Train Sim World
TARGET Profile for Thrustmaster Warthog

Introduction

This document explains how to use the included Thrustmaster TARGET script created for the HOTAS Warthog Throttle intended to work with Train Sim World (TSW). The script only uses the Warthog Throttle. The Warthog Joystick is not used. The Warthog axes are mapped as follows:

- The Friction Control axis controls the reverser
- The Right Throttle axis controls the engine throttle (SD40-2, GP38-2), or throttle/brake lever (AC4400CW).
- The Left Throttle axis controls either the dynamic brake, automatic brake, or independent brake, depending on the setting of the Flaps switch.
- Logitech/Saitek Throttle Quadrant profiles are also included to complement the Warthog by controlling the independent brake and automatic brake separately, allowing the Warthog Left Throttle axis to control the dynamic brake alone. It is possible, however, to use just the Warthog to control all the engine levers.
- Supported engines: SD40-2, GP38-2, AC4400CW - Other engines with similar throttle and brake controls should work, or can be made to work with minor adjustments. The GP40-2 should work without modification, but has not been tested.

The TARGET script has several files included which are explained in the installation section. It is not necessary to understand how to program TARGET scripts to use this, but it will help a great deal if customizing the script is desired. This script is also a good example for showing what can be done with TARGET using more advanced techniques.

To avoid frustration, please read through this document to understand how to use the Warthog to control a train. Understanding how the real train controls work is a little confusing, which can make using this script even more confusing.

The script has two profiles, one for engines with separate throttle and dynamic brake levers like the GP38-2, SD40-2, and GP40-2. The other profile is for the AC4400CW which has a combined throttle/dynamic brake lever. Since I only have TSW: CSX Heavy Haul, I have only tested it with the GP38-2, SD40-2 and AC4400CW. The AC4400CW profile may work with the commuter engines in the other TSW DLC (with combined throttle/brake levers), provided those engines have similar throttle (backward) and brake (forward) notch ranges on the throttle/brake lever. See the technical Details Section of this document for an explanation.

Installation

Installation is a manual process. Here is a list of the included files.

- *Logitech-Saitek Throttle Quadrant Profiles*
  Profiles for the Logitech/Saitek Pro Flight Throttle Quadrant and Pro Flight Yoke (which has a Throttle Quadrant). It is optional if you want to use these.
- *TARGET Script files*
  The Thrustmaster Warthog Throttle Script files.
- *Train Simi World – TARGET Script Doc.pdf*
  The document you are reading now.
- *TrainSimWorld_Generic_Warthog Controller Layout.xlsx*
  Excel file that shows the layout of all controls mapped in the TARGET script file. Also includes mapping for the Saitek Throttle Quadrant if you use those profiles.
- *wBeep files*
  The wBeep.exe program used in the TARGET script – for audible feedback when using the script.

To install the TARGET script…

- Copy the *Drakoz_TrainSimWorld.ttm* and *Drakoz_TrainSimWorld_Warthog.tmc* files from "*Target Script files*" to where you normally store your TARGET script files. By default, this should be:
  c:\users\<your username>\AppData\Roaming\Thrustmaster\TARGET\Scripts
  The AppData folder is normally hidden. You can access it easily by hitting the Windows key and typing %appdata% <enter> (for Windows 7, 8, 10) and it will open the Appdata folder. Or it is OK to save the files anywhere as long as the TARGET Script editor can brose to them (see below).
- Copy wBeep.exe to C:\bin\wbeep.exe. You will likely have to create the C:\bin folder.
  - If you don't want to use or install wBeep.exe, see the Technical Details section below for how to comment out this function.
- If you want to use the Logitech/Saitek profiles, see the README file in that directory for where to copy them. You only need one file or the other (the Pro Flight Yoke or Pro Flight Quadrant files) as each profile does the same thing.

To run the TARGET script…

- Make sure your Thrustmaster HOTAS Warthog Throttle is plugged in. The Warthog Joystick is not needed, but it does not hurt to have it plugged in. The same is true for any other Thrustmaster TARGET compatible controllers. This script will ignore them.
- Run the TARGET Script Editor (not the TARGET GUI).
- Select the TOOLS tab.
- Make sure FILE PATHS in TARGET points to where you copied the .tmc and .ttm files. If you copied the script files to a non-standard place, this is where you need to tell TARGET where they are.
  - Click on Options
  - Select "FILE PATHS" tab
  - Make sure the *Drakoz_TrainSimWorld.ttm* and *Drakoz_TrainSimWorld_Warthog.tmc* files are in a folder that is in the list of INCLUDE FILE LOCATIONS.
- Load the *Drakoz_TrainSimWorld_Warthog.tmc* file.

- Click on MENU
- Click on OPEN
- Find the file “Drakoz_TrainSimWorld_Warthog.tmc” and open it.
- The .ttm file will be loaded automatically for you when you run the script.

- Compile and run the script by clicking “RUN”.
- If all was installed correctly, it should compile with no errors, and you should hear a beep. The beep is part of the script using the wBeep.exe program. If you do not hear a beep, recheck that you have copied wBeep.exe to C:\bin or check your audio settings. The script will not run without wBeep.exe.
- If it fails to run for some other reason, it is probably a TARGET Script Editor setup thing, or you forgot to plug in your Warthog. Check the error log in the TARGET Script Editor, resolve the problem and try again.
- The script is now running and ready to be used in TSW. Please see the section “Using the TARGET Script below before continuing.

Once running, be careful about moving the axes on your Warthog Throttle when you are not in Train Sim World as it will send lots of keystrokes to the computer with unintended consequences.

To Install the Logitech Profiles (optional)

- See the README file in the *Logitech-Saitek Throttle Quadrant Profiles* folder.
- These profiles are only needed if you have and plan to use a Logitech Throttle Quadrant or the Throttle Quadrant that came with a Pro Flight Yoke. They are functionally the same, so I created a version of the profile for each as I have both. I only use one or the other, not both at the same time.

Using the TARGET Script

Basic usage

- Compile and run the TARGET Script.
- Select the desired engine profile (MSP+LDGH)
- Load Train Sim World and select a scenario.
- Before entering the scenario, set the axes and switches on the Warthog to match the current state of the engine.
- See the Excel file for button and axes configuration.

Please see below for detailed usage as several things will not be completely obvious when using this script.

Detailed Usage

Cheat Sheet – Open the *TrainSimWorld_Generic_Warthog Controller Layout.xlsx* file which has a layout of the controller and how the buttons, switches, and axes are mapped. Here are some notes on how to read this file.

- Each button and axis on the Throttle is listed in the gray boxes.
- **Black** text lists the **primary function** of a button (e.g. push the APENG button and it blows the horn).
- **Red** text is a **shifted function** controlled by the Thrustmaster IO shift button. The shift button is MSP (Mic Switch Hat center button). E.g. press MSP and the LDGH button, and it will change to the next engine profile (more on engine profiles below).
- **Orange** text is a **Press and Hold function**. e.g. Press and hold the Mic Switch HAT forward, and it will switch to Free View (view 8 in TSW).
- **Blue** text indicates **mode based functions** that change based on the position of the **Flaps Switch**. This is the Thrustmaster UMD mode switch.
- **Light Blue** text is **setup notes** or additional functional notes related to the sim. `
- The file also includes mapping information for the Saitek Throttle Quadrant, the Logitech G700S mouse which I use, and the unused Warthog Joystick.

Following is an explanation of specific controls programmed on the Warthog Throttle.

Reverser

The Warthog Throttle Friction Control axis (TFC axis) is mapped to control the engine reverser lever. Try it, it's pretty self-explanatory. If the Warthog TFC axis gets out of sync with the engine's reverser (which happens usually when starting a new scenario without first setting the TFC to match the reverser in game) just move the TFC axis lever full forward, then full backward, then back to center, and the TFC axis will be in sync with TSW again.

As a demonstration, while sitting in the engineer's seat of an engine, move the TFC axis forward and backwards rapidly several times. See how there is a short lag in TSW to similarly move the reverser. Also see how every movement of the TFC axis eventually happens in TSW. Remember this fact as you read through the rest of this description. This is the primary difficulty with using the TARGET script as it is easy to push the Warthog axis further or faster than TSW can accept the commands, causing things to get out of sync.

Throttle vs. Throttle/Brake Lever in TSW – and Engine Profiles

There are two engine profiles configured in the script, one for the GP38-2/SD40-2 and one for the AC4400CW. To change profiles, press and hold the MSP button, then press the LDGH button. TARGET will beep to indicate which profile was selected. The LEDs will also indicate the selected profile.

- Profile 1 – for the GP38-2, SD40-2, GP40-2 – 1 beep, LED 1 lit (default profile)
- Profile 2 – for the AC4400CW – 2 beeps, LED 2 lit

The difference between profiles is how the Warthog Right Throttle axis is mapped (separate throttle and dynamic brake vs. combined throttle/dynamic brake). The Left Throttle axis is mapped the same for both profiles, but the AC4400CW ignores the use of the Left Throttle axis as a dynamic brake (Flaps switch set to FLAPU) and only supports Automatic Brake (FLAPM) and Independent Brake (FLAPD).

Profile 1 – GP38-2/SD40-2/GP40-2 – Separate Throttle and Dynamic Brake

The GP38-2/SD40-2 have separate throttle and dynamic brake levers. This profile is designed for the Warthog Right Throttle to control the engine throttle, and the Warthog Left Throttle to control the three brake levers (as chosen by the Flaps switch). It should work for any engine with similar notch patterns as the GP38-2 and SD40-2 for these two levers (e.g. GP40-2 DLC), but other engines have not been tested.

GP38-2/SD40-2 Throttle

The GP38-2/SD40-2 throttle lever has 8 positions plus Idle. You can move the Warthog Right axis back and forth full deflection, and it will accurately map to the idle plus 8 positions on the locomotive throttle. If you move the Warthog Right axis quickly, the engine throttle will lag behind, but it will accurately track the movements.

The TARGET script produces a low frequency beep (using wBeep.exe) as feedback when the throttle is moved from one position to the next. Different beeps are used for increasing throttle and decreasing throttle (160Hz for increasing throttle, 100Hz for decreasing throttle. This gives feedback in the case that the Warthog Right axis is right on a threshold and jumps up and down unexpectedly. The beeps give notification which direction the throttle moved in TSW.

As a backup in case the Warthog Right axis gets out of sync with the TSW throttle, it is possible to press the Speedbrake switch forward (SPDF) to move the engine throttle lever forward, or backward (SPDB) to move the engine throttle lever backwards. This is the same as pressing the ‘a’ and ‘d’ keys on the keyboard. Or, it works to pause the came (ESC) re-adjust the throttle and unpause.

GP38-2/SD40-2 Dynamic Brake

The dynamic brake lever has two notched positions (OFF and SETUP), and a continuously variable section (no notches) numbered 1 through 8.

To use the dynamic brake on the GP38-2 or SD40-2 in TSW, the procedure is supposed to be to shut down the engine throttle, wait 10 seconds, then move the dynamic brake to SETUP. After a moment of setup, the dynamic brake system will start to make noise and a small negative current will appear on the amps gauge. Then it is OK to move the dynamic brake lever to the 1 through 8 position to apply dynamic braking.

In TSW, the . key increases the dynamic brake and the , key decrease it. If . or , are held down, the dynamic brake lever will sweep from one end of the scale to the other, including right through the OFF and SETUP positions. If it is desired to use the dynamic brake realistically, sweeping right through OFF to SETUP to 1 through 8 is not the correct way to do it. Instead, to go from OFF to SETUP, you should do a long press of the . key (greater than 100ms), wait for SETUP to do its thing, then another long press of . or press and hold to move to the desired braking position (1 through 8). Once in use, you would use small presses or taps on the , and . keys to minutely adjust the amount of braking. Because the dynamic brake has no notches, you could be partway between numbers. When switching back to throttle, you can just hold down , to move the dynamic brake back to OFF, then apply throttle.

In reality, it would be a mistake to move back to SETUP or OFF while you are still using dynamic braking. So TSW has built in the long presses needed to jump from OFF to SETUP to 1, and from 1 to SETUP to OFF to help the player delineate these settings from the actual braking zone 1 through 8. In the real world I assume the button on the end of the lever must be pressed to move past these zones. It prevents you from accidentally turning dynamic braking off while doing small movements of the lever or pulling it all the way back to 1.

For technical reasons explained in the Technical Details section, it is not easy to map these long vs. short presses to the Warthog axis (or any game controller with programming software). If I map the throttle to do a long press so it can reliably get from OFF to SETUP to 1 and back, we pay a price in having only long presses for moving the dynamic brake in the 1 through 8 area. So I compromised and set up the Warthog Left Throttle to do only short presses, and configured the Pinky Switch (PSF and PSB) to do long presses.

To use the Warthog for dynamic braking, when you want to switch to dynamic braking, you shut down the throttle and wait 10 seconds. Then make sure the Flaps key is in the FLAPU position (forward) to select dynamic braking mode for the Warthog Left axis. Then flip the Pinky Switch forward and it will give a long press to move the dynamic brake from OFF to SETUP. Recenter the Pinky Switch and flip it forward again to move from SETUP to 1. Now use the Left Throttle axis to adjust the dynamic brake from 1 to 8. It will do so in about 28 steps which is good enough resolution to have fine control over dynamic braking. When done with dynamic braking, pull the Left Throttle Axis all the way back. Then flip the Pinky Switch backwards twice to move from 1 to SETUP, then to OFF. It isn't 100% realistic, but it is precise. Think of the pinky switch as if it is the push button on the lever in a real locomotive. Again, the goal was to create a precise control to get the analog axis of the Warthog to match the lever in TSW.

Be aware, if the dynamic brake lever is in OFF or SETUP and you move the Left axis forward quickly, it will overcome the long press gaps between OFF, SETUP and 1 and move right into dynamic braking without using the Pinky Switch. But the result is the Warthog Left axis will be out of sync with the dynamic braking lever in TSW. It will still work, but you will not be able to reach max braking. You can similarly pull the Left Throttle axis all the way back quickly and it may overcome the long press zones and go back to SETUP or OFF. If that happens often, try moving the Left Throttle axis a little slower instead of throwing it full speed to the stops.

If you do get out of sync, you can press and hold the Left Throttle Button (LTB) and it will temporarily disable the Warthog Left axis. You can then move the axis to a new position, release LTB, and it re-enables the axis. Or pause the sim (ESC), readjust the Left axis, and unpause.

Note, TSW does not enforce the need to wait 10 seconds after shutting down the engine throttle, or the need to go from OFF to SETUP and wait before going to 1 for dynamic braking. Nothing seems to get damaged, and dynamic braking still works. Again, if you want to perform these actions realistically, follow the instructions above. Or you could just choose not to care, get ham fisted with the Left axis and use the LTB button to readjust when things aren't in sync. Play with it to understand how it works and decide what you care about. It's just for fun after all.

There is a chance, due to different computer performance, that a short and long press on my computer will not match what is needed on your computer. In that case you will need to adjust these delays in the TARGET script. See the Technical Details section for assistance with this.

Use of the automatic and independent brake functions will be explained below.

Profile 2 – AC4400CW – Combined Throttle and Dynamic Brake

The AC4400CW has a single combined throttle and dynamic brake lever. This profile is designed for the Warthog Right Throttle axis to control the combined throttle/brake lever and the Warthog Left Throttle axis to control the automatic and independent brake levers (as chosen by the Flaps switch). It should work for any engine with a similar notch pattern as the AC4400CW throttle/brake lever, but this profile has not been tested with any locomotive other than the AC4400CW. Also, the dynamic braking mode (FLAPU) on the Left axis is functional but ignored by the AC4400CW.

AC4400AC Throttle/Dynamic Brake

The AC4400AC throttle/brake lever has 8 positions plus IDLE for the throttle side, a hump in the middle to switch to the dynamic braking SETUP position and a continuously variable (no notches) section for the dynamic brake side marked SETUP to 1 through 8.

It is important to remember that the AC4400CW throttle/brake lever cannot be moved to dynamic braking if the reverser is in the center position. Hence it is easy to get the Warthog Right axis out of sync with the AC4400CW throttle/brake lever if you push it forward while the reverser is centered.

Before entering an engine, select Profile 2 (press and hold MSP, then press LDGH). You will hear two beeps indicating profile 2 has been selected. LED 2 will be lit on the Warthog. Now center the Warthog Right axis. As you move the axis, you will hear low beep tones and some high tones. These tones delineate between the various notches on the AC4400CW throttle/brake lever. Different beeps are used for increasing throttle/braking and decreasing throttle/braking (160Hz for increasing throttle or brake), 100Hz for decreasing throttle or braking). This gives feedback in the case that the Warthog Right axis is right on a threshold and jumps up and down unexpectedly. The beeps give notification which direction the throttle moved in TSW.

To understand how the Warthog Right axis works with the beeps, do the following.

- Load TSW and get in the engineer's seat for a AC4400CW.
- Select Profile 2 (MSP and LDGH).
- Make sure the reverser is centered.
- Slowly pull the Warthog Right Throttle all the way back.
    - You may hear some thumps (low beeps at 100Hz and 160Hz), some chirps (high pitch beeps), and a long dull tone. These sounds are generated by the TARGET script and are intended to represent the different notches on the throttle/brake lever in the AC4400CW.
- Press and hold the Speedbrake switch back (SPDF) on the Warthog for a moment (or the a key) and make sure the throttle/brake lever in the AC4400CW is all the way back, on setting 8. The Warthog and AC4400CW throttle levers should now be in sync.

- Slowly move the Warthog Right axis forward. You will hear 8 thumps (100Hz) followed by a chirp on the 8th thump. The chirp means you are at IDLE. The AC4400CW throttle lever should also be at idle.
- Don't go past the chirp or you will cross the throttle/braking hump and get things out of sync (because the reverser is centered and the AC4400CW will not go into braking mode). Try it if you want, then redo the above steps to get back to the throttle at idle.
- Move the reverser to forward or backwards (direction doesn't matter).
- Slowly push the Warthog Right Throttle axis forward until you hear a long dull tone followed by a chirp. This means the Warthog Right axis has moved to the dynamic braking SETUP position. The AC4400CW throttle/brake lever should have followed by moving past the hump to SETUP.
    - This is the tricky part of the TARGET script. Like the Profile 1 long and short presses, Profile 2 must use a long press (about 1 second) to get past the hump, but short presses to move the lever through the throttle IDLE and 1 through 8 positions, but even shorter presses to give good resolution on the dynamic braking side which has no notches.
- Slowly push the Warthog Right axis forward all the way. You will hear 13 thumps (160Hz).
    - Each thump moves the AC44400CW dynamic brake lever a small increment. Even though the dynamic brakes do not have notches, the 13 thumps give feedback as to when you have moved the Warthog axis. When you hear a thump you know you actually made a change on the AC4400CW lever. It is just the compromise of how the TARGET script is done. There should still be plenty of resolution to effectively control dynamic braking.
- Slowly pull the Warthog Right axis backward until you hear the 13th thump (100Hz) and a chirp. You should now be at SETUP on the AC4400CW throttle/brake lever
- Slowly pull the Warthog Right Throttle backward again until you hear a long dull tone followed by a chip. You should now be back at the AC4400CW throttle/brake IDLE position.
- Play with it to get used to it.
    - Some people may like it, some may hate it. This is just the compromise we have to deal with mapping an analog axis on a game controller to press keyboard buttons to move a lever in TSW. It took C Code programming in TARGET to pull this off as even the normal TARGET script commands don't have the ability to do this normally.
    - It takes a little practice to get used to it, but works well once you understand it and can do it without thinking about it too much.

If the above procedure does not work, it is likely because the timing for key presses on your computer vs. my computer are not consistent, and the TARGET script will have to be modified to change the timings for your computer. See the Technical Details section for more information.

If you get the Warthog and AC4400CW throttle/brake lever out of sync, you can adjust the throttle/brake lever with the Speedbrake switch on the Warthog (SPDF and SPDB). This is the same as pressing 'a' and 'd' on the keyboard.

Automatic Brake

In TSW, the Automatic Brake lever is moved using the ; and ' keys. Like the other levers, there are ranges that require long presses to overcome, and short presses to have fine control.

The Warthog Left axis can work as the automatic brake by setting the Flaps switch to the middle positon (FLAPM). It works the same for both profiles.

GP38-2/SD40-2 Automatic Brake

As defined by TSW, the GP38-2/SD40-2 automatic brake has the following positions:

- Release – release the brakes (recharge the Equalizing Reservoir and Brake Pipe)
- Minimum Reduction – apply a small amount of braking
- Service - a non-notched range of movement until you reach Full Service (full brakes)
- Handle Off – the position to allow removing the brake handle – does not apply any further braking. There are actually two Handle Off positions. TSW does not simulate removing the handle.
- Emergency – dump the equalizing reservoir and apply max brakes to all cars

By "position", I mean these are special positions defined by TSW that do not necessarily exist in a real locomotive. The real locomotive has a notch that you have to overcome to apply Emergency braking, but otherwise, the lever moves freely back and for (as far as I understand) from Release to Handle Off. TSW, however, defines these positions with the difference of using a long press vs. a short press. Long presses are required to move from Release to Minimum Reduction to Service and back. Also, long presses are required to move to and from the Handle Off and Emergency positions. In the GP38-2 and SD40-2, it seems to require two long presses to move through the Handle Off position as there are two Handle off positions.

Again, you can hold down the ; or ' keys and the lever will sweep through all positions without stopping. But if you want to move from Release to Minimum Braking, you need to do a long press (about 200ms) to overcome the "notch" as defined in TSW. In the Service area, the smallest press or tap of the ; and ' keys will move the lever giving precise control.

To give maximum precision, the Warthog Left axis was programmed to only do short presses (50ms). This allows increasing the Service braking by small amounts as needed. The down side of this is, the Warthog will not easily move the automatic brake lever out of Release or Minimum Reduction, or into Handle Off and Emergency. Instead, you must use the Pinky Switch (PSF and PSB). Flip it forward or backwards to perform a single long press and move past the "notches" on the lever. The upside to programming it this way is, you are far less likely to accidentally apply emergency braking when you did not intend.

Procedure to apply the Automatic Brake with the Warthog Left axis (assuming the automatic brake lever is in the Release position):

- Set the Flaps switch to the middle position (FLAPM) to select automatic braking mode.
- Flip the Pinky Switch forward once (PSF) to move to Minimum Reduction. Recenter the Pinky Switch.
- Flip the Pinky Switch forward again (PSF) to move to Service. Recenter the Pinky Switch.
- Slowly push the Warthog Left axis forward to set the desired amount of braking.
    - The Left Axis will not easily go past Full Service to Handle Off. This is normal and desired.

To remove automatic braking:

- Slowly pull the Warthog Left axis back until you reach the minimum Service position.
    - The Warthog will not easily be able to go into the Minimum Reduction position.
- Flip the Pinky Switch backward (PSB) to move into Minimum Reduction. Recenter the Pinky Switch.
- Press PSB again to move to Release. Recenter the Pinky Switch.

To move to Handle Off and back:

- Apply the brakes as stated above.
- At Full Service, flip the Pinky Switch forward (PSF) to move to Handle Off. Recenter the Pinky Switch.
- Press the Pinky Switch back (PSB) to move back to Full Service. Recenter the Pinky Switch.

To apply Emergency Brakes and back:

- Apply the brakes to Handle Off as explained above.
- Flip the Pinky Switch forward (PSF) two more times to reach the Emergency brake.
  - This moves past the second Handle Off position and into the Emergency brake position.
  - Recenter the Pinky Switch each time.
- Flip the Pinky Switch backwards (PSB) twice to move through Handle Off back to Full Service.
  - Recenter the Pinky Switch each time.

This all sounds complicated, but in practice it is pretty simple. If, however, you want a lever that will got right to Emergency Braking by slamming the lever all the way forward, you don't get that here. As emergency braking is not something you should do accidentally, this is a good thing.

You can move the Warthog Left axis quickly through its travel and it will jump over the positions, but it's not enough to go straight to Emergency braking, and it may cause the Warthog lever to get out of sync with the TSW lever. On the other hand, having a precise way to move to Minimum Reduction or the lowest Service position is a benefit because often that's all the braking you want to keep from exceeding a speed limit. Any more, and you end up dropping speed too quickly, and having to make up for it.

If the Warthog Left axis and the Automatic Brake lever get out of sync, you can press and hold the Left Throttle Button (LTB) and it will temporarily disable the Warthog Left axis. You can then move the axis to a new position, release LTB, and it re-enables the axis. Or pause the sim (ESC), readjust the Left Throttle axis, and unpause.

AC4400CW Automatic Brake

The AC4400CW automatic brake works the same as the GP38-2/SD40-2 with one exception. There is no notch between Minimum Reduction and the lowest Service position (it isn't called Service for the AC4400CW). To apply the automatic brake, flip the Pinky Switch forward (PSF) once to move to minimum reduction, then move the Warthog Left axis forward to apply the desired braking. To go back to Release, pull the Left axis all the way back, then flip the Pinky Switch backward (PSB) to move to Release.

Normally for the GP38-2/SD40-2, I use two levers on my Saitek Throttle Quadrant for automatic and independent brakes (see the included Saitek profiles) and use the Warthog Left axis for dynamic braking only. But because the dynamic brake is combined with the throttle on the AC4400CW, I select automatic braking mode for the Left axis (switch the Flaps Switch to the middle position, FLAPM) and use the Left Axis to control the automatic brakes, leaving only the independent brake on the Saitek Throttle Quadrant.

Independent Brake

In TSW, the Independent Brake is moved using the [and] keys. Unlike the other levers, there are no notches to overcome, but there is the Bail position which is a spring loaded position on the lever. Bail out is used to evacuate the brake cylinder only on the engine, releasing the engine brakes, but without affecting or releasing the train/automatic brake.

To apply the Independent Brake with the Warthog Left axis, select the Independent Brake mode by flipping the Flaps Switch backward (FLAPD). Then move the Left axis forward or back as needed. The TSW independent brake lever will follow the Warthog Left axis.

To apply Bail Out, pull the Left Axis all the way back. Then flip the Pinky Switch back (PSB) and it will apply Bail Out long enough to release the locomotive air brakes. Recenter the Pinky Switch.

If the Warthog Left axis and the Independent Brake lever get out of sync, you can press and hold the Left Throttle Button (LTB) and it will temporarily disable the Warthog Left axis. You can then move the axis to a new position, release LTB, and it re-enables the axis. Or pause the sim (ESC), readjust the Left Throttle axis, and unpause.

Views – Mic Switch

The Mic Switch is mapped to control TSW views and enable/disable HUD elements. See the Excel file for all mappings. Here are a few notes on the Mic Switch.

- MSR – In Cab View – TSW 1 key
- MSR Press and Hold – Free View – TSW 8 key
- MSL – Exterior View – TSW 3 key
- MSL Press and Hold – Boom View – TSW 2 key
- MSD – Toggle HUD Markers toggles the 3 icon markers that show next way point, next speed zone, and next signal.
- MSD Press and Hold – HUD Toggle turns all HUD elements on or off (TSW CTRL-1, CTRL-2, CTRL-3)
- MSU – 2D Map View – TSW 9 key
- MSU Press and Hold – Score Toggle (CTRL-6) toggles display of the Score HUD element, but because of what I believe is a bug in TSW, it also toggles the next signal and next speed zone HUD elements. These elements have their own toggle key (CTRL-4) but using them seems to have problems with them not working (a bug). So I just use Score Toggle.

Using the Mic Switch for views, the Coolie Switch as the arrow keys (see below), and some mappings on my Logitech G700S Mouse (see the Excel file) allows me to control most external views and First Person activities without touching the keyboard.

Arrow Keys – Coolie Switch and Slew Control

I normally have my left hand on the Warthog Throttle and right hand on the mouse. I mapped the following Warthog buttons so I don't have to use the keyboard arrow keys to move around. I can quickly switch views using the Mic Switch, and then fly around.

I haven't decided which I like better, so I mapped the arrow keys to both the Coolie HAT switch and the Slew Control. Though I tend to use the Coolie HAT. This controls the UP, DOWN, LEFT, and RIGHT arrow keys for switching views in the cab or moving around in external views.

MSP + the Left or Right Coolie buttons (CSL and CSR), does a CTRL-LEFT ARROW or CTRL-RIGHT ARROW which is used in external views.

The Slew Control center button (SC) maps to the keyboard SHIFT key. When SHIFT is held with an arrow key, you move faster in external views. This also works with First Person walk around mode.

Using the Logitech/Saitek Throttle Quadrant Profiles

Refer to the Excel spreadsheet to see the mapping for the Saitek Throttle Quadrants. Although you can control all the brake levers with the Warthog Left axis, it is much nicer to have a Throttle Quadrant set up so you have three separate levers to control the brakes.

The Saitek left and middle (X and Y) axes were configured to use Directional Axis mode. This mode is very simple. Move the lever up and it presses one key. Move the lever down and it presses another key. The right (Z) axis is not used.

When moving the Throttle Quadrant levers, move them slowly. Unlike the Warthog that guarantees that every key press will be sent to TSW, which keeps things in sync and gives great precision, the Saitek programming software will skip keypresses if you move the lever too quickly and fall short when you reach the end of travel. It works OK for the automatic brake and independent brake, but this way of doing keypresses simply does not work for the throttle or throttle/brake lever in TSW.

The Throttle Quadrant, however, has a nice feature missing on the Warthog – the Reverser position at the bottom of the lever throw. The Warthog has a similar position, but the Throttle Quadrant's position has a physical notch you have to overcome to use it which makes it perfect for the automatic and independent brakes.

Automatic Brake using the Saitek Throttle Quadrant

The Throttle Quadrant's left lever is mapped to move the engine's automatic brake lever, but it uses only short keypresses. It is not intended to overcome the notched areas that require long presses. Instead, the T1 and T2 buttons perform a long press to get past any notches.

When the Left lever is pulled back to the Saitek's Reverser position, it does two long presses which will move the automatic brake from Service to Minimum Reduction to Release. Then when moved out of the Reverser position, two long presses are made again, which moves from Release to Service.

To move to the Minimum Reduction, Handle Off, or Emergency positions, press the T1 or T2 buttons as needed to create the long presses.

Independent Brake using the Saitek Throttle Quadrant

The Throttle Quadrant's middle lever is mapped to move the engine's independent brake lever. It only does short presses as long presses are not needed.

To actuate Bail Out, press the middle lever down into the Saitek's Reverser position. To release Bail Out, pull the lever out of the reverser position.

Technical Details

This section assumes you have some familiarity with TARGET scripting (not the TARGET GUI, but the TARGET Scripting Editor). It also helps to understand C Programming at least a little. If you don't have such experience, don't be afraid to dive in. That's how you learn. But you'll want to keep the TARGET Scripting Documentation handy to understand things.

The information below was written before my final release of the script, so some details may not match the actual script. Refer to the TARGET scripts for the real commands. But the information below is still correct from a learning perspective.

To follow along, open up the .tmc and .ttm TARGET scripts for Train Sim World in the TARGET Script. Or use a program like Notepad++ and tell it that .tmc and .ttm files are C code to add proper coloring. Notepad++ is a much better text editor than using the TARGET Script editor or Windows Notepad. You can edit the files in Notepad++ and load them in the TARGET Script Editor at the same time. Make edits in Notepad++, save the file, and press the RUN button in the TARGET Script Editor to compile and run file. It will always compile and run the saved file even if the open file in TARGET doesn't match anymore. Just never save the file in the TARGET Script Editor. That will over write the saved copy from Notepad++.

TARGET scripting is really C Programming. It uses a group of functions like MapKey() to define the configuration for key presses or KeyAxis() or MapAxis() to change how the analog axis work. These functions make TARGET scripts look somewhat user friendly. In truth, you can do so much more with TARGET by knowing how to write your own functions and execute them either with the EXEC() function, or through the EventHandle() function. It is beyond the scope of this document to explain all the deep details of how TARGET works, but hopefully by reviewing my TSW TARGET script, some light bulbs will go on and you'll see a greater understanding of what TARGET can do and how.

One clue is to understand that every time a button is pressed, or a switch is flipped, or an axis moves on the Warthog, the TARGET driver calls EventHandle(). This is your link to write custom code to take over, and that is where I call the MapThrottleDirectional() function which makes this TSW script work.

EventHandle() takes in three parameters, type, o, and x. Parameter o is the device (e.g. &Throttle). Parameter x is the button or axis that made the event (e.g. MSP for the Mic center push button, or THR_RIGHT for the Warthog Right Throttle axis). Parameter type is not used apparently.

Note in my script in EventHandle(), I have custom code to call Beep() when the MSP button is pressed. Because MSP is the IO shift button, you can't assign a key press or command to it. But we can using EventHandle(). The MSP button is sometimes difficult to press, so Beep() tells when MSP was actually pressed. EventHandle() is also used to press the Push to Talk button for VAC (a voice recognition program), but that command is commented since it is not used in TSW.

The Difficulty of Mapping an Axis to Press a Key in TSW

Currently Train Sim World (TSW) does not support joysticks or other analog game controllers. Many would like to use analog axis levers like the Thrustmaster Warthog Throttle, or Logitech/Saitek Throttle Quadrant to control the locomotive throttle, reverser, and brake levers. Making such devices work with TSW requires turning game controller analog axis movements into key presses. Most programmable game controllers have the ability to do this with "**directional**" programming or "**zone**" programming. The TARGET GUI or TARGET

Script Editor do it for Thrustmaster. Saitek has what used to be called SST or Saitek ST software, but is now a nameless programming tool since Logitech bought Saitek. CH Products, and others have their own versions of this software. Other key mapper programs like AutoHotKeys, FreePIE, and others can do this for game controllers that do not have advanced programming software. Any one of these tools should be able to do Directional or Zone axis mapping.

Locomotives in TSW uses A and D to increase and decrease the throttle respectively. **Directional programming** works such that as you move the analog axis forward, the programming software sends multiple presses of the D key for example, and when moving it backward, it sends multiple presses of the A key. In TARGET, this is done with the KeyAxis() and AXMAP1() commands. KeyAxis() programs an analog axis to press keys, and AXMAP1() defines those keys and how to press them.

Zone programming means that as the lever enters a zone, a certain key is pressed. It is not directional. For example, if there are 3 zones defined, you could have the 1 key pressed in zone 1, nothing pressed in zone 2 (the middle zone), and the 3 key pressed in Zone 3. It doesn't matter which direction the axis moved to reach the zone. When entering that zone, it always presses the assigned key. In TARGET, AXMAP2() is used for this purpose.

But in TSW, there is a problem that makes it difficult to use simple directional or zone programming for all the control levers. Sometimes you need a long press, sometimes a short press, and sometimes a lot of very short presses (taps). This is needed in TSW to overcome the notched positions on certain levers as explained previously. Furthermore, these presses need to be different if the axis is moved forward vs. backward. None of the game controller programming tools listed do this easily if at all (SST and the TARGET GUI, for example cannot do it). TARGET, AutoHotKeys, or FreePIE can do it using custom programming. I chose to use TARGET to do it, but you could replicate everything I have done here in AutoHotKeys or FreePIE as well and support controllers other than Thrustmaster.

Now for some specifics. Follow along in the real Script files as you read this.

Pro_ConfigureSD40GP38() – Control the GP38-2/SD40-2 Throttle Lever

Using the example of the throttle lever, the problem with directional programming is setting how many A or D keypresses to send throughout the full movement of the analog axis. The TSW GP38-2 or SD40-2, for example, have 9 positions on the throttle (idle and steps 1 through 8). If set up correctly, moving the Warthog Right axis through its full movement should send exactly 8 presses of the A key when moved backward, and 8 presses of the D key when moved forward. AXMAP1() does this very well by telling it exactly how many presses you need for the full travel of the axis. You can specify 9 equal zones and every time the Warthog throttle axis crosses a zone, it sends an A keypress if the throttle is moved backward, or a D keypress when moved forward. You can further define the zones to be different sizes with the LIST() command.

Consider the following TARGET command:

```
KeyAxis(&Throttle, THR_RIGHT, 'ioumd', AXMAP1(9, ‘d’, ‘a’));
```

This maps the Warthog Right Throttle axis to have 9 zones evenly divided. Hence each zone takes up about 11% of the throttle movement, or rather every time you move the throttle 11% of its movement, TARGET will send an A as the throttle is moved backward, and a D as it is moved forward.

Now a more complicated example:

```
KeyAxis(&Throttle, THR_RIGHT, 'ioumd', AXMAP1(LIST(0,10,20,30,40,50,60,70,80,100), ‘d’, ‘a’));
```

Here, the LIST() command is used to define the exact size in % for each zone. Each zone is 10% of the Right axis travel except for the last zone, which is 20%. Imagine in this case, the 20% zone is used for the Idle setting on the locomotive throttle, and the other 8 zones are used for the 1 through 8 settings on the locomotive throttle. Using the above commands, it is easy to get a precise movement of the Warthog Throttle to match a precise movement of the TSW locomotive throttle. At least that is true for a simple throttle lever as found in the GP38-2 or SD40-2 engines.

In the TSW TARGET Script, these AXMAP1() commands are implemented in the function pro_ConfigureSD40GP38(). Here is the KeyAxis() command used (find this in the actual script file in your text editor to make it easier to read).

```
KeyAxis(&Throttle, THR_RIGHT, 'ioumd', AXMAP1(LIST(0,11,22,33,44,56,67,78,89,100),
    CHAIN(LOCK+DOWN+Throttle_Decrease, D(120), UP+Throttle_Decrease, D(50), LOCK,
EXEC("Beep(100,20);")),
    CHAIN(LOCK+DOWN+Throttle_Increase, D(120), UP+Throttle_Increase, D(50), LOCK,
EXEC("Beep(100,20);")))
);
```

AXMAP1() is given a LIST() command that delineates 9 zones, 11% per zone approximately. Then CHAIN() commands are used to send the Throttle_Decrease or Throttle_Increase keys (d and a respectively). Here is a breakdown of the CHAIN() command:

- LOCK+ – locks the entire series of actions so they are performed uninterrupted by any other actions.
- DOWN+Throttle_Decrease – Presses the d key down
- D(120) – this is the short press delay, 120ms
- UP+Throttle_Decrease – this releases the d key
- D(50) – this is a short delay to make sure each press of the keys has a small gap between them (50ms).
- LOCK – this unlocks the LOCK+ command from the start of the line.

MapThrottleDirectional() – Control the AC4400CW Throttle/Brake Lever

The commands used for the GP38-2/GP40-2 will not work the GE AC4400CW engine as that engine uses a combined throttle/brake lever. Different length presses (short, vs. long, vs taps) are needed to move the lever forward and backward depending on what part of the lever travel you are on. In the throttle zone, a short press is needed to move exactly one step. In the dynamic braking zone, a tap press (a very short press) is needed to move the lever the smallest possible amount to provide precision as the dynamic brake zone has no notches. To overcome the hump between throttle and dynamic braking requires a long press – a really long press in fact – about 700 to 1300ms.

With TARGET, it is not possible to mix short, long, and tap presses in a single AXMAP1() command. It is possible to do this with AXMAP2(). AXMAP2() is similar to AXMAP1() in that it has zones that can be set up with a LIST() command. As the Warthog axis is moved into each zone, TARGET sends the commanded key for that zone. AXMAP2() allows programming the A key to be pressed for a short time, a long time, or just a tap depending on the zone. Each zone can be set with a different delay for how long the key is pressed. But, AXMAP2() does not do directional programming. I cannot tell TARGET to send one letter, D for example, when moving up and another letter, A for example, when moving down.

The TSW TARGET script uses a custom function, MapThrottleDirectional() to do this. MapThrottleDirectional() programs the Warthog Right axis to do an AXMAP2() zone map that presses D when the Right Throttle is moved forward, and reprograms the Right axis to a different AXMAP2() zone function that presses A when moved backward. Both directions have the same number of zones and zone spacing (percent space between zones), but MapThrottleDirectional() creates a <u>directional</u> version of AXMAP2().

Here is the KeyAxis() definition for when the Right Throttle axis moves forward (find these lines in the actual TARGET script and they will be easier to read than here):

```
KeyAxis(&Throttle, thr_Axis, 'ioumd', AXMAP2(LIST(thr_ZoneList),
    thr_UpAction,thr_UpAction,thr_UpAction,thr_UpAction,thr_UpAction,thr_UpAction,thr_UpAction,th
r_UpAction,  // Engine throttle position 8 to 1
    thr_UpAction,             // Throttle Idle position
    thr_UpActionLong,         // Dead zone - represents the hump between idle positions
    0,                                        // Dynamic Brake Idle position
    thr_UpAction,thr_UpAction,thr_UpAction,thr_UpAction,thr_UpAction,thr_UpAction,thr_UpAction,th
r_UpAction,thr_UpAction,thr_UpAction,
    thr_UpAction,thr_UpAction,thr_UpAction)                // Dynamic brake positions - 13
positions provides greater resolution than just the AC4400 marked positions.
);
```

And here is the KeyAxis() definition for when the Right Throttle axis moves backward:

```
KeyAxis(&Throttle, thr_Axis, 'ioumd', AXMAP2(LIST(thr_ZoneList),
    thr_DownAction,thr_DownAction,thr_DownAction,thr_DownAction,thr_DownAction,thr_DownAction,thr
_DownAction,thr_DownAction,    // Engine throttle position 8 to 1
    0,                                        // Throttle Idle position
    thr_DownActionLong,       // Dead zone - represents the hump between idle positions
    thr_DownAction,                // Dynamic Brake Idle position
    thr_DownAction,thr_DownAction,thr_DownAction,thr_DownAction,thr_DownAction,thr_DownAction,thr
_DownAction,thr_DownAction,thr_DownAction,thr_DownAction,
    thr_DownAction,thr_DownAction,thr_DownAction)        // Dynamic brake positions - 13
positions provides greater resolution than just the AC4400 marked positions.
);
```

They are the same with the exception that one does a thr_UpAction and thr_UpActionLong for each zone defined in AXMAP2() vs. thr_DownAction and thr_DownActionLong. Here is the definition of these actions:

```
define     thr_UpAction        CHAIN(LOCK+DOWN+Throttle_Decrease, D(120), UP+Throttle_Decrease,
D(25), LOCK)                   // Key actions for moving throttle forwards
define     thr_UpActionLong    CHAIN(LOCK+DOWN+Throttle_Decrease, D(700), UP+Throttle_Decrease,
D(25), LOCK)                   // Key actions for moving throttle forwards long press
define     thr_DownAction             CHAIN(LOCK+DOWN+Throttle_Increase, D(120),
UP+Throttle_Increase, D(25), LOCK)                // Key actions for moving throttle backwards
define     thr_DownActionLong  CHAIN(LOCK+DOWN+Throttle_Increase, D(950), UP+Throttle_Increase,
D(25), LOCK)                   // Key actions for moving throttle backwards long press
```

Each keypress is done with a CHAIN() command that includes the following:

- LOCK+ – locks the entire series of actions so they are performed uninterrupted by any other actions.
- DOWN+Throttle_Decrease – Presses the d key down
- D(120) – this is the short press delay, 120ms

- D(700) or D(950) – this is the long press delay for moving past the hump on the throttle/brake lever. 700 and 950 ms respectively.
- UP+Throttle_Decrease – this releases the d key
- D(25) – this is a short delay to make sure each press of the keys has a small gap between them (25ms).
- LOCK – this unlocks the LOCK+ command from the start of the line.

Throttle_Decrease and Throttle_Increase are just defined as ‘d’ and ‘a’ respectively in the .ttm file.

In the previous discussions where I said it may be necessary to change the timing of button presses for your computer vs. mine, it is the above mentioned D() delays that I am referring to. It helps a lot to bring up the TARGET Event Tester (from the TOOLS bar in the TARGET Script Editor) to see these keypresses, delays and key releases. Use the Event Tester while in TSW to see just how long of a press it takes to move a lever when you manually press the keyboard keys. Then see how that delay compares to what the TSW TARGET script is doing. Then adjust the D() numbers to achieve the result. Hopefully this won’t be necessary because hopefully TSW is looking for an exact delay in ms, and not a variable delay that is influenced by computer performance. But just in case, that is why this explanation is given.

The reason for having thr_UpAction and thr_DownAction repeated so many times is because the key press/key release pattern (with LOCK+, DOWN, Delays, UP, and LOCK) has to be repeated for every zone in the AXMAP2() command. It looks messy, but it works.

Note, after I wrote the above, I made a few minor adjustments to the MapThrottleDirectional(), but the concept is still the same.

The zones for AXMAP2() are defined by the LIST() commands as follows:

```
AXMAP2(LIST(thr_ZoneList)), ...
```

Where thr_ZoneList is defined like the following:

```
define     thr_ZoneList
    0,5,9,14,18,23,27,32,36,46,47,57,60,64,67,70,74,77,80,84,87,90,93,97,100
```

There are 24 zones (25 numbers in the list). Each number is the boundary between zones for AXMAP2(). Remember the long dull tone between throttle IDLE and dynamic brake SETUP on the AC4400CW throttle/brake lever? That dull tone is made during the zone marked between 46 and 47 %. The throttle IDLE zone is marked between 36 and 46 %, and the dynamic brake zone is marked between 47 and 57 %. The IDLE and SETUP zones are large on purpose to make sure it is difficult to blow by them too quickly. The long dull tone zone is short (1% wide) on purpose to make sure you always get that dull tone immediately before entering the IDLE or SETUP zones. The tones help a lot to keep in sync with the lever in TSW.

If trying to make this script work for a different engine with combined throttle/brake lever, it may be necessary to change the number of zones as explained above, as well as the gaps between those zones. This is where it is done.

Also, if changes are made to the number of zones in thr_ZoneList, make sure to change the following line:

```
define     thr_NumOfZones      24
```

If there are 25 numbers in the list, there are 24 zones – thr_NumOfZones is always 1 less than the number of numbers in thr_ZoneList. Then there must also be an action for each zone in the KeyAxis() commands explained above. If there are 24 zones, there must be 24 actions.

Note, an action can be 0. This is simply no action. When moving into IDLE or SETUP from the long dull tone zone (called the Dead Zone in the TARGET Script), no action is needed. The Dead Zone performed the action required to get over the hump, and hence there is a 0 in the IDLE or SETUP zone to perform no action.

Profiles in the TSW TARGET Script

The profiles are implemented in the TARGET script using the variable pro_ProfileSelection. This variable is defined as follows:

- pro_ProfileSelection = 1, GP38-2/SD40-2 mode. The Warthog Right axis is programmed for this engine type in the pro_ConfigureSD40GP38() function.
- If pro_ProfileSelection = 2, AC4400CW mode. The Warthog Right axis is programmed for this engine type in the MapThrottleDirectional() function.

Changing profiles is accomplished by pressing the MSP+LDGH buttons. This toggles pro_ProfileSelection between 1 and 2. At startup, pro_ProfileSelection = 1 which selects the GP38-2/SD40-2 profile. This can be changed near the top of the script if desired.

There is a pro_ConfigureAC4400CW() function which is called to do initial setup for the AC4400CW engine, but right now, that function just prints a message saying the profile was selected. If it is desired to do custom key mapping for a specific engine, MapKey() functions should be placed in pro_ConfigureSD40GP38() or pro_ConfigureAC4400CW() for there respective engines. MapThrottleDirectional() is not the place do this since it is called regularly every time the Warthog Right axis is moved, but pro_ConfigureSD40GP38() and pro_ConfigureAC4400CW() are only called when the script is started, or when the MSP+LDGH keys are pressed to change profiles.

Beep() Function – Calling wBeep.exe

Beep() is a function that simply calls wBeep.exe using the system() command. Here is the coding for Beep():

```
//*******************************************************************
// Beep()
//  Beep function - calls wbeep.exe which must be located at the path below.
int Beep(int freq, int duration)            // Beep(frequency, duration in ms)
{
    char buffer;Dim(&buffer, 64);
    sprintf (&buffer, "spawn c:\\bin\\wbeep.exe %d %d", freq, duration);
    system(&buffer);
}
```

The sprintf() function builds the command to be send to the Windows command prompt to call wBeep.exe. The command it executes is:

```
spawn c:\bin\wbeep.exe %d %d
```

Where the %d %d is the frequency and duration of the beep, and spawn is just a command to run the program as it's own process. To use the system() function correctly with a Windows command prompt pathname, it is required to escape the \ with a \\. If the location for wBeep.exe is changes, the path should be modified in the Beep() function, remembering the \\ in place of \.

See the readme file in the "*wBeep files*" directory for more details including the source code of the wBeep.exe program and how to make and compile it yourself.

If it is desired to remove the beeps from this script. The easy way to do it is comment out the "system(&buffer);" line as follows. Then wBeep.exe will not be called, and it does not have to be installed in C:\bin.

```
//spawn c:\bin\wbeep.exe %d %d
```

Conclusion

Most of the rest of the TSW TARGET script is using common methods that are clearly explained in the TARGET Script Editor Basics manual.

Currently I only have TSW: CSX Heavy Haul. When I get other DLC engines, I will update the script to match them if they don't already work with the script as is. I am not so happy with how messy this first script is - it is not modular enough to make adding other engines easy. I will fix that the next time I have to add an engine to the script. Also, as more engines come out for TSW, the idea of using profiles in a single script may have to go as pressing the MSP+LDGH button a dozen times to select that last engine will get ridiculous. I did it this way so one script would still have all the common stuff instead of duplicating effort in two separate scripts. In the future, I'll have to break the common stuff out into a separate file and have a separate main script for each type of engine.

Hopefully, Dove Tail Games will add game controller support to TSW so all this programming is no longer needed. The issues with trying to keep the analog axis in sync with the lever in the engine would not matter because TSW would read the analog axis position directly.

TrackIR Support

You may have noticed in the Excel file that I have "TrackIR Enable/Disable" mapped to a key on my mouse. See FreePIE for this. The FreePIE website has a simple example script that maps TrackIR to act like a mouse. It isn't perfect, but like all this, once you get used to it, it is worth the effort if you are a TrackIR fanatic.

Questions, Comments? Email me…

Michael Lohmeyer
mike@akhara.com