D&B Model Railroad Operators Guide

Don Buczynski

11-05-2020

Table of Contents

Introduction

Model railroading has come a long way since the early days of my childhood. Use to be that a power pack that provided DC voltage for the trains and AC voltage for the accessories was all that was needed. Operation involved many manual procedures to position track switches and block power controls. Over time, model railroad layout controls have evolved to handle many of the low-level operational details. In many ways, operating a model railroading has become easier. The overall enjoyment and satisfaction from the hobby has greatly improved. A model railroad can now easily operate in a manner that closely simulates the full size prototype. Most model railroad clubs operate in this way.

Though the D&B is a simple model railroad, operation requires more than just powering up the layout and turning the throttle knob. The model locomotives operate by a system called *Digital Command Control*, DCC for short. This permits one or more trains to operate simultaneously and independently on the layout. The D&B layout utilizes a commercial product called *Empire Builder* that is manufactured by the company *Digitrax*. An old system, yes, but it still works.

In addition to the electrical power for the locomotive, DCC also places a digital messaging signal on the track. This standardized signal is monitored by a DCC decoder that is installed within the locomotive. The decoder is configured with a layout unique address, typically the last two digits of the locomotive number. In this way, the locomotive responds only to DCC message data sent to its own address and ignores everything else. The decoders are user configurable and can be set to behave in varied and highly precise ways. DCC has many capabilities and eventually, a read of the Digitrax User manual is recommended to better understand and operate the D&B.

The stuff that is unique to the D&B is the means by which turnouts (track switches), trackside signals, and track power polarity are controlled. Originally, the D&B used multiple microprocessors called Basic Stamps. (The D&B was even featured in the Parallax 2004 product catalog, see appendix.) Now, a more powerful hobby microprocessor called a Raspberry Pi, RPi for short, handles the automated aspects of the layout. Off-the-shelf RPi compatible interface products, custom designed circuitry, and a few dozen servos complete the system. The RPi runs custom software that uses input from infrared (IR) sensors, track block occupancy detectors, and user keypads, to position turnouts and set appropriate trackside signaling.

As a final note of introduction, for those looking to enhance their knowledge of electronics and software programming, refer to the companion volume *D&B Model Railroad Raspberry Pi Control*. It contains far greater detail about the inner workings of the automated portions of the D&B layout. While many of these automations could have been achieved using off-the-shelf commercial model railroad products, the author preferred a more do-it-yourself approach as a means to greater knowledge and satisfaction from the hobby.

D&B Layout

The D&B model railroad is organized into track sections and power blocks. Each section has dedicated software which automates and supports train operations. Some of the automations rely on consistent train movement rules; for example, the train travel direction through the holdover loops. Such rules are a compromise between operational variability and the control system's complexity. With some planning, and efficient use of the Holdover, Midway and Yard tracks by the engineer, four concurrent trains on the layout are easily supported when you're in the mood. At other times, "watching 'em roll" with one or two trains is a good way to show the layout to casual visitors. An unseen twenty car train emerging from a hidden holdover track and snaking its way up the mainline makes a good impression.

Mainline

The mainline is all track beginning at the lowest elevation of the layout (the holdover tracks) and ending at the T08 and T11 yard entrance turnouts. The mainline is further divided into three major sections that support specific operational needs (detailed later). The remaining mainline tracks interconnect these sections. The mainline consists of ten power blocks (Bx) that are isolated by electrical gaps in the rails. Each end of the mainline is a reverse loop and input to the software code provides for automatic rail power polarity control. A train can traverse the mainline track without the need of operator input.

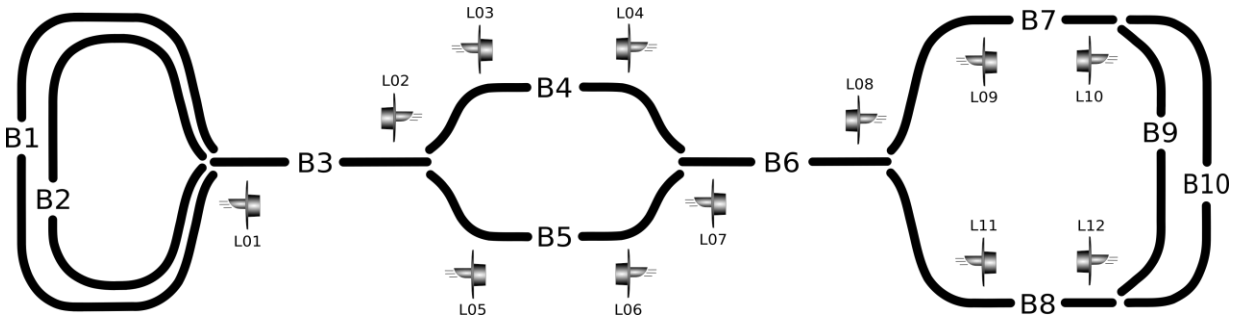


Figure 1: Mainline Block and Signal Diagram

Each power block is monitored by a detection circuit. When a power drawing locomotive or car is present in the block, the detection circuit reports the activity. This input is used by the software code for signaling purposes. The following rules are used to illuminate the mainline signals.

Signal	Condition
Off	Unoccupied block not being approached.
Green	Approaching an unoccupied block.
Red	Approaching an occupied block.
Yellow	Approaching an unoccupied block, subsequent block is occupied.

As an example, for a train located in power block B3, L01 and L02 will be red, L03 and L05 will be green, and L04 and L06 will be yellow. To maintain an active block after the locomotives move forward into the next power block, the last car of the train can be equipped with resistor wheels in its trucks. The resistor wheels draw a small amount of electrical power which maintains the block detector activity and signals.

Holdover Section

The tracks involved with this section are hidden and used for train holdover operations. That is, trains seemingly leave the layout for other distant locations and then return some time later. Two sidings are available. Each has a separate power block (Bx) and infrared sensors (Sx) for detection of train position. Three turnouts are used to direct trains into and out of this section. Each siding can accommodate a train length of about twenty cars.

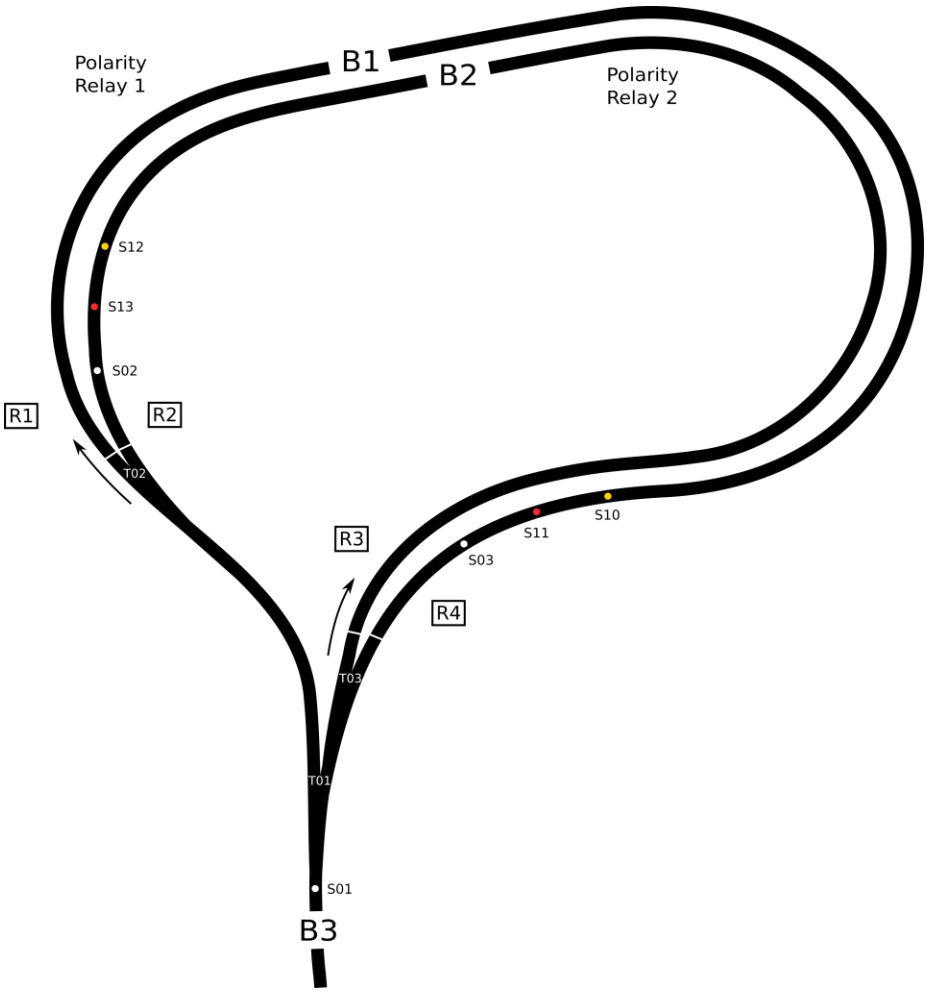


Figure 2: Holdover Track Diagram

Inbound train movement from block B3 toward the holdover tracks is detected by sensor S01. The software code uses the following rules to route an inbound train.

1. If blocks B1 and B2 are both unoccupied, alternate routing of the train into B1 and B2.
2. If unoccupied, route the train into block B1 by setting turnouts and polarity relay 1.
3. If unoccupied, route the train into block B2 by setting turnouts and polarity relay 2.
4. If both blocks B1 and B2 are occupied, sound the train wreck warning!

Inbound trains use the 'straight' side of turnouts T02 and T03 to help minimize derailments. This results in trains always moving clockwise through block B1 and counter-clockwise through block B2.

Yellow and red panel LEDs provide an indication of train position within each hidden holdover track. As a train approaches the end of the siding, first the yellow and then the red panel indicators will flash. If the train is to be stopped within the holdover track, do so as soon as the red indicator begins to flash.

Outbound train movements are detected by sensors S02 and S03 which are located at the exit ends of blocks B1 and B2. The software code uses the following rules for an outbound train.

1. Set turnouts and power polarity relay based on activated sensor.
2. Ignore sensor S01 activity for the outbound train.

The holdover section forms two reverse loops in which a train enters and then departs on the same B3 lead track. Reverse loop operation requires that for an inbound or outbound movement, the rail polarity must match while the power drawing equipment transits the rail gaps. Otherwise, a short circuit will occur and the Digitrax DB150 booster will automatically turn off track power to protect the circuitry. The software code sets the B1 and B2 power polarity relays as required.

Operationally, the two loops in the holdover section are generously spaced so long cars and locomotives can safely operate on the B2 loop without sideswiping a train on the B1 loop.

Midway Section

The track involved with this section provides a place for mainline trains to pass each other. The turnouts in this section simulate full size prototype turnouts that are mechanically set to a specific position. When entering from the point side of the turnout, trains move onto a preset siding track. During siding exit, the turnout point rails are pushed open by the wheels of the train. Since model railroad cars are too light to reliably push open point rails, software code sets the turnout position based on sensor input.

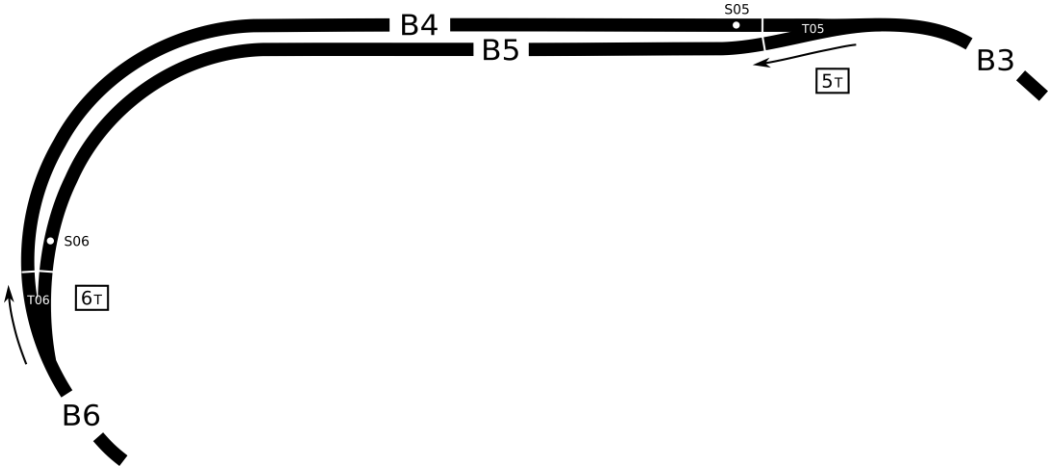


Figure 3: Midway Track Diagram

On the D&B, normal turnout position routes a train approaching T05 from B3 onto siding B5. A train approaching T06 from B6 will normally be routed to siding B4. A train leaving B4 or B5 will be detected by sensor S05 or S06. The turnout rail points of T05 or T06 will be set to direct the train back onto the mainline. About fifteen seconds after the last car transits the sensor, the turnout will be repositioned to

normal. Avoid stopping a train that leaves the sensor unblocked and cars of the train still occupying the turnout.

Operationally, for two trains to pass within the midway section, one of the trains must be short enough to fit completely within a siding. Also keep in mind that the curves in this section are closely spaced. Long cars and locomotives moving through siding B5 may sideswipe a train on siding B4. For passing maneuvers, trains with long equipment should be directed onto the B4 siding.

Wye Section

The track involved with this section forms a wye. The legs of the wye are the approach tracks leading to opposite ends of the yard. A reverse loop is formed that includes track blocks B7 through B10 and all of the yard tracks. Collectively, these blocks are individual only for the purpose of signaling. Tracks leading to and including the yard tracks from turnout T07 are wired to a power polarity relay. Sensors are used by the software code to detect train position.

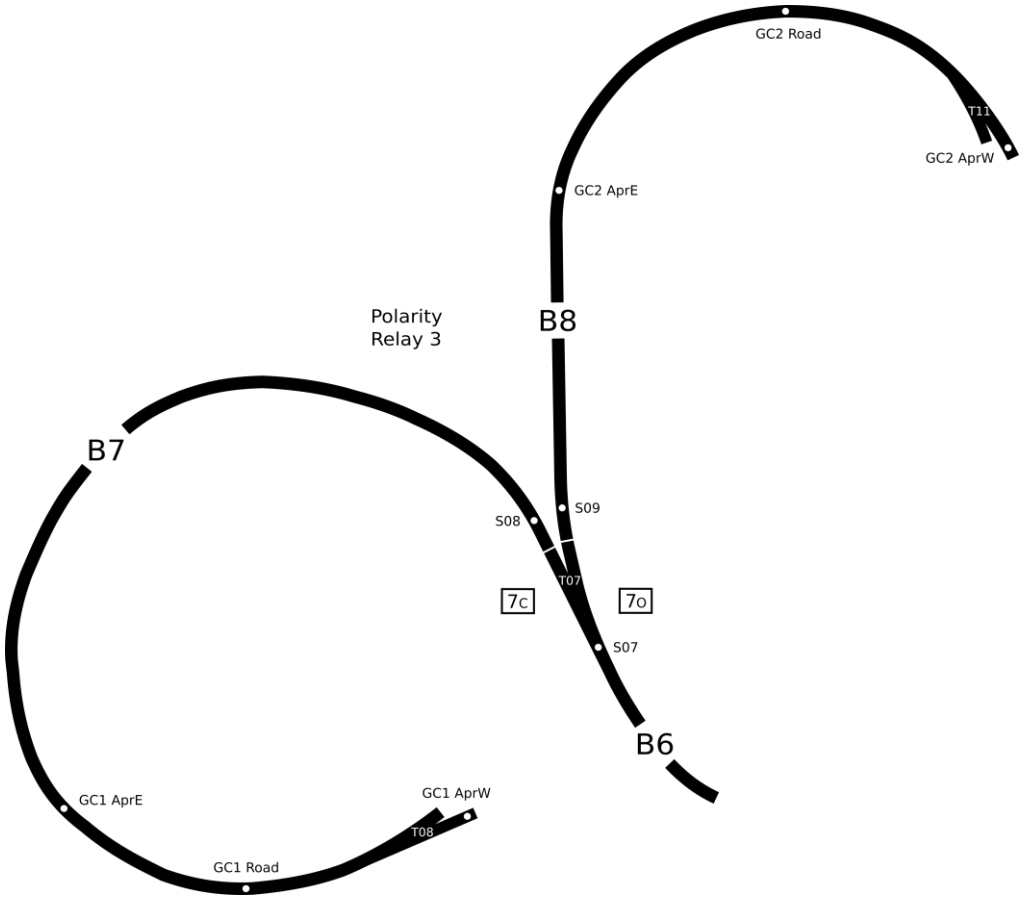


Figure 4: Wye Track Diagram

A train approaching T07 via power blocks B7 or B8 is detected by sensor S08 or S09. The software code sets the T07 point rail position and power polarity relay appropriately. A train approaching T07 via power block B6 is detected by sensor S07. The software code sets only the power polarity relay based on the current point rail position of turnout T07.

Yard Section

The yard section includes classification, industrial, and turntable tracks. These tracks are each assigned a number, 1 through 16. This number corresponds to the 16 button keypad. Power blocks B9 and B10 are used only for signaling and have no affect on yard operations.

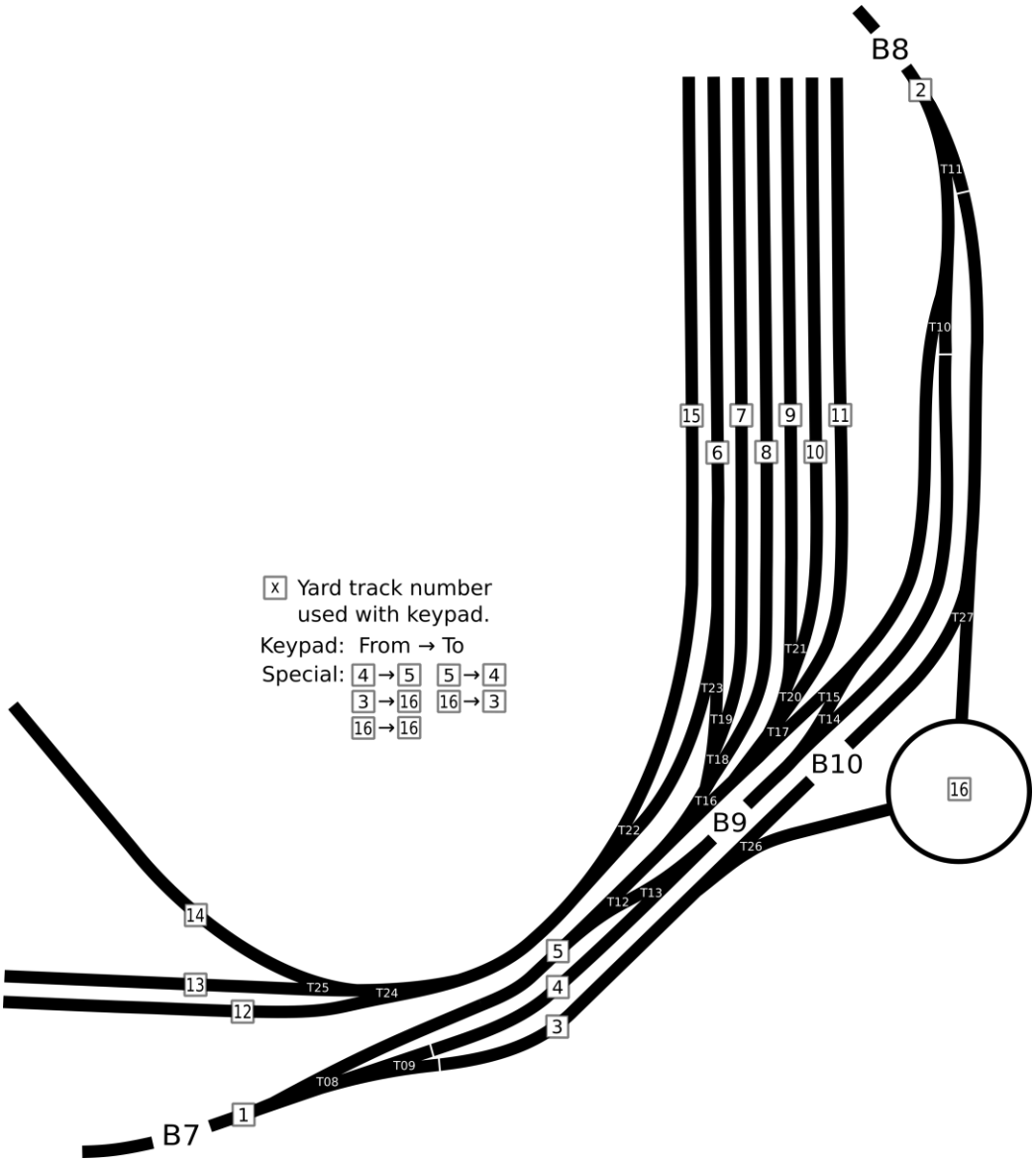


Figure 5: Yard Track Diagram

Setting of the yard turnouts is detailed in the Yard Turnouts section of this document.

Grade Crossings

Two grade crossings are modeled in the wye section. Both have flashing signal lamps and grade crossing bell sound effect. Grade crossing 2 includes servo actuated gates. As shown in the Wye Track Diagram, three sensors for each grade crossing are used to detect train position and activate/deactivate the

signals. Operational control of the grade crossing is a bit involved. See the source code for a detailed description.

Cleaning the Track

Cleaning the oxidation from the track rails will be required from time to time. If the layout is operated often, the metal wheels on the cars will help to minimize the frequency of cleaning. There are also two Southern Pacific boxcars that have a fiberboard *scuffer* installed between the trucks. They help to keep the track free of oil and dust when included in a train. The scuffers can be easily cleaned with some alcohol and paper towel. A couple of track cleaning cars have been tried but their results are little better than the scuffers.

If the layout is idle for a month or more, a thorough track cleaning will be required. Cleaning involves rubbing the rails with a track cleaning product called a *Bright Boy*; a few are kept in the parts drawers. It is basically an eraser embedded with some very fine soft metal particles. Don't use sandpaper; the much harder grit is damaging to the soft metal track rails. Slide the flat side of the Bright Boy along the rails with some moderate pressure. The narrow side can be used lightly on the turnout moveable point rails.

The hidden tracks can be reached from under the layout without too many contortions. The Bright Boy will slide a bit easier once the oxidation is removed; usually 4-5 passes along the track is sufficient. The harder part is not doing damage to the scenery and electronics close to the track. Work slowly and carefully. The mainline track needs less effort due to higher train traffic. The yard and industrial tracks will need more effort. It usually takes about 30 minutes to do the job.

The locomotive wheels generally don't require separate cleaning. When first used after long storage, the locomotive will need some coaxing to move and a bit higher throttle setting. But the wheels will clean themselves during a few circuits of the mainline and smooth operation will be restored.

A small keyboard vacuum can be used to sweep up dust from the layout. Pass it along the track to remove any loose scenery and gravel. This could result in jammed moveable points at one or more of the turnouts. Run the turnout test using the console mode command: **DnB.pl -t 1:27**. Check that the moveable point rails contact each fixed rail with no gap. Use a small brush or toothpick to clear debris lodged between the moveable rails, wing rails, frog, and ends of the tie bar.

Layout Power

The layout is powered by a number low voltage power supplies that are plugged into a surge protection power strip. All layout power is switched on/off using the power strip switch which doubles as a safety circuit breaker. It is safe to access by reaching under the electronics shelf beneath the Digitrax DB150. The DB150 uses a separate 115 VAC to 14 VAC transformer that is connected through a dedicated fuse to the power strip. If the DB150 does not power on, check the fuse. The fuse holder illuminates when the fuse is blown.

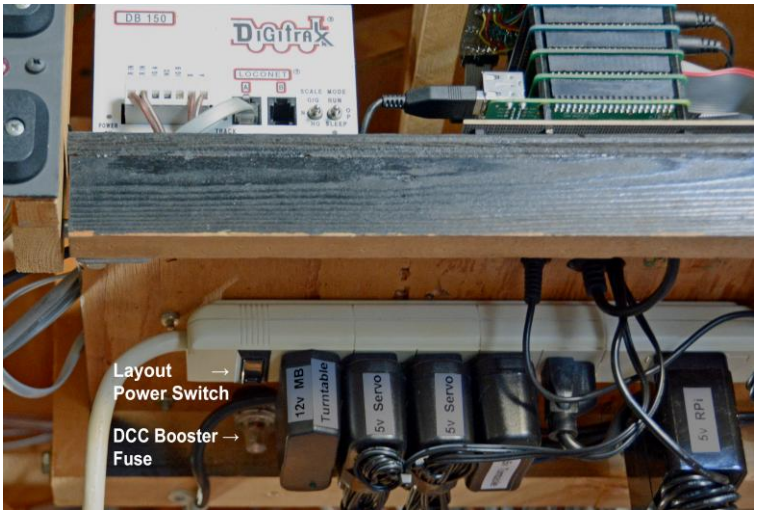


Figure 6: RPi Electronics Power Supplies

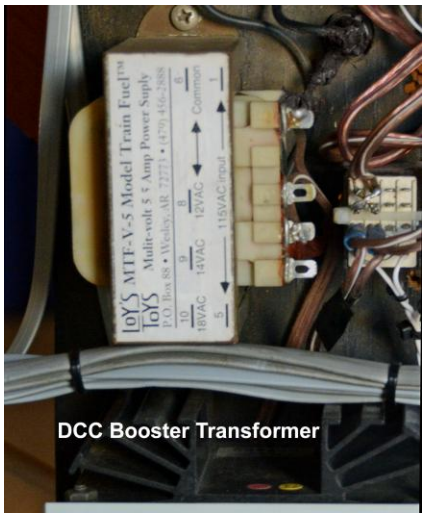


Figure 7: DB150 Booster Transformer

Starting at the switch end of the power strip, the purpose of each power supply is as follows.

- 12 volts for mainboard sound modules, turntable, and oil pumpjack.
- 5 volts for lower servo hat; servos 1-16.
- 5 volts for upper servo hat; servos 17-32.
- 5 volts for turntable.
- 120 VAC connection to DB150 transformer.
- 5 volts for RPi and associated electronics.

Power On

Use the switch on the power strip to apply power to the layout. The DB150 will emit a power on tone and its green power on indicator will illuminate. The Digitrax DT400 throttle has provision for an internal 9 volt battery. During power on or connection of the tether cable, the throttle will display the battery voltage. The battery is not required on the D&B. It is used for untethered infrared throttle operation which is not currently wired on the layout. On the DB150, LOCONET A is used to connect the remote DCC panel by the viaduct. LOCONET B is used to connect (tether) the DT400 throttle.

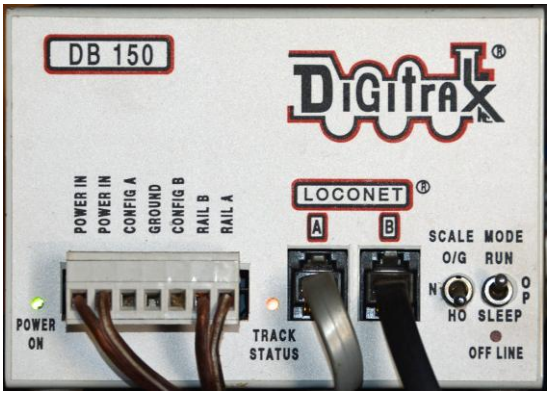


Figure 8: DCC DB150 Booster

Figure 9: DCC DT400 Throttle Battery Power

Verify the SCALE switch is in the HO position. Then set the MODE switch to the RUN position. The TRACK STATUS indicator should illuminate yellow. If not, refer to the troubleshooting section.

A number of LEDs will illuminate on the electronics board stack. After about 20 seconds, during which time the RPi green activity LED will flicker prominently indicating that the software is booting, the DnB.pl program will sound a power up tone. At this point, the program is ready to accept keypad input. It will begin processing block detector and sensor inputs and activate trackside signals based on those inputs.

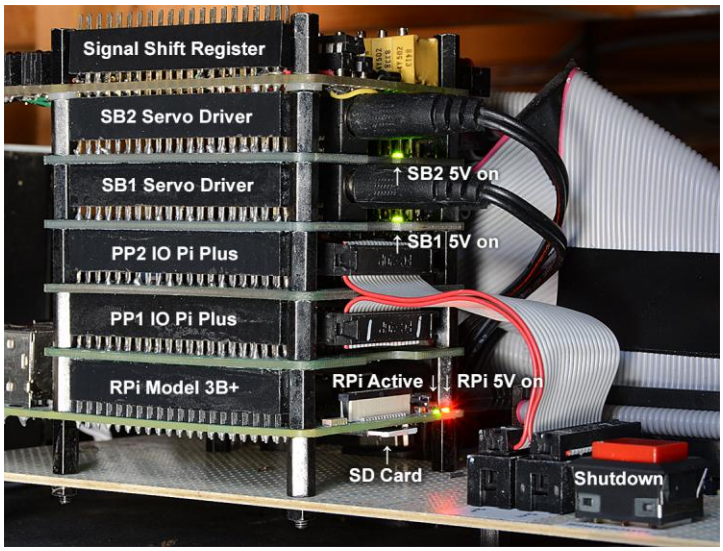


Figure 10: RPi Electronics Circuit Boards

The DnB.pl program is configured to automatically start following Raspbian OS boot. Press and hold down the shutdown button during the boot period to terminate the DnB.pl program. DnB.pl termination will be acknowledged by a double confirmation tone after which the shutdown button can be released. USB wireless keyboard/mouse and HDMI connected 1080p monitor can be used to launch a Raspbian OS command line session to perform maintenance and testing as needed. Refer to the appendix for more information.

Power Off

The DnB.pl program must be shutdown before removing powering from the layout. This will prevent corruption of the RPi microSD memory card by performing an orderly shutdown of the layout electronics and Raspbian OS.

1. Place the DB150 MODE switch into the SLEEP position. The TRACK STATUS indicator will extinguish, the OFF LINE indicator will illuminate red, and the POWER ON indicator will flash.

2. Press and release the shutdown button. This initiates a 10 second countdown during which five tones will sound. At the end of this delay, the DnB.pl program performs an orderly shutdown of the software processes and places the hardware interfaces into a safe condition. Safe condition serves to help protect the servos, sound modules, and signal lamps should the layout power

remain on for an extended period. During the countdown, shutdown can be aborted by another press of the shutdown button.

3. Once the RPi green activity LED is no longer flashing, about 10-15 seconds, it is safe to power off the layout electronics using the switch on the power strip.

Positioning Turnouts

The track turnouts on the D&B use miniature R/C servos to move the turnout points to the open and close positions. The servos are driven by a PWM (pulse width modulation) signal that is under control of the DnB.pl software. During servo installation, the open and close PWM positions were configured using a procedure that is detailed in the appendix. Open position is defined as the path that leads into the diverging or curved track. Closed position is the path that leads into the main or straight track. Once configured, turnout operation need only be concerned with setting an Open or Closed position. Sensor and keypad input is read by the DnB.pl program and used to position the turnout points.

Holdover Turnouts

Figure 11: Holdover Turnout Positioning Buttons

During normal operation, the turnouts of this track section are automatically positioned by the DnB.pl program based on block detector and sensor input. However, a keypad control is available for setting these turnouts to a specific route, **R1** through **R4**. Use of this control may be needed if a train derailment occurs involving these turnouts. When a route is selected, the following occurs.

1. Turnouts are positioned for the specified route.
2. S01, S02, and S03 sensor activity is ignored.
3. A confirmation tone is sounded and the keypad indicator is illuminated.

A holdover route will remain set with automated routing suspended until one of the following occurs.

1. One of the keypad buttons is pressed.
2. There is no S01, S02, or S03 IR sensor activity for 60 seconds.

Midway Turnouts

Figure 12: Midway and Wye Turnout Positioning Buttons

During normal operations, turnouts T05 and T06 in the Midway section are automatically positioned by the DnB.pl program based on sensor input. To facilitate special train movements, these turnouts can be

manually positioned using control buttons **5T** and **6T**. These two buttons can generate single and double press input. A double button press is registered when the second press is within 1 second.

A single button press will toggle the turnout position with each press. For example, manually toggling T05 to Closed will direct an approaching B3 train onto siding B4. As when moving in the normal travel direction in this siding, after the last car transits the sensor, T05 will be repositioned back to Open.

A double button press will toggle and then lock the turnout position. No automatic turnout reposition after sensor transit will occur. This is useful, for example, if a train is to be stopped on a siding for an extended period. A single control button press will unlock and reposition the turnout.

Tones are sounded to confirm each button press. If a turnout cannot be positioned due to a conflicting train movement, the button ignore tone will sound. Retry the turnout positioning after fifteen seconds.

Wye Turnout

Control buttons **7O** and **7C** are used to manually set the position of turnout T07, Open or Closed, for trains approaching T07 from power block B6. An active sensor S08 or S09, caused by a train that is approaching T07 from power block B7 or B8, will immediately override any manually set T07 position.

Yard Turnouts

Figure 13: Yard Turnout Positioning Buttons

A train moves through the yard via a route. A route is specified by keying in two track numbers. The first number entered is the ***from*** track. It is the track currently occupied by the train. The second number entered is the ***to*** track. It is the desired destination track for the train. Once both track numbers are input, the turnouts for the specified route will be set appropriately. Key combinations that do not correspond to a valid route will be ignored and an error tone will sound.

Keypad entries are acknowledged by a tone and a keypad LED illuminates when a ***from*** track has been entered. If the ***to*** track is not entered within 5 seconds, the ***from*** entry is discarded.

Yard routes were devised in an effort to balance flexibility and minimize keypad entries. For example, for a train on approach to the yard on track 2, to pass through the yard on track 4, the approach track

(4/2), middle track (4/4), and departure track (4/1) must be specified. This allows for a concurrently departing train on track 3 (e.g. 3/1) while the approaching train is routed to track 4.

After some operating sessions, the need to specify the middle route for a yard pass through was deemed unsatisfactory. Some additional functionality was implemented. In addition to the three entry method, if the approach and departure routes are specified within 5 seconds of each other, the middle route will be automatically set. Yard tracks 3, 4, and 5 utilize these two route entry methodologies.

Note that whenever turnouts T12 or T13 are involved in a route, both turnouts are similarly set. Having one of the pair in opposition is not useful. The same is done for turnouts T14 and T15. Additionally, the following keypad input is supported by the yard code.

- Entering the same number for the ***from*** and ***to*** tracks (e.g. 5/5) will route just the specified track.
- Route 16/16 will open the turnouts T12 through T15 for a run around operation. A consecutive 16/16 entry will close turnouts T12 through T15.

There are some special cases handled by the yard code. These involve ***from***/***to*** tracks that are dependent on train movement direction. Since train direction is unknown, the code initially sets the turnouts for a left-to-right route movement relative to the figure 5 yard diagram. If the same ***from***/***to*** input is consecutively entered, the code sets the turnouts for the right-to-left route movement.

- Routes 3/16 or 16/3
  Initial: T26 open
  Consecutive: T27 open
- Route 5/4
  Initial: T12 and T13 open
  Consecutive: T15 and T14 open
- Route 4/5
  Initial: T14 and T15 open
  Consecutive: T13 and T12 open

Turntable Operation

The turntable uses the "legacy" Basic Stamp controller.

Figure 14: Turntable Positioning Buttons

A track number and turntable bridge end is entered by the operator using the 12 button keypad. This results in the turntable bridge rotating to the specified track. When the turntable is not in use, about every five minutes, the bridge is randomly moved to simulate area activity.

Two modes of turntable operation are provided, program mode and operate mode.

Program Mode

Program mode is used to enter turntable bridge position data. This data is saved and used during operate mode to set the turntable bridge to the user specified track. To program a position, buttons are used to rotate the bridge. Once in position with rails aligned, the position is associated to a turntable track number and saved. The following summarizes the keypad buttons and their corresponding functions in program mode.

Button	Function
0	Find home position and calculate circle size.
1	Micro-step turntable bridge counter-clockwise
2	Head-end of turntable bridge track aligned (default)
3	Micro-step turntable bridge clockwise
4	Step turntable bridge counter-clockwise
5	Tail-end of turntable bridge track aligned
6	Step turntable bridge clockwise
7	Slew turntable bridge counter-clockwise
8	Remove current position data for track number/alignment
9	Slew turntable bridge clockwise
**	Enter/exit program mode
#x	Associate current bridge position and head/tail input with specified track number.

Turntable motion keys 1, 3, 4, 6, 7, and 9 will auto-repeat when held down for more than 1/2 second.

The following warning tones may occur in program mode.

1 tone - Circle size too large. Need less steps in circle.
2 tones - Unknown turntable bridge position or circle size.
3 tones - Invalid keypad input.

Gear Backlash Consideration

Drive gear backlash must be taken into account when programming the bridge track positions. The final rail alignment stepping should be done in a clockwise direction. The program incorporates a constant that is used to account for gear backlash. In operate mode, the value of this constant is used to overstep the specified track position when moving in a counter-clockwise direction. The turntable bridge is then immediately stepped the same amount in the clockwise direction resulting in the final position. In this way, the effect of the gear backlash is eliminated.

During bridge programming, a locomotive or equivalent amount of weight should be placed on the bridge track. The weight has a slight effect on the final bridge track position due to bearing friction.

Program Mode Example

The following example illustrates the steps used to input the information for an approach track and two service track positions. The approach track is assigned to button 1 and the service tracks to buttons 2 and 3. Service track 2 is then removed. For reference in this example, the head-end of the bridge track is the end with the operators shack.

The keypad buttons **1**, **3**, **4**, **6**, **7**, and **9** are collectively referred to as the **move** buttons.

1. Press the ***** key twice to enter program mode.
2. Press the **0** key. This finds the turntable bridge home position. If not found, the turntable bridge will rotate continuously in a counter-clockwise direction indicating a wiring or sensor problem. The turntable bridge is then rotated once in a clockwise direction to determine the circle size.
3. Use the **move** buttons to rotate the turntable bridge and accurately align the head-end bridge track with the approach track. The move to final alignment should be in a clockwise direction. Press the following buttons in order: **2 # 1**
4. Use the **move** buttons to rotate the turntable bridge and accurately align the tail-end bridge track with the approach track. Press buttons: **5 # 1**
5. Use the **move** buttons to rotate the turntable bridge and accurately align the head-end bridge track with storage track one. Press buttons: **2 # 2**
6. Use the **move** buttons to rotate the turntable bridge and accurately align the tail-end bridge track with service track one. Press buttons: **5 # 2**
7. Use the **move** buttons to rotate the turntable bridge and accurately align the head-end bridge track with storage track two. Press buttons: **2 # 3**
8. Use the **move** buttons to rotate the turntable bridge and accurately align the tail-end bridge track with service track two. Press buttons: **5 # 3**
9. To remove the head-end aligned entry for storage track two, press buttons: **8 2 # 3**
10. To remove the tail-end aligned entry for service track two, press buttons: **8 5 # 3**
11. Press the ***** key twice to exit program mode.

Operate Mode

In operate mode, two button presses are used to rotate the turntable bridge track to a programmed position. The first button press is the desired track position. The second button press is the turntable bridge end to align. The bridge track is rotated following the second button input. The keypad entry indicator will flash during the turntable bridge move operation.

The keypad LED is illuminated to show that a track number has been input and the program is waiting for the turntable bridge end input. If the second input is not entered within 5 seconds, the track number input in ignored.

Following power on, the program initiates a search for the home position sensor. This involves stepping the turntable bridge in a counter-clockwise direction until the home position sensor returns an active

state. This search operation can be aborted by pressing any key during the search operation. The next bridge movement, in program or operate mode, will redo the home position search.

The following summarizes the keypad buttons that are used in operate mode.

Button	Function
x#	Position bridge head-end to track number x. x is a button 1 through 9.
x*	Position bridge tail-end to track number x. x is a button 1 through 9.
00	Re-index turntable bridge to home position sensor.

The following warning tones can occur in operate mode.

1 tone - Unprogrammed track number specified.
2 tones - Unknown turntable circle size. Redo program mode key '0'.
3 tones - Invalid keypad input.

Digitrax DT400 Throttle

The Power On and Power Off sections have discussed the DB150 booster switches and indicators. This section will cover some of the more common aspects of the DT400 throttle. See the *Advanced Digitrax Command Control Starter Set Manual* for a complete description of its features and functions.

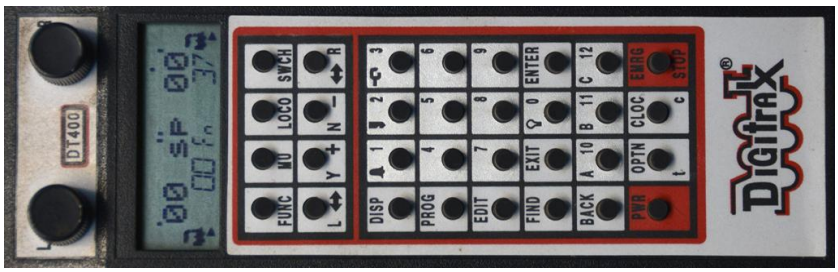


Figure 15: Digitrax DT400 Dual Throttle

The throttle must be plugged into the Digitrax LocoNet to control locomotives. It can be unplugged and reattached to the remote panel at any time, even when trains are in motion on the layout. Up to two trains can be controlled simultaneously and independently using the Left and Right throttle knobs.

Select Single Locomotive

Finally, we get to the part where we operate a train on the layout. Refer to the D&B Locomotive Roster below and select a locomotive that is not MU'd or is the *top* unit of a MU'd group. Place the locomotive on the track and then do the following on the throttle.

1. Rotate or press one of the knobs to select the left-hand or right-hand throttle. The smoke of the locomotive icon will flash to show that it is selected.
2. Press the **LOCO** button.
3. Rotate the right-hand knob until the locomotive address number is shown on the bottom line. For locomotive numbers greater than 99, use the left-hand knob to help set the upper digits.
4. Press the **ENTER** or **LOCO** button to assign the locomotive to the throttle knob.

5. Rotate the throttle knob to control the speed of the locomotive. Use the throttle's ↔ button to change the direction of movement.

There are throttle buttons that activate decoder functions wired to lamps, and for sound decoder equipped locomotives, whistle and bell. These buttons have symbols to aide in identification.

Select Multiple Locomotives

Locomotives can be grouped into a Multiple Unit and then operated as if a single locomotive. This is called MUing. Many of the D&B locomotives are currently MU'd and these configurations are retained by the DT400 throttle. The *top* locomotive of a MU'd group is selected in the throttle. When operated, the other locomotives in the MU group, if present on the track, will move in a similar manner. In fact, the *top* locomotive could be physically removed from the track. The remaining locomotives in the MU group will continue to be controlled by the *top* unit's address number. To return a MU'd locomotive to independent operation, it must be removed from the MU.

Existing MU Group

To operate an existing MU'd group, refer to the D&B Locomotive Roster and place the locomotives on the track, uncoupled, a short distance apart. Arrows on the bottom of each locomotive indicate the normal travel direction. All arrows should point in the same direction. Perform steps 1 through 4 above selecting the locomotive number marked as *top* in the group. The *top* designation will be shown in the throttle display. Then, cautiously rotate the throttle knob a few clicks. Verify that all locomotives move in the same direction. If not, either flip the offending locomotive's orientation or recreate the MU group. Once the movements are the same direction, couple the locomotives and begin normal operation.

Create Locomotive MU Group

Use the following procedure to create a MU locomotive group. Each locomotive to be added to the MU should respond similarly with respect to speed/throttle setting as the *top* unit. When MU'd, if the sound of wheel grinding is evident, the offending locomotive should be speed matched as described in a separate section.

1. Place the locomotive to be the *top* unit on the track.
2. Assign the locomotive to the right-hand throttle.
3. Use the throttle to move the locomotive a short distance in its forward direction.
4. Place the locomotive to be added to the MU on the track behind the first unit.
5. Assign this locomotive to the left-hand throttle.
6. Use the left-hand throttle to move the locomotive toward the first and couple them.
7. Press the **MU** button on the throttle. Then press the **Y +** button to add the left-hand throttle's locomotive to the *top* unit in the right-hand throttle.
8. Use the right-hand throttle to verify that both locomotives move in the forward direction.
9. Repeat steps 4 through 8 if additional locomotives are to be added to this MU.

Once the MU is created, the *top* unit can be assigned to either the left-hand or right-hand throttle for normal operations. Using the above procedure, a MU group on the left-hand throttle can be MU'd to a locomotive or MU on the right-hand throttle. In this way, helper locomotives can be coupled to the rear

end of the train and then MU'd to the train's lead or *top* locomotive. The right-hand throttle will then control the train's locomotives. Once helper service is no longer needed, they can be uncoupled and removed from the MU.

Remove Locomotive from MU Group

Use the following procedure to remove a locomotive from a MU group.

1. Assign the *top* locomotive of the MU'd group to the right-hand throttle.
2. Assign the locomotive (or MU) to be removed to the left-hand throttle. It will show as ***cn*** in the throttle display.
3. Press the **MU** button. Then press the **N -** button to remove the left-hand throttle's locomotive from the *top* unit in the right-hand throttle.
4. Use the left-hand throttle to move the removed locomotive(s) away from the MU group.
5. Repeat steps 2 through 4 as needed. Once all locomotives are removed from the MU, the *top* unit will lose this designation and become just a single locomotive.

Accessing MU'd Locomotives

When locomotives are MU'd using the consisting method described above, called Universal Consisting, speed and direction of all locomotives is controlled by the ***top*** locomotive address. Other changes, such as headlight on/off, will only affect the ***top*** locomotive. A headlight equipped consisted locomotive may not show correctly and need to be changed directly. In this case, select the consisted locomotive in the other throttle using its DCC address. When selected, it will show as ***cn*** in the throttle. The locomotive specific functions can then be changed.

Decoder Configuration Variables

Configuration variables (CV) are used to tune and customize the behavior of the DCC decoder installed in the locomotive. Customizations include direction dependent lighting effects, locomotive speed, and behavior when operated in a MU group. Sound capable decoders include CVs for selecting the horn and bell type, diesel prime mover or steam driver chuff sound, and background sounds like air release valves and coupler noise. CV1 through CV30-ish are standardized and all decoder brands support them. Using CVs, a locomotive can be set up to operate in a prototypically unique and predictable manner.

All decoders are factory programmed with the common CVs set to specific values. For example, CV1, the decoder's address, is set to 3. Following decoder installation in the locomotive, address 3 can be used to test the installation with no CV programming required. Decoders support a factory reset function which will return the decoder's CVs to these initial values. If a decoder starts behaving strangely, particularly after one or more of its CVs have been changed, it can always be reset to its initial state.

The following summarizes the D&B locomotives and some of the more important configuration variables. See the decoder user manual for lots more information about the available CVs.

D&B Locomotive Roster

Speed configured: 7-31-2019

Locomotive	MU	Decoder	Addr	CV02	CV03	CV04	CV05	CV06	CV21	CV22	CV29	Comment
Genesis F3A - Union Pacific 903 [1]	top	TSU-GN1000	89	0	0	0	255	128			2	→ cab forward
Genesis F3B - Union Pacific 903B	cn	DH123D	90	48	0	0	252	180			6	→ numbers at back
Genesis F3B - Union Pacific 905B	cn	DH123D	91	30	0	0	230	165			6	→ numbers at back
Athearn SW1500 - Southern Pacific 2551		DH123D	51	44	0	0	205	150	255	255	6	→ cab behind
Genesis F7A - Santa Fe 37A	top	TSU-885013	37	0	0	0	255	95			2	→ cab forward. CV37=2 [2]
Genesis F7B - Santa Fe 37B	cn	DH123D	38	43	0	0	255	180			6	→ feather points backward
Genesis F7B - Santa Fe 37C	cn	DH123D	39	31	0	0	255	170			7	← feather points forward
Genesis F7A - Santa Fe 37L	cn	DH163D	40	8	0	0	240	126	255	255	7	← cab backward. CV113=71 [2]
Spectrum 4-6-0 - Union Pacific 1584		DZ143	84	20	0	0	255	146			6	
P2000 GP7 - Santa Fe 2792	top	DH123D	92	48	0	0	255	186	255	255	6	→ cab forward
P2000 GP7 - Santa Fe 2678	cn	DH123D	78	56	0	0	255	184	255	255	7	← cab backward
BLI E7A - Southern Pacific 6003 [5]		QSI Paragon2	03	14	0	0	236	n/a [3]				Sound volume adjust in exhaust stack
P2000 GP30 - Santa Fe 3223	top	DH123D	23	48	0	0	255	160			6	→ cab forward
P2000 GP30 - Santa Fe 3200	cn	DH123D	32	58	0	0	255	170			6	→ cab forward
P2000 GP30 - Santa Fe 3233	cn	DH123D	33	5	0	0	245	100			7	← cab backward
Genesis 4-6-6-4 - Union Pacific 3943		TSU2200	43	0	0	0	32	n/a [4]			2	Snd volume, center middle hatch.
Mantua 0-6-0 - A.T. & S.F. 1999		DH163D	99	0	0	0	255	128			6	
P2000 SD60 - Union Pacific 6044	X	DH163D	44	25	0	0	255	150			6	Can't MU to ref, gearing.
Genesis 4-6-2 - undec, no decoder												Display only, drive gear bump.

1. Speed reference locomotive.
2. With forward lamp on, use function 3 for gyra light on/off.
3. CV6 unsupported. Use CV25=5, CV49=4, CV56=0.
4. Factory installed decoder using default CV values.
5. This locomotive, and its B unit, have trouble negotiating some turnouts. Avoid yard entry track 2 into track 4 or 5, yard crossovers track 4 and 5, and midway section block B6 into block B5.

The cells with numbers indicate the programmed value for the CV. The D&B uses the older DB150 DCC booster that does not have decoder CV read back capability. If there is any doubt as to a decoder CV value, it should be re-entered.

Additional CVs are used for configuring lighting effects and for the locomotives equipped with sound decoders, sound effects. The values shown in the following tables establish these settings. Default sound volume levels are too loud to my ear and need reduction to more scale-realistic levels. The locomotive prime mover sound is good but tends to get a bit annoying after a while. Lamp related settings configure behaviors based on locomotive movement direction and lamp type, e.g. Flashing gyra light effect.

TSU-PNP 885013 Sound Settings			
CV	**Name**	**Range**	**Value**
CV113	Quiet Mode Timeout	0 - 255	0
CV114	Engine Notch + Auto-start	0 - 47	47
CV120	Airhorn Select	0 - 43	0
CV122	Bell Select	0 - 49	13
CV123	Prime Mover, EMD 567	0 - 8	1
CV128	Master Volume	0 - 255	60
CV129	Airhorn Volume	0 - 255	130
CV130	Bell Volumes	0 - 255	90
CV220	Breaking Distance	0 - 255	0

TSU-PNP 885013 Light Settings			
CV	**Name**	**Range**	**Value**
CV37	F3 map to FX3 output		2
CV49	Forward Light		1
CV50	Backward Light + Brightness1		20
CV51	FX3 effect		2
CV57	Forward Enable		2
CV58	Backward Enable		5
CV61	Brightness1	0-255	128

TSU-PNP 885013 Default Volume Levels			
CV	**Sound Effect**	**Volume**	**D&B**
129	Airhorn	225	
130	Bell	85	
131	Prime Mover	150	
132	Air Compressor	100	
133	Dynamic Brake	125	
134	Radiator Fans	75	40
135	Alarm Bell	60	
136	Gas Turbine	200	10
137	Coupler	128	60
138	Train Brake Apply/Release	128	
139	Independent Brake Apply	100	
140	Independent Brake Release	70	
141	GenSet Prime Mover 2	170	
142	GenSet Prime Mover 3	200	
143	Poppet Valve	128	60
144	Steam Generator	200	90
145	Cab Doors	128	0
147	Relay Clicks	128	0
148	E-Brake App.	70	
149	Glad Hand Release	150	0
150	All Aboard/Coach Doors	192	0
153	Clickety-Clack	150	0
154	Sander Valve	10	
155	Fuel Loading	50	0
156	Air Conditioner	20	0
157	Wrenches	50	0
158	Pneumatic Oilers	40	0
159	Toilet Flush	20	0
160	Cab Chatter	60	0

TSU-GN1000 Sound Settings			
CV	**Name**	**Range**	**Value**
CV115	Airhorn Select	0 - 15	10
CV128	Master Volume	0 - 255	110
CV129	Airhorn Volume	0 - 255	255
CV130	Bell Volume	0 - 255	65
CV131	Exhaust Volume	0 - 255	80
CV132	Air Compressor Volume	0 - 255	90
CV133	Dynamic Brake Volume	0 - 255	90

TSU-GN1000 Light Settings			
CV	**Name**	**Range**	**Value**
CV49	Forward Light Effect		2
CV50	Backward Light Effect		1

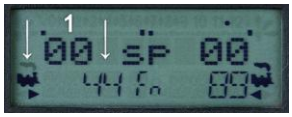
Setting Configuration Variables

The DT400 throttle supports four programming modes for compatibility with most decoders. Some of the programming modes require a dedicated programming track since any locomotive on the track will have the specified CV changed. The D&B currently does not have a dedicated programming track. However, Operations Mode or Ops Mode programming is safe to use with other locomotives on the track. Use the following programming modes.

- **Po** or Ops Mode - Normal CV programming. Only the selected locomotive is affected.
- **PG** or Paged-Digitrax - Factory reset the decoder. Ensures that all CVs are reset. Remove or fully derail all other locomotives on the layout before using.

The following example will change CV2 to 25 for the locomotive at address 44. Press the **EXIT** button at any time to abort the programming.

1. Assign locomotive 44 to a throttle. Verify the locomotive number is shown and the smoke icon is flashing (throttle is active).



2. Press the **PROG** button until **Po** (Ops Mode) is displayed. Verify correct locomotive number.

3. Use the left knob to select the CV number 2 and the right knob to select the value 25.

4. Press the **ENTER** button to write the CV value. Verify **Good** briefly shown.

5. Press the **EXIT** button to end programming or repeat steps 3 and 4 to change additional CVs.

Decoder Reset

Use the following procedure to reset an unresponsive or errant locomotive decoder to factory defaults.

1. Remove or fully derail all other decoder equipped locomotives on the layout.
2. Press the **PROG** button until **Pd** (Paged-Digitrax Mode) is displayed.
3. Use the left knob to select the CV number 8 and the right knob to select the value 8.
4. Press the **ENTER** button to write the reset value to the CV.
5. Pd mode turns off DB150 track power. Press the DT400 **PWR** button to turn it back on.
6. The decoder will now be set to address 3.

Speed Matching Locomotives

When speed matched, locomotives can smoothly operate in a MU group with minimal wear and tear on the drive, couplers, and wheels. Speed matching does not need to be exact; the railcar load will tend to even things out. But if too far out of balance, locomotive wheels will slip and grind on the rails causing above normal wear and possibly flat spots; just like the full size prototype.

Any locomotive can serve as a speed reference. At a minimum, the ***top*** unit of locomotives that are always MU'd should be used. But to be able to MU additional helper locomotives, a common reference should be used. On the D&B, the F3A Union Pacific 903 locomotive has been designated the common speed reference. Unless otherwise noted in the D&B Locomotive Roster, the other locomotives were matched to its speed performance curve. If you need more detailed instructions than presented here, see: https://tonystrains.com/news/locomotive-speed-matching-made-easy/.

Speed matching involves the entry of Configuration Variable (CV) data into each locomotive decoder. The decoder references this data when applying electrical power to the locomotive motor. Since all locomotives and motors are different, with variations in gearing and drive train friction, the derived CV speed data is locomotive specific. It is used to raise or lower the applied motor electrical power for a given throttle speed setting. There are two means for storing this speed data in the decoder, *User Speed Table* and *3 Point*. The *User Speed Table* is more accurate over the full range of throttle speed settings. But the simpler *3 Point* method has proven to be sufficient for locomotives on the D&B.

The *3 Point* method utilizes values specified by V-Start (CV2), V-Mid (CV6), and V-Max (CV5) to establish a throttle to motor speed response curve.

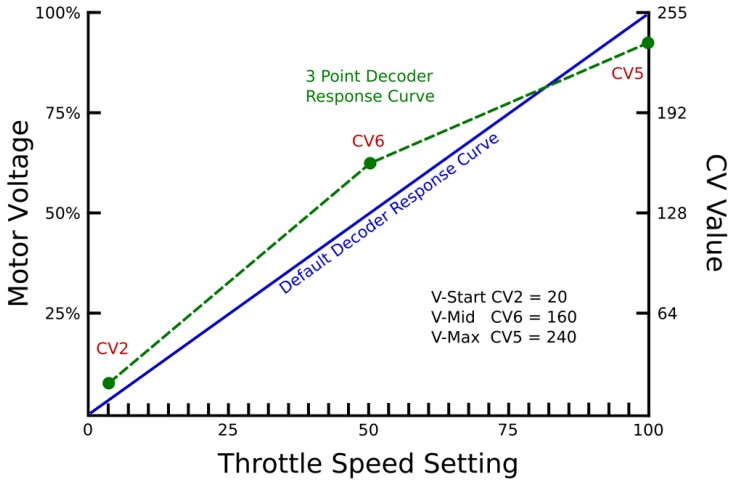


Figure 16: DCC Throttle Response Curve

This chart shows a locomotive (green dashed line) that needs a higher than default motor voltage to move at the first speed step (CV2). Likewise at a throttle setting of 50 (CV6), a higher voltage is needed to match the reference locomotive's speed. But at the full throttle setting (CV5), less than full voltage is needed. These three values are determined by trial and error. Use the following procedure, with clean Midway section block B4 track, and T05 locked in the closed position.

1. Place the locomotive to be speed matched on the track and assign it to the left-hand throttle.
2. Set initial CV values; CV2 = 0, CV5 = 255, CV6 = 128 (flat response curve).
3. Set CV3 = 0 and CV4 = 0 (no simulated momentum).
4. Slowly advance the left-hand throttle and note when the locomotive just starts to move.
5. Change the CV2 value as needed so movement begins at the first speed step.
6. Place the reference locomotive on the track about 3 to 6 inches behind.
7. Assign the reference locomotive to the right-hand throttle.
8. MU the two locomotives making sure they move in the same direction.
9. Set the right-hand throttle to speed step 2 or 3. Observe the locomotive spacing change.
10. Press or rotate the left-hand knob to select the new locomotive. Change its CV2 value by a small amount. Repeat steps 9 and 10 until spacing is maintained.
11. Move the locomotives to the start of the Midway section and separate them 18 to 24 inches.
12. Quickly advance the right-hand throttle to full. Observe the locomotive spacing change.
13. Change the new locomotive's CV5 value as needed so that spacing is maintained [1]. As the CV5 value is refined, the locomotive spacing can be reduced for easier observation.
14. Move the locomotives to the start of the Midway section and separate them 12 to 18 inches.
15. Quickly advance the right-hand throttle to 50. Observe the locomotive spacing change.
16. Change the new locomotive's CV6 value as needed so that spacing is maintained. As the CV6 value is refined, the locomotive spacing can be reduced for easier observation.
17. Repeat steps 8 through 14. Make minor value adjustments if needed.
18. Remove the newly speed matched locomotive from the MU.

Note: If a reference locomotive CV value is accidently changed (you will do this more than once), set it back to the value shown in the locomotive roster. It is not necessary to break apart the MU. Just press or rotate the right-hand knob to select the reference locomotive and correct the CV value.

[1] If the new locomotive CV5 needs a value higher than 255, then it cannot be run at full throttle in a MU. Leave CV5 at 255 and continue speed matching. Use CV2 and CV6 to set the middle range speed. It might still be possible to get the locomotive speed matched in a reduced throttle range. Note the top speed achievable and don't exceed it when the locomotive is MU'd. Otherwise, use the locomotive in standalone service.

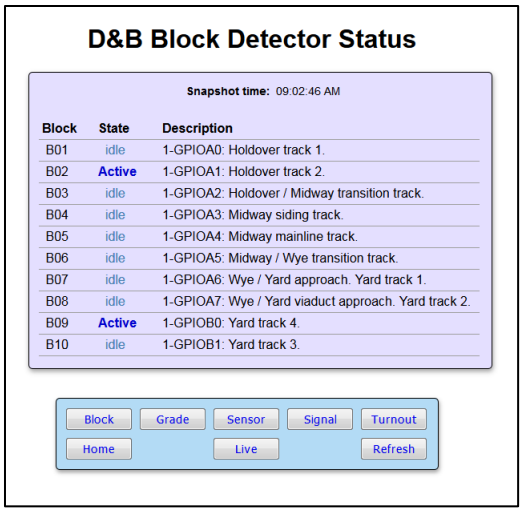
Web Browser Access

The D&B model railroad includes a web server portal when the main application is started with the **-w** option; e.g. **DnB.pl -w**. The portal provides read only access to runtime data using a web browser. These tabular data include track block and sensor activity, signal states, and turnout positions. A **Live** web page provides a simple graphical representation of the data.

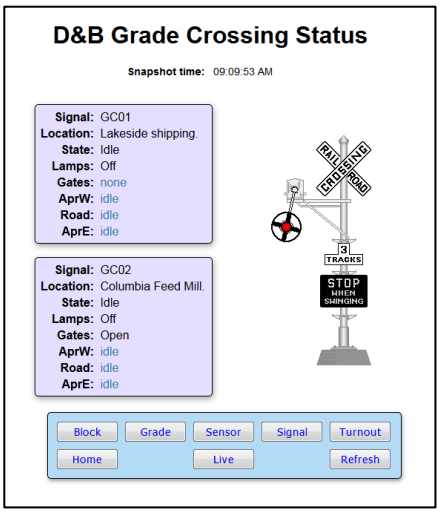
When started, a message is output on the console detailing the IP:Port value that is used to connect an external web browser. For initial connection, this IP:Port value is manually entered into the browser's address bar. The D&B Model Railroad home page is displayed upon successful browser connection. Use a browser bookmark to simplify future connections.

Connection address: **http://DnB-Model-RR:8080**

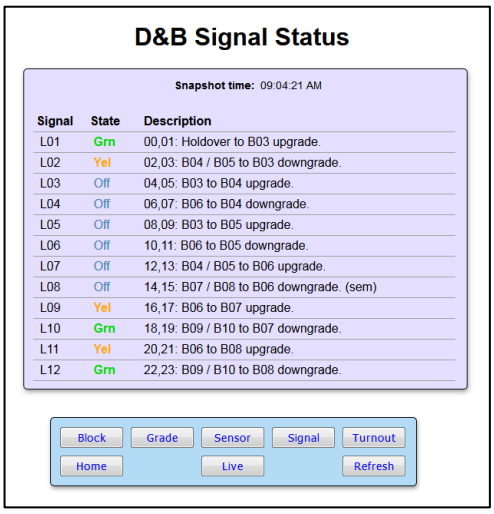
Home Page



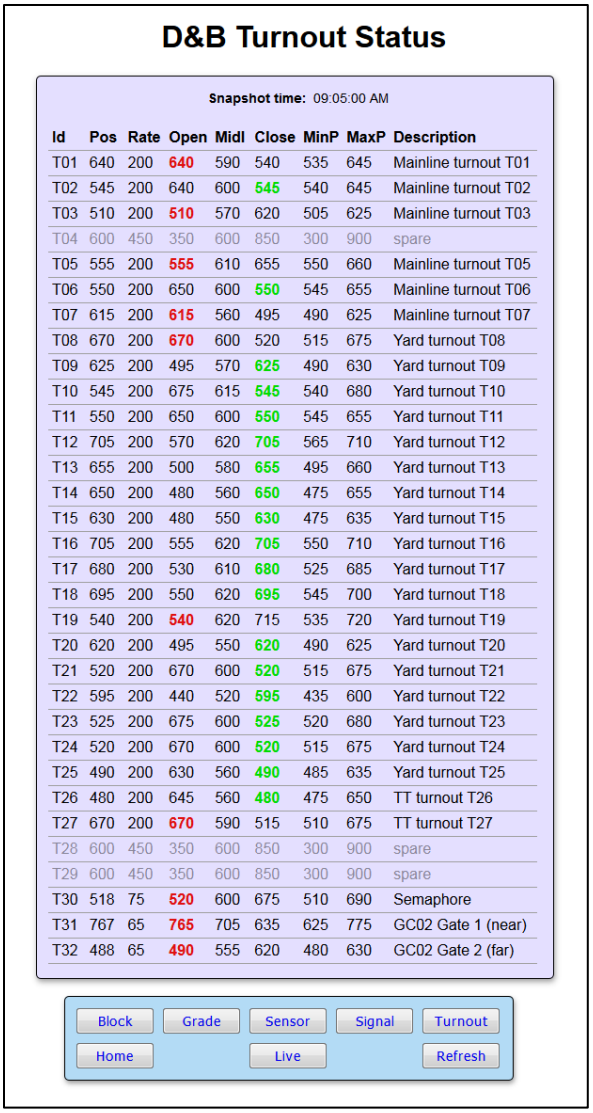
Block Detector Page



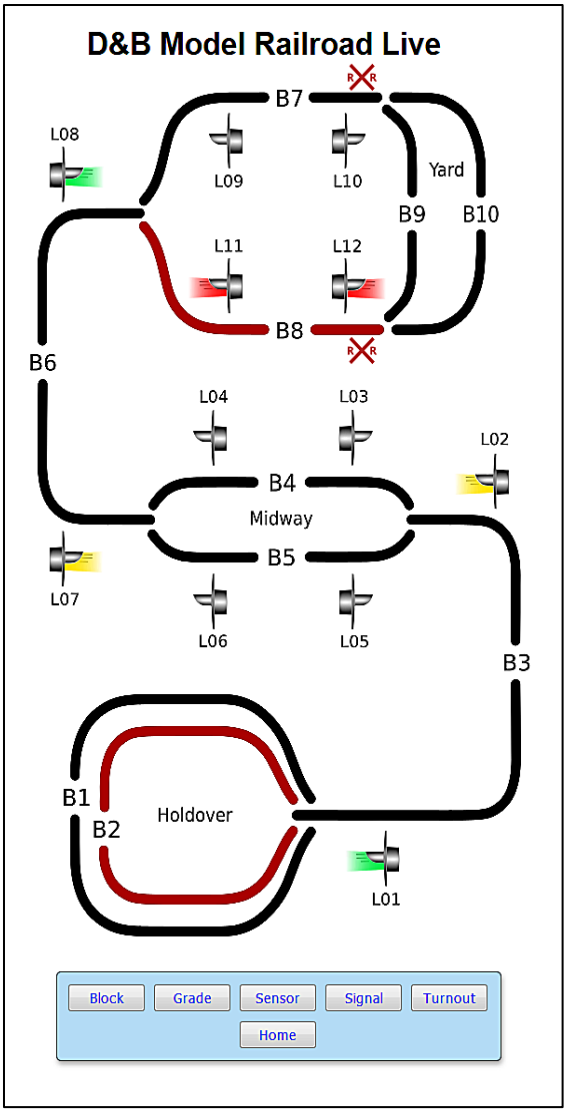
Grade Crossing Page



Semaphore Signal Page



Turnout Page



Live Page

For the pages that contain table based data, the values shown reflect their state at the indicated time stamp. Use the navigation bar **Refresh** button to update the displayed data values.

The turnout page shows the current position of each turnout; open red, green closed. All displayed data reflects the values obtained during DnB.pl startup from the TurnoutData.txt file.

The Live page shows occupied track blocks, containing a power drawing locomotive or car, in red. The semaphore signal heads are shown with their current color aspect. A flashing RXR, displayed at the yard end of block B7 and B8, is shown when the corresponding grade crossing is in an active state. The **Live** page auto-refreshes every couple of seconds

Troubleshooting

Sometimes things don't work as expected. If the problem isn't listed here, refer to the *Advanced Digitrax Command Control Starter Set Manual* for additional troubleshooting. Use the Digitrax website or Google search for additional corrective information.

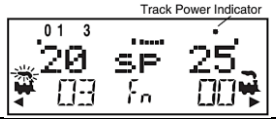
We'll assume you've verified that the layout main power strip is plugged into a live 120 VAC outlet.

DCC DB150 Booster

DB150 **Power On** green LED is dark.	1. Verify the booster transformer is plugged into power strip, power cord under electronics shelf. 2. Check the fuse, back left of electronics shelf. 3. If RPi indicators are also dark, bypass the power strip and plug booster transformer directly into a 120 VAC source. Replace power strip if booster now has power. 4. Measure across DB150 **POWER IN**, the left two terminals. Expect about 16 VAC. If not, check booster transformer and associated wiring. 5. If input voltage is present, DB150 may be defective.
Red DB150 **Offline** LED is lit. The green **Power On** LED is flashing.	1. Verify that the **SCALE** switch is set to **HO** and the **MODE** switch is set to **RUN**.
DB150 **Track Status** LED is dark.	1. Verify that the **SCALE** switch is set to **HO** and the **MODE** switch is set to **RUN**. 2. If the throttle display shows **Idle**, on throttle, press and hold the **STOP** button followed by the **Y +** button. 3. If normal throttle display, press **PWR** button and then **Y +** button.
DB150 Voltage with **Track Status** LED illuminated.	1. Measure across DB150 **RAIL A** and **RAIL B** connections. Expect 14 - 15 VAC. 2. Measure across DB150 **RAIL A** or **RAIL B** to **GROUND** connections. Expect 7 - 7.5 VDC. 3. If voltage is not good, the DB150 may be defective.
Beep tones from DB150.	***1 Tone*** - Powered on successful or programming command sent. ***3 Tones*** - A loco address has been purged due to non-use. ***5 Tones*** - DB150 short circuit shutdown. Fault Alarm. ***6 Tones*** - Powered up as a command station in a system with another command station active. ***9 Tones*** - DB150 transmit failure. There is a device blocking proper message action on the LocoNet. ***16 Tones*** - Software timeout failure. No action required, the unit will resume operation. ***Clicks*** - Op switch 41 is closed. Diagnostic clicks will sound when a valid LocoNet command is received.
Digitrax manual references.	DB150 Description: Section 5. DB150 Options: Section 26.

DCC DT400 Throttle

Operation of the DT400 throttle is dependent upon proper connection to, and operation of, the DB150 booster. The DT400's primary power is supplied by the DB150 booster. The DT400's internal battery is only used for infrared untethered operations. DT400 related troubleshooting should be performed with the throttle plugged into the DB150 booster.

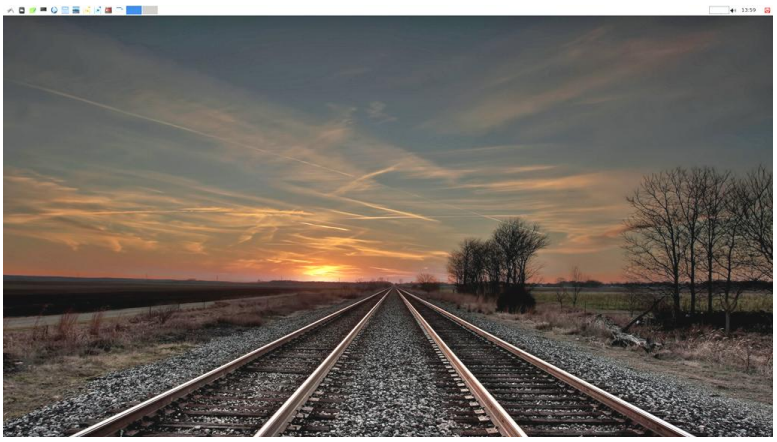
DT400 display is dark or shows **Idle**.	1. Verify DB150 Track Status LED is illuminated. If not, see above *DCC DB150 Booster* troubleshooting chart. 2. Verify throttle tethering cable is plugged into the LocoNet socket. Try using the socket connection in the remote DCC panel by the viaduct.
DT400 is unresponsive.	1. Press the **EXIT** button. 2. Unplug and reconnect DT400 LocoNet cable. 3. The DB150 might be in thermal shutdown. Power off the layout for a few minutes. If okay after power on, reduce the number of power drawing locomotives on the layout.
DT400 **Track Power On** indicator is blinking. 	System is in STOP. 1. Press the **PWR** button then the **Y+** and **EXIT** buttons. 2. Press and hold the **EMRG STOP** button then press the **Y+** button.
Digitrax manual references.	DT400 Description: Section 6. DT400 Options: Section 25.

RPi Electronics

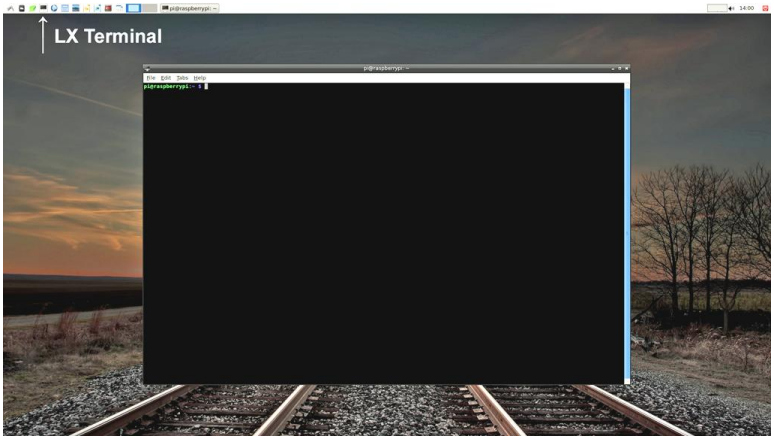
Troubleshooting of suspected RPi related problems involves power supply, hardware, and D&B software related aspects. The D&B software is coded to include information and error messages that are output to the Raspbian OS console. Viewing this message output is a good first step when troubleshooting an unexpected operational behavior.

To boot the RPi into console mode, proceed as follows.

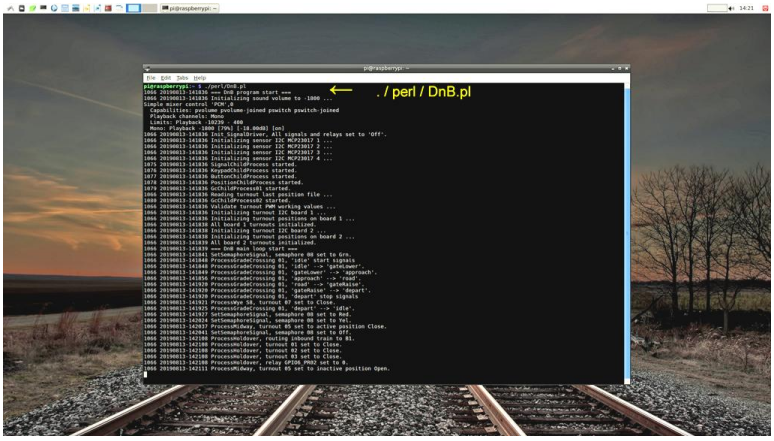
1. Perform a normal shutdown and power off the layout if currently powered on.
2. Connect a display monitor to the RPi using the HDMI cable. Power on the monitor.
3. Power on the wireless keyboard and mouse using the switches on each device. Verify the associated wireless receiver is plugged into an RPi USB port.
4. Power on the layout using the power strip switch. Press and hold down the red Shutdown button on the mainboard.
5. Continue holding down the Shutdown button while the RPi boots. Raspbian OS messages will be displayed on the monitor and replaced with the following GUI screen. If this does not occur, or the Raspbian OS locks up, the software on the RPi microSD memory card may be corrupt. Use the backup microSD memory card. See appendix.
6. Once a double beep tone is heard from the RPi, the Shutdown button can be released.
7. Upon successful boot, the monitor will display the GUI screen.



8. Launch a command line window from the menu bar. 4th icon, LX Terminal.



9. To run code in normal operation mode, enter **./perl/DnB.pl -w** in the command line window and press enter. Operational messages will be displayed in the window. If other testing will be performed, Ctrl + C can be used to terminate the program. Use with caution, orderly shutdown of the hardware components is not performed when Ctrl + C is used.



10. Enter **./perl/DnB.pl -h to** display the program help text. Other options (**-t**, **-f**, etc.) will run test code. When testing is complete, the layout should be properly shutdown. In the console, enter **./perl/DnB.pl** to start in normal operation mode. Then use the layout Shutdown button.

Power strip on/off switch trips.	1. Reference figure 6. Unplug all power supplies. 2. Set power strip switch to on. 3. Temporarily plug a table lamp into the power strip. If power strip switch trips when the lamp is turned on, replace the power strip. 4. Working left-to-right, plug in the power supplies. If power strip switch trips, unplug the power supply and investigate related circuitry.
Power LED not illuminated.	1. Reference figure 10. Verify the servo board power LEDs are illuminated. 2. If a servo board power LED is dark, swap the 5 volt servo board power plugs. 3. If the dark power LED moves to other servo board, replace the defective power supply. 4. If RPi power LED is dark, replace the RPi power supply.
Mispositioned or Inoperative servo.	1. Verify that the associated servo board power LED is illuminated. 2. Verify that the servo is not mechanically jammed. e.g. loose scenery gravel in the turnout points or tie bar. 3. Boot the RPi into console mode. 4. View the TurnoutData.txt file and verify that the Open/Close position values for the servo are correct. If not, use **DnB.pl -f** to regenerate the TurnoutData.txt file. 5. Run servo test, **DnB.pl -t x**, where **x** is the suspect servo. 6. If the servo is inoperable under test, unplug the servo from its cable under the layout and plug in a known good servo. 7. If the replacement servo functions properly under test, the original servo is defective. 8. If the replacement servo is also inoperable, verify the servo associated wiring. Use a digital meter to measure voltages at the 3 pin servo socket. Red to Black: ~4.9 VDC White to Black: ~0.9 VAC, test must be running.
Misposition of crossing gate or semaphore flag board.	1. The actuating servo is spring mounted to the layout frame. Gently twist or shift the servo to correct the positioning error. 2. If unable to correct, or positioning error reoccurs, perform the servo adjustment procedure detailed in the appendix.
Unexpected searchlight signal color or Incorrect train routing into holdover tracks.	1. Remove locomotives and resistor wheel equipped cars from the track. 2. Boot the RPi into console mode. 3. Verify that DCC track power is on, track status indicator on DB150 booster is illuminated. 4. Run sensor test, **DnB.pl -n**. 5. Place a locomotive or resistor wheel equipped car on the track one power block at a time. Verify that power block number of tones are sounded.

Inoperative or intermittent infrared sensor.	1. Verify that no bright light is shining on the infrared detector. 2. Check for infrared light path blockage by scenery or dust accumulation. Use a small brush to clean. 3. Check infrared emitter/detector physical alignment. 4. Use Radio Shack Infrared Sensor card and verify there is emitter LED infrared output. (Kept in parts drawer with track gauge.) If no output, check wiring and/or replace the emitter LED. 5. Boot the RPi into console mode. 6. Run sensor test, **DnB.pl -n**. Verify sensor operation by blocking/unblocking the infrared light path. Tones should sound when blocked/unblocked. 7. Run sensor test, **DnB.pl -b 1,2**. Refer to %SensorBit hash in the DnB.pl code. Verify the bit in question is not always shown as set (1) by the test output. Check under different room light conditions. Check the detector circuit wiring.
Random grade crossing signal operation.	1. During layout power on, signal operation will occur if a grade crossing approach sensor is blocked. The blockage triggers a grade crossing approach state. After timeout of the approach state, the grade crossing reverts to its inactive state. This is normal behavior. 2. This behavior is sometimes seen when no grade crossing sensors are blocked. Suspected causes include electronic noise and stray infrared light. More corrective action is needed.
Grade crossing improper operation. • Approach sensor issue. Grade crossing is not activated by approach sensor but activates when road sensor is blocked. • Road sensor issue. Grade crossing operation stops while the road sensor light path is blocked. • Road sensor issue. Continuous grade crossing operation when no sensor is blocked.	1. Verify that no bright light is shining on the infrared detectors. 2. Check for sensor infrared light path blockage by scenery or dust accumulation. Use a small brush to clean. 3. Use Radio Shack Infrared Sensor card and verify there is emitter LED infrared output. (Card kept in parts drawer with track gauge.) 4. If no infrared output, check sensor wire connections at the grade crossing interconnect board. Unplug and re-plug the connectors to clean the contact points. 5. Check infrared emitter/detector physical alignment.
One or more grade crossing lamps are not illuminating.	1. Check the lamp wire connections at the grade crossing interconnect board. Unplug and re-plug the connectors to clean the contact points. 2. If only half the lamps are operational on the signal stands, check wiring connections between interconnect board and mainboard. 3. Check the ULN2803A driver chip on the mainboard.
No grade crossing sound effect.	1. Check sound module volume control setting. 2. Check for 12 volts at the sound module power connectors. Check 12 volt power supply and wiring. 3. Check grade crossing speaker wiring. Sound module audio output is wired through an external board to the speaker. Verify all connectors are secure.

No confirmation or shutdown tones.	1. Check audio cable connections between RPi and the speaker module. 2. Check the USB cable connections between RPi and the speaker module. The USB cable is only used to charge the speaker module battery. 3. By the right-front mounting screw, partially hidden by the mounting wire, is a red LED that illuminates when the speaker module is powered. The speaker module's on/off switch, located under the base of speaker, is inaccessible and therefore always on. If the layout is not used for a few days, the speaker battery will become discharged and its LED will go dark. However, once the layout is powered on, the USB cable will power the speaker and recharge the battery.
Main DnB.pl program is not running. This is evident by one or more of the following: • RPi green activity LED does not flash. • Mainline turnouts don't auto-position. • Yard turnout keypad is unresponsive. • Semaphore signals do not illuminate. • Grade crossing signals do not function.	1. With the power strip switch on, verify that all board power indicators are illuminated. Refer to Figure 10. 2. If not illuminated, check associated power supply. 3. Use a digital multi-meter to verify the voltage of all power supplies. 4. Boot the RPi into console mode. If successful boot, start normal operation mode and watch for error messages in the console output. 5. If unable to boot into console mode, power off the layout and replace the RPi microSD memory card with the backup microSD memory card. See appendix. 6. Boot the RPi into console mode. If successful, start DnB.pl in command window and check for error messages. See appendix for procedure to create another SD memory card backup. 7. If still unable to boot, the RPi or other board may be defective. 8. Power off the layout. 9. Remove the retaining spacers holding the topmost board. Using a screwdriver blade and twisting action, alternately lift the connector/board a small amount off the stack at each end. The connector is soldered to top board so lift from lower connector edge. 10. Place a small piece of cardboard between the black connector and the pins of the lower board to prevent socket and pin contact. 11. Boot the RPi into console mode. 12. Repeat steps 8 through 11 until the RPi boot is successful or only the RPi board remains. 13. Last board removed or the RPi is the defective board.

Turntable

The turntable uses the legacy Basic Stamp based circuitry and control software. Refer to the appendix for more detailed information.

Turntable minor bridge misalignment	1. Minor bridge misalignment (less than ½ rail width) may be due to temperature variation or dust. Use program mode and micro-step keys 1 and 3 to align the bridge. Then exit program mode. 2. Press 0 key twice to re-index the home position. Then, re-enter desired bridge position. 3. If the misalignment is consistently greater than ½ rail width, reprogram the bridge position.
Turntable major bridge misalignment	1. A large misalignment and intermittent buzz while the bridge is rotating indicates the stepper motor is losing sync. This can be caused by: • Interference by dirt or foreign matter. • Improper stepper motor gear mesh. • Low stepper motor voltage (12 volts). Power off and investigate.
Turntable unresponsive to keypad input.	1. Check for flashing heartbeat LED on turntable control electronics. If not flashing, check 5 and 12 volt power connections. 2. Check keypad connections.
Turntable bridge does not rotate.	1. Check stepper motor connection. 2. Check 12 volt power connection. 3. Verify operation using test mode. a. Power off the turntable electronics. b. On turntable electronics board, set switch 1 position to **ON**. c. Power on the turntable electronics. d. Verify expected turntable bridge motion.
Turntable bridge nonstop counter-clockwise rotation	1. Check home position sensor connection.
DCC short circuit when a locomotive moves on or off the turntable bridge track.	1. Check turntable DCC power auto-reverser.

General

Can't change the position of a lead track turnout using the turntable keypad.	1. The lead track turnouts are controlled by the Yard route keypad in the RPi version of the layout.
Train departing holdover track derails due to improperly positioned turnout.	1. Refer to figures 2 and 11. Use the holdover turnout positioning buttons R1 through R4 and verify the proper turnout positions are set. 2. Boot the RPi into console mode and run the sensor tone test using the **-n** option. 3. One at a time, block IR beam of sensors S02 and S03. Verify expected tone count is heard. If not, check IR emitter/detector alignment. 4. Verify turnout operation using the **-t** option.

DCC short circuit when a train enters or leaves a holdover track.	1. Refer to figure 2. Verify the train traversed the holdover track in the proper direction. 2. When the lead locomotive activates holdover IR sensor S02 or S03 during exit, the track power polarity is changed. If the train is long, with power drawing units at its end, a short circuit will occur when the inbound rail gaps are bridged. Shorten the train length. 3. Boot the RPi into console mode and run the sensor tone test using the **-n** option. 4. One at a time, block the IR beam of sensors S01, S02 and S03. Verify expected tone count is heard. If not, check IR emitter detector alignment. 5. Verify the holdover section power polarity relays P1 and P2 are functioning properly. Boot the RPi into console mode and run the relay test using the **-r 1** or **-r 2** option. 6. If relay is not operating, check related circuitry, RPi GPIO and 2803A driver chip.
Train routed into occupied holdover track. Train routed backwards into holdover track.	1. Refer to figure 2. The software uses S01 to detect a train approaching the holdover section. If not detected, the holdover processing will not be performed and the last holdover turnout positions will be used. 2. Boot the RPi into console mode and run the sensor tone test using the **-n** option. 3. Block the IR beam of sensor S01. Verify expected tone count is heard. If not, check S01 IR emitter detector alignment.
Intermittent or repeating single low tone sounds that not associated with button input. Might be accompanied by improper layout operation.	1. Error message(s) are likely being reported to the RPi console. Since the console is normally not active, the tone provides a basic error notification. 2. Boot the RPi into console mode. 3. Use command **./perl/DnB.pl -w** to run the program. 4. Monitor the console output for error messages during normal layout operation. 5. Refer to the program code for error message context as an aid for corrective action.

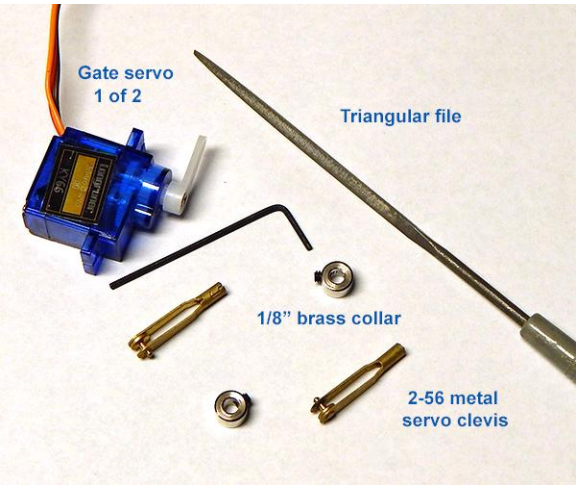
DCC short circuit when a train enters or leaves a track in the Wye section.	1. Refer to figure 4. For trains approaching wye on track blocks B7 and B8, verify the infrared sensors are causing a power polarity change. Alternately block the S08 and S09 beam path. A relay click should be heard and turnout T07 should set to the proper point position. If not, check the physical alignment of the infrared emitter detector. 2. For trains approaching wye on track block B6, verify proper operation of sensor S07. Since this beam path is within the tunnel, proceed as follows. Refer to figure 12. a. Place 3 coupled cars on track block B6 outside of tunnel. b. Use the keypad **7o** button to open T07. c. Carefully push the cars into tunnel. A relay click will be heard if a power polarity change from current setting is needed. d. Withdraw the cars from the tunnel. Wait 5 seconds. e. Use the keypad **7c** button to close T07. f. Carefully push the cars into tunnel. A relay click should be heard this time because a power polarity change will be required. g. Redo steps d, b, and c if no relay click was heard the first time. A relay click should be heard this time because a power polarity change will be required. h. Check S07 sensor wiring if a relay click is not heard in step f or g. 3. Verify the Wye section power polarity relay P3 is functioning properly. Boot the RPi into console mode and run the relay test using the **-r 3** option. 4. If relay is not operating, check related circuitry, RPi GPIO and 2803A driver chip.
Can't select a locomotive in DT400 throttle.	1. Ensure the locomotive is not already part of a MU; ***cn*** will show in the throttle display if currently MU'd. Remove from MU to make it individually selectable. 2. Reset the decoder.

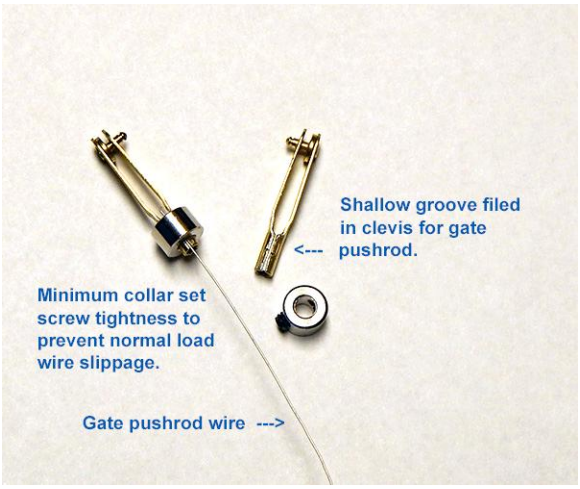
Appendix

The appendix includes some useful information from the D&B website.

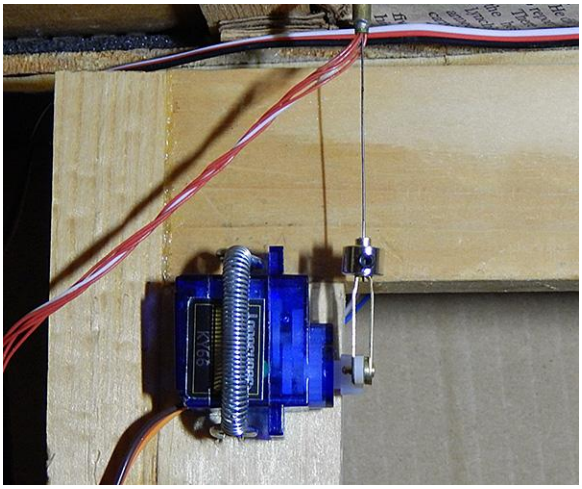
Grade Crossing Gates

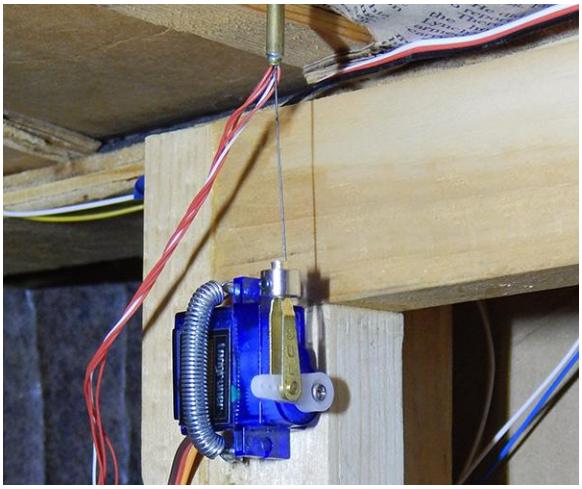
A servo clevis is attached to the existing gate pushrod wires using a 1/8 inch collar. A shallow groove is filed into the clevis to allow the collar and wire to slip on.





The servos have enough torque to potentially damage the gate mechanism if erroneously positioned. So they are mounted to the layout frame with a small spring. This allows them to twist if a gate mechanical limit is encountered. The servos are easily repositioned by reaching under the layout.





Gate Servo Adjustment Procedure

If the servo has been disconnected from the gate pushrod, begin at step 1. Otherwise, begin at step 9.

1. Verify clevis attachment to inner hole of servo arm. Clevis should move freely but have a snug fit. Lubricate if needed using dry graphite (door lock lubricant).
2. Verify that the servo is square and snugly mounted to the layout frame. A small piece of 100 grit sandpaper glued to the servo body will provide additional twist rigidity.

3. Join the gate pushrod, collar, and servo clevis. Leave collar setscrew loose so pushrod wire can easily slide.
4. Disconnect the 3 pin servo cable from the external board.
5. Using RPi console mode, edit the TurnoutDataFile.txt file. Set the Open, Middle, and Close values for the gate servo to 595, 600, and 605 respectively.
6. In the RPi console, enter **DnB.pl -m x** (x is servo number) to set gate servo to the middle position.
7. Connect the 3 pin servo cable to the external board. The servo will move to the middle position.
8. Manually move the signal gate to mid position. Then tighten the collar setscrew to attach the gate pushrod to the servo clevis.
9. Enter **DnB.pl -c x** console command to set the gate servo to the Close position.
10. Note gate position and change value in TurnoutDataFile.txt in small increments to achieve desired gate lowered position.
11. Enter **DnB.pl -o x** console command to set gate servo to the Open position.
12. Note gate position and change value in TurnoutDataFile.txt in small increments to achieve desired gate raised position.
13. Repeat for the second gate servo.
14. In RPi console, enter **DnB.pl -g2** to run test and verify desired gate operation. Some dry graphite lubrication inside the pushrod guide tube will help smooth out the movement.
15. Edit the TurnoutDataFile.txt file. For the gate servos, set the MinPos and MaxPos values to the appropriate Open and Close value. This will help prevent erroneous servo positioning beyond a physical limit which could possibly damage the gate mechanism.

Wiring boards are mounted external to the mainboard at the grade crossing locations. They are used as wiring consolidation points for signal LEDs, infrared sensors, speaker, and gate servos. A connector is added to each signal lamp LED wire to simplify connections.

Lamp Connection Procedure

1. In RPi console, enter **DnB.pl -g1** to run grade crossing test.
2. Connect signal LED common to external board lamp header pin 1.
3. Attach LED wire to pin 2 of lamp header. Note flashing lamp.
4. Attach the remaining LED wires to the lamp pin header to achieve desired lamp pattern.
5. Repeat for second signal.

Semaphore Servo Adjustment Procedure

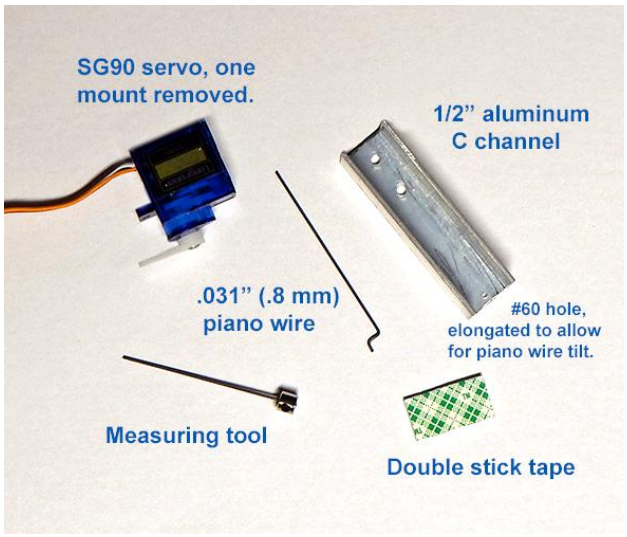
In a manner similar to the grade crossing gates, a clevis is used to attach the semaphore flag board pushrod wire to the servo. For this procedure, if the servo has been disconnected from the pushrod, begin at step 1. Otherwise, begin at step 9.

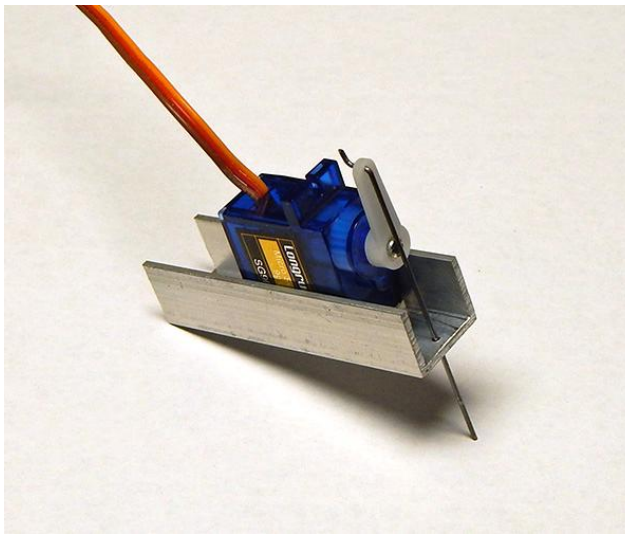
1. Verify clevis attachment to inner hole of servo arm. Clevis should move freely but have a snug fit. Lubricate if needed using dry graphite (door lock lubricant).
2. Join the pushrod, collar, and servo clevis. Leave collar setscrew loose so pushrod wire can easily slide.
3. Verify that the servo is square and snugly mounted to the layout frame. A small piece of 100 grit sandpaper glued to the servo body will provide additional twist rigidity.

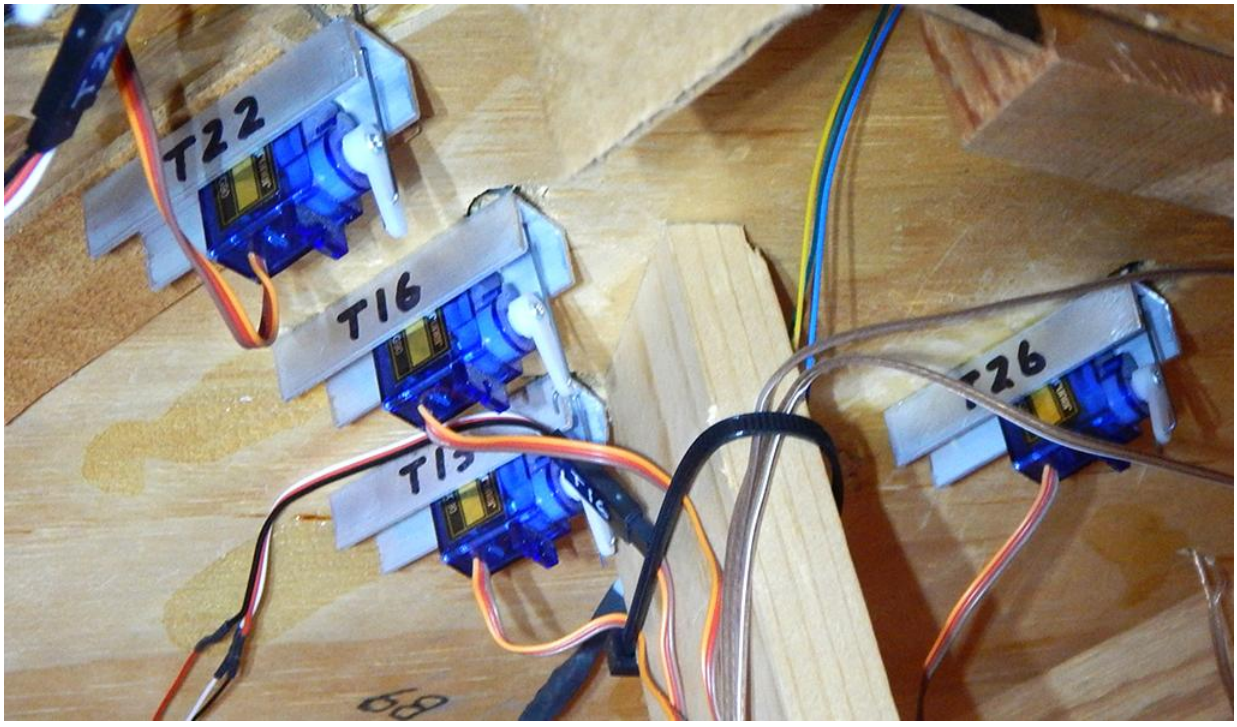
4. Disconnect the 3 pin servo cable from the external board.
5. Using RPi console mode, edit the TurnoutDataFile.txt file. For the semaphore servo, set the Open, Middle, and Close values to 595, 600, and 605 respectively.
6. In RPi console, enter **DnB.pl -m x** (x is servo number) to set the semaphore servo to its middle position.
7. Connect the 3 pin servo cable to the external board. The servo will move to the middle position.
8. Manually move semaphore flag board to yellow (mid) position. Then tighten the collar setscrew to attach the pushrod to the servo clevis.
9. Enter **DnB.pl -c x** console command to set semaphore servo to the Close (Red) position.
10. Note flag board position and change value in TurnoutDataFile.txt in small increments to achieve desired red position.
11. Enter **DnB.pl -o x** console command to set semaphore servo to the Open (Green) position.
12. Note flag board position and change value in TurnoutDataFile.txt in small increments to achieve desired green position.
13. In RPi console, enter **DnB.pl -s x** (x is signal number) to run test and verify semaphore operation. The semaphore can be positioned to each color using '**DnB.pl -s Grn:x**', '**DnB.pl -s Yel:x**', '**DnB.pl -s Red:x**', or '**DnB.pl -s Off:x**'
14. Edit the TurnoutDataFile.txt file. For the semaphore servo, set the MinPos and MaxPos values to the appropriate Open and Close value. This will help prevent erroneous servo positioning beyond a physical limit which could possibly damage the semaphore.


Turnout Servos

A piece of 1/2 inch aluminum C channel, about 2 1/4 inches long, is used to attach SG90 servos to the underside of the track board. Holes slightly larger than the mounting screws allow for some lateral positioning of the C channel. A hole is drilled near the opposite end of the C channel to permit passage of the piano wire actuating rod, a #60 drill for .031 inch (.8 mm) piano wire. The #60 hole must be elongated slightly to permit wire tilt when moved left/right by the servo. The servo mounting tab is removed flush with the servo case allowing a piece of double stick tape to hold the servo in place.







Turnout Servo Adjustment Procedure

If the servo has been dismounted from the layout, begin at step 1. Otherwise, begin at step 8.

1. Ensure the servo/C channel is mounted snug (not tight) to track board. This allows for some twist positioning of the turnout middle position.
2. Temporarily remove the servo horn from the servo drive shaft.
3. Using RPi console mode, edit the TurnoutDataFile.txt file. For the turnout servo, set the Open, Middle, and Close values to 550, 600, and 650 respectively.
4. In RPi console, enter **DnB.pl -m x** (x is servo number) to set the turnout servo to its middle position.
5. Connect the 3 pin servo cable to the mainboard. The servo will move to the middle position.
6. Press the servo horn onto the servo drive shaft as shown in the image. That is, pointing away from the track board. If the spline prevents optimal horn placement, change Middle value by 10 (step 3) and re-enter command (step 4) as necessary. Servo horn securing screw will be re-attached later.
7. Twist C channel if necessary to approximately center the points between the outer rails. Then tighten the C channel mounting screws.
8. If the Middle value was changed in step 6, edit the TurnoutDataFile.txt and change the Open/Close values to +/- 50 steps from the new Middle position value.
9. Enter **DnB.pl -c x** console command to set the turnout servo to the Close position.
10. Note turnout point position and change the turnout's Close value (step 3) by a small amount until the proper turnout point lightly[1] touches the fixed rail. Perform step 9 after each value change. It may be necessary to swap the turnout Open/Close values in TurnoutDataFile.txt to achieve proper direction of motion.
11. Enter **DnB.pl -o x** console command to set the turnout servo to the Open position.
12. Note turnout point position and change the turnout Open value in TurnoutDataFile.txt by a small amount until the proper turnout point lightly[1] touches the fixed rail. Perform step 11 after each value change.
13. In RPi console, enter **DnB.pl -t x** (x is turnout number) to verify turnout operation.
14. Disconnect the 3 pin servo cable. Carefully install the servo horn screw. Then reconnect the 3 pin servo cable.
15. Edit the TurnoutDataFile.txt file. For the turnout servo, set the MinPos and MaxPos values to the appropriate Open and Close values. This will help prevent erroneous servo positioning beyond a physical limit which could potentially damage the turnout.
16. Once the turnout is operating properly, the turnout values in TurnoutDataFile.txt should be entered into the %TurnoutData hash of the DnB.pl file. This ensures that the proper turnout position values will be regenerated if the DnB.pl **-f** option is used.

[1] The point rails should contact the stock rails firm enough to make electrical contact but not cause the servo to stall. Stall is evident by a servo buzz, vibration, or pulsing when no longer in motion. This condition results in servo high power draw and heating which may cause premature failure.

D&B Software Description

The DnB.pl program file contains a large amount of description that is not repeated here. Please refer to this documentation and the comments in the other program files for details regarding how the software operates and a clearer understanding of its code structures and design.

The DnB.pl file contains the program configuration and working data variables. A turnout configuration file is created during program shutdown using the current positions of the turnouts. This file is loaded during the next startup, if present, to persist the turnout positions from the previous layout operating session. This file can be user edited between starts to change the Open, Close, Rate, and Limit values for each turnout. This is typically needed when turnout servos are installed or replaced on the layout.

Data structures in the main program are referenced by pointer when calling subroutines. Subroutines use and update this data and return status code values. Forked child processes pass data to the main program using functionality provided by the Forks::Super module. This module overloads the default fork functions and provides a wealth of additional functionality. This module has helped to simplify the design of the RPi based program code.

The **Main Program Loop** , running at about ten cycles per second, orchestrates the overall program flow and sequence of operations. In general, it reads the sensor inputs and then calls the various turnout, signal, and track section subroutines for processing. Many of the called subroutines are coded as state machines which rely on being periodically called; though not for strict timing. Essentially, subroutines store a needed future activation time in a working variable. With each main loop call, the subroutine compares the current time with the stored future time and performs its functions based on the result.

The **Grade Crossing** code utilizes a state machine and child process. The child process is responsible for flashing the signal lamps. The state machine logic is called once per main program loop iteration. State data that is used for grade crossing control is persisted in the GradeCrossingData hash. Each grade crossing is in one of the following states; ***idle***, ***gateLower***, ***approach***, ***road***, ***gateRaise***, or ***depart***. The persisted values, current sensor bit values, and transition logic move the signal through these six states. Refer to the DnB_GradeCrossing.pm file for more details.

The **Mainline** code coordinates automated and user input functions related to the Holdover, Midway, and Wye track sections. Automated functions include train presence detection using block detectors and across-the-track infrared sensors. This input is used to position turnout points that direct a train onto a specific track. User input via keypad button can be used to override the turnout positioning behaviors.

The **Sensor** code is used to initialize the I2C GPIO extender boards and get user keypad related input during program operation.

The **Signal** code, using block detector input, sets each trackside signal to the required color. A track block is 'occupied' when a power drawing locomotive or car is present. The block protecting trackside signals have LEDs that are two lead red/green devices. Each LED is wired to two consecutive shift register bits. Red is illuminated with one current flow direction and green is illuminated with the

opposite current flow direction. The current flow direction is determined by which of the two register bits is set high/low. The semaphore color is set by its flag board position which is servo actuated. Once the flag board is moved to the required color position, the signal lamp is illuminated.

The **Turnout** code provides functionality for positioning the servos that are physically attached to the turnout points. The open/close movement is accomplished over a ¾ second interval using a forked process. After fork, the parent stores the PID of the child process. The operation is in-progress until this PID is reset. The child code completes the servo movement independent of other turnout positioning or software activities. Upon completion, the child updates the turnout position in the parent, resets the PID, and then terminates. This is accomplished using Forks::Super stdout/stderr related functionality.

The **Yard** code sets the yard turnouts based on user keypad input. Two keypad button presses are required which specify the desired ***from*** and ***to*** tracks. If the input corresponds to a valid yard route, the appropriate turnout points are set. A keypad indicator and audio tone confirms button input to the operator.

The **Message** module contains console messaging functions, general program support functions, and the Raspbian sound player interface function. The various tones that sound during operation were created using a tone generation feature in Audacity and saved as WAV and MP3 files.

The **Simulate** module contains the code and data that is used when the **-a** option is specified on the DnB.pl startup CLI. The initialization function loads the simulation commands into a working hash. The SimulationStep function is called by the main loop to get the sensor bits for the step. The SimulationStep code also initiates any simulation specified yard routes or individual turnout positionings. Refer to the **EndToEnd** routine for details.

The **Webserver** module contains the code that is used when the **-w** option is specified on the DnB.pl startup CLI. When enabled, an external web browser can be used to view various layout operational data. The browser connection point (IP:Port) is displayed on the console output. The IP value is the RPi hostname or corresponding numeric address (xxx.xxx.xxx.xxx). Port value is defined by the $ListenPort variable.

The DnB program code displays operational messages on the console or optionally to the RPi serial port. For headless operation, all console messages can be suppressed by specifying the **-q** option on the startup command line. Output of built in program debug messaging is enabled by a command line specified debug option which includes a numeric value to control message verbosity. There are some sections of code, primarily child processes that use the STDOUT and STDERR filehandles, where debug code is commented out. This code can be uncommented for program debug if needed. This code will interfere with normal program operation and should be commented out when debug is completed.

D&B Startup and Operation Summary

1. Process these command line options if specified and then terminate.
   - Program help (-h).
   - I2C address display (-i).

- Creation of new TurnoutData file (-f).

2. Check for and terminate any DnB associated orphan child processes.
3. Open serial port (-y) for redirect of console message output.
4. Display DnB program start message on the console. $MainRun variable set to 1.
5. Initialize GPIO pins and I2C devices.
   - RPi GPIO pins.
   - I2C I/O Plus board 1; MCP23017 chips 1 and 2 (sensor input).
   - I2C I/O Plus board 2; MCP23017 chips 3 and 4 (keypad scan and button input).
6. Start child processes.
   - KeypadChildProcess (16 button keypad)
   - ButtonChildProcess (two 4 button keypads)
   - PositionChildProcess (holdover red/yellow train position indicators)
   - SignalChildProcess (Signal shift register driver)
   - GcChildProcess 1
   - GcChildProcess 2 (with crossing gates)
   - Webserver (if -w is specified on CLI)
7. Initialize turnout servos.
   - Load previous TurnoutData file if available.
   - Initialize servo boards 1 and 2; PCA9685 chip.
   - Set servo channels to CLI (-o|-m|-c) or TurnoutData specified PWM value.
8. Display ambient temperature in degrees C.
9. Perform command line specified testing. Test termination results in program exit.
   - Sensor test (-b, -n)
   - Keypad test (-k)
   - Grade crossing test (-g)
   - Signal test (-s)
   - Turnout servo test (-t)
   - Power polarity relay test (-r)
   - Sound player test (-p)
   - Servo Temperature adjust test (-u)
   - Simulation mode (-a)
10. Display DnB main loop start message on the console. $MainRun variable set to 2 (simulation) or 3 (operation).
11. Run main loop until terminated by ctrl+c or shutdown button.
    A. Top of loop. Read sensors or get simulation data (32 bits).
    B. Process holdover track section sensors. Set turnouts as needed.
    C. Process midway track section sensors. Set turnouts as needed.
    D. Process wye track section sensors. Set turnouts as needed.
    E. Process both grade crossing track sections.
    F. Process signals. Set colors based on block detector inputs.
    G. Set next turnout for an in-progress yard route request.
    H. If no in-progress yard route, get new yard route input.
    I. Process holdover track section route button input.
    J. Process midway track section turnout button input.
    K. Process wye track section turnout button input.
    L. Collect and save data for webserver.
    M. Check shutdown button. Initiate shutdown sequence if set.
    N. 90 msec loop delay.

12. User initiated program termination; ctrl+c or shutdown button.
    a. Ctrl+c input: Stop child processes. Program exit.
    b. Shutdown button: Continue main loop processing and -
        • Sound 5 countdown tones, one each second.
        • Cancel if shutdown button pressed again.
        • Perform orderly shutdown of child processes, signals, relays, and indicators.
        • Save turnout position data to file.
        • Shutdown the Raspbian OS.

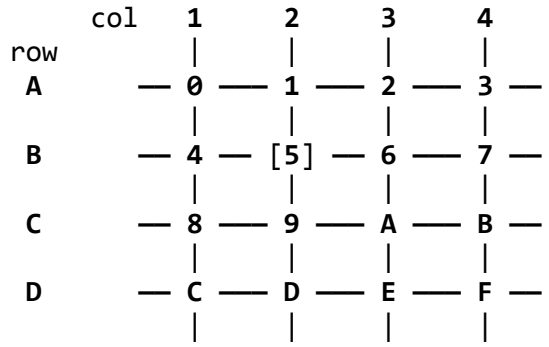
GcChildProcess

A grade crossing child process is launched during main program startup for each grade crossing. This child process is used to start and stop the signal lamp flashing and bell sound effects. Crossing gates, if present, can't be driven by this child process due to a Forks::Super restriction. Gate servo positioning is handled by the grade crossing state logic which also sends the start/stop messages.

start:apr	Start flashing lamps with bell sound 1
start:road	Start flashing lamps with bell sound 2 (not used)
stop	Stop lamp flash and bell sound
exit	Terminate GcChildProcess

KeypadChildProcess

The keypad child process is launched using Forks::Super and processes user input from the 16 button keypad. Inside the keypad, the buttons are arranged in a 4 by 4 grid. When a button is pressed, a row and column are electrically connected. For example, in the following diagram, pressed button [5] connects row B to column 2.

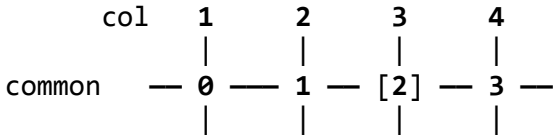


The keypad is connected to eight pins of a MCP23017 port. Four of these pins are configured as output and drive the keypad columns. The other four are configured as input with pull-up and connect to the keypad rows. By sequentially driving a single column pin low and then sequentially sampling the row pins one at a time, a pressed button can be identified. With no button pressed, all button coordinates will read high (1) due to the pull-up. A pressed button coordinate will read low (0). For multiple pressed buttons, the first button detected is used.

The keypad child process scans the keypad 10 times a second. When a button press is detected, the child process sends the button number (0-F) to its STDERR filehandle. The button press value can then be read by the parent code using Forks::Super.

ButtonChildProcess

The button child process functions in a manner similar to the keypad child process. It is launched using Forks::Super and processes user input from the two 4 button keypads. When a button is pressed, a column is electrically connected to common. For example, in the following diagram, pressed button [2] connects column 3 to common.



The keypad button columns are connected to a MCP23017 port. These pins are configured as input with pull-up. The keypad common is connected to ground. With no button pressed, all button coordinates will read high (1) due to the pull-up. A pressed button will read low (0). For multiple pressed buttons, the first button detected is used.

The button child process scans the keypad 20 times a second and differentiates between single and double button presses. A double button press is reported if the same button is pressed within a one second time period. When a button press is detected, the child process sends a string, **s**(0-3) or **d**(0-3), to its STDERR filehandle. The button press string can then be read by the parent code using Forks::Super.

PositionChildProcess

The position child process reads the IR sensors associated with the hidden holdover blocks B01 and B02. These IR sensors are physically located near the exit ends of these hidden holdover tracks. When a train interrupts an IR beam, the PositionChildProcess illuminates the corresponding panel LED. This provides a visual indication of train position within these hidden holdover tracks. Warning point (yellow) and stop point (red) LEDs are used. In this way, the engineer can stop the train prior to activating the S02 or S03 sensor which causes the holdover turnouts to be set for train departure. No other data is exchanged between this child process and the main program.

SignalChildProcess

The signal child process is launched using Forks::Super and is used to drive the 74HC595 shift register that illuminates the trackside signals. This simplifies the code and timing requirements needed to show yellow signals by toggling between red and green. This is accomplished by using two 32 bit variables. In one, the signal color bits are set to red. In the other, the signal color bits are set to green. The two variables are alternately sent to the shift register by the SignalChildProcess. For non-yellow signal positions, the bits are set to the same color in each variable.

Two time delays are used to balance the red and green 'on' times. This provides for a coarse adjustment of the yellow color for all signal positions. The variable resistors on the driver board are then used for fine yellow color adjustment of each signal.

The combined time delays further control the repetition rate of the signal child's while loop. This rate should be just high enough to eliminate flicker when the yellow color is displayed; about 25-30 cycles per second. The lowest possible cycle rate is desired to minimize CPU loading by the signal child's while loop. To further minimize CPU load, the signal child optimizes itself by checking for any yellow signal indications. When no yellow signals are being displayed, the loop repetition rate is reduced to 4 cycles per second.

Forks::Super is used to communicate new signal settings. The new data is read and used by the signal child process until subsequently updated. To minimize child processing, the message data is formatted for immediate child use by the main program code.

SignalMask 32 bit mask, all 1's. Changed signal two bits set to 0.
SignalColor1 32 bit mask, all 1's. Changed signal bit value; red (01), green (10), **yellow (01**), or off (00).
SignalColor2 32 bit mask, all 1's. Changed signal bit value; red (01), green (10), **yellow (10)**, or off (00).
Terminator "-\n". Used to remove processed messages from input queue.

Webserver Child Process

The webserver child process is launched, when **-w** is specified on the DnB.pl startup CLI, using Forks::Super. A socket access point is created at web address <RPi hostname>**:**$ListenPort. The web server code then monitors the socket for browser connections. Incoming requests are passed to **NewConnection()** where they are validated and file paths adjusted for physical location; $WebRootDir or $WebDataDir. The request is then passed to **RequestHandler()** where the response HTML is generated and sent back to the browser. This dynamically created HTML is stored and served from $WebDataDir; ramdisk /dev/shm. Static files are served from the $WebRootDir directory; e.g. /home/pi/perl/web.

Cascading Style Sheets (CSS) is used for web page decoration and element placement. Two CSS files are sent to the browser. Each is optimized for window size larger or smaller than 800 pixels. The browsers selects which CSS to use based on the current browser window size. Transparent overlay files are used on the **Live** page to color the active track blocks and signals. Java-script is used to periodically auto-refresh the track block and signal overlay files.

Test Code

A number of tests are available for use with code verification and layout operational adjustments. They are invoked when needed using the appropriate DnB.pl command line option. Tests are run individually and do not involve the D&B main program loop. This helps to focus testing to specific functions and features. Each test runs until terminated by ctrl+c console input.

Sensor test (-b option)

This test reads the user specified sensor chip(s) and displays the value(s) on the console. This continues once per second until the test is terminated. While running, each sensor bit can be manually activated to verify the associated circuitry is working as expected. The MCP23017 chip(s) to be tested are specified using a comma separated list or range. For example:

DnB.pl -b 1,3 Read and display chips 1 and 3.
DnB.pl -b 1:4 Read and display chips 1 through 4.

Sensor Tone test (-n option)

This test reads the sensor inputs defined in the SensorBit hash twice a second. When a bit change from inactive (0) to active (1) is detected, tones are sounded corresponding to the hash key of the active bit. For example, SensorBit{00} one tone, SensorBit{07} eight tones. A double tone is sounded when the bit becomes inactive (0). A console message is also output detailing the bit change. This facilitates sensor operability testing at remote layout locations; such as by manually blocking the infrared emitter to detector light path. This test runs until terminated by ctrl+c.

Keypad test (-k option)

This test reads the stderr messages generated by KeypadChildProcess and ButtonChildProcess and displays the results on the console. Reads are performed at a two second interval. For the 4x4 keypad, the 1st entry LED is toggled between on and off with each button press. For the 1x4 keypad, single and double press buttons are reported.

DnB.pl -k Read and display keypad inputs.

Grade Crossing test (-g option)

This test bypasses the grade crossing sensors and sends the **start:apr**, **start:road**, and **stop** commands directly to the grade crossing child process. The gate servos are positioned if present on the grade crossing. The sound effects circuits are also activated.

DnB.pl -g 1 Test grade crossing 1.
DnB.pl -g 1,2 Test grade crossings 1 and 2.

Signal test (-s option)

This test exercises the functions associated with the block protection signals and runs until terminated. The signals to be tested are specified using a comma separated list or range. For example:

DnB.pl -s 1,5 Cycle signals 1 and 5; red, green, yellow, and off.
DnB.pl -s 1:12 Cycle all signals; red, green, yellow, and off.

Prefacing the range with a **r** causes random signal selection and color setting.

DnB.pl -s r4:9. Random signal 4 through 9; set random color red, green, yellow, or off.

Prefacing the sequential or random range with a **g** causes the grade crossing signals to be included in the test run. Grade crossing 1 and 2 are started for color green and yellow respectively. Grade crossing 1 and 2 are stopped for color off and red respectively. Crossing gates are not positioned.

DnB.pl -s g1:12 Cycle all signals; red, green, yellow, and off. Include grade crossings.

Turnout Servo test (-t option)

This test exercises the functions associated with the turnout servos and runs until terminated. The servos to be tested are specified using a comma separated list or range. For example:

DnB.pl -t 1,3,5 Exercise servos 1, 3, and 5; sequential positions Open, Middle, Close.
DnB.pl -t 1:10 Exercise servos 1 through 10; sequential positions Open, Middle, Close.

Prefacing the range with a **r** causes random servo selection and Open/Close positioning.

DnB.pl -t r1:10. Exercise servos 1 through 10; random order, positions Open and Close.

In the above commands, about 10 servo position changes are performed per second. This verifies the functionality of multiple concurrent servo moves. This rate exceeds the maximum expected number of concurrent position changes, (the longest yard route is 8 turnouts) during normal layout operation. The console output shows the in-progress operations. This can also be viewed in a separate command console using the linux **top** command. The number of DnB.pl processes shown, minus 1 for the test code, is the concurrent in-progress operations. The CPU utilization can also be seen in the top command output.

Prefacing the sequential or random range with a **w** causes the positioning to be done one servo at a time. This verifies the child/Forks::Super PID reset functionality.

DnB.pl -t w1,3,5 Exercise servos 1, 3, and 5; wait for each operation to complete before next.
DnB.pl -t wr1:10 Exercise servos 1 through 10; wait for each operations to complete before next.

Set Servo Position (-o x, -m x, -c x)

These commands set the specified servo number **x** to its open, middle, or closed position as defined in the TurnoutDataFile.txt file. Used for servo mechanical and turnout point rail adjustments. The program exits once the position is set. When **x** is specified as 0, all servos are set to the specified position. For example:

DnB.pl -m 0 Set all servos to the middle position.
DnB.pl -c 1 Set servo 1 to its closed position.

Sound Player test (-p option)

This test is used to verify the linux sound player configurations and audition the available sound files. The available sound files in the D&B sound file directory are listed along with an associated number. When a number is entered by the user, the corresponding sound file is played. Entry of sound number zero (0) terminates the test.

DnB.pl -p

Power Polarity Relay test (-r option)

This test is used to verify the power polarity relay operation. The relays are sequentially energized for 5 seconds and then de-energized for 5 seconds. This test runs until it is terminate with ctrl+c.

DnB.pl -r

Servo temperature adjust test (-w option)

This test is used to verify the servo position temperature calculation. Input parameters specify a servo number (gate or semaphore) and one or more temperatures in degrees C. The first temperature is set and the servo is positioned. The cycle repeats for each specified temperature. Each position (Open, Middle, Close) is tested unless a single position is specified; o, m, or c. For example, to test the semaphore flag board positioning (servo #30) 'Close' position, the following command would be used.

DnB.pl -w 30c:30,25,20,15

A low tone is sounded at the start of each position. A high tone sounds for each temperature change at that position. The temperature value is changed at 2 second intervals. This test runs until terminated by Ctrl+C.

Simulation mode (-a option)

This test simulates train movements and turnout operations on the layout. The default **EndToEnd** simulation runs until terminated by Ctrl+C. Sensor bits, yard routes, and turnout positions that are stored in the %SimulationData hash are used instead of the actual layout input. In this mode, the operational code is exercised without actually running a train on the layout. Refer to the %SimulationData hash for details. Use debug level 0 to display additional simulation data on the console.

DnB.pl -a

Raspbian OS Startup and Shutdown

The following Raspbian OS configurations are used to automate the startup and shutdown of the RPi. In this way, the D&B software can be launched and terminated without need of keyboard/mouse input. It is very important to shutdown the D&B software, using the shutdown button, before powering off the layout to prevent corruption of the RPi microSD memory card.

DnB.pl Program Start

The /etc/rc.local file is used to automatically launch DnB.pl once the Raspbian OS has completed boot. Configure rc.local using the CLI as follows.

1. Edit the file: **sudo nano /etc/rc.local**
2. Add the following to the file just before the exit 0 line. Change the path to the DnB.pl file if stored in a different location.

   /home/pi/perl/DnB.pl -w -q
3. Use the editor commands **^O** and **^X** to save the file and exit the editor.

Alternatively, the line ' **>> /dev/shm/DnB.log 2>&1** ' (without quotes) could be used in place of the **-q** to send the DnB.pl console output to a log file. The log file could then be monitored using **tail -f /dev/shm/DnB.log** in a seperate command window. Use a path in **/home/pi** if the log needs to be retained when the RPi is powered down.

During Raspbian OS boot and automatic DnB.pl launch, if the shutdown button is held down, the DnB.pl program will acknowledge the button hold and then terminate. No DnB child processes will be started. The RPi will then be usable for Raspbian CLI/GUI interaction using monitor, mouse, and keyboard. Use the CLI/GUI from this point to shutdown Raspbian.

DnB.pl Program Shutdown

The shutdown button is monitored by the DnB.pl program. Detection of a button press initiates a 10 second countdown during which five tones will be sounded. At the end of the delay, the main program performs an orderly shutdown of the software processes and places the hardware interfaces into a safe condition. During the countdown, shutdown can be aborted by another press of the shutdown button.

Safe condition serves to help protect the servos, sound modules, and signal lamps should layout power remain on for an extended period. The following shutdown steps are performed.

1. Stop all child processes.
2. Raise crossing gates and semaphore flag board.
3. Wait for in-progress turnout moves to complete.
4. Turn off all servo channels.
5. Turn off all signal lamps.
6. Turn off all GPIO driven relays and indicator lamps.
7. Turn off holdover indicator lamps.
8. Save the current servo positions to TurnoutDataFile.txt.
9. Shutdown Raspbian OS using: **sudo shutdown -h now**

Once the Raspberry Pi green activity LED is no longer flashing, about 10-15 seconds, it is safe to power off the layout electronics.

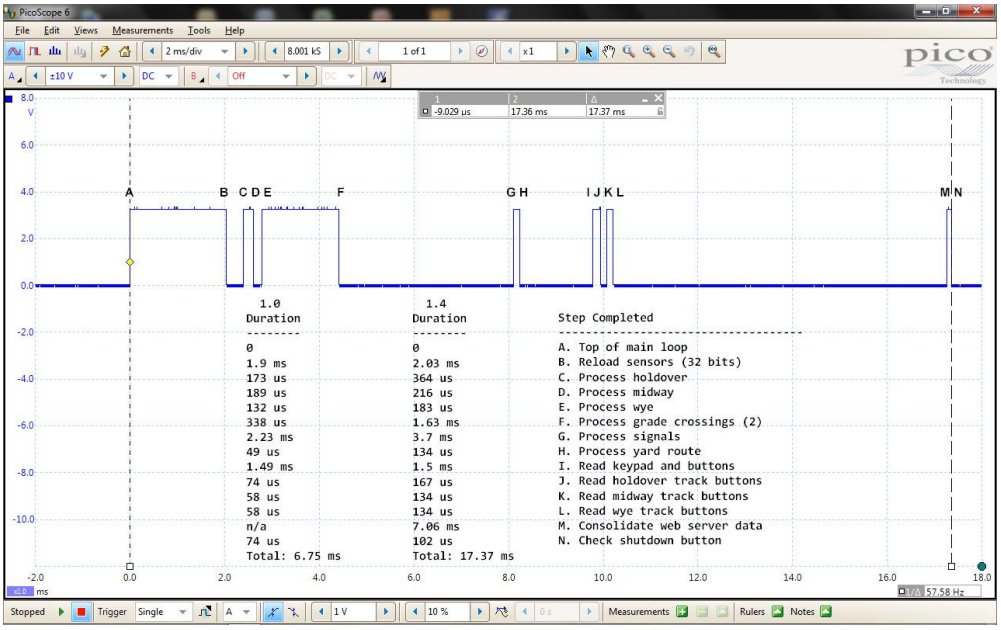
DnB.pl Main Loop Timing

The following shows the timing of the main processing loop when the -z option is used to enable toggle of the GPIO20_TEST pin. Each bit toggle delimits a section of the main loop code to aid in code analysis and tuning.

Timing for the 1.0 and 1.4 code versions are shown. The additional demands on the RPi CPU in the 1.4 code version has impacted the main loop iteration rate. The grade crossing and sensor/turnout data consolidations have added the most time; about 8.3 milliseconds. But these data consolidations are only performed every 10^{th} iteration and no noticeable impact to overall responsiveness is observed.

The main loop code includes a static time delay (not shown) to set the iteration period to about 10 cycles per second. This helps to ensure timely operation of Raspbian and the child processes. Tests using

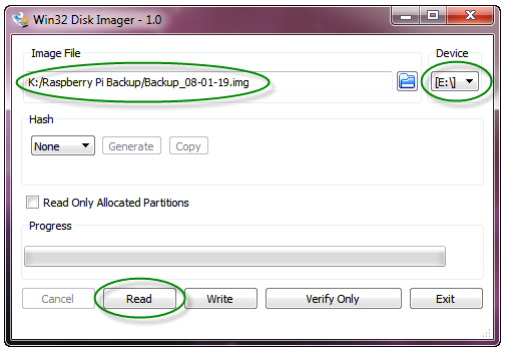
higher iteration rates consumed more CPU resources while having minimal/no benefit to overall operation. Analysis of the 1.4 code version using the Raspbian CLI **top** command indicates a 20% to 23% CPU utilization for the combined D&B related processes.



RPi microSD Memory Card Backup

Use the following procedure to create a backup image of the RPi microSD memory card. The backup image will contain all required software including the bootable and configured Raspbian OS, the D&B software, and the required perl modules. The utility program, Win32DiskImager, can be downloaded from the web. A Mac alternative is also available. The Windows version of this program is stored on the microSD memory card in the /usr/pi directory.

1. Install the Win32DiskImager program on a computer with a USB port.
2. Refer to figure 10. With layout power off, use a pair of tweezers to slide out the RPi microSD card. Note that the microSD card gold fingers are oriented on the top left side.
3. Insert the microSD card into a USB adapter and then insert the adapter into the computer USB port.
4. Launch the Win32DiskImager program.

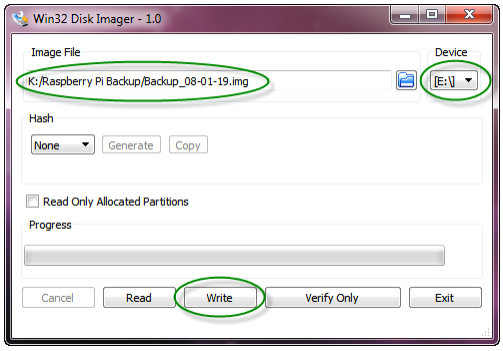


5. Specify the folder and filename to hold the backup image.
6. Specify the USB disk device holding the microSD card to be imaged.
7. Click the Read button to copy the microSD card contents to the specified image file.

RPi microSD Memory Card Restore

Use the following procedure to restore the RPi microSD memory card or create a new RPi microSD card from a previously created backup image. Upon completion, the microSD card will contain all required software including a bootable and configured Raspbian OS, the D&B software, and perl modules. The utility program, Win32DiskImager, can be downloaded from the web. A Mac alternative is also available. The Windows version of this program is stored on the microSD memory card in the /usr/pi directory.

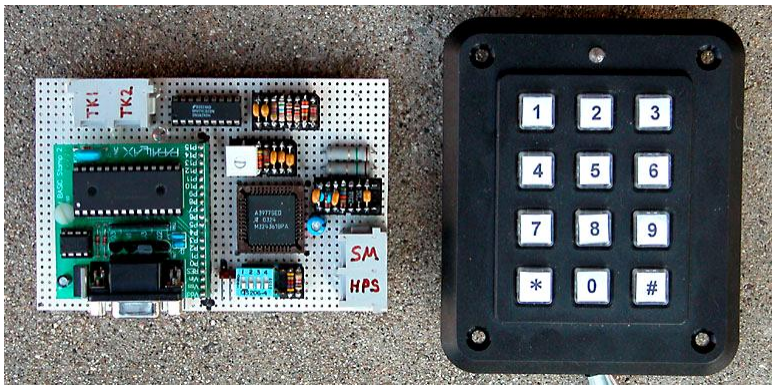
1. If necessary, install the Win32DiskImager program on a computer with a USB port.
2. Insert a 32GB microSD card into a USB adapter and then insert the adapter into the computer USB port.
3. Launch the Win32DiskImager program.



4. Specify the folder and filename holding the previously created backup image.
5. Specify the USB disk device holding the microSD card to be imaged.
6. Click the Write button to copy the image file contents to the microSD card. If warning messages are displayed, it is possible that the new microSD card has memory location defects that make it unsuitable for use with the Raspbian OS. Try a different microSD card; recommend SanDisk.
7. Refer to figure 10. Orient the microSD card with the gold fingers on the top left side. With layout power off, use a pair of tweezers to carefully slide the microSD card into the RPi card holder.

Legacy Turntable

The turntable used on the D&B is the Walthers Cornerstone Series 90' turntable, part number 933-3171. A maximum of ten track positions for both ends of the bridge track is supported; twenty total. The bridge track is rotated using a stepper motor. A track number and bridge end are entered using a 12 button telephone style keypad. This causes the turntable bridge to rotate and align to the associated track. Track positions are initially stored into the program using the keypad. This provides for completely arbitrary approach and service track positions.

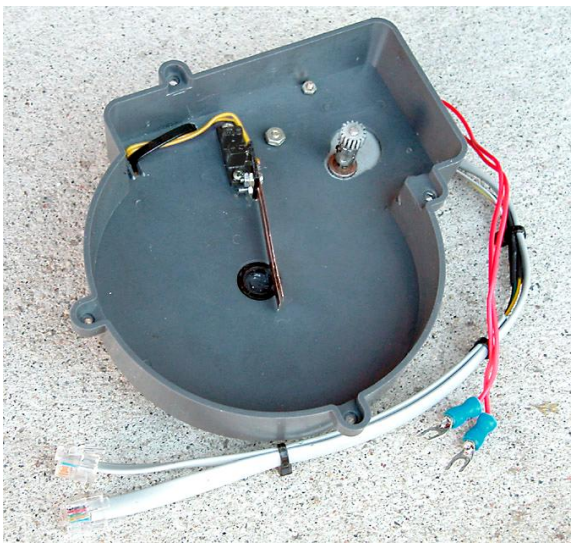


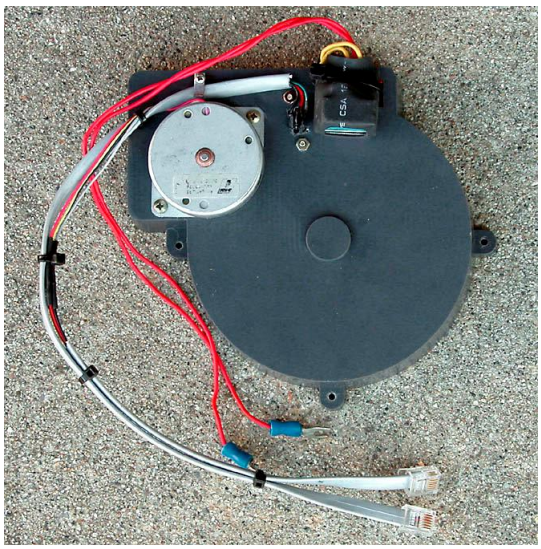
The design provides for a wide selection of stepper motor and turntable bridge gear ratios. For example, a 12 to 1 stepper motor gear to turntable bridge gear ratio results in a step circle of about 4,800 steps. This results from using a 3.6 degree stepper motor (100 steps per revolution) that is 4x micro-stepped (400 steps per revolution). With this number of total steps in the circle, a single step is about one quarter the width of a HO scale rail. Other combinations of gear ratios and micro-step settings can be used for higher accuracy up to a maximum of 65,535 steps in the circle (16 bit word). Higher ratios may introduce more gear backlash that must be taken into account. Some planning and compromise that involves desired rotation speed, position accuracy, and gear backlash is necessary.

No program changes are necessary if other gear ratios are used. The total steps in the circle are computed and stored by the program in response to the programming mode '0' key. The bridge home position sensor is used in determining the circle size. Once the circle size is determined, all turntable bridge movements are performed based upon the total steps in the circle and the home position reference.

The D&B model railroad prototype turntable uses the final gear from the Walthers turntable drive motor assembly. This yields about a 12 to 1 ratio when driving the bridge gear in the Walthers Cornerstone Series (R) 90' turntable. The drive gear was drilled and epoxied to a short length of aluminum rod. Two wheel collars were soldered together and used to join the gear to the stepper motor shaft.

The flag for the bridge home position sensor is attached to the large bridge gear. The sensor is mounted on the inside of the drive housing such that the flag interrupts the light beam. Some careful measurements and trial fitting is necessary and best accomplished before the bridge is built. The sensor must be precisely positioned and firmly bolted to the housing base with the sensor leads passing through holes in the base. Slightly elongated sensor mounting holes in the housing base will help in positioning the sensor.





The bridge power wipers must be repositioned so as not to interfere with the home position sensor flag. The plastic piece that the wipers are attached to can be trimmed to clear the bridge gear spokes and attached to the home position sensor with epoxy. The wipers will need to be shortened a bit also. There is some flexing of the wipers which is why the sensor should be bolted to the base. The wires from the wipers exit through the original wiper mounting slot.

The stepper motor is mounted to the housing with one of the mounting bolt holes slightly elongated to permit gear mesh adjustment. After the motor is mounted, the drive gear is adjusted to the proper height on the stepper motor shaft.

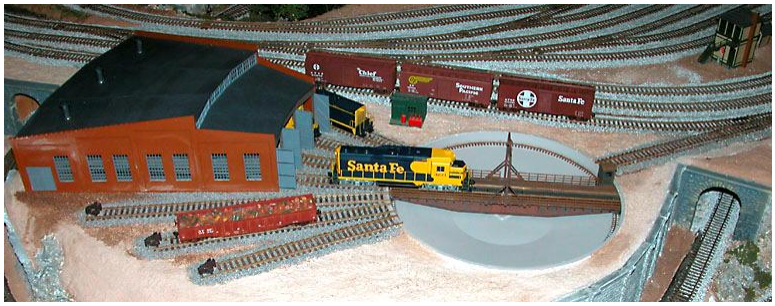
Turntable Bridge Track Power

The turntable bridge track must be powered by a polarity reversing circuit. An auto reverser module is used to control the power polarity of the turntable bridge track. All service tracks are wired to the same power polarity as the lead track. These tracks are all connected to the non-switched auto reverser output. The turntable bridge track is connected to the auto reverser switched output. The power reversing module can be mounted to the housing base with the switched wires attached to the wipers.

When ready to join the housing to the main turntable base, the wipers must be flexed to be on the proper side of the bridge commentator. Referencing the above picture, the wipers are flexed to the left side of the main bearing toward the slot in the housing base. This can be accomplished with a length of string looped around the wipers and brought out through the slot in the housing base. The string holds the wipers flexed out of the way until the housing is joined to the turntable base. Releasing tension on the string and carefully removing it allows the wipers to properly contact the commentator.

Turntable Construction

Use care when constructing the turntable to ensure that the bridge rotates freely. Trim the extra spruce from the bogie wheels completely so there are no high spots. It is important that they roll smoothly when they are attached to the ends of the bridge. During final assemble, a small amount of plastic compatible greese should be applied to the main bearings, bridge gear and drive gear to reduce friction. This helps ensure that the gear backlash is consistent throughout the full circle of bridge movement.

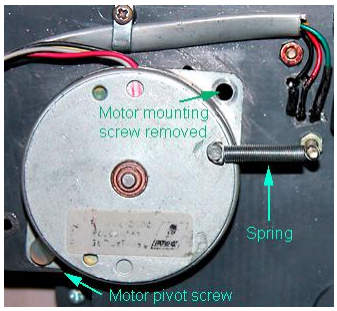


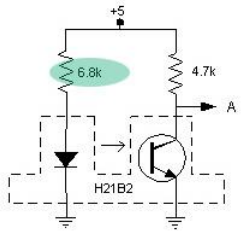
Turntable Design Updates

A number of design related issues with the turntable were identified during operational use over an extended period of time. Updates to the original design have been implemented to address the issues and are detailed as follows.

The D&B model railroad is located in a garage that is not temperature controlled. It was found that the turntable pinion and crown gear mesh varied due to winter/summer temperature changes. This resulted in a varing amount of gear backlash. Also, when the gear mesh became too tight due to expansion, the stepper motor would sometimes slip sync at certain points on the turntable rotation circle due to a slightly out of round turntable crown gear.

In the original design, the stepper motor was firmly mounted with two screws to the turntable gear housing. In the updated design, only one mounting screw is used and tightened only slightly so that the motor can pivot. A small spring is incorporated to provide tension to mesh the pinion and crown gears. Very little spring tension is required. This change eliminates almost all of the gear backlash when compared to the original design. The motor can now reposition itself due to temperature variations or an out of round crown gear. The result is a much smoother operating turntable.





Turntable Home Position Sensor

It was also found that the bridge track positioning would sometimes vary from power cycle to power cycle. Investigation and testing revealed that the home position sensor LED was being over driven. The 1k ohm current limiting resistor was changed to a 6.8k ohm resistor. The lower LED light intensity results in a consistent and repeatable home position.

Version 2 of the turntable code adds some additional documentation detail in the program listing. It also changes the micro-stepping to 8x for better position accuracy and less jitter during bridge track rotation. Some of the timing variables were adjusted to make the bridge track rotate at an appropriate rate using 8x micro-stepping.

Raspberry Pi Software

The Raspberry Pi runs the Raspbian operating system. Raspbian is a Debian Linux OS customized to run on the Raspberry Pi hardware. Like all software, updates are periodically provided to improve security and correct operational issues. Under normal use, it should not be necessary to update the Raspbian OS. However, if the need arises, launch a RPi console session (see RPi Electronics section) and enter the following commands. A wifi or wired network connection is required. Ensure the current contents of the microSD card are backed up before proceeding.

sudo apt-get update
sudo apt-get dist-upgrade

The first command updates the local software library with the available updates and their dependencies. The second command initiates the update process consisting of software module download followed by installation. Following successful update, reboot the Raspberry Pi using the GUI. If unsuccessful, examine the console output for errors and search the internet for possible corrective actions. Restore the microSD card to its previous software content if necessary.

The D&B model railroad control software is written in perl. This language has a large body of free downloadable support code on the internet. A number of advanced perl programming techniques, primarily forks and child processes, are used to implement the necessary functionality. A base version of the perl language is included in the Raspbian OS distribution. It should not be necessary to update the perl language software. The website https://opensource.com/article/17/3/perl-raspberry-pi details perl support for various Raspberry Pi operating systems.

The D&B code needs perl version 5.24.1 or later. Enter the following command at the RPi console to display the perl version.

perl -v

Enter the following command at the RPi console to update perl to the latest version.

sudo apt-get install perl

In addition to perl, the WiringPI and Forks::Super perl packages are required. The Webserver functionality requires HTTP::Daemon , HTTP::Status , and Sys::HostAddr.

sudo cpanm Forks::Super
sudo cpanm HTTP::Daemon
sudo cpanm HTTP::Status
sudo cpanm Sys::HostAddr

Wireless Printer

A Raspbian OS wireless printer, the Brother 2275DW laser printer, is configured in the microSD card. Raspbian uses CUPS to configure printers. A web browser based user interface simplifies the printer configuration process. Launch the Raspbian web browser. Enter **http://127.0.0.1:631** into the browser address bar. Use the various control in the displayed web page to select and manage the available printers. It might be necessary to download and install vendor specific drivers.

Parallax 2004 Product Catalog