# RAILROAD & CO.™

## TrainController™

## Silver and Gold

## Version 9

## Change Description

**September 2017**

# RAILROAD & CO.™

# TrainController™ Silver and Gold

# Version 9

# Change Description

**September 2017**

**Contact**: Freiwald Software
Kreuzberg 16 B
D-85658 Egmating, Germany
e-mail: contact@freiwald.com
http://www.freiwald.com

# Table of Contents

# About this Document

**RAILROAD & CO.** is the leading product line of computer programs for digitally or conventionally controlled model railroads. It contains the following members:

- **TrainController™** is the world's leading software for model railroad computer control.
- **TrainProgrammer™** is the program, which makes programming of DCC decoders as simple as a few clicks with your mouse.
- **+SmartHand™** is the world's premium handheld railroad control system designed for computer controlled model railroads.
- **+4DSound™** is a module, that recreates realistic spatial sound effects for each model railroad layout controlled by **TrainController™** without the need to install onboard sound into each decoder.
- **+Street™** is a module for control of car systems with **TrainController™**.
- **+Net™** is a module, that allows to control your layout with a network of several computers running **TrainController™**.

### The Editions of TrainController™

**TrainController™** is offered in three variants:

- **TrainController™ Bronze** provides a low-cost entry into computer controlled model railroads. It is primarily designed for users with small and medium size layouts and average requirements. Novice users, who do not know **TrainController™**, may consider doing their first steps with **TrainController™ Bronze**. The reduced functionality of this variant makes it easier to identify and to learn the basic functions of **TrainController™**.
- **TrainController™ Silver** is the successor of the established and well-known version **TrainController™ 5**. It addresses users with high demands and also users, who are not reluctant to puzzle to accomplish individual goals.
- **TrainController™ Gold** is the flagship of the **TrainController™** family and in a class of its own. **TrainController™ Gold** is primarily designed for users with supreme requirements, who want to operate their layout like the real professionals. While **TrainController™ Silver** is already able to operate even very large layouts, **TrainController™ Gold** provides much more convenience, efficiency and security for design and operation – especially for larger layouts.

## RAILROAD & CO. TrainController™ 9 Change Description

This document provides an overview of the features, which are unique in **TrainController™ 9** and which distinguish **TrainController™ 9** from **TrainController™ 8**. It is mainly intended for users, who know **TrainController™ 8**, and who want to learn about the differences between **TrainController™ 9** and **8**. It is assumed, that the reader is familiar with **TrainController™ 8** and the **TrainController™ 8 Users Guide**. New users of **TrainController™ 9** should focus on the **TrainController™ 9 Users Guide** rather than this document.

The numbers of the particular chapters and sections of this document are inherited from the concerning chapters and sections of the **TrainController™ 8 Users Guide**. This allows readers, who are familiar with that Users Guide, to implicate the contents of both documents. This is also the reason for the gaps occurring in the numbering.

All text sections, that describe features of **TrainController™ 9 Gold**, which are not provided by **TrainController™ 9 Silver**, are marked with a specific marking on the left side of the text in the same way as this section. Contents marked in this way do not apply to **TrainController™ 9 Silver**. Users of this program version or readers only interested in **TrainController™ 9 Silver** can safely ignore these contents.

All text sections, that describe characteristics of **TrainController™ Silver**, which do not apply to **TrainController™ Gold**, are marked with a specific marking on the left side of the text in the same way as this section. Users of this program version or readers only interested in **TrainController™ Gold** can safely ignore these contents.

Unless otherwise indicated all screen shots show the user interface of **TrainController™ Gold**. This means in particular, that user interface options may be visible, which are not available in **TrainController™ Silver**.

### The new Functions at a Glance

In the following all new functions are listed, that apply to **TrainController™ 9 Silver** and **Gold**. Outstanding improvements are marked with an *.

**Misc:**

1. *** New user interface with Ribbon as known from the current Microsoft Office versions (see page 16, “User Interface: Ribbon vs. Menus and Tool Bars)
2. More than 30 new user interface designs (see page 17, “User Interface Design”).

3. Specific design with the possibility to adjust the primary colors of the user interface to personal taste.
4. Multiple windows of the same type (e.g. all train windows) can be closed simultaneously with a single call of a menu command.
5. Floating windows can be easily maximized or half-screen docked to the boundaries of each computer screen by drag & drop with the mouse (page 18).
6. Improved, HTML based tool tips with text formatting and images.
7. Improved clarity of the printout of project data with icons of categories, object states and other details like the display in the Inspector window.
8. The built-in image editor provides the possibility to enter text directly into the edited bitmap image.
9. Clipboard support (copy, cut & paste) for the entries in the **Operations**, **Condition** and **Trigger** tab.
10. By clicking to an entry in the **Operations**, **Condition** and **Trigger** tab with the right mouse button it is possible to replace the object with another object of the same type. This is useful, for example, if groups of entries have been copied with the clipboard and if the new groups will differ from the original group only by individual objects.
11. The **Memory** tab of block markers provides the special entry **Reference Indicator** in the indicator list, when the marker is to be turned off by another indicator. This allows a quick and safe selection of the reference indicator associated with this marker for use in the marker's memory.

**Hardware and Digital Systems:**

12. The complete functionality provided by **+Hardware** Version 8 is available in the **TrainController™** core product.
13. The **Info** button in the **Connection** tab of many objects displays help information about the currently selected digital system.

**Switchboard:**

14. *** Switchboard data and switchboard windows are now separated from each other like trains and train windows or block diagrams and dispatcher windows. Switchboards and switchboard windows are created and destroyed independently from each other with separate menu commands. It is possible to create several switchboards and display them in one single switchboard window. It is also possible to create several switchboard windows for the same switchboard.
15. Different grid settings can be applied to different switchboard windows. It is for example possible to display one switchboard in a first window with small grid settings as an overview and another (or the same) switchboard in another window with large grid settings for details.

16. Active or occupied routes can be highlighted in the switchboard or dispatcher window in the individual color of the schedule, which is currently using the route. This makes it possible, for example, to highlight the same route in a certain color for schedules, which use the train in one direction, and in another color for schedules, which use the route in the other direction.
    Since the properties of AutoTrain can be edited like regular schedules (see feature 73) it is for example possible to highlight routes used for regular schedules, AutoTrain runs and spontaneous runs in different colors.
17. Smart gate symbols for control of the gates of engine sheds or other gates passed by trains (see page 20).
18. Crossing gate symbols with own intelligence for control of the crossing gates at single- or multi-track railway crossings (see page 20).
19. Text elements can contain more than one line of text.
20. The layer of text elements (in front of vs. behind track symbols) can be specified explicitly and does no longer implicitly depend on the transparency of the particular text element.

**Train Control:**

21. **High Precision Scaling (HPS)** improves the calculation of engine scale speed and leads to more accurate compliance with distances and ramps (see page 75).
22. **Extended Profile Generation (EPG)** causes trains to stop more exactly in situations, where they travel a certain distance at slow speed (see page 75).
23. **Adaptive Braking Procedure (ABP)** improves the compliance with calculated stop locations (see page 63).

**Train List and Train Management:**

24. Export and Import of vehicle functions is possible separately from engine or car records.
25. Several improvements of the **Function Library** dialog; among others resizing, sorting, clipboard support (copy, cut & paste), etc.

**Dispatcher and Automatic Operation:**

26. Improved User Interface for AutoTrain by Drag and Drop (see section 5.8)
27. Improved calculation of the block diagrams of schedules: paths in dependent schedules are recreated accordingly in many cases, when blocks have been inserted or deleted in the switchboard.
28. Clipboard support (copy & paste) for the **Rules** tab of the **Schedule Properties** dialog.
29. Waiting Times in blocks of schedules are specified as real-time data (see page 75).

30. Trains do not necessarily stop on a turntable bridge, when entry and exit are done via opposite tracks.

**Inspector Window:**

31. The current display status (e.g. whether a certain category is currently collapsed or expanded) is maintained as far as possible, when the focus changes to another object.
32. Detailed display of engine functions.
33. Display of the rules for spontaneous runs for each vehicle.
34. List of the schedules assigned to the current object (e.g. marker).
35. Extended display of block properties: enabled directions, critical block, etc.
36. Extended display of schedule properties: general settings, complete list of start and destination operations, rules, details of successors, etc.
37. List of all sections of a schedule including the schedule specific section settings. Since these settings are only displayed, if they are different from their default, it is easily possible to determine those sections, where non default section settings have been made.
38. The list of schedules, which contain the current block or route, includes the details of the schedule specific schedule section sections for this block or route and each schedule.
39. List of all objects, which the current schedule is assigned to.
40. List of all schedule sections, which contain the current object in their operations.
41. List of all schedule sections, which contain the current object in their conditions.
42. Display of all schedules, in whose start or end operations the current object is contained, even if these operations are lists.
43. Display of all engine functions, in whose list the current object is contained.

**Message Window and Pins:**

44. *** Pins (see chapter 8.1, “Pins”)

**Explorer Window:**

45. Display of markers.

**+SmartHand™ Mobile:**

46. Full and individual customization of the switchboard view (colors, track styles, highlighting, grid style, grid size, etc.) for each particular mobile device.
47. The driving mode currently selected in the train view is applied to **AutoTrain** runs.

### New Features of TrainController™ Silver

The following differences only apply to **TrainController™ 9 Silver**.

**Train Control:**

48. Multiple unit operation is possible by means of train sets (see section 11.1, “Train Sets in TrainController™ Silver”).
49. The train operation to execute an auxiliary function provides options to specify the vehicle, when applied to a train set. It is possible to select the first or all vehicles in a train set.

**Dispatcher and Automatic Operation:**

50. *** The automatic calculation of block diagrams is now possible for all switchboards.
51. Connector symbols in the switchboard allow to connect the block diagrams of different switchboards with each other.

### New Features of TrainController™ Gold

The following differences only apply to **TrainController™ 9 Gold**:

**Misc:**

52. * The status of the user interface can be optionally stored in a separate file, which allows to edit the project file on different computers with individual settings and window arrangement of the user interface for each of these computers (see page 18).
53. The **Multiple Changes** command provides the possibility to change the color of multiple objects in one step.
54. The **Multiple Changes** command supports the **Memory** tab of indicators, flagman objects and markers. In this way the memory of multiple indicators can be changed in one single step. Multiple markers, in particular, can be set to be turned off by their reference indicator in one single step by applying feature 11. (see above) in this context.

**Switchboard:**

55. Route operation by right mouse button clicks to track symbols: clicking with the right mouse button to a track diagram symbol in the switchboard opens a context

menu, which contains – among others – a menu of routes, which pass this track symbol. By selecting a route from this menu it is possible to toggle the state of this route.

56. Crossover symbols (see page 20).
57. Text elements can display HTML-based content for text or graphic formatting.
58. The editor for controls of extended accessories in the **Controls** tab of the **Extended Accessory properties** dialog now supports clipboard copy, cut and paste. In this way it is much easier to create a description for an extended accessory with a plurality of similar controls.

**Train Control:**

59. Engine functions can be directly operated with menu commands in the main window or in the context menu of each vehicle after pointing to that vehicle on the screen. It is not necessary to activate a train window first.
60. It is possible to assign two lists of functions to each engine function, one list for each state of the function (see page 23).
61. The train operation to execute an auxiliary function provides options to specify the vehicle, when applied to a train set. It is possible to select the first, last or all vehicles in a train set, to apply function forwarding as in previous versions or to select the vehicle by marking a position in a train description (see page 45).
62. The train operation to separate a train set allows to select the vehicle, where the train set will be separated, by marking a position in a train description (see page 45).
63. The train operation **display name** allows to set a variable train name for display on the computer screen, that depends on the currently executed schedule, for example, or other operational aspects (see page 45 ff).
64. The train operation **short distance move** allows to move trains by a short distance to a certain direction – also if the train is being controlled by a schedule (see page 45 ff).
65. The Engine Function Library (see section 3.6, “Headlights, Steam and Whistle”) provides an option, which causes a function, which is turned on or off in the train window for a vehicle in a train set, to be applied to all vehicles in this train set. In this way it is for example possible to toggle the interior lights of all vehicles in a train set with one click to a function button in the train window.
66. The measurement of the speed profile can be performed by using almost arbitrary third party speed measurement facilities (see page 22). The measurement is by default performed in a semi-automatic guided procedure. But it can also be done in a full automatic mode, if supported by the speed measurement device.

**Train List and Train Management:**

67. The tool tip of each train set displays a symbol of each vehicle in the train set.
68. The **Speed Profile** or **Advanced Fine Tuning** dialog box can be accessed directly from the main menu bar.
69. The **Function Library** can be accessed directly from the main menu bar.
70. Clipboard support (copy, cut & paste) for the **Trains** tab.

**Dispatcher and Automatic Operation:**

71. *** All lists of objects (blocks, routes, schedules, etc.) in the dispatcher window now provide an optional tree mode. In this mode the objects are not displayed in a plain list, but structured in a tree view. The folders in this tree are inherited from the folders in the explorer window. These folders can be created or deleted in the dispatcher window and will also appear in the explorer window, or vice versa.
72. Occupancy highlighting of blocks can optionally reflect the position and extent of the currently active indicators (according to the position and extent of each indicator in the block editor).
73. * AutoTrain can be edited like regular schedules via its own block diagram. It is possible to remove blocks or routes from this diagram, which excludes them from being used in AutoTrain runs. It is furthermore possible to edit the properties of AutoTrain with almost the same range of options (such as start and destination operations, driving mode, rules, permitted trains, condition, etc.) like regular schedules.
74. * Spontaneous runs have an own block diagram, which can be edited like the block diagram of other schedules. It is possible to remove blocks or routes from this diagram, which excludes them from being used in Spontaneous runs. In this way it is easily possible to limit spontaneous runs to certain areas of the model railroad layout.
75. * Many program settings can be made much more flexible with variables (see section 14.14, “Variables”).
76. System events and states can be evaluated in triggers or conditions (see page 44).
77. The menu command **Restart all Schedules** restarts all schedules, that were active, when the **Terminate All Schedules** menu command, the global **Power Off** command or the edit mode was most recently called.
78. The prerequisite control flow operation does not only allow to evaluate the state of a single object, but also to evaluate complex conditions (see page 45).
79. The label in the label and goto control flow operations may contain more than 4 characters.
80. The fact, whether an AutoTrain or spontaneous run is active can be evaluated in triggers or conditions of other objects.

81. Recording of blocks and routes with the recorder (e.g. during creation of operation list, triggers or conditions) is also possible via the block diagram and the diagram of schedules.
82. The **Multiple Changes** command supports the **Rules** tab of schedules. In this way several or all rules of multiple schedules can be changed in one single step.
83. * The extension of the rule **Start the oldest train** (see section 15.5, “Overview of all Schedule Rules”) provides improved balance between coincidence and predictability with regard to the started train.
84. New schedule rule **Do not reset resting time** (see section 15.5, “Overview of all Schedule Rules”) .
85. The new rule **Control car pulls** extends the effect of the existing rule **Train may only start in forward direction** to train sets with a control car at their head (see section 15.5, “Overview of all Schedule Rules”).
86. * New schedule rule **Maximum Detour** (see section 15.5, “Overview of all Schedule Rules”) .
87. The new schedule rule **Use start block as destination block** (see section 15.5, “Overview of all Schedule Rules”) allows dynamic specification of home blocks for circular schedules.
88. The new schedule rule **Include turntables** (see section 15.5, “Overview of all Schedule Rules”) allows turntables to be limited to AutoTrain runs from or to adjacent blocks (e.g. in the attached roundhouse).
89. Additional rules allow to affect the calculation of internal block signals (see section 15.5, “Overview of all Schedule Rules”).
90. Line-up and automatic move up of multiple trains in a single block with one single sensor.
91. * Stations (see section 15.7, “Stations”)
92. Specific calculation of internal block signals for shunting moves (see page 66, “Local Schedules and calculated Signals”)
93. * Booster and Booster Management (see section 15.8, “Booster”).
94. The schedule tab for markers and virtual contacts has been extended. It is now possible to limit the validity of markers, for example, not only to certain schedules, but also optionally to trains under control of schedules or not, to AutoTrain runs, Spontaneous runs, local or non local schedules (see page 65, “Local Schedules”) or to schedules, which control trains with specific driving modes.
95. The state **reserved by a local schedule** of a block can be evaluated in triggers and conditions. This provides the possibility to distinguish, whether a block is reserved by a local schedule (see page 65, “Local Schedules”) or not. This option is for example useful to establish specific signaling or other logic for shunting moves.
96. The internally calculated aspect of distant signals can be evaluated in trigger and conditions (see page 48).

97. Extended accessories can be used to issue complex or non-standard sequences of digital locomotive commands by the functions of an engine or car (see page 49).

**Inspector Window:**

98. Display of the properties of AutoTrain.
99. Display of the settings for turnout position control.

**+SmartHand™ Mobile:**

100. Train sets can be arranged on mobile devices with the train set view (see page 77)
101. Extended accessories with more than 2 states can be operated with the command bar in the switchboard view.

# 1 Introduction

## 1.3 Fundamentals of Use

### User Interface: Ribbon vs. Menus and Tool Bars

The user interface of **TrainController™** is either controlled via the Ribbon known from the current versions of Microsoft office or via menus and toolbars as in the previous versions of **TrainController™** (classic user interface).

**Ribbon:**
All the commands in the new user interface are organized into logical groups on a series of tabs called the Ribbon. The Ribbon provides an accessible interface to rich functionality that **TrainController™** provides.

When you start **TrainController™ 9** for the first time, you will notice that the Ribbon has replaced the old style menus and toolbars. Think of the Ribbon as a set of "results-oriented" tabs.

Each tab of the Ribbon is like a rich toolbar organized around a high-level task and contains commands for accomplishing that task. The **Track** tab, for example, contains the frequently-used commands for editing the track layout in the switchboard, while the **Operation** tab includes commands for operation of the model railroad. When a tab is selected, the commands associated with it become visible in the upper part of the screen.

The commands on each tab are ordered into groups to further organize the available features. For example, the **Edit** tab includes groups of commands called **Undo**, **Clipboard**, **Train**, **Schedule**, etc.

You will find all the traditional file menu commands under the **File** menu button left of all tabs in the top left corner of the Ribbon.

**Classic User Interface:**
The **File** menu also provides a command, which allows to change to the classical user interface with menus and tool bars known from the previous versions of **TrainController™** for users who prefer the old style.

**Quick Access Toolbar:**

Everyone has a set of commands they use most often, so it is possible to add commands to the **Quick Access Toolbar**, a small toolbar positioned in the title bar of the application window. The **Quick Access Toolbar** provides a location for heavily-used commands that need to be available with one click (regardless of the current Ribbon state). You can add any item to the **Quick Access Toolbar**. From then on you can access the command from wherever you are in the application, and avoid situations where you need to repeatedly switch between Ribbon tabs to accomplish repetitive tasks.

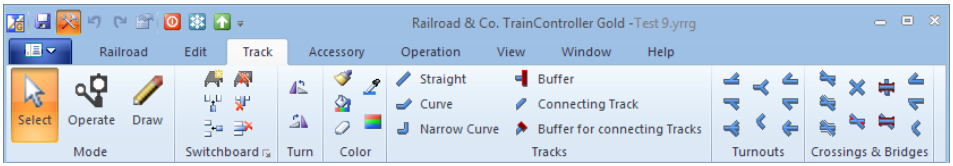



**Diagram 1: Ribbon**

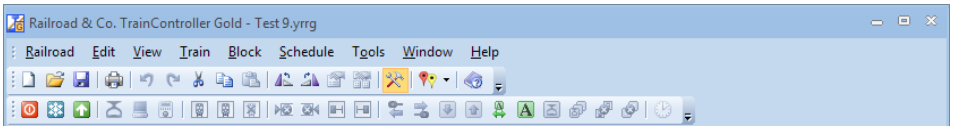



**Diagram 2: Classic User Interface**

## User Interface Design

In addition to the user interface styles known from previous versions **TrainController™ 9** provides more than 30 new user interface designs.

. Among others the following styles are new in version 9:

- Several Office 2013 styles
- Visual Studio 2015, 2013 and 2012
- Windows 10, Windows 8 and Windows 7
- Railroad & Co. 9, 8 and 7
- Several extra styles
- A custom style with the possibility to adjust the primary colors of the user interface to personal taste.

## Window Handling

Floating windows can be maximized or half-screen docked to the boundaries of each computer screen. Maximizing and docking as well as later normalizing and undocking can be performed with the mouse by drag & drop (similar to the Aero Snap feature known from the current versions of Microsoft Windows).

In this way a floating switchboard and a floating dispatcher window, for example, can be easily arranged on top of each other by drag & drop with the mouse, each covering half of the computer screen.

## File Handling

**TrainController™ Gold** provides the option to store the status of the user interface in a separate file, apart from the project file; and to load the status from this separate file.

The status of the user interface concerns all existing windows, whether they are visible or not, their size and position, all customization options for toolbars and menus, and so on.

This option takes effect, when the project file is being stored or loaded. If this option is set, then the status of the user interface is stored twice – once in the project file as usually, and once in the mentioned separate file. When the file is being loaded, then **TrainController™** loads the status of the user interface from the separate file, if available, otherwise from the project file.

This is useful if the project file is edited on several computers. After storing the project file and the status of the user interface in a separate file on computer A and moving the project file (without the separate file) for the first time to computer B, the recent status of the user interface on computer A will be loaded first. This happens, because the status of the user interface (here the status on computer A) is also always stored in the project file and because no separate file is available in this moment on computer B. After storing the project file and the status of the user interface in a separate file on computer B and moving the project file (without the separate file) back to computer A, the original status of the user interface on computer A will be loaded now, because the separate file is still available here.

In this way it is possible to adjust the status of the user interface individually to several computers (e.g. their individual monitor configuration), to move the project file back and forth between several computers and to load on each computer that status of the user interface, which fits best to the particular computer. And if the project file is provided to a new computer, where this file was never loaded, then the most recent status of the user

interface of that computer, where the project file was most recently stored, will be loaded on the new computer. This causes the same result, as if this optional feature is not used, which is the default case.

# 2 The Switchboard

## 2.3 Drawing the Track Diagram

### Smart Gates and Crossing Gates

Gates are smart switchboard symbols, which can be used for automatic control of the gates of engine sheds or all other gates, which can be passed by trains.

To work properly, a gate must be attached to a straight element or a curve element in the switchboard. The gate is automatically opened, if a route passing this track symbol is activated. The gate is automatically closed again, if this route is deactivated.

By using conditions it is possible to influence the operation of gates furthermore.

Crossing gate symbols can be used for automatic control of crossing gates. Crossing gates can expand over more than one switchboard cell. To work properly, the switchboard cells covered by a crossing gate must contain at least one straight element or curve element. The crossing gate is automatically closed, if a route passing one of these track symbols is activated. The gate is automatically opened again, if the last active route passing the crossing gate is deactivated.

The latter covers proper control of multi-track railway crossings. Even if two trains pass the railway crossing at the same time, the crossing gates remain closed, until the last train has left the crossing and cleared the route across the crossing.

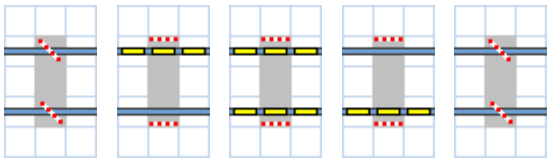


**Diagram 3: Crossing Gates and multi-track Railway Crossing**

The above picture series shows the crossing gates at a multi-track railway crossing, which are automatically closed, when the first route passing over the railway crossing is activated, and opened again, when the last route is deactivated.

### Crossovers

Crossover symbols can be used in **TrainController™ Gold** to form crossovers between parallel straight track sections. Crossover symbols are always arranged in pairs.

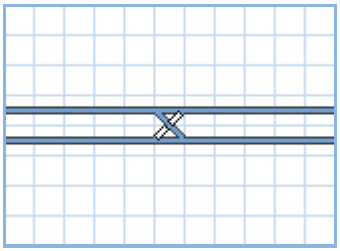


**Diagram 4: Crossover formed by two crossover symbols.**

# 3 Train Control

## 3.5 The Speed Profile

### Using a third party Speed Measurement Facility

**TrainController™ Gold** provides the possibility to perform all speed measurements with third party measurement facilities, such as roller test benches, tacho wagons, speed measurement devices based on photo sensors, etc. This can normally considerably reduce the time needed for the measurement. For this purpose a speed measurement facility is required, which is able to display the measured speed to the human end user. It is not necessary, that the speed measurement facility can be connected a computer.

The following methods for measurement are available:

Measurement of the complete speed profile with a third party speed measurement facility.

By default the measurement will be performed with a semi-automatic, guided procedure. At first the end user places the locomotive accordingly on the track or on the roller test bench, for example. Then he starts the first test run with a mouse click. **TrainController™** puts the locomotive in motion with the first speed needed to be measured. As soon as the measured speed value is displayed by the measurement device and, if necessary, the locomotive has reached a position, where it can start the next run in opposite direction, the end user stops the locomotive with another mouse click. Then he enters the displayed speed value into **TrainController™**. Afterwards he starts the next test run in opposite direction. These test runs are repeated with different speed until all required speed values are measured.

**TrainController™** furthermore provides the ability to perform the complete measurement of all speed values automatically, without the need for intervention by the end user. To support this, the used speed measurement facility must be able to be connected to the computer and must provide some application software on the computer, which is able to (automatically) copy each measured speed value as text to the clipboard. Vendors of speed measurement facilities, which are interested to support the automatic measurement

of the speed profile with their products, should contact Freiwald Software for technical details.

## 3.6 Headlights, Steam and Whistle

Engine functions can be associated with lists of operations. It is possible to assign two lists of functions to each engine function. The first list of operations is executed, when the engine function is turned on; the other list, when the function is turned off. Engine functions, which are associated with lists of operations, can be configured as push buttons, on/off switches or hidden.

### The Engine Functions Library

The engine function library contains the predefined names (e.g. “Light”) and symbols for the available functions. Each function to be assigned to an engine is selected from this library. **TrainController™** is delivered with a default set of predefined functions and symbols. You can also add new functions and draw your own symbols; you can also customize the names and symbols of existing functions to your personal needs.

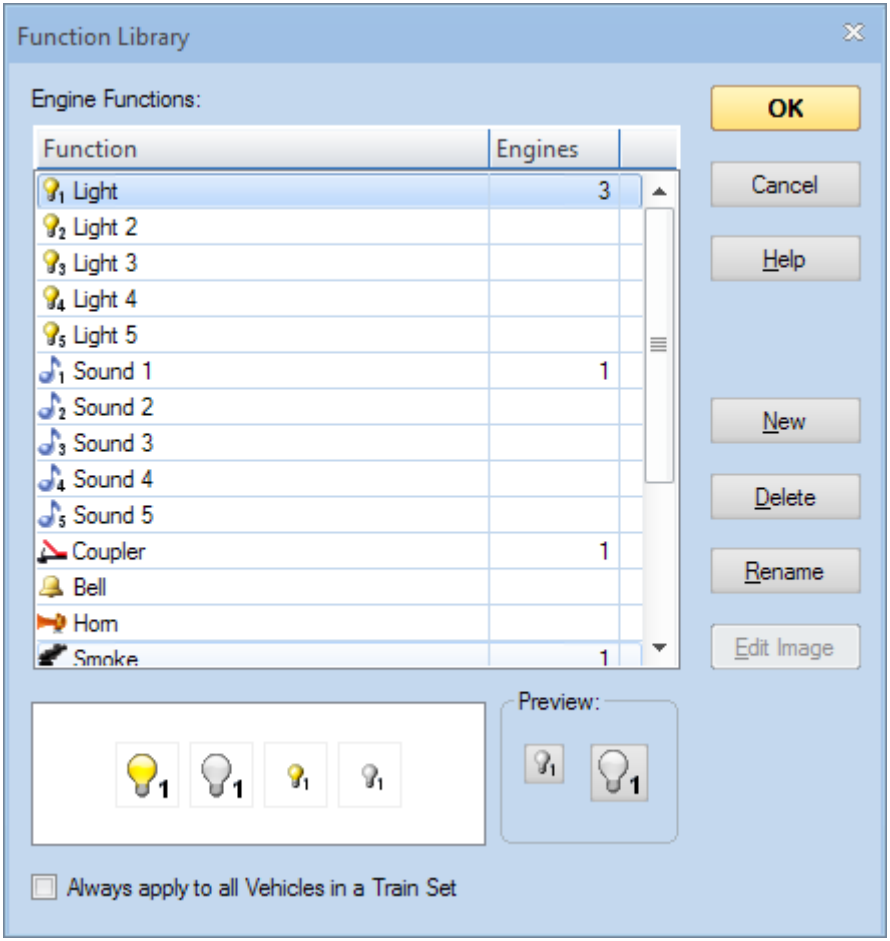



**Diagram 5: Engine Functions Library**

By using the option **Always apply to all Vehicles in a Train Set** the currently selected function will always be applied to all vehicles in a train set. This option is useful for functions like interior lights of vehicles, for example, which will be always turned on or off for all vehicles in a train set at the same time. If this option is set for a certain function and this function is turned on or off in the train window for a vehicle in a train set, then this function will be toggled accordingly for all other vehicles in the same train set, too.

# 5 The Visual Dispatcher I

## 5.8 Arranging Indicators and Markers in a Block

**Variable Stop Locations in a Block – Stopping for Coupling and Line-Up**

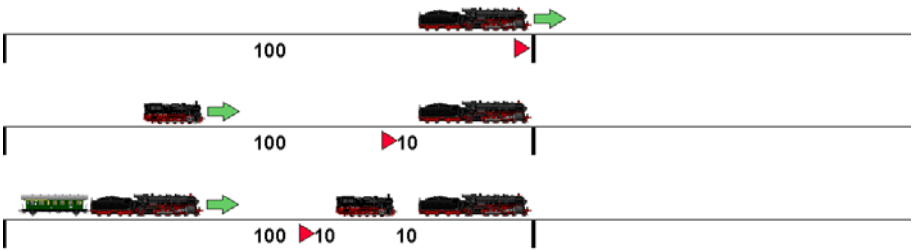
In addition to the features described in the Users Guide of **TrainController™ 8 Gold** the following features apply to **TrainController™ 9 Gold**:

Wildcards begin with one of the characters **%**, **#** or **?**

**%** The wildcard represents the total length of the vehicles, which this wildcard refers to.
**#** The wildcard represents the number of the vehicles, which this wildcard refers to.
**?** The wildcard is replaced by the value **1**, if one or more vehicles refer to it. If no vehicle refers to this wildcard, then the value is **0**.

**Examples:**

- **100-%BA-?BA*10:**




This formula can be used for line-up of vehicles in a block (see also page 36). ?BA*10 describes the gap between the lined-up vehicles.
When the first train enters the block, %BA and ?BA are both 0 and the train stops at 100 cm.
When the second train enters the block, %BA equals to the length of the first train and ?BA is 1. This causes the second train to stop 10 cm behind the first.
When the third train enters the block, %BA equals to the total length of the waiting vehicles (including the gap between them) and ?BA is still 1. This causes the third train to stop 10 cm behind the second.

The variable wildcard **%V** (see page 56) can be used in these formulas, too. It has a specific meaning here. These wildcards are replaced in the first pass of the calculation by the content of the corresponding variable. The resulting formula is then calculated in the second pass in the usual manner like a formula without **%V** wildcards.

## 5.13 AutoTrain – Start of Schedules made Easy

**AutoTrain™** is another outstanding feature of **TrainController™**. With **AutoTrain™** you can run automatic trains at any time during operation without the need to define schedules in advance.

### Auto Train by Drag & Drop

The fastest way to run **AutoTrain™** is Drag & Drop with the mouse:

- Select the **Operation** tab (or the **Schedule** menu in the classic user interface) and call the **AutoTrain by Drag and Drop** command.

- Move the mouse cursor to the start block.
- Press and hold the left mouse button on one of the following symbols:
   , if you want the train to exit the start block to the left.
   , if you want the train to exit the start block to the right.
- Hold the left mouse button pressed and drag the mouse to the block in the block diagram or in the switchboard, where the train will stop.
- Release the left mouse button over one of the following symbols:
   , if you want the train to enter the destination block from the right to the left.
   , if you want the train to enter the destination block from the left to the right.
- The train will now start and run automatically to the destination block.

Some additional options are available:

 Click this option, if you want to leave the **AutoTrain by Drag and Drop** command turned on after releasing the mouse button. In this case another AutoTrain run can be initiated by drag and drop without the need to call the **AutoTrain by Drag and Drop** command once more.

 By pressing and holding the mouse button on this symbol it is possible to specify all blocks in a station (see section 15.7, “Stations”) as start blocks with exit of the train to the left. This station is determined by the block, where the mouse is pointing to. It is the station, where the block is located in.
This option works in the same way as if all blocks in the said station are specified as start blocks of a schedule.

 Same as above but with exit of the train to the right.

 By releasing the mouse button over this symbol it is possible to specify all blocks in a station as destination blocks with entry of the train from the right to the left. This station is determined by the block, where the mouse has been dragged to. It is the station, where the block is located in.
This option works in the same way as if all blocks in the said station are specified as destination blocks of a schedule.

Same as above but with entry of the train from the left to the right.

Click this option, if you want to perform a local AutoTrain run (see page 65, “Local Schedules”).

The above options allow to perform an AutoTrain run from a station to a station with only one single drag and drop operation. It is also possible to mix the options for blocks and stations. In this way an AutoTrain run can be performed from an individual block to a station or vice versa.

The above symbols are valid for horizontal blocks. Symbols for vertical blocks look accordingly.

**AutoTrain with Start and Destination Keys**

In addition to the possibilities provided by **TrainController™ 8 Gold** it is also possible in **TrainController™ 9 Gold** to perform AutoTrain with start and destination keys also from an arbitrary block in a station or to an arbitrary block in a station.

These operations are also applied to blocks like the operations for AutoTrain with start and destination keys known from **TrainController™ 8 Gold** and work in a similar way. But instead of selecting only the block, where this operation is applied to, all blocks in the same station are selected, where this block is located in.

Therefore these new operations do not only select one single block as start or destination block of the AutoTrain run, but all blocks in the same station.

It is also possible to mix the operations effective for individual blocks with the new operations effective for all blocks in a station. In this way it is possible to perform an AutoTrain run from an individual block to a station or vice versa.

## 5.18 Putting it all together – The Dispatcher Window

The *dispatcher window* serves as display for the block system of your layout. It lists and displays all diagrams, blocks, routes and schedules.

In **TrainController™ Gold** the dispatcher window furthermore displays all stations (see section 15.7, “Stations”) and boosters (see section 15.8, “Booster”).

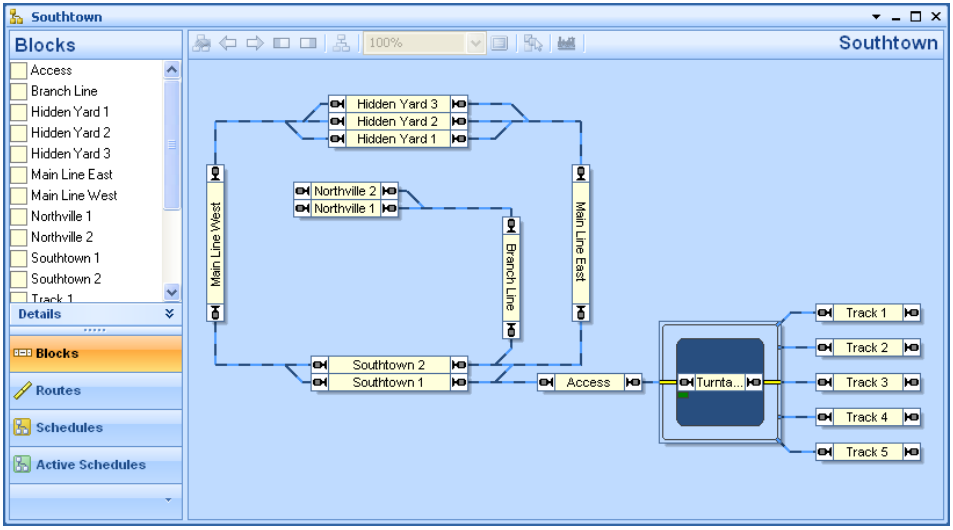



**Diagram 6: Dispatcher Window**

The dispatcher window is split into two parts. The left part lists the blocks, routes or schedules of your layout.

The left part of the dispatcher window of **TrainController™ Gold** furthermore lists all stations and boosters.

The right part of the dispatcher window displays the currently selected block or schedule diagram.

The right part of the dispatcher window of **TrainController™ Gold** may also display the currently selected station or booster diagram.

# 8 The Message Window and Pins

## 8.1 Pins

Pins are an outstanding and unique feature of **TrainController™**. Pins make error diagnostics and debugging significantly easier and much more intuitive by showing information, where the events happen, rather than in a plain list like in the message window. Thereby troubleshooting becomes literally much more targeted.

Pins are an important expression of our aspirations to make train operations not only as realistic as possible, but also to make the path to this goal as easy and intuitive as it can be.

Pins are displayed in different types and colors on the computer screen with small markers.

Pins can be made visible with commands in the **Pins** group of the **View** tab of **TrainController™**. While pins are visible, the color of the windows, where pins are displayed, is being changed to gray scale. Thereby it is easier to locate the pins.

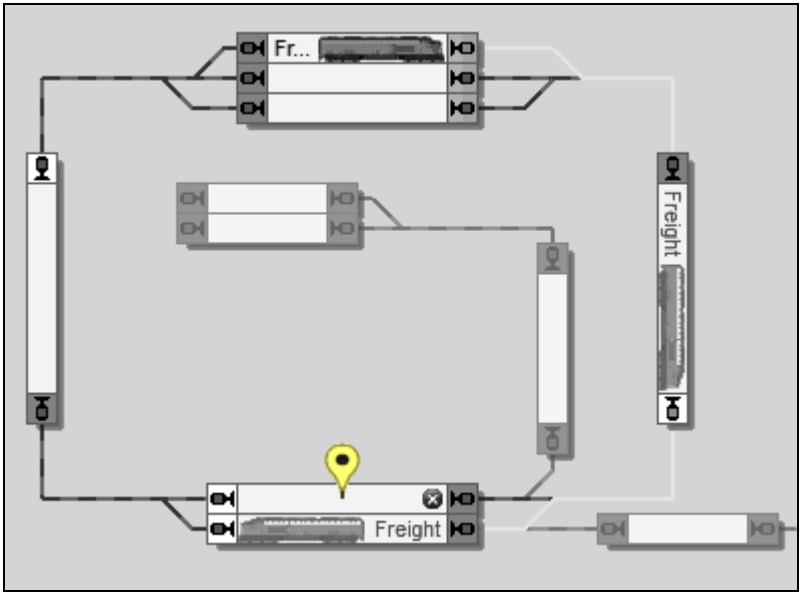



**Diagram 7: Display of a pin in the Dispatcher Window**

By clicking on the marker of a pin the pin can be expanded and collapsed. An expanded pin shows a window, which displays the information of this pin.

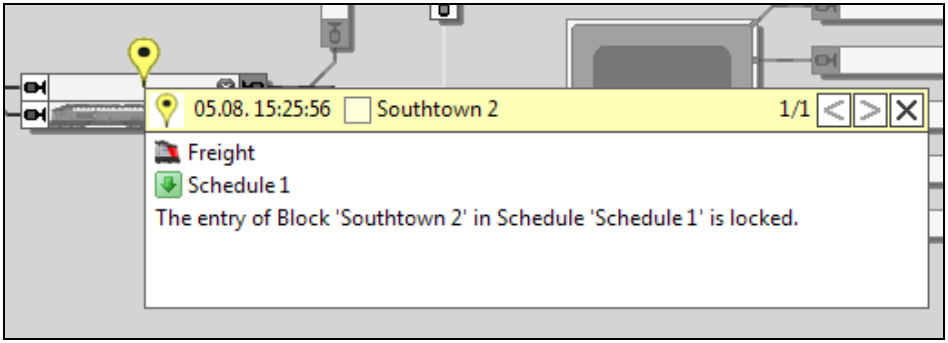



**Diagram 8: Expanded pin in the Dispatcher Window**

Diagram 7 shows an active schedule, which contains the blocks “Southtown 1” and “Southtown 2” at the bottom. The train chooses “Southtown 1” for a certain reason and a yellow pin is displayed in block “Southtown 2”. If you want to know, why the train did not choose “Southtown 2”, click to the pin. The pin is expanded and displays the reason (Diagram 8): “Southtown 2” was not choosen, because its entry was locked.

This is a very simple example, but in more complex situations you will come to appreciate, that important information is displayed, where the events happen.

There are the following types of pins:

- System pins
- Dr. Railroad pins
- User pins

### System Pins

System pins are automatically created by **TrainController™** during the operation of the layout. In many cases system pins are created together with the messages, which can be made visible in the message window. But there are also many pins, which do not have a corresponding message.

System pins are implicitly filtered to the context of the window, where they are displayed. By selecting a certain train in the switchboard or dispatcher window, for example, only the systems pins associated with this train are displayed. The diagram of a schedule in the dispatcher window, for example, displays only the pins which were generated by the execution of this schedule. System pins can also be filtered explicitly to the pins generated in the current session or during the most recent execution of a schedule. These features support the focus on the key information for the current debugging.

System pins are displayed in three colors:

- **Red / Error:**
  Red pins appear, if a certain operation fails, e.g. the execution of a schedule.
- **Yellow / Warning:**
  Yellow pins appear, if a certain operation is performed with limitations or another operation is chosen instead.
- **Green / Ok:**
  Green pins indicate the successful execution of operations. Green pins can be used, for example, to track the path of a train under control of a schedule.
  The display of green pins is turned off by default to allow you to focus on possible problems.
  The diagram below shows the same situation like Diagram 7, this time with the display of green pins turned on. The path chosen by the train is clearly visible now.

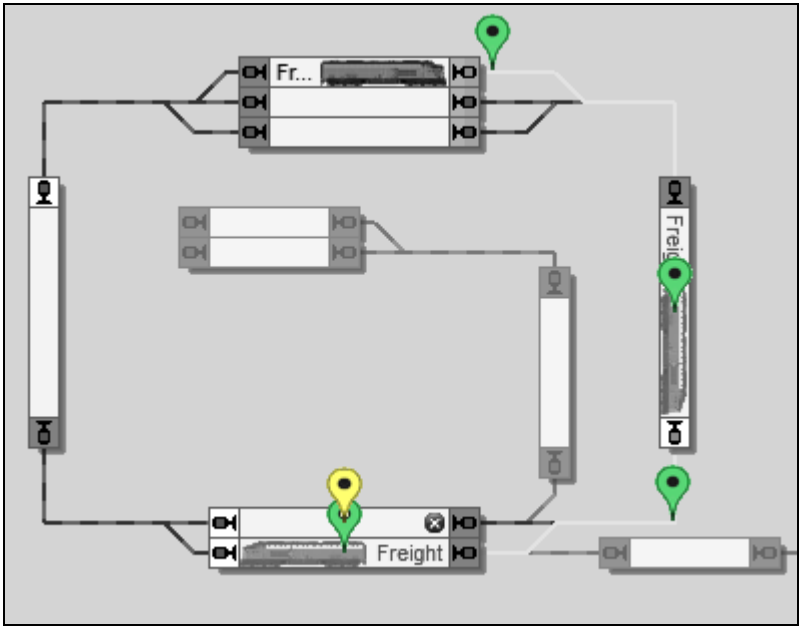



**Diagram 9: Green pins marking the path of a train in the Dispatcher Window**

### Dr. Railroad Pins

The information generated by Dr. Railroad can be made visible with pins, too. These pins correspond to Dr. Railroad errors and questions, which are normally displayed in the message window. Dr. Railroad pins appear at the location of erroneous objects on the computer screen.

Dr. Railroad pins are displayed in two colors:

- **Red / Warning**
- **Yellow / Question**

### User Pins

User pins can be created by the end user. After selecting an object on the computer screen and creating a user pin a marker appears at the location of this object. The text, which belongs to each user pin, can be freely edited.

User pins are displayed with blue color. They can be deleted at any time and are stored in the project file.

User pins are very useful to leave notes for further processing of the data for the end user himself or other users.

If an end user has to interrupt editing of data of a specific object, for example, he can create a user pin for this object with a note, what to do next, when the work is continued.

If an end user has a problem with the execution of a certain schedule by a certain train, and wants to consult an expert user, for example, he can create one or more user pins with details about his problem, e.g. for the block of the schedule, where the train is currently located, leave these pins expanded, store the project file and send the file to the expert user. When the expert user opens the project file, the user pins showing the location as well as the detail information of the problem will become visible in expanded state on the screen at once. This is a significant improvement for the exchange of relevant information in such cases.

# 11 Advanced Train Control

## 11.1 Train Sets in TrainController™ Silver

A train set is composed of a couple of engines. Train sets can be created, arranged and dissolved at any time during operation of the layout.

Train sets are used to accomplish realistic *multiple unit* operation, i.e. operation of trains, that contain more than one engine.

Similar to real railroads each single engine, which is operated separately, or each car, which is located isolated on your layout, can also be seen as a train. For this reason the term train is usually used in **TrainController™** as a generic term for each engine, isolated car or complete train set.

Train sets can be arranged via the Train List by dragging the symbols of engines with the mouse. They can also be arranged with the **Train Set dialog box**.

To operate the speed or direction of a train set select an arbitrary engine currently contained in this train set in the Train Window. Changing the speed or direction of this vehicle will also be applied accordingly to all other engines contained in the train set. The speed and direction controls of the Train Window always reflect the status of the selected individual vehicle rather than the complete train set. If there are several engines with different speed characteristics assigned to a train set, i.e. the engines run with different speed at the same speed step, then **TrainController™** is able to balance out the different behavior of the engines. This requires correct adjustment of the *speed profile* of each affected engine, however (see section 3.5, “The Speed Profile”).

Multiple unit operation via train sets is also possible with the throttle of your digital system. To operate the speed or direction of a train set select an arbitrary engine currently contained in this train set on the throttle of your digital system. Changing the speed or direction of this vehicle with the digital throttle will also be applied accordingly by **TrainController™** to all other engines contained in the train set.

To operate the auxiliary functions of a certain engine select this particular vehicle in the Train Window, too. The operation of functions, however, only applies to the currently selected vehicle. The function buttons of the Train Window always reflect the status of the currently selected vehicle. In other words: for manual operation of functions it does

not matter, if the vehicle is currently contained in a train set or not. The effect is limited to the particular vehicle.

Like in real railroads a certain vehicle can only run as part of one train at a time. If a vehicle is successfully added to a train set, then it is automatically removed from its previous train set, if any.

#### Operation of Additional Function Only Decoders in TrainController™ Silver

Function only decoders are often used to add additional functions to a decoder controlled locomotive or to other rolling stock. An example is lighting in passenger cars. These decoders can be controlled with **TrainController™ Silver**, too.

This is done by setting up a “dummy engine” with the digital address of the function only decoder. The speed settings of this decoder don't matter in this case. The function setup of this decoder is done as outlined in section 3.6, “Headlights, Steam and Whistle”.

Manual operation of the extra functions provided by the function only decoder is done by selecting the “dummy engine” in the *Train Window* and operating the function buttons of this engine.

For automatic operation of the extra functions provided by a function only decoder it is necessary to arrange a train set and to setup the “dummy engine” representing the function only decoder along with the actual engine as multiple unit. If different function symbols are used for the functions of the actual engine and the functions provided by the function only decoder, then it is possible to select and activate the specific functions of the function only decoder automatically without affecting the functions of the actual engine.

## 11.2 Cars and Train Sets

#### Line-Up of Vehicles and Trains in a single Block

X

With **TrainController™ Gold** it is possible to line up more than one vehicle or train in a single block without connecting them to a continuous train set. This is for example useful, if you want to park several locomotives in a single track of a hidden yard.

Several features of **TrainController™ Gold** must be combined in order to accomplish this.

First the schedule rule **Line up in destination block** (see page 61) must be used in a schedule. This rule enables trains under control of this schedule to enter a destination block of this schedule, even if this block is already reserved by other vehicles. This rule works similar to the rule **Enter reserved destination block for joining** with the difference, that the incoming vehicles are expected to be stopped in a certain distance from the already waiting vehicles. This distance must be specified to activate the rule described above. The incoming vehicles are not connected with the waiting vehicles to a continuous train set. Both groups of vehicles remain separated. It is furthermore possible to line up vehicles in more than two separated groups.

If a manually operated train enters a block, where several vehicles are already lined-up in separated groups as described above, then the incoming vehicles are added to the waiting vehicles as a new group, which remains separated from the waiting vehicles.

Note, that line-up of vehicles in a certain block can only be initiated with a schedule and the rule described above, but not by train tracking. Once the vehicles in a block are lined-up in at least two separated groups, then each manually operated train subsequently tracked into this block by train tracking will form a new separated group.

Line-up by a schedule requires, that the distances and ramps of stop and brake markers in the concerning block are configured accordingly. Appropriate formulas should be used to calculate the space already occupied by the waiting vehicles and the additional gap between the waiting vehicles and the incoming vehicles (see also page 25). This is very important, because each incoming train under control of a schedule is expected to be stopped in the distance, which has been specified to activate the rule for line-up.

The assumed distances are used, when the lined-up vehicles are moved up. This happens automatically, when a group of vehicles at one of the two sides of the block leaves the block. If a group of vehicles at one end of the queue is moved to another block, then the group, which previously was the second group in this queue, is automatically moved towards the end of the block. The distance of this move results from the length of the vehicles in the first group plus the gap between both groups. Once the second group has completed this move, then the group, which previously was the third group in the queue, performs the same move, afterwards the fourth group, and so on. To enable the software to calculate the distances of this move accordingly, it is very important to specify all distances and lengths correctly. This includes also the maximum train length of the concerning block, all distances and ramps of all involved markers as well as the length of each involved vehicle, engine or car.

Note also, that line-up and move up of vehicles can only be performed in one direction at a time.

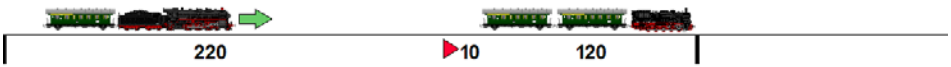



**Diagram 10: Line-Up – Step 1**

Assume, for example, that a train with length 100 cm is directed by a schedule from the left to the right into a destination block with maximum train length of 350 cm and another train with length 120 cm already waiting there.

The distance specified in the schedule rule for line-up is 10 cm. In this case the software assumes, that the waiting train is located at the right end of the block and the remaining space in the left part of the block is 220cm (350 cm minus 120 cm minus 10cm).

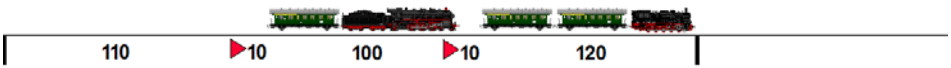



**Diagram 11: Line-Up – Step 2**

The incoming train will be stopped in a distance of 10 cm to the waiting train. Now 110 cm (220 cm minus 100cm minus 10cm) are still available in the left part of the block.

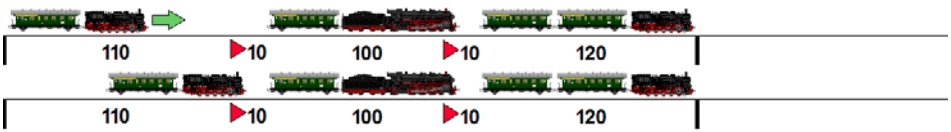



**Diagram 12: Line-Up – Step 3**

With the same schedule a third train with a length of up to 110 cm can be directed here – but also only if it enters the block from the left to the right.

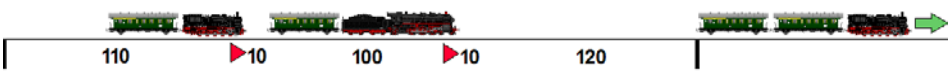



**Diagram 13: First train has left the block.**

When the first train (length 120 cm) leaves the block to the right later, then free space of 130 cm (120 cm plus 10 cm) becomes available in the right part of the block.


**Diagram 14: Move up – Step 1**

The second train in the queue will now automatically move up by 130 cm to the right.

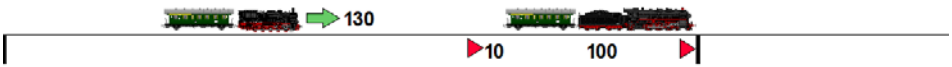



**Diagram 15: Move up – Step 2**

When this has been done, the third train will perform the same move.

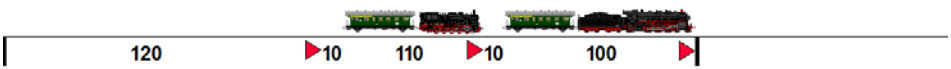



**Diagram 16: Move up finished**

Finally both trains are located at the right side of the block with 120cm free space in the left part of the block. Now another train with a length of up to 120 cm can be directed here - if it enters the block from the left to the right.

Theoretically the filled block could also left by the last incoming train to the left. For reasons of simplicity we strongly recommend, however, to perform line-up and move up for a certain block only in one direction – following a strict FIFO (first in – first out) principle.

**!**

**For reasons of simplicity the software does not record the information, in which direction line-up and move up have been performed most recently. It just compares the available and the required space with each other.** It assumes, that waiting trains are always located at suitable positions in the block and it derives the distance for the move up of the remaining trains solely from the length and the gap of the leaving train.

To perform line-up and move up accordingly, perform the following:

- Specify the maximum train length of the concerning block accordingly.
- Specify the length of each involved vehicle – engine and car – correctly.

- Capture the speed profile and brake compensation of each involved engine.
- Specify a sufficient value for the gap between two groups of vehicles in the schedule rule for line-up (at least 10cm or 4 inches – better more).
- Specify correct formulas for the distances and ramps of all involved markers (see also page 25). With according formulas, line-up in a block can be controlled with one single brake and one stop marker (for each direction).
- Ensure that the required markers are triggered. Line-up can be performed with one single sensor in a block. The sensor, however, which triggers the brake and stop markers used for line-up, must not be occupied by waiting trains.
- Perform line-up and move up in one direction (FIFO principle).

Additional information: the default value assumed by the software for the gap between vehicles, which are lined-up by train tracking is 10cm (4 inches). Hence if you drive a train manually to a block, where it is lined up with waiting vehicles, ensure, that the train is stopped with a distance of about 10cm (4 inches) to that vehicles.

## 11.3 Approved Trains

### Use of Vehicle Groups in TrainController™ Silver

For all blocks, routes and schedules in **TrainController™ Silver** it is possible to specify, which engines may use the concerning object. To open one block only for certain engines, for example, the concerning engines are added to a list in the properties of the block. Such list may also contain vehicle groups.

Such a vehicle group can contain locomotives and other vehicle groups.

An engine is included in a vehicle group when the engine is entered directly in the vehicle group, or if the vehicle group contains another vehicle group, in which in the engine is included.

### Vehicle Groups and Train Descriptions in TrainController™ Silver

In **TrainController™ Silver** it is possible for blocks, routes and schedules to specify those engines or train sets, that are allowed to use the object in question. This is done with train descriptions.

A train description consists of a list of engines. This list may also include vehicle groups, where other engines or again vehicle groups are included.

For a train description it is also possible to specify, whether at least one vehicle or whether all vehicles in a train set must be included in this list. If a train consists of a single locomotive, this information plays no role. But when it comes to a train set, it can thus be determined, whether the train set may contain engines, which are not contained in the list, or whether all the engines in the train set must be covered by this list.

If, for example, all trains on the layout do not contain more than one engine, and the non-electrified branch line is only open for steam and diesel engines, then this can be covered with a train description, which contains only steam and diesel engines. For a train with an electric locomotive, but without steam and diesel engine on the other hand, the description does not apply.

It is also possible to exclude individual entries from the list of vehicles in a train description.

X

**Train Descriptions with marked Vehicle Positions**

For specific purposes it is possible to mark selected positions in a train description. These markings can be used to apply certain train operations to individual vehicles in a train set.

Examples are:

- Execution of a vehicle function (e.g. operation of interior lights only by individual cars in a train set.
- Separating a train set left or right of a certain vehicle

.

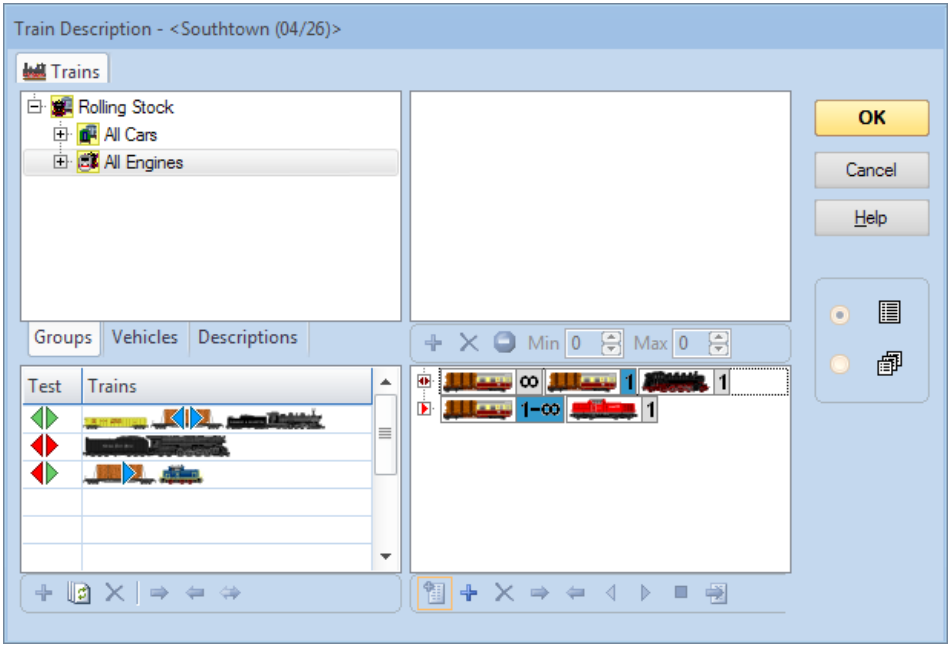



**Diagram 17: Train Description with marked Vehicle Positions**

Diagram 17 shows a train description, which applies to trains with one engine and one or more cars. The position of the first car is marked. This is indicated with a blue marking in the lower right area of the dialog box.

In the test window in the lower left area of the dialog box the matching cars are marked with a blue marking for each direction of travel. Note, that the matching cars may be different for the different directions of travel.

If a train description contains more than one line, then the marked positions in the first line, which matches a given train, apply to the vehicles of this train.

In Diagram 17 in the test area, three train sets were arranged. The first train is a steam locomotive with two freight cars. This train fits exactly on the first line of the train description. A green marking appears. The first car is marked with a blue arrow for both directions of travel, because this car matches the marked position in the first line for both directions of travel.

The second train being tested is a single steam locomotive. The description always requires at least one car. Therefore, here a red marking and no blue markings appears.

The third train matches the second line of the train description, but only in forward direction. Therefore the matching car is displayed with a blue marking only for the forward direction.

# 14 Extended Control and Monitoring Functions

## 14.3 Protection and Locking with Conditions

### System Events and States

With **TrainController™ Gold** it is possible to evaluate the occurrence of certain global events or states in conditions or triggers (e.g. of flagman or signal objects).

A system event or state occurs globally and is not bound to the status of an individual object.

System events signal the occurrence of a certain system wide situation. They are visible for evaluation only for a short moment and can be evaluated only in triggers of objects.

System states indicate, that the system is currently in a certain state. System states cannot only be evaluated in triggers, but also in the conditions of objects.

Among others the following system events and states can be evaluated:

- **Begin of Session (Event):**
  This event occurs at the beginning of a session, after the project file has been completely loaded and everything has been completely initialized. By using this event in a trigger of a flagman, for example, it is possible to execute certain operations automatically at the beginning of each session.
- **Continue after Stop or Freeze (Event):**
  This event occurs, when operation is continued after freeze or stop.
- **Outside Edit Mode (State):**
  This state indicates, whether the software is running outside of the edit mode.
- **Offline Mode (State):**
  This state indicates, whether the software is running in offline mode or not.
- **Open Connection to Digital Systems (Event):**
  This event occurs, when the connections to the attached digital systems have been opened.

- **Loss of Connection to a Digital System (Event):**
  This event occurs, when the connection to one of the attached digital systems got lost.
- **Clock Running (State):**
  This state indicates, whether the clock is currently running or not.
- **Active Schedules (State):**
  This state indicates, whether at least one schedule is currently running or not.
- **Schedule Aberration (Event):**
  This event occurs, when the limited aberration protection detects an aberration in an active schedule.
- **Schedule lost Car (Event):**
  This event occurs, when an apparent lost car has been detected during an active schedule.
- **Schedule Watchdog (Event):**
  This event occurs, when the schedule watchdog detects a train under control of a schedule, that apparently got stuck.
- **Turnout Position Control (Event):**
  This event occurs, when turnout position control of a turnout used in an active route detects an erroneous turnout position.

**For security reasons the software asks the user for the permission to evaluate system events or states in triggers, when a project file is being loaded, which contains triggers with system events or states. This security question can be disabled. But you should disable it only, if you are loading only files from yourself or from people you trust.**

## 14.4 Operations

### System Operations

The system operations **Execute Program**, **Sound File**, **Message**, **Popup Message**, **Stop all Trains**, **Set Clock**, **Select Object** allow to use a variable or formula (see section 14.14, “Variables”) as value.

### Control Flow Operations

The following control flow operations have been extended in **TrainController™ 9 Gold**:

- **Prerequisite:**
  This operation is always associated with a condition. If the condition is valid, then that operation is executed, which follows directly after the prerequisite. If the condition is not valid at the time of execution of the operation, the next operation will be ignored and the subsequent operation will be executed.
  If the operation immediately following a prerequisite is a **Goto**, then the execution of blocks of operations can be made dependent on the prerequisite.

**TrainController™ 9 Gold** provides the following new control flow operation:

- **Access to Variable:**
  This operation can be used to access a variable (see section 14.14, “Variables”).

The operations **Delay**, **Random Delay** and **Probability** allow to use a formula as value.

### Train Operations

The following train operations have been extended in **TrainController™ 9 Gold**:

- **Execution of a train function:**
  This train operation now allows to specify, which vehicle performs the operation, if this operation is called for a train set.

  **First Vehicle:**
  The auxiliary function of the first vehicle in the train set is operated.

  **Last Vehicle:**
  The auxiliary function of the last vehicle in the train set is operated.

  **All Vehicles:**
  The auxiliary function of all vehicles in the train set is operated.

  **Forwarding:**
  The vehicles are determined by the state of the function forwarding.
  (This was the default in **TrainController™ 8**.)

  **Description:**
  The vehicles are determined by a train description with marked vehicle positions (see page 41).

- **Separation of train sets:**
  This train operation now provides the following options:

  **Separate Locomotive:**
  This train operation separates the locomotive from the rest of a train set.

  **Separate rightmost/bottommost/leftmost/topmost Locomotive:**
  This train operation separates that locomotive from the rest of a train set, which is located at the according end of the train set.

  **Separate all rightmost/bottommost/leftmost/topmost Locomotives:**
  This train operation separates all locomotives from the rest of a train set, which are located at the according end of the train set.

  **Separate rightmost/bottommost/leftmost/topmost Locomotive or Car:**
  This train operation separates that vehicle from the rest of a train set, which is located at the according end of the train set.

  **Separate right/bottom/left/top**
  This train operation is usually applied to a single vehicle. It separates a train set at the specified side of this vehicle.

  **Separate to the right/below of/to the left/above of the marked vehicle:**
  This train operation separates the train set at the specified side of a vehicle. This vehicle is determined by a train description with a marked vehicle position (see page 41).

**TrainController™ 9 Gold** provides the following new train operation:

- **Display Name:**
  This train operation sets a display name for the corresponding train. This name is used instead of the stored name of the train for display on the computer screen. This operation allows to display variable train names, which depend on schedules, for example, or other operational aspects. This train operation can only be used for trains under control of a schedule (e.g. at the beginning of a schedule). When the train terminates its current schedule the display name is deleted.

.

- **Short distance move:**
  Performs a move of the train at slow speed by the specified distance to the specified direction. Unlike all other train operations, which set directly the speed or direction of the train, this operation is also allowed for trains under control of a schedule. However, a train under control of a schedule must perform a scheduled wait, when this operation is initiated and the short distance move must be completed, before the scheduled wait ends. Furthermore a train under control of a schedule must not leave its current block due to execution of this operation.
  This train operation can be used for coupling maneuvers during active schedules.

The train operations **AutoTrain**, **Start Schedule**, **Temporary Speed Limit**, **Short Distance Move** allow to use a variable or formula (see section 14.14, “Variables”) as value.

## 14.7 Prototypical Signaling

### Evaluating the state of distant Signals

The internally calculated aspect of distant signals can be directly evaluated in **TrainController™ Gold** in triggers and conditions. This is useful for distant signals, which reflect the state of more than one main signal.

Assume a distant signal located at the entry into a station with five tracks. The state of the distant signal reflects the state of the main signal located at that track, which will be passed by the train. In total there are five main signals. In order to evaluate the right main signal in the trigger of the distant signal it is necessary to combine the status of all main signals with the status of the possible routes to the five tracks. This results in a relatively complex structure of AND and OR conditions containing the aspects of the five main signals as well as the status of all possible routes to the five tracks. Such structure must be created for each aspect of the distant signal.

To make this much easier **TrainController™ Gold** provides the possibility to evaluate directly the internally calculated aspect of the distant signal for the said entry block. In the trigger of each aspect of the distant signal located at the exit of a certain block, the complex structure described above can be replaced by one single entry, which represents the internally calculated aspect of the distant signal for this block.

## 14.13 Extended Accessories, Cranes and Functional Models

X

### Using Extended Accessories in Engine Functions

In order to understand the following description, we assume a fictive locomotive decoder in which specific functions are not controlled by digital function commands, but by speed or direction commands. For the sake of simplicity, we assume a decoder, with which the couplers of a locomotive are controlled by the locomotive function F1. The side (front or rear) of the vehicle is selected by setting the direction of travel in the decoder to forward (for selecting the front coupler) or reverse (for the rear coupler) before calling F1.

We assume that this decoder is installed in a locomotive in addition to the decoder for motor control.

The couplings of the locomotive should now be controlled by the function buttons in the locomotive.

The two decoders cannot be controlled as a multiple unit or train set, since otherwise the control of the couplers would influence the driving direction of the locomotive, which of course is not desired. It is also not possible to transmit digital commands for the direction of travel directly via train operations or the like.

However, extended accessories can be used for this purpose. For this purpose, a simple extended accessory with a switch with two states ("front coupler" and "rear coupler") as the only control is sufficient. In the operations for the "front coupler" state, operations are entered for setting the direction of travel to forward and turning on F1, and, if necessary, after a delay, to turn off F1. The operations for the "rear coupler" state are arranged correspondingly.

For each engine in which this decoder is installed as an additional function decoder, a symbol of this extended accessory with the corresponding engine address as the basic address must be created. The call to this symbol is then entered as an operation for a locomotive function of the corresponding engine.

Let us assume, for example, that these are the three locomotives with the addresses 14, 24 and 34, and that the digital address of the function decoder is by 1 higher (i.e. 15, 25 and 35). In order to control the couplers in these three locomotives, three symbols of the extended accessory must be created, which control the addresses 15, 25 and 35 accordingly with their operations,

The call of the symbol for the address 15 is then entered as an operation for a locomotive function of engine 14. The other engines are configured accordingly.

### Extended Accessories and Variables

In the example of the previous section, for each additional locomotive with the described function decoder a symbol of the extended accessory must be created again.

It would be more elegant and clearer if a single symbol of this extended accessory could be used for all such engines. For this purpose, however, the address of the calling engine would have to be evaluated when calling the symbol in an engine.

To support this, formulas with variables can be used instead of fixed values in certain operations of extended accessories (see section 14.14, “Variables”).

The values, which result from the calculation of these formulas, can not only hold the parameters of the digital command (such as speed step or function number), but also the address offset, which is added to the base address of the extended accessory to determine the actual digital address to be controlled.

If such digital command is executed by the operations of an extended accessory, which again is called by an engine function, then it is even possible to store the digital address of the vehicle, which is currently executing these operations, in a variable (by means of the context vehicle; see page 52). The value of this variable can then be used to calculate the address offset for the issued digital command.

In short terms: The digital address of a vehicle, which calls an extended accessory in its functions, can be used to calculate the address, which the digital commands are issued to by the extended accessory.

Thus, in the example described above, it is possible to use a single symbol of the extended accessory. The digital address of the corresponding function decoder (15, 25, or 35) can then always be derived from the address of the calling locomotive (14, 24 or 34).

It is also possible to evaluate the position and orientation of a vehicle in its current train set (if any) with variables. This information can then even be used in the example described above to control not only the coupler in terms of “front” and “rear”, but also in terms of “right or left side of the locomotive”, regardless how the train set is standing on the track and how the locomotive is set up in the train.

## 14.14 Variables

### General

Variables can be used to make operations, conditions, triggers and many other options more flexible. By using a variable instead of a fixed value for a specific option it is possible to change the value of this option at runtime and depending of the current operational situation.

Variables can be used

- to count events happening during operation and evaluate the status
- to change the name of trains displayed in blocks dynamically
- to perform arithmetic calculations and evaluate the result
- to select an object out of a plurality of identical objects for operation with one simple generic macro
- to change the delays or probabilities of operations in a list of operations dynamically
- to affect the speed of temporary speed limits dynamically
- to affect the distances or ramps of markers or the distances of short distant moves dynamically
- and so on …

The possibilities are virtually not limited.

Variables are usually created by editing the operations, which access the variables. With the exception of local variables (see page 58) all other variables can be created, edited and deleted via the Explorer window.

### Type of Variables

Each variable has a specific type, which is determined during creation of the variable. The following types of variables are available:

- **Number:**
  Number variables are used to store numeric values. Numeric variables can be used among others as counters, for numeric calculations and for program options, which numeric values are assigned to (e.g. the train operation **temporary speed limit** with a variable speed setting).

- **Text:**
  Text variables are used to store text strings. Text variables can be used among others for messages or for other program options, which text strings are assigned to (names of sound files, labels of goto operations, display name of trains, etc).
- **Time:**
  Time variables are used to store time values. Time variables can be used among others for operations, which use a time value (e.g. to set the current time of the clock, delay operations, etc.). The unit of the values stored in time variables is milliseconds.
- **Object:**
  Object variables are used to store references to objects. Object variables can be used to perform operations with the objects stored therein. Object variables are always bound to a specific type of object (e.g. two aspect signal), which is determined during creation of the variable. Only references to objects with this type can be stored in the variable. With object variables it is for example possible to create multi-purpose macros.
  Assume a macro, which contains an operation of a variable for two aspect signals in its list of operations. This operation sets the signal currently stored in the variable to green. This single macro can be used to operate a plurality of two aspect signals, one at a time. If the reference to a specific signal is assigned to the said variable prior to the call of the macro, then this signal will be operated by the macro. Assigning another signal prior to the next call of the macro will cause the macro to operate the other signal.

### Context Objects of Variables

If a train operation is executed by a marker in **TrainController™**, then this operation is applied to the train reserving the block in which the marker is located. At the time of the operation, this is the *context* train. This already known principle is extended for variables as follows:

Variables are mainly used in the operations, triggers or conditions of objects. This determines also the *context* objects, which an access to a variable applies to.

The following table describes the conjunction between the access of a variable and the particular context objects for the most common cases:

| Use of a variable by: | Context Train | Context Block | Context Route | Context Schedule |
|---|---|---|---|---|
| train operations | the train itself | current block of the context train | - | current schedule of the context train |
| engine function | the containing train set, if any; otherwise the engine itself | current block of the context train | - | current schedule of the context train |
| Block | the train, which is currently reserving the block | the block itself | - | current schedule of the context train |
| marker or indicator in a block | the train reserving the context block | the block, which contains the marker or sensor | - | current schedule of the context train |
| push button or switch associated with a block | the train reserving the context block | the block, which the push button or switch is associated with | - | current schedule of the context train |
| Route | the train, which is currently reserving the route | current block of the context train | the route itself | current schedule of the context train |
| Schedule | the train executing the schedule (only after start) | the current block of the context train | - | the schedule itself |
| Turnout | the train, which is currently reserving the context route | - | the currently active route across the turnout | current schedule of the context train |
| Turntable | the train reserving the block on the bridge | the block, which is associated with the bridge | the currently active route, across the turntable | current schedule of the context train |
| Macro | context train of the caller | context block of the caller | context route of the caller | context schedule of the caller |

**Table 1: Context Objects**

The context vehicle exists only in conjunction with the execution of engine functions. It is the vehicle, which executes these functions. Note, that in many cases the context engine and the context train may be identical.

There are furthermore context stations and context boosters. These are just the stations (see section 15.7, “Stations”) and boosters (see section 15.8, “Booster”), respectively, which the context block (see above) is located in.

### Operations for Access to Variables

The access to values to variables is performed by the control flow operation **Access to Variable** (see page 45).

Among others the following types of access operations are available:

- **= (Assignment):**
  This operation is available for all types of variables.
  It assigns a fixed value or the content of another variable to a variable.
  For number and time variables it is also possible to assign the result of a calculation based on a formula.
  It is furthermore possible to assign system wide status values, the status of the system clock or the fast clock, or certain status values of context objects (such as the speed of the context train or the number of trains located in the area of the context booster) to a variable. More details can be found in the **Help** menu of **TrainController™**.
- **+ (Add):**
  This operation is available for number, text and time variables.
  It increments the value stored in the variable by a fixed value or the content of another variable. In case of text variables the value is appended to the text already stored in the variable.
  For number and time variables it is also possible to increment the stored value by the result of a calculation based on a formula.
- **- (Subtract):**
  This operation is available for number and time variables.
  It decrements the value stored in the variable by a fixed value, the content of another variable or the result of a calculation based on a formula.

- **Random:**
  This operation is available for number variables.
  It assigns a random number between 0 and a specified value to a variable. This value can be a fixed number, stored in another variable or the result of a calculation based on a formula.
  If, for example, the specified value is 12, then a random number between 0 and 11 will be assigned to the variable. With a subsequent **+ (Add)** operation it is also possible to create a random value in a range with a lower boundary, which differs from 0.
- **! (Operation):**
  This operation is only available for object variables.
  It performs a specific operation (e.g. set signal to green) with the object currently stored in the variable.
- **= (Query):**
  This operation can only be applied to object variables.
  It is available in various forms, among others **= (Query Train)**, **= (Query Block)**, **= (Query Schedule)**.
  The operation can be used to retrieve an object currently linked to a given object, e.g. the train currently reserving a given block, or the current schedule of this train, or another object. This object is then stored in the variable.
  To turn on the lights of that train, which is currently reserving block “Southtown 3”, the following operations are executed:
  First the operation **= (Query Train)** with specification of block "Südstadt 3" is executed with an object variable. Thus, the train which is currently reserving the block "Südstadt 3" is stored in this variable. Then the operation **! (Operation)** with specification of the "light on" train operation is executed with this variable. This turns the light on of that train, which is currently stored in the variable, i.e. the train, which is currently located in "Südstadt 3".
- **@ (Reference):**
  This operation is available for all types of variables.
  It assigns a reference to another variable to a variable.
  This operation should be used only by very experienced users. If a reference to variable X is assigned to variable Y, then variable Y acts as a kind of “pointer” to variable X. Whenever variable Y is accessed or evaluated, then actually variable X is accessed or evaluated. If the reference to X is stored in Y and Y is incremented by 5 with the **+ (Add)** operation, for example, then actually the value stored in X is incremented.

### Use of Variables in Operations

Many operations support the use of formulas instead of fixed values. One example is the **Delay** control flow operation. Instead of specifying a fixed delay it is also possible to specify a formula. During execution of this operation the value, which results from the calculation of the formula, determines the time of the delay.

Such formula consists of a text which contains numbers, arithmetic operators **+**, **-**, *****, and **/** or brackets like an ordinary mathematical formula. The formula can also contain variable wildcards (see below).

### Evaluation of Variables in Triggers and Conditions

The content of variables can be evaluated in triggers or conditions.

- **Numeric, Time and Text Variables:**
  The value stored in numeric, time or text variables can be evaluated with the logic operations **= (equal)**, **<> (not equal)**, **>= (greater or equal)**, **> (greater than)**, **<= (less or equal)**, **< (less than)** with fixed values or the value currently stored in another variable.
  For numeric and time variables it is also possible to compare with the result of a calculation based on a formula.
  For text values the comparison is not case sensitive.
- **Object Variables:**
  The object referred to by object variables can be tested, whether it is equal or not equal to a fixed object or to the object referred to by another variable.
  It is also possible to evaluate the states of the stored object.
  For object variables with the object type train it is furthermore possible to check, whether the train currently stored in the variable matches a certain train description.

### Variable Wildcards

In many options of the program with text as value it is also possible to insert the content of variables into the text by means of wildcards. Whenever this text is used (e.g. during execution of the system operation **Message**), the wildcard is replaced by the current content of the variable.

The content of number and time variables will be accordingly converted to text. The content of object variables will be converted to empty text.

To enter a variable wildcard into a text field type the two characters **%V**. If the particular text field supports the variable wildcard, these two characters are immediately extended to **%V[?]**. From now on this wildcard can only be edited as a whole, like one single character. Double clicking to this wildcard opens a dialog box, in which the variable to be used for the wildcard can be selected. After confirming the selection the '?' in the wildcard is replaced by the name of the variable.

If the text fragment **%V** is not automatically extended to **%V[?]**, then the corresponding text option does not support variable wildcards.

**Example:**

Assume, that the following message is to be displayed during operation with the **Message** system operation:
"Currently active schedules X – Currently activated routes Y", where X and Y will be replaced by the corresponding numbers.
Assume, that the number of active schedules is counted in the number variable "Count-S"; and the number of active routes is counted in the variable "Count-R".
Create a **Message** system operation and type in the following text:
"Currently active schedules %V – Currently activated routes %V"
This text will be automatically extended to
"Currently active schedules %V[?] – Currently activated routes %V[?]"
Double click the left instance of **%V[?]** and select the variable "Count-S". Double click the right instance of **%V[?]** and select the variable "Count-R".
The text now changes to
"Currently active schedules %V[Count-S] – Currently activated routes %V[Count-R]".
Assume that the value stored in "Count-S" is 5 and the value stored in "Count-R" is 11, when the **Message** system operation is executed. Then the following text will be displayed:
"Currently active schedules 5 – Currently activated routes 11".

Variable wildcards are supported by the following features (list not exhaustive):

- System operations with text as value (such as **Message**, **Sound File**, etc.).
- Content of text labels in the switchboard.
- Formulae of ramps and distances of markers in blocks (see page 25).
- Assignments to variables or evaluation of variables with text or a formula as operand.

### Scope

The scope of a variable describes, in which context a value of a variable can be evaluated. Variables are able to store more than one value. The scope of a variable specifies, which of these values is accessed in the current situation.

There are the following scopes:

- **Global:**
  Variables with global scope can store one single value. This value can be accessed at all locations of the program, where variables can be used. If the value of a global variable is changed at one location of the program, this change will become effective at all other locations, too, where this variable is accessed.
- **Private:**
  Variables with private scope can store an individual ("private") value for each object in the program. This value can be set by the operations performed by this object and evaluated in conditions or triggers of the same object.
  The same private variable can be used by different objects. The private value stored in this variable for a specific object, however, is not visible for another object.
  Assume, for example three push buttons. The first push button may be pushed once, the second twice and the third three times. To achieve this, the same variable with private scope can be incremented (by one) in the operations of each push button. This will store a private value for each push button in the variable. In the conditions of each push button the variable can be compared with the individual limit (1, 2, or 3, respectively) of each button. The actual value used for this comparison is the private value of this push button stored in the variable.
  Macros are an exception. For macros no private value is stored. If a variable with private scope is used in the operations or conditions of a macro, then the private value of that object is used, which has called the macro. This allows, for example, to increment the private counters of the three push buttons mentioned above in a macro called by these push buttons.
- **Train:**
  Variables with train scope can store an individual value for each train in the program. The value stored in this variable for a particular train can be used in each object, which is currently associated with this train.
  Examples of such objects are blocks or routes reserved by this train; indicators or markers contained in blocks reserved by a train; turnouts contained in routes reserved by this train; etc.
  If, for example, a variable with train scope is contained in the condition of a marker, then the value stored in this variable of that train is used, which is currently reserving the block, where the marker is contained in.

- **Block:**
  Variables with block scope can store an individual value for each block in the program. The value stored in this variable for a particular block can be used in each object, which is currently associated with this block. Examples of such objects are the train located in this block; indicators or markers contained in this block; push buttons linked to this block etc.
- **Schedule:**
  Variables with schedule scope can store an individual value for each schedule in the program. The value stored in this variable for a particular schedule can be used in each object, which is currently associated with this schedule. Examples of such objects are the trains currently executing this schedule; blocks reserved by a train, which is currently executing this schedule, as well as all indicators and markers therein; routes reserved by a train, which is currently executing this schedule, as well as all turnouts contained therein; etc.
- **Route:**
  Variables with route scope can store an individual value for each route in the program. The value stored in this variable for a particular route can be used in each object, which is currently associated with this route. Examples of such objects are the turnouts contained in this route.
- **Extended Accessory:**
  Variables with extended accessory scope can store an individual value for each symbol of an extended accessory in the program. The value stored in this variable for a particular extended accessory can be used in each control of the extended accessory.
- **Local:**
  Variables with local scope can store one single value. Local variables are created in the context of operation lists. Their value can only be accessed by operations in the same list of operations. Local variables are not managed via the Explorer window. A local variable is automatically deleted, when there is no operation anymore, which refers to it.

**TrainController™ 9 Gold** tries to resolve references to variable values as far as possible. If, for example, a variable with schedule scope is used in the operations of a macro, which is called by a marker in a block, then the software tries to find out, which train is reserving this block and which schedule is controlling the train. If there is such schedule, then the value stored in this variable for this schedule is used. If there is no such schedule, then an empty value (0, empty string, etc.) is used instead.

# 15 The Visual Dispatcher II

## 15.5 Overview of all Schedule Rules

### Schedule Start

- **Start the oldest train:**
  This rule is effective if it is possible to execute the schedule with more than one train. Specify the number of oldest trains, where the train to be started will be selected from. The oldest trains are those, which are resting at their current position for the longest time.
  If **0** is specified for this rule, then this rule is disabled and the train to be started is selected by random from the available trains. However, with a schedule, which starts and end in a hidden yard and which is permanently repeated, this setting could cause a train to be started again very soon after terminating a run, or not to be started for a very long time.
  If **1** is specified for this rule, then actually the oldest train is always started. However, with a schedule, which starts and end in a hidden yard and which is permanently repeated, this setting causes the trains to be started in predictable order, which may become boring after a while.
  Specifying an appropriate value greater than **1** can provide a good balance between coincidence and predictability.
- **Do not reset resting time:**
  This rule is useful in conjunction with the use of the rule **Start the oldest train** in this or other schedules. The oldest trains are those, which are resting at their current position for the longest time. This resting time is usually reset whenever a schedule with the concerning train is terminated. By setting this rule the resting time is not reset, when this schedule is terminated. This is useful for schedules, which control the moving up of trains in a hidden yard, for example, or similar local maneuvers. The train which entered a certain yard first, for example, remains the oldest train in this yard, even if it is moved within this yard by a schedule with this rule set.

- **Control car pulls:**
  This rule is only effective together with the rule **Train may only start in forward direction**. Train sets are not only started, if this results in a pulled train, but also, if this results in a train with a control car at its head.
  In other words: If both rules are activated, then train sets are only started, if this results in a train with an engine or a control car at its head.
  If only the rule **Train may only start in forward direction** is activated, then train sets are only started, if this results in a train with an engine at its head.

### Reservation of Blocks and Routes

- **Maximum Detour:**
  This rule specifies the maximum number of blocks, that are accepted as detour on the path to the destination block. This rule is useful in all cases, where the train is allowed to select a path, which is longer than the shortest path (in terms of numbers of blocks), when the shortest path is locked by an obstacle. The previously existing rule **Ignore distances** could be used in such cases, too. But that rule often leaded to inacceptable long detours and required also much CPU capacity. The rule **Maximum Detour** is a much more accurate option for intervention.
- **Use start block as destination block:**
  This rule forces a train controlled by a circular schedule with several start and destination blocks, which are identical to the start blocks, to use that block as destination of the schedule, where the train started from. This rule is useful for example for schedules, which start and end in a hidden yard. In this way it is possible to define an implicit “pseudo home block” in this yard for each train starting and returning there without the need to specify this block explicitly as home for each train. By using this rule the pseudo home block is automatically (implicitly) assigned to the train, when the train is located in this block or when trains swap their (home) position in the yard by hand.

- **Include turntables:**
  This rule applies only to AutoTrain runs. If this rule is set, then turntables are included into the path search, even if they are not directly connected to a start or destination block of the AutoTrain run.
  Deactivating this rule prevents turntables from being included into paths from a distant start block to a distant destination. Turntables are limited to be used in paths, which start or end in an adjacent destination block (e.g. in the attached roundhouse).

### Train Sets

This category includes rules that specify, how train sets are treated.

- **Line up in destination block:**
  This rule allows trains to be lined up with other trains already waiting in the destination block. In this way several trains (usually locomotives) can be parked in a staging track without being connected with each other in a continuous train set. The value entered for this rule specifies the gap between the already waiting and the incoming vehicles.

### Signals

With the following rules it is possible to affect the calculation of the internal block signals:

- **Request yellow:**
  This rule requests the yellow signal for all blocks and routes in the schedule.

- **Request yellow for local schedule:**
  Same as before, but only if the schedule is a local schedule (see page 65, “Local Schedules”).

- **Reject yellow:**
  All requests for the yellow signal issued by blocks, routes or turnouts in the schedule are rejected. The internal block signal will be always set to green, when the train may proceed.

- **Reject yellow for local schedule:**
  Same as before, but only if the schedule is a local schedule (see page 65, “Local Schedules”).

- **Replace green by white:**
  The internal block signal is set to white, whenever the calculation of the signal results in a green aspect. This rule provides an optional alternate signal aspect for green – e.g. for alternate signaling of shunting moves.

- **Replace green by white for local schedule:**
  Same as before, but only if the schedule is a local schedule (see page 65, “Local Schedules”).

- **Replace yellow by white:**
  The internal block signal is set to white, whenever the calculation of the signal results in a yellow aspect. This rule provides an optional alternate signal aspect for yellow – e.g. for alternate signaling of shunting moves.

- **Replace yellow by white for local schedule:**
  Same as before, but only if the schedule is a local schedule (see page 65, “Local Schedules”).

#### Spontaneous Run

The following rules for spontaneous runs are new in **TrainController™ 9**:

- **Local:**
  This rule causes the associated train to perform a local spontaneous run (see page 65, “Local Schedules”).

#### Misc

- **Adaptive Braking:**
  This rule activates the Adaptive Braking Procedure (ABP) for the schedule (see also page 76). ABP improves the exactness of calculated stop locations. ABP causes the train to slow down with individual speed and ramp values, when it has to stop in a block. These values are adapted to the individual characteristics of the train to stop the train as exactly as possible at the desired stop location.

## 15.7 Stations

#### General

Stations can be used as an optional feature to control the traffic on your layout and the aspects of calculated signals for shunting moves.

Stations can control the traffic flow by preventing trains from the execution or termination of schedules in a station in certain situations. Furthermore stations can impair the calculation of internal block signal aspects for schedules, which are limited to the area of a single station (shunting moves).

Stations are established by assigning blocks and routes to them. This is done in a similar way like assigning blocks and routes to schedules. Each block or route can only be assigned to at most one station. It is not possible to assign a block or route to two stations or more. A route, however, can connect two blocks, which are located in different stations.

Stations are displayed in the station view of the Dispatcher Window (see section 5.18).

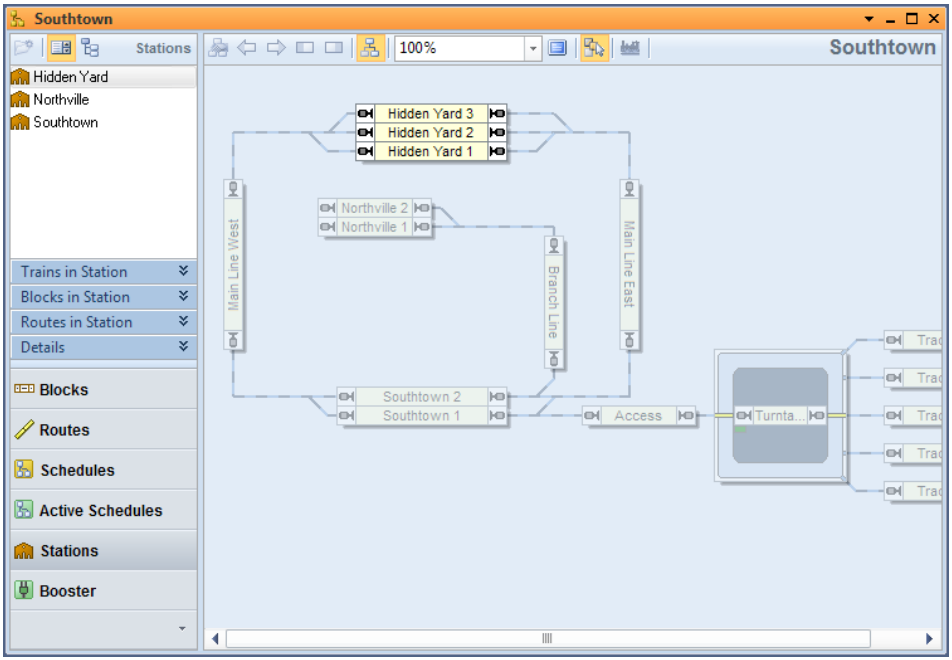



**Diagram 18: Station View in the Dispatcher Window**

Diagram 18 shows a layout with three stations. The station “Hidden Yard” is currently selected and the station diagram of “Hidden Yard” is currently visible in the right part of the dispatcher window. The blocks “Hidden Yard 1”, “Hidden Yard 2” and “Hidden Yard 3” are located in station “Hidden Yard”.

### Minimum and Maximum Number of Trains

For each station it is possible to specify a minimum and maximum number of trains. This allows to control the start and termination of schedules in a station.

The minimum and maximum number of trains are related to inactive trains, i.e. to trains not under control of a schedule.

If the number of inactive trains currently located in the station is smaller or equal than the minimum number of trains, then no schedule can be started with one of these trains. If the number of inactive trains currently located in the station is greater or equal than the maximum number of trains, then no schedule can be terminated in this station.

Note, that passing of trains through a station is not affected by these numbers. That means: a train may always pass a station – regardless of the values of these numbers.

### Conditions

For each station two conditions can be specified, one for the start of schedules in this station and one for the termination of schedules in this station.

If the condition to start schedules in this station is not true, then no schedule can be executed with a block located in this station as start block.

If the condition to terminate schedules in this station is not true, then no schedule can be executed with a block located in this station as destination block.

Note, that passing of trains through a station is not affected by these conditions. That means: a train may always pass a station – regardless whether these conditions are true or not.

### Stations, Trains and Schedules

It is also possible to assign trains and schedules to a station.

If trains are assigned to a station, then only these trains may start or terminate a schedule in this station.

If schedules are assigned to a station, then only these schedules may be started or terminated in this station.

### Local Schedules

A local schedule is a schedule, which contains only blocks located in the same station as well as blocks not located in a station, which are directly connected to blocks in this sta-

tion by routes. In other words: a local schedule contains only blocks located in the same station or directly adjacent to the same station.

A schedule with blocks located in different stations is no local schedule. This is a schedule, which connects different stations.

A schedule with routes, which connect two blocks not contained in a station, is no local schedule. This is a schedule with routes on a main track.

Trains controlled by a local schedule can only move within one station or into blocks adjacent to this station. Typical examples are switching moves.

Regular schedules are made implicitly local or non-local by their schedule diagrams, i.e. by the blocks and routes contained in the schedule and the stations, where these blocks and routes are located in.

A schedule established by AutoTrain (see section 5.8) can be forced to be local with a specific option. Such AutoTrain run will only contain those blocks and routes, which result in a local schedule according to the above description. This option is useful to force an AutoTrain run to be limited to a certain station.

Spontaneous runs can be forced to be local, too, by applying a specific rule for spontaneous runs to selected trains (see section 15.5, “Overview of all Schedule Rules”).

#### Local Schedules and calculated Signals

Local schedules are often used to perform shunting moves. In the real world signals are often set differently for shunting moves on one side and trips from one station to another on the other side. For these reasons **TrainController™ Gold** provides a set of schedule rules, with which the internally calculated block signals can be specifically affected for local schedules (see section 15.5, “Overview of all Schedule Rules”).

## 15.8 Booster

#### General

Boosters can be used as an optional feature to establish a booster management for your layout. This booster management allows – among others – to automatically stop trains,

to prevent trains from starting, or to lock and release certain areas of the layout depending on the load put on the track or status of physical boosters.

Boosters are established by assigning blocks and routes to them. This is done in a similar way like assigning blocks and routes to schedules. Each block or route can only be assigned to at most one booster. It is not possible to assign a block or route to two boosters or more at the same time. A route, however, can connect two blocks, which are assigned to different boosters.

Boosters are displayed in the booster view of the Dispatcher Window (see section 5.18).

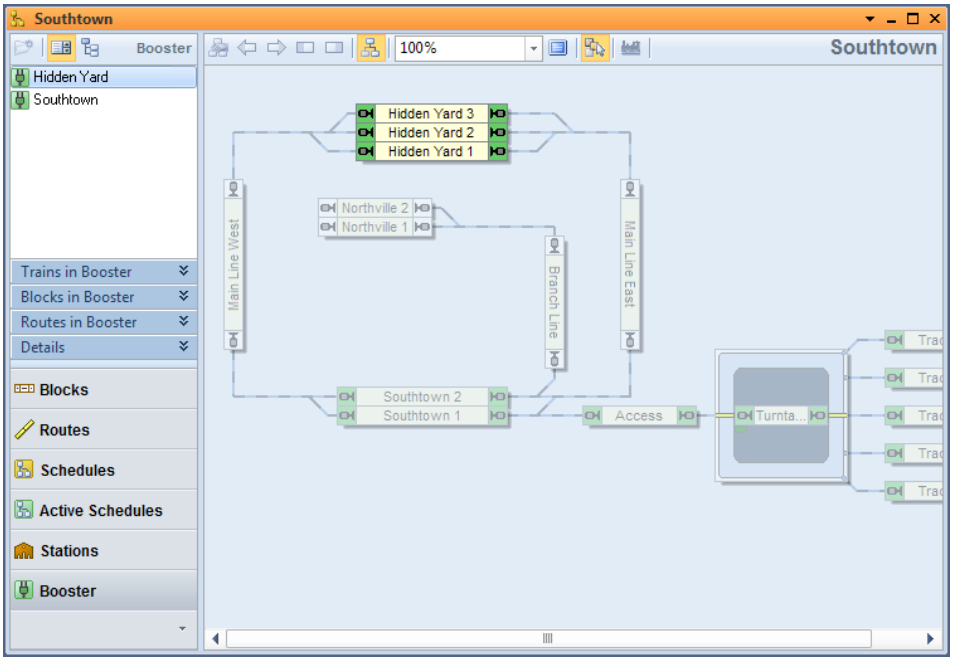



**Diagram 19: Booster View in the Dispatcher Window**

Diagram 19 shows a layout with two boosters. The booster “Hidden Yard” is currently selected and the block diagram of “Hidden Yard” is currently visible in the right part of the dispatcher window. The blocks “Hidden Yard 1”, “Hidden Yard 2” and “Hidden Yard 3” are assigned to booster “Hidden Yard”.

Boosters can be used to control the traffic on the model railroad as a function of the load currently put on the tracks or the number of trains currently busy in certain areas of the layout. In many cases the booster management is established to prevent physical boost-

ers from being overloaded. For this purpose booster objects in **TrainController™** are usually associated with physical boosters connected to the track. It is also possible, however, to establish a booster management for traffic control based on virtual boosters without physical counterpart.

### States of a Booster

Each booster provides the following states:

| Symbol | State | Meaning |
|---|---|---|
|  | On | The booster is turned on. No problems exist in the area of the booster. This is the normal state. |
|  | Off | The booster is turned off. The traffic is currently stopped in the area of the booster. |
|  | Warning | The booster is turned on. Certain action may be required to avoid subsequent problems. |
|  | Error | A severe problem occurred in the area of the booster (e.g. short circuit or overload of the associated physical booster, too many active trains, etc.) |
|  | Disabled | All features of the booster object are disabled. The booster object does not impair the traffic in the area associated with it. |

**Table 2: States of a Booster**

The colors associated with each booster state are also displayed in the exit boxes of the blocks in the block diagram of this booster (see Diagram 19).

### Rules

In **TrainController™ Gold**, the effect of each booster can be customized to individual needs with a variety of rules.

The rules are divided into the following categories:

**Off State:**

This category includes rules that specify the behavior of the booster object, when it is turned off:

- **Freeze Trains:**
  All trains located in blocks associated with the booster object are frozen, when the booster is turned off. Frozen trains are stopped. Schedules, which control these trains, remain active.

- **Stop Trains and Schedules:**
  All trains located in blocks associated with the booster object are stopped, when the booster is turned off. Schedules, which control these trains, are terminated.

- **Lock Exits:**
  The exits of all blocks associated with the booster object are locked, when the booster is turned off. This causes all trains in the booster area to stop, when they reach the next stop marker in their current block (unless the physical booster associated with the booster object cuts the power). Trains, that are currently stopped cannot be put in motion by schedules.

- **Lock Entries:**
  All entries of blocks located at the boundary of the booster area are locked, when the booster is turned off. This prevents trains currently not located in the booster area to enter the booster area.

**Error State:**

This category includes rules that specify the behavior of the booster object, when it is in the error state:

- **Freeze Trains:**
  All trains located in blocks associated with the booster object are frozen, when the booster changes to the error state. Frozen trains are stopped. Schedules, which control these trains, remain active.

- **Stop Trains and Schedules:**
  All trains located in blocks associated with the booster object are stopped, when the booster changes to the error state. Schedules, which control these trains, are terminated.

- **Lock Exits:**
  The exits of all blocks associated with the booster object are locked, when the booster changes to the error state. This causes all trains in the booster area to stop, when they reach the next stop marker in their current block (unless the physical booster associated with the booster object cuts the power). Trains, that are currently stopped cannot be put in motion by schedules.

- **Lock Entries:**
  All entries of blocks located at the boundary of the booster area are locked, when the booster changes to the error state. This prevents trains currently not located in the booster area to enter the booster area.

- **Off:**
  The booster is turned off, when it changes to the error state.

**Warning State:**

This category includes rules that specify the behavior of the booster object, when it is in the warning state:

- **Keep Frozen Trains:**
  Trains frozen in the error state are not released. This option is useful to prevent the booster from changing back to the error state right after changing from the error to the warning state.

- **Lock Exits:**
  The exits of all blocks associated with the booster object are locked, when the booster changes to the warning state. This causes all trains in the booster area to stop, when they reach the next stop marker in their current block. Trains, that are currently stopped cannot be put in motion by schedules.

- **Lock Entries:**
  All entries of blocks located at the boundary of the booster area are locked, when the booster changes to the warning state. This prevents trains currently not located in the booster area to enter the booster area.

- **Do not execute Schedules:**
  Do not execute any additional schedules with trains located in the booster area, while the booster is in warning state.

**Threshold Values:**

This category includes threshold values, which causes the booster to change to the error or warning state, when the actual value reaches the threshold value:

- **Running Trains:**
  The warning or error state, respectively, is turned on, when the number of trains running in the booster area at non-zero speed, is greater or equal than the specified threshold value. This option is useful to prevent the associated physical booster from being overloaded, when no other information from the booster (such as current or temperature) is available.

- **Current:**
  The warning or error state, respectively, is turned on, when the current of the track output of the associated physical booster is greater or equal the specified threshold value.

- **Temperature:**
  The warning or error state, respectively, is turned on, when the temperature of the associated physical booster is greater or equal the specified threshold value.

### Physical Connections of a Booster

In order to implement a booster management in conjunction with physical boosters, the following addresses can be specified for each booster object:

- Booster address: this address allows to associate the booster object in **TrainController™** directly with a physical booster connected to the layout. This provides the possibility to turn on or off the physical booster or to determine values for the current or temperature of the physical booster.
  The availability of certain features depends on the type of the physical booster. The following types of physical boosters are currently supported:

  - The internal booster of the Maerklin CS2 / CS3
  - The internal booster of the Roco Z21
  - The internal booster of the Uhlenbrock Intellibox II
  - Other boosters from Uhlenbrock such as the Power 4

- Turnout commands: for the on and the off state it is possible to specify an individual turnout command (turnout address and state). This command is sent to the specified turnout address, when the booster object is turned on or off. This option is useful for those physical boosters, which provide the possibility to turn the track power on or off from the distance via a turnout command.

- Feedback addresses and state: For each state of a booster object (on, off, error and warning) it is possible to specify an individual feedback address and state. This causes the booster object to change to the according state each time, when the specified feedback contact changes to the specified state. This option is useful for those physical boosters, which report their status via regular feedback events.

### Trigger

For each state of a booster object (on, off, error and warning) it is possible to specify an individual trigger. This causes the booster object to change to the according state each time, when the trigger specified for this state becomes true.

This feature is useful for those physical boosters, which report their state with more complex information than plain feedback events.

This feature can also be used to implement a virtual booster management without physical boosters (see below).

### Virtual Booster Management

**TrainController™** provides the possibility to establish a booster management for traffic control without the existence of physical boosters, too, or where there is no possibility to specify a connection to a physical booster.

Such virtual booster management can be accomplished by specifying a trigger for each state of a booster object. In this way it is possible to automatically turn a booster object to error state and to stop the traffic in a certain area of the layout, for example, when a certain logical condition in **TrainController™** becomes true.

The option to change the booster to warning or error state, when the number of active trains in the booster area reaches or exceeds a certain threshold value, and to stop the traffic in this area subsequently, is another way to control the traffic on your model railroad layout, which can be used without the existence of physical boosters or without a connection to them.

### Boosters and other Objects

**E**ach state of a booster object can be evaluated in the trigger or condition of other objects. Furthermore booster objects can perform operations, when they change their state. These features provide virtually unlimited possibilities to integrate booster objects into the automatic control of your model railroad layout and to accomplish a smart booster management or traffic control.

# Appendix

## Migrating Existing Data Files from TrainController™ 8

Project files generated with **TrainController™ 8** are automatically converted into the format of the new version when opened in **TrainController™ 9**. The following things should be noted, however.

### Custom Block Diagrams

Manually created custom block diagrams are deprecated. Project files with custom block diagrams will be loaded ok, however.

It is possible to continue using custom block diagrams. It is also possible to edit and extend these diagrams. The according menu commands are still available (only in the classic interface, though – see page 16, “User Interface: Ribbon vs. Menus and Tool Bars”). But these commands are not documented anymore.

It is possible to turn manually created block diagrams into calculated block diagrams. And it is recommended to do so. But it is not possible anymore to create new custom block diagrams.

**TrainController™ 8 Silver**, in particular, provided the possibility to create a custom block diagram for the entire model railroad layout, even if this layout was controlled via more than one switchboard window. This has been superseded in **TrainController™ 9 Silver** by the creation of a calculated block diagram for each individual switchboard (like in **TrainController™ Gold**) and the possibility to turn an existing custom block diagram into a calculated block diagram for one of these switchboards. We encourage you to make use of these new possibilities.

### Train Objects and Multiple Units in TrainController™ 8 Silver

In **TrainController™ 8 Silver** multiple units can be arranged only, when edit mode is turned on. For this purpose so called train objects are created.

**TrainController™ 9 Silver** provides more flexible features to arrange and release multiple units at any time during operation. Train objects created in previous versions of

**TrainController™** remain unaffected, when a data file created with such version is being loaded. Thus existing trains operate as before. It is not possible, however, to create new train objects in **TrainController™ 9 Silver**. This feature is not needed here anymore, because the possibility to arrange train sets is more flexible now. You should consider to delete your existing train objects yourself as soon as possible.

### Waiting Times in Blocks of Schedules

Waiting Times in blocks of schedules are specified in **TrainController™ 9** as real-time data. Existing data, which was scaled by the speed of the fast clock in previous versions, is adapted accordingly.

As long as the speed of the fast clock is not changed, the waiting times work just as before. However, changes in the speed of the fast clock have no effect on the waiting times anymore.

### High Precision Scaling

High Precision Scaling (HPS) in **TrainController™ 9** makes the calculation of the current scale speed of engines more exact. This affects in particular the accuracy of calculated distances and ramps.

HPS is automatically turned on for engines newly created in **TrainController™ 9**.

To ensure compatibility to **TrainController™ 8** all engines imported from **TrainController™ 8** are operated by default without HPS.

HPS can be turned on for existing engines created in **TrainController™ 8**, too, if desired. HPS is furthermore automatically turned on for these engines, if their speed profile is measured again.

### Extended Profile Generation

Extended Profile Generation (EPG) causes the measured speed profile of each engine to match the slow speed of each engine more exactly. This causes trains to stop more exactly in situations, where they travel a certain distance at slow speed; for example, when they enter the bridge of a turntable or when they perform a short distance move (see page 45).

EPG is automatically applied, when the speed profile of an engine is being measured.

### Adaptive Braking Procedure

Adaptive Braking Procedure (ABP) (see page 63) is automatically activated for all schedules imported from previous versions of **TrainController™** and for all new schedules.

ABP improves the exactness of calculated stop locations. ABP causes the train to slow down with individual speed and ramp values, when it has to stop in a block. These values are adapted to the individual characteristics of the train to stop the train as exactly as possible at the desired stop location.

However, ABP may cause each train to slow down in blocks prior to the block, where the train has to stop. If this effect is not desired, then Adaptive Braking can be deactivated via the **Rules** tab of each schedule's properties.

### Variable Stop Locations in a Block – Stopping for Coupling

In **TrainController™ 8** distances in formulas of markers in a block without unit were interpreted as centimeters. And distances with the unit "in" were interpreted as inches. The **Metric Units** command in the **View** menu didn't have any impact here.

In **TrainController™ 9** distances in formulas of markers in a block are always interpreted as cm or inches according to the setting of the **Metric Units** command in the **View** menu.

The values in formulas created with version 8 are automatically converted accordingly with the following exception:

If the **Metric Units** command is turned off and a formula contains cm values (i.e. values not followed by the unit "in"), then these values are interpreted in **TrainController™ 9** as inches (according to the setting of the **Metric Units** command). Affected formulas cannot be converted automatically to the new format and must be corrected by hand. This issue only applies to users, who do not use metric measurement units and who specified distances in formulas as cm values (by omitting the unit "in").

# New features of +SmartHand™ Mobile

## Train Set View

The train set view is only available for **TrainController™ Gold.** It displays a list of locomotives and cars. The current locomotive or the engines and cars in its current train set are displayed at the top of the list, followed by all other engines and cars.

The current train set can be arranged with the following commands:

 : Adds the selected vehicle to the current train set. The vehicle is appended to the tail of the train set.

 : Adds the current train set to the selected vehicle. The train is set appended to the vehicle.

 : Deletes the selected vehicle from the train set.

 : Moves the selected vehicle by one position to the head of the train set.

 : Moves the selected vehicle by one position to the tail of the train set.

 : Reverses the orientation of the selected vehicle in the train set.

 : Joins the train set.

 : **S**eparates the train set in front of the selected vehicle. If the vehicle is the first in the train set, then the separation is done behind this vehicle.

 : Moves the image of the selected (moving) vehicle to an adjacent block. This movement is performed according to the usual rules of train tracking. This means, that the speed and current direction of the vehicle as well as turnout positions are taken into account to determine this block. If other vehicles are already waiting in this block, then the waiting and the moving vehicles will form a new train set as soon as the moving vehicle is stopped. This option is useful, for example, if an engine is moved under control of the handheld device to an adjacent block with waiting vehicles. If there is only one occupancy sensor in this block, then the moving vehicle cannot generate an occupancy indication in the adjacent block, because this is already done by the vehicles already waiting there. For this reason usual train tracking will fail in such case. However, this option can be used in this case to inform the software, that the vehicle has moved to this block.

 : This command works in a similar manner like the previous command. It moves the image of moving engines or cars coming from an adjacent block to the block, where the selected vehicle is currently located.

# Index