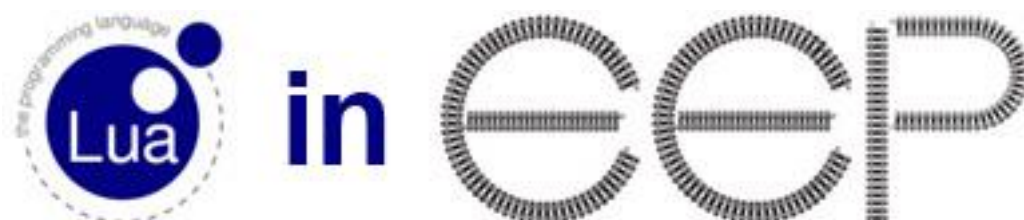


Grundlagen zu



bis einschließlich EEP 18.0 (Ausgabe: 06.08.2025)

1	Die Programmiersprache Lua	1—1
2	Lua-Sprachreferenz	2—1
3	Lua-Einbindung in EEP	3—1
3.1	Der (interne) Lua-Skript-Editor	3—1
3.2	Nutzung eines externen Editors	3—2
3.3	Von EEP generiertes Lua-(Start)-Skript	3—4
3.4	Ausgabe von Meldungen im EEP-Ereignisfenster.....	3—7
4	Aufruf von Lua-Funktionen in Kontaktpunkten.....	4—1
5	Lua-Wizard.....	5—3
5.1	Erstellung eines Lua-Skripts für einen EEP-Fahrplan	5—3
5.2	Erstellung eines Lua-Skripts für einen beschränkten Bahnübergang	5—4
5.3	Erstellung einer EEP-spezifischen Kamera-Funktion.....	5—6
5.4	Erstellung von Lua-Codes zu EEP-spezifischen Funktionen.....	5—8

Weitere Informationen zu Lua in EEP:

- [EEP-spezifische Lua-Variablen und -Funktionen](#)
- [EEP-Tutorial-Anlagen mit Lua](#)

Achtung: Diese Links funktionieren nur wenn alle 3 Dateien in einem Ordner liegen.

1 Die Programmiersprache Lua

Lua ist eine imperative und erweiterbare Skriptsprache, also eine Programmiersprache, die während der Ausführung des Programms in Computerbefehle übersetzt wird.

Lua wurde 1993 von Roberto Ierusalimsky, Luiz Henrique de Figueiredo und Waldemar Celes in der Computer Graphics Technology Group der Päpstlichen Katholischen Universität von Rio de Janeiro in Brasilien entwickelt. Lua ist freie Software und wurde bis zur Version 4 unter einer eigenen BSD-Lizenz veröffentlicht, ab Version 5 unter der MIT-Lizenz.



"Lua" (ausgesprochen LOO-ah) bedeutet "Mond" auf Portugiesisch. Als solches ist es weder ein Akronym noch eine Abkürzung, sondern ein Substantiv. Genauer gesagt ist "Lua" ein Name, der Name des Mondes der Erde und der Name der Sprache. Wie die meisten Namen sollte er klein geschrieben werden, mit einem Großbuchstaben am Anfang, also "Lua". Bitte schreiben Sie ihn nicht als "LUA", was sowohl hässlich als auch verwirrend ist, denn dann wird er zu einem Akronym mit unterschiedlichen Bedeutungen für verschiedene Leute. Schreiben Sie also bitte "Lua" richtig!

Ein Appell der Lua-Entwickler

Lua kann sowohl zum Verfassen eigenständiger Programme verwendet werden als auch als eingebettete Sprache dienen. Die Vorteile von Lua sind die geringe Größe von 120 kB, die Erweiterbarkeit und die hohe Geschwindigkeit, verglichen mit anderen Skriptsprachen.

Um einzelne Komponenten eines Computerspiels wie z.B. Konfigurationsdateien oder die künstliche Intelligenz von computergesteuerten Charakteren oder Gegnern von der Spiel-Engine zu trennen, kommt Lua bei der Entwicklung von Computerspielen oft zum Einsatz. Dies macht die meist teuer entwickelte Spiel-Engine flexibler und ermöglicht eine mit geringerem Aufwand verbundene Wiederverwendbarkeit, weshalb Lua im Bereich proprietärer Spiele verwendet wird. (Quelle: [Wikipedia](#))

Lua wurde in EEP mit dem Plug-in 2 zu "Eisenbahn-X" (EEP10.2) eingeführt.

Logo der Skriptsprache Lua: Autor Alexandre Nakonechnyj (Grafik-Design) und Lua-Team (PostScript-Code)

2 Lua-Sprachreferenz

Lua ist eine lebende Programmiersprache, d.h. sie wird laufend erweitert. Zu jeder Version wird ein Referenzhandbuch veröffentlicht, das die Syntax der Sprache beschreibt.

In EEP 18 ist die Lua-Version 5.3 enthalten. Referenzhandbücher sind für diese Version nur in [Englisch](#) und [Russisch](#) verfügbar. Für die Lua-Version 5.2 gibt es neben einem englischen Referenzhandbuch noch welche in [Polnisch](#) und [Portugiesisch](#). Ein Referenzhandbuch in deutscher Sprache ist bislang "nur" für die Lua-Version 5.1 unter

<https://www.lua.org/manual/5.1/de/>

verfügbar. Der Umfang dieses Referenzhandbuch dürfte aber - vielleicht mit Ausnahme für einige wenige Lua-Experten - durchaus mehr als ausreichend sein, um Lua-Skripte für EEP zu schreiben.

Für Lua-Anfänger sind sicher deutschsprachige Tutorials mit Beispielen erklärender:

- [Lua-- Erste Schritte in der Programmierung](#) - PDF von Knut Lickert;
- [Lua-Tutorial - Multi Theft Auto](#) - Online-Version (Text);
- [Lua lernen - Teil 1 \(Active VB\)](#) - Online-Version (Text);
- [Programmieren Lernen mit Lua \(maschinennah\)](#) – Online Version (Text und Videos);
- [Lua rennt – ein Einsteiger-Tutorial](#) – PDF von Götz Meyer mit EEP-Bezug.

Probieren geht über Studieren. Die schnellste und einfachste Art Lua-Code auszuprobieren, geht über einen Lua-Compiler. Lua.org hat leider seinen eigenen Demo-Compiler eingestellt und empfiehlt stattdessen die folgenden:

- <https://onecompiler.com/lua/>
- https://www.tutorialspoint.com/execute_lua_online.php
- <https://www.mycompiler.io/de/new/lua>

Code eingeben, auf "run", "Execute" oder "Ausführen" klicken und schon erscheint das Ergebnis oder eine Fehlerbeschreibung.

Neben dem in Lua enthaltenen Sprachumfang stehen in EEP zusätzlich noch eine Anzahl spezifischer Variablen und Funktionen zur Verfügung, die eine Verbindung zu EEP-Anlagen herstellen und diese beeinflussen können. Sie beginnen alle mit EEP (siehe [Lua-Handbuch](#)).

3 Lua-Einbindung in EEP

3.1 Der (interne) Lua-Skript-Editor

Zur Eingabe von Lua-Code enthält EEP einen Skript-Editor (Bild 3.1). Diesen Editor öffnet man mit einem Klick mit der linken Maustaste auf das in Bild 3.2 gekennzeichnete Symbol in der Menüleiste.



Bild 3.2

Der in EEP installierte Lua Skript-Editor ist ein sehr einfacher Editor. Er bietet u.a. keine farbliche Lua-Unterstützung, der Tabulator ist fest auf 4 Zeichen eingestellt und er bietet keine Erweiterungsmöglichkeiten z. B. für Vorschläge zur Lua-Syntax und spezifischen EEP-Funktionen.

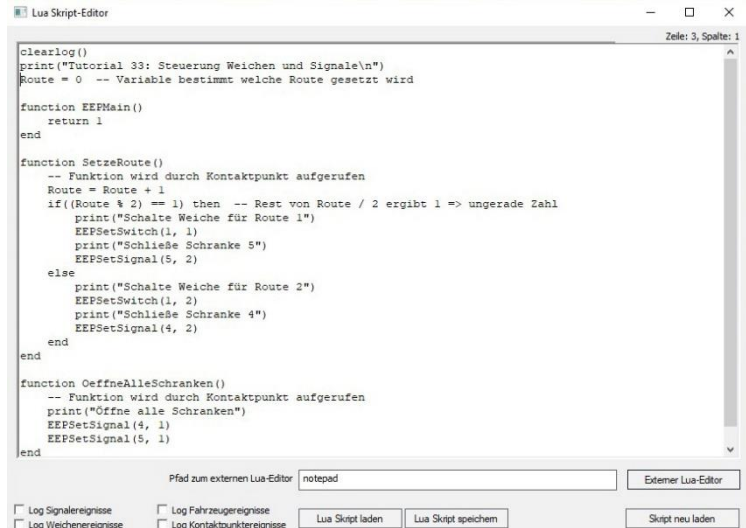


Bild 3.1

Wichtig: Ein erstelltes Skript muss immer an EEP übertragen werden.

Nachdem ein Skript erstellt oder Änderungen daran vorgenommen wurden, muss stets die Schaltfläche "Skript neu laden" (Bild 3.3 und Bild 3.1 unten rechts) angeklickt werden! Ansonsten geht das Skript oder die daran vorgenommenen Änderungen verloren, wenn der Skript-Editor geschlossen wird. (Beachte hierzu auch den letzten Absatz dieses Kapitels.)



Bild 3.4

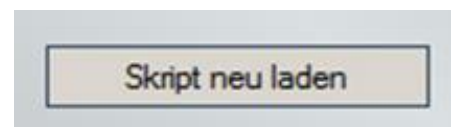


Bild 3.1

Wurde der Skript-Editor zur Bearbeitung aus dem 2D-Fenster aufgerufen, startet das neu geladene Lua-Skript immer erst nach der Umschaltung ins 3D-Fenster. Wurde der Skript-Editor aus dem 3D-Editor zur Bearbeitung aufgerufen, startet das Skript direkt nach dem Anklicken der "Skript neu laden"-Schaltfläche.

Ein erstelltes Skript wird automatisch gemeinsam mit der Anlage gespeichert und geladen.

Eine Speicherung des Skripts unabhängig von der Anlage - zum Beispiel als Backup -, erfolgt über die Schaltfläche "Lua Skript speichern" (rechts im Bild 3.4). Umgekehrt kann man ein extern gespeichertes Skript über die Schaltfläche "Lua Skript laden" (links im Bild 3.4) in den Skript-Editor holen. Aber **Achtung**, auch hier muss das neu geladene Skript mit der Schaltfläche "Skript neu laden" (Bild 3.3) nach EEP übertragen

werden.

Lua-Skripte in EEP sind auf maximal 67634 Zeichen begrenzt. Sollte es notwendig sein, diese Kapazitätsgrenze zu überschreiten, so muss ein Teil des Codes in andere .lua-Dateien ausgelagert werden. Dies sollte möglichst für zusammenhängende Blöcke/Module erfolgen, z.B. Code für Bahnübergänge, Ampelsteuerungen, Fahrstraßenschaltungen, um nur einige Möglichkeiten zu nennen. Diese "Auslagerungsdateien" sollten bevorzugt im LUA-Ordner der EEP-Version oder in einem Unterordner darunter gespeichert werden. Dann sind sie leicht über die allgemeine Lua-Funktion *require()* in das Hauptskript *MeineAnlage.lua* einzubinden: `require ("Ampelsteuerung")` oder `require ("MeineAnlage/BUe")` oder `require ("MeineAnlage\\Fahrstrassen")`. Die Pfadangabe hat relativ zum Ordner LUA im EEP-Verzeichnis und ohne die Dateiendung .lua zu erfolgen. Trennzeichen zwischen Ordner- und Dateinamen ist entweder ein einfacher Schrägstrich oder ein doppelter Backslash.

Wenn man den Lua-Skript-Editor über das "Fenster schließen"-Kreuz im Fenster oben rechts abbricht/schließt, wurde bis einschließlich EEP 17.1 Plug-in 1 das Lua-Skript danach trotzdem einmal ausgeführt. Dies geschieht **mit dem Plug-in 2 zu EEP 17.2 sowie EEP 18 und höher** nicht mehr.

3.2 Nutzung eines externen Editors

Bei der Nutzung eines externen Editors zur Erstellung von Lua-Code für eine EEP-Anlage muss man unbedingt darauf achten, dass als Kodierung **ANSI** (und nicht UTF-8) eingestellt ist.

Bis einschließlich EEP 16 kann man bei geschlossenem EEP nur den Data-Slot-Teil am Ende der *MeineAnlage.lua* bearbeiten, der im internen Lua-Skript-Editor nicht sichtbar ist. Wird auch der Code geändert, ignoriert EEP ihn vollständig und zeigt stattdessen den Standard-Eröffnungscode an. Dies kann man umgehen, indem man den Code im externen Editor nach der Bearbeitung in die Zwischenablage kopiert und ihn daraus dann in den internen EEP-Lua-Editor einfügt.

Ab EEP 17 gibt es zwei Methoden wie man Lua-Code mit einem externen Editor bearbeiten kann.

a) Bei geschlossenem EEP:

Ab EEP 17 kann man den kompletten Code bei geschlossenem EEP mit einem externen Editor bearbeiten. Und man muss nach dem Start von EEP noch nicht einmal das Skript neu laden. Dies ist eine sehr sichere Methode einer Codebearbeitung mit einem externen Editor, da bei geschlossenem EEP immer ein definierter Anlagenzustand gewährleistet ist. Außerdem kann die Codebearbeitung mit beliebigen Editoren erfolgen, solange sie in ANSI kodieren.

b) Bei geöffnetem EEP:

Ab EEP 17 kann man auch einen externen Editor aus dem internen Lua-Skript-Editor heraus aufrufen. Dies funktioniert allerdings nur mit 64bit-Editoren.

Den Data-Slot-Teil des Skripts kann man weiterhin nicht bei geöffnetem EEP bearbeiten. Das gilt sowohl für den internen Skript-Editor als auch für den daraus aufgerufenen externen Editor.

Der Windows eigene Editor "Notepad" ist als externer Editor in EEP 17 voreingestellt (Bild 3.1). Da er in Windows integriert ist, genügt hier der Name "notepad" ohne Suffix und ohne weitere Pfadangabe im entsprechenden Feld des internen Editors. "Notepad" bietet aber keine Vorteile gegenüber dem internen Lua-Skript-Editor.

Bei jedem anderen Editor muss man den vollständigen Installationspfad des Editors inklusive Suffix in das entsprechende Feld im internen Editor eintragen (Bild 3.5).

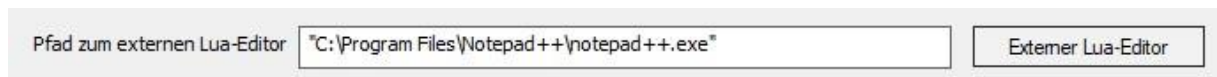


Bild 3.5

Man kann den Pfadeintrag jederzeit gegen den eines anderen Editors ändern oder den bestehenden Eintrag löschen, wenn man keinen externen Editor nutzen möchte. Der interne Lua-Skript-Editor ist weiterhin wie gehabt nutzbar.

Empfehlenswert als externer Editor ist "[Notepad++](#)". Er kann auf Lua als Sprache eingestellt werden. Die Tabulatorsprünge lassen sich individuell einstellen. Man kann **EEP-Funktionen** farblich und/oder im Schriftstil hervorheben und sogar mit einem Plug-in automatisch ergänzen lassen (siehe auch Bild 3.6).

Wenn man Notepad++ an dem von ihm vorgeschlagenem Ort installiert hat, gibt es 3 mögliche Pfadeingaben im Eingabefeld des EEP-Lua-Skript-Editors:

- "C:\Program Files\Notepad++\notepad++.exe" (mit Anführungszeichen)
- C:\Program Files\Notepad++\notepad++.exe (ohne Anführungszeichen)
- C:\Programme\Notepad++\notepad++.exe

"Programme" ist ein windowsinterner Hardlink auf den eigentlichen Speicherort "Program Files". Dieser Ordner kann aber bei einigen Benutzern als unsichtbar eingestellt sein.

Man kann aber auch im Windows Explorer sich den vollständigen Pfad in die Zwischenablage kopieren, in dem man die Datei notepad++.exe **bei gedrückter Shift-Taste mit der rechten Maustaste** anklickt (Die Shift-Taste ist wichtig, sonst erscheint "als Pfad kopieren" nicht!) und ihn dann aus der Zwischenablage in das Eingabefeld einfügen.

Und noch etwas ist **WICHTIG**: Nach der Pfad-Eintragung des externen Editors muss man **einmal die Schaltfläche "Skript neu laden" anklicken. Nur dann bleibt der Pfad in EEP gespeichert.**

Nach dem Klick auf die Schaltfläche "Externer Lua-Editor" wird dieser mit der temporären Lua-Datei *TempEdit.LUA* geöffnet. Während der Codebearbeitung kann man externe Programme (z.B. Browser, PDF-Reader, Windows Explorer etc.) öffnen, um etwas nachzusehen, daraus zu kopieren oder sonst was tun. Man kann sie aber muss sie nicht anschließend schließen. Man kann zwischendurch auch andere Dateien mit dem externen Editor bearbeiten. Hat man den Lua-Code fertig bearbeitet,

muss man im externen Editor die temporäre Lua-Datei **abspeichern**. Dies erfolgt automatisch im LUA-Ordner von EEP. Man sollte sie daher **nie unter** einem anderen Namen und/oder einem anderen Pfad **speichern**.

Abschließend **muss man** immer **den externen Editor schließen!!!**

Hat man während der Codebearbeitung andere Programme geöffnet, so kann es sein, dass diese ansichtsmäßig über EEP liegen. In einem solchen Fall muss man durch einen Klick in das Fenster des internen Editors, das Fenster von EEP oder auf das EEP-Icon in der Windows-Taskleiste wieder den Focus auf den internen Editor richten. Im Normalfall liegt er aber nach dem Schließen des externen Editors wieder beim internen Editor.

Aber **Achtung**, nach der Rückkehr in den internen Lua-Skript-Editor muss man nach einer Änderung des Codes unbedingt mit der gleichnamigen Schaltfläche (Bild 3.3) das "Skript neu laden".

Will man eine Bearbeitung im externen Editor abbrechen, geht man am besten folgendermaßen vor. Man schließt im externen Editor die temporäre Datei und beantwortet ggf. die Abfrage, ob man sie speichern möchte, mit "nein". Danach schließt man den externen Editor (WICHTIG) und bricht auch den internen Editor über das X in dessen Kopfzeile ab.

Was man nicht tun sollte ist, aus dem externen Editor (ohne diesen zu schließen) durch einen Klick in das Fenster des internen Editors, in das EEP-Fenster oder auf das EEP-Icon in der Windows-Taskleiste den Fokus wieder auf EEP setzen. In dem Fall färbt sich EEP milchig ein und entweder in der Kopfzeile des internen Editors oder in der des EEP-Fensters steht "keine Rückmeldung". Nun sollte man möglichst nicht auf eine der X-Schaltflächen in den Kopfzeilen klicken, denn dann erscheint die Windows-Warnung "EEP reagiert nicht" mit den Möglichkeiten zu schließen oder zu warten. Nun sollte man auf keinen Fall auf "Programm schließen" klicken, denn dann sind alle ggf. gemachten Änderungen verloren.

Es gibt aber (selbst noch an dieser Stelle) einen Ausweg, denn der externe Editor ist ja noch offen. Durch einen Klick in den externen Editor oder auf dessen Icon in der Windows-Taskleiste holt man ihn wieder in den Fokus. Nun kann man die temporäre Datei schließen oder auch nicht. Auf jeden Fall sollte man nun den externen Editor schließen. Danach erscheint EEP als wäre nichts gewesen.

3.3 Von EEP generiertes Lua-(Start)-Skript

Mit jeder neuen Anlage generiert EEP eine Lua-Datei mit dem Namen der Anlage und der Endung .lua. **Bis EEP 17** enthält diese Datei folgendes Lua-Skript (Bild 3.6 nächste Seite).

In Zeile 1 des Skripts wird die Variable $\mathbb{I}$ mit 0 initialisiert. $\mathbb{I}$ ist eine globale Variable, d.h. sie steht im gesamten Skript in allen Funktionen zur Verfügung. Auf die Bedeutung von $\mathbb{I}$ wird nachstehend genauer eingegangen.

In Zeile 2 werden mit dem Aufruf der Funktion `clearlog()` ^[1] alle Einträge im EEP-Ereignisfenster (siehe [Kapitel 3.4](#)) gelöscht.

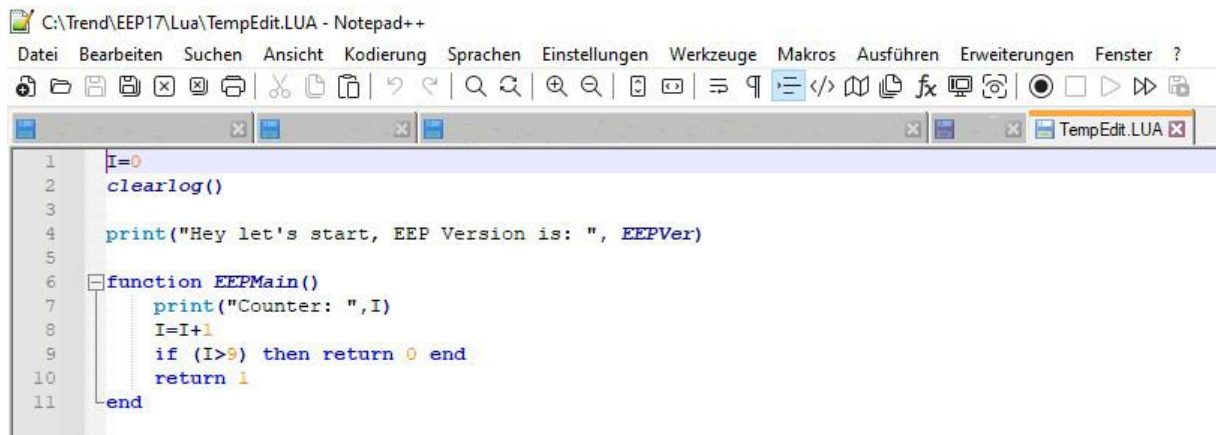


Bild 3.6

Die `print`-Funktion^[1] in Zeile 4 gibt einen ersten neuen Text im EEP-Ereignisfenster aus. Hierbei muss alles, was ausgegeben werden soll, zwischen den runden Klammern (...) hinter `print` stehen. Reiner Text muss wie bei einer wörtlichen Rede in einem Buch zwischen Anführungszeichen"..." geschrieben werden. "Hey let's start, EEP Version is: " (zu Deutsch: "Hey, fangen wir an, die EEP-Version ist: ") kann natürlich gegen einen eigenen Text ausgetauscht werden. `EEPVer`^[2] ist eine spezifische EEP-Lua-Variable, die die aktuell geöffnete EEP-Version zurückgibt. EEP verbindet den Eigentext mit dem Wert der Variablen zu einem Gesamttext: Hey let's start, EEP Version is: 17.

Das Herzstück jedes Lua-Skripts in EEP ist die Funktion `EEPMain()`^[1], hier in Zeile 6. Wie jede Funktionsdefinition beginnt diese mit dem englischen Schlüsselwort `function` (zu Deutsch: Funktion). Dahinter folgt der Name der Funktion, hier `EEPMain`. Hinter dem Funktionsnamen stehen immer die beiden sich schließenden runden Klammern (). Bei vielen Funktionen können sog. Parameter übergeben werden, die dann durch Kommata getrennt zwischen die runden Klammern geschrieben werden, wie z.B. oben bei der `print`-Funktion. An die `EEPMain()` können keine Parameter übergeben werden. Eine Funktionsdefinition endet immer mit dem englischen Schlüsselwort `end` (zu Deutsch: Ende), hier in Zeile 11.

Die `EEPMain()` wird in der Regel 5 x pro Sekunde – also alle 0,2 Sekunden – von EEP aufgerufen. Was während eines Aufrufs der `EEPMain()` durch EEP passieren soll, muss zwischen den Zeilen function EEPMain() und end stehen.

In Zeile 7 wird über einen `print`-Befehl nicht nur der aktuelle Wert der Variablen `I` ausgegeben, sondern auch das davor gesetzte Wort "Counter: " (zu Deutsch: Zähler). Mit ihm wird klar, was `I` ist. `I` zählt die Durchläufe der Funktion `EEPMain()`. Hierzu wird `I` in Zeile 8 bei jedem Aufruf der `EEPMain()` durch EEP um 1 erhöht.

Somit kann man die Variable `I` nutzen, um z.B. beim x-ten Durchlauf der `EEPMain()` etwas passieren zu lassen. Im von den EEP-Programmierern vorgefertigten Skript geschieht das in Zeile 9: if (I>9) then (zu Deutsch: *wenn (I größer 9) ist, dann ...*). Doch zu Beginn ist das ja noch nicht der Fall. Beim 1. Aufruf der `EEPMain()` ist `I` ja noch 0 und Lua springt sofort zur nächsten Zeile 10.

`return` (zu Deutsch: zurück/Rückgabe) ist ein weiteres Lua-Schlüsselwort. Mit `return` wird der dahinterstehende Wert an EEP übergeben und die Funktion beendet. Ist dieser Rückgabewert gleich 1, so bedeutet das für EEP, dass EEP weiterhin alle 0,2 Sekunden die `EEPMain()` aufruft.

D.h., dass mit jedem Aufruf der `EEPMain()` `I` um 1 erhöht wird. Nach 2 Sekunden ist `I` dann aber größer 9, Zeile 9 tritt in Kraft und der Wert 0 wird an EEP zurückgegeben. Der Rückgabewert 0 bedeutet für EEP, die `EEPMain()` ab sofort nicht mehr aufzurufen. Alle anderen eventuellen Lua-Funktionen, die z.B. durch Kontaktpunkte aufgerufen werden (siehe [Kapitel 5](#)) werden weiterhin abgearbeitet, aber nicht mehr die `EEPMain()`.

Möchte man also den zyklischen Aufruf der `EEPMain()` in seinem Lua-Skript nutzen, was in der Regel der Fall ist, so muss man als erste Maßnahme Zeile 9 aus dem vorgegebenen Skript löschen!

In dem Fall wird aber dauernd (also alle 0,2 Sekunden) der Wert von `I` im Ereignisfenster ausgegeben. Hierdurch kann man sehr schnell andere wichtige Ausgaben übersehen. Deshalb empfiehlt es sich, entweder auch Zeile 7 zu löschen oder wenigstens direkt vor `print 2` Minuszeichen zu setzen: `--print("Counter: ", I)`. 2 Minuszeichen interpretiert Lua als sog. Kommentarzeichen und ignoriert alles was in der Zeile hinter den beiden Minuszeichen steht. Möchte man z.B. für eine Kontrolle doch einmal den Wert von `I` angezeigt haben, so muss man nur die beiden Minuszeichen wieder löschen.

Mit dem Plug-in1 zu EEP 17 wurde aus dem in Bild 3.6 gezeigten Startcode sowohl die Willkommens-`print`-Zeile (4) als auch alles was zum Zähler `I` gehört (Zeilen 1 und 7 – 9) entfernt. Damit wurde der Startcode auf das unbedingt Notwendige reduziert:

```
1. clearlog()
2.
3. function EEPMain()
4.     return 1
5. end
```

In den meisten Fällen ist es egal, wo man andere Funktionen in das Skript einträgt, da es nach jedem neuen Laden 1 x durch EEP komplett abgearbeitet wird. Es kann jedoch passieren, dass bestimmte Funktionen nur dann ausgeführt werden, wenn die Funktionen vor der `EEPMain()` im Skript stehen. Aus diesem Grund ist es ratsam, die `EEPMain()` immer ans Ende der `.lua`-Datei mit dem Anlagennamen zu stellen.

[1] Siehe Kap. 2 bei den [EEP-spezifische Lua-Variablen und -Funktionen](#) oder [hier im Online Lua-Handbuch](#).

[2] Siehe Kap. 1 bei den [EEP-spezifische Lua-Variablen und -Funktionen](#) oder [hier im Online Lua-Handbuch](#).

3.4 Ausgabe von Meldungen im EEP-Ereignisfenster

Lua-Fehlermeldungen oder mit der Lua-Funktion `print()` im Skript erstellte eigene Meldungen werden im EEP Ereignis-Fenster ausgegeben (Bild 3.7).

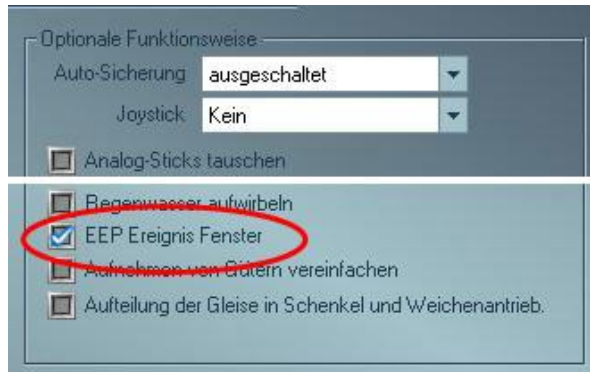


Bild 3.8

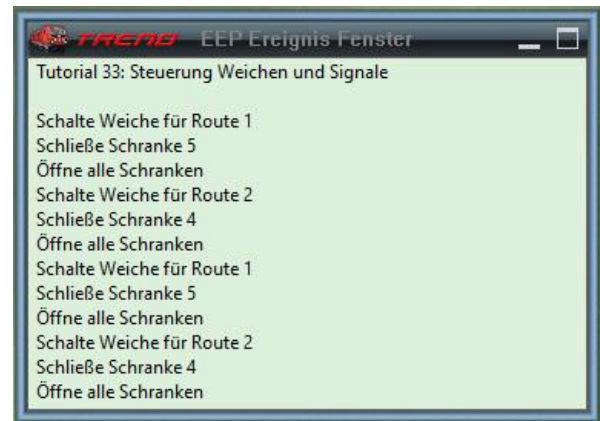


Bild 3.7

Damit das Ereignisfenster auf dem Bildschirm dargestellt wird, muss es in den Programmeigenschaften (erreichbar über den Menüpunkt "Datei") im Bereich

"Optionale Funktionsweise" aktiviert werden (siehe Bild 3.8).



Bild 3.9

Neben den Lua-Fehlermeldungen und eigenen Meldungen können auch Statusmeldungen von Signal-, Weichen-, Fahrzeug- und Kontaktpunkt ereignissen im Ereignisfenster angezeigt werden, wenn dies entsprechend im Skript-Editor aktiviert wird (siehe Bild 3.1 links unten und Bild 3.9).

Mit dem Plug-in 2 zu EEP 17.2 hat sich die Anzahl der letzten im Ereignisfenster sichtbaren Ausgabezeilen von 500 auf 1.024 Zeilen erhöht, so dass man über einen mehr als doppelt so großen Bereich die Ausgaben zurückverfolgen kann.

Außerdem ist es nun möglich diese Ausgaben über das Fenstermenü des Ereignisfensters zu speichern (siehe Bild 3.10).

Die gespeicherte Log-Datei heißt `MeineAnlage.eep.log` und befindet sich im selben Ordner, in dem sich auch die Datei `MeineAnlage.anl3` befindet. Dabei entspricht natürlich "MeineAnlage" dem Namen, unter dem die Anlage zuletzt gespeichert wurde.

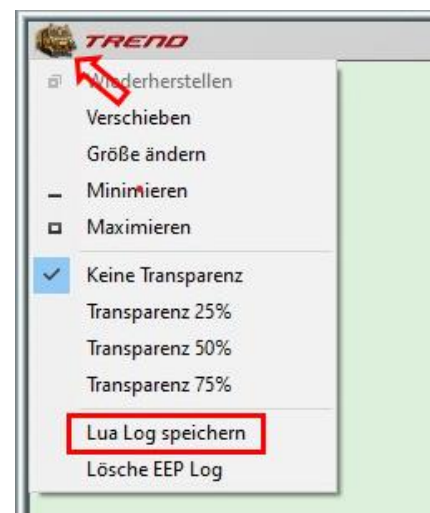


Bild 3.10

4 Aufruf von Lua-Funktionen in Kontaktpunkten

Im Lua-Skript definierte Funktionen können prinzipiell in jedem Kontaktpunkt aufgerufen werden (siehe auch Bild 5.6). Hierzu trägt man in dessen Objekteigenschaften den Funktionsnamen ohne runde Klammern und ohne Parameter ein (rote Markierung in Bild 4.1). Will man allerdings "nur" eine Lua-Funktion aufrufen, so empfiehlt sich hierfür ein Sound-Kontakt, wie hier abgebildet.

Wichtig: Bevor man aber eine Lua-Funktion in einen Kontaktpunkt einträgt, muss diese im Skript eingetragen (wobei ggf. "function Funktionsname" innerhalb eines Kommentars ausreicht) und nach dem Laden des Skripts dieses einmal (durch einen Wechsel in den 3D-Modus) ausgeführt worden sein. Ansonsten erhält man beim Klick auf OK eine Fehlermeldung.

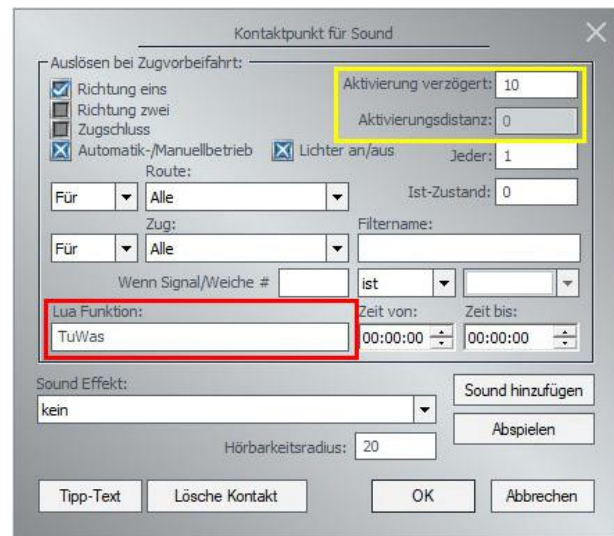


Bild 4.1

Ab EEP 17 sind Kontaktpunkte, in denen eine Lua-Funktion aufgerufen wird, sowohl in der 3D- als auch der 2D-Ansicht entsprechend gekennzeichnet (siehe Bild 4.2).

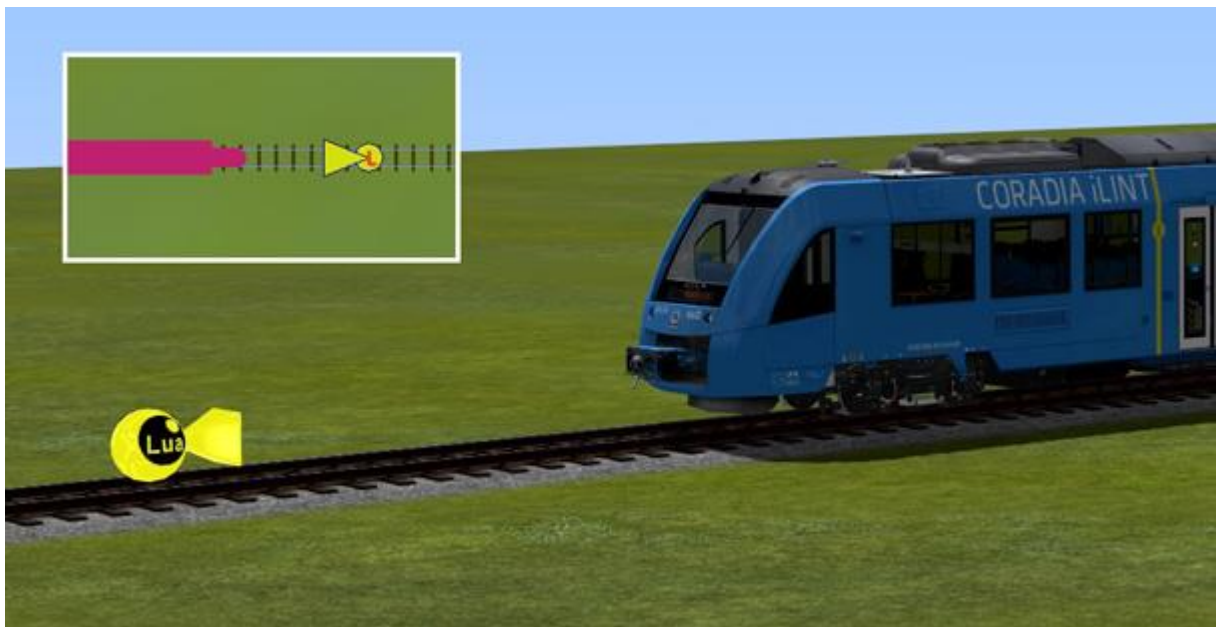
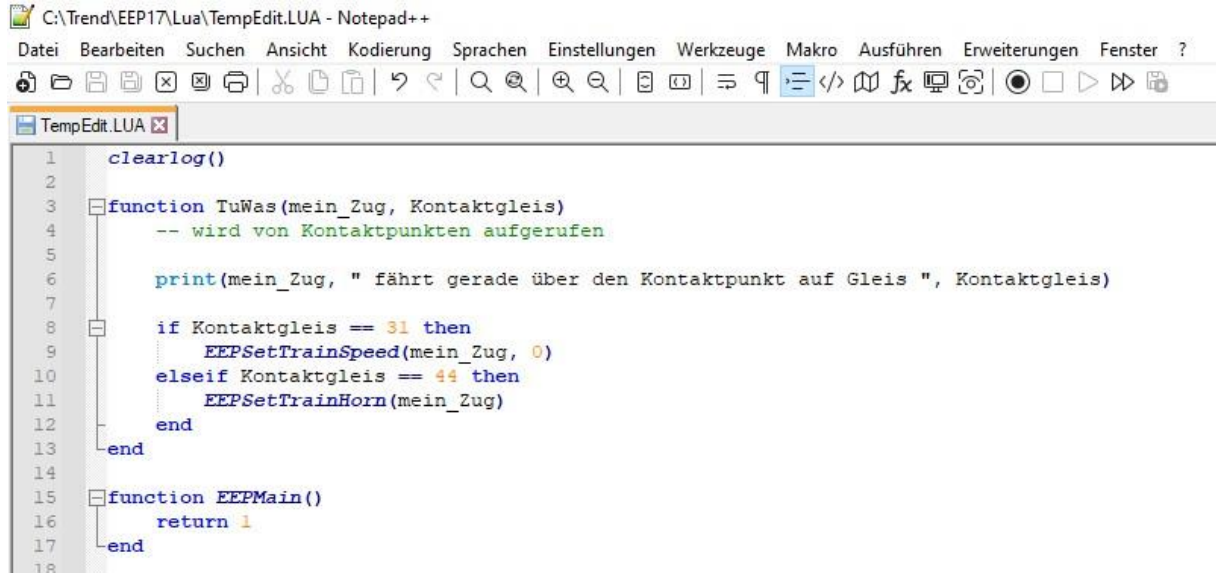


Bild 4.2

Seit EEP11.2 Plug-in 2 übergibt EEP den Namen des Fahrzeugverbandes, der einen Kontaktpunkt überfährt, als Funktionsparameter an die im Kontaktpunkt eingetragene Funktion. Diesem Parameter kann man einen beliebigen Namen geben, z.B. *mein_Zug*.

Ab EEP 16.3 Plug-in 3 übergibt EEP zusätzlich die ID des Gleises (der Straße, des Wasserweges etc.) auf dem der Kontaktpunkt liegt als zweiten Parameter an die im

Kontaktpunkt eingetragene Funktion. Auch dieser Parameter ist frei wählbar, z.B. *Kontaktgleis*. Durch diesen 2. Parameter kann dieselbe Lua-Funktion für verschiedene Zwecke verwendet werden, je nachdem auf welchem Gleisstück sich der Kontaktpunkt befindet (siehe Bild 4.3).



```
1 clearlog()
2
3 function TuWas(mein_Zug, Kontaktgleis)
4     -- wird von Kontaktpunkten aufgerufen
5
6     print(mein_Zug, " fährt gerade über den Kontaktpunkt auf Gleis ", Kontaktgleis)
7
8     if Kontaktgleis == 31 then
9         EEPSetTrainSpeed(mein_Zug, 0)
10    elseif Kontaktgleis == 44 then
11        EEPSetTrainHorn(mein_Zug)
12    end
13 end
14
15 function EEPMain()
16     return 1
17 end
18
```

Bild 4.3

Die beiden Parameter werden nur in der genannten Reihenfolge in der Funktion im Skript eingetragen. In die Objekteigenschaften des Kontaktpunktes darf auch hier nur der Funktionsname eingetragen werden.

Die beiden von EEP an Lua übergebenen Parameter müssen immer in der Reihenfolge 1. Zugname, 2. Gleis-ID erfolgen. Wenn man die Gleis-ID nicht benötigt, kann man sie weglassen. Aber wenn man nur die Gleis-ID benötigt, muss eine Variable für den Zugnamen eingetragen sein. Für eine Variable, die nicht benötigt wird aber vorhanden sein muss, hat sich der Unterstrich "_" bei vielen Lua-Nutzern als Variablenname eingebürgert. Benötigt man auch den Zugnamen nicht, ist die Eingabe als Parameter nicht erforderlich.

Die Möglichkeit andere oder weitere Parameter direkt in einem Kontaktpunkt einzutragen, bietet das Lua-Modul [BetterContacts](#) von Benjamin Hogl (BH2). Download, Beschreibung und Einbindungsanweisung ist dem vorgenannten Link zu entnehmen.

5 Lua-Wizard

Der in EEP enthaltene Lua-Wizard liefert die Möglichkeit Lua-Skripte bzw. EEP-spezifische Funktionen auf einfache Art zu erstellen. Den Wizard öffnet man über das Menü Extras -> Wizard. Nach dem Start legt er sich über das Menü und die Werkzeugleiste von EEP, ist aber frei beweglich.



Bild 5.1

Der Wizard besteht zunächst neben einem Kurzmenü nur aus einer Auswahlleiste für die zu erstellenden diversen Lua-Aufgaben (Bild 5.1). Die Schaltfläche am Ende der Leiste schließt den Wizard. Nach Betätigen einer der mit ersten vier Schaltflächen erscheint grundsätzlich zuerst ein Fenster mit der Frage, ob man die Anlage vor dem Aufruf des Wizards gespeichert hat. Hierzu wird dringend geraten. Erst nach Bejahung der Frage erscheinen nacheinander weitere Fenster, die einen durch den Wizard führen.



Wichtig zu wissen: Der Wizard ist ein externes Programm zu dessen Nutzung Sie die Grundressourcen über den Menüpunkt "Extras – Ressourcen-Extraktor" unpacken müssen. Dies nimmt einen zusätzlichen Speicherbedarf von 10,7 GB in Anspruch.

5.1 Erstellung eines Lua-Skripts für einen EEP-Fahrplan



Durch einen Klick auf die erste Schaltfläche im Bild 5.1 erscheint das Hauptfenster für die Fahrplanerstellung (Bild 5.2).

Durch Anklicken der Schaltfläche "Hinzufügen" erscheint ein Detailfenster zur Eingabe der entsprechenden Daten (Bild 5.3).

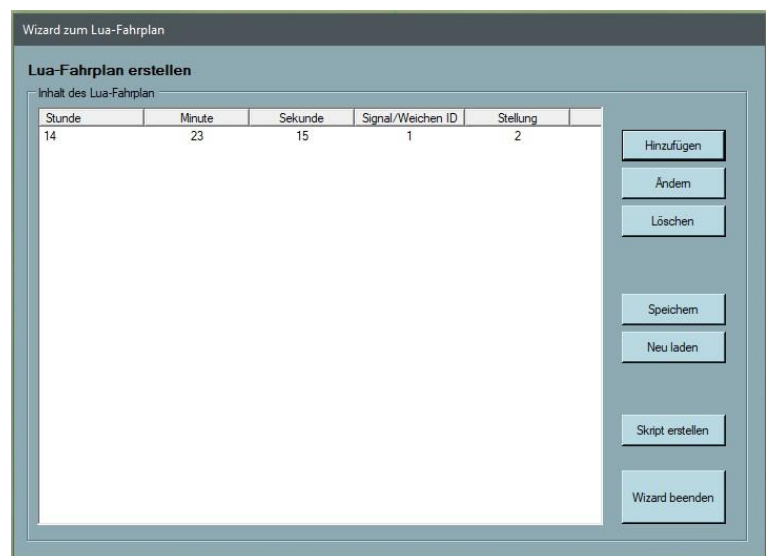


Bild 5.2

Dort klickt man nach der Dateneingabe auf die Schaltfläche "Angaben übernehmen".



Bild 5.3

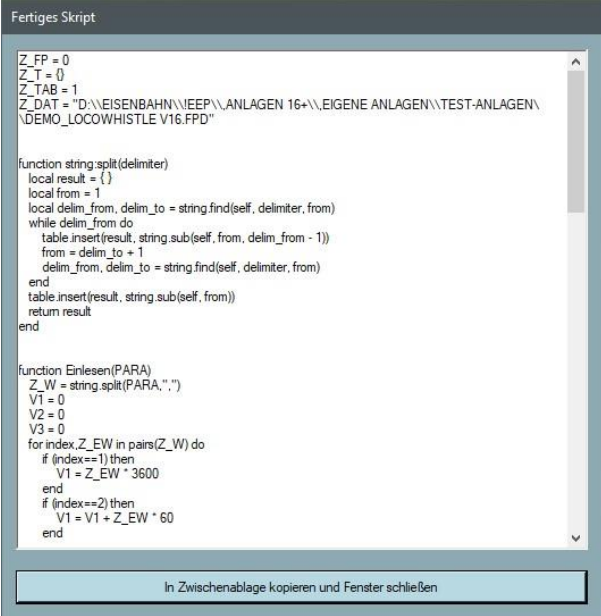
Um eine Datensatz zu ändern, markiert man ihn zunächst durch Anklicken im Hauptfenster (Bild 5.2) und klickt dann auf die Schaltfläche "Ändern". Es erscheint hierfür wieder das Detailfenster (Bild 5.3).

Um einen Datensatz zu löschen, verfährt man ebenso: Datensatz markieren und Schaltfläche "Löschen" anklicken.

Sind alle Datensätze eingegeben, klickt man auf die Schaltfläche "Skript erstellen". Es erscheint ein neues Fenster (Bild 5.4) mit dem fertigen Skript inklusive des Lua-Codes, der bereits vorher in der Anlage gespeichert war. Dies kopiert man in die Zwischenablage mit einem Klick auf die gleichnamige Schaltfläche.

Nachdem sich das Skript-Fenster geschlossen hat, wird man aufgefordert im Lua-Skript-Editor den vorhandenen Text durch den soeben kopierten zu ersetzen und das Skript neu zu laden.

Hierzu öffnet man den Lua-Skript-Editor (siehe Kapitel 3), klickt einmal in den vorhandenen Text und markiert ihn durch gleichzeitiges Drücken der Tasten **Strg** + **A**. Danach ersetzt man den markierten Text durch das in der Zwischenablage hinterlegte neue Lua-Skript durch gleichzeitiges Drücken entweder der Tasten **Strg** + **V** oder der Tasten **Shift** + **Einf** (gleichwertige Möglichkeiten). Zum Schluss überträgt man das Skript mit der Schaltfläche "Skript neu laden" (Bild 3.3) an EEP.



```

Z_FP = 0
Z_T = 0
Z_TAB = 1
Z_DAT = "D:\\EISENBAHN\\EEP\\ANLAGEN 16+\\EIGENE ANLAGEN\\TEST-ANLAGEN\\
\\DEMO_LOCOWHISTLE V16.FPD"

function string.split(delimiter)
    local result = {}
    local from = 1
    local delim_from, delim_to = string.find(self, delimiter, from)
    while delim_from do
        table.insert(result, string.sub(self, from, delim_from - 1))
        from = delim_to + 1
        delim_from, delim_to = string.find(self, delimiter, from)
    end
    table.insert(result, string.sub(self, from,))
    return result
end

function Einlesen(PARA)
    Z_W = string.split(PARA, ",")
    V1 = 0
    V2 = 0
    V3 = 0
    for index, Z_EW in pairs(Z_W) do
        if (index==1) then
            V1 = Z_EW * 3600
        end
        if (index==2) then
            V1 = V1 + Z_EW * 60
        end
    end
end
    
```

In Zwischenablage kopieren und Fenster schließen

Bild 5.4

Um nach einem späteren erneuten Aufruf der Anlage gegebenenfalls den EEP-Fahrplan zu ändern, ruft man einfach erneut den Lua-Wizard wieder auf.



Hinweis: Einen Fahrplan, der auch Züge und Routen berücksichtigt und kein Lua-Skript benötigt, kann im Menü "Routen" unter der Option "Fahrplan" erstellt werden (siehe hierzu Kapitel 7.3.2 im [EEP-Handbuch](#)).

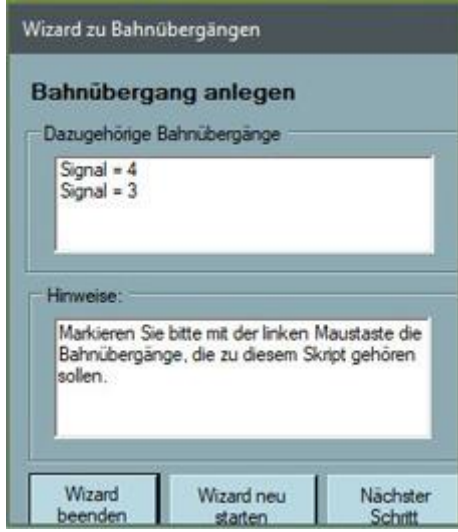
5.2 Erstellung eines Lua-Skripts für einen beschränkten Bahnübergang



Durch einen Klick auf die 2. Schaltfläche im Bild 5.1 wird man nach der Speicherabfrage gebeten, in den 2D-Modus "Signalssystem" zu wechseln. Nach der Bestätigung mit OK erscheint das Eingabefenster zum Bahnübergang (Bild 5.5). (Achtung: Es erfolgt keine Plausibilitätsprüfung, jedoch ist der nächste Schritt nur im 2D-Modus "Signalssystem" möglich.)

Folgen Sie den Angaben im Hinweisenfenster und klicken Sie die dem Bahnübergang zugehörigen Schranken nacheinander an. Der Wizard ermittelt selbständig die Signal-ID der Schranken und trägt sie in das obere Fenster ein.

Sind alle Schranken eingegeben, schließt man das



Wizard zu Bahnübergängen

Bahnübergang anlegen

Dazugehörige Bahnübergänge:

Signal = 4
Signal = 3

Hinweise:

Markieren Sie bitte mit der linken Maustaste die Bahnübergänge, die zu diesem Skript gehören sollen.

Wizard beenden Wizard neu starten Nächster Schritt

Bild 5.5

Fenster über die Schaltfläche "Nächster Schritt". Es öffnet sich das Fenster für das fertige Skript (wie Bild 5.4 nur mit dem Skript für den Bahnübergang). Auch hier enthält das Skript den Code, der bereits vorher in der Anlage gespeichert war. Über die gleichnamige Schaltfläche kopiert man alles in die Zwischenablage.

Danach wird man aufgefordert im Lua-Skript-Editor den vorhandenen Text durch den soeben kopierten zu ersetzen und das Skript neu zu laden (Vorgehensweise siehe oben "Erstellung eines Lua-Skripts für einen EEP-Fahrplan".)

Lua-mäßig ist der Bahnübergang damit abgeschlossen. Damit sich die Schranken aber schließen und wieder öffnen, müssen einige Kontaktpunkte gesetzt werden.

Zunächst setzt man einen Zug-Kontaktpunkt (KP) vor den Bahnübergang in einer Entfernung, dass die Schranken geschlossen sind, bevor sie vom schnellsten Zug auf der Anlage erreicht werden. Im KP setzt man die Häkchen zur Fahrtrichtung (rote Markierung in Bild 5.6) so, dass das Dreieck des KPs in Richtung des Bahnübergangs zeigt. Unter Lua-Funktion (gelbe Markierung) trägt man den Funktionsnamen zum Schließen der Schranken aus dem Lua-Skript `BUE_1_SCHLIESSEN` ohne die runden Klammern ein.

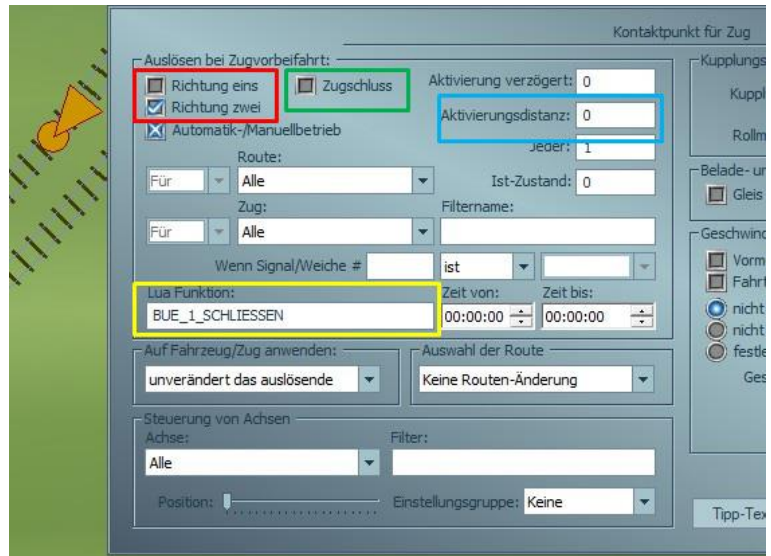


Bild 5.6

Den nächsten Zug-Kontaktpunkt setzt man auf demselben Gleisstrang direkt hinter dem Bahnübergang. Hier setzt man im KP die Häkchen zur Fahrtrichtung so, dass das Dreieck des KPs von dem Bahnübergang weg zeigt. Der KP muss hier vom Zugschluss durch das entsprechend gesetzte Häkchen (grüne Markierung) ausgelöst werden. Unter Lua-Funktion trägt man den Funktionsnamen zum Öffnen der Schranken aus dem Lua-Skript `BUE_1_OEFFNEN` ohne die runden Klammern ein.

Für jeden weiteren Gleisstrang zwischen den Schranken setzt man in gleicher Weise KPs.

Wird ein Gleisstrang von beiden Richtungen befahren, so sind Schließen-Kps wie oben beschrieben auf beiden Seiten des Bahnübergangs zu erstellen. Zum Öffnen genügt ein KP und zwar genau in der Mitte des Bahnübergangs. Im KP sind beide Häkchen für die Fahrtrichtung (rote Markierung in Bild 5.6) zu setzen. Auch hier muss der KP durch das entsprechende Häkchen durch den Zugschluss ausgelöst werden. Damit der KP aber wirklich erst nach dem Passieren des Bahnübergangs durch den Zug ausgelöst wird, setzt man eine Aktivierungsabstand von 15 Metern (blaue Markierung). Letztlich ist unter Lua-Funktion `BUE_1_OEFFNEN` einzutragen.

Befinden sich auf einer Anlage mehrere Bahnübergänge, so ruft man für jeden den

Wizard auf. Der Wizard erkennt, dass bereits ein Skript für einen Bahnübergang existiert und erhöht im nächsten Skript entsprechend die ID des Bahnübergangs.

Wenn ein Bahnübergang zur Zufriedenheit funktioniert, könnte eventuell die dauernde Ausgabe der Meldungen "Bahnübergang 1 schließen" und "Bahnübergang 1 wird geöffnet" nerven. In dem Fall öffnet man den Lua-Skript-Editor und setzt vor die *print*-Anweisungen 2 Minus-Zeichen, also `-- print("Bahnübergang ...")`.

5.3 Erstellung einer EEP-spezifischen Kamera-Funktion



Durch einen Klick auf die dritte Schaltfläche im Bild 5.1 kann eine EEP-spezifische Kamera-Funktion erstellt werden. Standardmäßig werden die "Vorhandenen Kameras" in einer Auswahlliste angezeigt. (Bild 5.7), wenn mindestens eine statische Kamera in der Anlage gespeichert ist. Markieren Sie die gewünschte in der Auswahlliste. Danach erscheint der hierfür erstellte Befehl direkt im Anzeigefeld "Erstellter Befehl". Klicken Sie dann auf die Schaltfläche Befehl in die Zwischenablage kopieren, um sie danach in das Lua-Skript einfügen zu können (siehe unten).

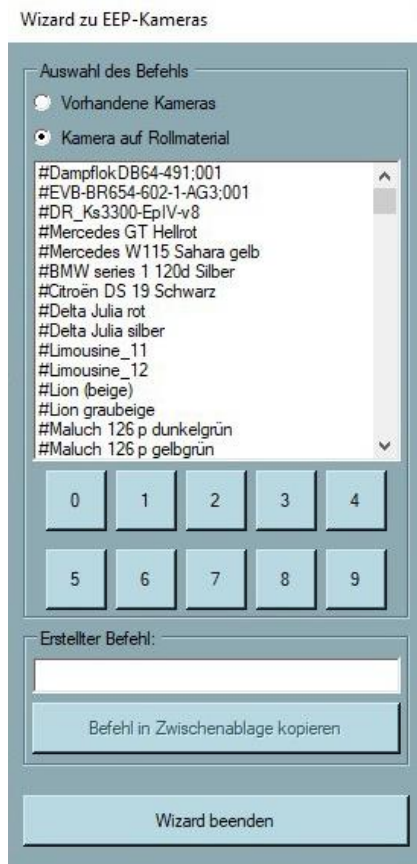


Bild 5.8

Fahrzeugverbandes,

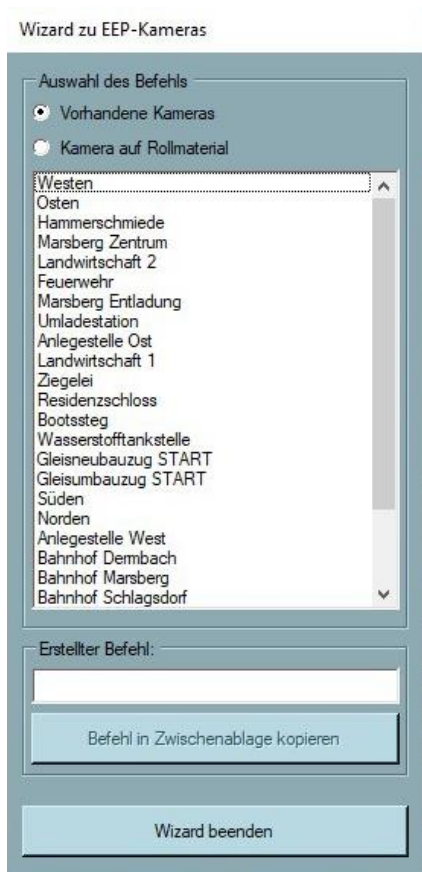


Bild 5.7

Um eine Lua-Funktion für eine "Kamera auf Rollmaterial" zu erzeugen, klicken Sie auf das gleichnamige Optionsfeld unter "Auswahl der Befehle". Dann erscheinen alle Rollmaterialien, die sich auf der Anlage befinden in der Auswahlliste. (Bild 5.8).

Wählen Sie das gewünschte Rollmaterial durch Anklicken aus.

Unter der Auswahlliste befinden sich 10 Schaltflächen mit den Ziffern 0 bis 9. Sie entsprechen den numerischen Tasten für die Kameraauswahl:

- 0 = Perspektive der "alten Kabine" (ggf. andere Perspektive als bei 8)
- 1 = direkt auf die linke Seite des Fahrzeugverbandes,
- 2 = direkt auf die rechte Seite des Fahrzeugverbandes,
- 3 = seitlich von oben auf die linke Seite des Fahrzeugverbandes,
- 4 = seitlich von oben auf die rechte Seite des

- 5 = von der Front des Fahrzeugverbandes in Fahrtrichtung,
- 6 = von vorne auf die Front des Fahrzeugverbandes,
- 7 = aktiviert diejenige automatische Kamera, die dem ausgewählten Fahrzeugverband am nächsten steht,
- 8 = aus dem Führerstand (ggf. andere Perspektive als bei 0),
- 9 = oberhalb des Fahrzeugverbandes in oder gegen die Fahrtrichtung bzw. wenn vorhanden die vom Konstrukteur (außerhalb des Führstands) vorgegebene Mitfahrkamera bzw. wenn definiert die letzte vom Benutzer mit `EEPRollingstockSetUserCamera()` definierte Kameraposition

Bei diesen Fahrzeugverfolgerkameras wird zwar die erstellte Funktion im untersten Anzeigefeld unvollständig angezeigt, aber mit Klick auf die entsprechende Schaltfläche korrekt in die Zwischenablage übertragen.

Wie oben beschrieben kann die generierte Funktion aus der Zwischenablage in das Lua-Skript eingefügt und danach das Skript neu geladen werden.

Zusätzlich notwendige Arbeiten

Im Gegensatz zu den Anwendungsmöglichkeiten "[Fahrplan](#)" und "[Bahnübergang](#)", bei denen fertige Lua-Skripte erzeugt wurden, ist das sowohl bei diesen Kamera-Funktionen als auch bei den im nächsten Kapitel beschriebenen "[EEP-spezifischen Funktionen](#)" nicht der Fall. Hier wird jeweils nur der Funktionscode mit den Parametern und Rückgabewerten erzeugt. Diese können zwar auch über die Zwischenablage im Skript-Editor (siehe [Kapitel 3](#)) in den dort vorhandenen Text kopiert werden, um sie aber während des Anlagenbetriebs zur Anwendung kommen zu lassen, sind mindestens Lua-Grundkenntnisse zur Eingabe von umgebendem Code notwendig.

Auf alle Konstellationen hier einzugehen, ist nicht möglich. Jedoch sei kurz aufgezeigt, welche Eintragungen notwendig sind, um eine solche EEP-spezifische Funktion über einen Kontaktpunkt aufzurufen.

- Kopieren Sie den im Wizard erzeugten Code über die Zwischenablage z.B. ans Ende des vorhandenen Codes
- Fügen Sie davor folgende Zeile ein: `function TuWasBestimmtes()`. Hierbei ist *TuWasBestimmtes* ein Synonym für einen von Ihnen gewählten Ausdruck für das, was die EEP-Funktion tun soll.
- Fügen Sie hinter der Zeile mit der EEP-Funktion eine Zeile mit dem Wort `end` ein.

Damit könnte ein Code im Skript wie folgt aussehen:

```
function Kamera_Uebersicht()  
    hResult = EEPSetCamera(0, "Uebersicht")  
end
```

Kamera_Uebersicht wäre dabei das Synonym für *TuWasBestimmtes*. Die rote Zeile ist der vom Wizard erzeugte Code.

Den Funktionsnamen - im Beispiel also *Kamera_Uebersicht* - fügt man ohne die

Klammern in den auslösenden Kontaktpunkt unter Lua-Funktion ein (siehe Bild 5.6 gelbe Markierung).

Solange man sich in einem Wizard-Bereich - z.B. für Immobilien-Funktionen - befindet, kann man zwischen dem Lua-Skript-Editor und dem Wizard hin und her bewegen, um z.B. mehrere Funktionen für dieselbe Immobilie oder für andere Immobilien zu erstellen.

Sobald man aber von einem Wizard-Bereich in einen anderen wechseln will, muss man vorher im Lua-Skript-Editor das bis dahin erstellte Skript über die Schaltfläche **"Skript neu laden"** an EEP übertragen und danach den Skript-Editor schließen. Der Wizard selbst kann derweil geöffnet bleiben.

5.4 Erstellung von Lua-Codes zu EEP-spezifischen Funktionen



Durch einen Klick auf die vierte Schaltfläche im Bild 5.1 erscheint nach der Speicherungsabfrage das Wizardfenster für EEP-Lua-Funktionen (Bild 5.9).

Im großen Auswahlfeld auf der linken Seite sind baumartig die Funktionen aufgelistet, von denen der Wizard eine Lua-Funktion erstellen kann.

Wenn Sie im Eingabefeld darüber einen oder mehrere Buchstaben eingeben, werden in der Auswahlliste die Funktionen und/oder Rubriküberschriften, die diese Buchstabenkombinationsmöglichkeit enthalten farblich markiert.

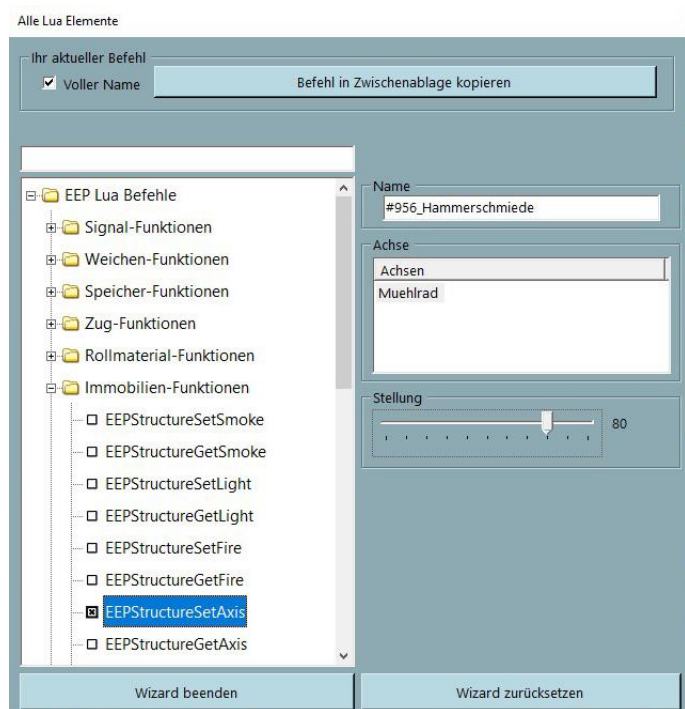


Bild 5.9

Klicken Sie auf eine der Lua-Funktionen, so erscheinen im rechten Teil des Wizard-Fensters (Bild 5.9) Felder für die Funktionsparameter. Dies können Eingabefelder, Auswahlfelder, Checkboxes oder auch Schieberregler sein. Geben bzw. stellen Sie darin die gewünschten Werte ein. Am oberen Bildschirmrand wird Ihnen die ausgewählte Funktion mit ihren Rückgabewerten und Parametern angezeigt (Bild 5.10), wobei letztere parallel zu jeder Eingabe ergänzt werden.

```
Ok = EEPStructureSetAxis("#956_Hammerschmiede", "Muehlrad", 80)
```

Bild 5.10

Um eine Funktion zu erstellen, müssen Sie sich entweder im 3D-Objekt-Editiermodus befinden oder im 2D-Fenster den entsprechenden Editor einschalten, d.h. z.B. für

Immobilien den Immobilien-Editor, für Signale den Signal-Editor. Klicken Sie nun das gewünschte Objekt an, wird dessen Lua-Name automatisch in das Namensfeld übernommen.

Funktionen für "Züge" und Rollmaterialien können nur im 3D-Objekt-Editiermodus erstellt werden. Klickt man ein Rollmaterial in einem Fahrzeugverband an, erscheint zunächst automatisch der Lua-Name (mit #-Zeichen) des Fahrzeugverbandes. Wählt man dann eine Rollmaterialfunktion aus, wird der Name des Rollmaterials angezeigt.

Der Wizard erkennt ebenso automatisch, welche EEP-spezifischen Funktionen erstellt werden können.

Je nach Auswahl der gewünschten Funktion ändert sich das Feld darunter. Wenn Achsen vorhanden sind, muss diese ausgewählt werden, auch wenn nur eine existiert.

Wie oben beschrieben kann die generierte Funktion über die Zwischenablage in das Lua-Skript eingefügt und danach das Skript neu geladen werden.