



# TECHNICAL MANUAL

*A complete reference to the loader, all settings, the input services, per game firmware signing and protection, troubleshooting tips, reports and game bring-ups, and the arcade boards it supports.*

*Technical Manual · Resource Guide · Settings Descriptions*

**V1.0.0**

# Contents

|                                                         |    |
|---------------------------------------------------------|----|
| About Ultra Loader (The Manifesto) .....                | 4  |
| Ultra Loader's Three Mechanisms .....                   | 8  |
| The Ultra "Omni" Input Services.....                    | 8  |
| Installation and Folder Layout.....                     | 9  |
| Quick Start.....                                        | 10 |
| Reading the Log .....                                   | 10 |
| The Main UI Window.....                                 | 11 |
| Display and Rendering Options.....                      | 12 |
| Audio Settings .....                                    | 14 |
| Game Settings .....                                     | 15 |
| Controllers and Firmware.....                           | 16 |
| Configure Controls Dialog (Chapter 8 Cont'd).....       | 20 |
| The Omni Input Services (In Detail).....                | 21 |
| Ultra Omni Play (gamepad).....                          | 21 |
| Ultra Omni Steer (flight stick) .....                   | 22 |
| Ultra Omni Wheel (racing wheel) .....                   | 23 |
| Ultra Omni Track (lightgun) .....                       | 24 |
| Ultra Omni Bridge (gamepad to keyboard/mouse).....      | 26 |
| Credits and Coins.....                                  | 27 |
| HOTKEY & BASE COIN .....                                | 27 |
| AUTO INSERT ON GAME BOOT .....                          | 28 |
| ON SCREEN COIN PILL.....                                | 28 |
| Boards and Systems.....                                 | 29 |
| Firmware Signing and Protection .....                   | 31 |
| Ultra.ini Reference (The Settings File).....            | 33 |
| Environment Variables (For the devs) .....              | 38 |
| Basic Troubleshooting.....                              | 43 |
| Ultra Loader Advanced Features.....                     | 44 |
| What the "Enable Advanced Features" option reveals..... | 44 |
| The Hypothesis Engine .....                             | 45 |
| Why I Created This.....                                 | 45 |
| The One Idea That Makes It Work.....                    | 45 |
| What the Engine Is Allowed to Change.....               | 45 |
| The Golden Layer .....                                  | 46 |

|                                                                            |    |
|----------------------------------------------------------------------------|----|
| Before You Start: A Fair Warning.....                                      | 46 |
| Running an Engine Session .....                                            | 46 |
| What the Engine Knows .....                                                | 47 |
| Why It Refuses to Start With a Broken or Insufficient Knowledge Base ..... | 47 |
| Doing Bring up Work by Hand.....                                           | 47 |
| Frequently Hit Walls.....                                                  | 48 |
| Bring up Reports.....                                                      | 49 |
| Why Reports Exist.....                                                     | 49 |
| How to Send a Report.....                                                  | 49 |
| What Is in a Report.....                                                   | 49 |
| What Is NOT in a Report .....                                              | 49 |
| Why Run the Game at Least Twice.....                                       | 50 |
| Please Search Before Retesting.....                                        | 50 |
| ULDP and ULCM: Device Persistence and Connected Mesh .....                 | 51 |
| ULDP: Ultra Loader Device Persistence.....                                 | 51 |
| The Seat Panel.....                                                        | 52 |
| HOLD and FILL: Two Ways to Handle a Missing Controller .....               | 52 |
| When the Original Controller Comes Back.....                               | 52 |
| Manual Seat Controls.....                                                  | 53 |
| What Carries with a Seat .....                                             | 53 |
| Which Boards Support Live Seat Changes .....                               | 54 |
| ULCM: Ultra Loader Connected Mesh.....                                     | 54 |
| Your Cabinet Identity.....                                                 | 54 |
| Discovering Other Cabinets.....                                            | 54 |
| The Three Tiers of Netplay Support.....                                    | 55 |
| Session Flow .....                                                         | 56 |
| Scores and Ghosts for Non-Linkable Games .....                             | 56 |
| What ULCM Does Not Do.....                                                 | 56 |
| In Summary.....                                                            | 56 |
| The “Purge Module Cache” button.....                                       | 57 |

## About Ultra Loader (The Manifesto)

*What Ultra Loader is, what it runs, its overall purpose, and the concept of native execution.*

To begin, Ultra Loader, at its core, is a *native* launcher for modern arcade games. This is, first and foremost, a very *important* distinction. Ultra Loader was not built to emulate in a standard emulator sense. It was built to run modern arcade games in their native environment, running them as if they were built for the machine, in a clean and optimized setting, as if it were sitting in its original home cabinet. This distinction alone, separates Ultra Loader from mostly everything that has come before it.

Arcade launchers, such as teknoparrot and a few others, devise mechanisms to play many of these games at the surface level; giving the game what it needs to run through modified profiles and other tactics that mostly work ok enough for the game, but Ultra Loader's chosen style is not to do this. For all intents and purposes, the loader does things the *hard* way, and when I say the *hard* way, I truly mean it. To get some of these modern arcade games to run, I had to literally go function by function, and bridge the gap between the game and the running environment, all to keep to the concept of *native arcade execution*.

However, I am not the only one doing this. To be fair, the Linux Loader (formerly Lindbergh Loader) guys are seeking to do the same thing as well, and knowing this, I would happily love to combine efforts if our paths do cross one day. I have already personally figured out Lindbergh emulation in Windows, and what it entails, but would love to combine our work to the betterment of both projects, as I find their goal of preserving native arcade cabinets to be admirable. I think this is personally the right way to do things at this point, and this is truly the only way that these games can run and play in their original form and be made available for the whole community...for free.

Over time, we will also discuss more of the underlying motivations behind Ultra Loader (and there is certainly a lot), but I wanted to take some time to talk about what it does. So at the bare minimum, Ultra Loader takes a game that was written for a specific arcade board, let's say, Taito Type X, or Sega Lindbergh, or Namco ES3 and so on, and gets it running on an ordinary PC, supplying the pieces of the cabinet that are missing: the I/O board, the coin mechanism, the dongle or key chip, the display mode, and of course, the controls. This is fairly easy to do if you know what you're doing.

However, what is not easy, (as some boards will quickly show you), is the concept of native emulation for games that were *not designed* for ordinary PC's, such as games designed for specific custom hardware setups, or perhaps games that run in other operating systems. One such popular board, is one we already mentioned above, which is Sega Lindbergh. Lindbergh was built to run in Linux, which is why the Linux Loader guys have been so successful in getting it to run rather smoothly, and any frontend systems that use Linux also benefit from running their core as well, such as Batocera, etc.

However, the challenge is getting all Linux games to run properly in Windows, which has proven to be, for a long time, a bit of a hassle. There are many differences in Linux vs Windows, and it is not so simple to emulate Lindbergh or Raw Thrill for example, in Windows, much less so, to *natively execute* Linux ELF's. However, it can be done, and the Linux Loader guys have done it, and I have also done it. However, it was not easy, and being that I was new to deep coding and programming, it took a lot of painstaking work over many nights, but I have figured out how to natively load linux ELF's in Windows. The process was actually long and tedious, but it does work, and in truth, I really would not have released Ultra Loader if I couldn't figure it out, as this was also the key to unlocking many other cool games that only run on Linux.

Thus I was able to keep to the original premise of native execution, even for boards that do not work natively in Windows or on common PC hardware. Ultra Loader, in this sense, is very strict. My goal is to run all arcade games natively. This means that the path for this loader is slightly harder than others. But this is good, because I think that this is what the community needs. A free and clean alternative, not tied down or dependent on any other software. Currently, all we have are certain types of pseudo-hacks and compatibility layers in order to get many of these games working. But with the advent of

Ultra Loader, now we don't need any of that. All we now need, is time and community dev work on fine tuning the per game code in the board modules, so that each game runs clean and fast, completely free of any other software. As it stands now, Ultra Loader launches all supported games natively, using its custom-built firmware modules without any other barriers. Some games, however, do need special bypasses to make them believe that they are talking to their parent servers that unfortunately no longer exist. Thus I had to take their software, study it and reverse engineer it to insert code to make the game behave as if it is still connected. We can talk about that later when we have discussions on the technical details in real time.

Now as stated above, Ultra Loader is not an emulator, but rather a native execution environment. If you define an emulator as a piece of software that emulates the function of the game by replicating its code in clean room concepts (much of what MAME does) and runs a rom or software package, then perhaps a case can be made to say that this is the same thing, and therefore I am only arguing semantics. However, as I stated before, the mechanism of Ultra Loader is *true* native execution, and especially for the PC-based arcade boards, many of the games are already normal Windows executables, and Ultra Loader runs them directly, injecting a small hook that stands in for the hardware the game expects to find.

This is, for all intents and purposes, ***native execution***. This is the concept behind the loader's full mechanism, and for all games that run on Ultra Loader, this is the standard and bar it must pass. For older non PC-based boards, such as retro titles, it hosts a core instead, but both paths are driven from the same window and the same settings. The reason I chose to develop the loader in this way, was to establish a baseline of clean room execution (which is fully defensible in court if it comes down to it) and to overcome all the myriad workarounds and hacks the community has had to deal with and endure.

The one thing most people will agree on, is that everyone just want things to be easy to use, and to simply *work*. Anyone who plays these games knows of the horrors of modern arcade emulation setups, wrestling with controls and drivers and accompanying soft "warez" and all that kind of other stuff, just to get a single game to boot, which may very well still break later. Teknoparrot's launcher suffers greatly from exactly this. The setup is no less than hell, (even for some savvy users) and a game that works today may not work tomorrow. Indeed, many pieces of software can be accused of this at one time or another in this space, but it need not be this way. We can certainly do better. Then coupled onto that, we have the wonderfully glorious concept of *gatekeeping* and *paywalls* which you will hear me talk a lot about. Quite a few of these games are indeed available and can be played, but of course we have to talk about and discuss the tekno-elephant in the room. Lol, *teknoelephant*.

What I refer to, is the decade long establishment of the false gods. What has been happening with the community over the last ten years has not been good. The ones who run tekno-parrot have profited off of the community in ways it should not have. Say what you will of capitalism and its claims, you should not gatekeep games you do not own. This is simple and clear. I am also a business-minded person, and I currently have a few side hustles going on, all in addition to my studies and my professional career, and I even have quite a few ideas, some of which I am now currently developing, to make money and be successful, and this is what everyone should aim to do if that is your dream and goal. But it cannot and should not be done, by profiting off of, and gatekeeping games that people want to play, especially when you did not *create* them.

These games were developed for arcades, where people can play them with a coin or a ticket. How then, have we allowed one person (and his team?) to monopolize this right under our noses for so long, and we have done little to nothing about it? As it is right now, to play some of these games, you must pay someone who did not make them. This is fundamentally wrong and can *never* be justified. If money is to be paid for games, it should go to the game developer, not a bloody gatekeeper. Many of these games are no longer supported and cared for by their devs. These games should be free to play. It may not be an immensely large amount of games, but it is substantial, and over time, the method itself has become something that the community has rightfully developed ill-will over, and remains a point of contention for many people. Inevitably, it was bound to happen that *someone* would eventually make Ultra Loader.

If, you are like some, who actively support and defend teknoparrot's gatekeeping methods, then you might argue that *someone* had to do it, and in spite of their gatekeeping and profiteering, it funded the development of methods to play these games, and thus it was a necessary evil. But that is a false gods lie. Look at myself. I needed no funding or even access to these physical boards in order to develop Ultra Loader, and as stated before, the modules are entirely clean room, and I did not take source from any other project.

This was one of my hard set rules, just so I could prove that it could be done. All I used was the documented board info and specs available to the public, submitted by hardware preservationists who loved tinkering and playing around with technology, and then I did a massive deep dive into the world of emulation and simply reverse engineered the necessary components. This could have been easily done by anyone. Therefore, the argument that we needed the false gods to do this at the community's expense was all a lie. Many do not know this, but the one called Reaver (the head/owner/creator of teknoparrot) actually took founding work from someone else, and used that to develop the early days of his "arcade drive loader" that launched some of these games in the early days.

The original work was not his. So even then, no one can make the case that they should have done things how they did it, and they had a *right* to charge. And even if that *was* the case, and we needed to pay or "fund" people to do this work, then why not state it as such, and let the community choose a few well-seasoned, and well trusted devs to explore and dig into these boards, and publish their results to the community at large? The community would have happily funded this pursuit, as long as everyone benefited.

A perfect example of this, is the amazing guy that goes by the now famous moniker, "Glorious Eggroll". For those who don't know him, his work with the Proton Compatibility layer, forked specifically to enhance compatibility with many PC games has earned him legendary status, and people happily send him funds and support him, so much so, that he now does it as his full time job, and all his findings go downstream directly to the community with ZERO gatekeeping. So the model works, and could have been implemented for modern arcade games the same way. But no, greed was allowed to hold sway, and many people simply went along with it, as they felt they had no other choice. This, in a sense, is what we allowed, and we shouldn't have.

However, I won't wallow in the past any more than that, because people genuinely felt that these things were a mystery, and only the oldest and wisest amongst us could do it. We truly felt there was no other option. But, as of this new release, that has all changed, and now finally we have another viable and sustainable option, and one that will last. I am just sad that it took ten years for someone to finally build something, and my sorrow is not that we now have an alternative, but that if we had only come together back then and established a better way, shunning this greedy practice we let go on for so long, we would be so much farther along by now. I think that if we had done things differently back then, we would have modern arcade emulation as easy and as smooth as Genesis or SNES emulation today.

Hence my personal stance is this. No one should have the sole authority to gatekeep games to the public, and only a game developer should have that right, as it's their game. I will not drone on about this much longer at this present moment, as we need to get back to the manual, but suffice it to say, the community is eager, ready and calling for a change. Ultra Loader is that change. Ultra Loader will be free, and will always be available to the community. The work done for the loader will benefit all. No one will be charged to play a game, and the only thing preventing a popular game from being played will be the speed at which we can get them running natively in their custom arcade environments. The amazing work that my predecessors did, like the folks who developed Jconfig and the many devs on places like Emuline and other sites, along with the work that many others have put in, have been invaluable to the community, and this has allowed people to play these amazing games outside of teknoparrot's domain.

I am, no, *we* are all deeply grateful for this.

Now while I did not take from Jconfig or any other source (all firmware modules were built from scratch) it is because of their separate work that I was confident enough to do this. I simply told myself that if

they could, then I could at least try, and look what resulted from it. I am also fully aware that Jconfig also has elements of openparrot work, but the fact remains. Ultra Loader has chosen to do things differently, and I am proud to say that the games that are currently supported, run natively, smoothly and fast, with high levels of stability. The loader also offers convenience to the standard of what even a professional would need for their arcade cabinets, and I am very proud of the work that I have done.

It's truly time for a change. The community wants an easy to use tool that just works and won't cost several nights of blood, tears and other bodily fluids just to play a single game. We can do better, and I am happy to be a part of this community, and to bring this new tool to you. I truly hope you enjoy Ultra Loader, and that you come along with me, as we make this thing even more awesome, and more enjoyable and accessible for everyone. First and foremost, and before anything else, this remains my *ultimate* goal.

Now my friends, without further ado, let's dive into the manual.

## Ultra Loader's Three Mechanisms

| PIECE                                                  | WHAT IT IS                                                                                                                                                                                                                                                |
|--------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>The loader itself</b>                               | The program you interact with: ultra loader.exe. It picks a game, works out which board that game belongs to, applies your settings as environment variables, and hands off to a module.                                                                  |
| <b>A custom game firmware module</b>                   | One DLL per arcade board, installed under %LOCALAPPDATA%\Ultra Loader\modules\. It launches the game suspended, injects the board's hook, and waits. Each board owns its own module and its own hook source, so work on one board cannot disturb another. |
| <b>A custom arcade hook (packed with the firmware)</b> | A DLL injected into the running game. It answers the game's requests for JVS I/O, DirectInput, DirectSound, Direct3D and the dongle, essentially all of the hardware that isn't there.                                                                    |

### The Ultra “Omni” Input Services

This is by far my most favorite feature of Ultra Loader, and I loved creating it. There are five “omni” services that run inside the loader and publish a single, shared picture of what the player is doing. Every game's hook reads from that same shared block, which is why one controller setup applies everywhere rather than having to be repeated per game. This is the *solution* to per game setups.

Furthermore, each service comes with a smart “auto calibration” setup, that handles mostly all controllers and gamepads and other HID compliant USB or Bluetooth gameplay devices. I built it specifically to help users simply get into the game and play. No more complicated setups. No more annoying workarounds to setup your controllers, racing wheels or lightguns. Mostly all hardware will be recognized and will work, even *clone* controllers, lol. The ultra omni services does it all for you.

The five services are outlined below:

| SERVICE                  | HANDLES                                                               |
|--------------------------|-----------------------------------------------------------------------|
| <i>Ultra Omni Play</i>   | All Gamepads, Arcade Sticks, Xinput and DirectInput, up to ten slots. |
| <i>Ultra Omni Steer</i>  | Flight sticks and HOTAS setups, including throttle and rudder.        |
| <i>Ultra Omni Wheel</i>  | Racing wheels, Racing pedals and shifters, with full force feedback.  |
| <i>Ultra Omni Track</i>  | Lightguns, plus the full aiming and calibration that goes with them.  |
| <i>Ultra Omni Bridge</i> | Keyboards and Mouses. Many games only work with HID peripherals.      |

As mentioned above, these services auto calibrate almost *everything*. This includes all types of Zero Delay arcade setups and mostly all arcade peripherals and devices. Within the Ultra Omni-Track Lightgun service especially, you will find some super useful options in there, that will effectively eliminate 99% of the problems people currently face when using Lightguns and zero delay encoders for many games. I won't spoil the surprise though. You will have to dive in and see for yourself...

#### Personal Notes:

*The goal here was to effectively eliminate the nightmare of these types of loaders having insufficient or inadequate controls that barely work. The five Omni services cover almost ANYTHING a player would need in order to play mostly any modern arcade game, including in a 2 player or 4 player arcade cabinet. Also, keep in mind, that anything you set in the main window is a global default. But anything a board module sets is a board default, but it gives way to an explicit setting of yours from the main window. If a game behaves differently from what the main window says, a board default override is usually the reason, and a quick look at the ultra.log should tell you everything.*

## Installation and Folder Layout

*Where everything lives in the main folder and on the backend.*

| PATH                                  | WHAT IT HOLDS                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ultra loader.exe                      | The loader itself. Self-contained.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| art\                                  | Icon art by the boot console. Ultra Loader will bundle the icons for most games to be displayed in the loader. As for the art folder, These will be available in the form of a .pak file and you can then install it directly from the main ui menu, and each logo will appear per game. I made available two logo packs. The first one is clear logos. The second one is the well-known silver ring logos that teknoparrot uses. Some people really like them, so I figured I would include them for people to use if they want them. You may also source your own logos, or mix and match if you prefer. Make sure your custom logos are labelled correctly per game, then copy them into the art folder by its system. For example:<br><i>Ultra Loader/art/Mario Kart GP DX.png</i> |
| ultra.ini                             | Sits in the same folder as ultra loader.exe. It contains your settings. <i>Chapter 14</i> lists every key.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| ultra.log                             | Also sits in the same folder as ultra loader.exe. It shows what happened on the last run. This is the first thing to read when something does not work.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| %LOCALAPPDATA%\Ultra Loader\modules\  | Installed board modules, one folder each, plus their signatures and hooks.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| %LOCALAPPDATA%\Ultra Loader\gamedata\ | Per game saved data the loader keeps out of the game folder.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

### ***Your game dumps are never written to, or modified, ever.***

*Ultra Loader treats game folders as read only source material. This is to ensure all dumps keep their full integrity, and also detach reliance on internal file modification, along with future applications. Where a game genuinely must be modified, such as a language switch in its **own** config, the original is copied alongside as <file>.ultra lang bak before the first edit, and the change is always reversible. The loader will also copy the contents of its firmware module to the local cache, (%LOCALAPPDATA%) of which you can clear at any time directly from the loader.*

## Quick Start

*Getting one game running, and knowing whether it worked.*

- 1. Browse to your game collection, and select a game.** Use the game selector in the main window by hitting “Browse” and navigating to your folder where you hold your arcade game dumps, and search for and choose the actual game executable, (game.exe) not a launcher bat script or a shortcut. Shortcuts may or may not work, so prefer you to use the actual executable. Make sure the matching ultra firmware files are copied into the firmware folder by system, so Ultra Loader can find them.
- 2. It will quickly identify the board. (on the backend)** Ultra Loader reads the path and the executable from the firmware to decide which board the game belongs to, and it will say so in the log as [PREFLIGHT]. It can never guess wrong, as the firmware file contains the game’s board information. So even if you were to mistakenly mislabel a game’s firmware file or put it in another arcade system’s folder, it will still find the correct board. This is done in the background, so you won’t have to do anything here.
- 3. 3. Check or modify your display settings.** Most arcade games were written for a fixed resolution. See Chapter 5 for what the game upscaler does, and when to leave it alone for best results.
- 4. Launch the game.** Hit “Save & Launch” near the bottom of the UI to launch the game. The Ultra Loader blue init screen appears first, then the game’s boot console, (if any) and then the game should load. For most games, this is almost instant.

## Reading the Log

When a game does not behave, the *ultra.log* answers most questions on its own. These are the lines worth knowing:

| LOG LINE                            | MEANING                                                                                                          |
|-------------------------------------|------------------------------------------------------------------------------------------------------------------|
| [PREFLIGHT] auto detected system: X | The board Ultra Loader decided on. If this is wrong, nothing after it will be right.                             |
| [SIGCHECK] X: signature OK          | The firmware module passed its signature check and is allowed to load.                                           |
| [DISPATCH] X > ...dll               | The module that actually ran.                                                                                    |
| [FALLBACK] N candidate module(s)    | The first choice module could not load, so others were tried.                                                    |
| game exited with code 0             | A clean exit.                                                                                                    |
| game exited with code 3221225477    | <b>0xC0000005</b> an access violation. The game crashed; it did not quit. This is important for troubleshooting. |

# The Main UI Window

*Choosing a game, and the main window buttons.*

| OPTION                                        | WHAT IT DOES                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Browse...</b><br>ID_BROWSE                 | Pick the game file to launch.                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Game Library</b><br>ID_LIBRARY             | Open the Game Library: a visual grid of your games' artwork. Double click a tile to launch it.                                                                                                                                                                                                                                                                                                                                                      |
| <b>Scan folder for games</b><br>ID_SCAN_GAMES | Scan a folder for supported games and pick from the list.                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>Save &amp; Launch</b><br>ID_LAUNCH         | Save all settings and launch the selected game now. If that game has an Ultra Firmware container, the settings it carries are applied over yours just before launch, so a value the container pins stays pinned no matter what you set here. Your own ultra.ini is not rewritten the override lives only in the running game, so anything the container does not pin is still yours. Containers are only consulted for the native PC arcade boards. |
| <b>Save</b><br>ID_SAVE                        | Save all settings to ultra.ini without launching the game. You should get a confirmation that the setting was saved to ultra.ini                                                                                                                                                                                                                                                                                                                    |
| <b>Cancel</b><br>ID_CANCEL                    | Discard changes and close. Warning: This will exit the loader.                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>Exit</b><br>ID_EXIT                        | Close Ultra Loader without launching anything. Settings you have not saved will be discarded; make sure you hit “Save” first if you want to have them persist to the next launch.                                                                                                                                                                                                                                                                   |

## Display and Rendering Options

*Resolution, upscaling, frame pacing and the screen itself. Every control in the DISPLAY section.*

Arcade games are fixed resolution in a way PC games are not: many ask for exactly 640×480 and will not negotiate with you. Thus in order to render fullscreen or at a higher resolution, we must force the games to do it. The options here work specifically for that, either by asking the game for a larger buffer where it can cope, or by stretching what it produces to fill your screen.

### ***If a game renders black, start here***

*The upscaler and sharpening run as an extra pass over the game's own output. On some titles and some drivers that pass produces a stable black screen while the game itself runs perfectly; audio, inputs and logic are all correct, but no pixels. If that happens, just go and turn the upscaler off and set sharpening to zero before suspecting anything else.*

| OPTION                              | WHAT IT DOES                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Output resolution</b><br>ID_RES  | Output (window/back buffer) resolution. The game renders at its native size and is upscaled to this. 4K looks best with FSR.                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| <b>Display mode</b><br>ID_DMODE     | Windowed = resizable window. Borderless = full screen window (best for alt tab). Fullscreen is acted on by only two boards Namco ES3 (D3D11) and Taito Type X2 where it gives the same borderless window; on every other board it leaves the game's own display request alone, so a title that asks for an exclusive device gets one. Output resolution still matters in Windowed: boards with an upscaler resize the window to it, and the retro cores use it as their windowed size.                                                                                        |
| <b>Upscaler</b><br>ID_UPSCALE       | How the image is scaled to the window. Bilinear = plain. CAS = contrast adaptive sharpen. FSR = AMD FidelityFX (EASU+RCAS), the sharpest. (Yes, I built it into the loader. Cool, eh?)                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>Display fit</b><br>ID_ASPECT     | Stretch fills the screen (but may distort). Pillarbox keeps the 4:3 profile. Zoom fills and crops. Fit to Screen scales with no cutoff. How much of that survives depends on the board: the Lindbergh OpenGL path gets your exact choice, while the injected D3D8 and D3D9 hooks are only told keep or fill Pillarbox becomes keep, and Stretch, Zoom and Fit to Screen all become fill and they apply it only while rotation is running, with 90 or 270 sending keep whatever you picked. The D3D11 hooks read neither value, so on a D3D11 title this setting does nothing. |
| <b>Sharpness</b><br>ID_SHARP        | Upscaler sharpness, 0 100. Affects CAS and FSR's RCAS pass. Higher = crisper edges, too high = ringing.                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>FXAA</b><br>ID_AA                | FXAA post anti-aliasing: smooths jagged edges cheaply. A light blur is normal.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Screen rotation</b><br>ID_ROTATE | Rotate the whole display: 90/180/270. Use 270 (TATE) for vertical monitor cabs. You can also pair this with Ultra Omni Track's Swap X/Y for lightgun games. I know Shmups fans will appreciate this.                                                                                                                                                                                                                                                                                                                                                                          |
| <b>FPS cap</b><br>ID_FPS            | Frame rate cap. 0 = uncapped. Set to your refresh (e.g. 60) to save power and reduce tearing when V Sync is off.                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

| OPTION                                                                             | WHAT IT DOES                                                                                                                                                                                                                                                                  |
|------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>GL driver</b><br>ID_GLDRIVER                                                    | Which OpenGL driver arcade PC titles use. Native = your GPU vendor's driver (fastest). Mesa options are portable software or hardware accelerated alternatives; picking one downloads a small graphics pack the first time.                                                   |
| <b>Frame gen</b><br>ID_FRAMEGEN                                                    | Frame Generation for planned retro cores (PS3, Supermodel, Flycast, MAME, ...). Inserts blended in between frames so a 30fps or 60fps game runs at 60/90/120/180/240fps. Costs ~1 sim frame of latency. 2x is a safe universal pick; 3x/4x would need a high refresh display. |
| <b>V Sync (cap to refresh)</b><br>ID_VSYNC                                         | Synchronize presentation to the monitor refresh to remove tearing. Adds up to one frame of latency. Standard for most games.                                                                                                                                                  |
| <b>Mute splash sound</b><br>ID_SPLASH                                              | Silence the Ultra Loader boot jingle. The splash screen still shows.                                                                                                                                                                                                          |
| <b>Hide console window</b><br>ID_CONSOLE                                           | Hide the diagnostic console window while a game runs. Default is on. Turn off to watch the boot log.                                                                                                                                                                          |
| <b>Easy Config (auto per title)</b><br>ID_EASY                                     | Easy Config auto applies a known good preset (resolution, skips, fixes) for each recognized title on launch.                                                                                                                                                                  |
| <b>UI font</b><br>ID_FONT                                                          | Typeface for this settings UI. Custom arcade fonts ship in the fonts\ folder; Consolas and Segoe UI are built in.                                                                                                                                                             |
| <b>Always show menu on launch</b><br>ID_BOOTMENU                                   | On = always open this menu at launch.<br>Off = boot straight into the remembered game.<br>You must hold the CTRL key on launch to return to the main window.                                                                                                                  |
| <b>Theme</b><br>ID_THEME                                                           | Color theme for the loader UI. Switches live so you can preview each one. Small feature, but fun. My favorite is Tan with Consolas font.                                                                                                                                      |
| <b>Cabinet Mode: Boot directly to game (hold SHIFT to see menu)</b><br>ID_BOOTGAME | Cabinet/attract mode: ALWAYS launch the saved game and never show the menu. Takes precedence over the menu setting. Hold SHIFT while the loader starts to override and open this menu (escape hatch). Useful for dedicated cabinets running one title.                        |

**For some people**, they will use this in an arcade cabinet in attract mode. So I created these functions, so you can simulate a real cab from plug in and turn on, directly to game. You would just automate your OS to boot directly to the desktop, then launch Ultra Loader, and then the game will launch as you saved it.

To help you visualize the settings and how to implement them: here is a brief table:

| Setting: <i>Always Show Menu</i> | Setting: <i>Boot Straight to Game</i> | This Boots To:                        |
|----------------------------------|---------------------------------------|---------------------------------------|
| <b>ON</b>                        | <b>OFF</b>                            | <b>Menu (Boots to main ui window)</b> |
| <b>ON</b>                        | <b>ON</b>                             | <b>Game (SHIFT brings back menu)</b>  |
| <b>OFF</b>                       | <b>ON</b>                             | <b>Game (SHIFT brings back menu)</b>  |
| <b>OFF</b>                       | <b>OFF</b>                            | <b>Game (SHIFT brings back menu)</b>  |

\*If the “Always Show Menu” checkbox is mistakenly unticked (set to off) along with the “Boot Straight to Game” checkbox also set to off, it will force the loader to boot and launch the last known game that was played. If there is no game selected and no game ever run, it will then override and produce the game menu. This will occur until a game is either selected, or the “Boot Straight to Game” checkbox is ticked.

## Audio Settings

*The AUDIO section. Simple, straightforward, cool and normal.*

| OPTION                                              | WHAT IT DOES                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Volume (0 100)</b><br>ID_VOL                     | Master volume, Set it from 0 100.No muss, no fuss. This controls the main game audio, but you may also change in game audio for more loudness or quietness in the game's service menu or settings.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>Menu Sounds</b>                                  | Small but common feature. Mostly cosmetic. As you browse the pause menu, you will hear a navigation tone and a special auditory tone for accept or enter, and one for back or cancel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>Pause audio during pause menu</b><br>ID_PAUSEAUD | Silences the game audio while the in game pause menu is open. The pause menu's audio options page shows the same switch as 'Pause in Menu', and its Save Settings rewrites that one line in ultra.ini and leaves the rest of the file alone, so a change you make in game is what the loader reads next time it starts. Keep in mind, this only mutes the game audio during the Pause menu screen, so it isn't distracting while you make settings changes, but the arcade game will still continue running in the background. If I were to stop the game process while the pause menu remained up, this would cause memory overload issues that would crash the game. I'm working on a way to do this, but as of right now, the game continues to run in the background. |

# Game Settings

*Per title behavior: free play, language, attract mode and save states.*

| OPTION                                             | WHAT IT DOES                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Skip attract movie</b><br>ID_SMOVIE             | Skip the attract/intro movie middleware so games boot straight to playable. Useful for some Lindbergh titles and other boards. Also useful if you want to skip the intro movie and just go straight to the title screen or the coin insert screen.                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Skip audio initialization</b><br>ID_SAUDIO      | Skip sound hardware init. A troubleshooting switch for titles that crash in audio setup.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Skip JVS I/O enumeration</b><br>ID_SJVS         | Skip JVS I/O enumeration. Only for titles that hang when probing for arcade I/O you don't have. Leave this off normally.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Free Play (no coins needed)</b><br>ID_FREEPLAY  | Enable the game's native Free Play so no coins are needed to start. Many people enjoy this feature, but some games won't work with it.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Snapshots / save states</b><br>ID_SNAPSHOTS     | Enable in session save states so you can roll back to any earlier moment in play. Requires a relaunch and a small performance cost. Also, the feature can be buggy. The loader is doing top level memory control in games that do not support it, in order to perform snapshots, so it could potentially result in a crash. The feature works, but use at own discretion.                                                                                                                                                                                                                                                                                                                                             |
| <b>English (If game supports it)</b><br>ID_ENGLISH | Use the game's own English text <i>if</i> it ships it. Two ways, both reversible: flip its language/region config switch in game (Sega Rally 3, Ford Racing, Haunted Museum, Wacky Races), or load its English asset folder instead of the Japanese one (Mario Kart GP DX). Nothing is translated and no game files are modified; any config the loader does change is kept alongside as .ultra lang bak. If the game ships only in Japanese, then it stays in Japanese, and titles with no English at all are left completely alone. Untick to go fully back to the shipped language. Note: If you modify or patch your game with an English patch or mod, then make sure to turn this option <b>OFF</b> (unticked). |
| <b>Enable audio</b><br>ID_AUDIO                    | Master audio enable for the running game. If off, the game may still output sound, but the loader will not send the sound to your speakers. Check this option first if you experience no sound across all your games.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>Boot to console system menu</b><br>ID_BOOT_XMB  | Boot the console's built in system menu instead of the selected game. Available for Retro systems like: PS3 (XMB), PS2 (Browser), Dreamcast (BIOS menu), Chihiro (test menu), Vita (LiveArea), Xbox 360 (NXE). Non console titles are unaffected.*Requires the matching BIOS/firmware to be installed.                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Enable credit injector</b><br>ID_CI_ENABLE      | Universal Insert Coin: press a hotkey (or click the on screen pill) to add a credit function to a game. Works for arcade titles across every supported system (Lindbergh, Type X, in host libretro cores). Free Play, when turned on, silently no ops the injector.                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>Configure...</b><br>ID_CI_CONFIG                | Open the Credit Injector sub panel: choose the insert coin hotkey, credits per coin, and whether to show the on screen COIN pill. A highly requested feature for games that don't have it.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

## Controllers and Firmware

*Why I designed the controls this way, and what the per game modules (.ufw) are and how they work.*

For Ultra Loader, there are several distinct software allocations that decide what happens when you press a button, and each owns its own role. The Omni input services in the loader decides WHICH DEVICES EXIST, how they present, and what state they are in. The Omni services are responsible for enumerating all devices plugged into the backend and connected to your PC, examining and honestly confirming which gamepads, wheels, flight sticks, mouse and keyboard and lightguns are connected, along with which slot each one occupies, as well as how its coordinates are scaled and where its dead zones sit. That work is the same for every game and every board, which is why it lives in the loader and is configured universally. ULDP (please see Chapter 20) is the layer that sits underneath it. More on that later.

Another interconnected and indispensable part of Ultra Loader, and why each game runs as smoothly as it does, is the per-game, custom-built firmware “profile” or container, and that firmware container contains important runtime files and things needed to get each game working. This is proprietary Ultra Loader content, and is derived directly from and by this project. The per-game firmware modules decide what each control means on that game’s so called “cabinet” or environment, and which JVS switch bit a button drives. It also decides which analog channel a pedal feeds, and the calibration window the game’s own saved settings expect. That map travels inside the signed, protected firmware container, so it belongs to that game alone and no other game can see it or use it.

The system board module (contained within the game’s firmware profile, and is also custom and proprietary) is the translator between the two. It takes the container’s map and renders it onto the JVS wire the game reads. It holds no knowledge of any particular game, and that’s the whole point of it. The rule that stops each part from fighting each other, (which is common in such setups) is that where the container has an opinion, the container *always* wins, and this governs the whole input sequence, rather than for a part of it. The loader itself has its own mapping as a fallback for games that possess no internal controls (which is usually rare), and this applies only to titles that ship no map of their own.

Whilst initially developing the loader, I had a chance at many intervals to be lazy and not do this. I was tempted at many times, to simply fall back on publicly available data, and utilize source from already released projects. But I ultimately chose not to. But more importantly, if I had been lazy and done things the easy way, it would have resulted in chaos. No game would work *properly*, and it would then take over *ten years* (as the false gods did) to properly accumulate enough communal game data in order to write individual profiles for each game. Essentially, if I didn’t do things in this way, things would have worked now, but you would have eventually ended up with the hot control setup mess that teknoparrot currently enjoys. Controls would not work for many games, and at best, they would be unstable, as many can now attest, or would require super complicated or convoluted setups just to get a certain game running. Setups would constantly fail. Games would refuse to load, and it would be right back to square one.

This is because one game reads its buttons at the JVS positions that normal convention expects, and the next one puts START on the eighth switch and its VIEW button on the seventh, and its steering on analog on channel two rather than channel zero. This is what causes all of the problems. Each game wants things done their own way, and the solution was never to starve one and ingratiate the other. Each game is unique, and each game is right about themselves. This is where the concept of game profiles come from. So since we currently don’t have a universal layer for all games (which is nothing short of nigh impossible with current means) I have found that only the game can settle what it wants, and so the answer is to simply ship what the game wants for itself. While similar in concept to the “game profile” teknoparrot uses, my implementation and administration of it is both practical and wholly different.

If you are a dev, it would first appear that this answer was a most simple one to reach, but not so with these types of arcade games, especially in the overall process of “bring-up” as I like to call it. However, experiencing failure after failure to develop a universal hook (and trust me, I tried, even after losing all my hard-won per game controls, setting me back by weeks) I realized that this is the only answer. As from the beginning, one of my main goals was to ensure that Ultra Loader was stable and could NOT break on simple, everyday setups. This was primarily the motivation and drive, and this is why all game's controls will live in its own custom firmware module, and not in the loader itself. In addition to this, my other rule (which is solved by the first) is that no dump ever gets written to. *Ever*. My rule is that the arcade dump must stay *clean* (for preservation and future purposes), and as such, all contents and areas for modification must be dumped to a cache from the custom firmware included for the game, and then read back from there.

This also solves many other problems as well. Now, using this method, two cabinets on the same board can never disagree or fight each other for controls. This is one of the problems of teknoparrot as I understand it, although I have not used it in a while, so I am not sure if this was fixed. However, the principle remains. Using my method, no game can grab the lightgun for itself, and then hold it, and another game complains or breaks. All games will co-exist within the loader. All games will run natively and take only from what the omni services provide for them and say thank you after, lol. This is the way. I also wanted to unify things where I could, as many people, myself included, have found it quite difficult to access certain things when needed. So for all games, the thing I did set is that the TEST menu for all games will always be **L3** and the SERVICE menu will always be **R3** on gamepads.

This will ensure that whenever you need to enter the service menu or test menu for any game to calibrate controls or fiddle around, you can always get to it. I did this because I think there should be one standard, enforced across all arcade games as much as possible, where you can reach its internals no matter where you are. My philosophy is that operator controls should not move between titles, and both are deliberate presses on a gamepad that nothing normally uses for most if not all gameplay. Note, I also set credits/coins insert as the SELECT button, and nothing else should insert a credit, unless you set it that way from within the loader.

Now, as for the custom firmware system I employed for each game (and we will talk more about it in Chapter 13), the idea was simply that each game receive its own tiny little environment for gameplay. This pattern is not novel or new, and I didn't come up with it first to be honest. As stated above, even teknoparrot has its per-game .xml profiles, and I believe Jconfig uses .json files. My goal was broader and much more ambitious, in that I wanted to provide a mechanism whereby the community could help “bring up” and improve these games, and I needed to do so in a manner that provided protection and safety for everyone, while still allowing community devs and tinkerers to play.

Thus I developed the individual game firmware system, (for example: Mario Kart.ufw) which the loader loads alongside the game, and it contains the game's micro environment for play, but also in practice, much, much more. The technical details I will leave for the devs and my fellow tinkerers, but each firmware package is tiny, and it comes in the form of a top and bottom layer. The bottom layer is what boots the game and runs it. It is read only and never writable. The loader will only accept this layer for normal play. The top layer, is the tinker/dev area. This area can only be enabled one time per session in the loader, and can never be graduated to the bottom layer.

Devs and tinkerers and experienced coders better than myself may play here and work to improve all games for the community if they so wish and want to help. Once done, there will be an upstream pipeline, where all firmware packages will need to be properly signed before the loader will accept it again. We will talk more about this in Chapter 13, but this is what I came up with in order to ensure that the community is protected and we never deal with the gatekeeping and otherwise preventable problems that we are now facing today. For now, please take a look at the next page to do a deep dive into the loader's buttons and options, and will continue this conversation in Chapter 13.

| OPTION                                       | WHAT IT DOES                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Configure Controls</b><br>ID_INPUTCFG     | Open Controller Configuration: launchers for the Ultra Omni Play (gamepad), Ultra Omni Track (lightgun), Ultra Omni Wheel (racing), Ultra Omni Steer (flight stick) and Ultra Omni Bridge (keyboard and mouse)panels.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>View Controls</b><br>ID_CONTROLS          | Show the control map/button layout for the selected game. The first time you open it for a game the loader probes that game's own executable for its real button names, which takes a second or three the wait is the scan, not a hang and the result is remembered until you pick a different game. If the probe finds nothing, you get Ultra Loader's hand written layout for that title instead.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>Install Game Arts</b><br>ID_INSTALL_ART   | Install game arts/system logos (.pak). Used on the boot console and game tiles. Ultra Loader will include most logos, but you may also source your own if you wish to display custom logos in the title area. Just make sure to label the logos to match the name of the game folder in your game collection, so the loader can recognize it and display the correct logo. Please see Chapter 3 for the art folder. A pack is unpacked into art\<system>\ rather than copied in whole, so you can open the folder and see exactly which logos you have. You are asked Replace or Merge first Replace empties every system folder before extracting, your own logos included, while Merge keeps what is there and only fills the gaps. Case and punctuation are ignored when matching, so pokken_tournament.png and Pokken Tournament.png are the same name, but the name is cut at the FIRST dot keep dots out of it, or Ridge Racer V.2.png and Ridge Racer V.3.png collide. |
| <b>Add System BIOS Files</b><br>ID_BIOS_PACK | Pick a folder of BIOS files. Recognized files are sorted into the right module folders. PS3UPDAT.PUP is auto unpacked and copied into dev_flash automatically by the bundled pup_extract.exe note this covers PS3 only, as PS Vita firmware uses different keys and cannot be unpacked this way. No need for a third party install of any emulator software to install the bios files. Ultra Loader will do it for you. This is a part of the commitment to making things as easy and as simple as possible for users. An easy way to cover everything, is to download a mini bios pack, (many online) and just install that.                                                                                                                                                                                                                                                                                                                                                 |
| <b>View Firmware</b><br>ID_MODULES           | Open the Ultra Firmware container for the game selected in the main window: its board, identity, signature, what is packed inside it, and whether it is clean or carries an unblest test layer, which shows as EDITED. With no game selected, or one that has no container, you get the list of everything installed instead. Browse Firmware reaches any other .ufw on disk, including one you have not installed yet, and Enable Test Mode arms that one container to run unsigned off its test layer for this session every other game stays fully verified.                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>Install Firmware</b><br>ID_INSTALL_CORE   | Install Ultra Firmware containers pick one or more .ufw files, not a DLL. Each one is signature checked before anything is copied, so a container that fails the check is refused here with the reason, instead of failing later at launch where it would look like a broken game. Accepted ones are filed under firmware\<board>\ next to the loader; restart the loader to pick them up.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

| OPTION                                           | WHAT IT DOES                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Uninstall Firmware</b><br>ID_UNINSTALL_CORE   | Shows the firmware you have installed, then opens the loader's firmware folder in Explorer so you can remove a container by deleting its .ufw file. Deleting it is what stops that game running: the board module travels inside the container, so there is no separate module left behind to fall back on. To play that game again you will need to install its container again, so keep a copy.                                                                                    |
| <b>Uninstall BIOS Files</b><br>ID_UNINSTALL_BIOS | Delete BIOS and console firmware files from the installed system folders the retro and console systems that keep a BIOS, since a native arcade board carries what it needs inside its own firmware container. You are shown every file and its size before anything is deleted. Only files whose names are on the loader's known BIOS list are touched: module DLLs, your own files and anything it does not recognize are left exactly where they are. Only bios files are removed. |

## Configure Controls Dialog (Chapter 8 Cont'd)

| OPTION                                          | WHAT IT DOES                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Enable gamepad input</b><br>ID_PLENABLE      | Enable Ultra Omni Play gamepad input. Same master switch as inside the Ultra Omni Play panel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Ultra Omni Play »</b><br>ID_PLAY             | Open the Ultra Omni Play panel: up to 10 gamepads, with per slot dead zone/curve/rumble test, master switches (universal support, rumble, auto cal, trigger threshold), and auto detect for Xbox, PlayStation, Switch Pro, Steam, 8BitDo, Hori, Qanba, Razer, MadCatz, PDD, Power A and Logitech controllers.                                                                                                                                                                                                                                                                                                                                              |
| <b>Enable lightgun input</b><br>ID_LGEN         | Enable Ultra Omni Track lightgun input. Same master switch as inside the Ultra Omni Track panel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>Ultra Omni Track »</b><br>ID_OMNITRACK       | Open the full Ultra Omni Track lightgun panel: per gun brands, 1 8 players, calibration, TATE, off screen reload, and one click Auto Config. Also, stress free calibration.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Enable wheel input</b><br>ID_WHENABLE        | Enable Ultra Omni Wheel racing input. Same master switch as inside the Ultra Omni Wheel panel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>Ultra Omni Wheel »</b><br>ID_WHEEL           | Open the Ultra Omni Wheel panel: up to 4 racing wheels, per slot dead zone/sensitivity/curve/invert, FFB test buttons, and auto detect for Logitech, Thrustmaster, Fanatec, Moza and Simagic racing wheels.                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>Enable flight stick input</b><br>ID_FLENABLE | Enable Ultra Omni Steer flight stick input. Same master switch as inside the Ultra Omni Steer panel.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Ultra Omni Steer »</b><br>ID_FLIGHT          | Open the Ultra Omni Steer panel: up to 4 flight sticks/HOTAS setups, HOTAS pairing (stick + throttle + rudder), per slot dead zone/sensitivity/curve/invert X/Y/Z, throttle reverse detent, FFB, and auto detect for Thrustmaster, Logitech, Saitek, VKB, Virpil, CH, WinWing, Honeycomb, Turtle Beach, Microsoft and PXN flight sticks.                                                                                                                                                                                                                                                                                                                   |
| <b>Enable Keyboard Bridge</b><br>ID_BRENABLE    | For native PC arcade dumps that only read raw keyboard+mouse inputs (no JVS/DirectInput/gun API).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>Ultra Omni Bridge »</b><br>ID_BRIDGE         | Open the Ultra Omni Bridge panel: cursor source (gamepad stick left/right), dead zone, sensitivity, P1/P2 click button mapping, and START/COIN/SERVICE/TEST key taps. Ultra Omni Bridge synthesizes real cursor moves, clicks, and keyboard taps for a connected gamepad or lightgun so those titles are playable, even without a physical mouse and keyboard. It fully maps all controls to your gamepad automatically, without having to custom select the letters on the keypad. However, you may also enter the settings to adjust your custom preference as you like. This is safe to leave on, and does nothing when no gamepad/lightgun is present. |
| <b>OK</b><br>ID_OK                              | Apply all controller changes and close.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Cancel</b><br>ID_CANCEL                      | Discard all controller changes and close.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

## The Omni Input Services (In Detail)

*Gamepads, wheels, flight sticks and lightguns all configured once, then applied to every game.*

The following pages will go into each omni service in detail, along with its per slot settings. This is the immense level of work I put into making sure that people NEVER have to worry about controller setup or compatibility again. I aimed for something like 99% compatibility, and for most people, this should hopefully be enough.

Each service has a master switch plus a row of settings for every slot. The per slot settings are the same for each slot, so they are listed once per service below with the slot number written as N.

### Ultra Omni Play (gamepad)

| OPTION                                                                                                              | WHAT IT DOES                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|---------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Enable Ultra Omni Play gamepad input</b><br>s_op[IOP_ENABLE]                                                     | Master switch. When on, Ultra Omni Play routes every connected gamepad (Xinput slots 0-3 and DirectInput slots 4-9, up to 10 players) into the running game.                                                                                                                                                                                                                                                                                            |
| <b>Auto detect pads (zero setup)</b><br>s_op[IOP_AUTOPROF]                                                          | Zero setup: read each pad's USB VID/PID, identify the brand (Xbox / PlayStation / Switch Pro / 8BitDo / Hori / Qanba / Razer / MadCatz / PDP / Power A / Logitech / Brook / Xin Mo / generic HID / zero delay encoders), and pick sensible dead zone + curve + rumble defaults per slot. Zero delay USB encoder boards, Sanwa and Seimitsu arcade sticks, and Brook fighting boards are all recognized and work plug and play with no additional setup. |
| <b>Universal gamepad support (Xbox / PlayStation / Switch / arcade sticks / generic HID)</b><br>s_op[IOP_UNIVERSAL] | Accept ALL gamepad families at once: Xinput, DirectInput, generic HID pads, arcade sticks, Xbox / PlayStation / Switch Pro / 8BitDo / Hori / Qanba / Razer / MadCatz / PDP / Power A / Logitech. Leave on for plug and play.                                                                                                                                                                                                                            |
| <b>Rumble / force feedback (Xbox motors +Xinput)</b><br>s_op[IOP_RUMBLE]                                            | Global rumble/force feedback gate. On = the service sends motor effects to any pad that supports them (Xbox motors, DualShock/DualSense actuators, Switch Pro HD Rumble). Off = every pad is silent.                                                                                                                                                                                                                                                    |
| <b>Intelligent stick auto calibration (corrects drift + centre offset)</b><br>s_op[IOP_AUTOCAL]                     | One of my favorite options in Ultra Omni Play. This setting will continuously correct stick drift and center point offset. This is for all those who can't afford or don't want to purchase those new fancy hall effect sticks, or still use good, solid older controllers, like myself. Turn this off only for new controllers, and/or for premium pads that already calibrate onboard.                                                                |
| <b>Trigger threshold (1 100%)</b><br>s_op[IOP_TRIG]                                                                 | How far an analog trigger (LT / RT / L2 / R2 / ZL / ZR) must be pulled before it registers as a digital press. Range 1 100%. Lower = hair trigger, higher = firm pull.                                                                                                                                                                                                                                                                                  |
| <b>Save</b><br>s_op[IOP_OK]                                                                                         | Apply and persist Ultra Omni Play settings to ultra.ini, then close.                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Cancel</b><br>s_op[IOP_CANCEL]                                                                                   | Discard changes and close.                                                                                                                                                                                                                                                                                                                                                                                                                              |

**Per slot settings.** One row per slot; N is the slot number.

| OPTION                                          | WHAT IT DOES                                                                                                |
|-------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| <b>Enable slot</b><br>IOP_SLOT0_PLR + slot      | Enable pad slot N (Xinput for slots 1 4, DirectInput for slots 5 10).                                       |
| <b>Brand / status</b><br>IOP_SLOT0_BRAND + slot | Detected pad in slot N, brand + connection status. Auto detect fills this in.                               |
| <b>Dead zone %</b><br>IOP_SLOT0_DZ + slot       | Dead zone % (0 50) for slot N. Ignores tiny stick movement around center; kills drift.                      |
| <b>Response curve %</b><br>IOP_SLOT0_CV + slot  | Response curve % (50 200) for slot N. 100 = linear, <100 = softer near center, >100 = sharper at the edges. |
| <b>Rumble</b><br>IOP_SLOT0_RMB + slot           | Rumble on/off for slot N, overrides the master rumble gate for this pad only.                               |
| <b>Test</b><br>IOP_SLOT0_TEST + slot            | Test: fire a short rumble burst on slot N so you can confirm the pad is alive and mapped to the right slot. |

## Ultra Omni Steer (flight stick)

| OPTION                                                                | WHAT IT DOES                                                                                                                                                                                                                                                        |
|-----------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Enable Ultra Omni Steer flight stick input</b><br>s_os[IOS_ENABLE] | Master switch. When on, Ultra Omni Steer feeds roll/pitch/yaw/throttle + trigger + cabinet buttons into every flight aware system (Lindbergh JVS analog for After Burner Climax, Type X flight cabs, libretro flight cores).                                        |
| <b>Auto detect sticks + HOTAS pairing</b><br>s_os[IOS_AUTOPROF]       | Zero setup: reads each connected stick's USB VID/PID, identifies the brand (Thrustmaster / Logitech / Saitek / VKB / Virpil / CH / WinWing / Honeycomb / Turtle Beach / Microsoft / PXN), and pairs HOTAS splits (stick + throttle + rudder) into one logical slot. |
| <b>Enable force feedback (FFB)</b><br>s_os[IOS_FFBON]                 | Global FFB gate. On = the service sends effects to FFB capable sticks (Microsoft Sidewinder FFB2, Thrustmaster T.16000M FORCE, VKB MCG Ultimate, Virpil Mongoost 50 FFB). Off = every stick is silent.                                                              |
| <b>Save</b><br>s_os[IOS_OK]                                           | Apply and persist Ultra Omni Steer settings to ultra.ini, then close.                                                                                                                                                                                               |
| <b>Cancel</b><br>s_os[IOS_CANCEL]                                     | Discard changes and close.                                                                                                                                                                                                                                          |

**Per slot settings.** One row per slot; N is the slot number.

| OPTION                                           | WHAT IT DOES                                                                                                                                                     |
|--------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Enable slot</b><br>IOS_SLOT0_PLR + slot       | Enable flight stick slot N (HOTAS: stick + throttle + rudder paired here).                                                                                       |
| <b>Brand / status</b><br>IOS_SLOT0_BRAND + slot  | Detected stick in slot N, brand + connection status. Auto detect fills this in.                                                                                  |
| <b>Dead zone %</b><br>IOS_SLOT0_DZ + slot        | Dead zone % (0 50) for slot N. Ignores tiny stick movement around centre.                                                                                        |
| <b>Sensitivity %</b><br>IOS_SLOT0_SN + slot      | Sensitivity % (50 200) for slot N. 100 = 1:1 axis.                                                                                                               |
| <b>Response curve %</b><br>IOS_SLOT0_CV + slot   | Response curve % (50 200) for slot N. 100 = linear, >100 = expo (sharp at edges).                                                                                |
| <b>Invert X</b><br>IOS_SLOT0_IX + slot           | Invert X (roll) for slot N.                                                                                                                                      |
| <b>Invert Y</b><br>IOS_SLOT0_IY + slot           | Invert Y (pitch) for slot N. On = pull back climbs (classic flight sim).                                                                                         |
| <b>Invert Z</b><br>IOS_SLOT0_IZ + slot           | Invert Z (yaw / rudder) for slot N.                                                                                                                              |
| <b>Throttle reverse</b><br>IOS_SLOT0_TREV + slot | Throttle reverse detent for slot N, jet style throttles that pull below idle for reverse thrust. Off = throttle is 0..+1. On = throttle is 1..+1 with idle at 0. |
| <b>FFB test</b><br>IOS_SLOT0_FFB + slot          | Test: fire a short FFB burst on slot N so you can confirm the stick is alive and mapped to the right slot.                                                       |

## Ultra Omni Wheel (racing wheel)

| OPTION                                                          | WHAT IT DOES                                                                                                                                                                                                                           |
|-----------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Enable Ultra Omni Wheel racing input</b><br>s_ow[IOW_ENABLE] | Master switch. When on, Ultra Omni Wheel feeds wheel steer + pedals + shift + cabinet buttons into every wheel aware system (Type X/X2 JVS, Sega Europa R/RingEdge, libretro racing cores).                                            |
| <b>Auto detect wheels (zero setup)</b><br>s_ow[IOW_AUTOPROF]    | Zero setup: reads each connected wheel's USB VID/PID + HID product string, identifies the brand (Logitech G29 / G920 / G923 / DFGT, Thrustmaster T300 / T500 / T150, Fanatec CSL / CSW / DD, Moza, Simagic) and loads its calibration. |
| <b>Enable force feedback (FFB)</b><br>s_ow[IOW_FFBON]           | Send road feel + rumble to wheels that support FFB. Turn off for a quiet wheel or when a broken driver keeps engaging the motor at zero speed.                                                                                         |
| <b>Save</b><br>s_ow[IOW_OK]                                     | Apply and persist Ultra Omni Wheel settings to ultra.ini, then close.                                                                                                                                                                  |
| <b>Cancel</b><br>s_ow[IOW_CANCEL]                               | Discard changes and close.                                                                                                                                                                                                             |

**Per slot settings.** One row per slot; N is the slot number.

| OPTION                                          | WHAT IT DOES                                                                    |
|-------------------------------------------------|---------------------------------------------------------------------------------|
| <b>Enable slot</b><br>IOW_SLOT0_PLR + slot      | Enable wheel slot N.                                                            |
| <b>Dead zone %</b><br>IOW_SLOT0_DZ + slot       | Dead zone % (0 50). Ignores tiny wheel movement around center.                  |
| <b>Sensitivity %</b><br>IOW_SLOT0_SN + slot     | Sensitivity % (50 200). 100 = 1:1 steering.                                     |
| <b>Response curve %</b><br>IOW_SLOT0_CV + slot  | Response curve % (50 200). 100 = linear, >100 = expo (sharp at edges).          |
| <b>Invert</b><br>IOW_SLOT0_INV + slot           | Invert steering axis for slot N.                                                |
| <b>Brand / status</b><br>IOW_SLOT0_BRAND + slot | Detected wheel in slot N, brand + connection status. Auto detect fills this in. |

## Ultra Omni Track (lightgun)

| OPTION                                                                           | WHAT IT DOES                                                                                                                                                                                                                                              |
|----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Enable Ultra Omni Track lightgun input</b><br>s_ot[IOT_ENABLE]                | Master switch. When on, Ultra Omni Track feeds gun aim + buttons into the running game (Lindbergh JVS, Type X / X2 and every libretro core). When off, the loader falls back to pad/keyboard.                                                             |
| <b>Horizontal sensitivity</b><br>s_ot[IOT_XSENS]                                 | Horizontal aim gain, 10 300%. 100% = exact 1:1 tracking. Applied around screen center so it never shifts your zero.                                                                                                                                       |
| <b>Vertical sensitivity</b><br>s_ot[IOT_YSENS]                                   | Vertical aim gain, 10 300%. 100% = exact 1:1 tracking. Applied around screen center.                                                                                                                                                                      |
| <b>X calibration offset</b><br>s_ot[IOT_XOFF]                                    | Horizontal trim, 50..+50. Negative shifts aim left, positive shifts right. Fixes a constant left/right bias.                                                                                                                                              |
| <b>Y calibration offset</b><br>s_ot[IOT_YOFF]                                    | Vertical trim, 50..+50. Negative shifts aim up, positive shifts down.                                                                                                                                                                                     |
| <b>Mirror horizontal axis (invert X)</b><br>s_ot[IOT_INVX]                       | Flip aim left/right. Use for mirror mounted cabs or a gun whose camera is upside down on the X axis.                                                                                                                                                      |
| <b>Mirror vertical axis (invert Y)</b><br>s_ot[IOT_INVY]                         | Flip aim up/down. Use if up and down are reversed (common on some IR mounts).                                                                                                                                                                             |
| <b>Swap X / Y axes : TATE mode</b><br>s_ot[IOT_SWAP]                             | Transpose the aim axes for portrait / rotated (TATE) cabinets so the gun still tracks correctly after the screen is turned 90 degrees.                                                                                                                    |
| <b>Off screen auto reload</b><br>s_ot[IOT_OFFSCR]                                | The classic arcade reload: pull the trigger while pointing OFF the screen to reload. Turn off if a game uses a dedicated reload button instead. Right click always reloads regardless.                                                                    |
| <b>Auto detect gun &amp; load its profile (zero setup)</b><br>s_ot[IOT_AUTOPROF] | Zero setup: on launch Ultra Omni Track reads each connected device's USB VID/PID + product name, matches the gun maker, and auto loads its profile (recoil, border, button map). No manual picking needed. If you have a lot of guns, you will love this. |

| OPTION                                                       | WHAT IT DOES                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Aim border (thin white, auto fit)</b><br>s_ot[IOT_BORDER] | Yes, my dear Sinden fans, I made this one especially for you. This setting will draw Ultra Omni-Track's <i>own</i> thin white border around the screen edge: a present time overlay that auto fits any resolution. This COMPLETELY replaces Sinden's thick border for camera guns. You no longer need Sinden's buggy border software in order to play lightgun games. You can not only draw a thin white border, I also allow you to customize how thin or thick you want it. This is also harmless for IR guns. |
| <b>Select Gun Hardware</b>                                   | This will allow you to select your lightgun manufacturer profile. Selecting this helps Ultra Loader apply the correct settings to your guns. I included profiles for almost every lightgun manufacturer out there, and should work for mostly everyone.                                                                                                                                                                                                                                                          |
| <b>Px size</b><br>s_ot[IOT_BORDERPX]                         | Aim border thickness, 1 40 px. Thin = a few px; it auto scales to your screen resolution. No more adjusting your screen resolution and games to adhere to the border. The border adheres to you.                                                                                                                                                                                                                                                                                                                 |
| <b>Auto Config (smart setup)</b><br>s_ot[IOT_AUTOCFG]        | This setting is made for simple and easy setup. Don't want to fiddle with your lightguns? Click this and Ultra Omni Track detects your attached pointing devices, turns on the lightgun, creates one player per device set to Auto detect, and restores sane aim defaults. You won't ever run into USB device collisions or errors ever again. Each gun gets its own distinct ID.                                                                                                                                |
| <b>Save</b><br>s_ot[IOT_OK]                                  | Apply all Ultra Omni Track settings to ultra.ini, then close.                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Cancel</b><br>s_ot[IOT_CANCEL]                            | Discard changes made in this panel and close.                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

**Per slot settings.** One row per slot; N is the slot number.

| OPTION                                       | WHAT IT DOES                                                                                                               |
|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| <b>Enable player slot</b><br>IOT_PLR0 + slot | Enable Player N's gun slot.                                                                                                |
| <b>Assigned gun</b><br>IOT_ASG0 + slot       | Choose which gun drives Player N. Auto detect binds the next gun that moves; or pin a specific brand profile to this slot. |

## Ultra Omni Bridge (gamepad to keyboard/mouse)

| OPTION                                                                                           | WHAT IT DOES                                                                                                                                                                                                                                                             |
|--------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Enable Keyboard Bridge</b><br>s_ob[IOB_ENABLE]                                                | Master switch. When on, a connected gamepad or lightgun drives synthetic mouse cursor moves, clicks, and key taps for games that only read raw OS keyboard+mouse input (no JVS/DirectInput/gun API). Safe to leave on: does nothing when no gamepad/lightgun is present. |
| <b>Drive cursor from gamepad stick when no lightgun pose is present</b><br>s_ob[IOB_GAMEPAD_SRC] | A real lightgun pose (presence/position) always wins. When no lightgun is present, this lets a connected gamepad's analog stick move the synthetic cursor instead. You can now play your favorite lightgun games with ease using your gamepad.                           |
| <b>Aim with right stick (default: left stick)</b><br>s_ob[IOB_STICK_RIGHT]                       | Which analog stick moves the synthetic cursor. Off = left stick (default). On = right stick.                                                                                                                                                                             |
| <b>Stick deadzone (0 50%)</b><br>s_ob[IOB_DEADZONE]                                              | Ignores tiny stick movement around center before it moves the cursor.                                                                                                                                                                                                    |
| <b>Cursor sensitivity (10 300%)</b><br>s_ob[IOB_SENS]                                            | Stick to cursor speed multiplier. 100% = normal speed.                                                                                                                                                                                                                   |
| <b>Player 2's trigger is the RIGHT mouse button (off = left)</b><br>s_ob[IOB_P2_RIGHT]           | Player 1's trigger is always left click. This picks which physical button P2's trigger maps to. On (default).                                                                                                                                                            |
| <b>Save</b><br>s_ob[IOB_OK]                                                                      | Apply Omni Bridge settings to ultra.ini, then close.                                                                                                                                                                                                                     |
| <b>Cancel</b><br>s_ob[IOB_CANCEL]                                                                | Discard changes and close.                                                                                                                                                                                                                                               |

## Credits and Coins

*Free play, the on screen coin pill, and automatic coin and credit insertion.*

Most arcade games expect a coin mechanism. Ultra Loader can supply one by three ways:

1. By switching the game into its own free play mode where it has one,
2. By giving you a coin button, or
3. By inserting credits automatically when the game boots.

### HOTKEY & BASE COIN

| OPTION                                           | WHAT IT DOES                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Insert coin hotkey</b><br>ID_HK               | You can set the keyboard key that drops a coin while a game is running. The loader watches for this key globally, so it works even though the game owns the keyboard. Just be sure to pick a key the game itself does not use.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>Credits per coin (1 9)</b><br>ID_CPC          | Essentially, how many credits one coin is worth, ranging all the way from 1 to 9. Some games will make you insert 2 coins to get 1 credit. Pokken Tournament is one. There are many others. Some may even ask for up to 4 or more coins for one single gameplay credit (greedy, I know). This setting in fact, mirrors the cabinet operator setting, and a machine set to "2 coins = 1 credit" can also be configured in the game's own test/service menu if it allows it. Some games don't have that option though. So the idea is that if you don't want to do a deep dive into the game's coin/credit mechanism or play around in the test or service menu, you can adjust it here. For most games though, I recommend you leave it at 1 unless you know the game expects otherwise. For instance, in a scenario where you put 2 coins as base instead of 1, for a game that requires 2 coins to establish a gameplay credit, you will have just basically saved yourself a single coin insertion. Minor, but the fun is in the details. If you are establishing a proper arcade cabinet, this can make the difference between 2 clicks of the coin button, or 1. Plus if you so happen to charge per coin, (a super niche scenario) then you might like this option. |
| <b>Player (which slot receives)</b><br>ID_PLAYER | Which player slot the coin is credited to, P1 through P4. On a two player cabinet each side has its own coin slot, and some games will only start the side that was paid for. Again, for most users, this may not be needed, but I wanted to be thorough.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>JVS pulse duration (ms)</b><br>ID_HOLD        | Yes, I went there. I told you, this is the level of detail I threw into this thing. This is an option to adjust how long the simulated coin switch is held closed, in milliseconds (range is 30 200). A normal real life coin mechanism holds the switch for roughly 60ms, which is the default. You can raise the number if a game misses coins on insertion, or lower it if one press (or real life coin insert) registers as two coins. If we are making arcade software that will sit in authentic cabinets, then we gotta do it <i>right</i> , and we gotta do it <i>tight</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

## AUTO INSERT ON GAME BOOT

| OPTION                                                            | WHAT IT DOES                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Automatically insert credits when a game starts</b><br>ID_AUTO | This is a cool feature I thought people might like. This setting drops credits automatically as soon as the game boots, so you never see the insert coin screen. Similar to free play, but this does not disable the insert coin mechanism. It just speeds past it without your input. This is super convenient for a game you are testing repeatedly. You can just turn it off if you want to have normal cabinet behavior. |
| <b>How many credits (1 9)</b><br>ID_AUTO_N                        | How many credits to insert automatically at boot, 1 to 9. Only used when automatic insertion is switched on. Mirrors the Credits Per Coin setting, but specific to the above setting only.                                                                                                                                                                                                                                   |

## ON SCREEN COIN PILL

| OPTION                                                      | WHAT IT DOES                                                                                                                                                                             |
|-------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Show clickable COIN pill during gameplay</b><br>ID_SHOW  | Draw a small clickable COIN button on top of the running game, so you can add credits with the mouse instead of the hotkey. Useful on a desktop; but turn it off on a real cabinet.      |
| <b>Pill position</b><br>ID_POS                              | Which corner the COIN pill sits in: bottom right (default), bottom left, top right or top left. Move it if it covers something the game draws or renders in that corner.                 |
| <b>Show live credit count inside the pill</b><br>ID_SHOWCNT | Print the current credit count inside the pill, so you can see at a glance how many credits the game has banked.                                                                         |
| <b>Play cabinet coin drop sound on insert</b><br>ID_SOUND   | Play a coin drop sound when a credit is inserted, the way a real cabinet clunks. Cosmetic only; it does not change what the game receives. It's just a auditory indication/confirmation. |
| <b>Save</b><br>ID_OK                                        | Apply all controller changes and close.                                                                                                                                                  |
| <b>Cancel</b><br>ID_CANCEL                                  | Discard controller changes and close.                                                                                                                                                    |

## Boards and Systems

*Every arcade board Ultra Loader supports, and what each one needs.*

Ultra Loader identifies the board from the game's path and executable, then loads the matching firmware container. The table below is the loader's full catalogue.

### **Board detection is built in**

*Board keywords are matched against the game's path, using the longest and most specific name first. Please be sure to name your game collections accurately (Taito Type X, Sega RingWide, etc.) however, the loader has been built with smart logic to recognize games by its firmware container's internal header, so in truth, even if you label a game wrong, the loader should still be able to find it. Your art and logos are also tied to this detection system as well, so most logos should show. If a logo is not detected, you may also rename it to give it the best chance to be found.*

| MODULE ID          | BOARD (USE THESE NAMES) | I/O | GRAPHICS  |
|--------------------|-------------------------|-----|-----------|
| 1. sega_lindbergh  | Sega Lindbergh          | JVS | GeForce 7 |
| 2. taito_typex     | Taito Type X            | JVS | GeForce   |
| 3. taito_typex2    | Taito Type X2           | JVS | Radeon    |
| 4. taito_typex3    | Taito Type X3           | JVS | GPU       |
| 5. taito_typex4    | Taito Type X4           | JVS | GPU       |
| 6. taito_typezero  | Taito Type X Zero       | JVS | GPU       |
| 7. taito_typexplus | Taito Type X+           | JVS | GPU       |
| 8. taito_typexzero | Taito Type X Zero       | JVS | GPU       |
| 9. taito_nesica    | Taito NESiCAXLive       | USB | GPU       |
| 10. taito_nesica2  | Taito NESiCAXLive2      | USB | GPU       |
| 11. sega_ringedge  | Sega RingEdge           | JVS | GPU       |
| 12. sega_ringwide  | Sega RingWide           | JVS | GPU       |
| 13. sega_ringedge2 | Sega RingEdge 2         | JVS | GPU       |
| 14. sega_nu        | Sega Nu                 | USB | GPU       |
| 15. sega_nu2       | Sega Nu 2               | USB | GPU       |
| 16. sega_nuc       | Sega Nu SX              | USB | GPU       |
| 17. sega_all       | Sega ALLS               | USB | GPU       |
| 18. sega_all_ux    | Sega ALLS UX            | USB | GPU       |
| 19. sega_europar   | Sega Europa R           | JVS | GPU       |
| 20. namco_n2       | Namco System N2         | JVS | GPU       |
| 21. namco_es1      | Namco System ES1        | JVS | GPU       |
| 22. namco_es2      | Namco System ES2        | JVS | GPU       |
| 23. namco_es2plus  | Namco System ES2+       | JVS | GPU       |
| 24. namco_es3      | Namco System ES3        | JVS | GPU       |
| 25. namco_es4      | Namco System ES4        | JVS | GPU       |

| MODULE ID                  | BOARD (USE THESE NAMES)    | I/O      | GRAPHICS |
|----------------------------|----------------------------|----------|----------|
| 26.namco_bna1              | Namco System BNA1          | JVS      | GPU      |
| 27.konami_pc               | Konami Arcade PC           | I/O      | GPU      |
| 28.konami_bemani           | Konami Bemani PC           | ACIO     | GPU      |
| 29.konami_python           | Konami PYTHON              | JVS      | PS2 GS   |
| 30.cave_pc                 | Cave Arcade PC             | COM      | GPU      |
| 31.exa_arcadia             | exA Arcadia                | JVS      | GPU      |
| 32.unico_pc                | UNICO Arcade PC            | JVS      | GPU      |
| 33.rawthrills_pc           | Raw Thrills Arcade PC      | JVS      | GPU      |
| 34.rawthrills_g5           | Raw Thrills g5             | LPT/FTDI | OpenGL   |
| 35.wahlap_wpc              | Wahlap WPC                 | JVS      | GPU      |
| 36.andamiro_pcplus         | Andamiro PC+               | JVS      | GPU      |
| 37.global_vr               | Global VR System           | JVS      | GPU      |
| 38.igs_pc                  | IGS Arcade PC              | JVS      | GPU      |
| 39.incredible_technologies | Incredible Technologies PC | JVS      | GPU      |
| 40.play_mechanix           | Play Mechanix              | JVS      | GPU      |
| 41.examu_exboard           | Examu eX Board             | JVS      | GPU      |
| 42.generic_pc              | Arcade PC                  | I/O      | GPU      |

## Firmware Signing and Protection

*The most important chapter of this entire manual. Please read this to understand what I did and why.*

Continuing from Chapter 9, where we were discussing per-game firmware modules, I felt it prudent to talk about exactly why I chose to do things in the way I did, for future reference and documentation.

Given the current problem of gatekeeping and profiteering on the community by certain individuals using these modern arcade games, along with the development and potential of this software, I realized that the responsibility fell on me to deliver some form of protection against something like this ever happening again. Call me grandiose. Call me whatever you want to call me. I care about the community, and I care about preservation, and I am committed to doing my part to see that it remains open, free, and accessible by everyone.

Growing up, I would watch my friend's parents take them to fancy arcades and other cool places, and I would dream about being able to go to places like Chucky Cheese's and other fun arcade game places for kids. But, as many other children who are now grown can attest, their parents were poor, and could not afford it. Now, in this current day and age, we have mechanisms to allow people to play these amazing and wonderful games, and we have people gatekeeping and profiteering off of them? No. I simply refuse to accept this. You may charge for your expertise, but not to play games that you did not develop or help create. This is a very real distinction.

On the flip side however, there are people who will happily let you download their content, filling it first with viruses, knowing how deeply you wish to play these games, and they too, are just as bad, if not worse.

So I want to be clear when I say that my reasoning and sentiments for doing this was not merely to prevent or stop the gatekeeping and profiteering, but also very importantly, to protect the community. Many people love playing these games (myself included), and the types of software that launch them usually live on the edge of what most operating systems (looking at *you*, Windows) would report as exploits and viruses, which as anyone knows, is not true, but is still a major problem facing the majority of computer users. Malware exists, and many ill-intentioned people use it to devastating effect.

There are also very malicious actors out there, people who will write code and deliberately target you and your important files, and we all know that the risk of that happening is even more prevalent now, with the rise of AI and other tools made available to the public. I personally also use AI to correct and enhance my work, and I have come to see how easy it would be for *bad actors* to write malicious code. As such, I thought deeply about it, and I decided that I must take steps to ensure that this software is safe, clean and stable for everyone to use, regardless of whether you have knowledge about software or none at all. Everyone should be able to use this software with ease and without fear, knowing that it is fully verified. As such, I developed a firmware *signing* system for all games, permanently locked against the loader.

What this means, is that every game's firmware is signed, and the loader verifies that signature *before* it will load the game. This is not to restrict any user, and it will NOT affect your gameplay. It was built solely for your protection and to give the community a safe and stable way to help bring up these games. The check is also fully enforced, and any module that has been edited or tampered with, even in the slightest of ways, will be refused, and the log will show a [SIGCHECK] error with the reason. This matters greatly if the community is to build upon these modules to run on Ultra Loader.

The per-board system modules will be made source-available, so anyone can read them, build them, and help edit, improve, or cultivate them for use with Ultra Loader. The per-game firmware, however, stays closed and proprietary. The signing mechanism will protect everyone from malicious code writers who will think they now have an opportunity to spread malicious code and viruses to everyone in the community. Furthermore, after careful consideration, I have decided that the loader itself will, for the time being, remain closed source.

This is to ensure that the upstream flow of modules edited and improved by the community can be checked against a clean, sandboxed instance of the loader, and can be scanned for malicious code and viruses, and also fully verified, before pushing it out to the public for download and consumption. Essentially, a fully rebuilt module must be re-signed before it will load the game again. Rebuilding the firmware container alone will leave you with a module that the loader will not touch. This will also prevent random, unsigned modules from being loaded by people who will happily and naively download an untrusted, unverified firmware, and attempt to load it.

The per-game firmware modules (which is separate from the system board modules, like Taito Type X, etc.) are individual community work. The loader will allow you to load modified firmware modules in test mode, which is something you will have to fully opt-into and agree to. I truly want everyone to be safe, but I can't protect people who willingly download an unsigned or unverified module, and then walk into danger and not know what they are doing. For those who *know* what they're doing, this setup will provide you the ability to test your work and confirm that games will run with your improvements. For those who *don't know* what they're doing, the liability then rests with that user, and I cannot be blamed for loading a malicious .dll on your system.

That being said, for most people, this will not be a problem, as most people who tinker, know what they're doing. Furthermore, while it appears that everything is in order, I don't intend to do this alone, as one guy working on all these games will take forever to work through the full catalog, and the community needs to catch up, if we are to stop the gatekeeping and profiteering. I think that the only way forward then, is to trust the community to do the right thing. So as time goes on, we will select (based on community recommendation) a few trustworthy folks, who have a long track record of helping the community, and I will then provide them with the signing keygen and project tools I have developed, and they will have the full authority to verify, build and resign modules for the general public, always giving back and publishing everything for the community benefit.

My plan is not to keep the authority to myself, as I know that this will be the first complaint people will lobby against me. I fully know and realize that openness is the best way to garner trust, and as such, I happily make my custom-built system board modules source-available for community editing, for use with Ultra Loader and on a non-commercial basis, while the per-game firmware stays proprietary. We must do something to protect this new tool, or people will take it, improve upon it, and then create another teknoparrot scenario, or worse, hoard it all for themselves, where they once again monetize access to these games, and people will once more be forced to "pay for play", which is something I am strongly against.

Therefore, the plan (for now) is to have all improved system board modules run upstream to me through the loader's own submission system, and as community devs and knowledgeable people make improvements, I will test and sign the modules and then release them back to the public. This is the best plan I could think of for now, and I am fully open and amenable to any idea or plan of action that anyone has or can suggest, that will offer both protection and assurance of long term open and free community benefit. I hope everyone understands my reasoning behind why I did this, and why for now, the loader itself will remain, mostly closed source.

## Ultra.ini Reference (The Settings File)

*Every key the loader reads and writes, in the order it stores them.*

All settings are saved to the ultra.ini file generated and sitting beside the loader. You can edit it by hand while the loader is closed, and you can also save to it directly from the pause overlay menu. It is also fully rewritten atomically on save, so a crash won't leave a half written file.

Please see the next page for the full .ini settings table with over 100+ settable functions.

| KEY                                      | Default     |
|------------------------------------------|-------------|
| <b>[display]</b>                         |             |
| <b>width</b>                             | <b>1920</b> |
| <b>height</b>                            | <b>1080</b> |
| <b>display_mode</b>                      | <b>1</b>    |
| <b>aspect</b>                            | <b>3</b>    |
| <b>vsync</b>                             | <b>1</b>    |
| <b>fps_cap</b>                           | <b>60</b>   |
| <b>rotate</b>                            | <b>0</b>    |
| <b>upscaler</b>                          | <b>2</b>    |
| <b>sharpness</b>                         | <b>50</b>   |
| <b>antialias</b>                         | <b>1</b>    |
| <b>anisotropy</b>                        | <b>1</b>    |
| <b>cutscene_fill</b>                     | <b>1</b>    |
| <b>framegen</b>                          | <b>0</b>    |
| <b>[loader]</b>                          |             |
| <b>show_splash</b>                       | <b>1</b>    |
| <b>mute_splash_sound</b>                 | <b>0</b>    |
| <b>hide_console</b>                      | <b>1</b>    |
| <b>easy_config</b>                       | <b>1</b>    |
| <b>boot_to_menu</b>                      | <b>1</b>    |
| <b>boot_to_game</b>                      | <b>0</b>    |
| <b>boot_console_menu</b>                 | <b>0</b>    |
| <b>skip_movie</b>                        | <b>1</b>    |
| <b>skip_jvs</b>                          | <b>0</b>    |
| <b>skip_audio</b>                        | <b>0</b>    |
| <b>freeplay</b>                          | <b>0</b>    |
| <b>snapshots</b>                         | <b>0</b>    |
| <b>force_english</b>                     | <b>0</b>    |
| <b>allnet</b>                            | <b>1</b>    |
| <b>allnet_elevate</b>                    | <b>1</b>    |
| <b>[credit_injector]</b>                 |             |
| <b>credit_injector_enable</b>            | <b>0</b>    |
| <b>credit_injector_hotkey</b>            | <b>45</b>   |
| <b>credit_injector_credits_per_coin</b>  | <b>1</b>    |
| <b>credit_injector_show_button</b>       | <b>1</b>    |
| <b>credit_injector_auto_insert</b>       | <b>0</b>    |
| <b>credit_injector_auto_insert_count</b> | <b>1</b>    |
| <b>credit_injector_player</b>            | <b>0</b>    |
| <b>credit_injector_hold_ms</b>           | <b>60</b>   |
| <b>credit_injector_pill_position</b>     | <b>0</b>    |
| <b>credit_injector_show_count</b>        | <b>1</b>    |

|                                       |                        |
|---------------------------------------|------------------------|
| <b>credit_injector_sound_feedback</b> | <b>1</b>               |
| <b>[vita]</b>                         |                        |
| <b>vita_homebrew_mode</b>             | <b>1</b>               |
| <b>vita_verbose_missing_modules</b>   | <b>1</b>               |
| <b>[audio]</b>                        |                        |
| <b>audio_enable</b>                   | <b>1</b>               |
| <b>audio_volume</b>                   | <b>50</b>              |
| <b>pause_game_audio</b>               | <b>1</b>               |
| <b>menu_sounds</b>                    | <b>1</b>               |
| <b>[controllers]</b>                  |                        |
| <b>rumble</b>                         | <b>1</b>               |
| <b>deadzone</b>                       | <b>12</b>              |
| <b>gun_cabinet</b>                    | <b>0</b>               |
| <b>stick_sensitivity</b>              | <b>100</b>             |
| <b>stick_curve</b>                    | <b>0</b>               |
| <b>stick_smooth</b>                   | <b>40</b>              |
| <b>stick_invert_x</b>                 | <b>0</b>               |
| <b>stick_invert_y</b>                 | <b>0</b>               |
| <b>trigger_threshold</b>              | <b>25</b>              |
| <b>autocal</b>                        | <b>1</b>               |
| <b>analog_cal</b>                     | <b>1</b>               |
| <b>lightgun_enable</b>                | <b>1</b>               |
| <b>lightgun_x_sens</b>                | <b>100</b>             |
| <b>lightgun_y_sens</b>                | <b>100</b>             |
| <b>lightgun_x_offset</b>              | <b>0</b>               |
| <b>lightgun_y_offset</b>              | <b>0</b>               |
| <b>lightgun_invert_x</b>              | <b>0</b>               |
| <b>lightgun_invert_y</b>              | <b>0</b>               |
| <b>lightgun_swap_xy</b>               | <b>0</b>               |
| <b>lightgun_offscreen_reload</b>      | <b>1</b>               |
| <b>lightgun_brands</b>                | <b>256</b>             |
| <b>lightgun_players</b>               | <b>1</b>               |
| <b>lightgun_assign</b>                | <b>0,0,0,0,0,0,0,0</b> |
| <b>lightgun_auto_profile</b>          | <b>1</b>               |
| <b>lightgun_border</b>                | <b>0</b>               |
| <b>lightgun_border_thickness</b>      | <b>3</b>               |
| <b>lightgun_show_experimental</b>     | <b>0</b>               |
| <b>universal_gamepad</b>              | <b>1</b>               |
| <b>[wheel]</b>                        |                        |
| <b>wheel_enable</b>                   | <b>0</b>               |
| <b>wheel_auto_profile</b>             | <b>1</b>               |
| <b>wheel_ffb_enable</b>               | <b>1</b>               |

|                                      |                             |
|--------------------------------------|-----------------------------|
| <b>wheel_players</b>                 | <b>1</b>                    |
| <b>wheel_deadzone</b>                | <b>8,8,8,8</b>              |
| <b>wheel_sensitivity</b>             | <b>100,100,100,100</b>      |
| <b>wheel_curve</b>                   | <b>100,100,100,100</b>      |
| <b>wheel_invert_steer</b>            | <b>0,0,0,0</b>              |
| <b>[flight]</b>                      |                             |
| <b>flight_enable</b>                 | <b>0</b>                    |
| <b>flight_auto_profile</b>           | <b>1</b>                    |
| <b>flight_ffb_enable</b>             | <b>1</b>                    |
| <b>flight_players</b>                | <b>1</b>                    |
| <b>flight_deadzone</b>               | <b>6,6,6,6</b>              |
| <b>flight_sensitivity</b>            | <b>100,100,100,100</b>      |
| <b>flight_curve</b>                  | <b>100,100,100,100</b>      |
| <b>flight_invert_x</b>               | <b>0,0,0,0</b>              |
| <b>flight_invert_y</b>               | <b>0,0,0,0</b>              |
| <b>flight_invert_z</b>               | <b>0,0,0,0</b>              |
| <b>flight_throttle_deadzone</b>      | <b>4,4,4,4</b>              |
| <b>flight_throttle_curve</b>         | <b>100,100,100,100</b>      |
| <b>flight_throttle_reverse</b>       | <b>0,0,0,0</b>              |
| <b>[play]</b>                        |                             |
| <b>play_enable</b>                   | <b>1</b>                    |
| <b>play_auto_profile</b>             | <b>1</b>                    |
| <b>play_players</b>                  | <b>1</b>                    |
| <b>play_deadzone</b>                 | <b>10,10,...</b>            |
| <b>play_curve</b>                    | <b>100,100,...</b>          |
| <b>play_rumble</b>                   | <b>1,1,...</b>              |
| <b>[bridge]</b>                      |                             |
| <b>bridge_enable</b>                 | <b>0</b>                    |
| <b>bridge_gamepad_source</b>         | <b>1</b>                    |
| <b>bridge_stick_right</b>            | <b>0</b>                    |
| <b>bridge_deadzone</b>               | <b>15</b>                   |
| <b>bridge_sensitivity</b>            | <b>100</b>                  |
| <b>bridge_p2_button_right</b>        | <b>1</b>                    |
| <b>bridge_vk_start</b>               | <b>49</b>                   |
| <b>bridge_vk_coin</b>                | <b>53</b>                   |
| <b>bridge_vk_service</b>             | <b>0</b>                    |
| <b>bridge_vk_test</b>                | <b>0</b>                    |
| <b>[game]</b>                        |                             |
| <b>game_path</b>                     | <b>(last launched game)</b> |
| <b>library_path</b>                  | <b>(empty)</b>              |
| <b>library_compact_console_tiles</b> | <b>0</b>                    |
| <b>font_name</b>                     | <b>Segoe UI</b>             |

| theme_name | Grey    |
|------------|---------|
| gl_driver  | (empty) |

## Environment Variables (For the devs)

*Overriding any setting from outside, for scripting and for testing.*

Every setting reaches the game as an environment variable, and an explicit variable set before launch wins over both the saved settings and any board default. This is the supported way to script the loader or to test one change without disturbing your configuration.

The loader itself publishes or reads the 150+ variables below. Individual board hooks add many more. Those are documented in the board's own source code and are not user settings.

| VARIABLE                 | REFERENCED IN                                             |
|--------------------------|-----------------------------------------------------------|
| ULTRA_AA                 | d3d8_sharpen.c, d3d8_sharpen.h, d3d8_throttle.c           |
| ULTRA_ALLNET_NETID       | allnet_setup.c                                            |
| ULTRA_ALLNET_SERIAL      | allnet_setup.c                                            |
| ULTRA_ANISO              | glx_emu.c, glx_emu.c, glx_emu.c                           |
| ULTRA_ART_TRACE          | settings.c                                                |
| ULTRA_ASPECT             | d3d8_throttle.c, d3d9_throttle.c, d3d9_throttle_examu.c   |
| ULTRA_AUDIO_VOL          | audio_engine.c, audio_engine.c, audio_engine.c            |
| ULTRA_AUTOCAL            | omni_client.h, main.c, settings.c                         |
| ULTRA_BOOT_AUDIO         | lindbergh_main.c, lindbergh_main.c, main.c                |
| ULTRA_BOOT_CONSOLE_H     | boot_console.h                                            |
| ULTRA_BOOT_CONSOLE_MENU  | main.c                                                    |
| ULTRA_BOOT_GPU           | lindbergh_main.c, lindbergh_main.c, main.c                |
| ULTRA_BOOT_IO            | lindbergh_main.c, lindbergh_main.c, main.c                |
| ULTRA_BOOT_SYSTEM        | lindbergh_main.c, inject_main.c, inject_main.c            |
| ULTRA_BOOT_TITLE         | lindbergh_main.c, inject_main.c, inject_main.c            |
| ULTRA_BOOT_XMB           | main.c                                                    |
| ULTRA_BORDERLESS         | d3d9_throttle_pcplus.c, d3d8_throttle.c, d3d9_throttle.c  |
| ULTRA_BRIDGE_DEADZONE    | hook_dllmain_unico.c, hook_dllmain_unico.c, omni_client.h |
| ULTRA_BRIDGE_ENABLE      | hook_dllmain_pcplus.c, omni_bridge.c, hook_dllmain.c      |
| ULTRA_BRIDGE_GAMEPAD     | hook_dllmain_unico.c, hook_dllmain_unico.c, omni_client.h |
| ULTRA_BRIDGE_P2_RIGHT    | hook_dllmain_unico.c, hook_dllmain_unico.c, omni_client.h |
| ULTRA_BRIDGE_SENS        | hook_dllmain_unico.c, hook_dllmain_unico.c, omni_client.h |
| ULTRA_BRIDGE_STICK_RIGHT | hook_dllmain_unico.c, hook_dllmain_unico.c, omni_client.h |
| ULTRA_BRIDGE_VK_COIN     | omni_client.h, main.c                                     |
| ULTRA_BRIDGE_VK_SERVICE  | omni_client.h, main.c                                     |
| ULTRA_BRIDGE_VK_START    | hook_dllmain_unico.c, hook_dllmain_unico.c, omni_client.h |
| ULTRA_BRIDGE_VK_TEST     | omni_client.h, main.c                                     |

| VARIABLE                  | REFERENCED IN                                                      |
|---------------------------|--------------------------------------------------------------------|
| ULTRA_CAP_GL_PRESENT      | lindbergh_module.c, ultra_module.h, settings.h                     |
| ULTRA_CAS_SHARP           | glx_emu.c, glx_emu.c, glx_emu.c                                    |
| ULTRA_CATALOG_H           | ultra_catalog.h                                                    |
| ULTRA_CI_PULSE            | credit_injector_service.c                                          |
| ULTRA_CI_PULSE_SLOT       | credit_injector_service.c, main.c                                  |
| ULTRA_CURRENT_ABI_VERSION | sign_module.c, module_manifest.h, module_sig_verify.c              |
| ULTRA_CUT_FILL            | glx_emu.c, glx_emu.c, glx_emu.c                                    |
| ULTRA_DISP_MODE           | glx_emu.c, glx_emu.c, glx_emu.c                                    |
| ULTRA_DI_AXIS_            | di_range.h, omniplay_service.c, omniwheel_service.c                |
| ULTRA_DI_AXIS_RX          | di_range.h, di_range.c, omniwheel_service.c                        |
| ULTRA_DI_AXIS_RY          | di_range.h, di_range.c                                             |
| ULTRA_DI_AXIS_RZ          | di_range.h, di_range.c, omniplay_service.c                         |
| ULTRA_DI_AXIS_SLIDER      | di_range.h, di_range.c                                             |
| ULTRA_DI_AXIS_X           | di_range.h, di_range.c, omniwheel_service.c                        |
| ULTRA_DI_AXIS_Y           | di_range.h, di_range.c, omniwheel_service.c                        |
| ULTRA_DI_AXIS_Z           | di_range.h, di_range.c, omniplay_service.c                         |
| ULTRA_FLIGHT_AUTO_PROFILE | omni_client.h, main.c, settings.c                                  |
| ULTRA_FLIGHT_ENABLE       | namco_es3_main.c, omni_client.h, main.c                            |
| ULTRA_FLIGHT_FFB          | omni_client.h, main.c, settings.c                                  |
| ULTRA_FLIGHT_PLAYERS      | main.c, omnisteer_service.c                                        |
| ULTRA_FPS_CAP             | andamiro_pcplus_main.c, d3d9_throttle_pcplus.c, cave_pc_main.c     |
| ULTRA_FREEPLAY            | ftdi_shim.h, ftdi_shim.h, ftdi_shim.h                              |
| ULTRA_FSR_SHARP           | glx_emu.c, glx_emu.c, glx_emu.c                                    |
| ULTRA_FULLSCREEN          | d3d11_throttle_es3.c, d3d9_throttle.c, settings.c                  |
| ULTRA_GAMEDATA            | jvs_serial_hook_adr.c, jvs_serial_hook_pcplus.c, jvs_serial_hook.c |
| ULTRA_GAME_CAL            | jvs_input.c, jvs_input.c, jvs_input.c                              |
| ULTRA_GL_DRIVER           | glx_emu.c, libgl_stub.c, lindbergh_main.c                          |
| ULTRA_HASH_BYTES          | sign_module.c, verify_module.c, module_manifest.h                  |
| ULTRA_HOST_FULLSCREEN     | main.c                                                             |
| ULTRA_HOST_HEIGHT         | main.c                                                             |
| ULTRA_HOST_WIDTH          | main.c                                                             |
| ULTRA_INPUT_PUMP          | ultra_input_pump.h, main.c, ultra_input_pump.c                     |
| ULTRA_IO_ANALOG           | hook_dllmain_pcplus.c, ultra_io_shared_pcplus.h, hook_dllmain.c    |
| ULTRA_IO_ANALOG_OWNED     | ultra_io_shared.h, ultra_io_shared.h, ultra_io_shared.h            |
| ULTRA_IO_COINS            | hook_dllmain_pcplus.c, ultra_io_shared_pcplus.h, hook_dllmain.c    |
| ULTRA_IO_DIAG             | ultra_io_publish.c                                                 |

| VARIABLE                    | REFERENCED IN                                                           |
|-----------------------------|-------------------------------------------------------------------------|
| ULTRA_IO_ENVVAR             | hook_dllmain_pcplus.c, ultra_io_shared_pcplus.h, hook_dllmain.c         |
| ULTRA_IO_FLAG_HOOK_ATTACHED | hook_dllmain_pcplus.c, ultra_io_shared_pcplus.h, hook_dllmain.c         |
| ULTRA_IO_FLAG_LOADER_ALIVE  | ultra_io_shared_pcplus.h, input_src_gun.c, typex2_gun.c                 |
| ULTRA_IO_MAGIC              | hook_dllmain_pcplus.c, ultra_io_shared_pcplus.h, hook_dllmain.c         |
| ULTRA_IO_PLAYERS            | hook_dllmain_pcplus.c, ultra_io_shared_pcplus.h, hook_dllmain.c         |
| ULTRA_IO_PUBLISH            | ultra_io_publish.h, ultra_io_publish.c                                  |
| ULTRA_IO_SHM                | hook_dllmain_pcplus.c, ultra_io_shared_pcplus.h, hook_dllmain.c         |
| ULTRA_IO_VERSION            | ultra_io_shared_pcplus.h, ultra_io_shared.h, ultra_io_shared_generic.h  |
| ULTRA_ISHM_PADS             | ultra_input_shm.h, ultra_input_pump.c, ultra_transport.c                |
| ULTRA_ISHM_STICKS           | ultra_input_shm.h, ultra_input_pump.c, ultra_transport.c                |
| ULTRA_ISHM_WHEELS           | ultra_input_shm.h, ultra_input_pump.c, ultra_transport.c                |
| ULTRA_JVS_LOG               | d3d9_throttle_pcplus.c, hook_dllmain_pcplus.c, jvs_serial_hook_pcplus.c |
| ULTRA_LIGHTGUN_ENABLE       | lindbergh_main.c, hook_dllmain_generic.c, lindbergh_main.c              |
| ULTRA_LIGHTGUN_XOFF         | lindbergh_main.c, lindbergh_main.c, omni_client.h                       |
| ULTRA_LIGHTGUN_XSENS        | lindbergh_main.c, lindbergh_main.c, omni_client.h                       |
| ULTRA_LIGHTGUN_YOFF         | lindbergh_main.c, lindbergh_main.c, omni_client.h                       |
| ULTRA_LIGHTGUN_YSENS        | lindbergh_main.c, lindbergh_main.c, omni_client.h                       |
| ULTRA_LOADER_LOG            | ultra_io_publish.c                                                      |
| ULTRA_LOGO                  | resource_loader.h, settings.c                                           |
| ULTRA_MAX_TRUSTED_KEYS      | trusted_keys.h                                                          |
| ULTRA_MODEL2_CORE           | main.c                                                                  |
| ULTRA_MODULE                | cave_pc_main.c, examu_exboard_main.c, lindbergh_main.c                  |
| ULTRA_MODULE_FALLBACK       | main.c                                                                  |
| ULTRA_MODULE_ID_MAX         | sign_module.c, verify_module.c, module_manifest.h                       |
| ULTRA_MUTE_SPLASH           | main.c, settings.c, splash.c                                            |
| ULTRA_NAMCO_ENGLISH         | jvs_serial_hook_es3.c, settings.c                                       |
| ULTRA_NO_LOGFILE            | main.c                                                                  |
| ULTRA_NO_SPLASH             | lindbergh_main.c, inject_main.c, inject_main.c                          |
| ULTRA_OV_AUTOCLOSE          | pause_menu_service.c                                                    |
| ULTRA_OV_PAGE               | pause_menu_service.c                                                    |
| ULTRA_OV_SHOT               | pause_menu_service.c                                                    |

| VARIABLE                 | REFERENCED IN                                                  |
|--------------------------|----------------------------------------------------------------|
| ULTRA_PAD                | omni_client.h, main.c, settings.c                              |
| ULTRA_PARALLEL_LOAD      | main.c                                                         |
| ULTRA_PAUSE_AUDIO        | settings.c                                                     |
| ULTRA_PLAY_AUTO_PROFILE  | omni_client.h, main.c, settings.c                              |
| ULTRA_PLAY_ENABLE        | omni_client.h, main.c, settings.c                              |
| ULTRA_PLAY_PLAYERS       | main.c, omniplay_service.c                                     |
| ULTRA_PRESENT_FILTER     | glx_emu.c, glx_emu.c, glx_emu.c                                |
| ULTRA_PRESENT_HZ         | settings.h, main.c, settings.c                                 |
| ULTRA_PS3_FW_INSTALLER   | settings.c                                                     |
| ULTRA_RES                | d3d9_throttle_pcplus.c, d3d8_throttle.c, d3d9_throttle.c       |
| ULTRA_RES_H              | d3d9_throttle_pcplus.c, d3d8_throttle.c, d3d9_throttle.c       |
| ULTRA_RES_W              | d3d9_throttle_pcplus.c, d3d8_throttle.c, d3d9_throttle.c       |
| ULTRA_ROTATE             | board_profile.c, d3d8_throttle.c, d3d9_throttle.c              |
| ULTRA_RUMBLE             | omni_client.h, main.c, settings.c                              |
| ULTRA_SCENE_SCALE        | lindbergh_main.c, maintrace.c, maintrace.c                     |
| ULTRA_SEGA_CONTENT_DIR   | truecrypt_bypass.c, truecrypt_bypass_nu2.c, truecrypt_bypass.c |
| ULTRA_SHARPEN            | d3d8_sharpen.c, d3d8_sharpen.h, d3d8_throttle.c                |
| ULTRA_SIG_FORMAT_VERSION | sign_module.c, verify_module.c, module_manifest.h              |
| ULTRA_SIG_MAGIC          | sign_module.c, verify_module.c, module_manifest.h              |
| ULTRA_SKIP_AUDIO         | patcher.c, segaapi_stub.c, patcher.c                           |
| ULTRA_SKIP_JVS           | patcher.c, patcher.c, patcher.c                                |
| ULTRA_SKIP_MOVIE         | patcher.c, patcher.c, patcher.c                                |
| ULTRA_SM_FRAMEGEN        | main.c, settings.c                                             |
| ULTRA_SM_SHARPEN         | settings.c                                                     |
| ULTRA_SM_SHARPNESS       | settings.c                                                     |
| ULTRA_STICK              | omni_client.h, main.c, settings.c                              |
| ULTRA_STICK_DIR_OFF      | ultra_io_publish.c                                             |
| ULTRA_STICK_DIR_ON       | ultra_io_publish.c                                             |
| ULTRA_TOUCH              | hook_dllmain_typex3.c, ultra_touch.h, ultra_touch.c            |
| ULTRA_TOUCH_BTN          | ultra_touch.h, ultra_touch.c                                   |
| ULTRA_TOUCH_CURSOR       | ultra_touch.c                                                  |
| ULTRA_TOUCH_DEADZONE     | ultra_touch.c                                                  |
| ULTRA_TOUCH_HZ           | ultra_touch.h, ultra_touch.c                                   |
| ULTRA_TOUCH_LOG          | ultra_touch.h, ultra_touch.c                                   |
| ULTRA_TOUCH_LX           | ultra_touch.c                                                  |
| ULTRA_TOUCH_LY           | ultra_touch.c                                                  |
| ULTRA_TOUCH_MOUSE        | ultra_touch.c                                                  |
| ULTRA_TOUCH_PAD          | ultra_touch.c                                                  |

| VARIABLE                           | REFERENCED IN                                               |
|------------------------------------|-------------------------------------------------------------|
| ULTRA_TOUCH_PAD_POINTER            | ultra_touch.c                                               |
| ULTRA_TOUCH_RX                     | ultra_touch.c                                               |
| ULTRA_TOUCH_RY                     | ultra_touch.c                                               |
| ULTRA_TOUCH_SLIDE                  | ultra_touch.c                                               |
| ULTRA_TOUCH_SPEED                  | ultra_touch.c                                               |
| ULTRA_TRACE                        | hook_dllmain_adr.c, hook_dllmain.c,<br>hook_dllmain_examu.c |
| ULTRA_TRACE_ERRTAB                 | ultra_tracer.h, ultra_tracer.c                              |
| ULTRA_TRACE_LOG                    | ultra_tracer.h, ultra_touch.c, ultra_tracer.c               |
| ULTRA_TRACE_MAXEXC                 | ultra_tracer.h, ultra_tracer.c                              |
| ULTRA_TRACE_VERBOSE                | ultra_tracer.h, ultra_tracer.c                              |
| ULTRA_TRIGGER_THRESHOLD            | omni_client.h, main.c, settings.c                           |
| ULTRA_UNIVERSAL_GAMEPAD            | omni_client.h, main.c, settings.c                           |
| ULTRA_VITA_FW_REPORT               | vita_firmware_check.h, main.c                               |
| ULTRA_VITA_HOMEBREW_MODE           | main.c                                                      |
| ULTRA_VITA_VERBOSE_MISSING_MODULES | main.c                                                      |
| ULTRA_VSYNC                        | settings.c                                                  |
| ULTRA_WHEEL                        | omni_client.h, main.c, settings.c                           |
| ULTRA_WHEEL_AUTO_PROFILE           | omni_client.h, main.c, settings.c                           |
| ULTRA_WHEEL_ENABLE                 | omni_client.h, main.c, settings.c                           |
| ULTRA_WHEEL_FFB                    | omni_client.h, main.c, settings.c                           |
| ULTRA_WHEEL_PLAYERS                | main.c, omniwheel_service.c                                 |
| ULTRA_WINDOWED                     | d3d9_throttle_pcplus.c, d3d8_throttle.c, d3d9_throttle.c    |

## Basic Troubleshooting

*The failures that come up the most, and what each one usually means.*

| SYMPTOM                                                                          | CAUSE AND FIX                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Game starts, then exits with code <b>3221225477</b>                              | That is 0xC0000005, an access violation. This is a <i>crash</i> , not a quit. Check [PREFLIGHT]first: a game running under the wrong board is the most common cause, because a board hook may carry per game code that is meaningless against another executable.                                                                                                                                                                                                            |
| Game runs perfectly but has <b>no controls at all</b>                            | Before adjusting any mapping, confirm the game's input path exists. A missing dependency DLL, or an input API the hook does not intercept (a game using DirectInput 7 where the hook only handles DirectInput 8), produces exactly this: everything else correct, no input.                                                                                                                                                                                                  |
| Game renders <b>black</b> but audio is heard and game logic runs                 | Turn OFF the upscaler and set sharpening to 0. That post process pass is the usual cause for issues on boards like Taito Type X and Taito Type X2 hardware. Also, if you have the “use English if supported” setting ON, that could potentially throw off a game, so you can also try to turn that off to see if it helps. Some games that do not have English language may not know what to do if the loader is asking for it to render in English, and the game may panic. |
| Game window is <b>small in a corner</b>                                          | The game asked for a fixed backbuffer smaller than the window. Enable the upscaler, or set the output resolution explicitly.                                                                                                                                                                                                                                                                                                                                                 |
| remote LoadLibraryA timed out (wait=0x102)                                       | Injection was blocked, almost always caused by your pesky antivirus behavior shielding. The reason this happens is because writing a DLL and injecting it is the same thing a virus or dropper does. Just add an exception for the Ultra Loader folder and the game folder to your antivirus and that should fix it.                                                                                                                                                         |
| [SIGCHECK] refuses a module                                                      | The module does not match its signature. If you rebuilt or edited a module, you must send it to me, or to a designated signer to get it re-signed before it can be loaded again. This is to protect the community and prevent gatekeeping and exploitation.                                                                                                                                                                                                                  |
| A board module is reported as <b>not installed</b> or the module refuses to load | The loader will say so and fall back to another module, which may run the wrong board's code. Install the correct module rather than relying on the fallback. Also make sure your arcade system folders are labelled correctly. Even a misspelling of Sega RingEdge as Sega Ringege (missed the “d”) can cause the loader not to find the right module, rendering your games unable to load.                                                                                 |
| Game reports itself <b>OFFLINE</b>                                               | Expected for titles with network authentication when the network files are absent from the dump. Most such games are fully playable offline. I may or may not be working on a solution for true <i>universal</i> online multiplayer...but for now, most arcade games play offline just fine.                                                                                                                                                                                 |

## Ultra Loader Advanced Features

Ultra Loader was built with almost everything you could ever need (or mostly ever need) to run your arcade titles. However, I wanted to add even *more* features, to ensure the loader covered everything. Therefore, I created an *advanced features* section, and these are additional diagnostic and community tools that are hidden by default from everyday use.

Go down to the near the bottom of the main ui window, and you will find an “Enable Advanced Features” checkbox. Checking it reveals a new row of controls intended for game bring up work, reporting issues, and community testing. None of these controls are needed for normal game launching, so I hid them behind the checkbox, but they are there for when you want to go deeper.

### What the “Enable Advanced Features” option reveals

|                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| “Hypothesis Engine” button  | This button launches an automated game bring up session for the selected game. It was built with smart logic, and it tries known fixes one at a time, measuring progress against a built-in boot ladder, and writes a full log of everything it attempted. See Chapter 18.                                                                                                                                                                                 |
| “Generate Report” button    | Packages the last few runs from the game log, (at least two) and it includes your current loader settings, as well as the firmware container details into a small zip file. If you also ran the Hypothesis Engine first, the report includes every step it tried and the outcome of each one. Please see Chapter 19.                                                                                                                                       |
| “Advanced Features” button  | This opens up a new menu dedicated to two new exciting features, ULDP™ and ULCM™. Both of these features are meant to make your life easier, and enable features most users want in any arcade software. Please see Chapter 20.                                                                                                                                                                                                                            |
| “Purge Module Cache” button | This button will allow you to clear your module cache for game firmware module data copied to your PC. This comes in handy if you downloaded newer or updated firmware modules to enable a game, but the game still doesn’t work. This is most likely because your system might still be using the old firmware module cache from before, so I placed a “purge module cache” button for users to clear their cache if they need to. Please see Chapter 21. |

# The Hypothesis Engine

This is an automated bring up tool built into the loader. It is mostly just a knowledge base, and will many times not work, but sometimes it can. What it does in a nutshell, is it runs a game repeatedly, applies one change at a time, measures whether each attempt got further than the last, and writes down everything it tried. Read more of the details below if you are interested to know more about it.

## Why I Created This

Around six hundred firmware modules are packaged and untested. One person cannot run them all. The work of getting one arcade game running is unglamorous and highly repetitive: launch the game, read the log, recognize the failure, try the fix that has worked on that failure before, see what changed. Most of a bring up session is not insight, it is that single loop, run twenty times. The Hypothesis Engine automates that loop. It knows what has gone wrong on this project before, what fixed it, and how to measure whether an attempt helped. It will do the first two hours of a person's work in about fifteen minutes, and unlike a person, it writes down every single thing it tried, which is the part that normally disappears the moment someone closes a window.

## The One Idea That Makes It Work

The engine never asks "does the game work?" It asks "did this attempt get further than the last one?" That distinction is the whole design. A game can exit with code 0 the code that formally means "finished normally" while in reality it showed a white screen and crashed. Another family of games crashes at the first frame in an unattended run by design and plays perfectly when a person is watching. Any tool that trusts an exit code will confidently report a dead game as working. "Did it get further?" is more answerable because every board announces the stages it reaches. The engine measures progress against a fixed boot ladder:

launched → module accepted → hook injected → graphics device created  
→ presenting frames → stayed up → controls reaching the game

A run cannot present a frame without having created a device, and it cannot create a device without the hook being inside the game. So the ladder is honest: each rung is a real, verifiable event, and climbing it means something real improved even when nobody yet understands why. Keep in mind: you are the one who can see the screen. The engine can only measure progress on the ladder. You must judge playability once it gets far enough to draw something.

## What the Engine Is Allowed to Change

You choose the scope of the engine at the start of every bring up session. These are the options you can choose from:

**Container only:** Settings and in memory patches, written into a scratch layer inside the game's container file. Your game files and your machine are never touched. This is completely reversible: the scratch layer is cleared when you stop the session.

**+Plus the game folder:** Also allowed to create files the game is looking for and not finding. This reaches a class of failure the container alone cannot solve. Every file created is recorded in a plain text journal inside that folder, and there is an undo function.

**+Plus display and OS:** Also allowed to change resolution, color depth, and compatibility settings per attempt. This reaches the 16 bit fullscreen class of failure, where a 2006 cabinet requests a display mode modern Windows refuses.

## The Golden Layer

Every game firmware module container has two configuration layers (Please see Chapter 9). The golden (bottom) layer is read-only, and both signed and protected, and it is what the game boots from under normal circumstances and is what ships in every firmware module you download. The top layer is unsigned and is only read when a container is explicitly placed in test mode. The hypothesis engine writes only to the top layer. That is why it can be aggressive: nothing it does can affect a normal launch of this game or any other. A session that ends badly leaves the full shipping firmware configuration exactly as it was when you downloaded it. It also means a trial configuration needs no signing key on your machine, which is why none is required to participate in bring up work.

## Before You Start: A Fair Warning

The Hypothesis Engine is a full game debugging tool, not a casual feature. At full scope it launches your game repeatedly, kills it each time, and changes your video mode between attempts. A game killed mid frame can leave your desktop in a corrupted display mode. The engine attempts to restore your display after every attempt including ones it aborted and ones that crashed and it recovers on the next start if it was killed outright. It is still not something to run on a PC you need available at all times. At best, a virtual machine or a spare PC is the right environment for full scope sessions. However this is entirely optional: running the engine on your production PC is fine if you understand what it does and accept the occasional need to reboot.

## Running an Engine Session

### 1. *Select your game*

The engine works on whatever game is currently selected in the Game section. Browse to the game's executable and leave the path there do not press Save & Launch.

If the loader reports that no container covers this game, that is itself worth reporting. A missing container usually means the game folder name does not match what the container expects, which is something only the project can fix.

### 2. *Enable Advanced Features*

Check the Advanced Features box near the bottom of the main window. The Hypothesis Engine button will appear.

### 3. *Press Hypothesis Engine and read the warning*

After reading the warning you will see the scope choice, then the session opening.

### 4. *Read the session opening carefully*

The opening is the most useful thing the engine says all session. It tells you:

**What the container already knows** about this game the executable it expects and the input classification derived from that game's own control list.

**What this board has already taught** the project if someone has got a game running on this board before, those lessons are handed to you before you launch anything.

**Any games solved on a board:** This can teach the engine the controller identity scheme, the report length, and which internal harnesses conflict with the game, and a new game on that board starts with all of that already applied.

**An honest refusal if the engine can't help:** when one applies some boards cannot run a game regardless of configuration, because a physical security chip or TPM binding makes it impossible on any machine other than the original cabinet. The engine names what would actually be needed and lets you decide whether to continue. It will still run a session and the report is still worth having.

## 5. *Work through the ideas*

Each step tells you what the engine thinks is wrong, what it wants to change, and why. You can apply it, skip it, or stop. When you apply one, the game launches and takes over your screen. When it finishes or when the engine stops it you are shown what changed on the boot ladder. Fixes will accumulate. Every attempt the engine makes is the best known configuration for the game, plus one or two new ideas on top. A game may need four or five independent fixes before it can run, and any one of them may not look like progress on its own but each one that helps stays in force for everything that follows. When an attempt gets far enough to draw something on screen the engine hands the decision to you, because you can see the screen and it cannot.

## 6. When it runs out of ideas

Rather than spinning and wasting your time, the engine tells you plainly that it has exhausted its knowledge, states the furthest the game reached, then clears the scratch layer, restores your display, and offers to undo any files it created. A session that found nothing is still worth reporting. See Chapter 19.

## **What the Engine Knows**

Approximately fifty diagnostic rules, each keyed to a string that appears in a real game log and each paired with the change that fixed that failure on this project before. Forty five boards are covered, with thirty five of them carrying a recipe of what games on that board usually need at the start of a session.

Examples of recognized failure classes:

*A game that quits cleanly without ever drawing.* A clean exit means the game failed an internal check and chose not to run. On several boards the usual gate is the network and authentication stack the original cabinet would have had.

*Alive and pumping but never drew a frame.* The watchdog counts frames five times a second, which distinguishes three states a normal exit code cannot: drawing, alive but stuck before first frame (the white screen signature), and wedged.

*Aim that works for a second at boot and then dies.* The game calculates aim inside its own small native picture, but the window was stretched to fill the screen so the pointer ends up outside the strip the game watches. It looks exactly like dead controls and it is not. Checked first on any lightgun game.

*Every analog control dead and staying dead.* If the input layer cannot identify the board, it falls back to a generic profile with fourteen buttons and zero axes. Every wheel, pedal, and stick channel reads a fixed value forever.

*Crashes that belong to the loader, not the game.* A module list read during a file unload, a tracing pass inventing errors, a patching loop briefly removing execute permission from memory pages the engine turns its own optional machinery off before blaming the title.

## **Why It Refuses to Start With a Broken or Insufficient Knowledge Base**

The engine works on prior accumulated knowledge from the previous bring-up work done for the project. It cannot currently predict new ideas or methods to make a game work outside what was loaded into it. Therefore, every setting the engine can propose is checked at startup against the list of settings the loader actually reads. If that self-check fails, the engine refuses to start rather than issuing a warning and continuing on an empty knowledge base that cannot help.

## **Doing Bring up Work by Hand**

As stated above, the engine is not exhaustive. If you want to go further than it can, here are the practices this project uses:

1. **Get a log first.** A run that leaves no evidence has wasted the launch. Every container sets a log path for exactly this reason.
2. **Work out how far it got,** not whether it worked. Use the boot ladder. The question is never "is it broken" it is "what is the last thing that succeeded and what was supposed to happen next."
3. **Read the last run only.** The log may hold many runs. Find the boundary and read from there. Old runs will mislead you, especially if you changed something between them.
4. **Suspect your own tools before the game.** A large share of failures that look like game bugs turn out to be the loader. Turn all optional machinery off and run the game bare. If it survives, the cause was the tools.
5. **Change one thing, then look.** An attempt that changes three settings at once and shows improvement is fine, but you have lost information: you cannot tell which setting mattered, and the next person inherits that uncertainty.
6. **Prefer the least invasive fix.** In order: configure it honestly; assert the flag the missing hardware would have set; retarget a branch keeping the instruction length identical; suppress the error being raised; force a success path as a last resort.
7. **See what shows on screen.** The log may show errors while the game runs perfectly in practice. If it renders and works, you can safely ignore the complaints.

## Frequently Hit Walls

|                               |                                                                                                                                                                                     |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Antivirus blocking the loader | The loader starts a game frozen, injects into it, then unfreezes it. A behavior shield may kill the injection silently. The tell is a log file that exists and is completely empty. |
| Pointing at a .bat launcher   | Pointing the loader at a .bat file in a dump instead of the actual game executable makes it stop before the board module even loads. The loader only runs true executables.         |
| Dump missing files            | The log names the exact path it could not open. This is often the whole answer.                                                                                                     |
| Encrypted content             | Several boards keep their game inside an encrypted volume unlocked by a chip in the cabinet. No setting can reach this. The engine tells you up front if the board has one of them. |

Note: Turning off Advanced Features after using it does not undo anything. Any fixes the Hypothesis Engine applied to the scratch (top writable) layer remain in place unless you explicitly clear them from within the engine itself.

## Bring up Reports

*How to generate a report, what's in it, and why sending one is valuable to the project.*

### Why Reports Exist

The catalog holds almost six hundred modern arcade games. This project is run by one person. Bring up time testing, debugging, troubleshooting a non-working game takes a lot of time and effort per title. Since this is a community project it makes sense to allow the community to participate in the bring up process. Reports are the mechanism for that.

Every line in a report reading "no change" is a cause ruled out, on real hardware, with a real copy of the game. Ruling things out is most of bring up work, and it is the part that normally evaporates the moment someone closes a window. A report full of negative results saves the next person six attempts and teaches the engine its next idea.

### How to Send a Report

Reports do not require a GitHub account, programming knowledge, or any additional software. To send a report, do this:

1. In Ultra Loader, check the Advanced Features box near the bottom of the window.
2. Run the game at least twice so there is something to compare against.
3. Press Generate Report.
4. Press Send It to submit the report to the project.

The same zip file is also written to your Desktop. If you prefer not to send the report from the loader, (for example, if you use a disconnected PC with no internet) you can share the file privately over Discord or any other channel.

### What Is in a Report

|                              |                                                                                                                                                                                                               |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 00 READ THIS FIRST.txt       | Plain language summary of the session.                                                                                                                                                                        |
| 01 report.json               | The same facts in machine readable form.                                                                                                                                                                      |
| 02 what the engine tried.txt | Every Hypothesis Engine step, what it changed, and what the outcome was. Numbered second on purpose: anyone opening the archive sees what has already been eliminated before they start eliminating it again. |
| 03 game log RUNS.txt         | The last few runs from the game's own log, split at run boundaries with the total count stated.                                                                                                               |
| 04 loader log RUNS.txt       | The matching runs from the loader's own log.                                                                                                                                                                  |
| 05 settings.ini              | Your current loader settings at the time of the report.                                                                                                                                                       |
| 06 container.txt             | Which firmware container and board were used.                                                                                                                                                                 |

### What Is NOT in a Report

Your Windows username and other identifying data are never included. Any tool that asks for that information is not a part of this project. Game file contents are never included. Reports contain logs and .ini settings only. Please feel free to open one at any time and examine it. All reports are public. If you still don't feel comfortable with sending a report, despite the lack of any identifying information, then you may refrain from sending with no detriment or issues whatsoever. However, regardless of if you send a report or not, we will never take or record identifying information from your PC.

## **Why Run the Game at Least Twice**

One run cannot distinguish "fails every time" from "failed once" and those are different problems with different fixes. Some titles behave differently on a second launch once logs and other per game state are established. The report notes which attached runs ended the same way and which did not. That single line is often the most useful thing in the file.

## **Please Search Before Retesting**

Before running a session on a game, search the project issues for that game's name. If someone has already tried the obvious approaches their report says so, and the point of this system is that nobody has to cover the same ground twice, and the logs do not inflate with hundreds of reports on the same issue.

## ULDP and ULCM: Device Persistence and Connected Mesh

There are two systems that live behind the “Advanced Features” button. Both are designed around a single principle: the machine should know what you have, and the community should be able to play together. This is ULDP and ULCM. This chapter covers what each system does, how to use it, and where something is not yet built or not yet possible for a specific game. You can access both from the main window. Click “Enable Advanced Features” near the bottom of the loader, then press the “Advanced Features” button that appears. You will then see a modal page that will come up with two cards:

1. Device persistence **ULDP**: Your controllers keep their players, whichever port they are in.
2. Connected mesh **ULCM**: Find other Ultra Loader instances nearby and play together.

Both are inactive unless you’ve turned them on in Advanced Features. Turning it off hides this button and leaves both services idle.

### ULDP: Ultra Loader Device Persistence

ULDP stands for *Ultra Loader Device Persistence*. This is the hardware bound player seat system. It directly addresses and answers one question that every multiplayer arcade setup eventually runs into: why does unplugging and re-plugging your controllers scramble who is Player 1 and who is Player 2?

The short answer is that Windows assigns player slots based on the order USB devices enumerate, and not necessarily on which physical controller they are. Unplug everything and replug in a different order and the game sees a different arrangement. Use a different port and the same thing happens. On a four player cabinet with zero delay encoders, floor vibration alone has unseated players mid-session. This is not a flaw in Ultra Loader or any loader for that matter. It is how every launcher and every game on Windows works, and it has been that way for over twenty years.

However, ULDP solves this. It builds a port independent identity for every HID device connected to your machine derived from the *Windows kernel path*, not from a VID/PID or a USB slot number, and it stores that identity in a file called `devices.ini`. From that point forward, when the loader starts a game, your designated Xbox controller is Player 1 because it is what you designated as the Xbox controller for Player 1, not because it happened to enumerate first. Your zero delay encoder is Player 2 because it is the one plugged into the second encoder port or “Seat” on your panel, not because of which physical USB socket it landed on.

The identity fingerprint uses the kernel path the device tree address that the Windows hardware stack assigns to an interface normalized and lowercased so that the identity that the Setup API, DirectInput, and Raw Input all produce when asking about the same physical device always compare equal. Two physically identical controllers get two different identities if they are in two different paths, which is exactly what you want for a four player encoder board. This is the founding reason the system exists: it can tell two identical gamepads apart. Previous to this, no VID/PID based approach can do that reliably.

While we now have tools that do this conveniently in hardware (shoutout to Pactotech) this problem was always solvable in software. Each device that ULDP has seen, is given a compact display ID. Gamepads read as GP xxxxxxxx 01, zero delay encoders as ZD xxxxxxxx 01, and so on. You will see these IDs in the ULDP panel and in the loader log under the tag [ULDP]. A log line reading “decider=uldp” means ULDP, not USB enumeration order, decided that seat assignment. More on those details on the next page.

## The Seat Panel

When you open ULDP, you will see the seats. The panel shows one row per player slot for the selected role, with three columns: player number, device name and ID, and status.

|           |                                                                                                                                                                                                     |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| connected | The device is present and owns this seat. Nothing can take the slot from the device, and regardless of game settings, that device will always hold this seat as long as it is connected to your PC. |
| held      | The device owns this seat but is not currently connected. The seat is reserved, and will stay reserved for it, indefinitely. Nothing else can take the slot unless you tell the loader to fill it.  |
| empty     | No device has been assigned to this seat. Any unassigned device can claim it.                                                                                                                       |

When a seat is held and its device is absent, the panel shows a note telling you which device is missing and offers a button to allow you to let a spare take it.

## HOLD and FILL: Two Ways to Handle a Missing Controller

When a device goes missing or unplugged, or cable failed, or controller battery goes dead, you have two policies to choose from. You set this in the “When a device goes missing” section of the ULDP panel.

**HOLD** is the default. A seat belongs to its device. If that device is absent the seat is reserved and marked HELD visibly and announced, but remains claimable by pressing a button on a spare, (if set) but never silently taken by something else without your permission. This is the point of ULDP. Player 2 is Player 2 because of which stick Player 2 is holding, not because of which USB port enumerated after the first. A seat that silently moves to a different device is the exact defect the registry exists to prevent.

**FILL** is the opt-in mode. A vacant seat can be claimed by the first unassigned connected device when that seat’s controller goes down or gets disconnected. This is automatic for your convenience once set. No need for a confirmation or a menu. If a gamepad dies mid-tournament, a spare can already be plugged in, and all you need do, is press a button on it and your “sparepad” instantly takes the slot and your play resumes. The **FILL** option never displaces a device that is present. It only fills a seat that has become empty *if* you set it to do so. A presently connected device cannot be knocked out of its seat by any other device pressing a button in FILL mode. That guarantee is hard set in the service.

## When the Original Controller Comes Back

If you were in **FILL** mode and a spare took that now vacant seat, and then the original controller reconnects, (for whatever reason) the loader notifies you. It does this on a transient pill overlay the same kind of non-blocking notification toast the coin injector uses which will appear on-screen without pausing the game and without stealing focus from the game window and without disrupting any input. It names the device by its friendly label and its player number, not by its fingerprint ID. It will simply say: "Player 2's controller is back."

At this point, you are given two choices:

**Restore:** The original arrangement comes back. The returning device takes back its seat. The spare is moved to the lowest free unheld seat so it stays in play as a spare. If there is no free seat for it, the spare becomes unassigned and idle. A user may choose this option more often than not, as the spare may truly just be their backup, and they don’t really enjoy or prefer to main with that secondary spare, so they might choose to go back to the original controller that they first started playing with. The spare may also only be a temporary stop gap, and may not be efficient for gameplay, so as soon

as the original controller can be returned to gameplay (such as if it was able to be charged enough), a user may opt to go back to their original settings.

**Queue:** The spare keeps playing. The original waits silently. The panel will show it as waiting with a “Promote” button. You can then take a moment, and promote back the originally seated controller at any time. However, keep in mind, that if you restart the loader, the original will “sit back down” and own its seat again, as its place was never rewritten, so the next launch simply removes the spare and the original is back home in its rightful place.

Also of note, if the toast prompt expires without an answer, I programmed the service to queue the original controller, rather than bumping the spare. This is a deliberate asymmetry. Bumping a connected controller without an answer (regardless of spare or main) can ruin a match or play in progress, as the player may not have chosen that in the moment. Queueing costs only one action later, like between rounds or at the next launch. In this, the prompt expires to the safer outcome.

If however, you don’t want to be asked every time a controller disconnects and comes back, the “*When the original comes back*” dropdown has three options:

1. *Ask me each time (the default):* you decide what happens when a controller comes back each time with the toast.
2. *Always restore my arrangement:* the loader restores your set configuration automatically and shows a brief confirmation with an Undo action.
3. *Always queue for promotion:* the loader queues automatically and shows a brief confirmation with a Promote action.

In all three cases, you will still get a notification of what occurred, so you know what is happening behind the scenes.

## Manual Seat Controls

The bottom of the ULDP panel has four buttons for managing seats directly:

|               |                                                                                                                                                                                             |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Rename        | Give a device a friendlier name than its hardware ID. The name appears in the seat table, in notifications, and in the loader log.                                                          |
| Reassign      | Move a device to a different player slot. The change is written to <code>devices.ini</code> and takes effect at the next game launch.                                                       |
| Swap players  | Trade between two device seats. Both seats are moved together, and both settings travel with them. Example: P1 swaps with P2. Now, P1 gets P2 seat settings. P2 then gets P1 seat settings. |
| Forget device | Remove a device from the registry. Its seat is cleared. On the next boot the loader treats it as new. Use this when you have replaced hardware and the old identity is no longer valid.     |

## What Carries with a Seat

When ULDP assigns or reassigns a seat, the per seat settings like dead zone, response curve and rumble level all travel with the *seat*, not with the device. This is important to remember. The player's tuning stays on their player number. A device that now *claims* Player 2's seat *inherits* Player 2's tuning. This was specifically done to make a controller swap feel smooth, and is the behavior that makes a hot swap feel seamless. Think of it this way: the spare picks up from where the original left off, with the same settings the player already had for that seat, only now, using a different controller.

## Which Boards Support Live Seat Changes

On most boards, a seat swap takes effect at the next game launch. The panel says "Applied takes effect next launch" in those cases. On boards that communicate with the loader through the ULTRA\_IO\_SHM shared channel primarily, such as NAMCO ES3 (Pokken Tournament and Mario Kart Arcade GP DX) a seat swap can in principle reach the running game once live re-enumeration is implemented. On boards where the game hooks its own input channel or where the module spawns the game directly, the shared channel is not in use and the game will not see the change until it is restarted. The log and the panel both say which case you are in.

## ULCM: Ultra Loader Connected Mesh

ULCM stands for *Ultra Loader Connected Mesh*. It is a local network discovery and cabinet linking system. It lets you name your cabinet, find other Ultra Loaders instances on the same network, and start a linked session for games that support it.

Before anything else, this is what you need to understand about ULCM and netplay in this loader: We are not writing netplay for most games in this library. For the games that can link, the game already knows how. Our job is to give it a network it recognizes and a cabinet number, and then stay out of the way. For now, this is the best option. Future iteration of the loader may change this, enabling newer or better features.

Thus for games that were never built with a cabinet link mode, there is no linked play. A game whose network imports are zero cannot be synchronized across two machines in any meaningful way at this time. ULCM is honest about this for Ultra Loader. It tells you which games on your current session can link, and which ones cannot. That being said, a demonstration of ULCM requires two physical PCs on the same switch or network. There is no single machine variant of a two cabinet link session, unless you run dual systems in the same case, connected by an ethernet cable, and push video towards two separate screens.

## Your Cabinet Identity

When you open ULCM, you will see two fields at the top: a name and a cabinet ID. The name is yours to set. This will be what other players on the network will see when they discover you. The cabinet ID is derived from your machine's hardware the same way ULDP derives device identities: port independent, fully recomputable, and never tied to a VID/PID or a changing address. You can name your cabinet anything you want. The ID however, is not editable. It is what the mesh uses to distinguish you from every other cabinet on the network.

## Discovering Other Cabinets

The "On this network" section lists every other Ultra Loader instance visible on your local network. Each row shows the cabinet's name, what game it's running, and the latency to that cabinet in milliseconds. If the other cabinet is running the same game as you, its row shows an "Invite" button.

If the cabinet is running a different game, that row is dimmed and then labeled "no match". The idea is that you would not need to link to a cabinet playing a different title besides simply needing to know it's there. That being said, I did make an exception for this for certain types of games, which do now sit in the majority. However, this functionality can be explored for future improvement and enhancement in light of community request and justification.

Now, when another cabinet sends you an invite, a notification appears in the ULCM panel with the cabinet name, the game, and you will see two buttons: "Accept" and "Decline". Accepting of course starts the link session. Declining declines the link. If you accept, the game then handles everything from that point, such as cabinet numbering, synchronization, and the linked session itself.

## The Three Tiers of Netplay Support

As stated above, not every game can link, and the ones that can do so in very different ways. ULCM divides the library into three tiers, and the behavior you see in the panel reflects which tier your current game belongs to.

### Tier A: The game ships its own cabinet link protocol.

This is the best case, and it is the reason ULCM was made. These games were built for multi-cabinet arcade play. They discover peers, negotiate a session, synchronize state, and handle disconnection entirely on their own. ULCM's job is simply to give the game a network it recognizes and assign it a cabinet number. The game does mostly everything else. An example of a few games that do this is shown below:

|                                        |                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Wacky Races<br>(Taito Type X2)         | Two to four linked cabinets. The game uses DirectPlay8 Peer over TCP/IP, assigns its own addresses from a built in table, and has a NETWORK ID setting in its own test menu. This is the first target. This requires two physical PCs on one switch and also administrator elevation (the game adds its own IP address to your adapter via <i>AddIPAddress</i> , which requires admin rights). |
| Mario Kart Arcade GP DX<br>(NAMCO ES3) | Two to four linked cabinets. The game broadcasts on UDP to find its neighbors, assigns cabinet numbers PCB ID 1 through 4, and works on the 192.168.112.x subnet.                                                                                                                                                                                                                              |

The games in Tier A are LAN scoped by design. Wacky Races self-assigns addresses from a fixed private table; Mario Kart broadcasts to 255.255.255.255. Two cabinets in different cities would each assign themselves the same private address. Making Tier A work across the internet would require an L2 overlay, putting remote cabinets into the same broadcast domain. That is not in the current scope at this time.

### Tier B: Emulated Cores where the host owns the frame.

These are the retro cores running inside ultra module host.exe. The host owns the clock and steps each core frame by frame. Several cores have link hardware compiled in and switched off by a single setting:

|                                                         |                                                                          |
|---------------------------------------------------------|--------------------------------------------------------------------------|
| Supermodel (Le Mans 24, Daytona 2, Scud Race, ~22 more) | The Sega net board is compiled and ticking inside the core and enabled.  |
| Flycast (Initial D 1/2/3, Club Kart, F355, ~14 more)    | Naomi network code is already compiled in.                               |
| FBNeo (CPS 2, Neo Geo, CPS 3)                           | FBNeo also boasts rollback capability for smoother sync and performance. |

Tier B linked play is real and deliverable per core but users can expect issues at this first iteration. As the service become mature and we work out all the kinks, this should improve.

### Tier C: Native x86 games with no link mode.

This is most of the modern arcade library. Silent Hill The Arcade, Gundam Spirits of Zeon, Haunted Museum, Chaos Breaker, Ghost Squad Evolution, After Burner Climax, and the majority of the catalog. These games have no cabinet link protocol. Some import no network libraries at all. The loader calls into one of these games and blocks for the life of the session, since there is no frame by frame hook, no step function, and no state the loader can serialize and synchronize across machines.

For Tier C, the only networked feature that is honest and deliverable then, is asynchronous: this comes in the form of scores, ghost data, and leaderboards. The probe mechanism to read game state out of a running process exists for some boards, and the information each game exposes differs per title. This is the "no match" you see in the ULCM panel when a cabinet is running a Tier C game

while you are on a different title, or when the game you both have running has no link mode. The loader will tell you plainly which games can link, which ones can share scores, and which ones offer nothing network related at all. Any tool that claims live cabinet linking for a Tier C game is wrong...at least, for now.

## **Session Flow**

Once two compatible cabinets have been discovered and one has accepted the other's invite, ULCM hands off to the game. From that point the game's own link protocol runs the session. ULCM shows the session status connected cabinets, latency, cabinet IDs but does not manage the gameplay itself. If the game has a test menu option for network mode or cabinet number, that is where you configure those things. ULCM has already done its part by the time the game launches.

## **Scores and Ghosts for Non-Linkable Games**

For Tier C games where live linking is not possible, ULCM's contribution is asynchronous. Where the game state probe has been built for a board, the loader can read whether a game is in a race or a round, capture a score at the right moment, and send it to the mesh. Other cabinets running the same game see it. This is not synchronized play, as the two machines are running independent sessions, but it gives a shared context to games that would otherwise have no community dimension beyond sitting next to each other.

The probe is currently per game work. One game's address profile applied to a different executable will crash it. ULCM's score sharing for any given title is available only after that title's probe has been written and tested. This will be shipped in an updated firmware module for that game.

## **What ULCM Does Not Do**

Remote play over the internet for Tier A games the games, nor will it self-assign private addresses, and no transport decision made inside ULCM changes that. An L2 overlay that places remote machines into the same broadcast domain would be required, and it is not in this scope.

Live synchronized play for Tier C games also cannot be done, which is not a matter of scheduling, but a matter of the games not having the correct and needed architecture. A lockstep barrier can be inserted at the frame boundary but it would produce two cabinets stepping in perfect time while showing different worlds. There is no checksum, no resync, no state transfer. That is worse than no feature.

Video streaming for remote second player input is also not possible at this time, and this would require a full video capture and streaming pipeline on top of the loader. It is not scheduled and is not implied by anything ULCM ships.

## **In Summary**

ULDP and ULCM are the two systems in Ultra Loader that deal with the hardware around you and the community around the loader. ULDP makes your controllers reliable, persistent, and honest about which device is which, across USB ports, across reboots, and across hot swap events. ULCM makes it possible to find other Ultra Loaders instances in your network, allowing you to link cabinets for the games that were built for it, and share scores for the ones that were not. Both systems are opt in. Both systems tell you exactly what they can and cannot do. Neither one promises a feature it does not deliver, and the documentation for each one says so explicitly in the places where something is still being built. That is the standard this project holds itself to.

## **The “Purge Module Cache” button**

This is the final button in the “Enable Advanced Features” area. This button allows you to clear the local cache that Ultra Loader uses to store per-game firmware module data for the games you play. When you launch a game, that game has a corresponding custom-built firmware container, and the loader extracts the necessary data from that container and stores it in a cache folder on your computer. It does this so the next launch is faster, (even if by a few hundred milliseconds) since the files are already sitting there ready to go. That cache folder is what this button clears.

To put it another way: the firmware container is the authoritative copy of everything the game needs from the loader. Think of it like a sealed package. The cache is just where the loader unpacks the contents of that now opened package so it can use what’s inside. The sealed package itself never changes. Therefore, the unpacked copy in the cache is disposable by design.

Nothing else is affected by this. Your settings are completely untouched. Your saves, snapshots, art packs, and everything else you have configured stay exactly where they are. The only thing that gets removed is that cached staging data, which the loader can always recreate on its own. So if you have a game that is working right now, purging the cache will not break it. The next time you launch that game, the loader reads from the firmware container sitting on your computer, and copies the module all over again, the same way it did the first time. You end up right back where you started, except with a clean cache.

However, over time, the cache can grow, especially if you download newer firmware containers for games that have been improved by Ultra Loader and the community, and they have been updated several times. Old extracted versions of modules do not automatically clean themselves up, so they sit there taking up space even though they are no longer used. A purge sweeps all of that out at once.

Another common reason to use this button is troubleshooting. If a game is acting strangely after a container update, or something seems off and you aren’t sure why, purging the cache forces the loader to start fresh with a clean extraction of the firmware module content on the next game launch. It is one of the first things worth trying before digging deeper into a problem, because it costs nothing and only takes a second. You might also reach for it any time you are doing housekeeping and want to reclaim disk space, (although negligible) or simply as a routine reset after installing a batch of new containers. There is no downside, really.

However, the reason this button lives in Advanced Settings rather than somewhere more visible is simply that it involves deleting files, and that kind of button deserves to be somewhere users are paying more attention. But to be clear: it is always safe to press it, regardless. There is no scenario where purging the cache causes data loss or breaks a working game. If anything, it gives the loader a clean slate to work from.

Remember: No settings or saves or snapshots or any other thing will be deleted. Only old firmware module data for the games you’ve played. If you have a working game, there’s no harm, as the system will copy over that working module to your cache. So it is always safe to use this button if you need to. I put it here in advanced settings since it is a “delete or purge” type of button, so I don’t want users using it carelessly. However, as stated above, the button will always be safe to use.