Schienenzeppelin von Märklin mit scriptbarem Sounddecoder von Zimo

ZIMO MACHTE ES MÖGLICH

Im November 2017 startete das Projekt „Märklin Schienenzeppelin“. Ziel war es, das Fahrzeug mit einem originalen Propellersound und einem passenden Antrieb für den Propeller auszustatten. Um eine vorbildnahe komplexe Ablaufsteuerung hinzubekommen, wurde einer der ersten scriptfähigen Zimo-645-Sounddecoder eingesetzt.
Einen Artikel über den mechanischen Umbau finden Sie im Eisenbahn-Journal 5/2019.

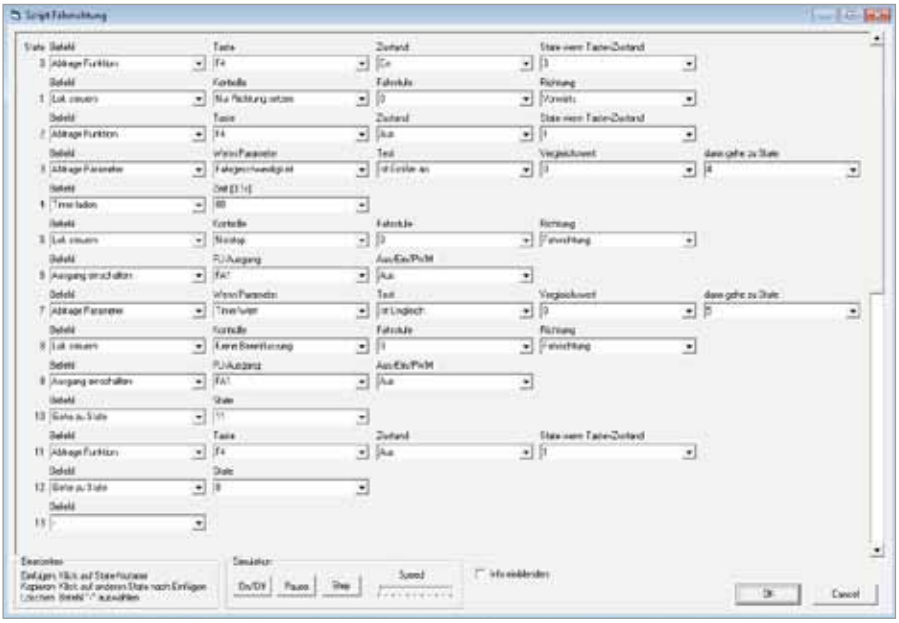
Eine Internetrecherche zu Beginn des Projekts ergab sehr schnell, dass es keinen Sounddecoder mit Propellersound zu kaufen gab. Ein Dieselmotorsound, wie ihn einige Umbauvorstellungen im Internet verwenden, kam nicht in Frage, denn die Geräusche des Propellermotors waren sehr markant und daher für den Umbau zwingend erforderlich.

Für dieses Projekt konnte ich ein paar Mitarbeiter bei Zimo begeistern. Schnell war klar, dass die geforderte Funktionalität mit einem normalen Sounddecoder kaum hinzubekommen wäre. Die geplanten scriptfähigen Decoder kamen hier gerade recht. Dieses Feature wurde dann im Dezember 2017 für ein paar Erlkönige aus der 645-Sounddecoderserie eingerichtet. Eine erste vielversprechende Hörprobe einer Sounddatei zeigte mir, dass wir auf einem guten Weg waren. Genaugenommen waren es zwei Soundprojekte: eines für den Propellersound und ein zweites für den Rangierbetrieb mit Elektromotorsound. Als auch die Programmersoftware des MXULF die Scriptbearbeitung möglich machte, konnten wir richtig loslegen.

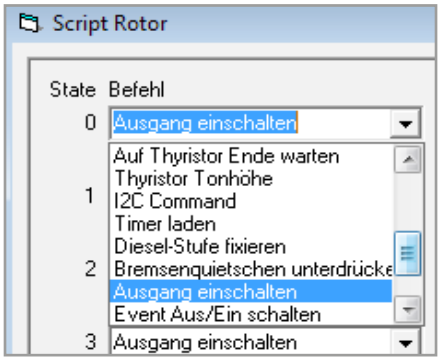
Mit Scripten kann man Abläufe programmieren, die über CV-Einstellungen nicht oder nur sehr umständlich einzustellen wären. Im ZSP-Programmer ist es möglich, Scripts anzulegen oder vorhandene zu bearbeiten. Programmierkenntnisse sind nicht erforderlich, die Fähigkeit zum logischen Denken jedoch schon.

Zum Öffnen eines Scripts genügt ein Doppelklick. Auch sie zu lesen ist keine große Schwierigkeit: Man beginnt oben links und macht zeilenweise weiter wie bei einem normalen Text. In der gleichen Reihenfolge wird später auch ein Script abgearbeitet. Es steht eine große Reihe von Aktionen und Parametern zur Verfügung, die man durch den Scriptablauf aufrufen und beeinflussen kann.

Das Script für den Schienenzeppelin ist umfangreich und setzt sich aus vier einzelnen Teilen zusammen. Mehr als vier miteinander verbundene Scripte sind nicht möglich, das Maximum der 645-Decodergeneration wurde ausgeschöpft. Die verbundenen Scripte stehen in gegenseitiger Abhängigkeit. Verschiedene Funktionstasten sowie die aktuelle Geschwindigkeit und die Fahrtrichtung modifizieren ihren Ablauf. Ein Exportieren und Importieren der Scripte ist ebenfalls möglich. So lassen sich Abläufe für ähnliche Loks auf dem PC sichern und bei Bedarf laden.



Beispielhaft ist hier das Script zur Festlegung der Fahrtrichtung abgedruckt. Screenshots der weiteren Scripte gibt es online und unter dem am Artikelende angegebenen Link.



Eine kleine Auswahl der verfügbaren Aktionen/Befehle

WIE ERSTELLT MAN EIN SCRIPT?

Ich möchte das Vorgehen anhand des Fahrtrichtungsscripts erklären: Wir hatten vorab festgelegt, dass der Schienenzeppelin nur vorwärts fahren darf, solange Propeller samt zugehörigem Sound läuft. Die Taste F1 soll diesen Betriebsmodus einschalten. Die Taste F4 wiederum aktiviert den Rangiermodus. Bevor aber das Vorwärts-/Rückwärtsfahren freigegeben wird, müssen der Propeller und sein Sound ausgeschaltet sein. Wenn dies der Fall ist, ist zu klären, ob der Schienenzeppelin fährt oder steht. Da die Taste F1 auch den Sound ein- bzw ausschaltet, erfolgt über sie ein automatischer Wechsel der unterschiedlichen Sounds. Die Steuerung des Propeller- und des Elektromotorsounds erfolgt über das Script „emotor“.

ALS BEISPIEL: DAS FAHRTRICHTUNGSSCRIPT

Ein weiteres Script ist für die Fahrtrichtung zuständig. Insgesamt waren zwölf Zeilen oder „States“, wie sie von Zimo genannt werden, nötig um die Fahrtrichtung zu steuern.

In der ersten Zeile wird abgefragt, ob die zur Taste F4 gehörige Funktion ein oder aus ist.In jedem Dropdown Menü gibt es verschiedenste Möglichkeiten der Aktion oder Prüfung. Aktionen sind:

- Sound starten
- Sound beenden
- Lok steuern
- Thyristor Sample wählen
- Thyristor Frequenzkurve setzen
- Timer laden
- Diesel-Sound auf Stufe zwingen
- Bremsenquietschen unterdrücken
- Ausgang setzen
- Event setzen/löschen
- EMotor Sample wählen
- Dieselablauf sperren/freigeben

Innerhalb der Abläufe kann/muss man bis zu vier Parameter definieren. Diese sind folgende:

- Ist-Geschwindigkeit
- Soll-Geschwindigkeit
- Sound Aus/Ein
- Timer-Wert
- Beschleunigungswert
- Bremswert
- Ist-Fahrtrichtung
- Soll-Fahrtrichtung

Angenommen, der Zustand von F4 sei aus. Unser State 0 (die erste Zeile) fragt ab, ob F4 ein ist. Wenn das zutrifft, wird zu State 3 gewechselt (Spalte „State wenn Taste=Zustand“). Da das hier jedoch nicht der Fall ist, wird in die nächste Zeile zu State 1 gewechselt. Die Fahrtrichtung wird hier auf vorwärts gesetzt. Danach wird in State 2 abgefragt ob die Funktion zur Taste F4 aus ist. Ist das der Fall, folgt die Anweisung, zu State 1 zu springen.

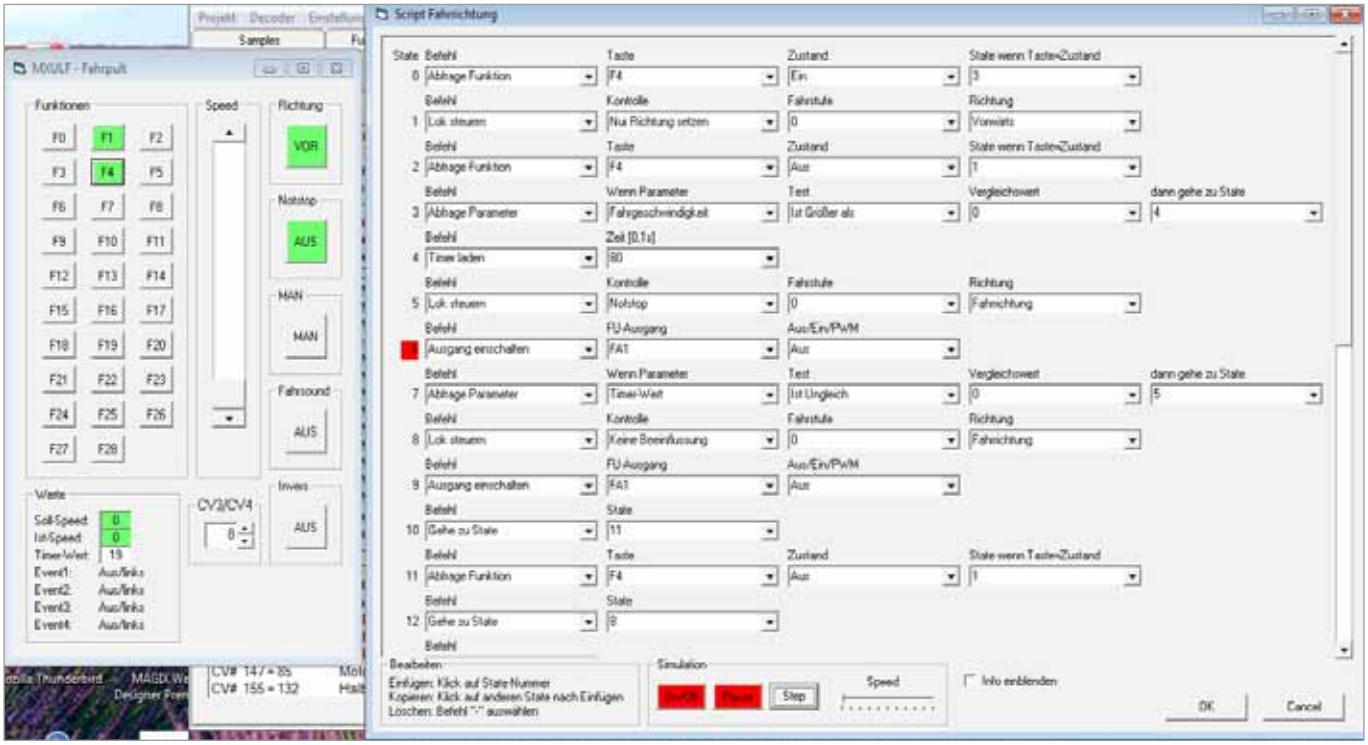
Das bedeutet, solange die Taste F4 aus ist, bleibt das Script in einer Schleife zwischen State 1 und 2. Der Zustand von F4 wird ständig innerhalb des Decoders abgefragt. Die Folge davon: Der Schienenzeppelin kann nur vorwärts gefahren werden, egal ob die Fahrtrichtung manuell gewechselt wird. Auch das Fahrlicht bleibt nur in Fahrtrichtung vorwärts an.

Wird nun F4 gedrückt und damit die zugehörige Funktion eingeschaltet, geht es mit State 3 weiter. Es folgt eine Abfrage, ob die Fahrgeschwindigkeit größer 0 ist. Wenn der Schienenzeppelin fährt, wird im State 4 ein 8-sec-Timer gestartet. Das Script springt zu State 5 und veranlasst einen Notstop. Im State 6 soll der Funktionsausgang 1 (FA1) ausgeschaltet werden (AUX 1 = Motor des Propellerantriebs)

Der Schienenzeppelin bleibt stehen und auch sein Propeller ist ausgeschaltet. Damit der Triebwagen nicht gleich wieder anfährt, wird der Timer in State 7 abgefragt. Solange hier nicht der Wert 0 erreicht ist, folgt die Anweisung, dass die Scriptausführung zurück zu State 5 springen soll.

Sobald der Timer abgelaufen ist (also den Wert 0 erreicht hat) wird zu State 8 weitergeleitet. Dort wird nun die volle Steuerung zugelassen.

Im State 9 soll immer noch der FA1 ausgeschaltet bleiben, da sich der Propeller im Rangiermodus nicht drehen darf. Im State 10 folgt die Anweisung „Wechsle zu State 11“. Dort wird dann die Funktion zur Taste F4 abgefragt. Wenn diese lange aus ist, wird zu State 1 zurückgesprungen. Ansonsten geht es in State 12 weiter, wo die Anweisung wartet, zurück zu State 8 zu gehen und damit die Schleife wieder zu beginnen. Diese Schleife von State 8 bis State 12 wird, solange die Funktion zur Taste F4 ein ist, immer wieder durchlaufen.



Beispielhaft ist hier das Script zur Festlegung der Fahrtrichtung abgedruckt.

Entsprechend kann das Fahrzeug im Rangiermodus in alle Richtungen gefahren werden. Erst beim Ausschalten des Rangiermodus bekommt der Decoder in State 11 die Anweisung, zu State 0 zu wechseln, wodurch alles wieder von vorne beginnt.

Anmerkung: Wenn ich am Anfang eine Aktion mit „Taste ein“ beginne, muss ich als vorletzten Befehl „Taste aus“ definieren. Wenn diese Prüfung dann positiv ausfällt, wird auf den letzten State des Scripts gesprungen, der sagt „Goto State -> 0“ und der Ablauf beginnt wieder von vorne. „State 0“ ist immer der erste Befehl in einem Script.

ABLAUFSIMULATOR

Mit dem Simulator hat man die Möglichkeit, einen Scriptaufbau und die Statereihenfolge zu überprüfen. Hat man „Pause“ aktiviert, stoppt die Simulation. Mit Klick auf „Step“ ist dann ein Schritt-für-Schritt- (State-für-State)-Durchlauf möglich. Somit können Werte und Funktionszustände gezielt gesetzt werden, um Schleifen, gegenseitige Abhängigkeiten und die weiteren Abläufe zu prüfen.

Läuft alles, wie man es sich vorstellt, ist der letzte Schritt, das Projekt zu speichern, indem alle Fenster mit Ok bestätigt werden. Danach können die Daten in den Decoder geladen werden, um das Script im Fahrzeug zu überprüfen.

Sollten in der Script-Logik Unklarheiten auftreten oder eigenartige Dinge passieren, schreibt Zimo: „Im Script den Fehler suchen … und falls das nicht hilft, ein Email an Zimo schicken!“

Konkret sollte man einen Fehler analysieren. Man kann mithilfe des Simulators feststellen, wo genau ein Fehler auftritt und dann die passenden Einstellungen vornehmen. Falls dies nicht zum Erfolg führt, speichert und schließt man das Script. In einem neu angelegten zweiten Script versucht man, den Fehler mit so wenig States wie möglich nachzubauen. Man fertigt eine genaue Beschreibung des gewünschten Verhaltens und eine des realen Scriptablaufs an und notiert, wo beide voneinander abweichen. Das alles schickt man dann an den Zimo-Service

Abschließend möchte ich noch anmerken, dass es eine Vielzahl von Stunden und unzählige Versuche und Experimente dauern kann, bis man den gewünschten Effekt erzeugt hat. Die Möglichkeiten, Scripte zu verschachteln und gegenseitige Abhängigkeiten zu erzeugen, sind fast unerschöpflich. Hier steckt natürlich auch die Gefahr drin, einen Decoder komplett unbrauchbar zu programmieren. Deshalb ist es wichtig, den vorherigen Zustand zu sichern, bevor neue Daten zum Decoder gesendet werden. Mit der Sicherung kann man Fehler rückgängig machen.

Bisher können die Scripts nur in der offenen ZPR oder in selbst erstellten Projekten bearbeitet werden. Zimo plant für die zweite Jahreshälfte 2019, auch die Scriptzugänglichkeit mit den ZPP „Ready-to-Use“-Downloadprojekten zu ermöglichen. Dazu wird es eine MFX-Version geben, die dann als „Ready-to-Use“-Projekt für den Schienenzeppelin perfekt sein wird. Wann es so weit ist, können Sie auf der Zimo-Internetseite in Erfahrung bringen. Ich möchte mich an dieser Stelle ganz herzlich bei Alexander Mayer bedanken. Ohne ihn wäre dieser Artikel und der Umbau des Schienenzeppelin smit Propellersound nicht möglich gewesen.

Manfred Grünig

LINKS

Update für Zimo MXULF	www.zimo.at/update/FlashFiles/ZSP_Su_V1_18_12.exe
Die weiteren Scripte	www.vgbahn.de/downloads/...xxx
Video vom Schienenzeppelin	www.vgbahn.de/QR/.....