

# Anleitung Readme zur universellen Joysticksteuerung

---

## Anleitung Readme zur universellen Joysticksteuerung für Zugsimulatoren mittels AutoHotKey

---

### Zweck

Mit dem Erwerb des [TSC-X Controllers](#) stellte sich die Herausforderung, das Gerät in Betrieb zu nehmen, um festzustellen, dass es dafür keine vollumfängliche Softwareunterstützung gibt. Einige Freeware-Tools z.B. von Cobra, oder JoyToKey oder Antimicro erwiesen sich als ungeeignet.

Entweder konnten nicht alle Bedienelemente nach Wunsch belegt und angepasst werden, oder die analoge Umsetzung war schlichtweg unzureichend oder gar nicht möglich.

Vor allem die analoge Bedienung macht dabei aber ein Großteil des "Feelings" im Fahrsimulator aus.

Letztlich war die universelle Verwendung nicht gegeben, d.h. jeder Simulator erforderte eine andere, aufwändige Unterstützung, oder es gab gar keine. (z.B. SimRail)

### Prinzip

Im Rahmen der Entwicklung ergaben sich einige spezielle Anforderungen:

1. Analoges "Feeling"
2. Steuerung in Echtzeit, soweit möglich
3. Betriebssicherheit
4. leichte Anpassbarkeit
5. keine Eingriffe in API's oder komplexe DLL's
6. Für alle Simulatoren geeignet: (Hier: ZuSi V3, SimRail, Trainsimulator Classic, TrainSim World TSW)
7. Freeware

Im Rahmen meiner Recherche nach Alternativen, bin ich dabei auf das Programm "[AutoHotKey](#)." gestossen, welches auch Joystickinformationen abfragen und in "Tastendrücke" umwandeln kann, welche alle benutzten Programme ohne Probleme verwenden konnten.

# Umsetzung

Das Programm AutoHotkey (V2) wird über Scripte gesteuert, welche sich in einfachen Textdateien (.AHK) befinden und mit jedem Editor bearbeitet werden können.

Die Syntax ist leicht erlernbar und gut dokumentiert. AHK-Dateien können auch in EXE kompiliert werden, wovon ich aber Abstand genommen habe, da dann intransparent.

- **Echtzeit**

Die anfängliche Verwendung eines "großen" Skriptes pro Spiel hat sich als nicht praktikabel erwiesen. Zum einen ergab sich eine unübersichtliche Komplexität, zum anderen war das Laufzeit- und Echtzeitverhalten nicht mehr gegeben, da zu viele Verzögerungen im Script programmiert sind, welche sich zu erheblichen Laufzeiten addieren.

Als Lösung bot sich die Verwendung mehrerer gleichzeitig laufender Scripte an, um damit das Multitasking dem Betriebssystem zu überlassen.

- **Analoge Hebel**

Die Umsetzung der analogen Hebel ergab sich als erste Aufgabe, da die o.a. Programme aus irgendeinem unerfindlichen Grund diese Option nicht anbieten.

Im Prinzip wird nur abhängig von der Weite der Hebelbewegung eine Anzahl Tastendrucke in die jeweilig Richtung simuliert, dabei wird von der Startposition einfach auf- oder abwärts gezählt, womit jede Hebelposition klar definiert und zielsicher erreichbar ist. Das funktioniert auch sehr gut. An den Endausschlägen der Hebel wird dabei stets eine Rekalibrierung ausgelöst, womit die Hebel stets in eine definierte Lage zurückgebracht werden können, wenn sie doch mal einen Input "übersehen". Dies wird beim Bremse auslösen, oder Leistung abschalten quasi "automatisch" mit erledigt.

- **Funktionstasten**

Alle anderen Funktionstasten werden einfach sequentiell abgefragt und senden ihren Tastencode entsprechend. Im Laufe der Programmierung hat sich herausgestellt, dass ein einfaches "Senden"-Event oft nicht ausreicht und statt dessen ein Tastendruck von definierter Dauer "Key down" und "Sleep" und "Key up" erforderlich ist. Das hat sich später auch für die Hebel als vorteilhaft erwiesen, abhängig vom verwendeten Simulator und Fahrzeug.

- **Anpassungen**

Somit ergibt sich die Notwendigkeit für jeden Hebel (4) und die restlichen Funktionstasten (15) jeweils ein Script laufen zu lassen. Das ist problemlos möglich. Es laufen also i.d.R. fünf Scripte. Sie werden einfach durch Doppelklick gestartet. Sofern erforderlich wird auf die initiale Position der Hebel hingewiesen.

Meine mitgelieferten Scripte folgen dabei stets dem Prinzip Linker, Mittlerer, Rechter (Analog-) Hebel, rechter (Digital-) Stick, übrige Tasten. Dazu kommen bei Bedarf zusätzliche Bedienelemente (z.B.

Dynamische Bremse, oder Auslegung bei Dampflokomotiven, Shunting Funktionen EP\_07).

Die Speicherorte der Scripte sind beliebig wählbar und organisierbar, sie haben keinen direkten Bezug zum jeweiligen Game. Gestartet werden sie per Doppelklick (5x). Die Scripte werden durch die Taste "^" beendet, oder durch Rechtsklick "Exit" im Symbolbereich der Taskleiste.

In den Scripten selbst sind letztlich nur drei Parameter anzupassen:

**1.** Die verwendeten Tasten für "Auf" und "Ab" der jeweiligen Funktion, sowie die Anzahl der "Schritte", die der Hebel zurücklegen muss. Das hängt vom Fahrzeug ab und kann von 5 Schritten (z.B. statische Tipsteuerung) bis zu 100 Schritten (für feine analoge Steuerung) reichen. Muss man ausprobieren. Der dritte Parameter ist die Zeitdauer der "Tastendrücke" (Sleep) von 10 bis 600 ms. Muss man auch ausprobieren. Sind die Zeiten zu klein, werden die Tasten "übersehen" sind sie zu groß, werden Tasten mehrfach ausgelöst, oder das Script wird zu träge. Die entsprechenden Stellen sind im Script dokumentiert.

Beispiel:

#Requires AutoHotkey v2.0

#SingleInstance Force



;(c) JHaase, 2026

; --- KONFIGURATION ---

**TasteAb := "{NumpadSub}" ; Taste für Fahrstufe +**

**TasteAuf := "{NumpadAdd}" ; Taste für Fahrstufe -**

JoystickNummer := 1 ; Meistens 1

**Schwellenwert := 1 ; Alle wie viel Grad soll ein Tastendruck erfolgen?**

Intervall := 10 ; Prüf-Rate in ms

*Bsp.: Tastenbelegung festlegen*

Beispiel:

```
if GetKeyState("1Joy14"){
```

```
    Send "{TasteAuf down}"
```

```
    Sleep(400) ; Millisekunden warten
```

```
    Send "{TasteAuf up}"
```

```
    Sleep(10) ; Kurze Verzögerung für das Spiel/System
```

*Bsp.: Definierte Dauer des Tastendrucks festlegen*

**2.** Komplexe Mehrzonen-Tiptasten (z.B. E186 Traxx, Pendolino in SimRail) erfordern u.U. eine aufwendige Unterprogrammierung, da kommt man aber schnell rein, das würde hier zu weit führen. Beispiele sind genug mitgeliefert, die Dokumentation von AutoHotKey ist hilfreich.

**3.** Die Rekalibrierung in den Endlagen kann vom Winkel des Hebels und der Anzahl der Schritte angepasst werden. Zu viele Schritte halten den Hebel zu lange in den Endlagen fest, zu wenige reichen u.U. nicht zum Endanschlag aus.

Beispiel:

```
if (posX > 46) {  
    if (MaxP = 0) {  
        Loop 10 {  
            Send(TasteAuf)  
            sleep (50)  
        }  
        MaxP := 1  
    }  
}  
  
if (posX < 46) {  
    ;Reset Maxpunkt  
    MaxP := 0  
}
```

*(Ab Position 46 (von 50) wird die Taste "Auf" 10 mal gesendet!)*

Damit kann sich jeder ausgehend von den Universalsteuerungen seine individuelle Konfiguration anpassen. Man kann diese auch während des Spiels anpassen, indem das Script bearbeitet und einfach neu gestartet wird.

## Schluss

Einen herzlichen Dank an dieser Stelle an die Entwickler der Freeware AutoHotKey, ohne deren komfortable Programmierung das hier nicht möglich wäre.

Alle Anbieter von Hard- und Software sind auf meiner Webseite <https://www.knorsch-morsch.de> verlinkt und erreichbar.

Meine Scripte biete ich unter der GPL Lizenz an, d.h. jeder kann Sie verwenden und weitergeben, sofern keine kommerziellen Interessen verfolgt werden.

Ich würde mir wünschen, dass bei Verwendung meiner Scripte der ursprüngliche Copyright Hinweis nicht entfernt wird.

Alle Informationen in der aktuellen Fassung auf der o.a. Webseite gelten als Anlage zu diesem Dokument.

Wer möchte, kann mir zur Unterstützung meiner Arbeit eine kleine Spende dalassen, Infos dazu ebenfalls auf der Webseite.

**Und nun viel Spass und vielen Dank für die Aufmerksamkeit!**