# EEP Lua Automatische Zugsteuerung

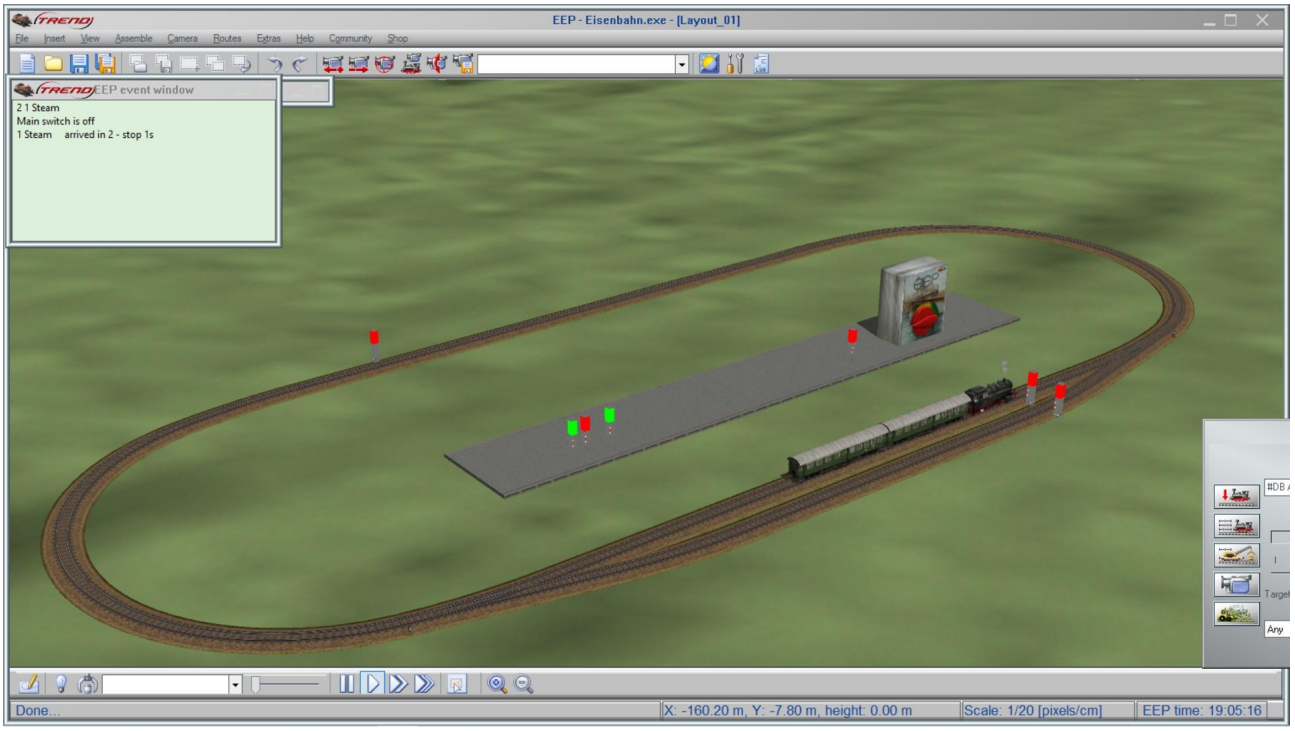
Rudy Boer, Februar 2022
Übersetzt mit www.DeepL.com/Translator

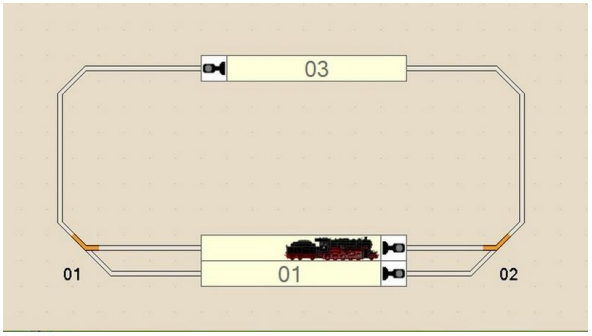
## Zweck

Dieser Lua-Code generiert automatischen Zugverkehr auf jeder EEP-Anlage, ohne dass Sie irgendeinen Code schreiben oder ändern müssen. Um eine Anlage zu steuern, müssen lediglich Daten, die die Anlage definieren, in eine Reihe von Lua-Tabellen und Variablen eingegeben werden ... Daten zu Zügen, Signalen und zu den Weichen, die pro Strecke geschaltet werden sollen.

## Beispiel für eine Layout-Definition

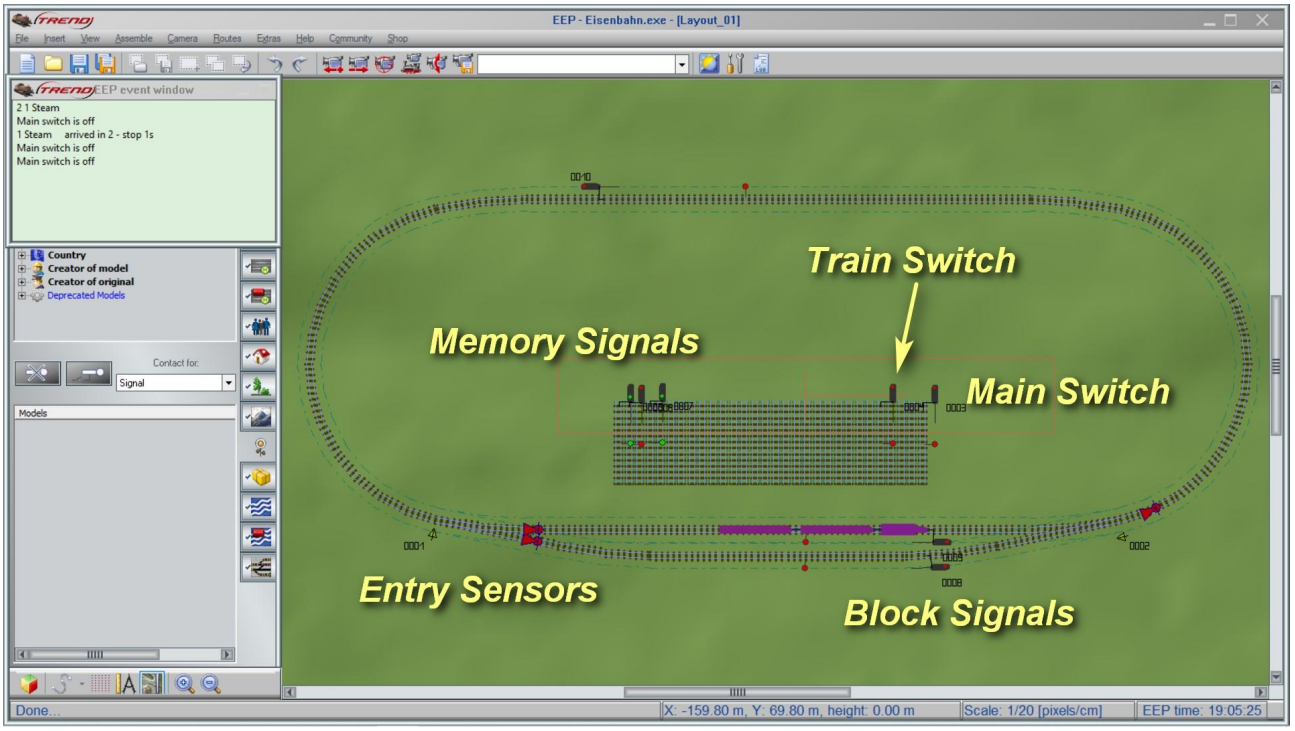
Die folgende Abbildung zeigt ein Beispiel für eine einfache Modelleisenbahnanlage in EEP.

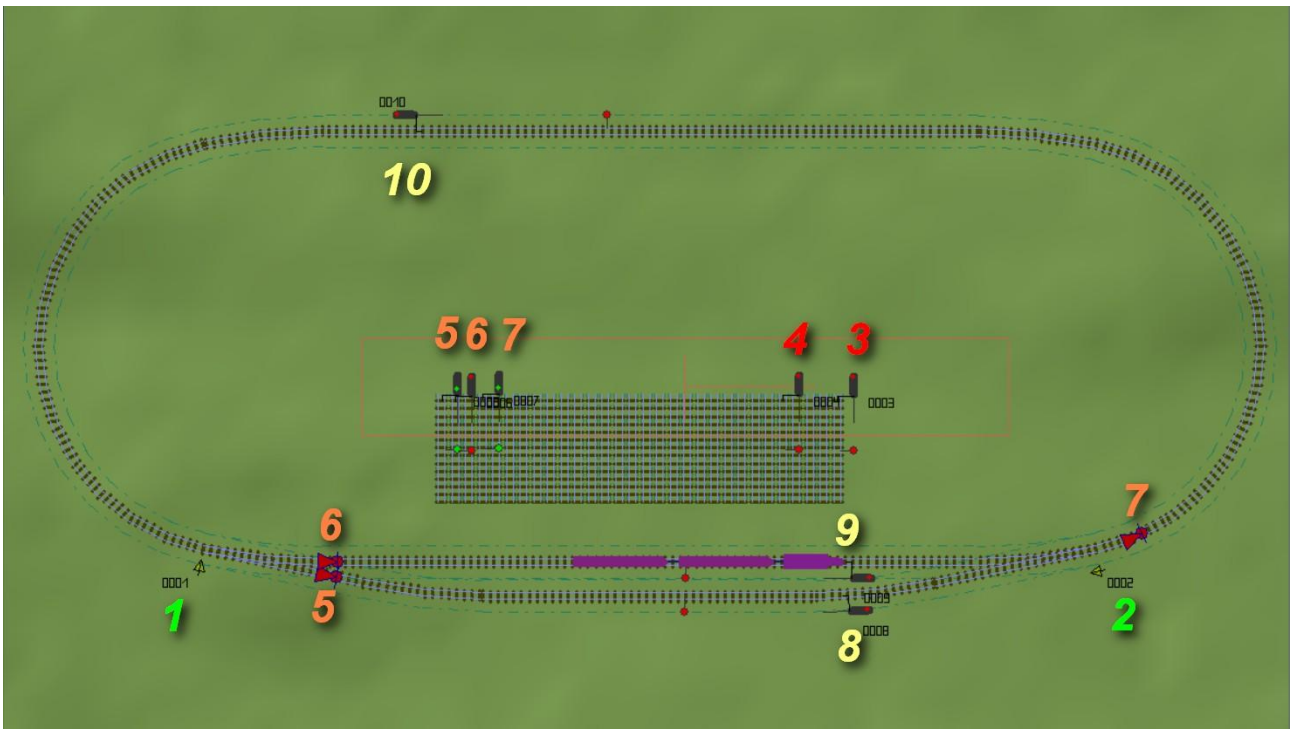



Beim automatischen Verkehr fahren die Züge von Block zu Block. Der erste Schritt ist die Entscheidung, welche Blöcke wir definieren wollen. Hier gibt es nur eine Regel: Weichen können nicht Teil eines Blocks sein, sie sind Teil der Strecke zwischen zwei Blöcken. Für diese Anlage scheint es logisch, drei Blöcke zu haben, wie hier gezeigt.

Jeder Block erhält ein Blocksignal, das den Zug anhalten kann. Jeder Block erhält auch ein Speichersignal. Diese werden auf einem virtuellen Bedienfeld platziert. Sie zeigen an, ob ein Block frei oder besetzt ist, und dienen gleichzeitig als Speicher, da die Signalzustände beim Verlassen von EEP gespeichert werden. Jeder Block erhält an seinem Eingang einen Gleissensor, der sein Speichersignal auf rot schaltet. Ein Hauptschalter und ein Ein/Aus-Schalter pro Zug (in diesem Beispiel gibt es nur einen Zug) werden ebenfalls auf dem Bedienfeld hinzugefügt.



Das obige Bild zeigt die Weichen- und Signalnummern, wie sie von EEP beim Aufstellen generiert wurden. Diese Nummern werden in den folgenden Tabellen und Variablen verwendet, um das Layout zu definieren:

```
train[1] = {name="Steam", onoff=4, route=0, block=2}

--     block  1, 2, 3
allowed[1]= {15, 1, 1} -- train 1 has a 15s stop time in block 1
twowayblk = { 0, 0, 0} -- all blocks are one way traffic
blocksig  = { 8, 9,10} -- numbers of the block signals
memsig    = { 5, 6, 7} -- numbers of the memory signals

route[ 1] = { 1, 3,turn={2,1}} -- from block, to block, turnouts to switch
route[ 2] = { 2, 3,turn={2,2}}
route[ 3] = { 3, 1,turn={1,1}}
route[ 4] = { 3, 2,turn={1,2}}

MAINSW    = 3 -- number of the main switch
MAINON    = 1 -- 'on' state of the main switch
MAINOFF   = 2 -- 'off' state of the main switch
BLKSIGRED = 1 -- 'red' state of the block signals
BLKSIGGRN = 2 -- 'green' state of the block signals
MEMSIGRED = 1 -- 'red' state of the memory signals
MEMSIGGRN = 2 -- 'green' state of the memory signals
```

Diese Tabellen, die sich am Anfang des Lua-Codes befinden, sind alles, was bearbeitet werden muss. Der Rest des Lua-Codes bleibt gleich, egal welches Layout wir steuern wollen.

# Wie Lua automatischen Zugverkehr generiert

Wenn ein Zug in einen Block einfährt, überfährt er den Einfahrsensor, wodurch das Speichersignal des Blocks auf Rot gesetzt wird. Der Lua-Code wird 5 Mal pro Sekunde durchlaufen. Bei jedem Zyklus prüft Lua, ob die Speichersignale von Grün auf Rot umschalten. Wenn ein solcher Übergang festgestellt wird, löst er die folgende Abfolge von Ereignissen aus.

Zuerst liest Lua die Zugnummer aus der Tabelle blockreserved[b], die die Nummer des Zuges enthält der den Block reserviert hat, und die dort eingetragen wurde als der Zug seine Fahrt begann.

Dann trägt Lua die Zugnummer in die Tabelle request[b] ein, um sich daran zu erinnern, dass dieser Block nun einen neu eingetroffenen Zug enthält, der eine neue Strecke anfordert. Lua füllt auch die Tabelle stoptimer[b] mit der Haltezeit für diesen Zug in diesem Block. Diese Haltezeit wird aus der Tabelle allowed[t][b] gelesen, die gleichzeitig die Tabelle zur Definition der Haltezeit ist. Die Haltezeiten werden in jedem Zyklus heruntergezählt, bis sie Null erreichen.

Lua liest nun die Streckennummer des ankommenden Zuges aus der Zugtabelle train[t].route, die beim Anfahren des Zuges dorthin geschrieben wurde.

Wenn die Route des Zuges bekannt ist, liest Lua den Abfahrtsblock aus der route[r]-Tabelle, die den Abfahrtsblock, den Zielblock und den "Haupt"- oder "Abzweig"-Zustand der dazwischen liegenden Weichen enthält, falls vorhanden. Lua setzt das Speichersignal des Abfahrtsblocks auf grün.

In jedem Zyklus prüft Lua, ob die Speichersignale von Rot nach Grün wechseln. So wird im nächsten Zyklus dieser Rot-Grün-Übergang erkannt, woraufhin der Abfahrtsblock in der Tabelle blockreserved[b] auf "frei" gesetzt wird und auch die Weichen der Strecke in der Tabelle turnreserved[to] auf "frei" gesetzt werden.

Die Phase "Ankunftskontrolle, Kennzeichnung neuer Streckenanfragen und Freigabe alter Blöcke und Weichen" des Zyklus ist nun abgeschlossen. Es folgt nun die Ermittlung der verfügbaren neuen Strecken, die zufällige Auswahl einer Strecke und deren Start.

Für alle oben generierten Fahrstraßenanfragen bestimmt Lua, welche neuen Fahrstraßen verfügbar sind, indem es prüft:

- ob der Hauptschalter und der Einzelzug-Ein/Aus-Schalter eingeschaltet sind
- ob die Haltezeit abgelaufen ist
- welche Fahrstraßen den aktuellen Block des Zuges als Abfahrtsblock haben
- welche Zielblöcke für diesen Zug erlaubt sind und ob diese Blöcke frei sind
- welche Weichen für neue Fahrstraßen benötigt werden und ob sie frei sind

Die verfügbaren Strecken (d. h. die Strecken, die alle oben genannten Bedingungen erfüllen) werden in die Tabelle available[r] eingetragen, die in jedem Zyklus gelöscht und neu aufgebaut wird. Aus dieser Tabelle wird eine Strecke nach dem Zufallsprinzip ausgewählt. Nun legt Lua:

- legt diese Strecke in der Zugtabelle train[t].route ab
- reserviert den Zielblock, indem es die Zugnummer in blockreserved[b] schreibt
- reserviert die Weichen durch Schreiben der Zugnummer in turnreserved[b] für alle Weichen auf dieser Strecke und schaltet die Weichen in den für diese Strecke benötigten Zustand
- setzt schließlich das Blocksignal auf Grün, so dass der Zug weiterfahren kann.

Der gesamte Prozess beginnt wieder von vorne, wenn der Zug in seinem Zielblock erkannt wird.

# Tabellen zur Definition des Layouts, die vom Benutzer konfiguriert werden

**train = {}**

```
train[1] = {name="Passenger", route=0, onoff=14, block=2}
train[2] = {name="Cargo", route=0, onoff=15, block=5}
```

- **name=”Passenger”**. Die Zugnamen werden auf dem Lua-Bildschirm bei der Ankunft in einem Block und beim Beginn einer neuen Strecke angezeigt. Sie haben keinen anderen Zweck als diesen.
- **route = 0**. Die Strecke, auf der sich der Zug befindet. Anfänglich müssen diese auf 0 gesetzt werden.
- **onoff=14**. Die Signalnummer, die als Ein/Aus-Schalter für diesen Zug verwendet wird.
- **block=2**. Die Blocknummer, in der sich dieser Zug derzeit auf der Anlage befindet.

Wenn der Lua-Code zum allerersten Mal ausgeführt wird, weiß er nicht, wo die Züge sind. Siehe das Kapitel "Wie man Züge auf das Gleis stellt und Lua initialisiert" weiter unten.

**allowed = {}**

```
/*       block 1, 2, 3, 4, 5, 6, 7 */
allowed[1] = { 1, 1, 1, 1, 1, 0, 0}
allowed[2] = {23, 1,35, 0, 0, 1, 1}
```

Diese Tabelle gibt an, ob ein Zug in einem Block zugelassen ist und ob er eine Haltezeit hat:

- 0: dieser Zug ist in diesem Block nicht erlaubt
- 1: dieser Zug ist in diesem Block erlaubt, keine Haltezeit
- >1: dieser Zug ist in diesem Block erlaubt und hat eine Haltezeit. Die Zeit, die der Zug benötigt, um vom Blockeingangssensor bis zum Blocksignal zu fahren, muss berücksichtigt werden. Beispiel: Wenn die gemessene Fahrzeit vom Sensor bis zum Signal dieses Zuges in diesem Block 15 Sekunden beträgt und eine Haltezeit von 8 Sekunden gewünscht wird, ist die einzugebende Zahl 15+8=23.

Im obigen Beispiel:

- Zug 1 ist in den Blöcken 6 und 7 nicht zugelassen. Es sind keine Haltezeiten angegeben.
- Zug 2 ist in den Blöcken 4 und 5 nicht zugelassen. Er hat eine Haltezeit von etwa 8s in Block 1 und 20s in Block 3 (unter der Annahme, dass die Fahrzeit vom Sensor zum Signal in beiden Blöcken 15s beträgt).

**blocksig = {14,15,16,…}**

Jeder Block hat ein Gleissignal, an dem die Züge anhalten, wenn es rot ist. Das Signal wird von Lua gesteuert. Es wird auf grün geschaltet, wenn der Zug eine neue Strecke zu einem angrenzenden Block beginnen darf. Es wird wieder auf rot geschaltet, wenn der Zug im nächsten Block ankommt.

Wird ein Block in zwei Richtungen verwendet, wird er als zwei Blöcke in entgegengesetzten Richtungen behandelt, jeder mit seinem eigenen Blocksignal und Speichersignal. Dies wird in der Tabelle twowayblock[b] angegeben.

**WICHTIG:** In EEP haben nicht alle Signale den gleichen Zustand. Bei einigen Signalen bedeutet 1 'geschlossen', 2 'offen', bei anderen ist es umgekehrt. Verwenden Sie daher nur Blocksignale für das gesamte Layout die den gleichen Zustand für Rot/Grün haben.

Die korrekten Zustände der verwendeten Block- und Speichersignale werden mit Variablen definiert, siehe das Kapitel 'Variablen ...' weiter unten.

**memsig = {39,20,21,…}**

Neben einem Gleissignal verfügt jeder Block auch über ein Speichersignal. Dieses befindet sich nicht auf den Zugschienen, sondern auf einem separaten (versteckten) Gleis. Sie dienen der visuellen Rückmeldung von besetzten Blöcken, als Teil eines Bedienfeldes. Falls gewünscht, können sie auch vollständig verborgen werden.

Sie werden von einem Sensor auf rot geschaltet, der ausgelöst wird, wenn ein Zug in den Block einfährt. Lua erkennt diesen Übergang von Grün nach Rot und weiß nun, dass der Zug in diesem Block angekommen ist. Lua schaltet das memsig[b] des Abfahrtsblocks auf grün (frei).

EEP speichert den Zustand der Signale beim Verlassen des Programms. Beim späteren Start wird die Tabelle memsig[b] durch Auslesen dieser Speichersignale gefüllt.

**WICHTIG**: Verwenden Sie nur Speichersignaltypen, die ähnliche Zustände für Rot/Grün haben.

**twowayblk = { 0, 0, 4, 3, 0, 0}**

In diesem Beispiel sind die Blöcke 3 und 4 "Zwillinge" von Zweiwegblöcken. Wenn ein Zug Block 3 für seine neue Strecke reserviert, liest Lua in twowayblk[3], dass er auch Block 4 reservieren muss. Umgekehrt, wenn ein Zug den Block 4 reserviert, liest Lua in twowayblk[4], dass er auch den Block 3 reservieren muss.

**route = {}**

```
route[1]={2,3}
route[2]={3,8,7,1,12,2} -- 1:main, 2:branch
```

Es werden Strecken von einem Block zum nächsten angegeben. Die Tabelle enthält den Abfahrtsblock, den Zielblock und eventuell eine oder mehrere Weichen, die geschaltet werden müssen. Im obigen Beispiel:

- für die Strecke 1 ist 2 der Abfahrtsblock, 3 der Zielblock, und es sind keine Weichen zu schalten.
- für die Strecke 2 ist 3 der Abfahrtsblock, 8 ist der Zielblock, die Weiche 7 muss auf "Haupt" und die Weiche 12 auf "Abzweig" geschaltet werden.

# Variables configured by the user

**PLACE_TRAINS = 0 or PLACE_TRAINS = 1**

Siehe das Kapitel "Wie man Züge auf das Gleis stellt und Lua initialisiert" weiter unten, um zu erfahren, wie diese verwendet wird.

**MAINSW = 27**
Die Adresse des Signals, das als Hauptschalter verwendet wird. Wenn der Hauptschalter eingeschaltet ist, können die Züge fahren. Darüber hinaus verfügt jeder Zug über einen individuellen Ein-/Ausschalter. Wenn ein Zugschalter oder der Hauptschalter ausgeschaltet wird, fahren die Züge zunächst auf ihrer aktuellen Strecke weiter, bis sie ihren Zielblock erreichen, wo sie durch ein rotes Signal angehalten werden. Von dort aus fahren sie erst wieder auf eine neue Strecke, wenn sie wieder eingeschaltet werden.

**MAINON = 1, MAINOFF = 2, oder umgekehrd**
Die Werte können entweder 1 oder 2 sein, je nachdem, welche Art von Signal als Hauptschalter verwendet wird. Der Benutzer muss dies im EEP überprüfen.

**BLKSIGRED = 1, BLKSIGGRN = 2, oder umgekehrd**
Der im EEP verwendete Wert für ein rotes und grünes Blocksignal, der je nach Art der verwendeten Blocksignale 1 oder 2 sein kann. Der Benutzer muss dies in EEP überprüfen. Achten Sie darauf, nur Blocksignale zu verwenden, die im gesamten Layout denselben Status haben.

**MEMSIGRED = 1, MEMSIGGRN = 2, oder umgekehrd**
Der im EEP verwendete Wert für ein rotes und grünes Speichersignal, der je nach Art der verwendeten Speichersignale 1 oder 2 sein kann. Der Benutzer muss dies im EEP überprüfen. Achten Sie darauf, dass Sie nur Speichersignale verwenden, die im gesamten Layout den gleichen Status haben.

# Tables internally used by Lua

**memsigold = {}**

In dieser Tabelle wird der vorherige Wert von memsig[b] gespeichert, so dass ein 0>1-Übergang bei der Ankunft eines Zuges bzw. ein 1>0-Übergang bei der Abfahrt eines Blocks erkannt werden kann.

**blockreserved = {}**

blockreserved[b] enthält 0, wenn ein Block frei ist. Freie Blöcke sind für eine neue Strecke verfügbar. Wenn ein Block reserviert ist, enthält er die Nummer des Zuges, der ihn reserviert hat. Diese Blöcke sind für neue Strecken nicht verfügbar. Wenn ein Zug am Zielblock seiner Strecke ankommt, wird der Abfahrtsblock freigegeben.

**turnreserved = {}**

Wenn ein Zug eine Strecke reserviert, reserviert er nicht nur den Zielblock, sondern auch die Weichen zwischen den Blöcken, damit sie nicht von anderen Zügen benutzt werden

können. Wenn ein Zug am Zielblock seiner Strecke ankommt, werden die Weichen, die für diese Strecke reserviert waren, wieder freigegeben.

**request = {}**

Wenn ein Zug am Zielblock seiner Strecke ankommt, füllt Lua request[b] mit der Zugnummer, um sich zu merken, dass dieser Zug nun eine neue Strecke anfordert.

**available = {}**

Diese Tabelle wird in jedem Lua-Zyklus geleert und dann neu aufgebaut. Sie enthält die Nummern der "verfügbaren Strecken" für alle Züge, die neue Strecken beantragt haben. Lua wählt zufällig eine Strecke aus der available[r]-Liste aus ... dieser Zug ist der einzige, der in diesem Lua-Zyklus eine neue Strecke beginnen darf. Lua reserviert nun den Zielblock und die Weichen dieser Strecke. Das kann bedeuten, dass andere Strecken, die in diesem Zyklus verfügbar waren, jetzt nicht mehr verfügbar sind, weshalb die Tabelle in jedem Lua-Zyklus geleert und neu berechnet wird.

**stoptimer = {}**

Die Haltezeiten pro Zug und Block werden in der Tabelle allowed[t][b] angegeben. Bei Ankunft wird stoptimer[b] mit 5x der definierten Haltezeit geladen. 5x, weil Lua 5x pro Sekunde zykliert und die Stoppuhren bei jedem Zyklus dekrementiert werden. Wenn stoptimer[b] 0 erreicht, ist die Haltezeit für diesen Zug in diesem Block abgelaufen und der Zug wird für eine neue Strecke verfügbar.

# Wie man Züge auf das Gleis stellt und Lua initialisiert

Wenn der Lua-Code zum ersten Mal ausgeführt wird, weiß er noch nicht, wo die Züge sind ... das müssen wir ihm sagen.

Es gibt eine Codezeile, die PLACE_TRAINS = 0 oder PLACE_TRAINS = 1 lautet. Lua prüft diese Variable und wenn sie 1 ist, schaltet Lua den Hauptschalter und die einzelnen Zugschalter aus und stoppt die Auswahl neuer Strecken.

Es gibt eine Codezeile, die PLACE_TRAINS = 0 oder PLACE_TRAINS = 1 lautet. Lua prüft diese Variable und wenn sie 1 ist, schaltet Lua den Hauptschalter und die einzelnen Zugschalter aus und stoppt die Auswahl neuer Strecken.

Wenn alle Züge manuell an die gewünschten Positionen gefahren werden, müssen wir die Tabelle trains[t] bearbeiten, um die aktuellen Blocknummern der Züge wiederzugeben und die Nummern der Ein- und Ausschaltsignale sowie die Zugnamen zu definieren.

Nachdem Sie die Zugtabelle bearbeitet haben, laden Sie den Code erneut, immer noch mit PLACE_TRAINS = 1. Lua liest die Tabelle train[t], reserviert die Blöcke, auf denen ein Zug steht, setzt die entsprechenden Speichersignale auf rot und schreibt den Blockstatus (frei: 0, reserviert: Zug nr) auf die Platte.

Ändern Sie nun den Code auf PLACE_TRAINS = 0 und laden Sie den Code neu. Dies bringt Lua in den Fahrmodus. Was jetzt passiert, wird im Kapitel 'Wie es funktioniert' oben erklärt.

# Anhalten der Fahrt und Speichern des aktuellen Zustands

Vor dem Beenden von EEP ist es ratsam, den Hauptschalter auszuschalten und zu warten, bis alle Züge an einem Signal zum Stillstand gekommen sind, da sonst der Status der Anlage möglicherweise nicht eindeutig ist und nicht korrekt gespeichert wird.

Es ist nicht notwendig, die Anlage über das Menü zu speichern, der aktuelle Zustand wird automatisch gespeichert, wenn EEP beendet wird.