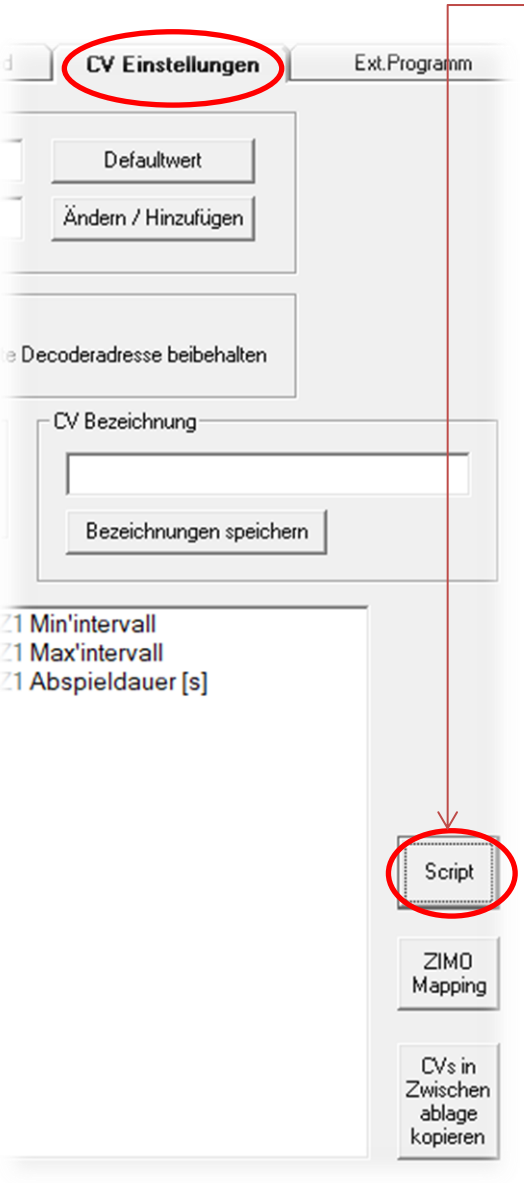
Scripts

Um Abläufe und Abhängigkeiten zu programmieren, gibt es in ZSP auch die Möglichkeit, mit „Scripts“ zu arbeiten. Scripts kann man sich als programmierte Abläufe vorstellen. Um sie zu erstellen braucht man allerdings keine Programmierkenntnisse, denn ZSP bietet die Möglichkeit, die gewünschten Abläufe in einer Eingabemaske zu programmieren. Es gibt definierte Funktionen, die man in einem Drop-Down Menü auswählen kann und je nach Befehl öffnen sich weitere Felder, in denen man die zugehörigen Parameter einstellen kann.

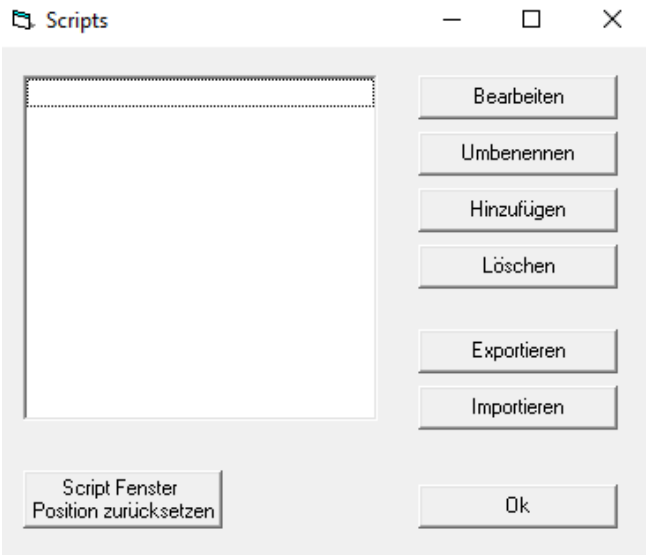
Zuerst aber zu den Grundlagen…

Wie komme ich zu den Scripts?

ZSP wie gewohnt aufrufen, wenn es nicht schon geöffnet ist, dann findet man die Scripts im Reiter „CV Einstellungen“, und dort klickt man rechts unten auf den obersten Button „Script“. Siehe linkes Bild:



Mit einem Klick auf diesen Button öffnet sich ein kleines Fenster mit einem leeren weißen Feld. Um ein Script zu erstellen, klickt man auf den Button „Hinzufügen“, und es erscheint ein blau markiertes Feld „Script1“. Mit „Umbenennen“ kann dem Script ein charakteristischer Name zugeordnet werden, der das Script beschreibt, damit man es in weiterer Folge leichter erkennt. Mit Doppelklick auf das Script öffnet sich ein leeres Script-Fenster. Falls sich dieses nicht Öffnen sollte (oder am Bildschirm nicht auffindbar ist) auf den Button „Script Fenster Position zurücksetzen“ klicken, dann erscheint das Fenster in der linken oberen Ecke des Bildschirms.



Ein Shortcut direkt zum Script Editor findet sich in der obersten Menüleiste des allgemeinen ZSP-Fensters mit Bezeichnung „Script“. Beim Anklicken klappt sich ein Menü auf, man kann direkt ein neues Script *hinzufügen*, oder bereits vorhandene Scripts öffnen. Ebenso kann man über „Übersicht“ das Kästchen aufrufen, in dem man Scripts erstellen, bearbeiten, importieren, exportieren und löschen kann.

Exportieren: Wenn ein Script fertig ist, kann man es an einen beliebigen Speicherort exportieren, um es z.B. in einem anderen Soundprojekt zu verwenden, einem Kollegen zu geben oder (im schlimmsten Fall) ein nicht funktionierendes Script an ZIMO zu senden.

Script importieren: wenn ein fertiges Script gespeichert und exportiert wurde, kann man es z.B.: in einem anderen Soundprojekt wieder importieren. ACHTUNG: Sound-Samples sind nur mit Nummern abgespeichert; diese sind von Soundprojekt zu Soundprojekt unterschiedlich und müssen daher nach dem Import manuell angepasst werden!

Notiz am Rande: Pro Soundprojekt ist es möglich, 4 Scripts zu erstellen (oder zu importieren). Pro Script kann man ungefähr 150 States (Befehle) programmieren. Deaktivieren kann man die Scripts in den zugehörigen Bits von CV #837.

Script-Editor

Rechts unten im Script-Editor ist ein Kästchen mit der Beschreibung „Info einblenden“. Wenn dieses aktiviert ist, kann man jedem „State“, also jeder Zeile einen Namen geben. Dies kann bei komplexen Scripts helfen, den Überblick zu bewahren, und auch später noch genau nachzuvollziehen, was man mit welchem Befehl bezwecken wollte. Den individuellen Namen gibt man ein, indem sie mit der Maus auf Info klicken und einfach zu schreiben beginnen. ZSP speichert die Bezeichnung ab, und man kann jederzeit (durch Setzen oder Entfernen des Häkchens rechts unten) zwischen den ursprünglichen Bezeichnungen der Befehle und der eigenen Beschreibung hin- und herschalten.

Links unten im Script Editor ist ein Infokästchen mit der Beschreibung „Bearbeiten“ für nützliche Aktionen während der Bearbeitung der Scripts. Diese sind folgende:

- Einfügen: Klick auf State-Nummer (fügt einen leeren State darüber ein)
- Kopieren: Klick auf einen anderen State nach Einfügen (fügt den kopierten State in den vorhin eingefügten – noch leeren – State ein. Die funktioniert nur, solange der eingefügte State noch rosarot hinterlegt ist)
- Löschen: Befehl ´-´ auswählen

In der Mitte unten im Script-Editor ist noch das Feld „Simulation“, dieses wird aber erst später erklärt, da es das fertig geschriebene Script testweise am Computer abspielt.

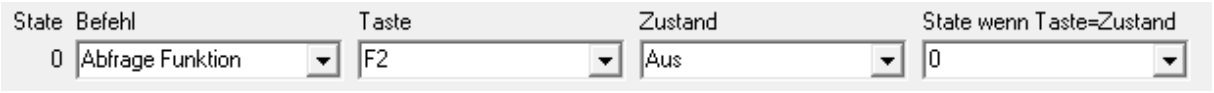
Wie schreibe ich ein Script?

Scripts zu lesen ist keine große Schwierigkeit; man beginnt links oben und „liest“ es Zeile für Zeile, also State für State, wie einen Text, bis rechts unten. Genauso macht es auch ZSP, und später der

Decoder, beim Abspielen der Scripts. Prinzipiell kann man mit den Scripts (innerhalb der vorgefertigten Aktionen und Parameter) sehr viele verschiedene Abläufe generieren. Bei der Konfiguration gibt es allerding ein paar Grundregeln, die man in jedem Fall beachten muss. Diese sind folgende:

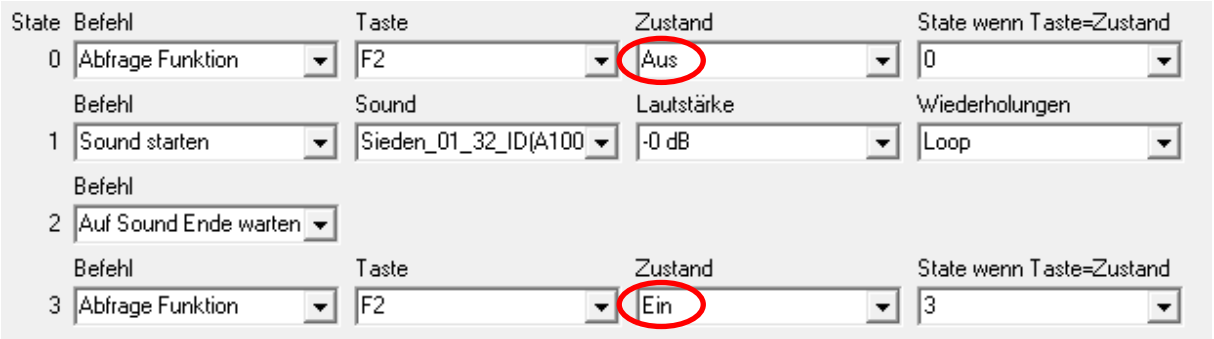
Wenn das Einschalten einer Taste abgefragt wird, muss auch das Ausschalten der Taste abgefragt werden

Dies erfolgt über die Aktion „Abfrage Funktion“. Die Abfragen funktionieren so, dass ein Zustand (einer Taste) abgefragt wird, und wenn dieser zutrifft, springt das Programm in den State, der in der letzten Spalte eingestellt ist, und führt den Ablauf an dieser Stelle fort. Wenn die Abfrage allerdings negativ ist, wird der Ablauf im nächsten State fortgeführt. Im folgenden Beispiel bleibt der Ablauf im gleichen State bis F2 eingeschaltet wird.



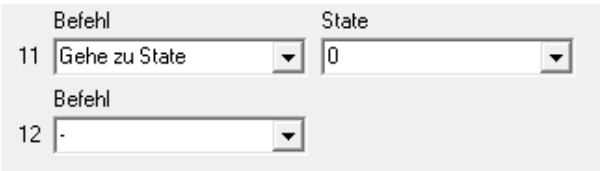
Zu lesen ist dies folgendermaßen: Der Befehl *Abfrage Funktion* prüft, ob *F2 Aus*geschaltet ist. Wenn dies der Fall ist, springt das Programm zu *State 0* (also bleibt in derselben Zeile). Wenn F2 nun (am Fahrpult oder über ZSP) eingeschaltet wird, ist die Abfrage negativ, und das Script springt automatisch in die nächste Zeile, also zum nächsten State. In diesem nächsten State sollte eine beliebige Aktion eingestellt werden, die mit dieser Funktionstaste ausgeführt werden soll.

Nach erfolgter Aktion muss man die Taste wieder ausschalten, was im Script aussieht, wie das gezeigte Beispiel, mit dem Unterschied, dass der „Zustand“ auf *Ein* geprüft werden muss, und der State auf sich selbst verweisen muss. Siehe folgendes Beispiel:



Am Ende eines jeden Scripts muss wieder nach oben gesprungen werden

Dies ist äußerst wichtig, da das Script sonst anhält und nicht mehr funktioniert. Die letzte Zeile in jedem Script sollte also folgendermaßen aussehen:



Es muss nicht auf State 0 gesprungen werden, es ist nur wichtig, dass das Script in sich einen Kreislauf bilden kann und nicht nach unten offen ist.

Beschreibung der Befehle:

Sound starten: Startet einen bestimmten Sound, der im ersten der 3 zugehörigen Parameter ausgewählt wird. Der erste Parameter ist ein Drop-Down Menü, in dem alle Soundsamples des Soundprojekts aufgelistet sind. Der zweite Parameter bestimmt die Lautstärke, mit der der Sound abgespielt wird und mit dem dritten Parameter kann man einstellen, ob der Sound geloopt wird, oder wie oft der Sound abgespielt werden soll (1x bis 8x). Sonderfunktion *Loop and Short*: Diese ist nur relevant, wenn das ursprüngliche Soundsample geloopt ist und danach der Befehl „Sound beenden" oder „Funktionstaste ausschalten" verwendet wird. Wenn beispielsweise F2 einen Sound mit *Loop and Short* startet, muss man danach F2 wieder ausschalten und/oder Sound beenden BEVOR das Script einen weiteren Befehl ausführen soll.

Sound beenden: Wenn der Sound mit *Loop* oder *Loop and Short* gestartet wurde (NICHT mit Wiederholungen), muss dieser Befehl gesetzt werden, um den Sound wieder zu beenden. Wenn nur einmalig oder mit Wiederholungen gestartet wurde, ist dieser Befehl nicht notwendig (da bei fix definierter Anzahl der Wiederholungen der Sound von selbst beendet wird).

Auf Sound Ende warten: Wartet so lange, bis das Soundsample (inklusive aller eingestellter Loops) zu Ende gespielt wurde und springt erst dann zum nächsten State. Dies hat nur Sinn, wenn der Sound nicht mit *Loop* oder *Loop and Short* gestartet wurde.

Abfrage Funktion: Prüft, ob eine Funktionstaste (welche man in Parameter1 einstellt) ein- oder ausgeschaltet (Parameter2) ist und definiert, wohin gesprungen werden soll, wenn diese Prüfung positiv ausfällt. Wenn nun ein Sound z.B.: mit F5 ausgelöst werden soll, prüft man, ob F5 ausgeschaltet ist – und springt zum eigenen State. So springt das Script automatisch zum nächsten State, wenn die Prüfung negativ ist, also F5 eingeschaltet ist.

Abfrage Parameter: Prüft, ob der abgefragte Parameter (Auswahlfeld *Wenn Parameter;* siehe Beschreibung der Parameter) gleich (=), ungleich (<>), größer (>) oder kleiner (<) (Auswahlfeld *Test*) als der definierte Wert (in Parameter3) ist. Genau wie bei *Abfrage Funktion* definiert man im letzten einstellbaren Parameter, wohin das Script springen soll, wenn diese Prüfung positiv ausfällt. Wenn diese Prüfung negativ ausfällt, wird der Ablauf im nächsten State fortgeführt.

Die Folgenden Parameter können im Auswahlfeld *Wenn Parameter* eingestellt werden:

- Fahrgeschwindigkeit: aktuelle interne Fahrstufe (0-255[1])
- Zielgeschwindigkeit: am Regler eingestellte Fahrstufe (0-255), auf die beschleunigt oder verzögert werden soll
- Fahrsound: Prüft, ob Fahrsound *Aus* oder *Ein*geschaltet ist.
- Ziel-Set: Diesel-Set, auf welches als nächstes gesprungen werden soll (wenn per Taste auf nächstes Diesel-Set umgeschaltet wird)
- Aktuelles Set: Prüft, welches Diesel-Set momentan abgespielt wird
- Diesel-Stufe: Prüft, auf welcher Stufe (also Drehzahl) sich der Diesel-Sound momentan befindet.

[1] Je nach eingestelltem Fahrstufenmodus bitte nachrechnen. Bei 126 Fahrstufen muss der Wert dementsprechend ungefähr mal 2 gerechnet werden

- Timer-Wert: Zählt von in „Timer laden“ gesetztem Wert hinunter auf 0. Mit diesem Parameter ist es möglich, innerhalb des Countdowns Aktionen auszulösen.
- Beschleunigung: Anzahl an Fahrstufen, um die beschleunigt werden soll. Wenn das Fahrzeug bei der Prüfung verzögert, liefert das Script den Wert 0.
- Verzögerung: Anzahl an Fahrstufen, um die verzögert werden soll. Es können nur positive Werte angezeigt und geprüft werden. Man prüft also z.B.: auf „um 40 Fahrstufen verzögert“.
- Ist-Fahrrichtung: Prüft auf aktuelle Fahrrichtung (Vorwärts, Rückwärts)
- Soll-Fahrrichtung: Prüft, welche Fahrrichtung die Lok laut Zentrale haben soll (Z.B: Lok noch im Bremsvorgang in „alter“ Richtung“)
- Consist aktiv: Prüft, ob sich Lok momentan im Consist-Betrieb befindet.
- Consist invers: Prüft, ob in CV #19, Bit 7 = 1 gesetzt ist (invertiert die Fahrrichtung im Consist-Betrieb)
- HLU-Limit: aktuelles Limit der HLU-Bremsstrecke
- ABC-Limit: aktuelles Limit der ABC-Bremsstrecke
- DC-Limit: aktuelles Limit der DC-Bremsstrecke
- Akt. Limit: Niedrigstes Limit der drei vorigen Parameter. Damit diese nicht alle drei im Script abgefragt werden müssen, testet dieser Parameter auf den niedrigsten gefundenen Wert

Lok steuern: Setzt Parameter für die Steuerung der Lok. Parameter1 definiert die Art der Kontrolle (Fahrstufe vorgeben, begrenzen, reduzieren, Notstop, etc.) in Parameter2 wird die Fahrstufe definiert und der letzte einstellbare Parameter definiert die Richtung (Vorwärts, Rückwärts, Ost, West, etc.).

Thyristor Sample: Wählt ein Soundsample des Thyristors aus (sofern es sich um eine E-Lok handelt). Das Script unterscheidet nicht, welches Soundprojekt momentan bearbeitet wird, daher gibt es diese Funktion auch bei Dampf- und Dieselloks.

Thyristor Tonhöhe: Bei diesem Befehl definiert man den Anstieg der Tonhöhe des Thyristors gebunden an bestimmte Fahrstufen. So definiert man in Parameter1 die Tonhöhe des Thyristors bei mittlerer Fahrstufe (siehe Parameter3) und in Parameter2 die maximale Thyristorhöhe bei maximaler Fahrstufe (Parameter4). Dabei ist zu beachten, dass der Wert „100“ die doppelte Frequenz des Samples bedeutet. In der Praxis würde ein Thyristor-Tonhöhen-Anstieg folgendermaßen funktionieren: Thyristor Sample 1 auswählen, Thyristor Tonhöhe bei Parameter1 auf 50, bei Parameter2 auf 100 stellen, danach Thyristor Sample 2 auswählen, und die Tonhöhen wie bei Thyristor-Sample einstellen. So bekommt man einen durchgehenden Tonhöhenverlauf.

I^2C Command: ZIMO-interne Testfunktion. Man kann damit am SUSI-Ausgang des MX645 ein beliebiges I^2C Paket aussenden und damit z.B. einen PCA9624 ansteuern um weitere FU-Ausgänge zu bekommen

Timer laden: Hier wird ein Timer gestellt, der im Hintergrund abläuft. Der Timer zählt in Zehntel-Sekunden bis auf 0 und bleibt dann dort stehen. Wenn nach einer bestimmten Zeit eine Aktion ausgelöst werden soll, den Parameter „Timer-Wert“ in *Abfrage Parameter* prüfen!

Diesel-Stufe fixieren: Aktiviert (oder Deaktiviert) eine fixe Dieselstufe auf: Stand - F20. Damit kann das Script die Kontrolle über den Dieselablauf[2] übernehmen, also eine bestimmte Stufe im Ablauf fixieren. Achtung: Wenn man „Dieselstufe fixieren“ in einem der nachfolgenden States wieder deaktiviert, muss man zusätzlich den Befehl „Diesel stufe setzen“ auswählen. Mit diesem Befehl definiert das Script, bei welcher Dieselstufe der Dieselablauf weitermachen soll. Beispiel: Eine F-Taste löst einen Sound aus, und es soll als alternatives Geräusch ein „Kaltstart“ abgespielt werden (Dieselstufe fixieren). Danach soll mit „Diesel-Stufe setzen“ das normale Standgeräusch wiedergegeben werden, und nicht erneut ein Startgeräusch.

Bremsenquietschen unterdrücken: Definiert, ob Bremsenquietschen unterdrückt werden soll oder nicht.

Ausgang einschalten: schaltet einen Funktionsausgang ein, und definiert auch die Dimmung (also PWM-Reduktion) für diesen bestimmten Ausgang. Im selben Parameter der Dimmung (Parameter2) kann der Ausgang auch ausgeschaltet werden. Diese Funktion kann man dazu einsetzen, mehrere Funktionsausgänge nacheinander auszulösen, wenn diese beiden Funktionen mit der selben Funktionstaste ausgelöst werden sollen. Beispielsweise soll zuerst FA6 eingeschaltet werden, und nachdem FA6 (Innenraumbeleuchtung) wieder ausgeschaltet ist, soll FAv mit 100% PWM aktiviert werden. Das alles soll aber innerhalb eines Scripts mit F5 gesteuert werden.

Event Aus/Ein schalten: Zuerst muss man sichergehen, dass im Tab „Ablauf Sound“ das Event, auf das man einen Servo im Script legen möchte, als Ansteuerungsquelle definiert ist. Danach muss man einen Eventmarker in einem .wav-file setzen. Dafür muss man im allgemeinen ZSP-Fenster das Soundfile auswählen, öffnen und darin einen Event-Marker setzen. Speichern Sie das Soundfile mit dem Eventmarker, und dann wird innerhalb des Scripts mit der Aktion “Event setzen“ an der Stelle des Markers der Servo auf die Position Links, Rechts oder Mitte bewegt, oder eine Funktion Aus- oder Eingeschaltet. Mit den Servopositionen kann so ein beliebig komplizierter Kupplungswalzer programmiert werden. Die Endpositionen der Servos werden in den entsprechenden CVs programmiert, sie können in den Scripts nicht verstellt werden.

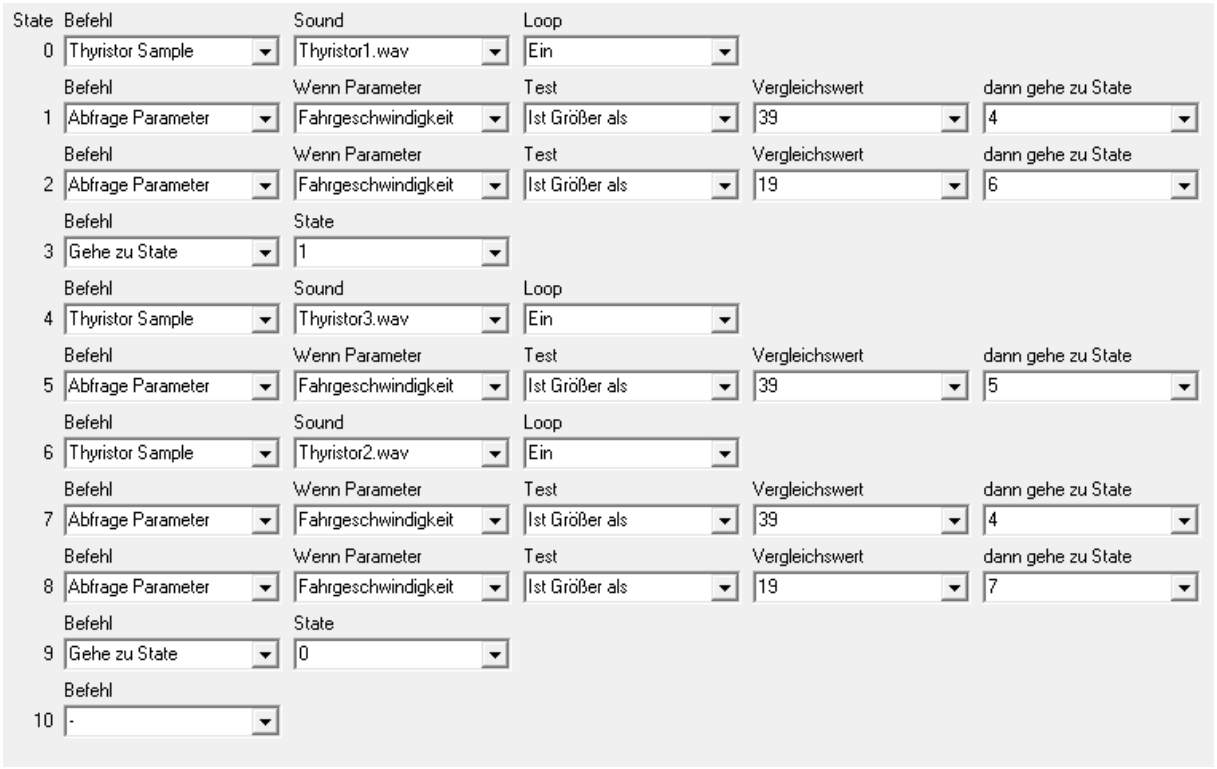
EMotor Sample: Definiert aus der Auswahl der im Soundprojekt vorhandenen Samples für den EMotor ob er geloopt wird oder nicht.

Diesel-Ablauf übernehmen: Siehe „*Diesel-Stufe fixieren*“ für den Fall, dass der Befehl dort wieder deaktiviert wird.

FA Effekt setzen: Damit kann man Effekte auf den in Parameter2 definierten Funktionsausgang legen. Zur Auswahl stehen nicht nur Lichteffekte (Mars, Ditch, Gyra, etc.), sondern auch Rauch, SoftStart, Dimmen, FA nach einer bestimmten Zeit ausschalten, und viele mehr. Hinweis: Der drittletzte Effekt, den man setzen könnte, ist noch unbenutzt, daher “-“ .

Am Einfachsten lässt sich die Anwendung der Scripts durch ein Beispiel erklären:

[2] Diesel Ablauf: Beim Beschleunigen oder Bremsen durchläuft der Dieselmotor verschiedene Stufen, bei denen die Tonhöhe unterschiedlich ist, also Stand, F1, F2, etc. und Übergangssamples (F1-F2, F2-F3, bzw. F4-F3, F2-F1, usw.)



Auf den ersten Blick sieht dieses Script sehr kompliziert aus, da es in sich relativ viel springt. Dieses Script baut auf 3 Thyristor-Samples auf, die ab bestimmten Fahrgeschwindigkeiten abgespielt werden sollen. Thyristor1 ist das erste Sample (auch von der Tonhöhe das tiefste). Dieses wird bis Fahrstufe 19 abgespielt, Thyristor2 ab Fahrstufe 20 (siehe State 2 und 6). Thyristor3 wird ab Fahrstufe 40 abgespielt (siehe State 1, 7 und 4). Die Thyristor-Samples sollen jeweils solange geloopt werden, bis das nächste Sample beginnt (siehe States 0, 4 und 6). Nun dazu, wie dieses Script „gelesen wird“:

0. Wird Sound-Sample „Thyristor1“ mit Loop abgespielt?
    a. Ja? → aktiviere das Script und gehe weiter zu State 1
    b. Nein? → bleibe in State 0
1. Frage ab, ob die Fahrgeschwindigkeit größer ist als 39
    a. Ja? → weiter zu State 4 (Thyristor3)
    b. Nein? → gehe weiter zum nächsten State, also State 2
2. Frage ab, ob die Fahrgeschwindigkeit größer ist als 19
    a. Ja? → weiter zu State 6 (Thyristor2)
    b. Nein? → gehe weiter zum nächsten State, also State 3
3. Dieser schickt den Ablauf wieder zu State 1 und führt die Abfragen weiterhin durch. Das macht das Script so lange, bis eine der beiden Abfragen in State 1 oder 2 mit „JA“ beantwortet wird, und das Script auf den definierten State springt.

Dieser Abfragen-Ablauf wiederholt sich in den nachfolgenden States.

States 1-6 definieren den Sprung nach oben, also die Thyristorgeräusche während des Beschleunigens. State 5 „fixiert“ Thyristor3 als Fahrgeräusch ab Fahrstufe 40 und höher. Ab State 6 („Spiele Soundsample „Thyristor2““) werden die Thyristorgeräusche für den Verzögerungs- bzw. Bremsvorgangs definiert. Wenn in State 7 abgefragt wird, ob die Fahrstufe größer als 39 ist, springt das Script solange zurück auf States 4 und 5, bis die Fahrstufe unter 39 fällt (eventueller

Bremsvorgang vor einer Kurve o.Ä.). Zu diesem Zeitpunkt wird Thyristor2 abgespielt, und erneut abgefragt, ob die Fahrstufe wieder größer ist als 39. Ist dies nicht der Fall, springt das Script weiter zum nächsten State, der abfragt, ob die Fahrstufe größer ist als 19. Die Abfragen in States 7&8 wiederholen sich so lange, bis die Fahrstufe 19 oder kleiner ist, damit das Script weiterspringen kann auf State 9. Dieser ist der letzte State, nämlich „Gehe zu State 0“ (soll in einem Script immer der letzte State – und so der letzte Befehl – sein), mit dem wieder das tiefste Thyristor-Sample abgespielt wird.

Wenn das Script fertig geschrieben wurde und man es testen will, oder zwischendurch testen will, ob und wie das Script läuft, hat man die Möglichkeit, das Script in einer Simulation durchlaufen zu lassen. Die funktioniert wie folgt: In der Mitte unten im Script-Editor gibt es den Button „Simulation“. Wenn das Script fertig programmiert ist, kann man es auf Fehler testen. Leider funktioniert das nur mit bestimmten Abläufen und Parametern, da ZSP bei einigen Dingen nicht in der Lage ist, sie vorzuführen (Servobewegungen etc.), grundsätzlich kann man aber jedes Scripts durchlaufen lassen, um zu sehen, ob es an einer Stelle hängen bleibt oder in einem Loop durchläuft (weil man vergessen hat, die F-Taste wieder auszuschalten o.Ä.). Wenn das Script zu schnell läuft, um die einzelnen Schritte nachverfolgen zu können, oder falls ein Fehler auftritt und man das Wo und Warum nicht exakt nachvollziehen kann, hat man die Möglichkeit, das Script Schritt für Schritt durchlaufen zu lassen. Dafür klickt man auf „Pause“, und dann geht man mit „Step“ jeden Schritt im Script einzeln durch.

Was mache ich, wenn ich das Script fertig habe, und es funktioniert wie gewünscht?

Mit Klick rechts unten auf „OK“ und man kommt zum ersten Script-Fenster zurück. In diesem kann man das erstellte Script exportieren (.zsc[3]) oder bereits erstellte importieren. Ebenso kann man weitere Scripts erstellen, jedoch sind pro Soundprojekt nur 4 verschiedene Scripts möglich.

Was mache ich, wenn ich das Script fertig habe, und es funktioniert NICHT wie gewünscht?

Im ersten Schritt analysieren Sie, wo der Fehler liegt. Dazu am Besten den Simulator starten, um zu sehen, wo genau der Fehler auftritt und diesen beheben. Falls beim Abspielen des Simulators kein Problem auftritt, speichern Sie das Script ab. Legen Sie ein neues Script an und versuchen Sie, den auftretenden Fehler mit so wenig States wie möglich nachzustellen.

Beschreiben Sie (in einem Word-Dokument oder Ähnlichem) zusätzlich das gewünschte Verhalten, das mit dem Script ausgelöst werden soll. Fertigen Sie dann eine Beschreibung des fehlerhaften Scripts an und vergleichen, wo diese beiden voneinander abweichen. Eventuell lässt sich auch so der Fehler finden und beheben. Falls das auch nicht hilft, die beiden Beschreibungen, das ursprüngliche Script und das „nachgebaute“ abspeichern, exportieren und (evtl. als .zip-Datei) an ZIMO senden.

Komfort-Features:

Da man beim Erstellen, Verändern und Testen der Scripts oft einige Zeit mit dem Script- Editor zu tun hat, gibt es einige Komfort-Features, die einem die Verweilzeit etwas erleichtern und angenehmer machen. Zum Einen sind das die schon beschriebenen Bereiche im unteren Teil des Fensters

[3] für **Z**IMO **Sc**ript

„Bearbeiten“, „Info einblenden“ und „Simulation“, zum Anderen hat dieser Editor noch zusätzliche spezielle Features.

Fenstergröße: Das Fenster des Script-Editors hat eine große Besonderheit (im Gegensatz zu den anderen ZSP-Fenstern): man kann es in der Größe verstellen und auch Vollbild machen. Dies bietet bei langen und komplexen Scripts den Vorteil, dass viele der States auf einen Blick sichtbar sind.

Scrollen: Zusätzlich kann man im Script auch scrollen, und ist nicht auf den rechten Balken angewiesen. Vorsicht: Beim Scrollen in einem der Befehlsfensterchen scrollt man nicht im Fenster, sondern nur innerhalb dieses Befehls- oder Parameterfensters!

Kopieren: Wenn man mitten im Script einen weiteren State hinzufügen möchte, braucht man nur auf den dahinterliegenden State klicken. Der neu eingefügte State ist rot eingefärbt. Wenn man hier einen anderen, bereits vorhandenen State, einfügen möchte, braucht man nur auf den klicken, der kopiert werden soll. Dies funktioniert aber nur, solange bei dem eingefügten State noch nichts eingestellt wurde, der State also noch rot eingefärbt ist. Wenn man den State nun kopiert hat, behält der eingefügte State seine Nummer und bekommt nicht die selbe State-Nummer wie der kopierte.