

# **rcman – Modellbahnsteuerung mit textueller Oberfläche**

Dr. Peer Griebel

# Inhalt

<b><u>rcman – Manuelle Modellbahnsteuerung mit textueller Oberfläche</u></b> .....	<b>1/23</b>
<u>Copyright und Lizenz</u> .....	1/23
<b><u>Einführung</u></b> .....	<b>2/23</b>
rcman.....	2/23
<u>Exkurs: resh</u> .....	4/23
<u>... zurück zu resh</u> .....	4/23
<u>Systemanforderungen</u> .....	4/23
<b><u>Die Bedienung von rcman</u></b> .....	<b>6/23</b>
1. <u>Start von rcman</u> .....	6/23
2. <u>Der Bildschirmaufbau von rcman</u> .....	7/23
3. <u>Steuerung von Lokomotiven</u> .....	7/23
<u>Steuerung der Lokomotiven über die Tastatur</u> .....	8/23
<u>Steuerung der Lokomotiven mit der Maus</u> .....	9/23
4. <u>Steuerung von Weichen, Weichenstraßen und Signalen</u> .....	10/23
<u>Steuerung über die Tastatur</u> .....	11/23
5. <u>Steuerung einer Drehscheibe</u> .....	11/23
6. <u>Auswertung der Informationen des Infoports</u> .....	11/23
<b><u>Konfiguration rcman</u></b> .....	<b>13/23</b>
<u>Definition der Lokomotiven</u> .....	13/23
1. <u>Definition von Weichenstraßen</u> .....	13/23
2. <u>Definition der Ausgangskonfiguration der Weichenstraßen</u> .....	15/23
3. <u>Definition des Gleisplans</u> .....	16/23
4. <u>Definition von Gleisbesetzmeldern</u> .....	17/23
5. <u>Definition der Drehscheibe</u> .....	17/23
<b><u>Weiterführende Informationen</u></b> .....	<b>18/23</b>
<b><u>Anhang: Algorithmus Erkennung von Weichenstraßen</u></b> .....	<b>19/23</b>
<b><u>Appendix A – GNU General Public License</u></b> .....	<b>A-1</b>

# rcman – Manuelle Modellbahnsteuerung mit textueller Oberfläche

Diese Dokumentation beschreibt *rcman* Version 0.9 (2002–03–17).

Die neueste Version der software und der Dokumentation ist zu finden unter <http://www.griebel-net.de/peer/rcman.html>.

Dieses Dokument soll eine Dokumentation zu *rcman* bieten, die zumindest einem durchschnittlich versierten Anwender die Möglichkeit gibt, die *rcman* unter Linux-Systemen erfolgreich einzusetzen. (*rcman* ist allerdings grundsätzlich auf allen Systemen einsetzbar, auf denen das Python-System verfügbar ist.) Sollte die Dokumentation Lücken oder Fehler aufweisen oder liegen einfach Verbesserungsvorschläge für *rcman* und seine Dokumentation vor, so würde ich mich sehr darüber freuen, eine entsprechende [Nachricht](#) zu erhalten.

## Copyright und Lizenz

*rcman* und *rcsh* mit allen Komponenten und Dokumentation: ©2000–2002 by Dr. Peer Griebel ([peer.griebel@web.de](mailto:peer.griebel@web.de)).

Dieses Programm ist freie Software. Sie können es unter den Bedingungen der [GNU General Public License](#), wie von der [Free Software Foundation](#) herausgegeben, weitergeben und/oder modifizieren, entweder unter Version 2 der Lizenz oder (wenn Sie es wünschen) jeder späteren Version. Die Veröffentlichung dieses Programms erfolgt in der Hoffnung, dass es Ihnen von Nutzen sein wird, aber OHNE JEDE GEWÄHRLEISTUNG – sogar ohne die implizite Gewährleistung der MARKTREIFE oder der EIGNUNG FÜR EINEN BESTIMMTEN ZWECK. Details finden Sie in der GNU General Public License.

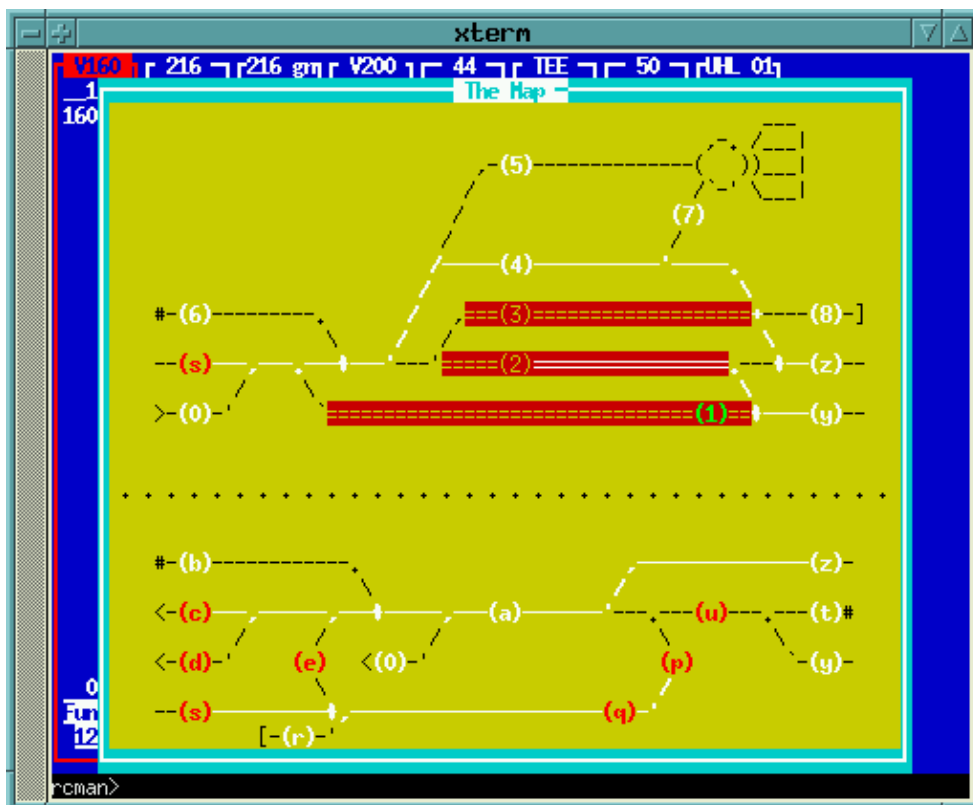
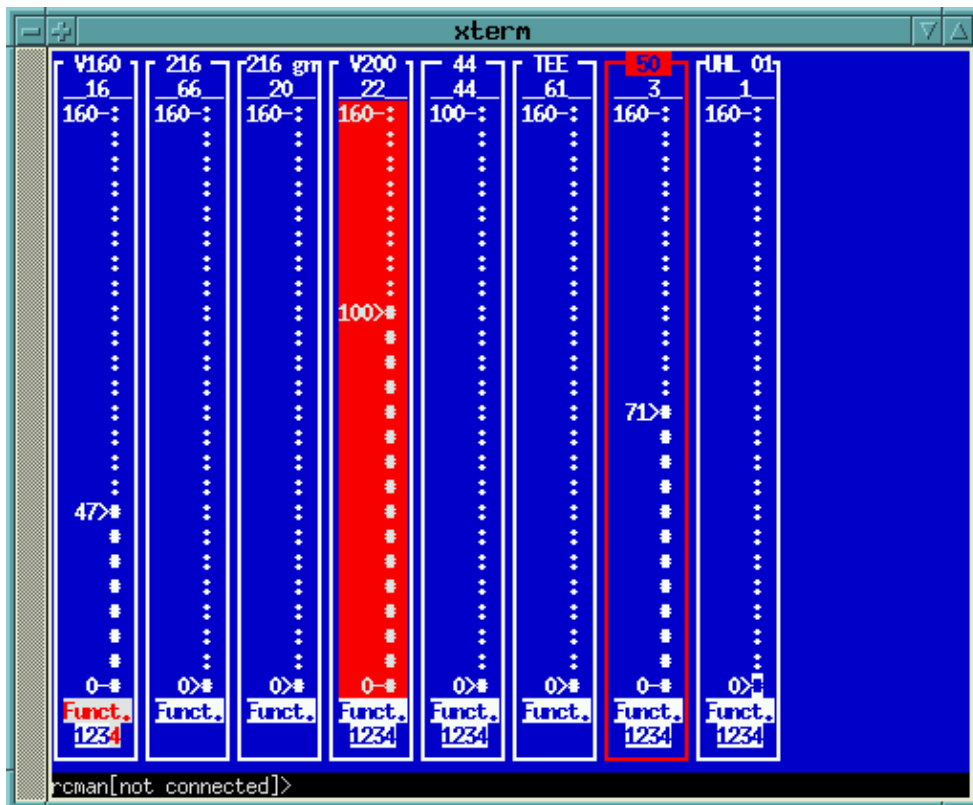
# Einführung

## rcman

*rcman* ist ein Programm zur Steuerung digitaler Modelleisenbahnen. *rcman* orientiert sich an *jman* aus dem [DDL-Projekt](#). *rcman* hat folgende Vorteile:

- ◆ Die Oberfläche ist nicht grafisch. Man muss daher nicht auf das X Window System zurückgreifen.
- ◆ Die Benutzerführung erfolgt durch eine textuelle Oberfläche, die intuitiv, praktisch und platzsparend ist.
- ◆ *rcman* baut auf *rcsh* auf und ist wie *rcsh* in Python implementiert.
- ◆ *rcman* ist insgesamt nicht speicherhungrig oder prozessorlastig. D.h. für *rcman* kann problemlos ein älterer Computer eingesetzt werden.
- ◆ Loks lassen sich sehr einfach mit Tastatur oder Maus steuern.
- ◆ Alle Lokdaten werden dargestellt.
- ◆ Informationen, die vom SRCP-Server über den Info-Port zu Lokomotiv-Einstellungen bekannt gemacht werden, werden ebenfalls dargestellt
- ◆ Auch Signale, Weichen bzw. Weichenstraßen lassen sich sehr komfortabel über die Tastatur steuern.
- ◆ Weichenstraßen werden textuell dargestellt
- ◆ Informationen, die vom SRCP-Server über den Info-Port zu Weichenstellungen bekannt gemacht werden, werden wieder in Weichenstraßen konvertiert und dargestellt
- ◆ Die Steuerung per IntelliMouse (mit dem Rädchen) ist sehr direkt: Mit den Maustasten lassen sich direkt die Funktionen steuern, mit dem Rädchen lässt sich die Geschwindigkeit sehr gut regeln. Ich setze eine kabellose Maus ein und kann daher an jeder Stelle die Anlage die Züge direkt steuern!
- ◆ Die Steuerung kann auch mit Hilfe von Infrarot-Fernbedienungen erfolgen.
- ◆ Alle Eingabegeräte (Tastatur, Maus, Infrarot-Fernbedienung) steuern eine eigene Lokomotive. Somit können mehrere Personen unabhängig voneinander steuernd eingreifen.

Die nachfolgenden beiden Bilder zeigen die Bildschirmdarstellungen für die Steuerung von Lokomotiven bzw. zur Steuerung von Weichenstraßen.



## Exkurs: rcsh ...

*rcman* baut auf *rcsh* auf und ist wie *rcsh* in Python implementiert. *rcsh* dient dazu, eine komfortable und vollständige Schnittstelle zu SRCP anzubieten. *rcsh* ist in der Programmiersprache Python geschrieben. Die Stichworte *rcsh*, Python, SRCP werden in der separaten [Dokumentation zu rcsh](#) mehr oder weniger ausführlich beschrieben. Da für das Verständnis von *rcman* zumindest rudimentäres Wissen über diese Stichworte vorhanden sein sollte, so empfiehlt sich die Lektüre dieser Dokumentation.

## ... zurück zu rcsh

Doch zurück zu *rcsh*. Oben wurden bereits die Möglichkeiten kurz vorgestellt, die *rcman* bietet. Nachfolgend sollen nun zunächst die Systemanforderungen beschrieben werden, bevor die Bedienung und die Konfiguration von *rcman* detailliert beschrieben werden.

## Systemanforderungen

*rcman* baut auf *rcsh* auf und damit auf einem SRCP-Server. Ich verwende den *erddcd* aus dem DDL-Projekt. Beim Server muss der Info-Port eingeschaltet sein. Ansonsten werden Steuerungen der Loks nicht dargestellt. Auch für die Darstellung von Weichenstellungen ist der Info-Port notwendig.

Im Gegensatz zu *rcsh* stellt *rcman* etwas höhere Anforderungen an die Python-Version. Konkret ist Python-Version 2.x notwendig, da nur in dieser Version das Paket *curses* leistungsfähig genug ist. Neben Python wird natürlich das Paket *rcsh* benötigt.

*rcman* benutzt das Paket *curses*, um den Bildschirm anzusteuern. Das heißt, es werden beliebige Bildschirmauflösungen unterstützt. Ich selbst lasse *rcman* auf einer der Linux-Consolen laufen. Normalerweise ist die Auflösung dort 80x25. Mit Hilfe des Programms [SVGATextMode](#) lässt sich die Auflösung jedoch auch verändern. Um möglichst viele Lokomotiven anzeigen zu können, bzw. um einen möglichst großen Gleisplan darstellen zu lassen, empfiehlt es sich, mit *SVGATextMode* die Auflösung zu erhöhen. *rcman* läuft natürlich auch unter X-Windows in einem *xterm* oder einer anderen Terminalemulation. Dort ist es viel einfacher, die Fenstergröße zu beeinflussen...

*curses* kennt verschiedene Arten, um Text darzustellen. In *rcman* wird das *Blink*-Attribut verwendet. Allerdings nicht, um blinkenden Text anzuzeigen, sondern um zusätzliche Farben verwenden zu können. Um diesen Modus zu aktivieren, muss beim Aufruf von *SVGATextMode* der Kommandozeilenschalter `colorBL` verwendet werden. Für *xterm* existiert eine ähnliche Einstellungsmöglichkeit über das Attribut `ColorBLMode`. Sollte dieses Feature hinderlich sein, so kann es auch komplett ausgeschaltet werden. Dazu dient die Variable `USE_BLINK` in der Datei `rcman.py`

Um mit einer Maus arbeiten zu können, muss eine Wheel-Mouse vorhanden sein, die über den PS/2 Anschluss mit dem Computer verbunden ist. Prinzipiell können auch andere Anschlüsse verwendet werden. Da ich aber nur über eine solche Maus verfüge, sind die anderen Anschlussarten derzeit nicht implementiert. (Bein Anfrage bin ich aber gerne bereit, *rcman* dahin gehend zu erweitern.)

Es ist zu beachten, dass *rcman* natürlich nicht gleichzeitig mit anderen Programmen laufen darf, die gleichzeitig die Maus verwenden wollen. So sollte beispielsweise *gpm* beendet werden mit dem Befehl:

## rcman – Modellbahnsteuerung mit textueller Oberfläche

```
/etc/rc.d.init/gpm stop
```

*rcman* wurde getestet unter den Betriebssystemen Linux und Windoof. Unter letztem Betriebssystem wurden die CygWin Utilities ([www.cygwin.com](http://www.cygwin.com)) verwendet, die Python 2.x beinhalten. Unter Windows werden weder eine Maus noch Infrarot-Fernbedienungen unterstützt.

# Die Bedienung von *rcman*

## 1. Start von *rcman*

*rcman* wird wie jedes andere Python-Programm mit einer Kommandozeile ähnlich folgender gestartet:

```
python -O rcman.py
```

Der Schalter `-O` dient dazu, den Python-Code zu optimieren. Im Fall von *rcman* bzw. *rcsh* schaltet dieser Schalter interne Zusicherungen (assertions) und Kontrollausgaben ab, so dass das Programm geringfügig schneller läuft. Näheres steht auch in der Dokumentation zu *rcsh*. Dort ist auch beschrieben, dass das Programm direkt in der Form

```
rcman.py
```

gestartet werden kann, wenn in der ersten Zeile der Datei der korrekte Pfad zum ausführbaren Python-Interpreter in folgender Form hinterlegt ist:

```
#!/usr/bin/python
```

*rcman* kennt folgende Kommandozeilenschalter:

Kommandozeilenschalter (kurz bzw. lang)		Erläuterung
<code>-h HOST</code>	<code>--host HOST</code>	Name des Rechners, auf dem der SRCP-Server läuft (Vorgabe: <code>localhost</code> ).
<code>-c PORT</code>	<code>--command-port PORT</code>	Kommando-Port, über den mit dem SRCP-Server kommuniziert wird. Der Pollport des Servers erhält automatisch die nächste Portnummer (+1), der Infoport die übernächste (+2).
<code>-s</code>	<code>--skip-splash</code>	Der Splash-Screen beim Start des Programms wird nicht angezeigt.
<code>-n</code>	<code>--no-init</code>	Die automatische Initialisierung der Weichenstraßen und des Fahrstroms ( <a href="#">Ausgangskonfiguration</a> ) wird unterbunden.
<code>-i</code>	<code>--info-refresh</code>	Schaltet die <a href="#">Analyse der Info-Meldungen</a> des SRCP-Servers an, um Weichenstraßen zu erkennen und darzustellen. (Die Darstellung von Informationen des SRCP-Servers über Lok-Informationen ist grundsätzlich eingeschaltet.)
<code>-m</code>	<code>--mouse</code>	Schaltet die Benutzung einer Maus (PS2-Format) an. (Die Maus wird unter Windows nicht unterstützt.)
	<code>--help</code>	Zeigt einen kurzen Hinweis auf die Aufruf-Syntax an.

## 2. Der Bildschirmaufbau von rcman

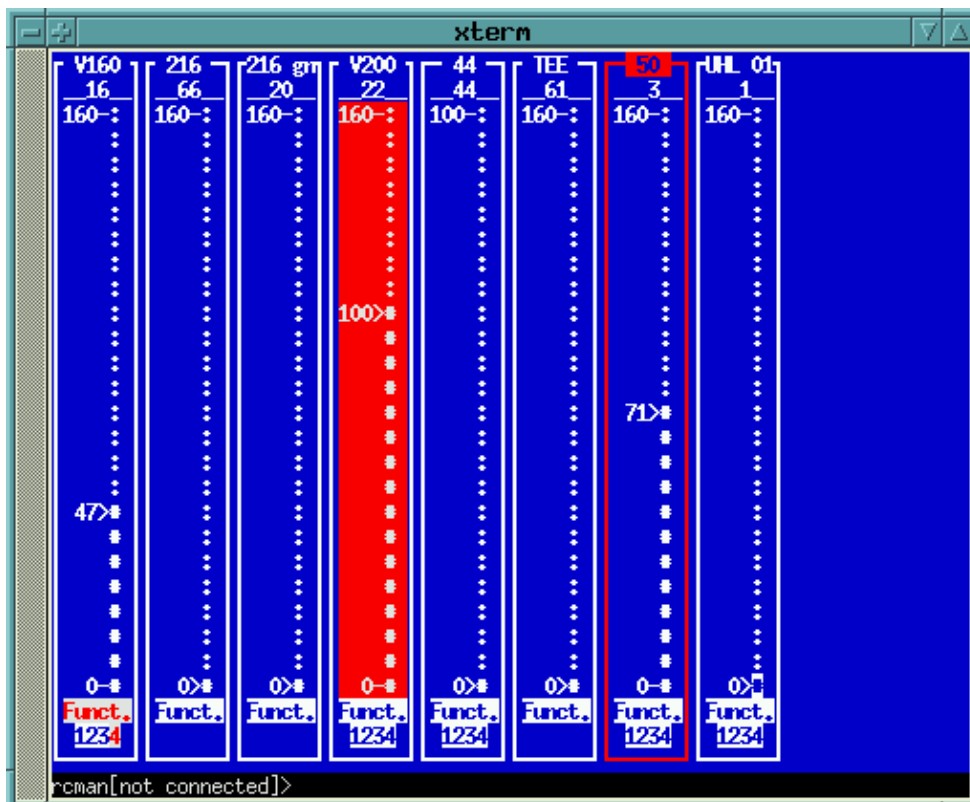
Der Bildschirm von *rcman* teilt sich in zwei Teile. Im oberen Teil werden Lokomotiv-Informationen bzw. Informationen über die Weichenstraßen dargestellt, wie es in den nächsten Abschnitten beschrieben wird.

Am unteren Rand des Bildschirms befindet sich eine Zeile, die kurze Statusinformationen darstellt und den Benutzer bei der Eingabe von Informationen behilflich ist, beispielsweise bei der Eingabe einer Weichenstraße.

Außerdem zeigt die Statuszeile an, ob der Fahrstrom eingeschaltet ist. Im ausgeschalteten Zustand wird das Wort *rcman* in rot dargestellt, ansonsten in weiß. Der Text [not connected] zeigt an, dass *rcman* keine Verbindung zum SRCP-Server aufbauen konnte. *rcman* ist trotzdem bedienbar – Loks, Weichen usw. werden in diesem Zustand natürlich nicht gesteuert.

## 3. Steuerung von Lokomotiven

Nach dem Start von *rcman* erhält man einen Bildschirm ähnlich folgendem:



Obige Grafik zeigt insgesamt 8 Lokomotiven, die zur Steuerung angeboten werden. Es können weitere Lokomotiven hinzugefügt werden, die rechts neben den vorhandenen Lokomotiven platziert werden. Sollte der Platz für weitere Lokomotiven nicht ausreichen, so werden die überzähligen Loks nicht dargestellt. Sie können bei Bedarf jedoch aktiviert werden und ersetzen eine der zum jeweiligen Zeitpunkt dargestellten Lokomotiven.

Jede Lokomotive wird über ihren Namen (1. Zeile, z.B. "V160") und ihre Digitaladresse (2. Zeile, z.B. 16) präsentiert. Darunter ist ein länglicher Bereich dargestellt, der die aktuelle Geschwindigkeit und Fahrtrichtung der Lokomotive repräsentiert. Die Fahrtrichtung "vorwärts" wird durch einen blauen Hintergrund, "rückwärts" durch einen roten Hintergrund

dargestellt. Schließlich werden die Einstellungen über die aktivierten Funktionen visualisiert. Je nach Dekodertyp erlaubt das Programm die Aktivierung der Basisfunktion (Function) und bis zu vier Zusatzfunktionen (F1 bis F4). Aktive Funktionen werden rot dargestellt, inaktive blau.

Die jeweils aktive, d.h. für die Steuerung vorgesehene Lokomotive wird durch einen farblich hervorgehobenen Rahmen dargestellt. Die mit einem roten Rahmen versehene Lokomotive wird über die Tastatur gesteuert. Sofern die Maussteuerung aktiviert wurde, wird die mit der Maus kontrollierte Lokomotive mit einem grünen Rahmen dargestellt. Die Steuerung der beiden Lokomotiven erfolgt unabhängig voneinander.

## Steuerung der Lokomotiven über die Tastatur

Die Lokomotiven können sehr einfach und intuitiv über die Tastatur angesteuert werden, um die Geschwindigkeit und Funktionen zu regeln. Die nachfolgende Tabelle listet die Funktionstasten und ihre Funktionalität auf, die zur Regelung der aktuell aktiven Lok dienen:

<b>Pfeil hoch</b>	Erhöht die Geschwindigkeit bis zu ihrem Maximum.
<b>Pfeil runter</b>	Reduziert die Geschwindigkeit bis auf 0.
<b>Leertaste</b>	Reduziert die Geschwindigkeit direkt auf 0. Es wird jedoch kein Nothalt gemacht. D.h. die Lokomotive reduziert ihre Geschwindigkeit entsprechend der am Dekoder eingestellten Verzögerung.
<b>Eingabe</b>	Schaltet die Fahrtrichtung der Lokomotive um. Gleichzeitig wird die Geschwindigkeit auf 0 reduziert. Diese Funktion kann auch dazu genutzt werden, um einen Nothalt der aktuellen Lokomotive durchzuführen.
<b>F10</b>	Schaltet die Grundfunktion der Lokomotive (Function) ein bzw. aus.
<b>F1</b> bis <b>F4</b>	Schaltet die Funktion 1 bis 4 der Lokomotive ein bzw. aus.

Weitere Funktionstasten dienen zur Steuerung des Gesamtsystems bzw. zur Auswahl von Lokomotiven:

<b>Pfeil links</b>	Wählt die nächste links liegende Lokomotive aus. Lokomotiven, die bereits von einem anderen Eingabegerät (Maus) verwaltet werden, werden dabei übersprungen.
<b>Pfeil rechts</b>	Wählt die nächste rechts liegende Lokomotive aus. Lokomotiven, die bereits von einem anderen Eingabegerät (Maus) verwaltet werden, werden dabei übersprungen.
<b>Bild hoch</b>	Ersetzt die aktuelle Lok durch die erste, derzeit wegen Platzmangel nicht dargestellte Lokomotive. Mit dieser Taste kann vorwärts durch die Liste der nicht dargestellten Lokomotiven geblättert werden.
<b>Bild runter</b>	Ersetzt die aktuelle Lok durch die letzte, derzeit wegen Platzmangel nicht dargestellte Lokomotive. Mit dieser Taste kann rückwärts durch die Liste der nicht dargestellten Lokomotiven geblättert werden.
<b>Backspace</b>	Schaltet den Fahrstrom ein bzw. aus. Dies entspricht einem Nothalt der kompletten Anlage.
<b>Tab</b>	Wechselt zum Bildschirm zur Darstellung von Weichenstraßen,

Signalen und Rückmeldern (und wieder zurück).

<b>?</b>	Zeigt einen kurzen Hilfebildschirm an.
<b>Q</b>	Beendet das Programm <i>rcman</i> .

## Steuerung der Lokomotiven mit der Maus

Wie erwähnt wird zur Steuerung von Lokomotiven eine Wheel–Mouse, also eine Maus mit einem Drehrad vorausgesetzt. Da die Eingabemöglichkeiten mit einer solchen Maus mit insgesamt drei Schaltknöpfen sehr beschränkt sind, werden die verschiedenen Funktionen auf verschiedene Zahl von Tastenklicks gelegt. Die folgende Tabelle listet die Aktionen auf, die mit unterschiedlichen Klicks erreichbar sind. Die Buchstaben L, M und R stehen dabei für die linke, mittlere (Rädchen) bzw. rechte Maustaste. Eine nachfolgende Zahl gibt die Anzahl der notwendigen Klicks an. (LR) 2 bedeutet also, dass die linke und die rechte Maustaste zwei mal hintereinander zu klicken sind.

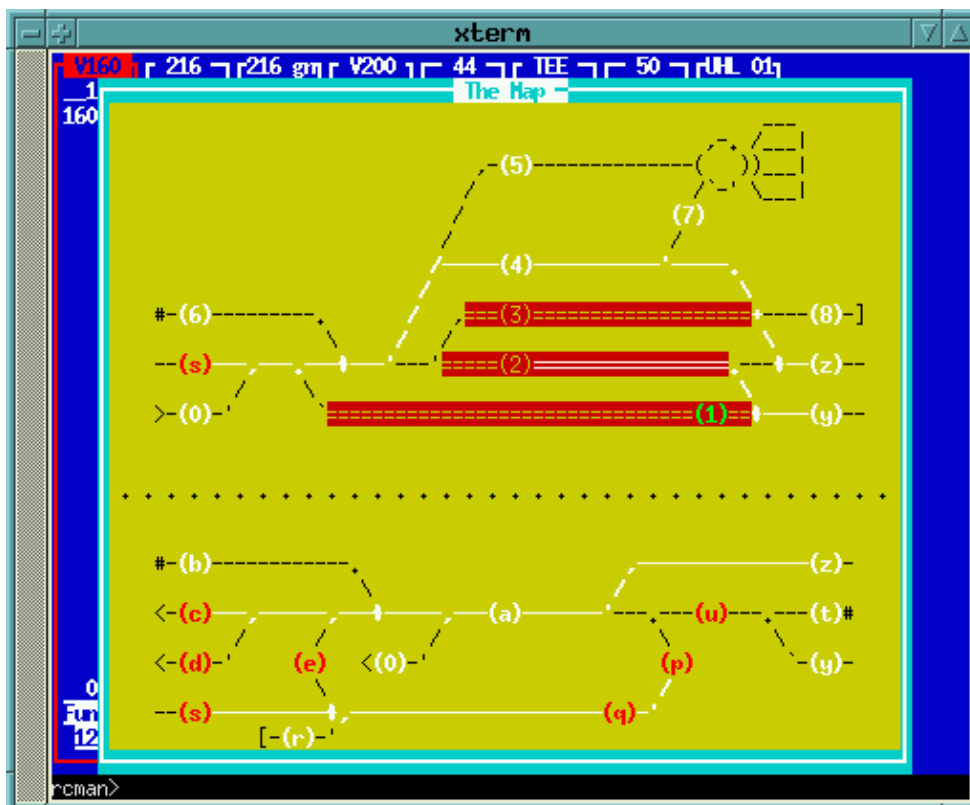
<b>Drehrad</b>	Das Drehrad verändert die Geschwindigkeit der Lokomotive.
<b>L5</b>	Schaltet die Grundfunktion der Lokomotive (Function) ein bzw. aus.
<b>L1</b> bis <b>L4</b>	Schaltet die Funktion 1 bis 4 der Lokomotive ein bzw. aus.
<b>M1</b>	Schaltet die Fahrtrichtung der Lokomotive um. Gleichzeitig wird die Geschwindigkeit auf 0 reduziert. Diese Funktion kann auch dazu genutzt werden, um einen Nothalt der aktuellen Lokomotive durchzuführen.
<b>(LR)1</b>	Schaltet den Fahrstrom ein bzw. aus. Dies entspricht einem Nothalt der kompletten Anlage.
<b>(LR)2</b>	Reduziert die Geschwindigkeit direkt auf 0. Es wird jedoch kein Nothalt gemacht. D.h. die Lokomotive reduziert ihre Geschwindigkeit entsprechend der am Dekoder eingestellten Verzögerung.
<b>R1</b>	Wählt die nächste rechts liegende Lokomotive aus. Lokomotiven, die bereits von einem anderen Eingabegerät (Maus) verwaltet werden, werden dabei übersprungen.
<b>R2</b>	Wählt die nächste links liegende Lokomotive aus. Lokomotiven, die bereits von einem anderen Eingabegerät (Maus) verwaltet werden, werden dabei übersprungen.
<b>(LR)2</b>	Reduziert die Geschwindigkeit direkt auf 0. Es wird jedoch kein Nothalt gemacht. D.h. die Lokomotive reduziert ihre Geschwindigkeit entsprechend der am Dekoder eingestellten Verzögerung.

Für jedes Eingabegerät, also Tastatur, Maus usw. lässt sich gezielt eine akustische Rückmeldung aktivieren. Dies ist vor allem bei der Verwendung einer kabellosen Maus von Vorteil. Steht man nämlich möglichst nah am Ort des Geschehens direkt an der Lok und somit relativ weit entfernt vom Computer, so ist es recht schwierig oder gar unmöglich, die Anzeige abzulesen. Um dennoch eine Rückmeldung über die mit der Maus gemachten Eingaben zu erhalten, so werden in der Grundeinstellung von *rcman* unterschiedliche Geräusche für unterschiedliche Änderungen an einer Lok gemacht. Die Erhöhung der Geschwindigkeit beispielsweise wird durch Töne mit steigender Frequenz angezeigt. Das Einschalten einer Dekoder–Funktion wird mit einem hohen, das Ausschalten mit einem tiefen Ton quittiert. Auch alle anderen Eingaben führen zu einer akustischen Rückmeldung. Somit weiß man nicht nur, dass eine Eingabe vom Computer entgegen genommen wurde, sogar der aktuelle Zustand wird über die Töne zurück gemeldet.

## 4. Steuerung von Weichen, Weichenstraßen und Signalen

Um zwischen der Anzeige der Lokinformationen und den Weichenstraßen hin und her zu schalten dient die Tab-Taste. Die Steuerung der Lokomotiven bzw. Weichen/Weichenstraßen kann unabhängig von der jeweils gewählten Anzeige erfolgen. Somit können Loks auch gesteuert werden (mit der Tastatur und mit der Maus), während die Weichenstraßen angezeigt werden und umgekehrt können Weichenstraßen aktiviert werden, während die Lokomotiv-Zustände dargestellt sind.

Die Darstellung von Weichen, Weichenstraßen und Signalen zeigt die folgende Abbildung: (Die Höhe des Plans ist 26 Zeilen. Daher wird *rcman* mit einer Fehlermeldung beendet, falls versucht wird, den Plan auf einem normalen Bildschirm mit nur 25 Zeilen darzustellen — siehe [Systemanforderungen](#).)



Die Grafik zeigt Weichenstraßen, Punkte auf den Weichenstraßen und weiteres Beiwerk, das nur zur informativen und gefälligen Darstellung dient, aber keine weitere Funktion hat. Punkte auf den Weichenstraßen werden durch einen in Klammern gesetzten Buchstaben (bzw. Ziffer) dargestellt (z.B. ( 1 )). Derartige Punkte werden einerseits verwendet, um Weichenstraßen über ihren Anfangs- und Endpunkt benennen zu können. Weiterhin dienen die Punkte zur Visualisierung von Signalen.

Weichenstraßen zwischen Anfangs- und Endpunkt können je nach aktueller Konfiguration verschiedene Darstellungen annehmen. Schwarze Verbindungslinien zeigen inaktive Straßen an, weiße Linien signalisieren aktive Straßen. Eine Straße kann rot hinterlegt sein, um anzuzeigen, dass sie belegt ist. In diesem Fall wird die Verbindungslinie auch doppelt dargestellt.

Signale werden entsprechend ihres Zustands rot bzw. grün dargestellt, im Gegensatz zu einfachen Start- bzw. Endpunkten, die einfach weiß sind.

## Steuerung über die Tastatur

Die Aktivierung von Weichenstraßen ist sehr einfach. Zunächst wird die Taste des Startpunkt gedrückt, also z.B. die Taste 0. *rcman* zeigt anschließend in der Statuszeile eine Liste aller gültiger Endpunkte an, beispielsweise [ 1 , 2 , 3 , 4 , 5 , 7 , 8 ]. Durch drücken der Taste, die dem gewünschten Endpunkt entspricht, wird die Weichenstraße aktiviert. Wird also die Weichenstraße 0-1 gewählt, so ergibt sich folgende neue Darstellung:

*Diese Grafik fehlt derzeit noch...*

Entsprechend werden auch die Signale gesteuert. Durch zweimaliges Drücken der für das Signal stehenden Zeichens wird der Zustand des Signals umgeschaltet.

## 5. Steuerung einer Drehscheibe

*Vorbemerkung: Ich weiß nicht, auf welche Art Märklin die Drehscheibe digital ansteuert. Ich habe also einen eigenen Ansatz entwickelt, der meinen Ansprüchen soweit entspricht. Für Verbesserungsvorschläge bin ich natürlich immer offen!*

Die Kontrolle der Drehscheibe erfolgt in *rcsh* einfach zeitgesteuert. Die Abgänge der originalen Drehscheibe von Märklin sind in 15° Winkel eingeteilt. (Ok, das stimmt nicht ganz. Teilweise sind es auch 17,5°. Das stört aber nicht weiter.) Somit ergeben sich theoretisch 24 Haltepositionen. Natürlich ist diese Anzahl nicht fix. Sie ist in der Konfiguration zu *rcman* einstellbar.

Die Ansteuerung der Drehscheibe erfolgt nun über die Anzahl der Haltepositionen, um die sich die Bühne in die eine oder andere Richtung drehen soll. Dazu muss zunächst die normale Zeit ermittelt werden, die die Drehscheibe für die volle Drehung benötigt. Diese Zeit geteilt durch 24 (bzw. der jeweils gültige Wert) ist somit die Basis aller Bewegungen.

In *rcman* kann die Drehscheibe über die Tasten F11 und F12 bewegt werden. F11 steht für eine Drehung gegen, F12 steht für eine Drehung im Uhrzeigersinn. Nach Eingabe einer der beiden Funktionstasten fragt das Programm die Anzahl der gewünschten Haltepositionen (eine Ziffer), um die sich die Drehscheibe drehen soll und die Drehung wird ausgeführt. Allerdings erfolgt keine visuelle Rückmeldung über den Vorgang. Auch ist es derzeit nicht vorgesehen, die Drehscheibe per Maus anzusteuern. Ich hatte derzeit nicht das Bedürfnis...

## 6. Auswertung der Informationen des Infoports

Der SRCP-Server sendet an interessierte Clients (sofern er entsprechend eingerichtet wurde – Stichwort Infoport) alle Änderungen, die durch einen beliebigen Client an den Lokomotiven, Accessories usw. durchgeführt wurden bzw. Änderungen an Belegungsmeldern.

*rcman* ist darauf vorbereitet, diese Informationen entgegen zu nehmen und auszuwerten. Dabei gibt es unterschiedliche Arten von Informationen, die unterschiedlich behandelt werden. Informationen über Lokomotiven (Geschwindigkeit, Funktionen, usw.) werden direkt im entsprechenden Fenster dargestellt. Auch die Änderung von Belegungsmeldern werden unmittelbar in der Darstellung der Weichenstraßen angezeigt, sofern sie entsprechend konfiguriert wurden.

Etwas anders ist es mit der Darstellung der Stellung von Weichen bzw. Weichenstraßen. Um aus den isolierten Informationen über die Änderung der Stellung einzelner Weichen den Zustand ganzer Weichenstraßen zu ermitteln, ist ein relativ aufwändiger Algorithmus

notwendig. (Dieser Algorithmus wird in einem eigenen [Kapitel](#) skizziert.) Daher ist dieses Verfahren normalerweise nicht eingeschaltet. Es muss mit `-i` eingeschaltet werden. Anschließend werden alle aktiven Weichenstraßen dargestellt, wie sie auch dargestellt werden, wenn sie innerhalb von rcman aktiviert werden.

# Konfiguration *rcman*

Die komplette Definition zu *rcman* befindet sich in der Datei `rcman.def`. Diese Datei ist eine einfache Python-Datei, die die Definition in Form von Funktions- bzw. Variabledefinitionen beinhaltet. Die Datei gliedert sich in verschiedene Teile, die im folgenden nun vorgestellt werden sollen.

## Definition der Lokomotiven

Um Lokomotiven definieren zu können, muss eine Funktion mit dem Namen `defineLocos` mit einem Parameter (`c`) definiert sein. Der Typ des Parameters ist an dieser Stelle nicht relevant. Lediglich die Tatsache, dass der Parameter `c` ein Objekt darstellt, das die Methode `addLoco` kennt. Hier ein Beispiel:

```
def defineLocos(c):
    c.addLoco('V160', locoM5(16), 160)
```

Obiges Beispiel definiert eine Lokomotive. Sie hat den Namen V160 (erster Parameter), der in der ersten Zeile in *rcman* erscheint. Die Lokomotive wird durch einen M5-Dekoder angesprochen mit der Adresse 16 (zweiter Parameter). (Die Klasse `locoM5` wird in der Dokumentation zu [rcsh](#) vorgestellt.) Die Höchstgeschwindigkeit der Lokomotive ist 160km/h (dritter Parameter). Dieser Wert ist der Maximalwert, der im Lokomotivfenster in *rcman* dargestellt wird. Der Wert hat ansonsten keine praktischen Auswirkungen.

Auf diese Weise können beliebig viele Lokomotiven definiert werden. Zusätzliche Lokomotiven werden rechts neben den bereits definierten Lokomotiven im Fenster von *rcman* angezeigt. Sollte der Platz für weitere Lokomotiven nicht ausreichen, so können diese Lokomotiven gezielt ausgewählt werden, wie es unter [Steuerung von Lokomotiven](#) erläutert wurde.

## 1. Definition von Weichenstraßen

Für die Definition steht wie für die Definition der Lokomotiven eine eigene Funktion mit dem Namen `defineStreets` mit ebenfalls einem Parameter `c` zur Verfügung. Das nachfolgende Beispiel zeigt einen unvollständigen ausschnitt dieser Funktion:

```
def defineStreets(c):
    w7 = turnoutM( 7) # Weiche 1/2,3,...
    # Definition weiterer Weichen ausgelassen!

    #####
    # Definiere die Weichenstraßen (sStreet) und Aktivierungen
    #####

    ##
    # start '0'
    #
    t04 = Street();           t04.add(w7.reset, w5.reset, w8.set, w4.set)
    t05 = Street();           t05.add(w7.reset, w5.reset, w8.set, w4.reset)
    t07 = Street(t04);        t07.add(w23.set)
    c.addStreet('0', '4', t04)
    c.addStreet('0', '5', t05)
    c.addStreet('0', '7', t07, '047')

    ##
    # Gleis 4
    #
```

## rcman – Modellbahnsteuerung mit textueller Oberfläche

```
t46 = Street();          t46.add(w4.set,w5.reset,w8.reset)
c.addStreet('4', '6', t46, '64')

##
# Startpunkt 6 (entspricht 0)
#
t67 = Street(t47);      t67.merge(t46)
c.addStreet('6', '7', t67, '647')

c.addSignal('1', signalM(17))
```

Die Definition der Straßen erfolgt üblicherweise in mehreren Schritten. Zunächst werden die Weichen selbst mit Hilfe der durch *rcsh* vorgegebenen Accessory-Typen (z.B. *turnoutM*) definiert. Anschließend wird eine leere Straße mit der Klasse *Street* definiert. Die Straße wird gefüllt, indem der Straße Weichen-Schaltbefehle hinzugefügt werden. Dazu dient die Methode *add*, die beliebig viele Schaltbefehle als Parameter akzeptiert. Ein Schaltbefehl wird durch die Methoden *set* bzw. *reset* eines Accessory-Objekts repräsentiert. Die Straße *t04* wird also aktiviert, indem die Weichen 5 und 7 auf gerade aus gestellt und die Weichen 4 und 8 auf abzweigen gestellt werden.

Weichenstraßen können aber auch aufeinander aufbauen. Beispielsweise solle die Straße *t07* die gleichen Weichen schalten wie *t04*, zuzüglich der Weiche 23. Daher wird zum Erzeugen von *t07* der Parameter *t04* angegeben. Anschließend wird *t07* noch das Schalten der Weiche 23 (abbiegend) hinzugefügt.

Eine Weichenstraße kann auch auf mehreren Weichenstraßen aufbauen. Dazu können die Ansteuerungen der Weichen mehrerer Weichenstraßen in einer Straße zusammengefasst werden. Dazu dient die Methode *merge*. In obigem Beispiel erbt *t67* zunächst die Einstellungen der Straße *t47* und wird anschließend um die Definitionen der Weichenstraße *t46* erweitert. Prinzipiell ließe sich mittels der Methode *add* weiteren Weichenansteuerung hinzufügen.

Bis hier hin sind die Weichenstraßen zwar definiert, so dass sie prinzipiell Weichen ansteuern können. Es bleibt die Aufgabe die Weichen durch den Benutzer in *rcman* bedienbar zu machen. Zu diesem Zweck muss die Methode *addStreet* des Parameters *c* unserer Methode *defineStreets* aufgerufen werden. Diese Funktion erwartet mindestens 3 Parameter. Der erste und zweite Parameter spezifizieren den Start- bzw. Endpunkt der Weichenstraße. Start- bzw. Endpunkt dienen einerseits dazu eine Straße durch den Anwender aktivierbar zu machen, wie es [Steuerung von Weichen, Weichenstraßen und Signalen](#) vorgestellt wurde. Andererseits dienen Start- bzw. Endpunkt dazu, die Straßen im Gleisplan entsprechend darstellen zu können. Der dritte Parameter der Methode *addStreet* gibt die zuvor definierte Straße an.

Anschließend können optionale Parameter folgen, die die komplette Weichenstraße im Gleisplan angeben. Sind diese Parameter nicht spezifiziert, so wird einfach die Weichenstraße angenommen, die sich aus den ersten beiden Parametern (Start- und Endpunkt) ergeben. Die Straße *t04* wird demnach im Gleisplan durch die Weichenstraße "04" repräsentiert. Da die Weichenstraßen im Gleisplan von links nach rechts identifiziert werden, kann es notwendig werden, den Namen der Weichenstraße explizit anzugeben. Kann der Benutzer eine Weichenstrasse von '4' nach '0' aktivieren, so muss dennoch die Straße "04" dargestellt werden:

```
c.addStreet('4', '0', t04, "04")
```

Noch komplizierter wird es, wenn die Straße über einen oder mehrere Zwischenpunkte geht. Dann müssen diese Zwischenpunkte zur Benennung der Weichenstraße im Plan mit

spezifiziert werden:

```
c.addStreet('0', '7', t07, "047")
```

Schließlich kann der Fall eintreten, dass die Straße im Plan aus mehreren Segmenten zusammengesetzt ist. In diesem Fall müssen alle Segmente angegeben werden, damit die Straße korrekt visualisiert werden kann:

```
c.addStreet('a', 's', tas, "ap", "sqp")
```

Im Rahmen der Weichenstraßen werden außerdem auch die Signale definiert. Dazu dient die Methode `addSignal()`. Die Methode akzeptiert 2 Parameter. Der erste Parameter gibt den Namen (bestehend aus genau einem Zeichen) an, über den das Signal zwischen den beiden Zuständen hin- und her geschaltet werden kann. Der zweite Parameter definiert das zu verwendende Accessory-Objekt.

## 2. Definition der Ausgangskonfiguration der Weichenstraßen

Ich persönlich bevorzuge es, beim Start von *rcman* alle Weichen in einen definierten Zustand zu setzen. Ich bin mir dann ziemlich sicher, dass kein Zug Gleise befährt, die derzeit nicht befahrbar sind (beispielsweise Anschlussgleise, die von meiner Anlage wegführen). Sollte dieser Grundzustand nicht gewünscht werden, so kann der Kommandozeilenschalter `-n` verwendet werden, oder aber die nachfolgend beschriebene Funktion einfach leer bleiben.

Die Ausgangskonfiguration wird in der Funktion `activateDefaultStreets` beschrieben:

```
def activateDefaultStreets(c):  
    c.activateStreet('0', '4')
```

`activateDefaultStreets` besteht aus einer Vielzahl von aufrufen der Methode `activateStreet`. Diese Methode akzeptiert zwei Parameter, die mit den ersten beiden Parametern der oben beschriebenen Methode `addStreet` übereinstimmen.

### 3. Definition des Gleisplans

```
mapDefinition = r"""
                                     ,-. /____|
                                     /  ^-' \____|
                                     (7)
                                     /
                                     /----- (4) -----'-----
                                     /
                                     /----- (3) -----'----- (8)-]
#-- (6) -----'-----
-- (s) -----'----- (2) -----'----- (z) --
>-- (0) -'----- (1) -----'----- (y) --
. . . . .

#-- (b) -----'----- (z) -
-- (c) -----'----- (a) -----'----- (u) ----- (t) #
<-- (d) -' (e) < (0) -' (p) \----- (y) -
<-- (s) -----'----- (q) -'
      [- (r) -'

"""
```

Der Gleisplan wird durch eine einfache Zeichnung mittels ASCII-Zeichen repräsentiert. Die Zeichnung wird zweckmäßigerweise als mehrzeiliger String in dreifachen Anführungszeichen als Python-Raw-String abgelegt (in diesem Modus habe Backslashes keine besondere Bedeutung).

Gleise werden durch Minuszeichen, Schrägstriche (Slashes) bzw. umgekehrte Schrägstriche (Backslashes) gezeichnet. Weichen werden je nach Ausrichtung durch Punkt, Komma, Hochkomma (') und umgekehrtes Hochkomma (^) gezeichnet. Endpunkte (und somit auch Signale) werden einfach auf die Gleise gezeichnet und in runde Klammern gesetzt. Alle sonstigen Zeichnungen, die nicht als Gleise identifiziert werden, werden während der Analyse des Gleisplans ignoriert und können somit zur weiteren Ausgestaltung des Plans (beispielsweise Lokschuppen) verwendet werden.

Der Gleisplan muss der Variablen `mapDefinition` zugewiesen werden. Er wird beim Start von `rcman` analysiert. Bei dieser Analyse werden alle Segmente des Gleisplans ermittelt. Segmente sind grundsätzlich von links nach rechts orientiert. In obigem Gleisplan gibt es beispielsweise ein Segment "2z", nicht jedoch ein Segment "z2". Segmente beinhalten immer alle Punkte, über die das Segment führt. Die Straße von '0' nach 'y' wird also mit "01y" benannt. Da Segmente grundsätzlich von links nach rechts analysiert und benannt werden, muss ggf. eine Straße durch mehrere Segmente zusammengesetzt werden. Dies wurde bereits unter [Definition von Weichenstraßen](#) vorgestellt. Die Weichenstraße von 's' nach 'd' verwendet demnach die Segmente "sqp" und "cap".

## 4. Definition von Gleisbesetzmeldern

Da ich bisher nur über eine sehr überschaubare Anzahl von Rückmeldern verfüge, reicht eine sehr einfache Art und Weise zu ihrer Definition:

```
#          01234567890123456
fbNames = " s23    qelup dc"
```

Die Rückmeldernamen werden einfach der Variablen `fbNames` in Form einer Zeichenkette zugewiesen. Der Name eines Rückmelders besteht aus genau einem Zeichen, das den Namen der Endpunkte für Weichenstraßen entspricht (s.o.). Die Rückmelder sind von 0 an durchnummeriert. Die neunte Position in der Zeichenkette oben benennt also den Rückmelder mit der Nummer 9. Im obigen Beispiel hat dieser Rückmelder den Namen `q`. Entsprechend werden auch alle anderen Rückmelder definiert. Nicht vorhandene bzw. belegte Rückmelder werden durch ein Leerzeichen repräsentiert, bzw. werden ganz weggelassen.

Löst ein Zug einen Rückmelder aus, so zeigt *rcman* automatisch an, dass das Gleis belegt ist, wie es unter [Steuerung von Weichen, Weichenstraßen und Signalen](#) vorgestellt wurde.

## 5. Definition der Drehscheibe

Wie bereits unter [4. Steuerung einer Drehscheibe](#) angemerkt, habe ich eine ganz eigene Ansteuerung der Drehscheibe entwickelt.

Die Konfiguration der Drehscheibe wird wie auch andere Definitionen in einer eigenen Funktion vorgenommen:

```
def defineTurntable(c):
    # die Richtung wird über die Dekoderadresse 22 definiert
    directionAccessory = accessoryM(22)
    # Die Drehscheibe wird durch die Dekoderadresse 21 aktiviert
    activationAccessory = accessoryM(21)
    # Definiere nun die Drehscheibe. Sie benötigt 160 Sekunden für eine
    # Umdrehung. Dabei werden 24 Ausgänge mit gleichem Abstand angefahren.
    # Die letzte 1 gibt an, dass die Drehscheibe mit action 1
    # gestartet wird.
    c.addTurntable(
        Turntable(160, 24, directionAccessory, activationAccessory, 1))
```

Sollte keine Drehscheibe vorhanden sein, so kann die Funktion auch leer bleiben. Ansonsten sind zwei Accessories zu definieren. Über das erste Accessory wird die Drehrichtung der Drehscheibe eingestellt. Dabei wird über `actuate(0)` die Drehrichtung gegen und mit `actuate(1)` die Drehrichtung mit dem Uhrzeigersinn geschaltet. Mit den zweiten Accessory wird die Drehscheibe aktiviert. Der letzte Parameter der Klasse `Turntable` gibt den Parameter zu `actuate()` an, ob also mit `actuate(0)` oder mit `actuate(1)` die Scheibe in Drehung gesetzt werden soll.

Der Konstruktor der Klasse `Turntable` erwartet zwei weitere Parameter. Der erste Parameter gibt die Zeit an, die die Drehscheibe für eine vollständige Drehung benötigt. Der zweite Parameter spezifiziert die Anzahl der anzusteuernenden Ausgänge der Drehscheibe, die jeweils eine (annähernd) gleiche Winkeldifferenz zueinander haben müssen. Über diese Angaben kann die Klasse `Turntable` die Zeit berechnen, die benötigt wird, um die Bühne der Drehscheibe zur gewünschten Zielposition zu bewegen.

# Weiterführende Informationen

*rcman* ist derzeit fast ausschließlich bei mir im Einsatz. Daher ist der Funktionsumfang komplett auf meine Bedürfnisse ausgerichtet. Deshalb ist auch die Dokumentation vielleicht noch nicht so ausführlich, wie sie an der einen oder anderen Stelle sein sollte. Daher möchte ich alle bitten, mir Anregungen und Wünsche — sei es zur Dokumentation, sei es zur Software selbst — zukommen zu lassen.

... und nun wünsche ich viel Spaß und viel Erfolg mit *rcman*!

# Anhang: Algorithmus Erkennung von Weichenstraßen

*muss noch geschrieben werden...*

# Appendix A – GNU General Public License

GNU GENERAL PUBLIC LICENSE  
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.  
59 Temple Place – Suite 330, Boston, MA 02111–1307, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

## GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves,

then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically

receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

#### **NO WARRANTY**

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED

OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**END OF TERMS AND CONDITIONS**