Remapping controls per emulator

Although you can configure the 'generic' controls within Batocera just fine, what if you wanted to change the controller configuration for just a specific emulator? Like say you wanted to move the Z trigger on the N64 core from L2 over to R2? This is the article for that.

Things can get confusing really quickly when talking about controls in this abstract fashion, so here we will establish specific terminology that will be used throughout this article to refer to specific aspects:

- **Controller** This is the physical, bonafide controller in your hands. This can essentially be anything, a PS4 controller, an Xbox 360 controller, a NES gamepad hooked up to GPIO pins, your keyboard, your mouse, your Wii Zapper, etc. As long as it's been configured in EmulationStation's “Configure a Controller” menu, it's a controller.
- **Retropad** This is the simulated virtual controller that RetroArch uses to interface with its cores. You can read more about that and its philosophy here (keep in mind that Batocera is the one doing the auto-configuring in this case, not RetroArch), but all you need to know for this article is that it assumes the physical layout of a PS3 controller with SNES face buttons. Not all emulators will use this virtual controller as is, but Batocera will attempt to keep its button IDs as similar to it as possible. Exceptions will be noted.
- **System's controls** These are the signals sent to the emulated system at the end of the pipeline. For instance, this would be the PlayStation's **Cross**, **Square**, **Circle** and **Triangle** inputs.
- **Controls menu** This is the menu accessed from going into **Quick Menu** ([HOTKEY] + ) → **Controls**. This handles how the emulated system (core) sees your controls. For example, if you wanted to change the N64's Z trigger from L2 on your controller to R2, this is where you would go to do that. Note that not all emulators, specifically non-libretro emulators, have this menu. Exceptions will be noted.
- **EmulationStation** Although not always the case, some controller settings are configured from within Batocera. These settings can typically accessed through the **GAME SETTINGS → PER SYSTEM ADVANCED CONFIGURATION** → <system name> before launching the game. These may include things like what type of device is connected to what port, whether Joy-2-Dpad is activated for non-analog controllers, what preset keyboard configuration is used, etc. dependent on the emulator. Some of these settings may be ignored when manually making an override in the emulator itself, so keep this in mind. Settings like this will be referred to as “ES's setting”.

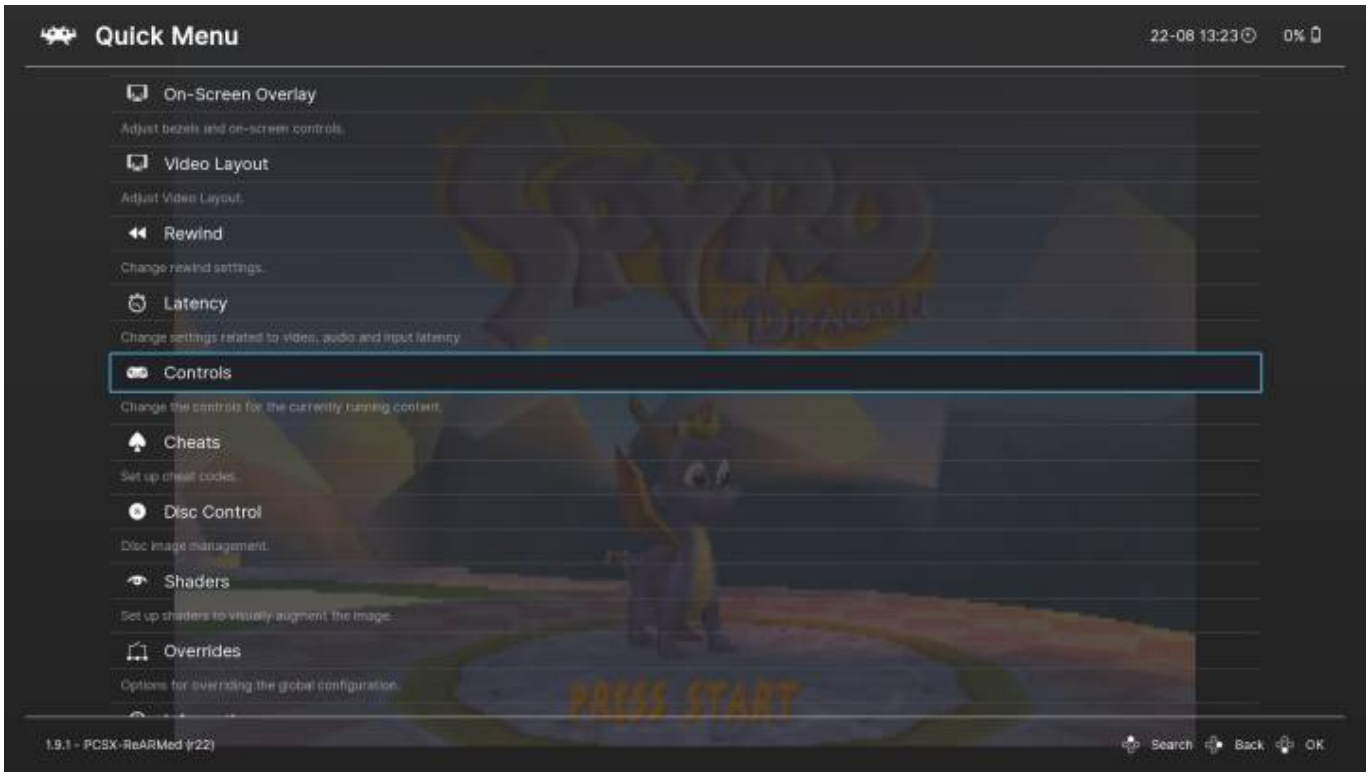
The similarly titled **Inputs** menu accessed by exiting out to the RetroArch main menu → **Settings** → **Inputs** is **not** the menu you should be going to in order to remap controls for a specific core. This is where Batocera translates your controller's raw inputs to RetroArch's Retropad, and is automatically generated based on what controller you're currently using. Again, **this menu is not accessed when only remapping per core controls**.

libretro cores

For the majority of its emulators, Batocera employs the use of RetroArch cores. These are indicated by the “libretro” prefix being in front of it. You can select your system's emulator by going into its games list from the system menu, pressing [SELECT] to go into that system's **VIEW OPTIONS** → **ADVANCED SYSTEM OPTIONS** → **<emulator>**.

The great thing about libretro cores is that they all use the same interface, and are (mostly) compatible with everything you can change there. One useful consistent menu option is the **Quick Menu** → ***Controls*** *menu*, which we will be using to remap our controls. Let's use the PlayStation libretro/PCSX-ReArmed emulator for example.

Let's say you wanted to swap the Circle and Cross buttons (very common for Japanese-to-Western game control schemes). First, open the **Quick Menu** with [HOTKEY] + and then go to the ***Controls*** *menu* item.



Then select your appropriate port (usually Port 1). You'll see the *Retropad*'s virtual buttons on the left and the emulated *system's controls* on the right.

You can go into any *Retropad* virtual button and assign it to a *system's control*.

And there you go, feel free to alter the controls how you like them!

However, you'll find that your remapping is lost upon exiting the emulator. To make these changes permanent, you must “Save Core Remap File” from the ***Controls*** *menu*.

If you receive an error message saying something similar to “cannot save remap file”, check for these folders:

- `/userdata/system/configs/retroarch/inputs/`
- `/userdata/system/.config/retroarch/config/remaps/`

If they don't exist, create them and try again.

You can also save just the game or directory's remap. The directory's remap will apply to all games in the same named folder.

In case you'd ever like to go back to the default settings, you can delete the remap file from this menu as well (you'll have to exit and re-enter the menu to see this).

You may have noticed the “Analog to Digital Type” setting. This setting is handled in *ES* as the Joystick-to-Dpad advanced system setting (or similarly named), however if you have a saved remap file then the setting in your remap file will take priority over ES's setting.

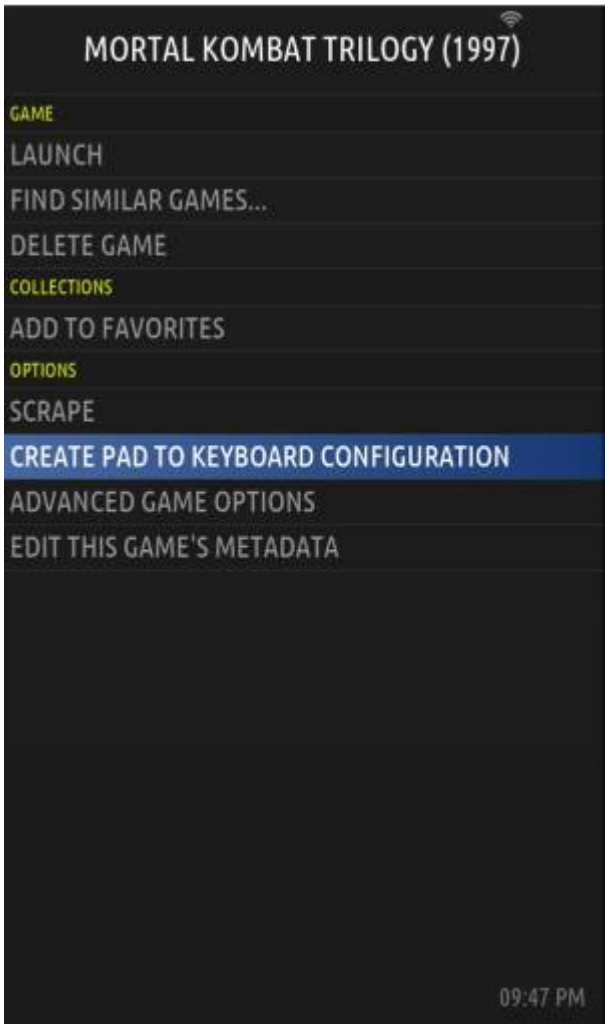
Remap files are saved to `/userdata/system/.config/retroarch/remaps/(core name)/(core/folder/game name).rmp` (you will need to enabled “Show hidden folders/files” in your file manager to view this directory).

Information on particular cases and exceptions for particular RetroArch cores need to be added here.

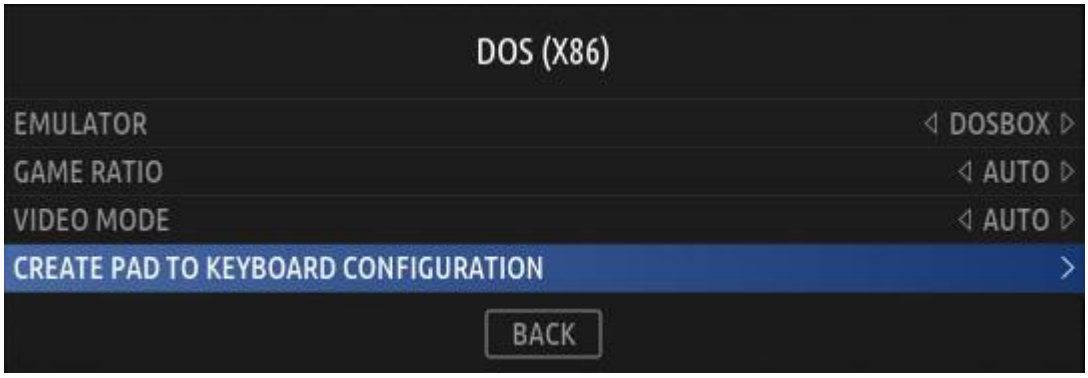
DosBox standalone

A part of the configuration of DosBox you can configure pad2key to assign your *controller* to keys on the virtual keyboard. This is especially useful for console-to-PC ports, but may not be as usable for games primarily focused on keyboard control.

This can be achieved by creating a pad to keyboard configuration for the game in *EmulationStation*. While hovering over your intended game to configure, press and hold to access this menu (press in Batocera v30 and earlier).



If you'd like to configure a pad2key configuration for every DOS game, you can do so from the **ADVANCED SYSTEM OPTIONS**. Press [SELECT] while in the DOS gamelist.



You can create this file manually by placing an appropriate `pad2.key` in the `(game).pc` directory of the game. No need to change any configuration setting, this file simply being present will activate pad2key. If your game is in a zip file in the `dos` directory instead, place the `pad2.key` file in the `dos` directory alongside it and rename it to `gametitle.zip.key`, where gametitle is the name of the zip file.

This is specifically for the standalone DosBox emulators. This will have no effect on the libretro cores like libretro/DosBox Pure unless you also set `dos.controller1_dosbox_pure` to 3 for “Custom keyboard bindings (best for Batocera Pad2Key)”.

Here is an example pad2key mapping file for Mortal Kombat Trilogy:

pad2.key

```
{
    "actions_player1": [
        {
            "trigger": "up",
            "type": "key",
            "target": "KEY_UP"
        },
        {
            "trigger": "down",
            "type": "key",
            "target": "KEY_DOWN"
        },
        {
            "trigger": "left",
            "type": "key",
            "target": "KEY_LEFT"
        },
        {
            "trigger": "right",
            "type": "key",
            "target": "KEY_RIGHT"
        },
        {
            "trigger": "start",
            "type": "key",
            "target": "KEY_ENTER"
        },
        {
            "trigger": "select",
            "type": "key",
            "target": "KEY_KP7"
        },
        {
            "trigger": "a",
            "type": "key",
            "target": "KEY_KP2"
        },
        {
            "trigger": "b",
```

```
            "type": "key",
            "target": "KEY_KP1"
        },
        {
            "trigger": "x",
            "type": "key",
            "target": "KEY_KP5"
        },
        {
            "trigger": "y",
            "type": "key",
            "target": "KEY_KP4"
        },
        {
            "trigger": "pageup",
            "type": "key",
            "target": "KEY_KP6"
        },
        {
            "trigger": "pagedown",
            "type": "key",
            "target": "KEY_KP3"
        },
        {
            "trigger": [
                "hotkey",
                "start"
            ],
            "type": "key",
            "target": "KEY_ESC"
        },
        {
            "trigger": [
                "hotkey",
                "x"
            ],
            "type": "key",
            "target": "KEY_F4"
        },
        {
            "trigger": [
                "hotkey",
                "y"
            ],
            "type": "key",
            "target": "KEY_F5"
        }
    ]
}
```

The “trigger” is your *Retropad* controller; you shouldn't need to change this (“pageup” and

“pagedown” refer to L1 and R1 respectively). The “target” is the virtual keyboard's key; feel free to change/swap these around.

DosBox Pure automatically detects the game running and loads its own preset configuration. The libretro/DosBox-Pure core can use the standard libretro core remapping process (recommended to use per-game remap file). If a game would rather benefit from complete control over the keyboard, press Scroll Lock to enable Game Focus mode (removes keyboard hotkeys).

DosBox can also emulate having a joystick plugged into the virtual serial port by enabling the `joysticktype` line in `dosbox.cfg`, however due to lack of consistency between joystick manufacturers not all games implement all aspects of these joysticks completely/correctly. It's recommended to use pad2key for most games (only a very few games took advantage of analog controls).

Mupen64Plus

Unlike with the libretro core, remaps of the *system's controls* for the standalone Mupen64Plus can be configured by editing `userdata/system/configs/mupen64/input.xml`. Back up this file before making edits to it. When you open it, you'll see the bindings like so:

input.xml

```
<inputList>
<input name="AnalogDeadzone" value="0,0" />
<input name="AnalogPeak"     value="32768,32768" />
<input name="l3"             value="Mempak switch" />
<input name="r3"             value="Rumblepak switch" />
<input name="a"                 value="C Button R" />
<input name="b"                 value="A Button" />
<input name="x"                 value="C Button U" />
<input name="y"                 value="B Button" />
<input name="start"             value="Start" />
<input name="select"            value="" />
<input name="pageup"            value="L Trig" />
<input name="pagedown"          value="R Trig" />
<input name="l2"                value="Z Trig" />
<input name="r2"                value="" />
<input name="up"                value="DPad U" />
<input name="down"              value="DPad D" />
<input name="right"             value="DPad R" />
<input name="left"              value="DPad L" />
<input name="joystick1up"    value="Y Axis" />
<input name="joystick1down"  value="Y Axis" />
<input name="joystick1left"  value="X Axis" />
<input name="joystick1right" value="X Axis" />
```

```
<input name="joystick2up"    value="C Button U" />
<input name="joystick2down"  value="C Button D" />
<input name="joystick2left"  value="C Button L" />
<input name="joystick2right" value="C Button R" />
</inputList>
```

The input names on the left are the *Retropad* inputs (“pageup” and “pagedown” refer to L1 and R1 respectively). You don't typically need to edit these.

The values on the right are the *system's controls*. Feel free to switch these around with each other.

Mupen64 can only accept one name per value; if multiple are detected it will only use the first occurrence. For example, if you bind the *system control* Z value to both L2 and R2 *Retropad* names, only the L2 button will activate the *system control* Z button.

You can assign various Mupen64 commands here like `Mempack switch`!

Dolphin

Gotta confirm Wii settings.

Wii

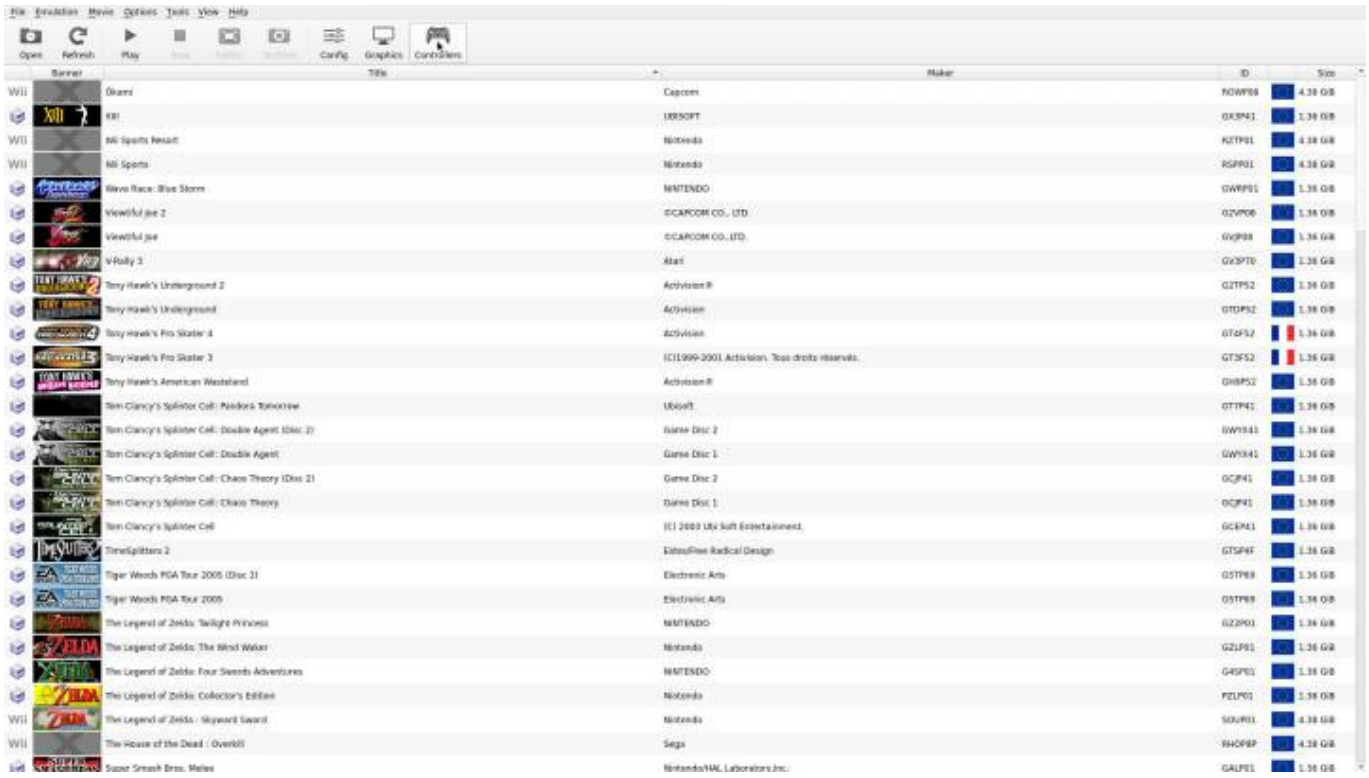
Firstly, if you'd like Batocera to handle emulated Wiimotes/Wii Classic Controller input, enable `wii.emulatedwiimotes` as described on this page. This works best for games that support control via the Wii Classic Controller or the Gamecube Controller. This can also be saved on a per-game basis by utilizing configuration files.

If you prefer to use Dolphin's interface for configuring controls instead of Batocera's configuration files, read on.

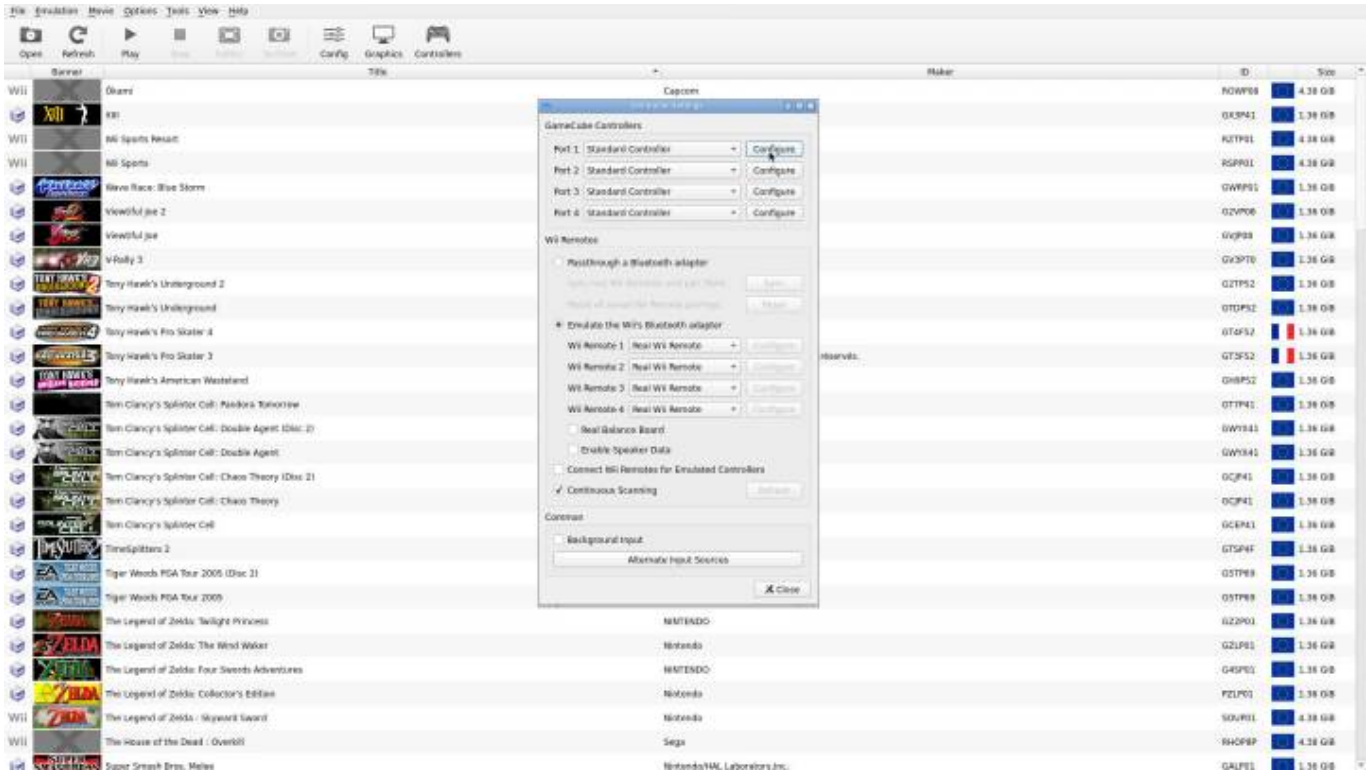
GameCube

You can configure the controls for this system within its own controller configuration utility (mouse

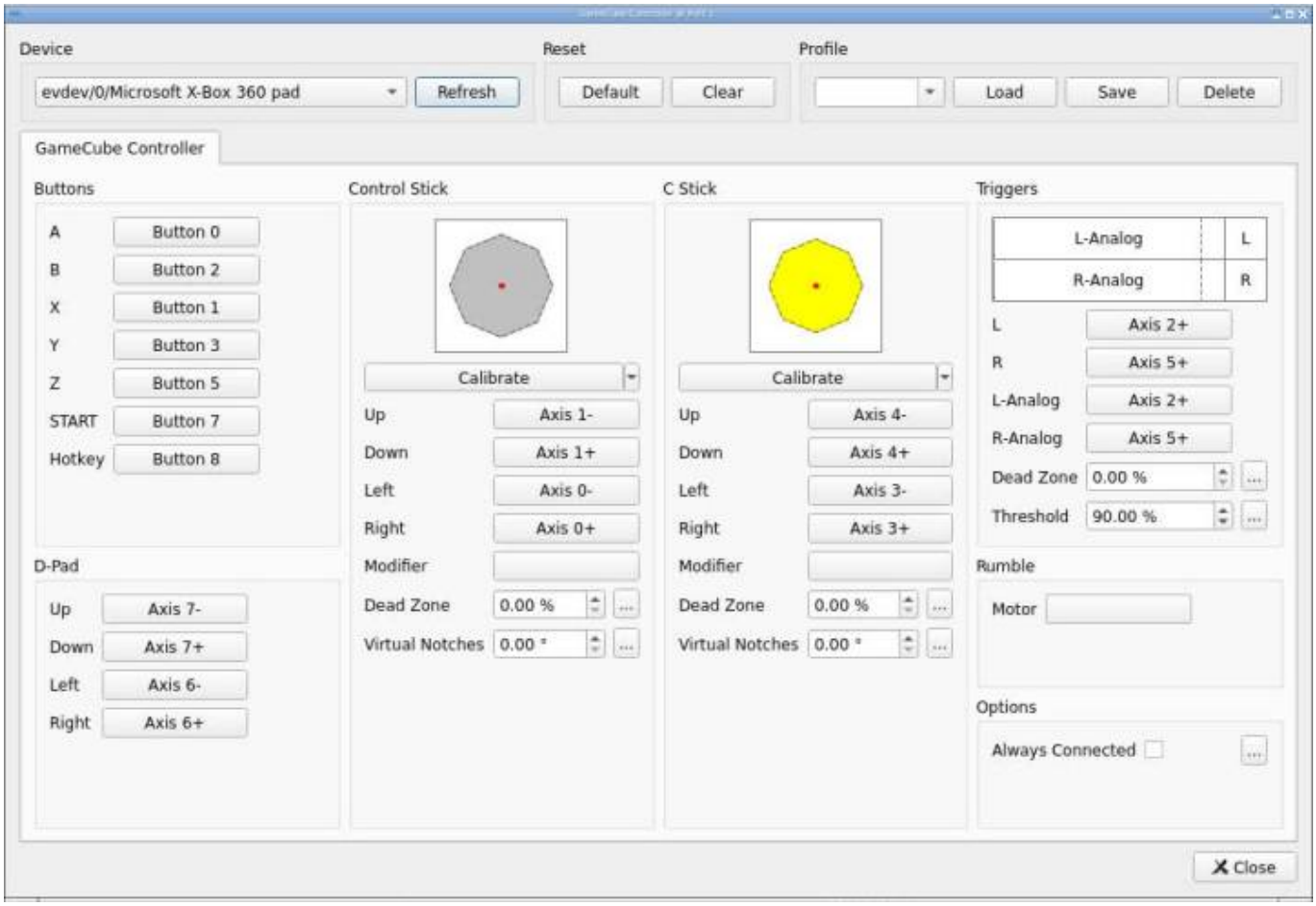
and keyboard required). On the system menu, open Files with [F1] on the keyboard and navigate to **Applications → dolphin-config**. This will open Dolphin's menu.



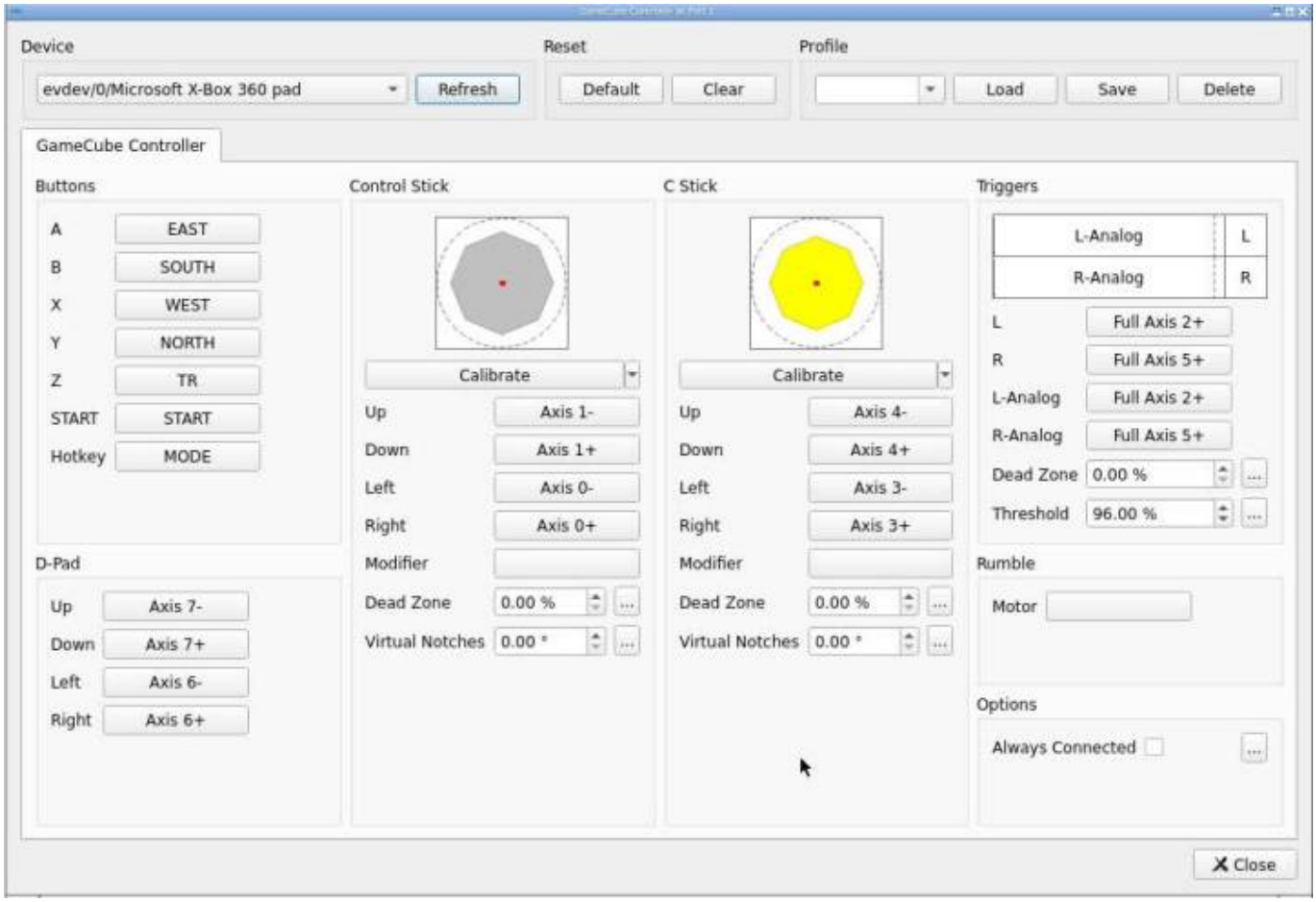
From here, navigate to **Controllers** in the toolbar (or press F1 on the keyboard).



Click on the **Configure** button for Port 1 of the GameCube Controllers. You'll notice that it has a bunch of “Button #” already assigned; this is the default configuration of Dolphin and is unrelated to your current controller.

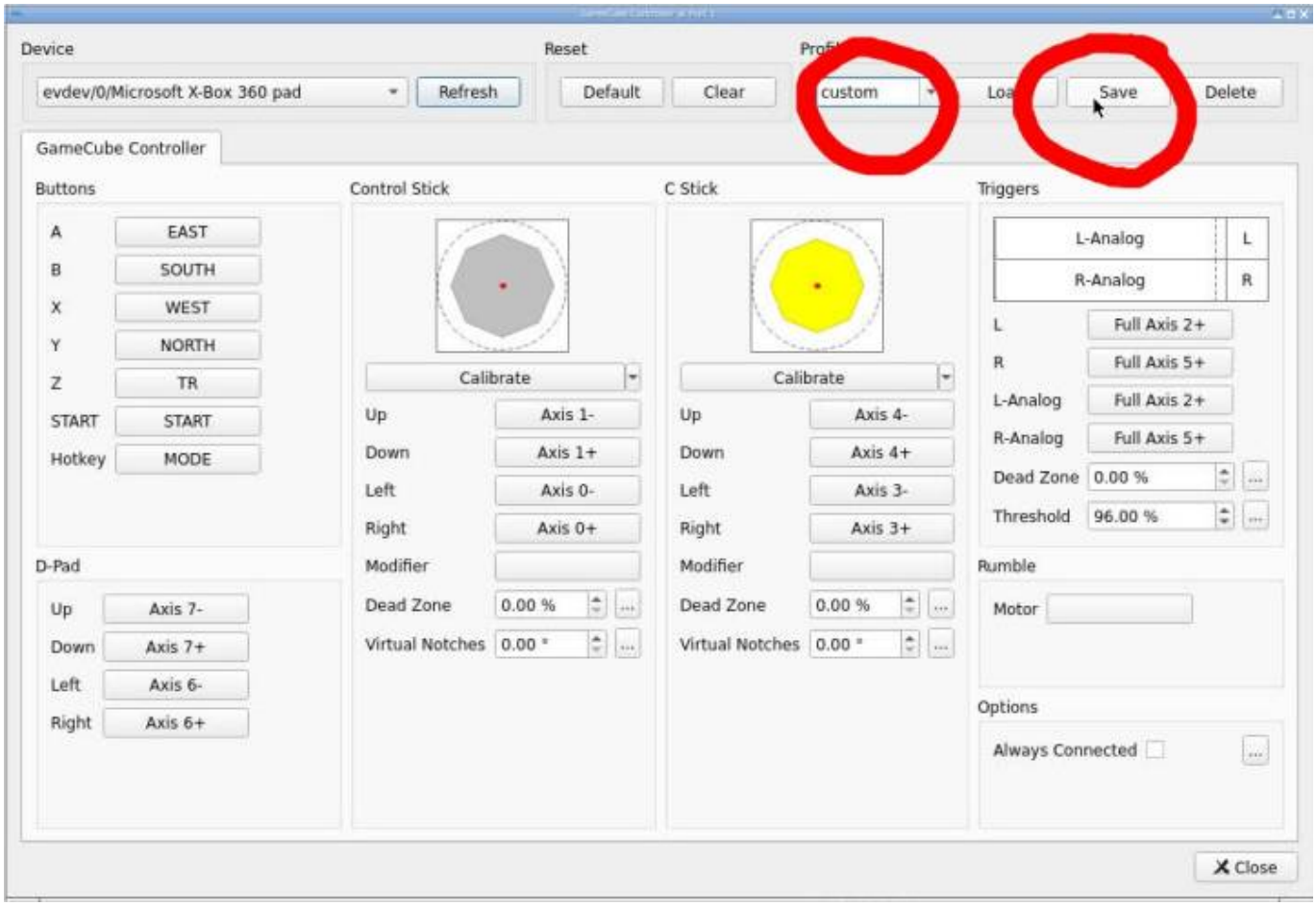


You can configure *system controls* by clicking on them and pressing a button on your *controller*. Let's say I had a SNES-style pad and wanted the labels to match the game's inputs, I would reassign it like so:



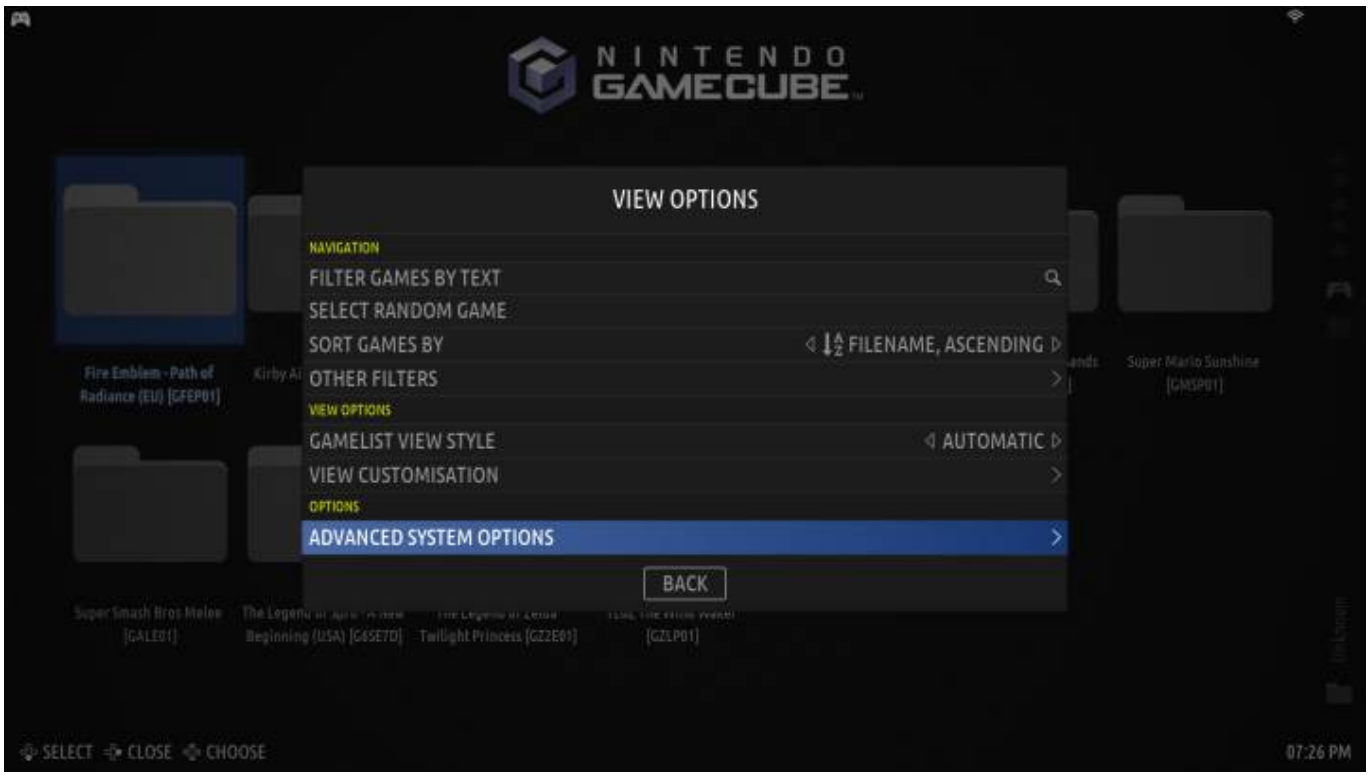
You'll notice that the buttons on your *controller* will turn into cardinal directions. This is the lingo used by Batocera to unify all the various controllers you can connect to it. You might also want to calibrate your sticks, as I've done here, and maybe increase the trigger threshold (good controllers should be able to go all the way up to 99% with no trouble!). You should also probably reassign the trigger axises so the triggers use the full axis instead of only a partial amount of it.

Once you're done, type a name into the **Profile** field at the top right and click **Save**.

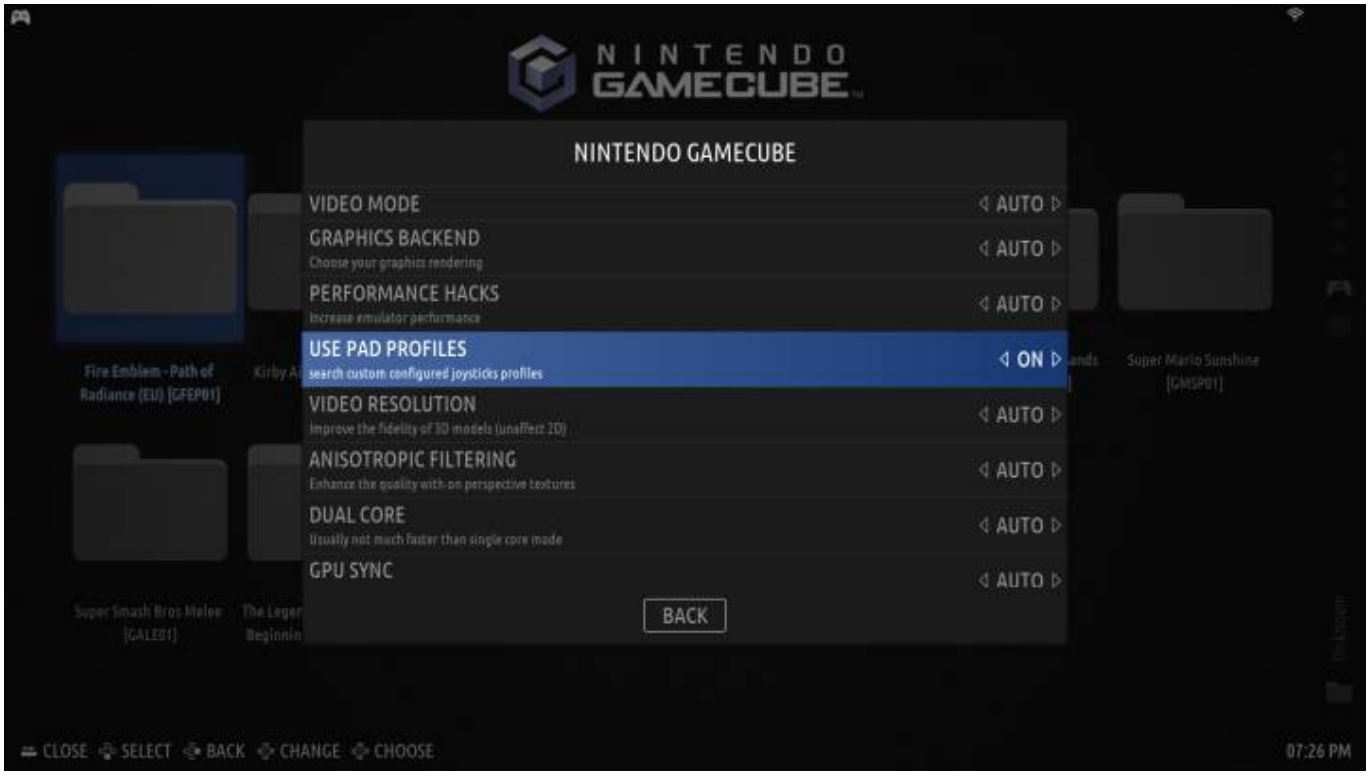


Click **Close** on all the windows and then **File** → **Quit** to leave Dolphin.

Now, before you launch any GameCube/Wii game, open the **ADVANCED SYSTEM OPTIONS** by pressing [SELECT] on the GameCube's game list screen.



And set **USE PAD PROFILES** to “ON”. If you don't do this, your profile will be ignored.



And now your controller is remapped in Dolphin.

This needs to be tested with other controllers.

The control panel itself is intuitive and easy to follow, but has extremely powerful scripting features. Further documentation available at https://dolphin-emu.org/docs/guides/configuring-controllers/#GameCube_Controller.

Controllers with only one analog stick

If your *controller* has only one analog stick (Odroid Go Advance, or if you have a Dreamcast or N64-style gamepad for example) and you want to use an emulator that by default uses the D-pad as the *system's controller* and the second analog as the *system's mouse*. You can edit the `/userdata/system/batocera.conf` and add the following lines in order to have the left analog as the *system's mouse*:

```
amiga1200.retroarch.input_player1_analog_dpad_mode=1
amiga1200.retroarch.input_player2_analog_dpad_mode=1
amiga500.retroarch.input_player1_analog_dpad_mode=1
amiga500.retroarch.input_player2_analog_dpad_mode=1
c64.retroarch.input_player1_analog_dpad_mode=1
```

```
c64.retroarch.input_player2_analog_dpad_mode=1
```

Cores like LR-PUAE and Vice (and some others like Flycast etc...) will then map the left analog on your *RetroPad* to the *system's mouse* (RetroArch only).

Reconfiguring the Keyboard for RetroArch

If you'd like to reconfigure the default keyboard bindings for all systems emulated by a libretro core (such as if you don't have a controller to use or you're in an all-in-one device like a netbook), refer to the keyboard controls section of the advanced RetroArch settings page instead.