

Instructions (Readme) for the universal Joystick control

Instructions (Readme) for the universal joystick control for Train simulators using AutoHotKey

Purpose

With the purchase of [the TSC-X Controller](#) The challenge arose of getting the device up and running, only to discover that there was no comprehensive software support for it. Several freeware tools, such as those from Cobra, JoyToKey, or Antimicro, proved unsuitable.

Either not all controls could be assigned and customized as desired, or the analog implementation was simply inadequate or not possible at all.

However, the analog controls are a major part of the "feeling" in the driving simulator.

Ultimately, universal usability was not a given; that is, each simulator required different, complex support, or none at all. (e.g., SimRail)

principle

During the development process, some specific requirements arose:

1. Analog "feeling"
2. Real-time control, where possible
3. Operational safety
4. Easy adaptability
5. No modifications to APIs or complex DLLs
6. Suitable for all simulators: (Here: ZuSi V3, SimRail, Trainsimulator Classic, TrainSim World TSW)
7. Freeware

During my research for alternatives, I came across the program "[AutoHotKey](#)" encountered [a system](#) that can also query joystick information and convert it into "key presses", which all used programs were able to use without problems.

Implementation

The AutoHotkey (V2) program is controlled via scripts, which are located in simple text files (.AHK) and can be edited with any editor.

The syntax is easy to learn and well-documented. AHK files can also be compiled into EXEs, but I've avoided this because it becomes opaque.

- **Real-time**

The initial use of a single "large" script per game proved impractical. Firstly, it resulted in an unmanageable level of complexity, and secondly, runtime and real-time performance were no longer guaranteed due to too many delays programmed into the script, which added up to significant runtimes.

The solution was to use several scripts running simultaneously, thus leaving the multitasking to the operating system.

- **Analog levers**

Implementing the analog levers was the first task, since the aforementioned programs do not offer this option for some inexplicable reason.

Essentially, the system simulates a number of keystrokes in each direction, depending on the lever's movement. The system simply counts up or down from the starting position, ensuring that each lever position is clearly defined and easily achievable. This works very well. Recalibration is triggered at the lever's end positions, allowing it to return to a defined position if it happens to "miss" an input.

This is done almost "automatically" when the brakes are applied or the power is switched off.

- **Function keys**

All other function keys are simply queried sequentially and send their corresponding key code. During programming, it became apparent that a simple "send" event was often insufficient, and instead, a key press of defined duration for "Key down," "Sleep," and "Key up" was required. This later proved advantageous for the levers as well, depending on the simulator and vehicle used.

- **Adjustments**

This necessitates running a script for each lever (4) and the remaining function keys (15). This is easily achievable. Typically, five scripts are running. They are started simply by double-clicking. If necessary, the initial position of the levers is indicated.

My included scripts always follow the principle of left, middle, right (analog) lever, right (digital) stick, remaining buttons. Additional controls are added as needed (e.g.,...)

Dynamic brake, or design in steam locomotives, shunting functions EP_07).

The scripts can be stored in any location and organized as desired; they are not directly related to the specific game. They are started by double-clicking (five times). The scripts are terminated by pressing the "^" key or by right-clicking and selecting "Exit" in the taskbar icon area.

Ultimately, only three parameters need to be adjusted in the scripts themselves:

1. The buttons used for "Up" and "Down" of the respective function, as well as the number of "steps" the lever must travel. This depends on the vehicle and can range from 5 steps (e.g., static tip control) to 100 steps (for fine analog control). This needs to be tested. The third parameter is the duration of the "button presses" (sleep), ranging from 10 to 600 ms.

You have to try it out. If the timings are too short, the keys will be "missed"; if they are too long, keys will be triggered multiple times, or the script will become too sluggish. The relevant sections are documented in the script.

Example:

#Requires AutoHotkey v2.0

#SingleInstance Force



;(c) JHaase, 2026

; --- CONFIGURATION ---

KeyDown := "{NumpadSub}" ; Key for gear +

KeyUp := "{NumpadAdd}" ; Key for gear position -

Joystick number := 1 ; Usually 1

Threshold value := 1 ; At what degree interval should a button press occur?

Interval := 10 ; Test rate in ms

Example: Setting key assignments

Example:

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

```
Send "{" ButtonUp " down}"
```

```
yyyyyyyyyy Sleep(400) ; wait milliseconds
```

```
Send "{" ButtonUp " up}"
```

```
Sleep(10) ; Short delay for the game/system
```

Example: Defining the duration of a key press

2. Complex multi-zone touch buttons (e.g. E186 Traxx, Pendolino in SimRail) may require extensive sub-programming, but you can get the hang of it quickly; that would go too far here. Plenty of examples are included, and the AutoHotKey documentation is helpful.

3. The recalibration at the end positions can be adjusted by the angle of the lever and the number of steps. Too many steps hold the lever in the end positions for too long; too few steps may not be sufficient to reach the end stop.

Example:

```
if (posX > 46)
{ if (MaxP = 0) {
Loop 10 {
Send(KeyUp) sleep
(50) }
```

```
MaxP := 1
```

```
}}
```

```
if (posX < 46) {
Reset Max Point
MaxP := 0
}
```

(From position 46 (out of 50) the "Up" key is sent 10 times!)

This allows everyone to customize their individual configuration starting from the universal controls. These can also be adjusted during gameplay by editing the script and simply restarting the game.

Ending

A heartfelt thank you to the developers of the freeware AutoHotKey, without whose convenient programming this would not be possible.

All hardware and software providers are listed on my website <https://www.knorsch-morsch.de> linked and accessible.

My scripts are offered under the GPL license, meaning anyone can use and distribute them, provided no commercial interests are pursued.

I would prefer that the original copyright notice is not removed when my scripts are used.

All information in its current version on the aforementioned website is considered an appendix to this document.

Anyone who wishes can leave me a small donation to support my work; information about this can also be found on the website.

And now, have fun and thank you for your attention!