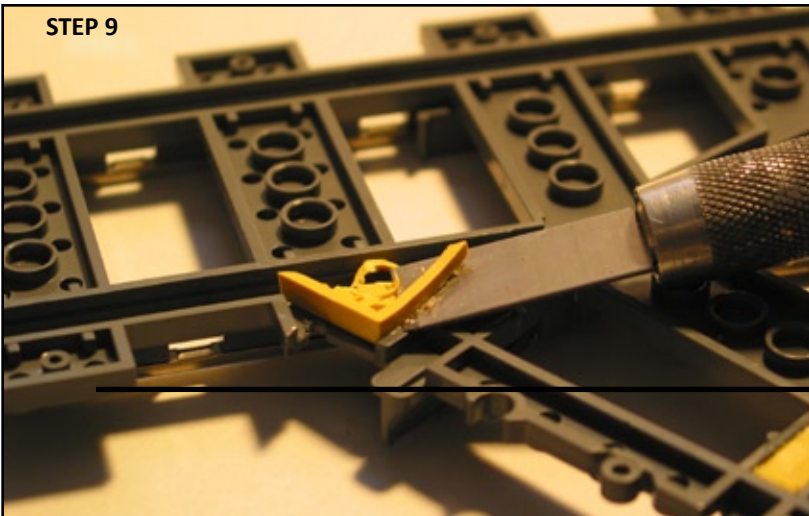
STEP 8



STEP 9

Use a flat bladed X-ACTO knife and cut off the excess that sticks out on the triangular shim. If you are modifying RC "plastic" track you are now done.

STEP 9

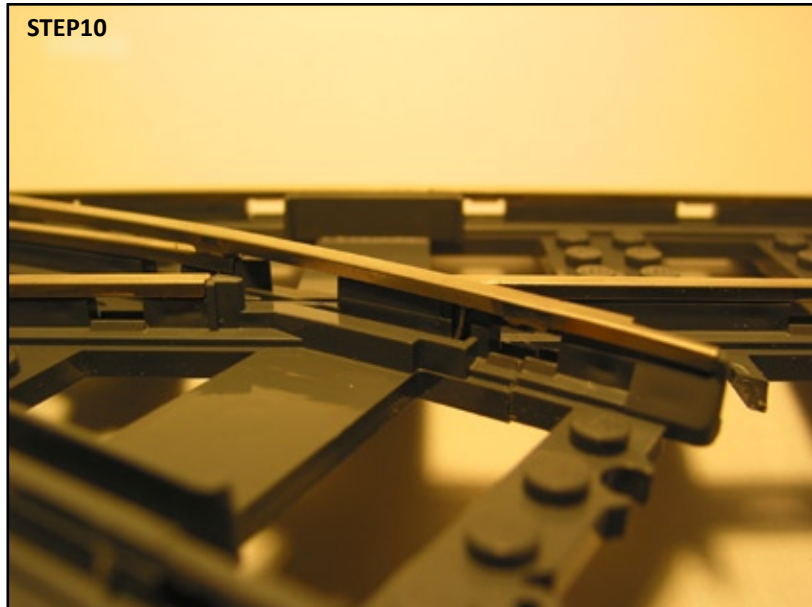


STEP 10

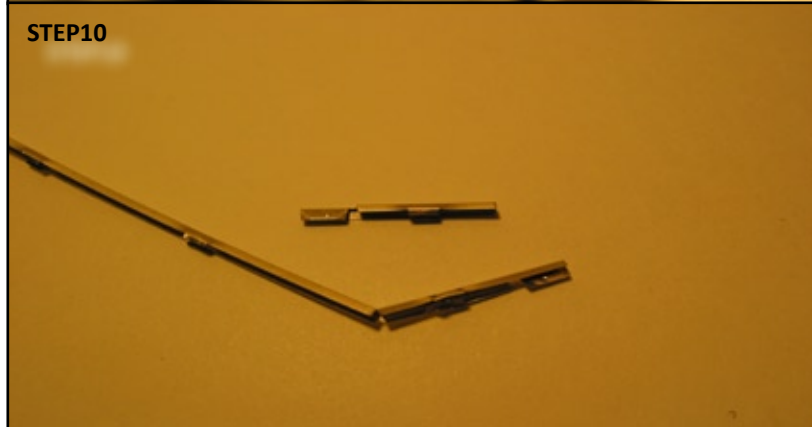
Fitting The Rails

Reattach the opposite rails on each switch, i.e., the left hand rails go on the right hand switch and vice versa. Note the inside and outside rails also switch places so that the mounting tabs align properly with the holes in the track. Set the inside rail in place where it should go. At this point the rail is way too long, so take your X-ACTO knife and scratch a mark on the metal rail where it lines up with the indent by the frog. Remove the rail, cut the sides with your rail nippers and snap off the excess. It should now fit. If it is slightly long, use a file or Dremel tool. Use your needle nose pliers to pinch the tabs back into place.

STEP10



STEP10



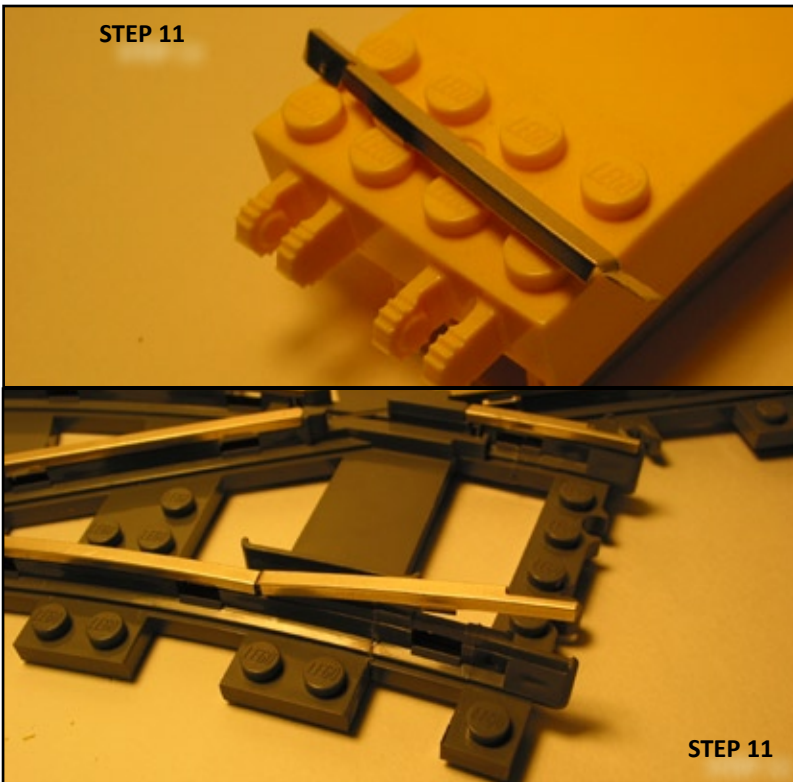
STEP 11

Fitting The Rails Continued

Fitting the outside rails is a little different. Like the inside rails you should set the rail in place and make a mark where it lines up with the other metal rail. Remove the metal and cut as you did in step 10, only this time a 1/4" past your mark, and snap it off, making the piece slightly long. Next cut the sides of the rail in line with your mark and snap off the sides only. This will give you a little flap you can place under the other metal rail to complete the circuit. You may need to narrow this flap just slightly to get a good fit. I use a dremel tool with an abrasive blade for this.

Install the rail by first sliding the tab under the other metal rail then pinch the tabs back into place.


STEP 11



STEP 12

Testing For Conductivity


Test your track to make sure it works. I've found that the inside rail sometimes fails to conduct. To fix this take a little screwdriver and bend the little metal wire that jumps the frog out just slightly. You are now finished!

Important! You should use extra care when handling your modified track. Typically they are more fragile than unmodified track. 

B

USE THE WORLD AS YOUR INSPIRATION

by Steve Barile

A great modeling tool is photographs. Google has recently started a new feature of Google Maps called "Street View". Just go to Google, click on Maps, and then click on the "Street View" button located along the top of the street map. The streets that are photographed are outlined in **blue**. You can then click anywhere along that street and a window will popup and show a street level photograph of that location. You can traverse the street, rotate and tilt, and even zoom! Below is an example of this looking at a downtown Portland street corner. Portland happens to have a ton of streets photographed and many great buildings and street level features to model. Enjoy surfing and building! 



Drag the yellow figure onto any blue marked street to activate the Street View



Pan, zoom, and navigate the city at street level. Tilt to view the details at the top of buildings.



A BRIDGE TOO FAR: BUILDING REALISTIC WOODEN TRESTLES

by Jeremy Spurgeon

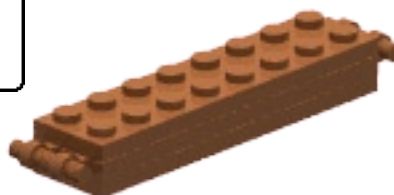
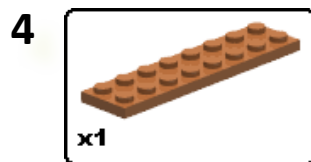
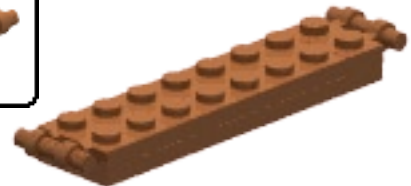
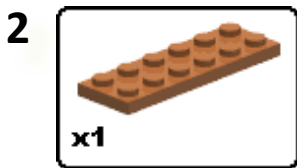
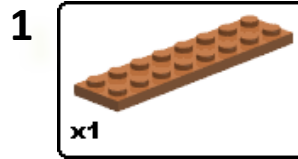
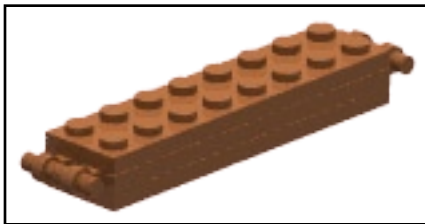
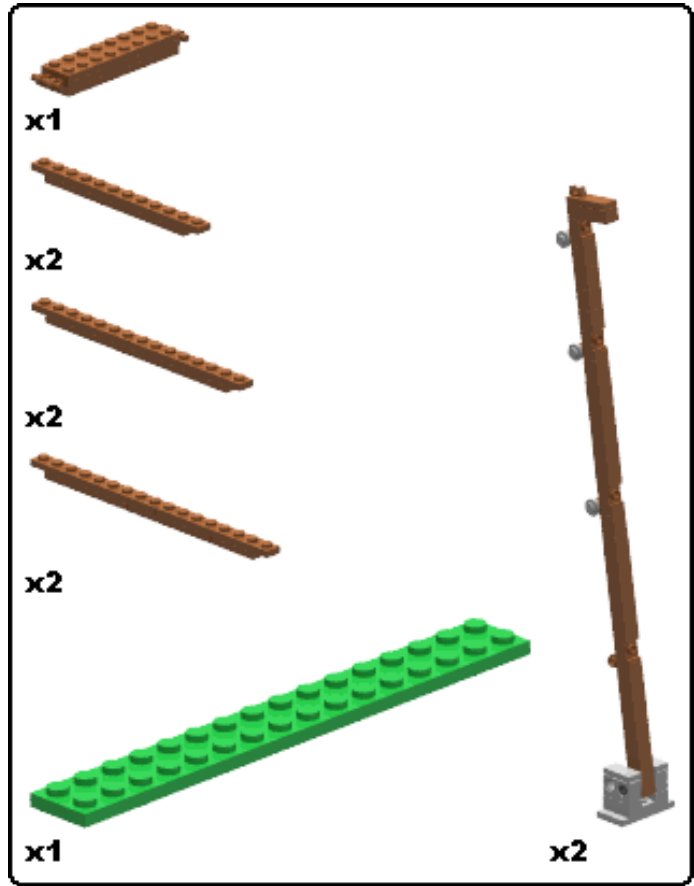
Back in February of 2005, I was inspired to begin a bridge project based on a local O scale club's curved trestle bridge. The idea was born within the preparations for the 2005 NMRA National Train Show in Cincinnati, while I was working on a section to compliment Brian Darrow's Mountain (see RAILBRICKS issue 2). The O scale bridge can be seen to the right.

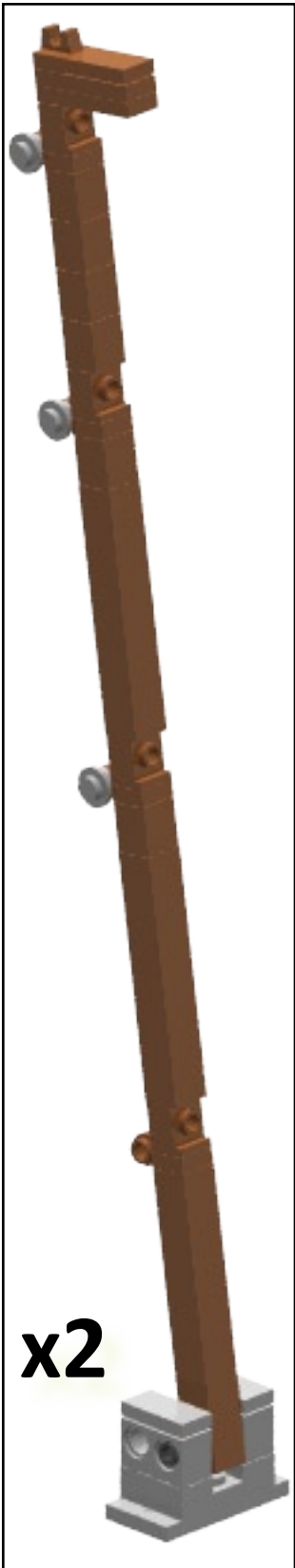
I had seen other LEGO trestle bridges in the past, with PNLTC's being one that stuck out in my mind as an excellent example. I was, however, aiming for a thinner planked look and wanted to make my bridge as close to the O scaler's bridge as possible. I went through many support designs, struggling mostly with getting the legs to attach at an angle, yet still be solid enough to support even the heaviest of trains. Within the next few pages, you'll see the design that I settled upon, which proved to be stronger than I could have imagined the end result could be.

Using the 1x1 plate with light clip, I created a cross brace system that works very well. The key to building this bridge lies in using the 1x1x5 brick on the legs, which helps to strengthen their height; whereas 1x1 bricks stacked to the same height would create a weaker leg. The legs must be offset by 1/2 stud on the baseplate because of the connection between the 1x1 clip at the top of the leg and the 1x2 plate with bar on the track brace creates the same offset. The feet of the legs should be spaces 16 studs apart. The in-

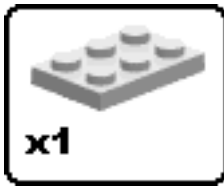


structions include a 2x16 plate as an example. Each brace is attached directly to every other sleeper on the track. Because the entire leg support is independent, creating a curved bridge is easy. The tiled bracing on the sides of the bridge will apply the final stiffening to the wobbly construction. This design can be applied to wider bridges with a little modification of the cross brace geometry. The top deck of the bridge can be left as is, or covered by 4x8 plates as shown in the finished example. The final detailing can include tiling the entire deck and adding railing along the sides, though most prototypical bridges do not have the railing. Most of the parts can be found fairly easily on Bricklink, however, the 1x1 plate with light clip in brown has gone up in price considerably.





1



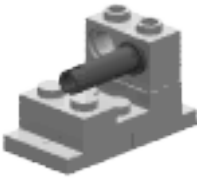
2



3



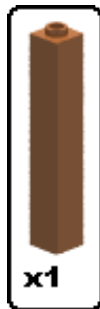
4



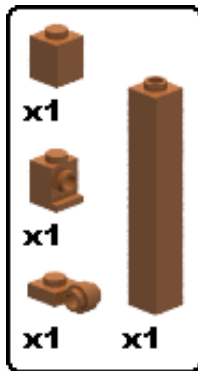
5



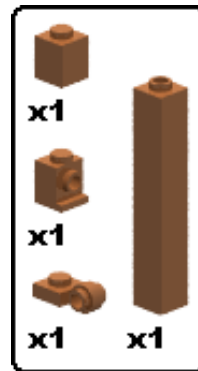
6



7

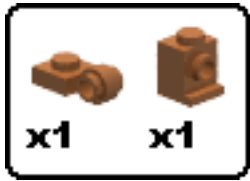


8



9





10



11

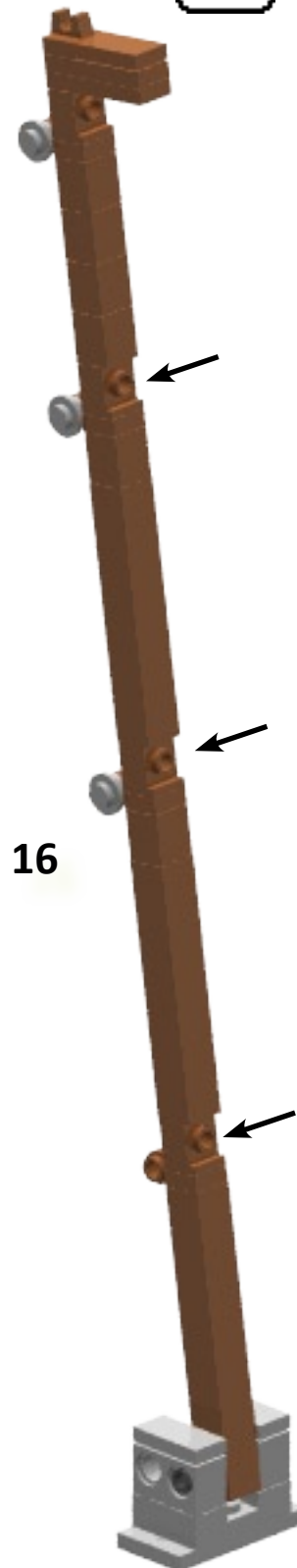
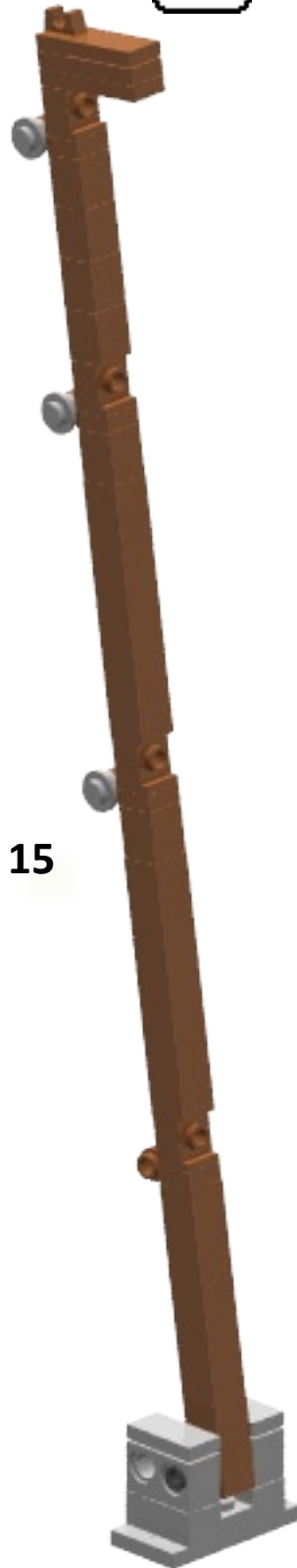


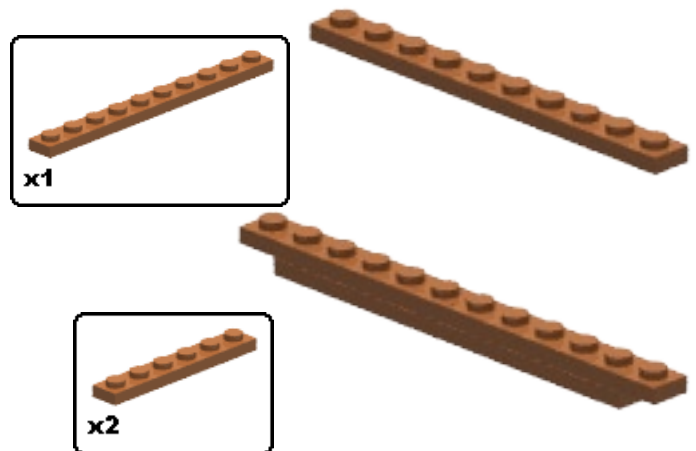
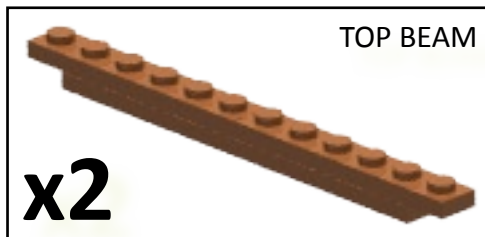
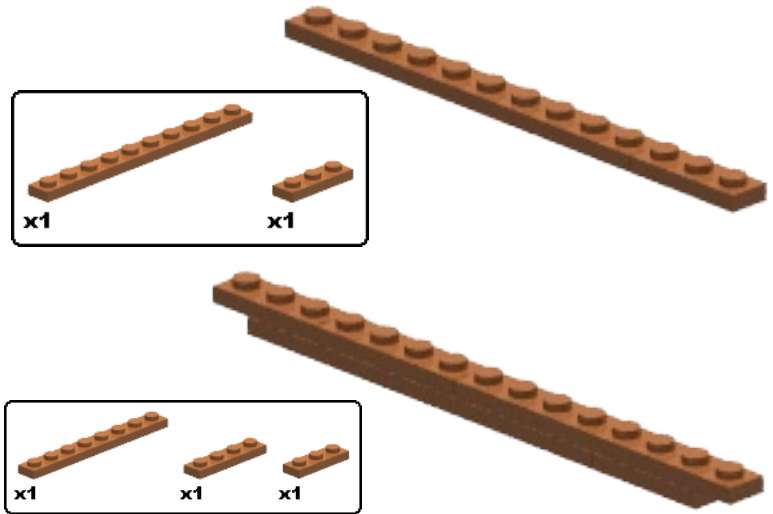
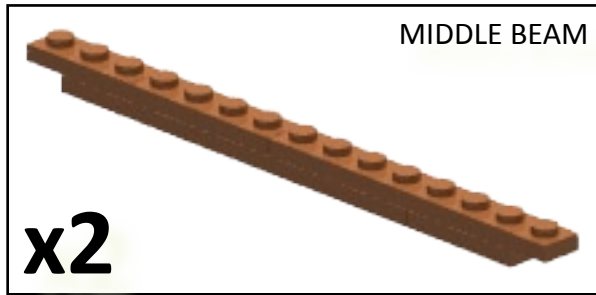
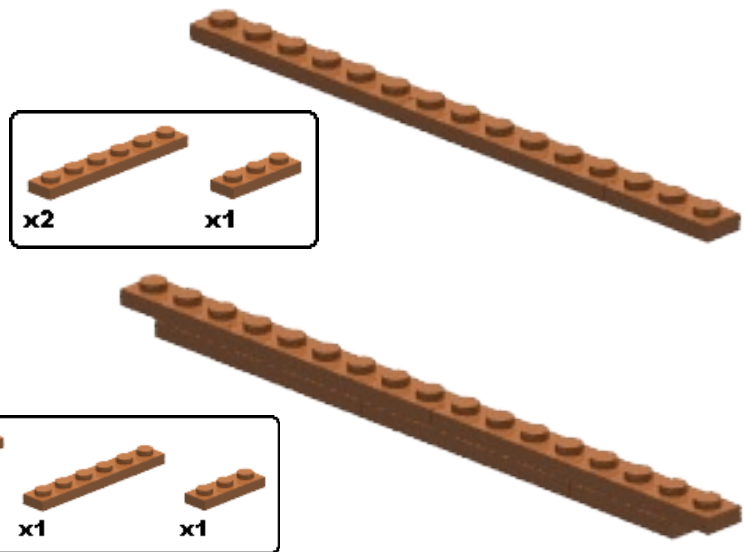
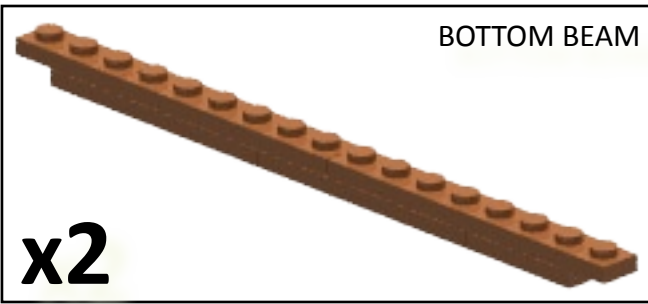
12

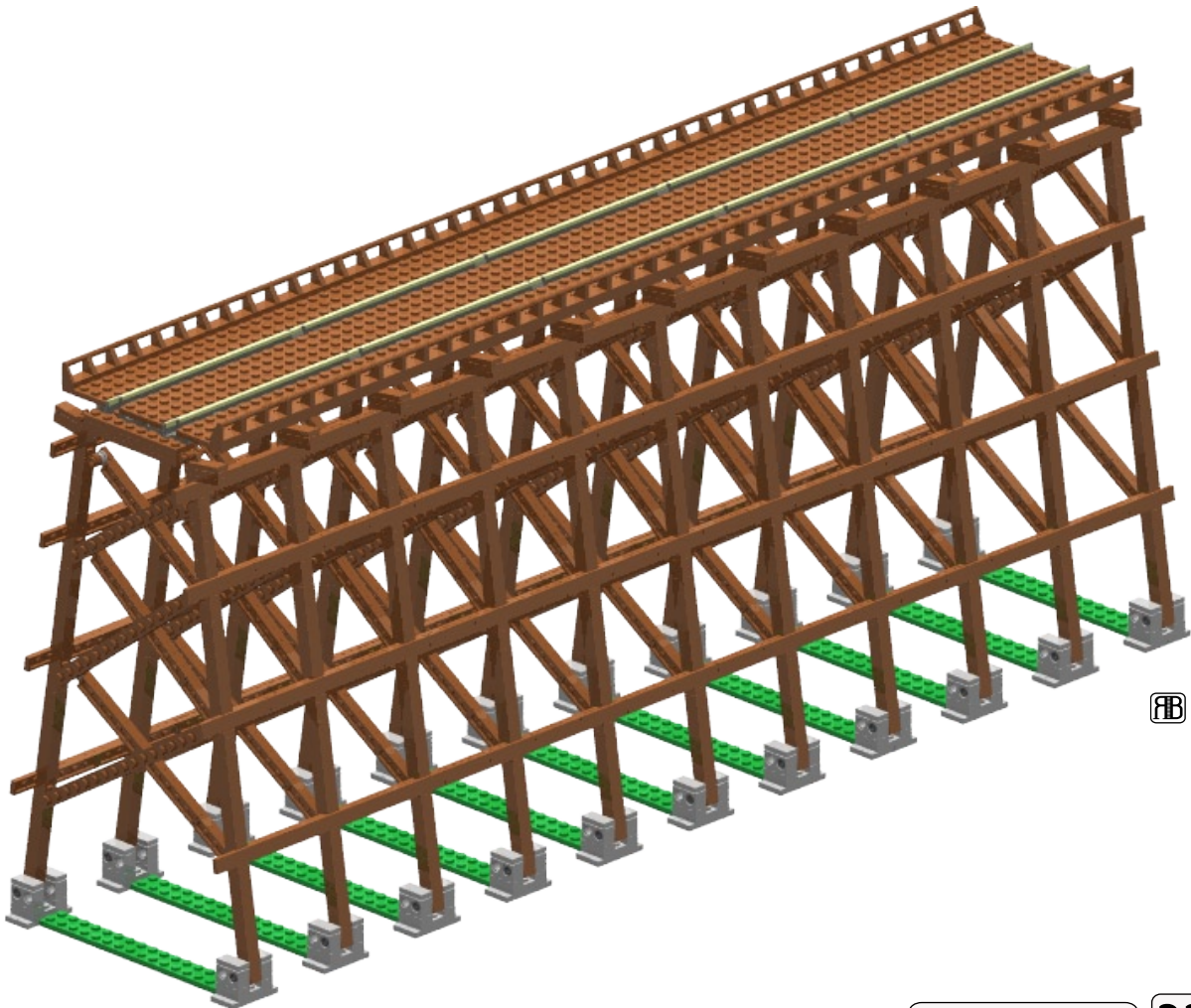
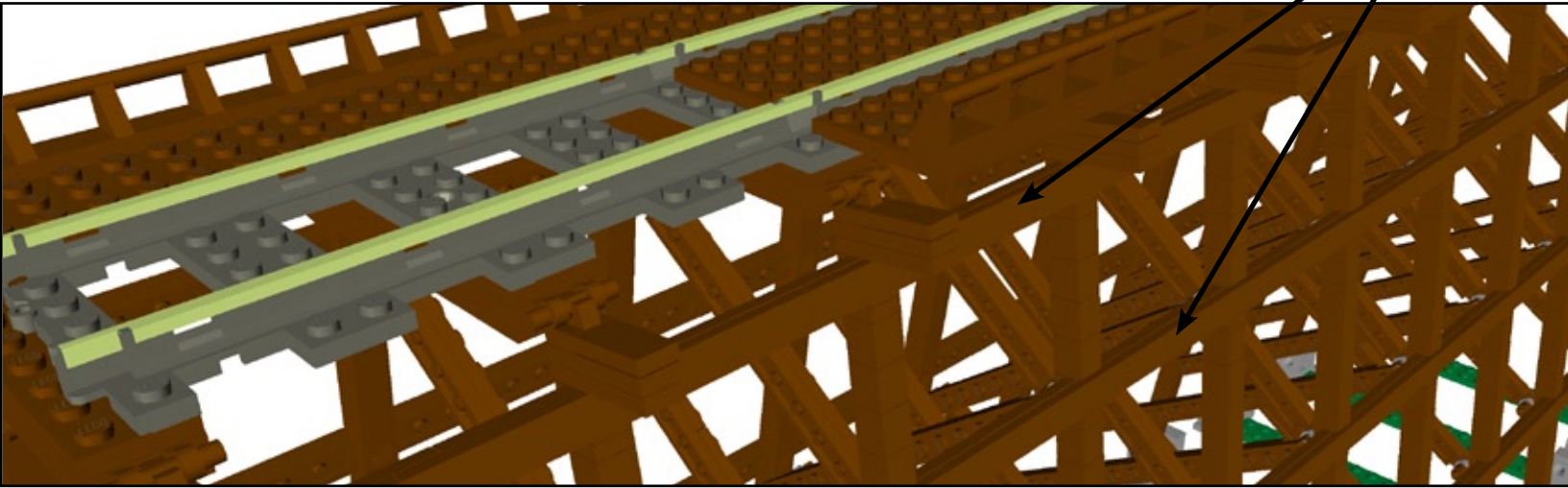
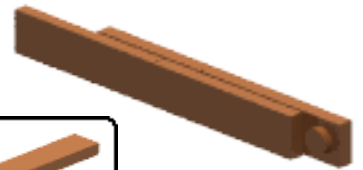
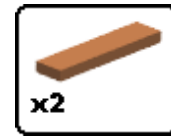
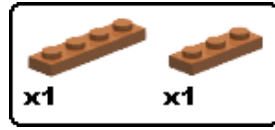
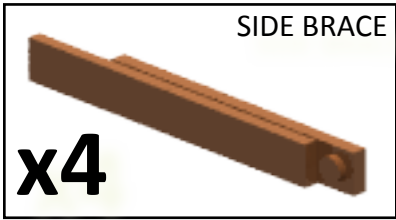


13









SELECTIVE COMPRESSION APPLIED TO MECHANICAL DETAILS

by Steve Barile

A while ago I was trolling around on BrickShelf and saw a stunning model of a Shay engine.

BrickShelf user: teknognome

<http://www.brickshelf.com/cgi-bin/gallery.cgi?f=312493>



The Shay is a narrow gauge steam engine that was used primarily in the logging industry. It has an offset boiler and tell-tale vertical pistons and cylinders on the opposite side that connect to a drive shaft that spans the length of the wheel base. The wheels are driven by crown gears at each axle.


Let's review the concept of selective compression. The first step is to identify the iconic design points while reducing or eliminating the rest while still capturing the esthetic of the original subject. Selective compression is generally used on visual details but what if the iconic details are mechanical functions.

The Shay does not have traditional steam driver wheels but instead only front and rear two axle trucks. What is tricky about modeling the Shay is the rotating, bending, and changing length of the drive shaft, while maintaining a 90° crown gear connection to each axel as well as a connection to the vertical piston rods. It barely makes mechanical sense while studying the real engine. Trying to achieve this in a six wide (even eight



wide) train with a pure LEGO solution seems to be impossible (ha, a challenge!).

What was unique about this AFoLs approach is that he simplified the mechanical mechanism without compromising the mechanic esthetic. He removed the ability of the trucks to rotate. This eliminates the drive shaft from having to bend and change length, thus greatly reducing complexity. As we know four fixed axles won't make it around curves so a trick is needed. The trick is the use of blind driver wheels (wheels with no flanges). The front and rear axles have blind drivers and the inner axles have regular drivers with flanges. This sets up a very simple two axle configuration with respect to the track. It is also worth noting that these driver wheels are not LEGO products, they are the medium size Big Ben Brand (BBB) drivers. Chaining or gearing the axles together ensures that all driver wheels are rotating at the same rate. The drive shaft is then simply connected to all four axles via crown gears. There is one blatantly missing mechanical element on this MOC, that is the lack of moving pistons which should be easy to add with out introducing much complexity.

Even though the example here is a Shay engine MOC, the lesson is really the concept of selective compression applied to iconic mechanical functions. Perhaps this can be applied to other areas... feel free to send in ideas that you may have. 

Editors Notes: In researching this article another fine example of a Shay was discovered. This model used a different approach to reduce the mechanical complexity; the trucks are floating under the chassis with the single (yaw) rotation point at the universal joint in the drive shaft.

By Nathan Proudlove (found on Flickr)

<http://www.flickr.com/photos/proudlove/833260516/>



www.RailFonts.com

OVER 70 DIFFERENT FONTS

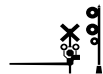
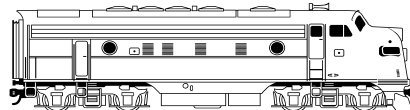


RAILROAD LETTERING

Heralds

RAIL ARTWORK

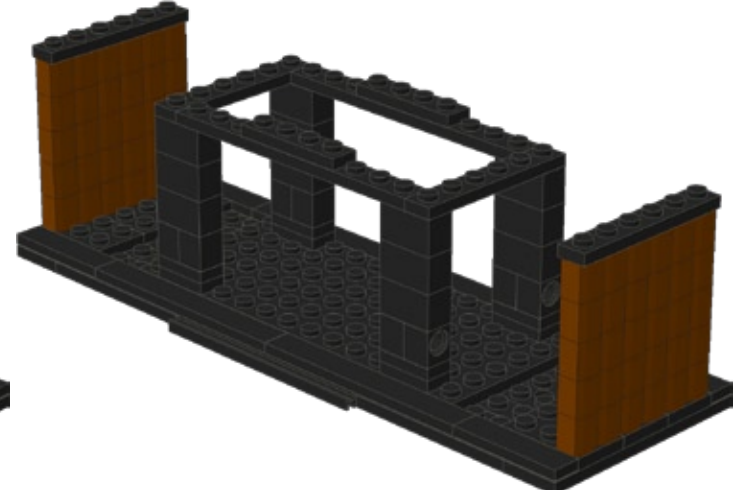
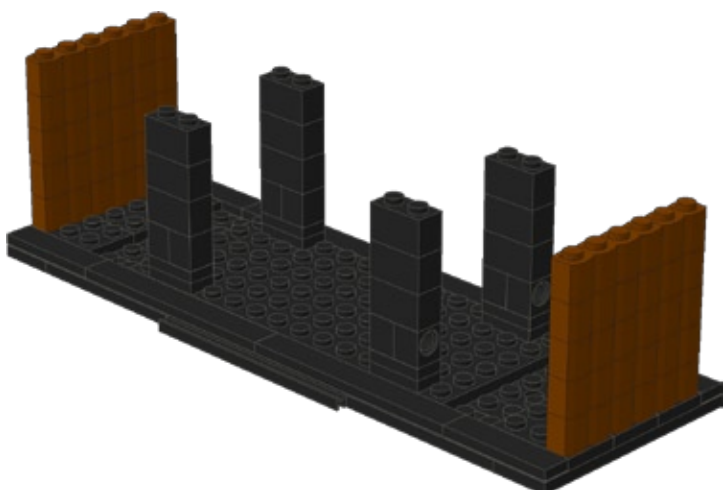
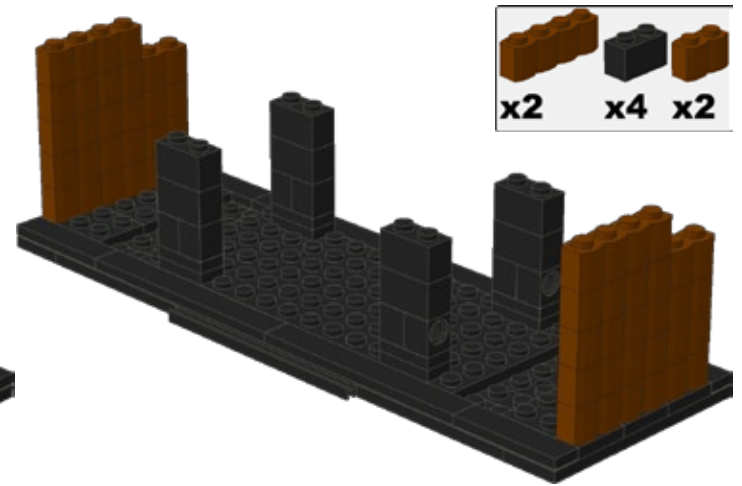
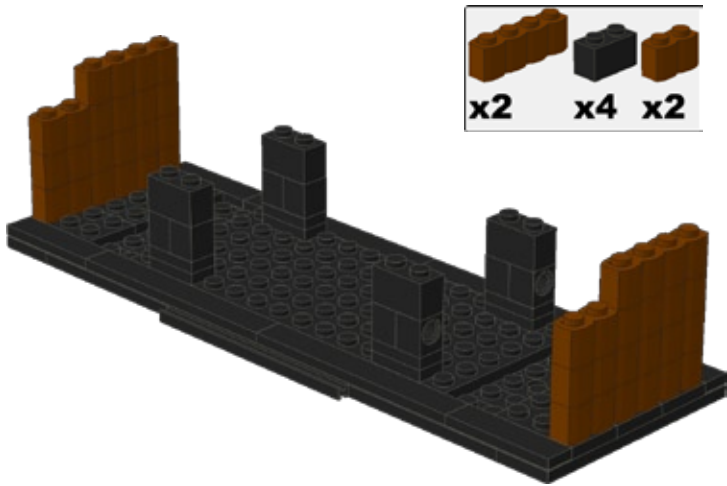
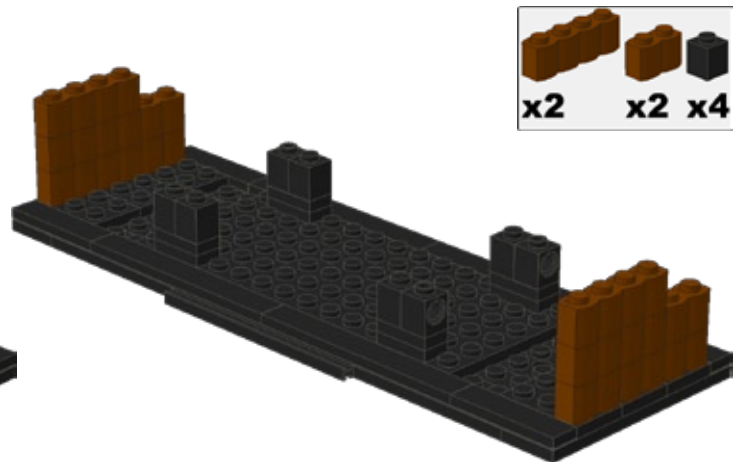
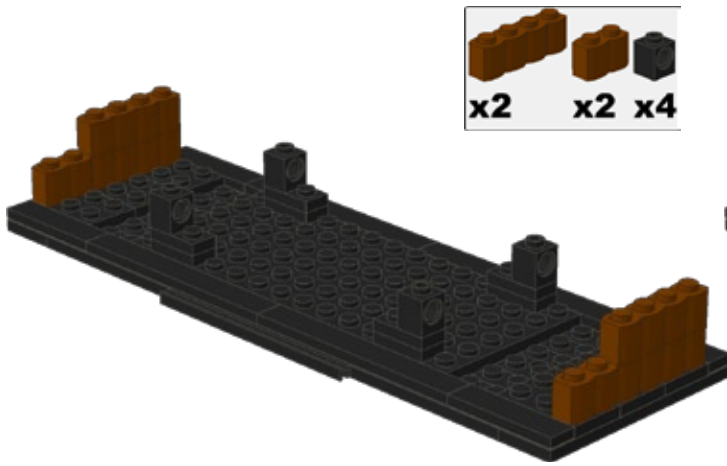
TRAIN SILHOUETTES

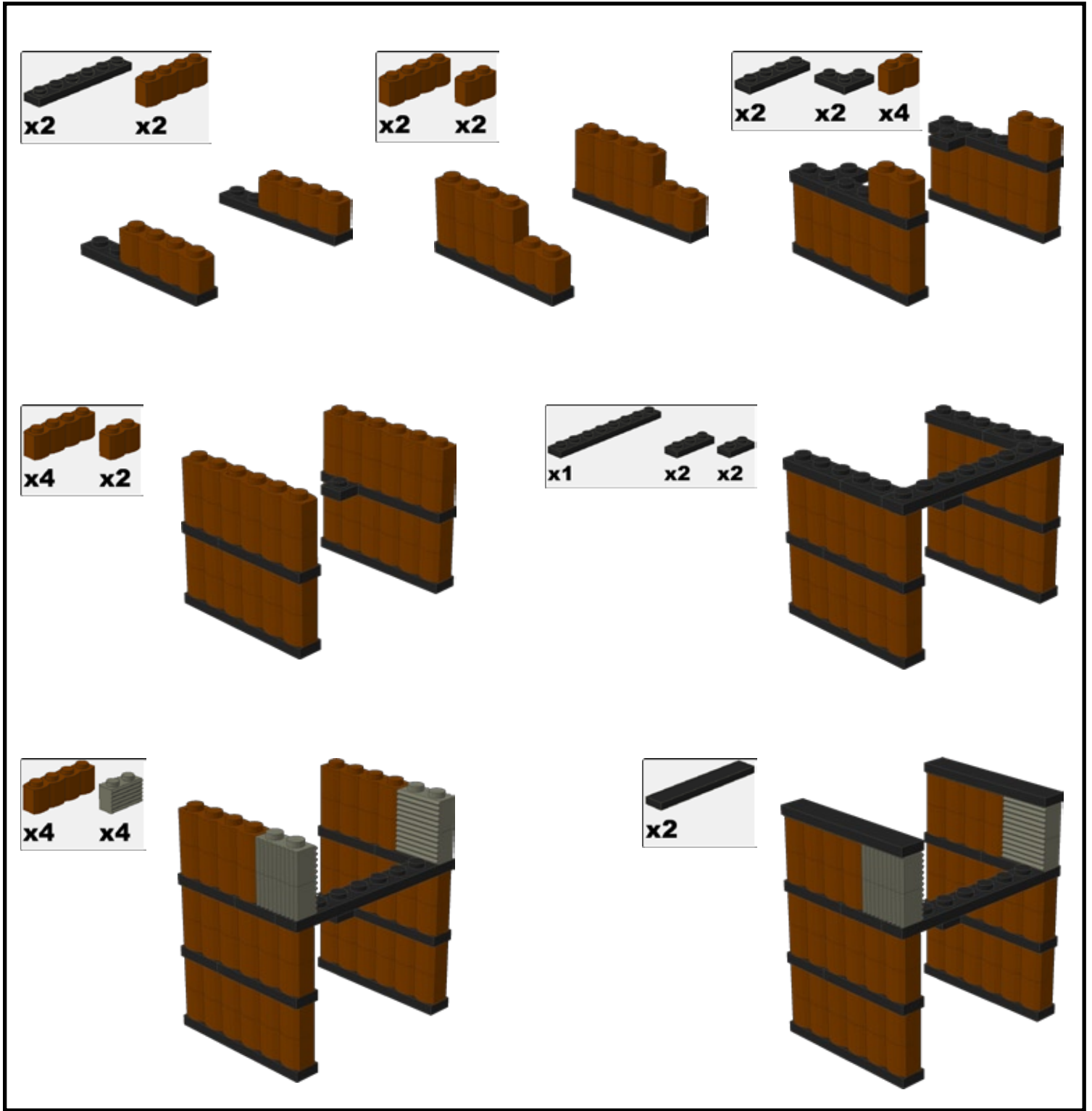


THEY WORK JUST LIKE ANY OTHER FONT

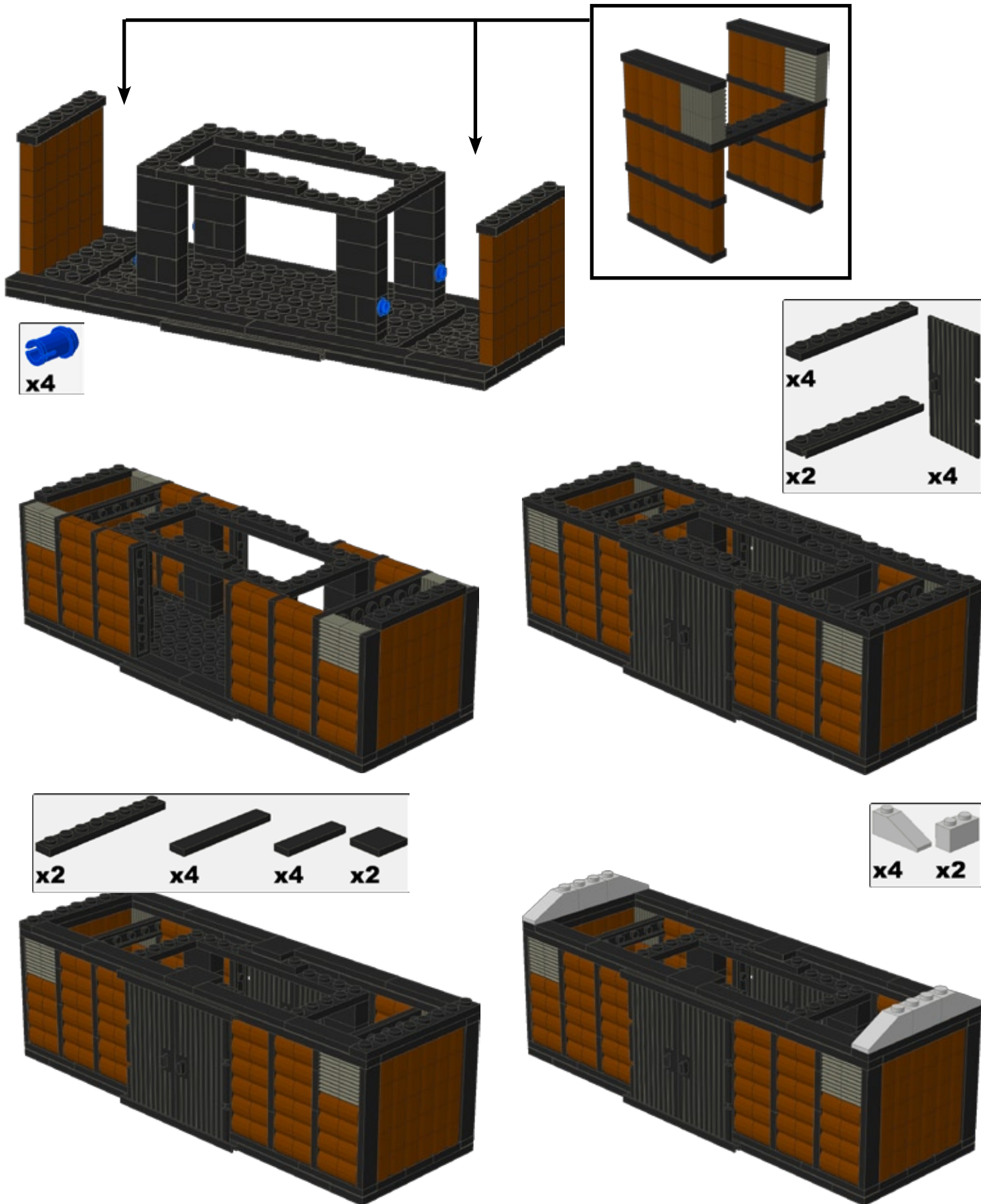


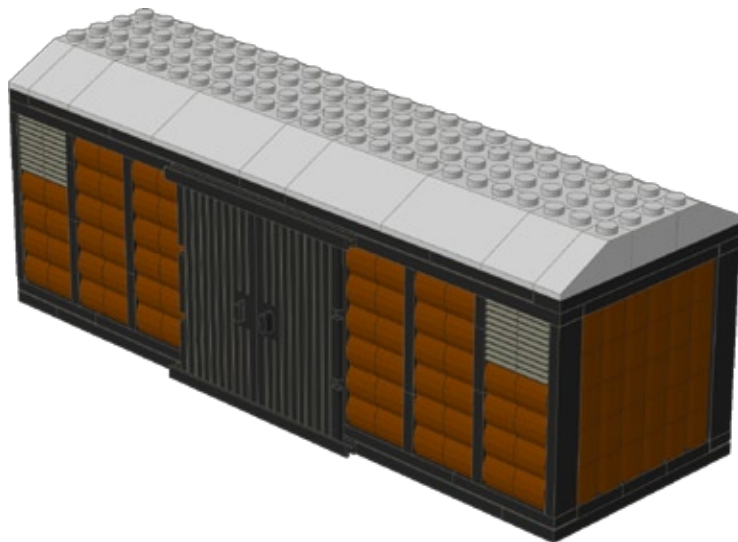
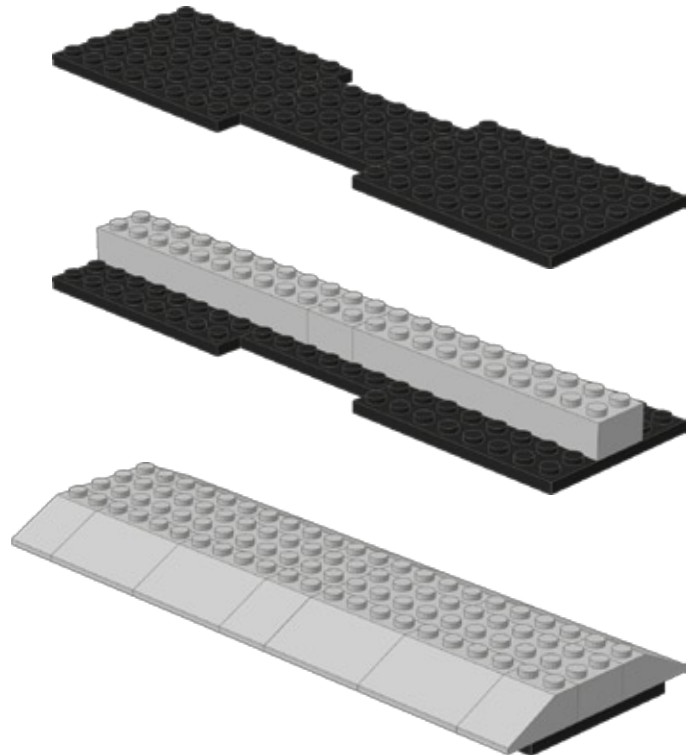
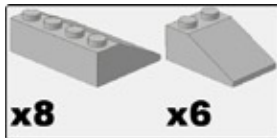
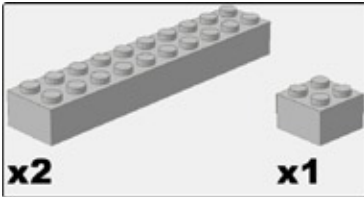
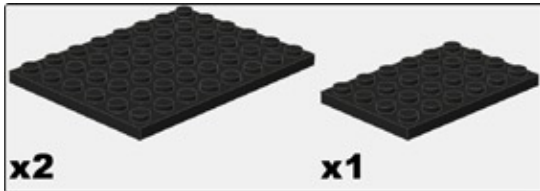
GREAT FOR DETAILING, DECALS, BOOKS, NEWSLETTERS, WEB PAGES, ETC.

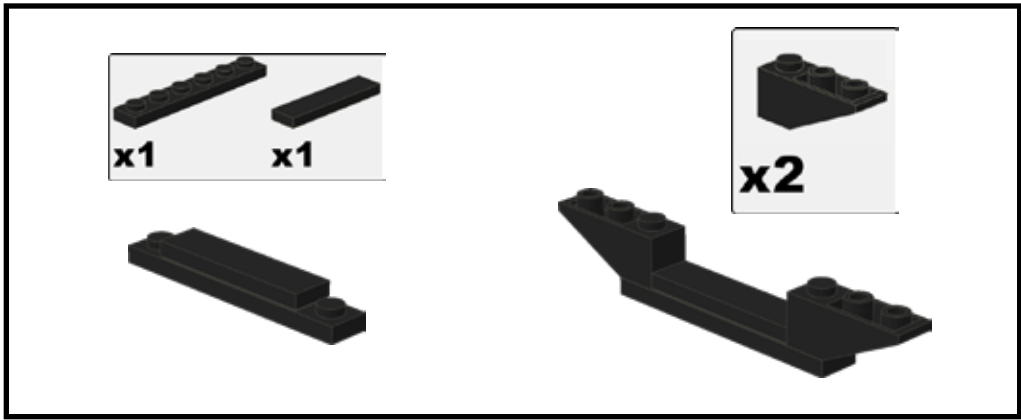




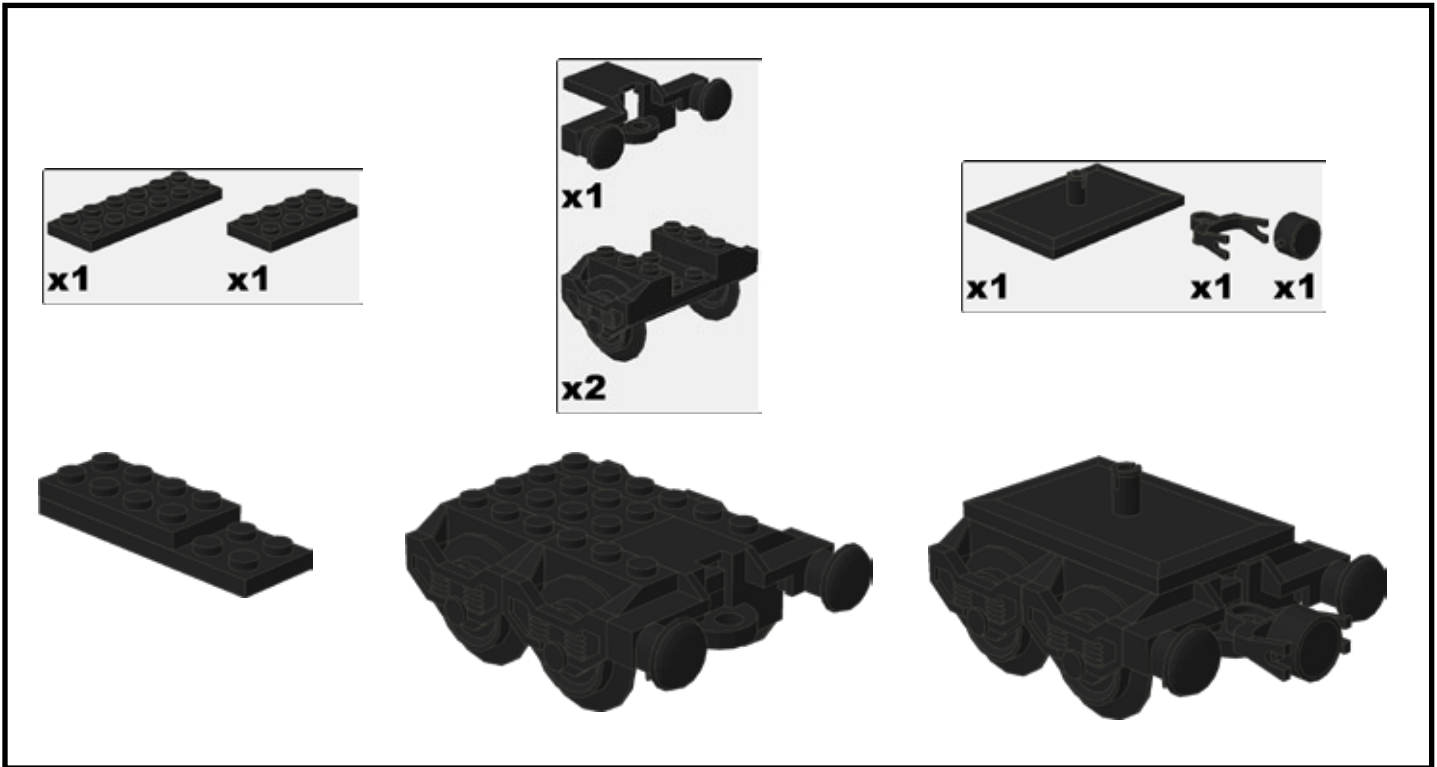
x2



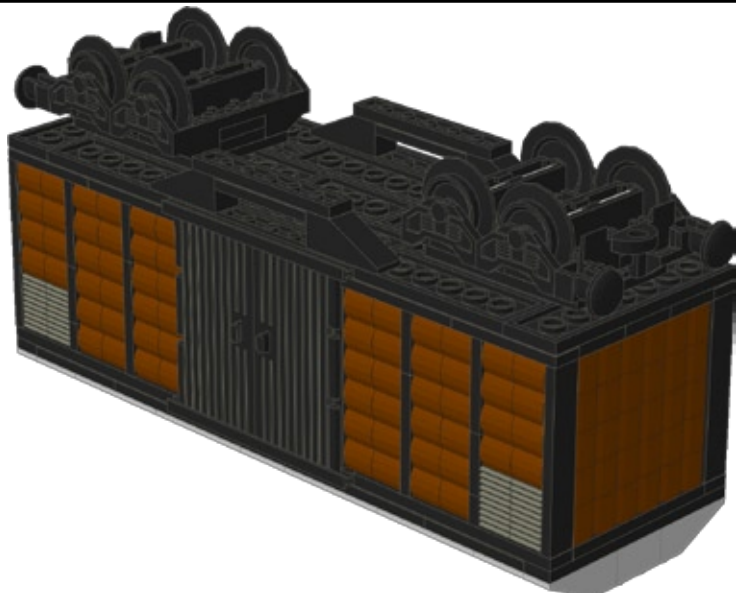


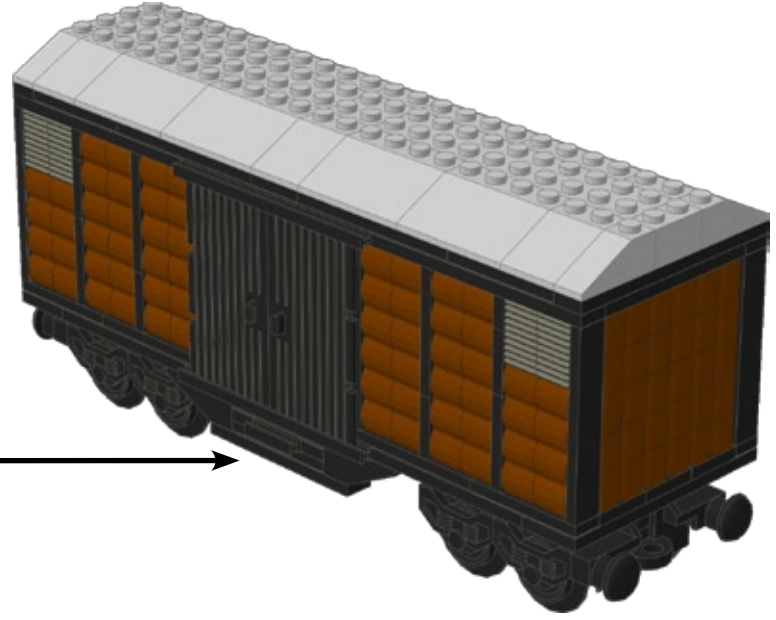
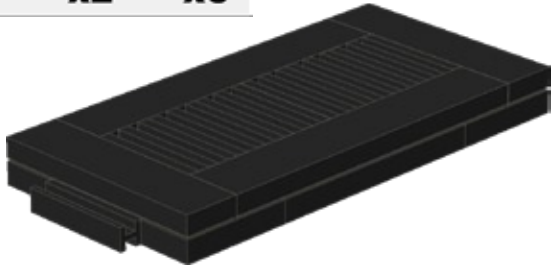
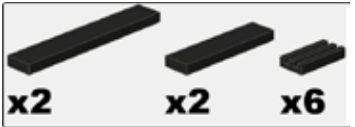
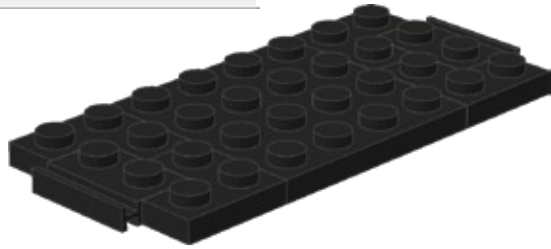
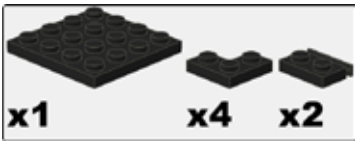


x2



x2





TRAINSPOTTING



<http://www.brickshelf.com/cgi-bin/gallery.cgi?m=rekok>



<http://www.brickshelf.com/cgi-bin/gallery.cgi?f=22280>



<http://www.brickshelf.com/cgi-bin/gallery.cgi?f=316878>



<http://www.brickshelf.com/cgi-bin/gallery.cgi?f=320042>



<http://www.brickshelf.com/cgi-bin/gallery.cgi?f=320578>



<http://www.brickshelf.com/cgi-bin/gallery.cgi?f=319793>

LEGO TRAINS AND LEGO PBRICKS

UNLEASHING REAL POWER FUNCTIONS

by Thorsten Benter

This contribution is basically about “building across multiple LEGO themes”. As if this was necessary – LEGO is *by definition* building across multiple themes. Nevertheless, it appears as if sometimes this definition is getting softened up – and from time to time just should be remembered ... maybe also high up there, where the big guys steer TLC through the challenges of present and future corporate adventures ...

It took a number of things to happen before I had the courage to ask someone deeply involved in the activities of the established LEGO 9V train community whether or not a report on my “programmable brick (PBrick) controlled trains and switch points” project would be of *any* interest to these folks at all. Knowing about the enthusiasm, creativity, and skills of the numerous LEGO train builders around the world from the internet – and the fact that all I am doing is basically taking existing ideas or solutions and pushing them a little in a certain direction – made me quite hesitant. From the pure building standpoint there is

nothing much exciting here. Then issue #2 of the RAILBRICKS was released with a very informative article by Steve Barile reporting on the new LEGO Power Functions (Pf) system “... It is a new (electro-mechanical) system designed to span all LEGO themes; space, train, town, technic, creator, etc... This is a fantastic business decision because it amortizes the cost of R&D across multiple themes’ budgets”. Now, *that* got me excited. At the end of the article an announcement said that a forthcoming issue of the magazine will have a story on automating 9V Pf trains using 9V DC/RCX/NXT integration innovations. That got me even more excited. For me as a physical chemist, the term “excited” has distinct but quite different meanings, ranging from “being very happy” to “having acquired enough energy to fall apart”. Well, I felt that with that announcement, time was ripe to go forward and ask someone ... and Jeremy responded very positively to my email within a day. Thank you very much RAILBRICKS for giving me the opportunity to share my ideas about multiple themes, power functions, PBricks, and 9V DC trains.



PROLOGUE

This part is just a summary of what accumulated to make me write up all this stuff. You may want to skip this section, it's basically about what I feel is "good and bad" about decisions TLC has made in the past and is not essential at all for the train automation project, just part of my motivation to carry on. I don't have as much time as I would like to have to play with the most inspiring toy in the world, but there are certain things that drive me to share as much time as possible.

First, there was the RCX. When I recognized the Mindstorms 1.0 box at Target in 1999, it was like seeing the light. I was an assistant professor at the University of California, Irvine, back then – we came to the US from Germany a year before – me, my wife and our two little girls. \$250 was a serious price tag but instantaneously I felt very strongly that that box did not deserve spending another dark night in the store. I am still very grateful to my wife that she did not argue one second – it must have been the light. With the salary of a second year assistant professor, \$250 is better spent for something other than LEGO bricks ...

Before that were my "dark ages", the usual stuff: I started my LEGO building career at age three, Santa got me the very first LEGO train set #323 back in 1965. 10 years later I was a proud owner of a medium sized storage box full of bricks and then simply other things became more important. Another 20 years later our first daughter was born, LEGO Duplo was around and after a couple of months we had a serious amount of Duplo train track along with two trains and a lot of rolling (Duplo) stock in our house. That basically ended my dark ages ... slowly but steadily I got my own LEGO System boxes, learned that technic bricks opened up whole new worlds, and much more.

Second, on the website of the North Georgia LEGO Train Club I not only found the superb TrackDesigner software, but also a couple of photographs showing an RCX built into a train car (http://www.nglhc.org/train_depot/rcxbc.htm) and some text talking about automating trains in the near future. Dean Husby's (Vancouver LEGO Club) TFM website (http://www.akasa.bc.ca/tfm/LEGO_rcxtrain.html) went even further: The photographs show an RCX powered train, and in the text, some necessary modifications of the train motor



Figure 1: My RCX powered GP40 and me at age 46, spring 2008.

were mentioned. The train motor assembly picks up the 9V DC, delivers that to the RCX which in turn powers the train motor. Just brilliant! This is the basic idea of my 9V DC train creations.

Third, after releasing the RCX 1.0, LEGO started to push the Mindstorms theme with new PBricks: The Scout and MicroScout were introduced and that was absolutely fabulous – intelligent bricks for different purposes. The Spybotics PBrick was soon added to that list, again adding new functionality to the existing line. Cybermaster PBricks apparently were around before, but unfortunately never showed up on my radar screen.

Fourth, very smart people were developing software for PBrick programming. Support from LEGO was next to nothing compared to what these people were developing. Seeing the reverse engineering efforts of the RCX firmware by Kekoa Proudfoot was just incredible (<http://graphics.stanford.edu/~kekoa/rcx/>). Dave Baum's "Not Quiet C" (NQC, <http://www.baumfamily.org/>) was and still is my favorite programming language; it is very well documented but more importantly: all RCX (1.0, 1.5, 2.0), Scout, Cybermaster, and Spybotic PBricks can be programmed using the same terminology, not to mention that even I, myself could manage to write reasonably functionable programs (my limited programming

knowledge is spanning QBASIC, a little VisualBasic, and NQC, that's it). Today NQC comes along with the integrated development environment (IDE) "BricxCC" (<http://bricxcc.sourceforge.net/nqc/>), originally developed by Mark Overmars, now pushed to the limits by John Hansen. Dick Swans' RCX firmware enhancement (that is quite an understatement, Dick himself termed his firmware that way on some website which unfortunately soon again disappeared – the "Swan firmware" is propelling the capabilities of an RCX to unbelievable performance) along with his RobotC IDE and C-programming language (<http://www.robotc.net/>) is another example of an extremely powerful addition to the suite of programming environments for the RCX and NXT PBricks. And there is much more out there: legOS, recently renamed brickOS, pbForth, and so on. I just prefer NQC – never change a running system.

Fifth, LEGO PBricks are essentially *micro controllers* plus power supply and some input/output drivers along with communication electronics. Micro controllers and buzz words like "embedded systems" and "distributed intelligence" are often found in the same context. The success of a micro controller or micro controller family depends heavily on *support* in terms of development environments, programming examples, application notes, and most importantly, compatibility (upward and downward). Let's take the NXT PBrick aside: The LEGO suite of PBricks was basically designed along that line. Software for the RCX, Scout, Spybotics, and Cybermaster PBricks was essentially compatible. Software for the MicroScout was not, but this particular PBrick understood the LEGO visible light link (VLL) communication protocol. Other "smart" LEGO bricks, e.g., code pilot compatible bricks, understood the VLL language too. The best part is that the Scout PBrick is a native VLL code speaker and can translate from one world to the other. With all that at hand, LEGO paradise seemed to become reality. TLC did it again; not only did they create the ingenious LEGO building bricks, but also the fantastic LEGO PBricks.

It just *seemed* to be paradise though; very soon I learned that the lifetime of a LEGO PBrick is not at all related to the lifetime of a mainstream LEGO System brick. Take the Scout; that beautiful piece of engineering came into the stores and after a year or two it simply disappeared. Why was that? Maybe because there was no appropriate support? For example, people *had*

to find out what the IR LED on the Scout was doing when the "C" button on the LEGO IR remote control (shipped separately or part of the Mindstorms Ultimate Accessory box, #3801) was pressed. That might be part of the *learning experience* – but almost certainly hurts revenue, because people are simply getting frustrated and don't recommend peers to buy such "toys". We all have learned the hard way that when it comes to revenues, TLC is (and needs) acting like a tough high-powered global enterprise and not like the good Old Danish Toy Factory from last century. Take the 9V DC train system. Removing the metal pieces from the track and powering the trains from *batteries* is not necessarily an environmentally friendly approach (even with rechargeable batteries you still need a lot of nasty chemistry to get reasonable amounts of electrical power from chemicals, I know), but the production of such plastic-only track it is simply dead cheap. The same thing happened to the RCX. Version 1.0 had a power jack that accepted everything from 9V DC to 18V AC. Version 1.5 and 2.0 did not have that jack anymore. Even worse, the RCX did not run reliably on rechargeable batteries, 7.2V was simply too close to the minimum voltage required. Now why on earth did TLC do that? No idea – other than saving a couple of bucks of production costs per PBrick – and the benefit of joining business with a battery manufacturer. Many new boxes requiring electrical power are powered by (free ...) Duracell batteries. And most recently the NXT system - yes, this PBrick is simply *beautiful*. TLC did it again. I have one of my own. Sound is superb, display is very nice, and programs are running very fast and efficiently. But, no power jack, no backward software compatibility, and no IR communication. Well, no comment here, I guess I made my point.

THE PROJECT IDEA

Motivated by all this "excitement", I wanted to unravel what TLC possibly missed. As if anyone cares – but this keeps me going. And although I am whining about all the "weird things" TLC is doing; If you ask me, LEGO is the best toy in the world and it seriously looks like it will always be. This of course means that these guys will continue to draw significant amounts of money out of my pockets ... and it appears, regardless of what they do.

My project idea is to operate *multiple, diverse* PBricks

within a rather *complex system* from one controlling center.

In other words, this is about “distributed intelligence” within the system for optimized operation of individual but coordinated tasks. This approach is found basically in every modern major house appliance, in cars, everywhere. And in trains of course. In modern trains there is the control center in the cab, one or more host computers somewhere in the body, and a diverse system of intelligent devices that execute, control, monitor, or do other things required for reliable operation. Which brings me back to whine about another thing I never understood completely. It seems to me that TLC tailored their PBricks towards creation of *independently operating robots*; robots that *look like* robots in movies. Robots like C3PO. Robots that challenge each other or master challenges on their own. There is absolutely nothing wrong with that, but in the real world, micro controllers do so many other things, it’s simply amazing. And they are *not* on their own. They are teamed up, and each of them does what it can do best or simply what it is designed for. It’s teamwork. But then, maybe this is another reason for believing that there is no need for the RCX power jack: you don’t want to see C3PO running around with a 500 foot long power cable plugged into an AC power outlet somewhere. That would have been too easy for Darth Vader.

This is just my personal perspective and opinion on a couple of things TLC did that I did not like too much. Now on to the project, and no more whining, I promise.

THE SET-UP

No doubt, this project needs to incorporate trains. As already mentioned, my first LEGO set was the #323 western train box some 40 years ago, so now is the time to demonstrate what PBricks can accomplish in combination with the 9V DC train system, as well as virtually all themes and brick styles TLC offers in its shelves.

- The “complex system” is represented by a 9V DC train track layout with a good number of switch points with multiple trains running on the track. The track is supposed to deliver power to all active devices, i.e., electrical power is available everywhere on the layout, independent from switch

positions. “Active devices” are envisioned as any LEGO creation that is hooked-up to some intelligence that understands LEGO byte code: a train, a switch point, lights for illumination, train turning platforms, signals, level-crossings.

- ... and a control center. Any computer with either a USB or RS232 port should do, because all I want to connect to this computer is the old RS232 or new USB LEGO infrared tower. Furthermore a control program is required; my choice is Microsoft Visual-Basic – simply because I grew up with PC’s, MSDOS, and QBasic. This is completely unimportant though, *any software* that is capable of interacting with the LEGO tower is perfectly well suited.

CONSIDERATIONS ON HARDWARE

The ultimate goal of this project is to unleash the power of distributed intelligence – or to unleash *real* power functions. “Intelligence” translates to “PBricks”, which are supposed to perform specific tasks. PBricks taken into consideration are all *compatible systems*; in other words systems that are easily integrated. Compatibility in this context is basically referring to communication and programming environments. The BricxCC IDE is my preferred choice because it allows communication with *all* LEGO PBricks. Using the “old” IR tower, RCX, Spybotics, and Scout PBricks can be programmed, with the “new” tower, in addition to MicroScouts. The Cybermaster PBricks are fully compatible in terms of programming (i.e., LEGO byte code) but that requires an additional RF communication tower. The NXT plays in another league. Both latter PBrick types, despite their large potential are thus not further considered here, at least for now.

Which leaves us with RCX, Scout, Spybotics, and MicroScout PBricks. Let us have a closer look at these systems:

- **The MicroScout PBrick.** Understands VLL code only. Can execute a formerly programmed sequence of commands of limited length. Cannot send VLL or any other signals but sound. Has a built-in motor. Permanent firmware is presently handling sound and motor action; e.g., “Turn on motor in forward direction for 1 second”. In other words: the MicroScout is basically a really smart LEGO motor. Operates with 3V DC, obtained from 2 AA size batteries.

- **The Spybotics PBrick.** Understands high-level LEGO byte code. Has an IR communication interface with half duplex capability. Has lots of memory, one built-in touch sensor, two built-in independently operated motors. Speaks the VLL language bi-directionally. In essence, the Spybotics PBrick is representing two very, very smart LEGO motors with strongly enhanced communication capabilities. Operates on 4.5V DC, obtained from 3 AA size batteries.
- **The Scout PBrick.** Understands high-level LEGO byte code. Has an IR communication interface with half duplex capability. Has rather limited free memory (about 400 bytes), two passive sensor inputs (i.e., analogue/digital converter inputs), and two 9V PWM controlled motor outputs (7 power levels and “off”). An additional third output drives a built-in LED emitting essentially VLL code (i.e., the command “Output C forward” results in a flickering light emission representing the appropriate VLL “forward” code). In other words, this output can *control* a MicroScout. Operates with 9V DC, obtained from 6 AA size batteries.
- **The RCX PBrick.** Understands high-level LEGO byte code. Has an IR communication interface with half duplex capability. Has extensive free memory (depending on firmware running, but generally in the kByte range), has three active sensor inputs, i.e., pulsed analogue/digital converters; when “on” signals are being processed, when “off”, DC voltage is applied to the input lines. This pulsed DC voltage can be used to deliver power to active sensors, i.e. sensors with some built-in electronics. Has three 9V PWM controlled motor outputs (8 power levels and “off”). RCX PBricks come as 1.0, 1.5, and 2.0 varieties boldly printed on their plastic case. This is very misleading; they are basically all the same with the exception that only the 1.0 version has an AC/DC power input (9V DC to 18V AC, alternatively with 9V DC obtained from 6 AA size batteries), and TLC has improved the firmware for version 2.0. As far as I know the PBrick hardware behaves exactly the same in versions 1.0 and 1.5 though. I have them all and have never noticed any other differences. An RCX 1.0 is running well with the RCX 2.0 firmware.

After inspecting the available resources I picked what I needed for my project.

MicroScouts appear to be ideal candidates to automate switch points. As “intelligent motors” they can handle a switch point independently. VLL Commands like “1 second forward” seem to be appropriate for this task. There are many powerful switch point design solutions on the internet. This very nice and comprehensive summary compiled by Philippe “FrogLeap” Label (http://www.freelug.org/article.php3?id_article=186) is the place where I got my ideas for an “intelligent” automated switch point drive.

As a native VLL speaker the Scout PBrick is perfectly suited to let the MicroScouts know when to change the position of a switch point. However, there is only one VLL optical output on the Scout and automation becomes expensive. There is a straight forward and relatively cheap way to hook-up multiple MicroScouts to one Scout with a de-multiplexing device (Figure 14). I haven’t seen anything comparable elsewhere yet, but I bet some other people have already done similar things. Furthermore, with two motor outputs, a Scout should also be capable of controlling a train equipped with either two motors or one motor and light. Any other tasks that incorporate motors are thinkable – sadly enough though, the Scout does not have an AC/DC power jack.

As far as I am concerned, the most versatile PBrick is the **RCX 1.0** with either the RCX 2.0 firmware or the Swan firmware loaded (See section on software). The latter requires RobotC as the programming environment, as already discussed. The RCX 1.0 has a power jack that can be used to permanently power the brick from virtually any AC/DC source. The powered train track is a very good source for this project, either directly by picking up power *from* the track with the 2x2x2/3 brick contacts (#5306c01) usually used for power delivery to the track, or with the above briefly addressed train motor modifications the RCX 1.0 is ideally suited as on-board train controller. Two outputs may be used for driving modified 9V train motors, one output for light. If you want to run all battery, then the new 9V RC train motors work also and no motor modifications are required. The RCX can perform more “real power functions” – it just depends on *your imagination* and some programming skills.

CONSIDERATIONS FOR SOFTWARE

As already mentioned, appropriate software is required on the host computer as well as on the PBricks. In the present setup, an “old” IR tower is connected to my Dell Inspiron 8600 laptop via the RS232 interface. The operating system is Microsoft Windows XP with service pack 2 installed and I am using Microsoft Visual Basic 6.0, service pack 5 as the programming language for the train control program. The RCX 1.0 PBricks are running the Swan firmware (fast0618.lgo) or the LEGO RCX2.0 firmware (firm0308.lgo, with “restricted” capabilities, see below), respectively. The Scouts and MicroScouts are running with their original configuration. I am using BricxCC version 3.3 as IDE which ships with NQC version 3.1 r5 as PBrick programming language.

CUSTOM PARTS

Never modify any original LEGO part, that’s one of my rules. If something doesn’t work out, I am trying to change my design until it does work with original LEGO parts. If it appears to be *impossible* to do the job with original LEGO parts, I am allowing myself to add custom things to my projects – so this rule is actually more a guideline than a rule. *Impossible* is a very tough call though and any such modifications should be as minimally invasive as possible. I hope that my short list of three custom parts does not stretch this guideline too much:

- Extend the communication range between host computer and PBricks with wireless technology.
- Modify 9V DC train motors so that they have a power output and a power input along with matched wires.
- Make use of custom optical fibers for secure VLL communication.

Here is my justification; infrared light is a reliable, efficient, relatively interference-free, and particularly *cheap* transport vehicle for communication purposes. The IR technology is very mature and many powerful electronic components are readily available on the market. This sounds perfect – if there wasn’t the “line-of-sight” requirement. In my project, the position of the host computer and the IR tower is rather static, i.e., on my desk. Furthermore, the whole track layout is built “into” my home office, which is about 4 x 8 m²

(approx. 12 x 24 sqft.) at floor level and 1 x 8 m² (3 x 24 sqft.) in 2 m (6 ft) height. My home office is located directly under the roof of our house with sloping walls everywhere. This translates into densely stacked tracks, lots of obstacles, and quite a lot of track behind side boards, shelves, and under my desk. Not good at all for reliable IR communication. With wireless technology however, communication is much smoother. Since all PBricks (with the exception of the MicroScout) are using bi-directional communication, each IR/RF translation device requires an IR transmitter and receiver as well as an RF transmitter and receiver, an IR/RF transceiver).

Modification of the 9V DC train motor is mandatory if more than one 9V DC train is supposed to run on the same section of powered track. As already pointed out, I simply don’t like the *battery* operated 9V Pf train line that much. Furthermore, my changes to the train motor are fully backward compatible. Just connect any (classical) 9V wire connector (i.e., the 2x2x2/3 wire connector brick or the less popular 2x2x1/3 version) to the power terminal on the modified train motor. In order to use the modified motor as a power source for an RCX 1.0 PBrick, I also needed modified power wires, as shown further on.

VLL communication is naturally subject to interference from other (visible) light sources. There are a lot of visible light sources, including sun light. In order to ensure reliable VLL operation, the information transported by visible light should thus be delivered “shielded” through fiber optics from the transmitter to the receiver. Both, the “new” USB IR Tower as well as the Spybotics, Scout, and MicroScout PBricks have optical terminals which are compatible with the LEGO technic pin 1/2 (#4274). This part is compatible with the LEGO optical fibers and the technic ribbed hoses; however, these are available in relatively short lengths only. Any cheap plastic fiber will do the same job as the original LEGO parts, and transfer of visible light is possible over several yards without significant light intensity loss.

WRAPPING UP

The following schematic represents a summary of the basic idea of this multiple PBrick train automation project. The host computer interacts with the PBricks via IR or RF in half duplex mode.

Figure 2: Schematic of my PBrick train automation project.

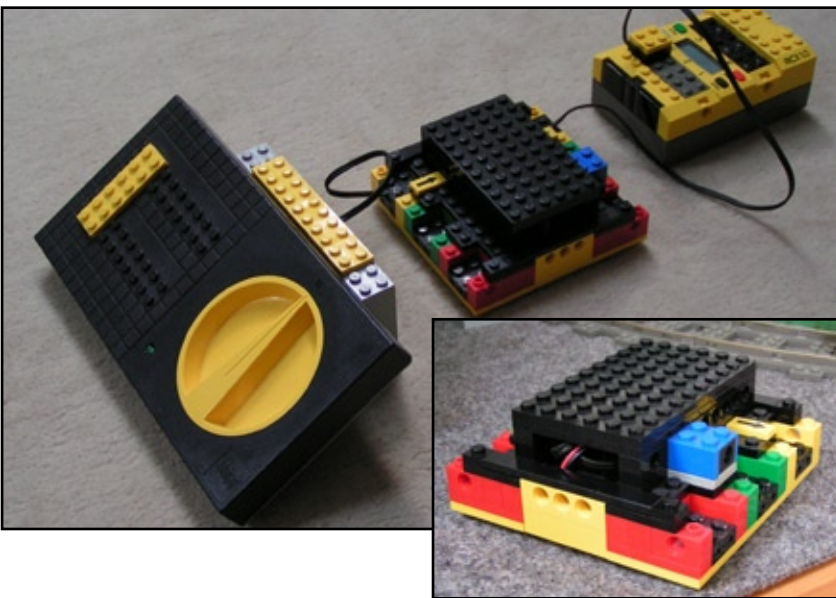
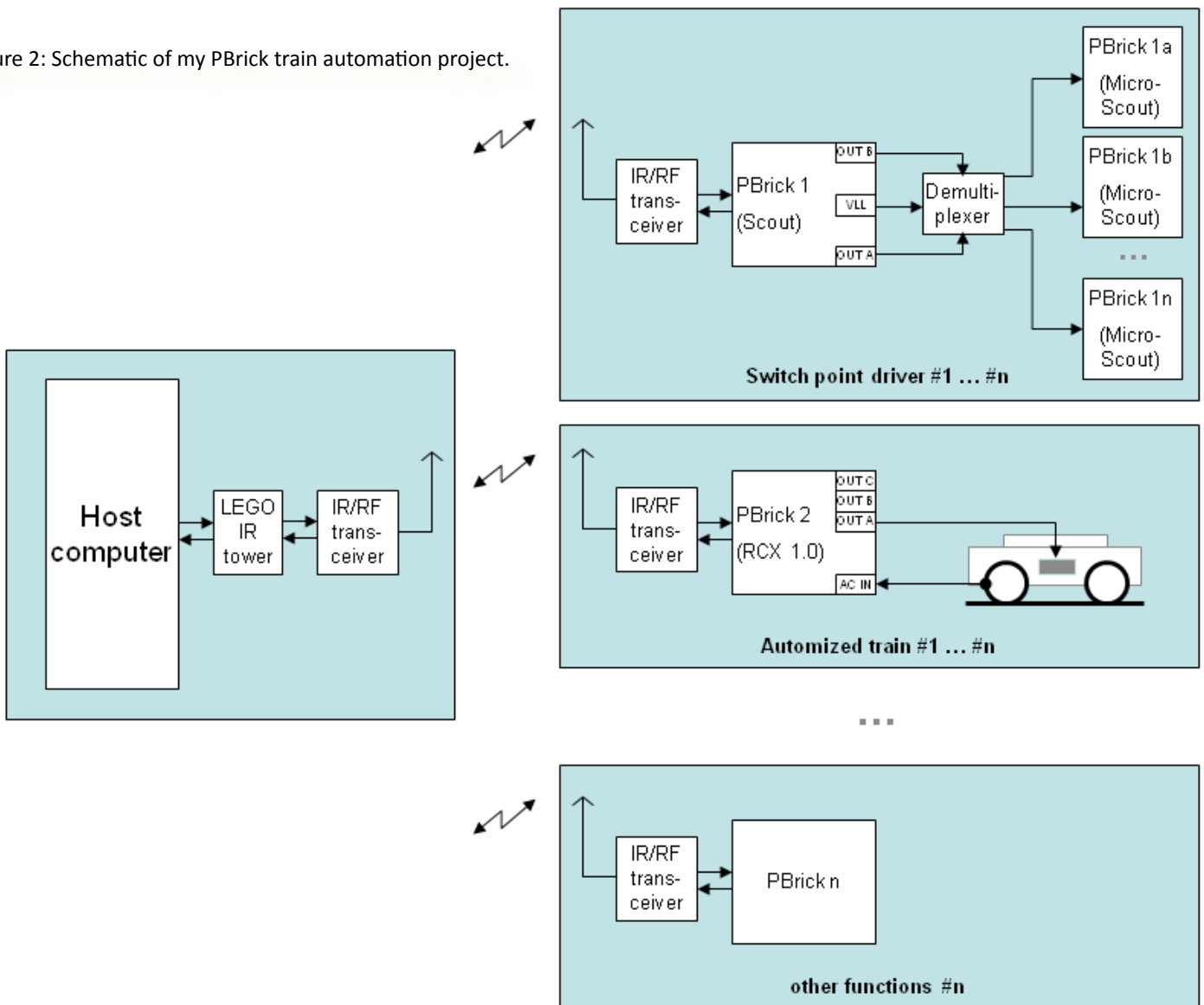


Figure 3: Custom train speed regulator voltage (+ ... - 9V) to RCX input voltage (0 ... +5V) converter.

Within this schematic, a PBrick may also run a host program (which issues commands rather than executing them), in addition to or entirely replacing the host computer.

I have built one such application as illustrated, in which three LEGO 9V DC train speed regulators are connected to the inputs of an RCX via a custom +/- 9V to 0-5 V converter. The RCX senses changes in any of the three speed dial position and sends the appropriate signal to the target PBrick controlling a train. The outputs of the speed regulators connect to the color coded terminals on the back of the voltage converter, whereas the outputs of the converter on the front are connected to the RCX inputs.

The speed regulators may also be stacked within



Figure 4: Speed regulator board assembly. The panels on the top right of the regulator board host 9 touch sensors. The construction is a little weird, but in this way a 2x8 electrical plate can be used as common connection for each set of 3 touch sensors.

a technic frame; the converter as well as the RCX fits into the two lower compartments.

The space on the right is occupied by a multiplexed switch board. The second RCX installed in the lower right compartment decodes which one of the 9 switches (#877 electric touch sensors) are pressed with a simple row/column algorithm. The three outputs of the RCX sequentially power the three “rows” in an endless loop. Once a row is powered, the three inputs are sequentially scanned and so on – same thing as in virtually every keyboard. The prototype version works quite well, but I don’t like the design. This clearly needs more work. The IR/RF transceiver is mounted in the back and is shared by both PBricks. In the future this is supposed to be a control center for 3 trains and 9 switch points (or groups of switch points) making any host computer obsolete – at least for relatively small layouts.

BUILDING, BUILDING, BUILDING ...

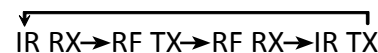
Brief overview of the custom parts

The IR/RF transceiver ensures smooth communication between the host computer (or the host controller) and the other moving or stationary PBricks, but it is *not* a prerequisite to the project – if there is mostly a line of sight between both or there are IR light repeaters installed, then RF is not required. Nevertheless, the design of this device is briefly addressed next.

It should be emphasized here that there are numer-

ous brilliant ideas and solutions for IR range extension with RF technology. A good starting point for homebrew applications is The Remote Control Store (<http://www.rentron.com/PicBasic/RemoteControl.htm>) run by Bruce Reynolds. He has also published a wonderful tutorial on how to “Transmit IR Signals Through Walls” (http://www.rentron.com/IR_TO_RF.htm) along with some nice code for 38 kHz generation with a PIC micro controller.

My transceiver operates in half duplex mode and it can communicate in both directions, but can do only one thing at a time: either receive IR signals (IR RX) and transmit them as RF signals (RF TX) or receive RF signals (RF RX) and transmit them as IR (IR TX). Just connecting the IR RX to the RF TX line and vice versa does not work. Without some additional electronics, a positive feedback loop between the onboard



components would be established. This means that when IR is received the IR TX channel needs to be “blocked”, and when RF is received, the IR TX channel needs to be “blocked”. The schematic illustrates one possible approach (I am emphasizing that without the experimental data of Dick Swan published on Lugnet (<http://news.lugnet.com/robotics/rcx/?n=2165>) this would not have been successful at all).

When the device is idling, both inverted outputs of



Figure 5: The IR/RF transceiver. The transceiver may also be positioned further away from the RCX IR window. In fact, I needed to put a layer of paper on the IR detector of the transceiver. Even if the IR power is set to “low” on the RCX, the detector was saturated without paper.

the MMs are H. When the IR RX module detects 38kHz signals, it decodes the bursts of modulated IR light (top left) into a bit stream which appears at the input of MM1 since AND gate A is “open” – the output of MM2 is still H. MM1 triggers, the output goes from H to L, and “closes” AND gate B. The bit stream also appears at the input of the RF TX module and corresponding 433MHz bursts are generated. This is of course detected by the RF RX module but the decoded signals don’t make it to the IR TX module (which would create a positive feedback loop) since AND gate B is still closed and kept close for the remaining IR bit stream.

Whenever the IR RX channel receives a signal, the corresponding monostable multivibrator 1 (MM1) is triggered and “closes” the logical AND gate connected to its inverted output. The IR data is fed to the RF transmitter and sent. The RF receiver detects this strong RF signal, but the data doesn’t make it to the IR transmitter because the AND gate is “closed”. The minimum time the AND gate is closed is determined by the time constant of MM1. This is set to just a little more time than the duration of one byte of data in addition to 1 start, 1 stop, and 1 parity bit. At 2400 baud serial communication rate (the RCX default value) this translates to about 4.6 ms. As we know from Dick’s measurements, the RCX needs about 9.6 ms to send first reply bytes. Thus a minimum closing time of 6 to 8 ms appears to be on the safe side. Furthermore, the monostable multivibrators need to be re-triggerable. This simply means that each time a rising slope is detected at the input, the 6 ms monostable time is refreshed. In other words; the whole set of bytes of each LEGO byte code packet (for example, sending a message from the tower to the RCX is represented by an 8 byte packet [0x55] [0xFF] [~0xFF] [0xOP] [~0xOP] [0xDn ~0xDn] [0xCS] [~0xCS];

OP = OpCode, Dn = n DataBytes – here the message content with n=1, and CS = checksum) is sent contiguously without interference from the onboard return channel. Whichever channel detects incoming signals first (either RF or IR) is transmitting the converted the signal at the corresponding output (either IR or RF).

My first generation transceivers used discrete TTL logic running on 5V, and the 38 kHz IR carrier was generated with a good old 555. It worked perfectly well; the problem however was that this thing was drawing more than 20 mA DC current and it was rather bulky. The “second generation” IR/RF transceiver operates with a PIC micro controller; all timing is now in “digitized” form, basically just counting loops. There are several advantages: the device is much smaller, much more stable, and DC current is down to 2 – 6 mA at 3.3V, depending of IR/RF traffic activity. This means that the transceiver can be operated from an active RCX input without any problems. The transceiver features a simple bridge rectifier, voltage stabilizer, and capacitors, and can be operated from anything between 4.5 V to 20 V AC or DC and fits into 3 stacked and hollowed out 2x4 bricks.

The PIC is programmed using MicroChip’s PICKit flash 1 starter kit and the MPLab 7.1 IDE. The PIC basically emulates the above described TTL hardware as well as the function of the 555 as a 38kHz 50% duty cycle rectangular pulse generator. When nothing happens, the program idles in a continuous outer loop. Whenever 38kHz modulated IR is detected, it branches off to the right part in the flow chart, waits for a certain amount of time, (just counting up the number of loops; each operation and thus each loop requires a fixed and nearly constant amount of time) and then returns to idle. In case of IR detection during looping, the coun-

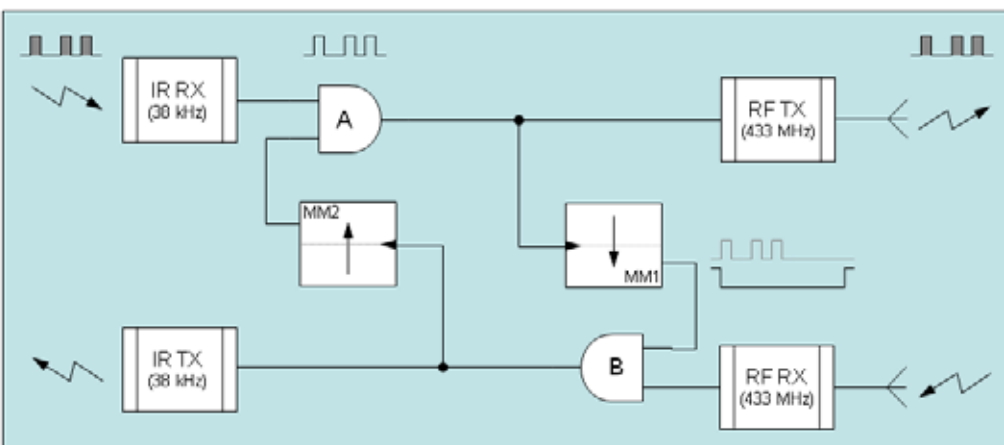


Figure 6: Schematic of the IR/RF transceiver electronics. Below (and above) the AND gates are re-triggerable monostable multivibrators (MM). The delay time of the MM for changing the inverted output from H to L is negligible; the delay time for returning to H is adjusted to the time one byte requires for transmission. Each positive slope in the bit stream received re-triggers the MM and the output is kept L.

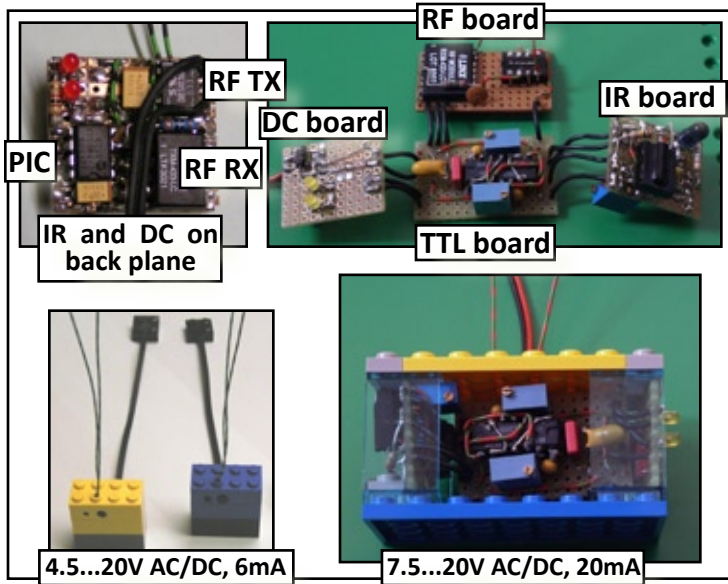


Figure 7: PIC and TTL based IR/RF transceivers. "Old" design on the right using TTL and other discrete components; "new" design on the left. With 6 mA (max.), the PIC based transceiver can be powered from an active RCX input. There are many more RF devices available. For example, the "DR3100" half duplex transceiver module from RFM. This module is even smaller than the LINX receiver alone. The transceiver would then fit into 3 stacked hollowed out 1x4 bricks.

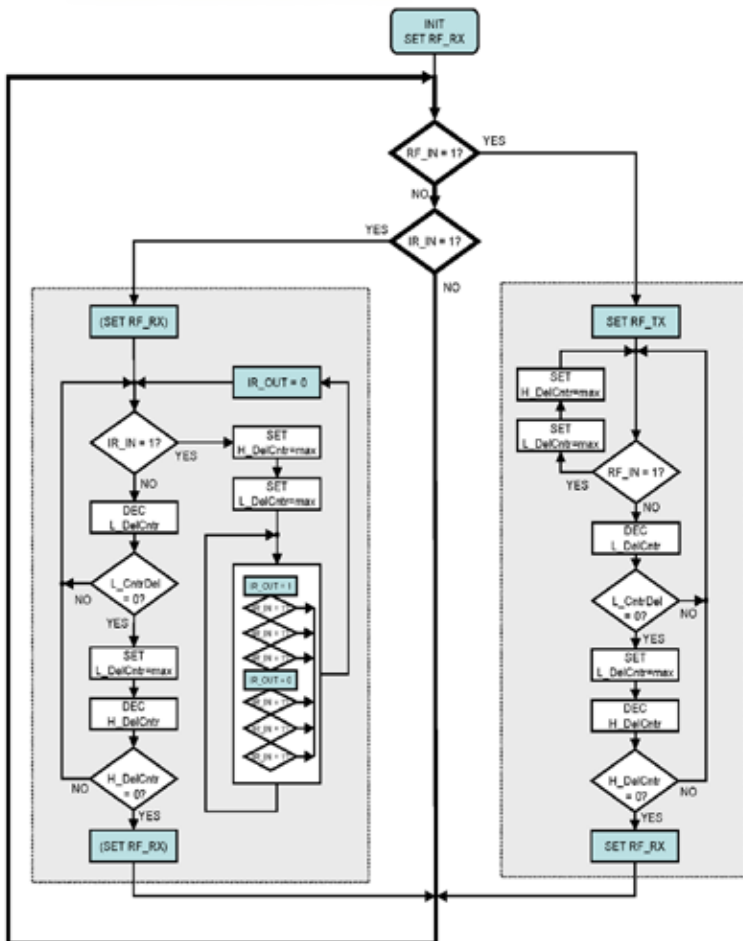


Figure 8: Flow chart of the PIC program.

ters are reset and the program continues to run in the inner loop (this emulates the retrigger function of the 74123 TTL monoflop). In the event of RF detection, basically the same thing happens on the left part of the flow chart, in addition, the looping includes 38kHz generation on one PIC output.

In summary: The PIC12F509 costs less than \$2 and does the work of at least 3 discrete powerful logic chips. Well, it's a micro controller ...

Modifications to the train motor have been described on the internet several times – I am just noting a couple of minor details on my approach. (Figure 9) How to "open" the train motor and dismantle the entire assembly is shown here: <http://www.lgauge.com/trains/dcc/dcc.htm>, section Decoder installation 1 – 3. Then, the motor is separated from the pickup wipers. The connections to the motor are cut at the base, and then the electrical connections are soldered as shown

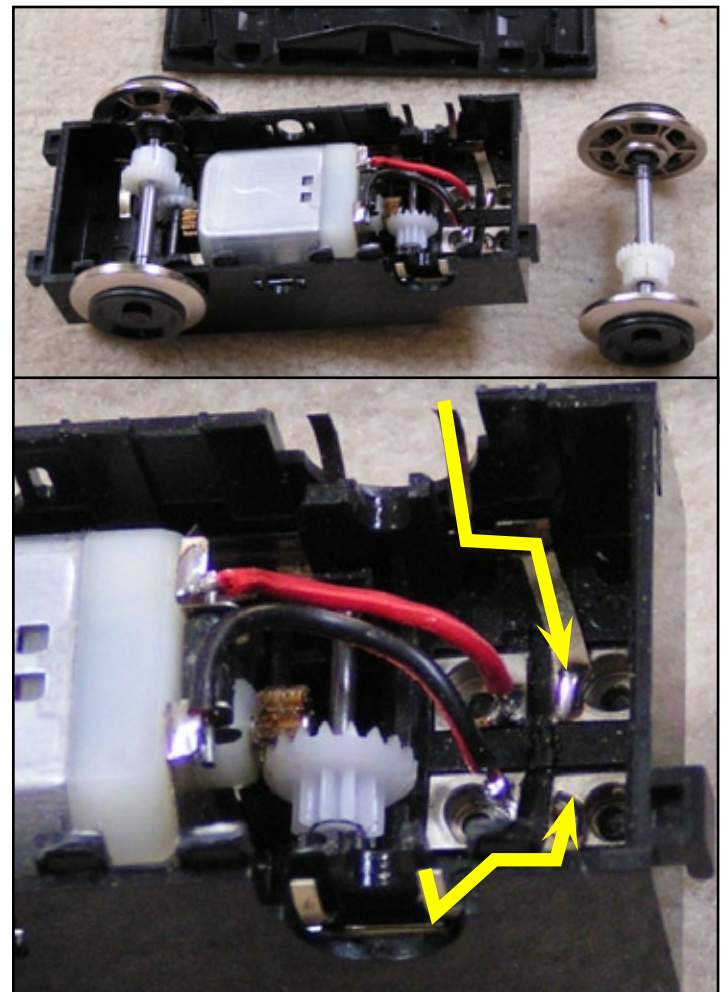


Figure 9: Rewiring of the LEGO train motor. The yellow arrows indicate the power pick-up from the wipers to the terminal. The black and red wires deliver power to the motor from the terminal.

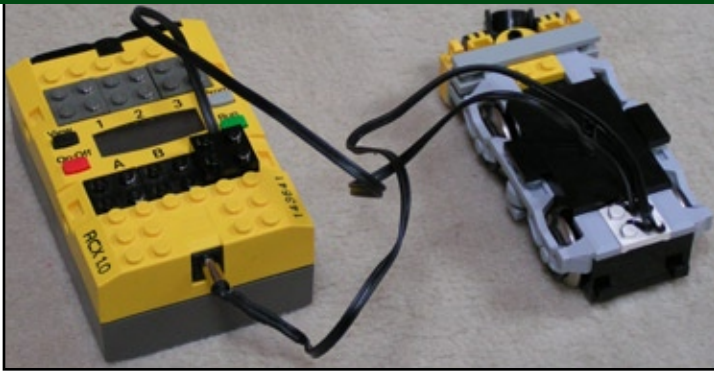


Figure 10: Custom 4-wire (2x2 wires) LEGO train motor connector. These modifications are relatively straight forward – and more importantly backward compatible with the original design.

in the photograph. Power is delivered via the wipers to the motor terminal “out”. The motor is wired to the motor terminal “in”.

The fully LEGO compatible operation of the motor is easily restored by connecting an electric wire to the modified electrical terminal of the motor. To use the motor as a separate power pickup device a custom 4-wire cable is required. Typically I am using 4x2 electric plates cut in half with a hand held rotary tool (a Dremel is one example) equipped with a thin cross cut blade. The custom 2x2 plate needs two more cuts through the metal stripes. Two 1x2 electric plates do the same job. Then a little careful soldering is required. One 1x2 plate is connected to a plug matching the RCX 9V DC power input terminal. The other plate is connected to a conventional 2x2 wire brick.

There is nothing much to say about the custom optical fibers. Virtually all cheap fibers transmitting visible or near infrared light work very well. I am using a 1mm core, 2.1 mm outer diameter plastic fiber, cutting to length with a knife edge. The ends are glued into a technic pin, friction works well also.

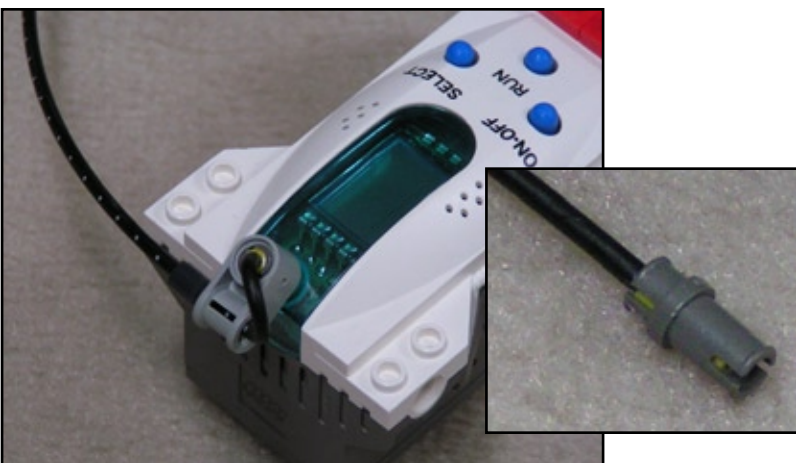


Figure 11: Fiber optical cable and terminal.

And now onto real building with LEGO bricks THE SWITCH POINT DRIVE

As already pointed out, the MicroScout is used in this automated switch as an intelligent motor. A versatile immediate VLL command is for example “Motor forward for 1 second”. The hardware needs to work properly with slightly varying “motor on times” though. This may be due to the beeping sounds the MicroScout produces every time it intends to do something – it takes some effort to do two things at the same time. The mechanical mechanism driving the lever which moves the pin on the LEGO switch is taking care of this. If the motor is at least on for moving the point blades completely into the opposite position, then everything is working well since the lever is pushed to the far end

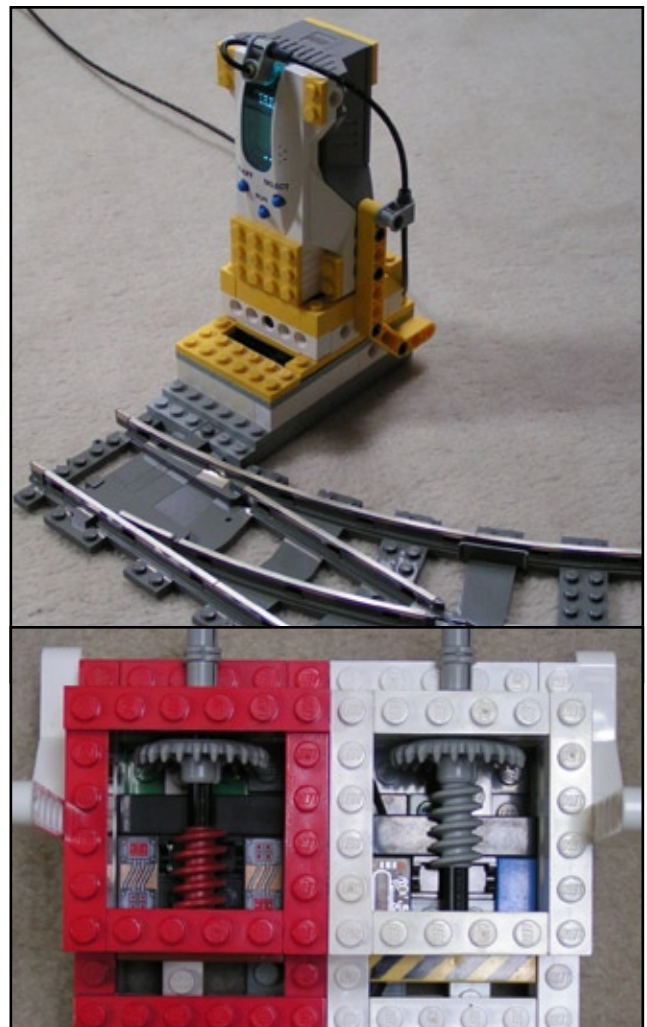


Figure 12: MicroScout operated switch point drive. The photograph on the bottom shows two tightly attached drives for automation of left and right switches that are directly attached. The “red” drive has the lever all the way pulled in and the “white” drive all the way pushed out.

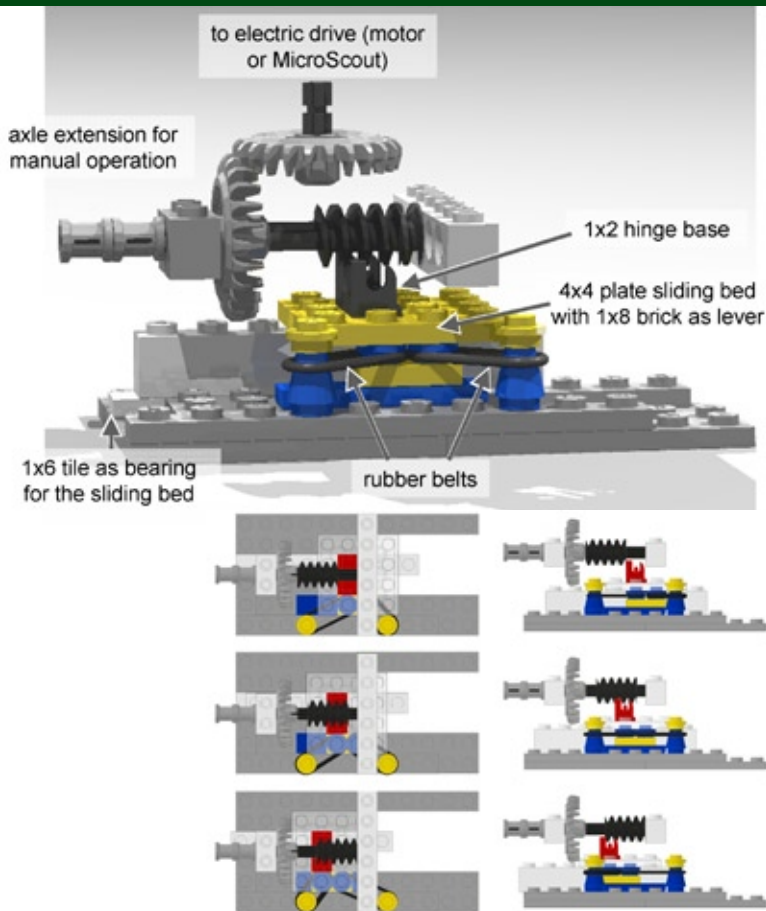
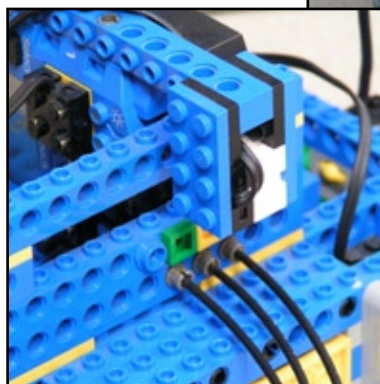


Figure 13: The switch point drive mechanism. The schematics on the bottom show how the lever moves from one end to the other. If the sliding bed is at one end point and the motor continues to spin in the same direction, the worm screw cannot push the 1x2 hinge base used as blade any further. The blade stays in close contact with the worm screw though, caused by the tension forces of the stretched rubber belt. This generates the snapping sound. When the motor spins in the opposite direction, the blade releases until the tension forces of the two rubber belts cancel. The worm screw slides freely to the other side of the axle until it cannot move further and begins to push the sliding bed against the tension force of the other rubber belt. And so on ...

by a worm drive which then loses “grip”. (Most automated switch designs that I am aware of feature an eccentric type drive where an offset section is used to power the lever in a reciprocating way – this is perfect for manual control but not for the automated design here. Every time the switch position is changed a small error is added to the final position and eventually the final switch position would be undefined). Two rubber bands take care that the worm drive keeps in touch with the lever. Thus, when the driving motor keeps going although the point blades are in their final position already, the



worm drive just produces snapping sounds.

It is of course possible to put standard 9V motors on the drive. In this case the motor may be connected directly to a Scout or RCX. However, only two (Scout) or three (RCX) switch drives can be operated in case no additional custom hardware (for example the output expander designed by Daniele Benedettelli (<http://daniele.benedettelli.com/sensors.htm>) is installed.

THE SWITCH POINT DRIVE DE-MULTIPLEXER

The Scout controlling the switch point drives has only one VLL output. The multiplexed RF or IR signal arriving from the host computer (in fact this is just a PBrick message containing the switch point number with a bit set or not indicating “straight” or “diverging”) needs to be de-multiplexed and routed to the corresponding switch point drive. For this purpose I have constructed a device shown in the photographs and MLCAD schematics in Figures 14 and 15. The basic idea is that the Scout aligns itself with the entrance of the target optical fiber that is connected to the corresponding MicroScout. Once that alignment is established, the VLL “turn motor on 1 s” command is issued, and the Scout is ready to accept another “turn switch command” from the host computer. The Scout is mounted on a sliding bed which is powered by a worm gear and a LEGO motor connected to one of the two Scout outputs. In the present design, a maximum of 8 fiber optical cables are connected to the lower 1 x 16 technic brick with holes,

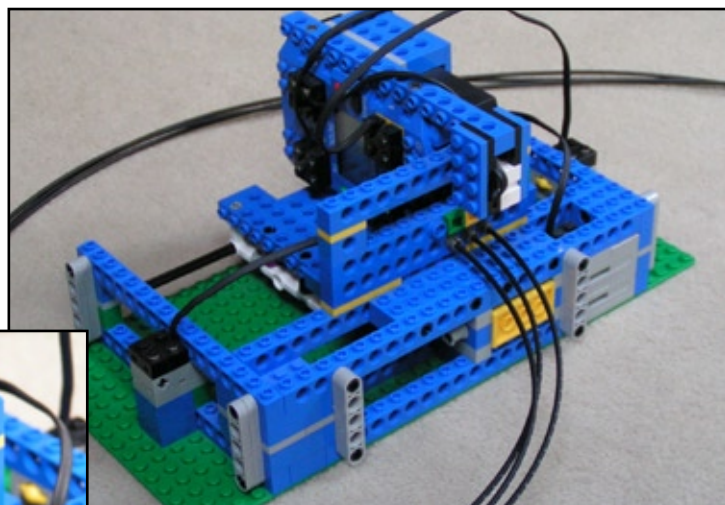


Figure 14: The Scout switch point de-multiplexer. The close-up on the left shows the VLL output channels with three optical fiber cables attached. The touch sensors on both sides of the frame are used as endpoint indicators when the device is recalibrating.

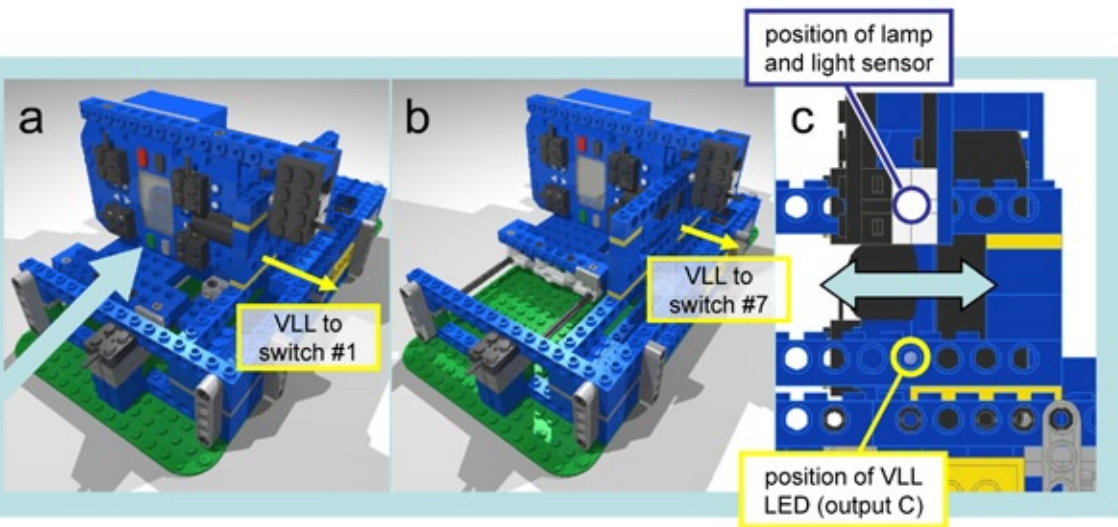


Figure 15: Controlled positioning of the Scout. The two touch sensors on the left and right of the base shut down the Scout program and float the motors – in case the sliding bed carrying the Scout has triggered one of the sensors, something went terribly wrong. This however happens very seldom; a flashlight aimed at the light sensor will certainly do though ...

which is part of the frame in front of the Scout. The upper 1 x 16 technic brick is used to align the Scout VLL diode properly. Upon receiving a “turn switch command” the lamp attached via the support to the Scout body is turned on. The lamp is mounted face to face to the Scout light sensor and in line with the holes in the upper 1 x 16 technic brick. This way, the Scout program can simply count dark-bright events and thus the number of holes that were passed upon moving to the left or right, and then stop when the light has reached a certain threshold (i.e., the lamp and light sensor are reasonably aligned with one of the holes). The VLL diode and the light sensor are mounted on one line parallel to the PBrick base. This means that the VLL diode is also aligned to the same hole on the lower technic beam. In other words, the Scout brick moves as long as it finds the right slot and issues the VLL command and the corresponding MicroScout toggles the switch position.

Further on there are some notes on the software, however, one thing is worth mentioning here. The MicroScouts automatically power down after about 10 minutes, when they don’t have anything to do in the meantime. As far as I know, there is no way to turn that firmware feature off. Since it might very well be that some switch positions are not toggled within that time, one would have to manually turn the MicroScouts back on. A way around this annoyance is to periodically address all MicroScouts and let them make a sound (this

is what they can do best). So every 9 or so minutes, the de-multiplexer sets its busy flag and starts to move from one end to the other. Besides letting the MicroScouts beep, the bright/dark light thresholds of the in-built light sensor are recalibrated and this takes care of changing ambient light conditions possibly interfering with the proper operation. The best part is

that all this business fits into the 400 byte memory of the Scout (Well, I asked Dave Baum in 2003 in an email about saving memory for this particular project, and he was willing to change the original NQC code. Instead of using the “long jump” as default jump opcode, he implemented a routine into NQC that decided whether a “short jump” opcode would do or not. That saves 2 bytes of memory every time a short jump is executed! I am very grateful to Dave for doing that for me – within 2 days or so. Otherwise, the program would simply not fit into the memory of the Scout).

RCX POWERED TRAINS AND ROLLING STOCK

No track layout makes much sense without trains. As already discussed, the trains are mostly RCX 1.0 powered. This creates some building constraints. The RCX is 8 studs wide; I prefer building 6-wide models though. These are compatible with the original LEGO models as well as with many of the fantastic creations from around the world. It follows that the RCX has to be mounted upright on the train base; 8 wide translates to 6 2/3 bricks height though – too much to reasonably fit the 6-wide scale. One way around this is to put the PBrick onto a 6 x 34 split level train base. A rail car carrying a Scout is one example (Figure 16). The Santa Fe Super Chief baggage car is another. (Figure 17)

For the construction of trains though, the split level base is of limited value, since it is tough to attach LEGO train motors – in this case, the train overall length tends to seriously violate the 6-wide scale. All my current train constructions revolve around this idea; the RCX is built into a frame constructed from technic beams. There are cut-outs for the 9V DC power plug and the IR

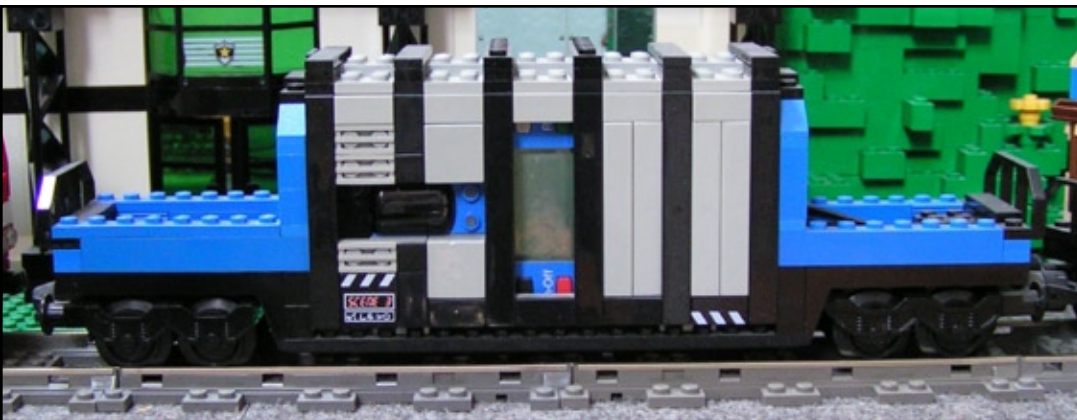


Figure 16: Split level freight train car with a Scout PBrick. The Scout operates one or two motors on the engine, which is not shown.



Figure 17: Split level Super Chief train car with an RCX PBrick. Top: The Super Chief train has PID controlled speed regulation. With reference to Figure 22: The PID controller along with the set-point input is the RCX and the IR/RF transceiver in the baggage car. The outputs are the two motors on the F7A engine, and the sensor is the rotation sensor mounted on the front bogie of the second (mail) car. The PID loop is closed by the rails.

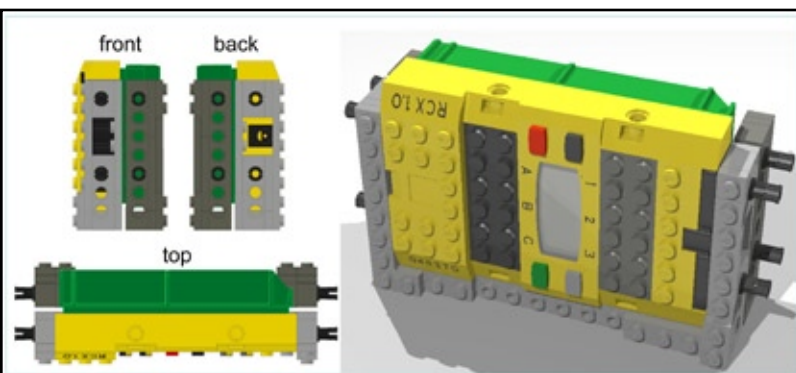


Figure 18: Technic beam frame for the construction of PBrick automated trains (and also cars). The technic friction pins may be replaced by bricks with pins already attached.

window of the RCX on both sides of the frame.

The front and back sections of the train are attached to the frame either with technic friction pins (#2780) or with pin equipped bricks (e.g., the #2458 1x2 brick). This way a number of designs are possible. Figures 19-21 show two examples: A GP40 mostly copied from the LEGO GP38 set (#10133)

and a German BR101 mostly copied from the LEGO Hobby Train set (#10183).

Some details are worth mentioning. First, having the powerful RCX on board means that real power functions are available on board. This is briefly addressed further below. Second, having an RCX 1.0 on board means that “looping back” using track “wyes” (<http://www.brickpile.com/track-layout-geometry/#wye>) is no problem whatsoever. Instead of isolating the tracks to prevent shortening of the powering circuit, just install an electrically isolating straight or curved track of the new RC train all-plastic track (I knew it; you actually can do reasonable things with the RC train track). The very moment the RCX is not powered anymore from the 9V track it simply runs on the on-board batteries. The program continues to run smoothly, the motors are still powered up and after the passage over insulating track, the RCX is back on full AC or DC again. Third, the relatively large memory of the RCX permits programming of numerous custom power functions. Light on/off is not that exciting, nor is letting the train make weird sounds. But how about speed control? True, nowadays that is something nearly every mainstream model train has. And true, it is somewhat lame that you don’t have to do anything anymore once the speed dial is set – the train runs always with the same speed, regardless of load or changing friction forces. But in this case I could build such a controller with original LEGO parts! In my research lab quite a number of such controlling devices where the data “set-point” and “actual reading” are used to adjust the output properly. They are mostly “proportional-derivative-integral” or PID controllers. But how do they really work? There are many references one can Google-up but I never really got it until I read John A. Shaw’s eBook “The PID Control Al-



Figure 20 (below): GP40 in pieces ... Left: Motor 1 and front cabin. Middle: RCX with in frame, undercarriage with wires, and motor 2. Right: Back bogey of the GB38 with the rotation sensor.

Figure 19 (above): An RCX PBrick automated GP40 / GP38 locomotive set with PID speed control. Only the GP40 is shown. PID controller with set-point input and motors are on the GP40, the sensor is mounted on the back bogey of the GP38.



Figure 21: A PBrick automated BR101 without speed control. The middle section is covered with plates and tiles.

Bottom: If the track delivers no AC/DC power (insulated wye loops, cheap all-plastic switch points ...) the RCX 1.0 switches to battery operation.

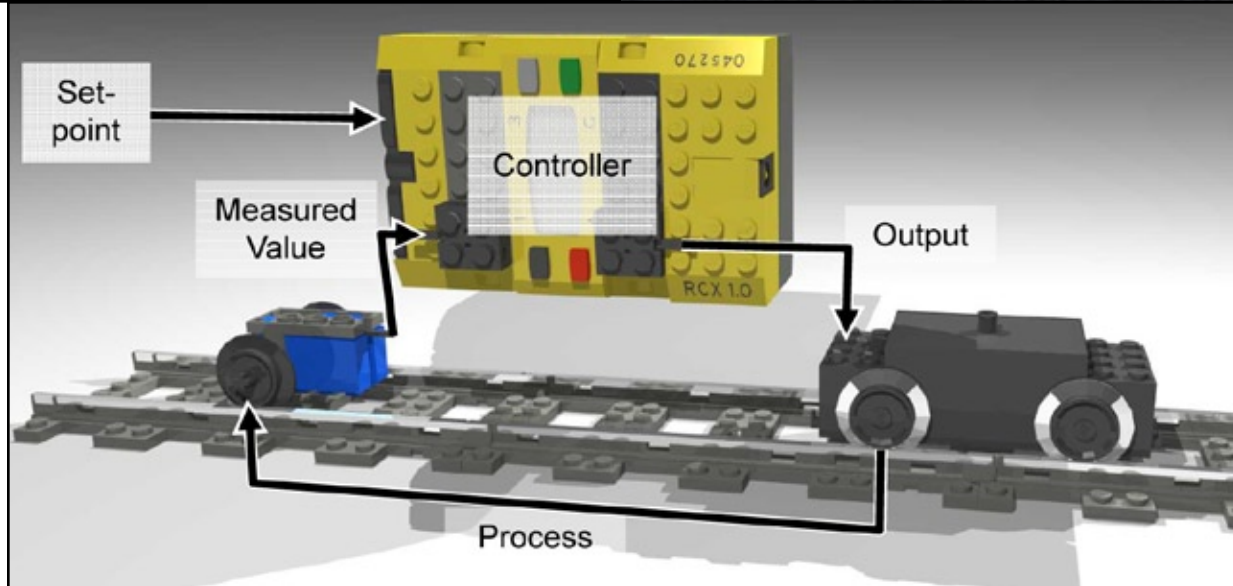


Figure 22: Principle of operation of controlling the train speed via PID.

gorithm. How it works, how to tune it, and how to use it". It is a wonderful hands-on reference with program examples written in plain BASIC. After translation into NQC with integer math, PID speed controlled train operation works beautifully. A PID controller needs a sensor input (LEGO rotation sensor), an output (the modified LEGO train motor), a set-point (sent from the host computer or from other speed controllers to the RCX), a process (motion of the train caused by the motor), and some brain handling the math and settings (the RCX). There are many, many more things you can do with a PBrick on board!

FINALLY: THE TRACK

As already mentioned, in my layout, basically all pieces of the 9V DC track are powered, avoiding short circuits by inserting insulating 9V RF pieces where appropriate. Also, the new RF train rail crossing (#7996) is not only saving an enormous amount of money compared to the equivalent of four 9V switch points, but is also saving a lot of space. In addition, it makes a perfect circuit divider. I haven't figured out how to switch that thing automatically using either a MicroScout or a plain 9V motor, but that appears to be a minor construction problem. I guess the driving unit needs to reside above the track; this could also be a good place for the nifty LEGO camera or something. Again, even the 9V RF track system has some advantages ...

Power is delivered to the tracks with conventional LEGO (#5305) connection wires. I am generally using multiple power feeds, since the voltage drop is considerably high when more than one train is drawing current from the tracks – and almost all of my locomotives are running on two LEGO train motors.

The power supply is not critical, any 9 to 12 V AC or 12 V DC will do. Some amperage is required when operating multiple trains – halogen lamp transformers are perfect, either the conventional AC types (heavy but cheap) or the electronic varieties (light-weight but considerably more expensive) will do, because of the beautiful design of the RCX 1.0. As a note, the IR/RF transceiver is fully compatible with this – designing across multiple themes means that a device – if possible at all – should be able to operate either on an active RCX input (5V switched, 10 mA max.) or on a bold 12V DC 10 A power supply.

A FEW MORE NOTES ON PROGRAMMING

There are many ways to convert a device *potentially capable* of doing things into a *smart* device. With respect to programmable LEGO PBricks (and all other micro controllers) this basically translates into creating a device consisting of smart hardware *and* smart software. Creating smart hardware calls for advanced building skills, since there are numerous bricks and pieces and virtually infinite ways to put them together. But when it comes to programming a micro controller one has to work with the possibilities and constraints of its firmware as well as accompanying programming environments. I have no idea on how to *create* both – I just play with what is available. TLC provided firmware – e.g., the LEGO byte code and other people (see above) successfully worked on that. As far as I am concerned Dick Swan has propelled the RCX firmware in view of designing across multiple themes (i.e., being compatible with LEGO byte code and thus Scout, Spybotics, Cybermaster ...) to unbelievable performance. Consequently, the NQC compatible Swan firmware (fast0618.lgo) is my preferred choice for operating an RCX. I use all other PBricks with the original firmware provided by TLC. PBricks with loaded firmware may be programmed using byte-code, a programming language such as NQC that translates the high-level language code into byte-code, or even an IDE for ease of use (e.g. BricxCC). There are many other combinations possible – even writing code in a high-level language with subsequent creation of an entire firmware from that code.

The flow-charts in figures 23 and 24 on the next page show the structures of the NQC programs running on the Scout de-multiplexer and RCX operated trains. The de-multiplexer program requires less than 400 bytes, whereas the train program is a memory monster of about 600 bytes. Without the PID algorithm containing subroutines, it would also fit into the memory of a Scout.

The PID loop code is not mandatory for RCX automated train operation. In fact, it is more fun to actively steer the train through all the various conditions due to changing friction and load forces. It should be mentioned though that the LEGO train motors do not match the output characteristics of an RCX well. The RCX uses pulse width modulation to generate the different power levels (0 ... 7 and off with the LEGO firmware or

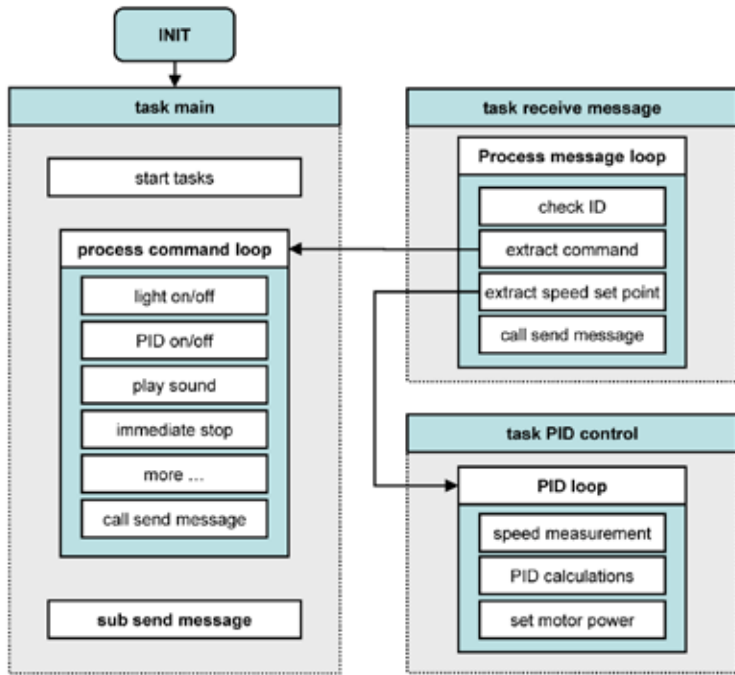


Figure 23: Flowchart of the train program.

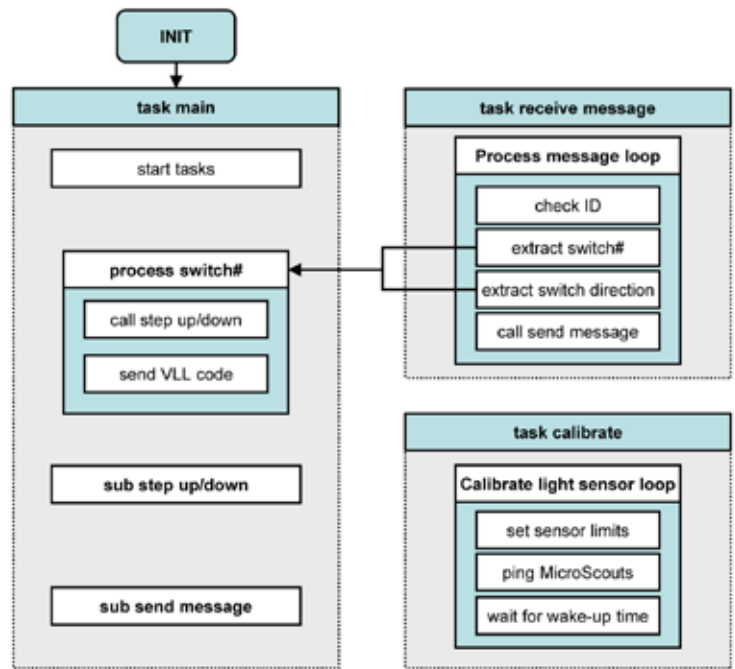


Figure 24: Flowchart of the switch point demultiplexer.

64 power levels with the Swan firmware). Let us consider the original LEGO firmware: All power levels are reasonably applicable with a geared LEGO motor, e.g., the #71427 or #43362 (a wonderful overview of the performance and characteristics of LEGO motors as well as many other exciting things are found on Philo's [Philippe Hurbain's] homepage at (<http://www.philo-home.com/motors/motorcomp.htm>), which means that even at RCX power level 0, a geared motor starts to spin and generates some torque. The LEGO train motor is also slowly spinning but only without a load. The torque generated by the train motor at this power level is next to nothing. At power levels larger than 3 the torque becomes much stronger; at power levels larger than 5 the motor runs like crazy with enough torque to derail trains very easily. Which leaves us basically with three RCX power levels: 2 ... 5. All my trains run on two train motors and the NQC program translates the setpoint power level (the transmitted speed value between 0 and 7 from the control program) to "reasonable" RCX output power levels. For example "power level 1" set in the control program is translated to power level 2 on motor 1 and power level 3 on motor 2. This works quite well. With the Swan firmware, the restrictions are far less since 64 power levels are available. Nevertheless, power levels 0 ... 28 don't do much even with a light-weight train; basically the same

arguments as above apply. But with PID control everything goes rather smoothly. If you want "power level 1" you'll get it.

Figure 25 shows the behavior of a light-weight "train". (See figure caption for details) The PID algorithm smoothly adjusts the power levels on the motors every tenth of a second. So a train moves almost with constant speed under very different external conditions, e.g., ten rail cars on a turning slope would be hauled as fast as one car on a long and even stretch of tracks. It should be emphasized again though that PID control becomes very tough to implement with the standard RCX 2.0 firmware. In fact, I could not do it. With the enhanced Swan firmware, the PID loop updates data every 10 ms (and even faster, if desired). Thank you, Dick!

My current host control program, written in Microsoft VBasic 6.0, is another long range project. Currently, it can control up-to 10 trains, 10 switch point demultiplexers, and up-to 30 switch points. The program is far from being stable, but works. Before running the program, I am designing my track layout with Matthew Bates' wonderful Track Designer software (http://www.nglrc.org/Train_Depot/td.htm). The finished layout is copied and pasted into a graphics program (I use Corel Draw) and stored as GIF file. My control program loads the layout and then the switch points are simply

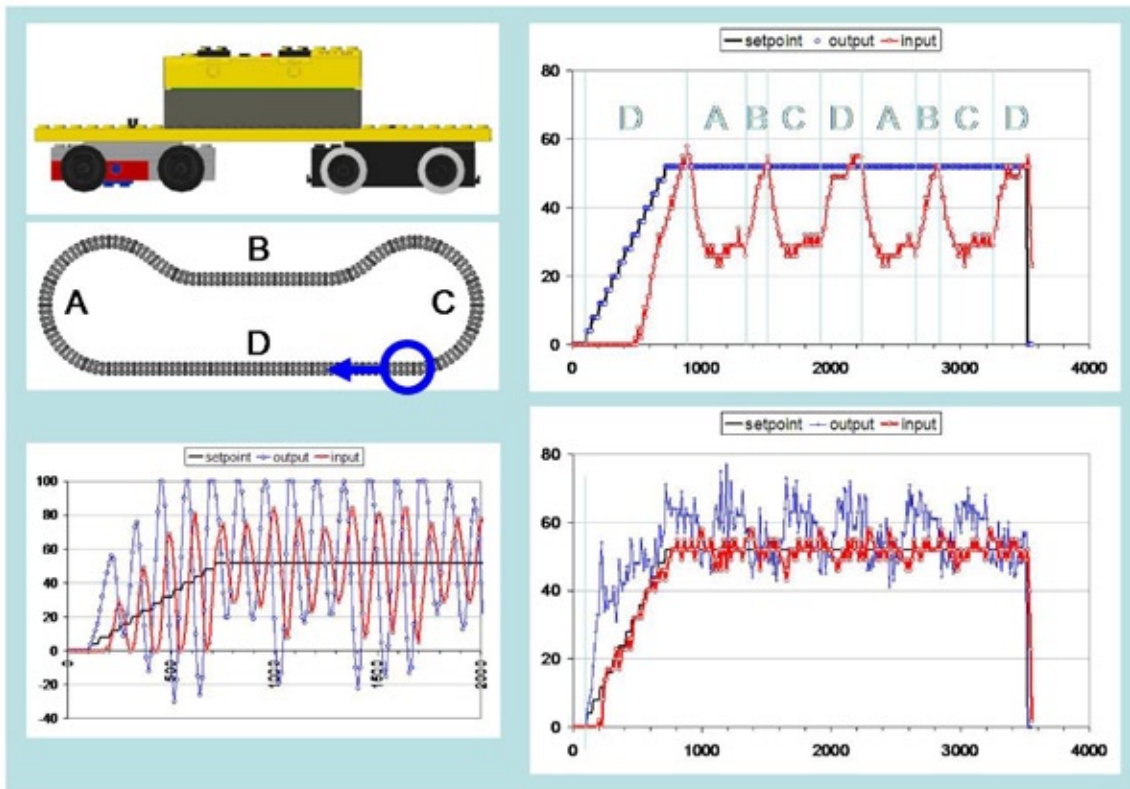


Figure 25: RCX PID control speed measurements using the set-up as shown on the top left. All data are % full power (y-axis) and 10 ms ticks (x-axis). The test-track is a simple loop. The RCX program smoothly increases the output power until it reaches a final value (“setpoint”, black line; around 50% full power). The train starts to slowly accelerate (blue circle) in the direction of the blue arrow on the straight D-section of the track. Measurements were taken with my current NQC train control program using the data log feature. Data were extracted with BricxCC and exported to Excel. Top right: No PID control. When the output power reaches about 30%, the train starts to move forward, as shown by the red speed data. When running through a curve (sections A and C), loss of speed is significant but speed increases again to reach the original final value on straight sections (B and D). In contrast, with near optimum PID settings, output power is rapidly raised until the error between setpoint and speed is minimized. The speed through curved and straight track sections is almost constant; output power however is not. Bottom left: Textbook example for wrong PID settings: The gain (without proper integral and derivative compensation) is much too high and thus the train speed starts to oscillate, as predicted by theory.

“placed” on top. Clicking on any switch symbol changes the position of the switch. Some other features currently built into the program are: grouping switches (the whole switch group changes positions one after the other just by clicking on the numbered group symbol above the switch), managing multiple PBricks (ID management, switch assignment to a particular PBrick), management and troubleshooting of communication parameters, and more. Complete switch layouts can be saved and loaded. Trains are controlled in a separate child window. The program can also “poll” trains and switch drive controllers; it simply browses through the PBrick ID space and waits for answers. If successful, a status byte is sent from the polled PBrick uniquely describing the type and capabilities (e.g., max. number of switch points, number of motors, and so on). Commu-

nication with the PBricks is by default bi-directional. The control program is asking for acknowledgement of the issued commands; if that fails, the command is resent up-to four times. This feature can be turned off. Now the program can even talk to rather dumb devices, for example the RC train base.

This is all not very exciting – and much better implemented in commercial DDC hard- and software, but we are playing with LEGO.

WHAT’S NEXT?

My track layout is far from being complete – most probably it will never reach this status. The demultiplexer is another thing that needs refurbishing. With fiber optics (even the original LEGO fiber optic cables will do) there is no reason to haul the entire Scout

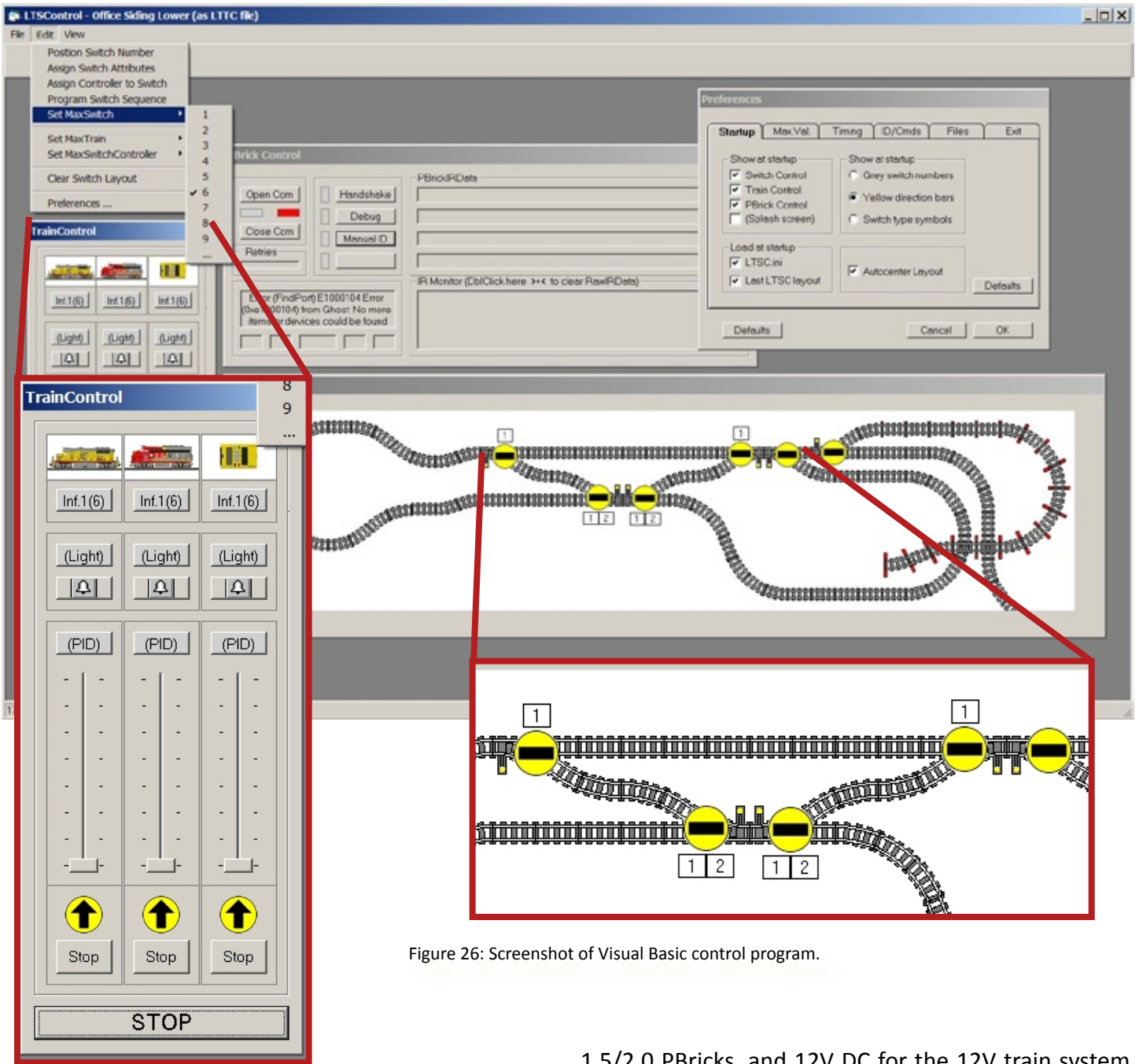



Figure 26: Screenshot of Visual Basic control program.

PBrick all the way from one end to the other. A lamp and two fibers would do. I believe that the RC train set #7897 needs some real power functions; two of those sets should result in a decent ICE train. Most importantly, there are still some ideas for custom bricks that I wish to make (and TLC should have made long ago). For example a “3/4.5/9 V DC conversion brick” with a wide range input, AC or DC. 3V DC is required for MicroScouts, 4.5 V DC for the Spybotics PBricks and the “old” 4.5 V LEGO line of electrical system, 9 V DC for the current system including 9V trains, Scouts and RCX

1.5/2.0 PBricks, and 12V DC for the 12V train system. In my present project, this custom brick could be located anywhere near the track in the range of the #5303 connection wires and deliver power to all the electrical equipment that TLC decided to save some money on by removing the AC/DC power jack. Along with this conversion brick, appropriate battery adapters are required. This is another custom brick project currently in the “design stage” that I am thinking about – maybe even for another couple of tens of years ... but LEGO lasts for ever and there is no hurry ...

Thanks for reading – and play well! 



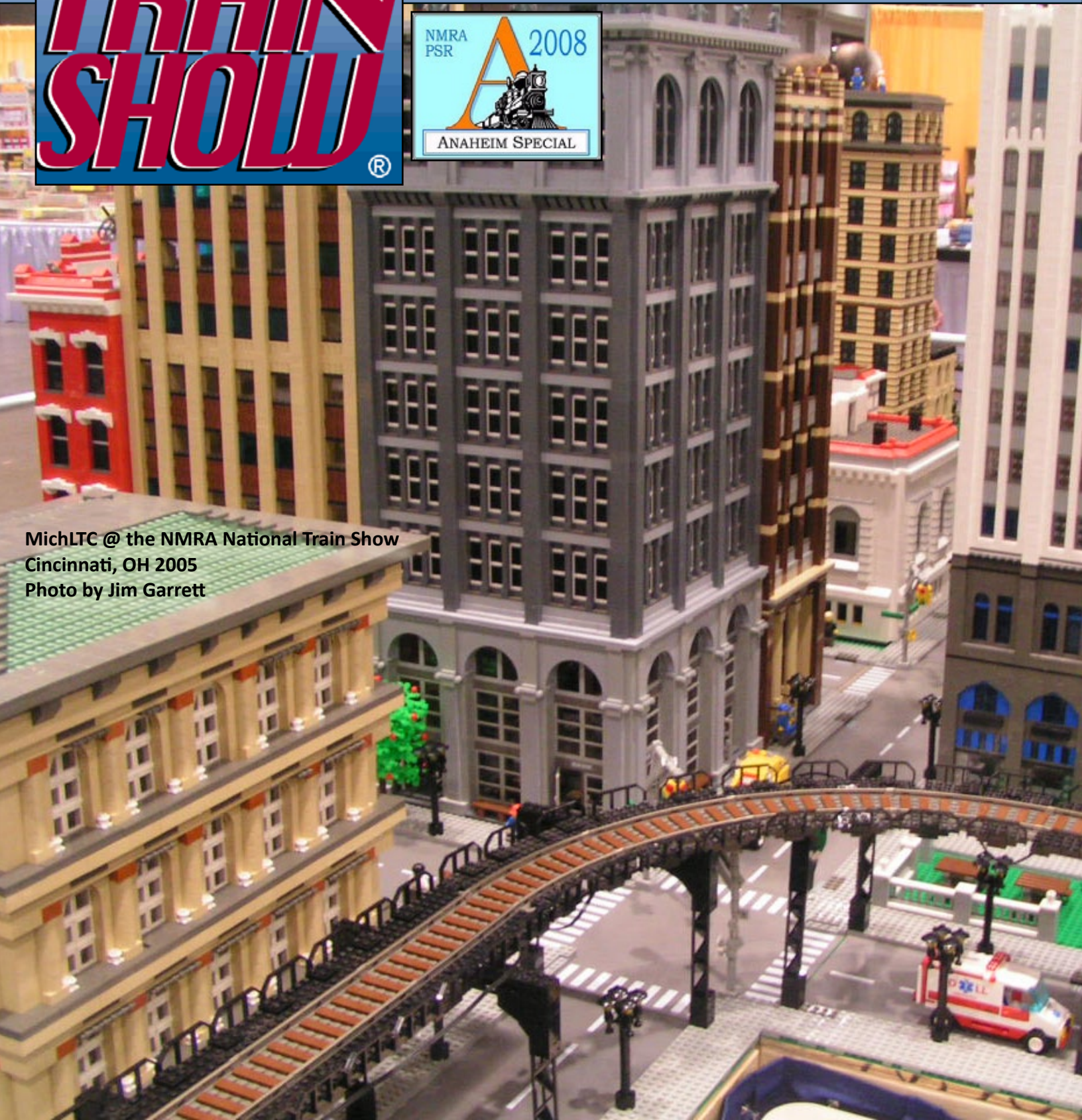
The International LEGO® Train Club Organization
NATIONAL MODEL RAILROAD ASSOCIATION®
National Train Show®
Anaheim Convention Center
July 18 - 20, 2008
Anaheim, California



THE NATIONAL TRAIN SHOW®



Visit <http://www.iltco.org> for more info.



MichLTC @ the NMRA National Train Show
Cincinnati, OH 2005
Photo by Jim Garrett