# 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”.

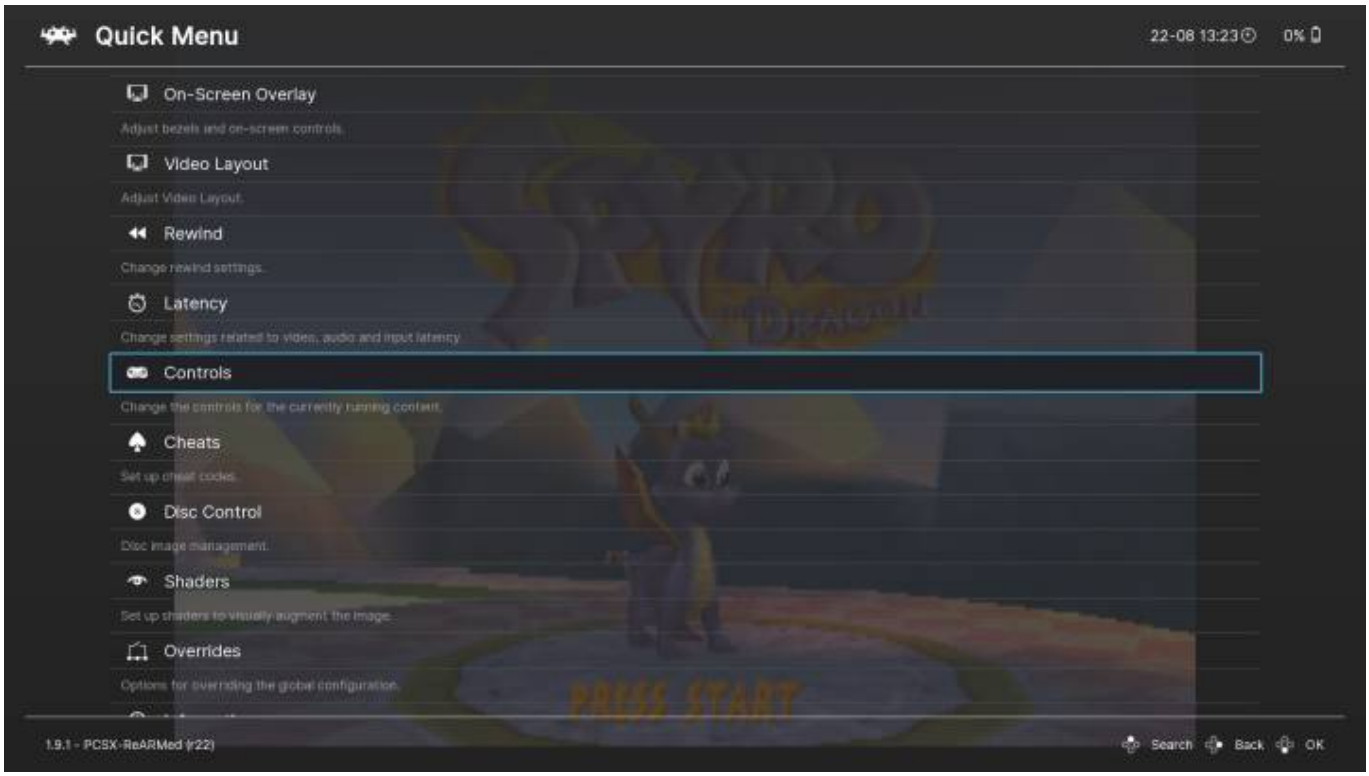
The similarly titled **Inputs** menu accessed by exiting out to the RetroArch's **Main Menu → Settings → Inputs** is **not** the menu you should be going to in order to remap controls for a specific core. This is where Batocera translates your controller's raw inputs to RetroArch's *Retropad*, and is 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/configs/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).

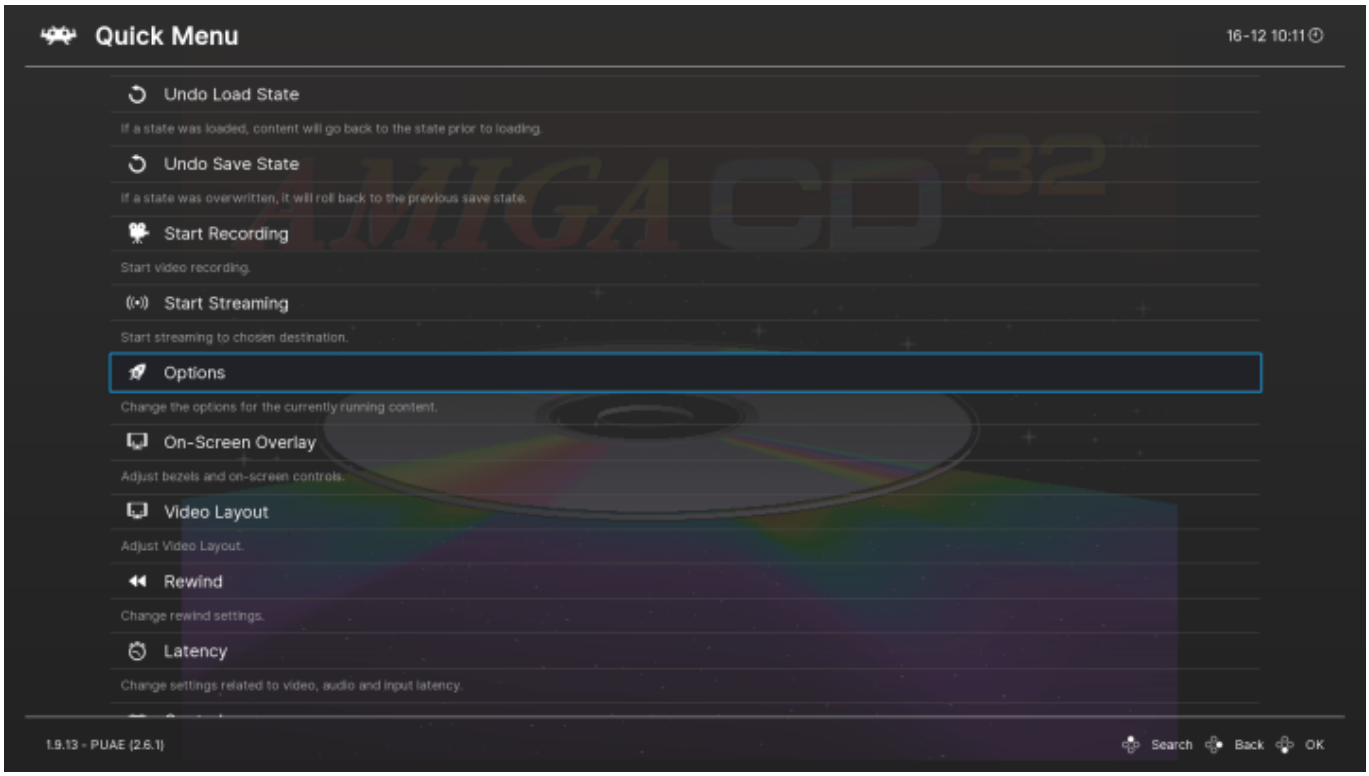

You may have noticed the “Analog to Digital Type” setting. This setting is handled in *ES* as the Joystick-to-Dpad advanced system setting (or similarly named), however if you have a saved remap file then the setting in your remap file will take priority over ES's setting.

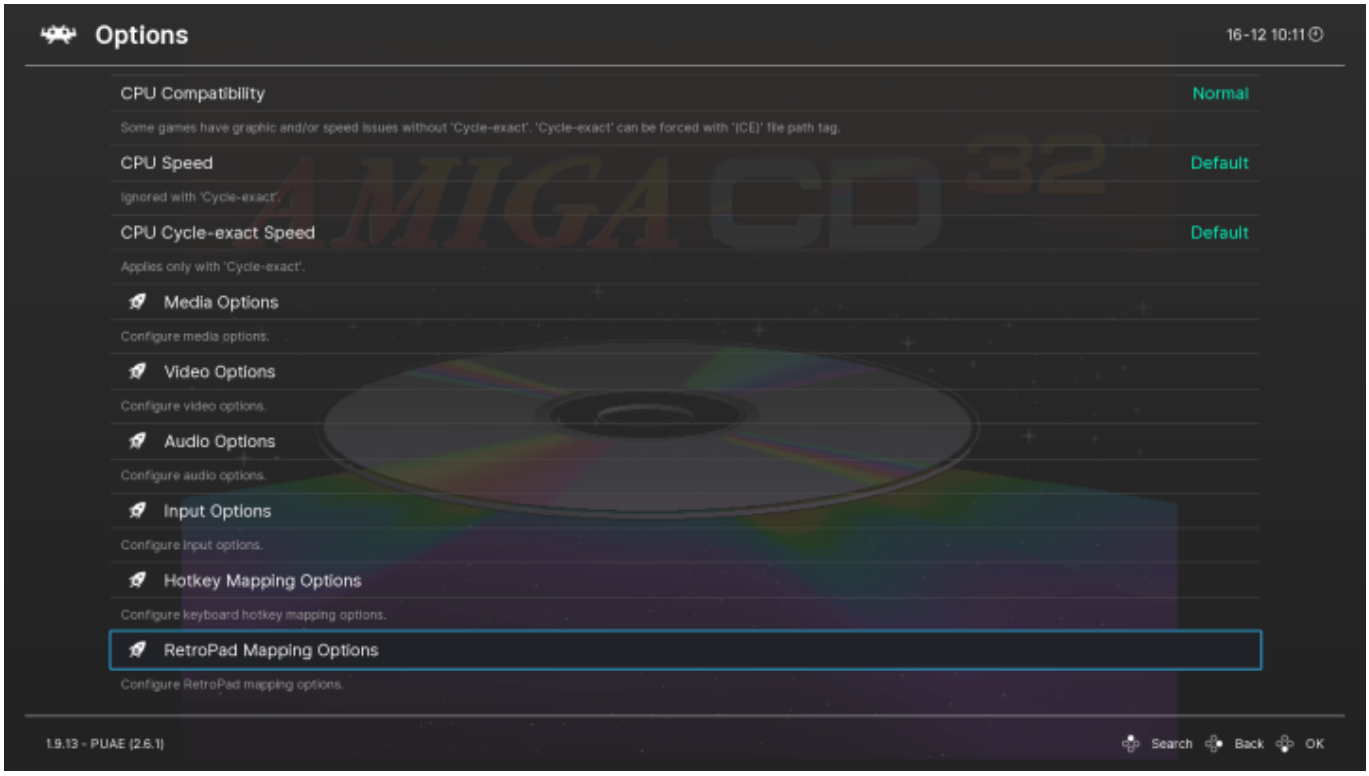
Remap files are saved to `/userdata/system/configs/retroarch/remaps/(core name)/(core/folder/game name).rmp` (you will need to enabled “Show hidden folders/files” in your file manager to view this directory).

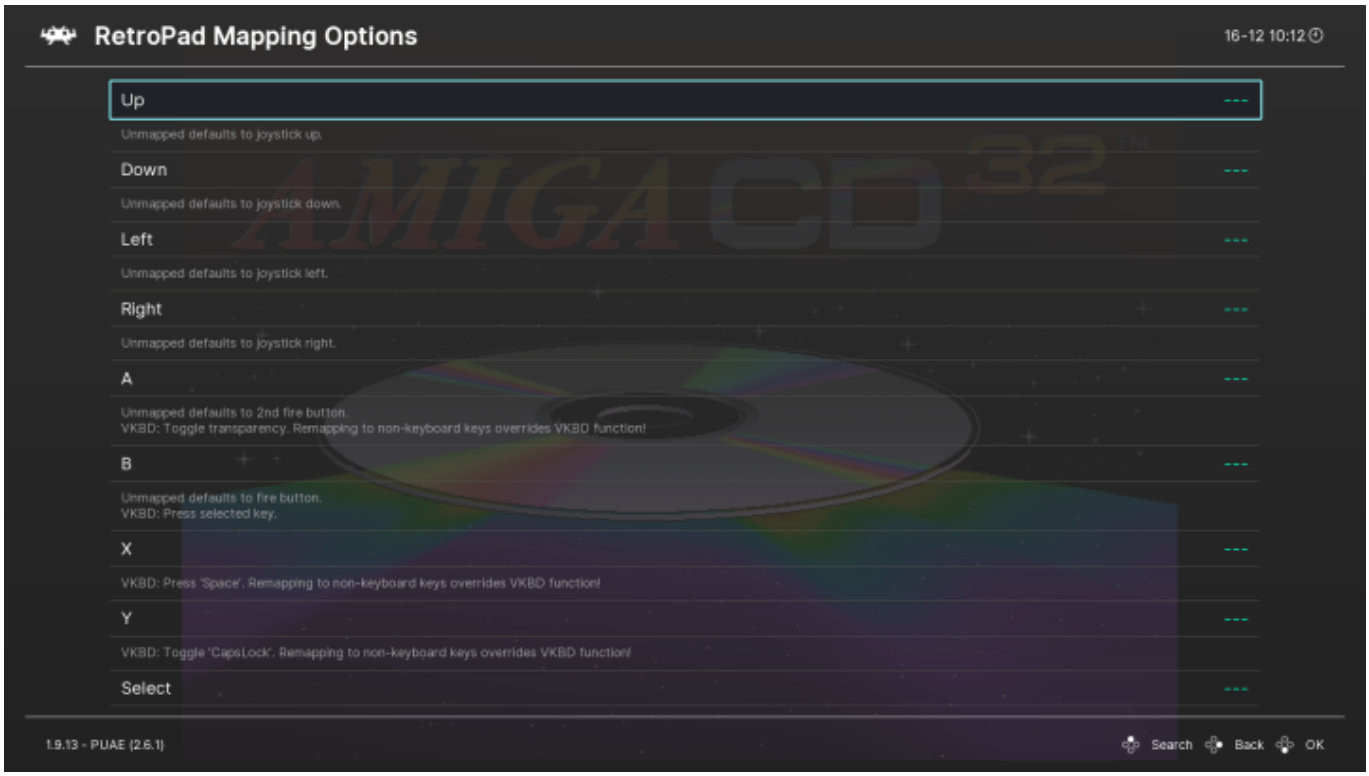
### libretro: PUAE/VICE

Controls for these computer emulators are stored in its core options, accessed via **Quick Menu → Options → RetroPad Mapping Options**. These take whatever you have assigned to the equivalent button in the **Controls** menu to another controller layer unique to this core.

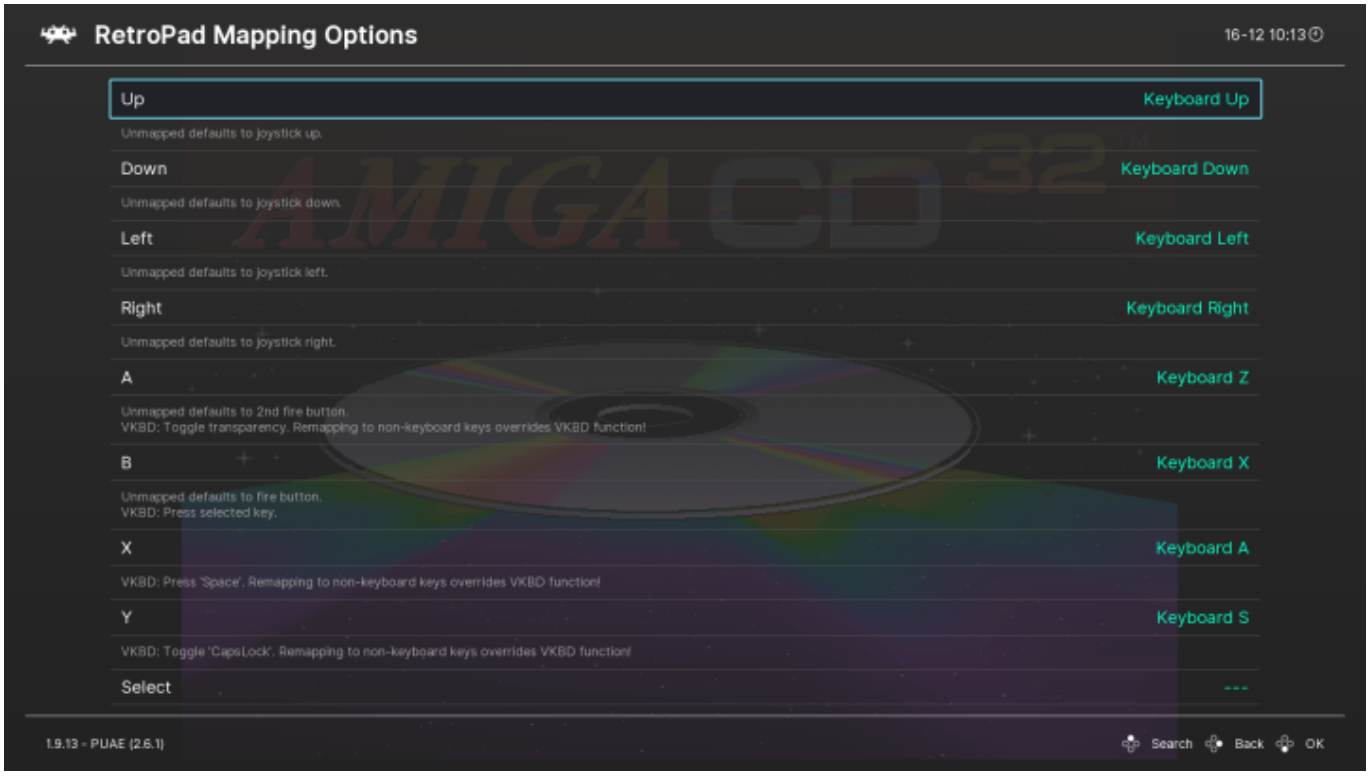

Quick Menu
16-12 10:11
Resume
Resume the currently running content and leave the Quick Menu.
Restart
Restart the content from the beginning.
Close Content
Close the current content. Any unsaved changes might be lost.
Take Screenshot
Capture an image of the screen.
State Slot
0
Change the currently selected state slot.
Save State
Save a state to the currently selected slot.
Load State
Load a saved state from the currently selected slot.
Undo Load State
If a state was loaded, content will go back to the state prior to loading.
Undo Save State
If a state was overwritten, it will roll back to the previous save state.
1.9.13 - PUAE (2.6.1)
Search
Back
OK


Quick Menu
16-12 10:11
Undo Load State
If a state was loaded, content will go back to the state prior to loading.
Undo Save State
If a state was overwritten, it will roll back to the previous save state.
Start Recording
Start video recording.
Start Streaming
Start streaming to chosen destination.
Options
Change the options for the currently running content.
On-Screen Overlay
Adjust bezels and on-screen controls.
Video Layout
Adjust Video Layout.
Rewind
Change rewind settings.
Latency
Change settings related to video, audio and input latency.
1.9.13 - PUAE (2.6.1)
Search
Back
OK

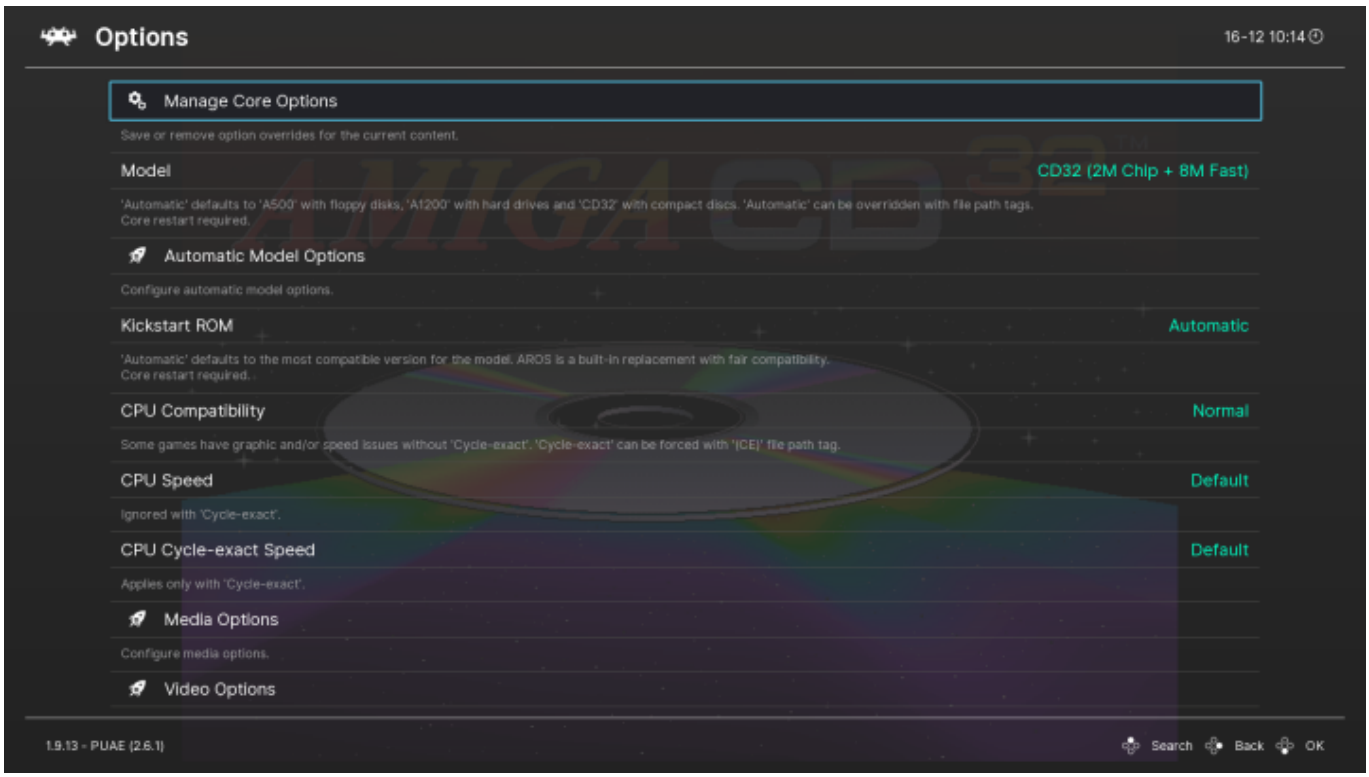


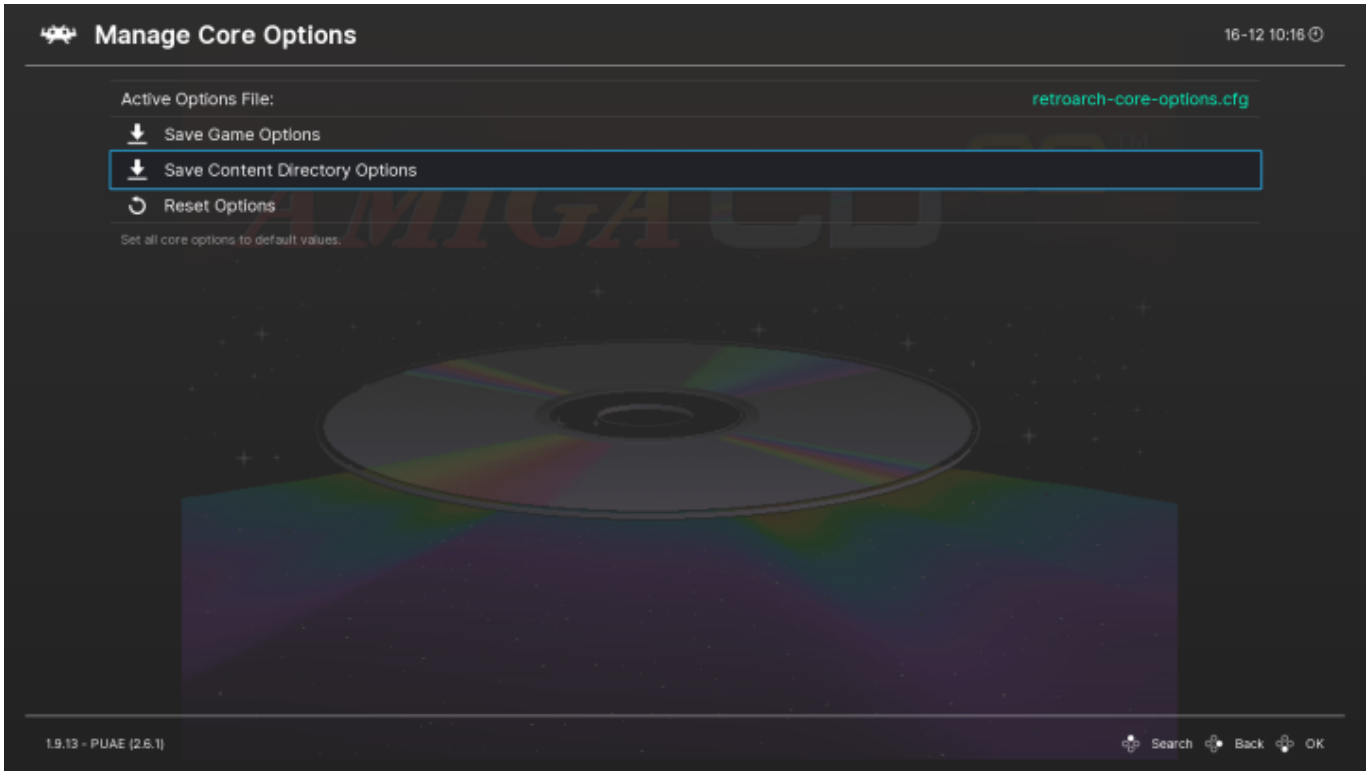




Then change your keymaps as appropriate. PUAE/VICE have decided to use the SNES layout for reference here, so A is  , B is  , X is  and Y is  .

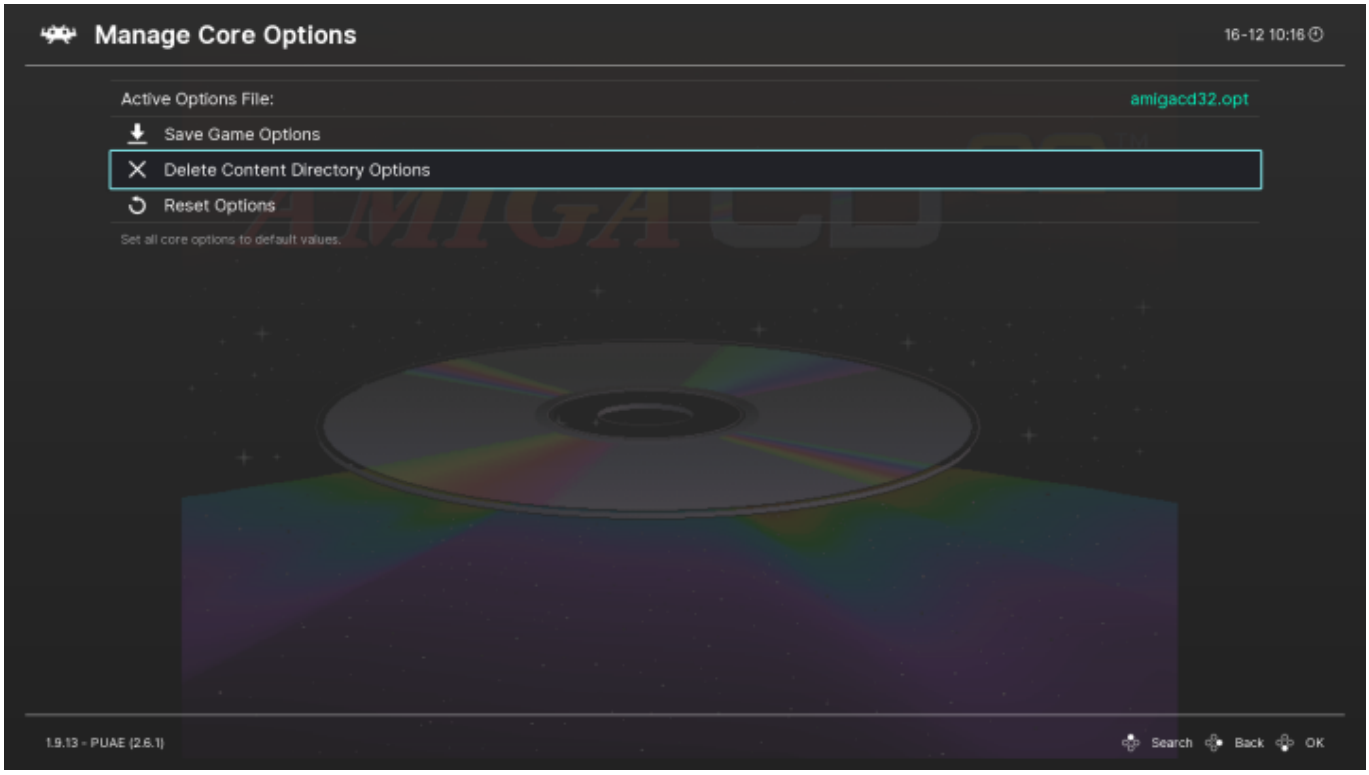



Once you're done, back out one menu and scroll up to **Manage Core Options**.






To save the remapping for just this game, **Save Game Options**. Otherwise, to apply it to all games in the same folder as this game, **Save Content Directory Options**.




# MAME

Massively improve this section.

MAME machines can be remapped in-game:

1. Press `[Enter]` or push in `[L3] + [R3]` to open up the MAME menu.
2. Go to **Input for this game** and press `[Enter]` or .
3. Select the desired input to remap and press `[Enter]`/, followed by the key/button desired to remap to.
    - Inputs can be cleared by double-tapping `[Enter]`/ instead.

Remap configuration files are stored at `\userdata\saves\mame\mame(version)\cfg\`

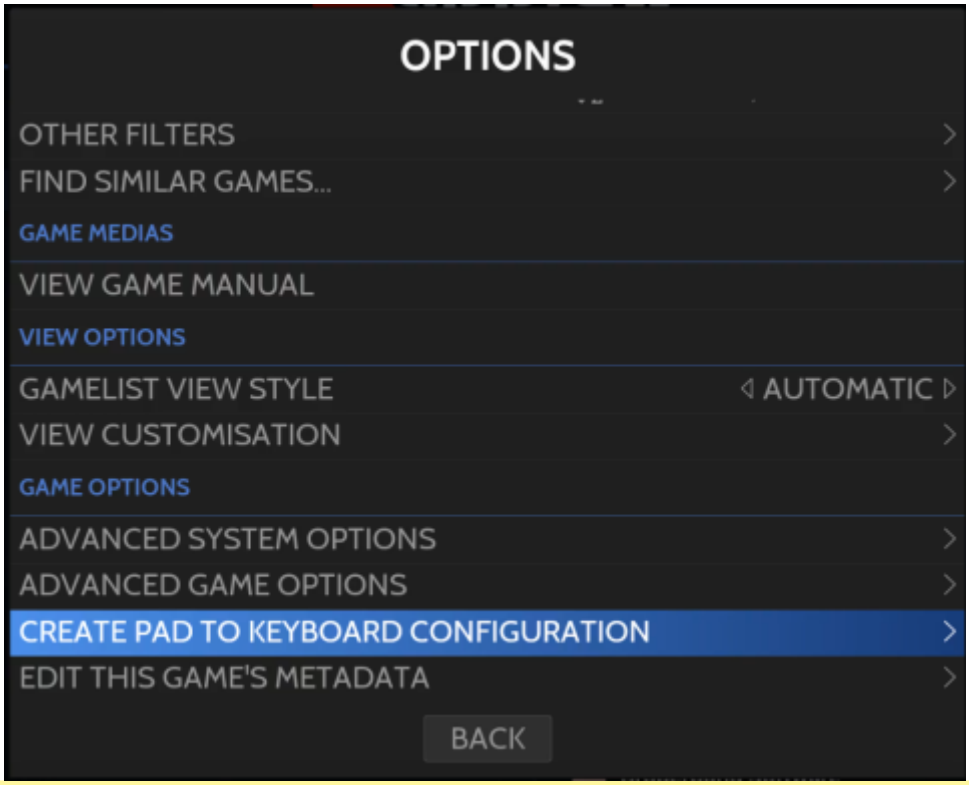
# Pad2key

This functionality can help you create a per-game mapping of your controller to keyboard keys, in order to play games from old computers like Apple II, Sinclair or DOS and Windows games that are finicky with their joypad support.

It can also, to some extent, be used to define additional emulator command shortcuts from the controller (such as force-closing the emulator with `[Alt] + [F4]`, not advisable if you like having non-corrupted save data).

## Define a system-wide pad2key mapping

The first thing to do is to select a game in the game list and press `[SELECT]`, go to **ADVANCED SYSTEM CONFIGURATION** and select **Create pad to keyboard configuration**.



If a game has a config setting for itself, it'll not use the system-wide config at all, so editing the system-wide config will have no effect on the game itself.

From there, select a key for each button on your Batocera RetroPad for Player 1 and/or Player 2.


To create a mapping to send a combination (“chord” or “combo”) instead of a single key, use to enter each of the initial keys in the combination, and use to enter the final key.

## Pad2key configuration file

Once your pad2key configuration is done for a specific game, it stays saved as a `padto.keys` or `game_name.keys` file next to your rom/in your Windows game directory folder. For a system-wide configuration, it'll be saved in `/userdata/system/configs/evmapy/system-name.keys`. It is a

plain JSON file with. Example:

```
  {
  "actions_player1": [
      {
          "trigger": [
              "hotkey",
              "start"
          ],
          "type": "key",
          "target": [
              "KEY_LEFTALT",
              "KEY_F4"
          ]
      },
      {
          "trigger": "up",
          "type": "key",
          "target": "KEY_UP"
      },
      {
          "trigger": "down",
          "type": "key",
          "target": "KEY_DOWN"
      },
      {
          "trigger": "a",
          "type": "key",
          "target": "KEY_RIGHTSHIFT"
      }
    ]
 }
```

### Action profiles

A singular keys file may contain the action profiles for multiple players. "actions_player1" will contain the actions for player 1, "actions_player2" for player 2 and so on. Each action profile contains its own unique set of triggers and targets. Example:

```
{
"actions_player1": [
    {
        "trigger": [
            "hotkey",
            "start"
        ],
        "type": "key",
        "target": [
            "KEY_LEFTALT",
```

```
            "KEY_F4"
        ]
    },
    {
        "trigger": "up",
        "type": "key",
        "target": "KEY_UP"
    }
  ],
"actions_player2": [
    {
        "trigger": [
            "hotkey",
            "start"
        ],
        "type": "key",
        "target": [
            "KEY_LEFTALT",
            "KEY_F4"
        ]
    },
    {
        "trigger": "up",
        "type": "key",
        "target": "KEY_UP"
    }
  ]
}
```

In Batocera **v37** and above, light guns can also contain their own actions. These are unique to the light guns, and thus use their own actions profile: `"actions_gun1"`. Example:

```
{
    "actions_gun1": [
        {
            "trigger": [
                "3",
                "2"
            ],
            "type": "key",
            "target": [
                "KEY_LEFTALT",
                "KEY_F4"
            ]
        }
    ]
}
```

## Triggers

#### Batocera Retropad names

The gamepad buttons are referred to specific values, they are as follows:

- D-Pad UP : `"trigger": "up"`
- D-Pad DOWN : `"trigger": "down"`
- D-Pad LEFT : `"trigger": "left"`
- D-Pad RIGHT : `"trigger": "right"`
- [START] : `"trigger": "start"`
- [SELECT] : `"trigger": "select"`
-  : `"trigger": "b"`
-  : `"trigger": "a"`
-  : `"trigger": "y"`
-  : `"trigger": "x"`
- Left stick UP : `"trigger": "joystick1up"`
- Left stick DOWN : `"trigger": "joystick1down"`
- Left stick LEFT : `"trigger": "joystick1left"`
- Left stick RIGHT : `"trigger": "joystick1right"`
- Right stick UP : `"trigger": "joystick2up"`
- Right stick DOWN : `"trigger": "joystick2down"`
- Right stick LEFT : `"trigger": "joystick2left"`
- Right stick RIGHT : `"trigger": "joystick2right"`
- [L1] : `"trigger": "pageup"`
- [R1] : `"trigger": "pagedown"`
- [L2] : `"trigger": "l2"`
- [R2] : `"trigger": "r2"`
- Push in [L3] : `"trigger": "l3"`
- Push in [R3] : `"trigger": "r3"`
- [HOTKEY] : `"trigger": "hotkey"`

#### Batocera light gun

Most Batocera light guns will automatically map themselves as a virtual keyboard and pointer device, associating their ID with a particular player controller. More info on each individual light gun mapping can be found on the light gun page.

- Primary fire : `"trigger": "left"`
- Secondary fire : `"trigger": "right"`
- [START] : `"trigger": "middle"`
- Aux button 1 : `"trigger": "1"`
- Aux button 2 : `"trigger": "2"`
- Aux button 3 : `"trigger": "3"`
- Aux button 4 : `"trigger": "4"`
- UP : `"trigger": "5"`
- DOWN : `"trigger": "6"`
- LEFT : `"trigger": "7"`

- RIGHT : `"trigger": "8"`

### Targets

#### Keyboard key names

The targeted keyboard key uses the same names as the ones reported by triggerhappy events they can be displayed using the `thd --listevents` command. Here are some examples:

- Alphabetical keys
  - [A] : `"target": "KEY_A"`
  - [Z] : `"target": "KEY_Z"`
- Number key (not the numpad)
  - [1] : `"target": "KEY_1"`
  - [0] : `"target": "KEY_0"`
- Numpad keys
  - [0] : `"target": "KEY_NUMERIC_0"`
  - [1] : `"target": "KEY_NUMERIC_1"`
  - * : `"target": "KEY_NUMERIC_STAR"`
- Function keys
  - [Esc] : `"target": "KEY_FN_ESC"`
  - [F1] : `"target": "KEY_FN_F1"`
  - [Enter] : `"target": "KEY_ENTER"`

### External sources

Alternatively Pad2keys config files can be scraped off screenscraper.fr, if the option is enabled for it.

You can find some examples from the Content Downloader for Windows games like Ri-Li or Super Mario War, both with embedded pad2key config files.

### Launching commands with the pad2key

As you saw earlier, the pad2key can be used to simulate keyboard presses, however, it can also be used to launch specific commands. Replace the “`type” : “key`” to “`type” : “exec`”, and set the “`target`” to the command.

For example:

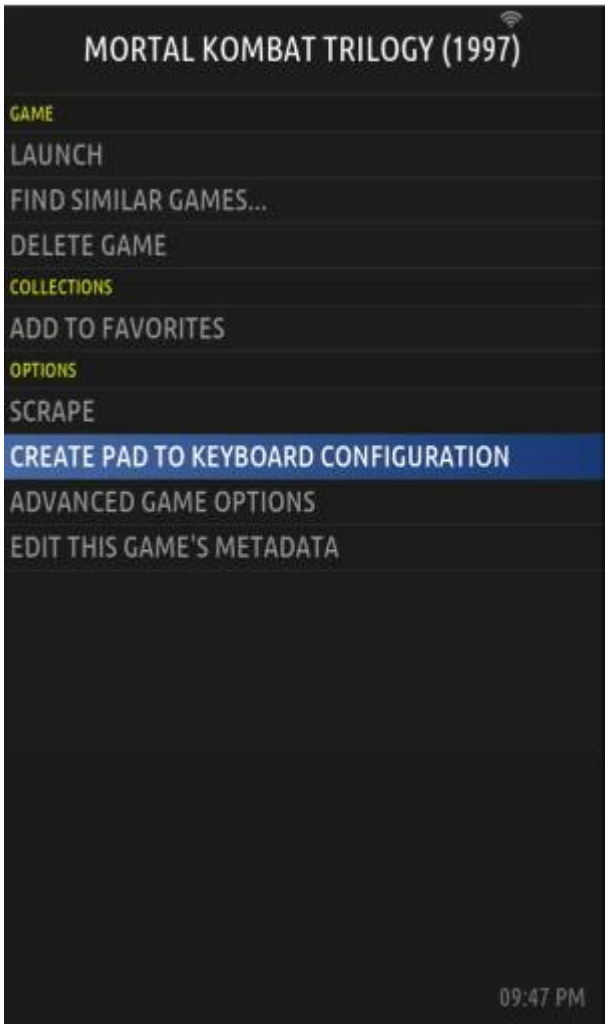
```
        {
            "trigger": [
                "hotkey",
                "pageup"
            ],
            "type": "exec",
            "target": "batocera-screenshot"
        }
```

can be used to launch the `batocera-screenshot` command with the hotkey+L combo, effectively letting you screenshot in systems that don't support a screenshot feature, but that support the pad2key.

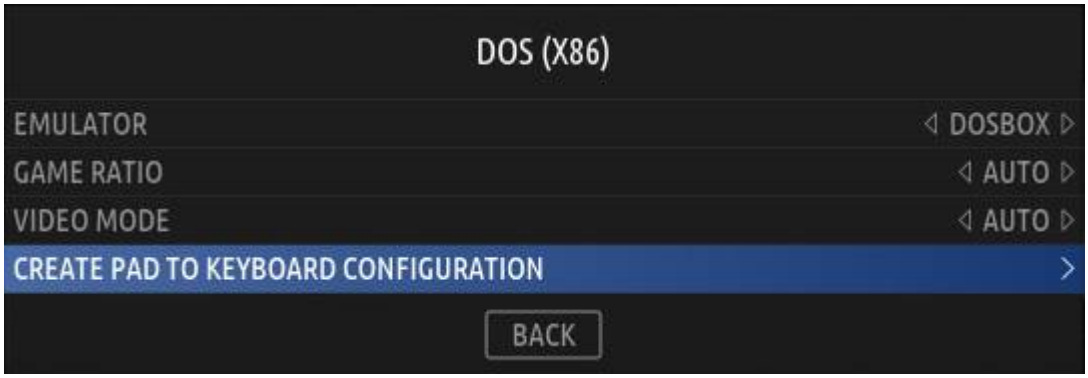
## DosBox standalone

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>
```

### input.xml - V38 and later

```
<inputList>
    <defaultInputList>
        <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" />
    </defaultInputList>
    <!-- Alt inputs for n64 style controllers -->
    <n64InputList>
        <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="B Button" />
        <input name="b"                 value="A Button" />
        <input name="x"                 value="C Button U" />
        <input name="y"                 value="C Button L" />
        <input name="start"             value="Start" />
        <input name="select"            value="Z Trig" />
        <input name="pageup"            value="L Trig" />
        <input name="pagedown"          value="R Trig" />
        <input name="l2"                value="C Button D" />
        <input name="r2"                value="C Button R" />
        <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="" />
        <input name="joystick2down"  value="" />
        <input name="joystick2left"  value="" />
        <input name="joystick2right" value="" />
    </n64InputList>
</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

Remapping Dolphin's controls are a bit more involved, especially since it supports two major systems. It is recommended to watch Batocera Nation's excellent video tutorial on the subject, however a written guide is available below.

## Wii

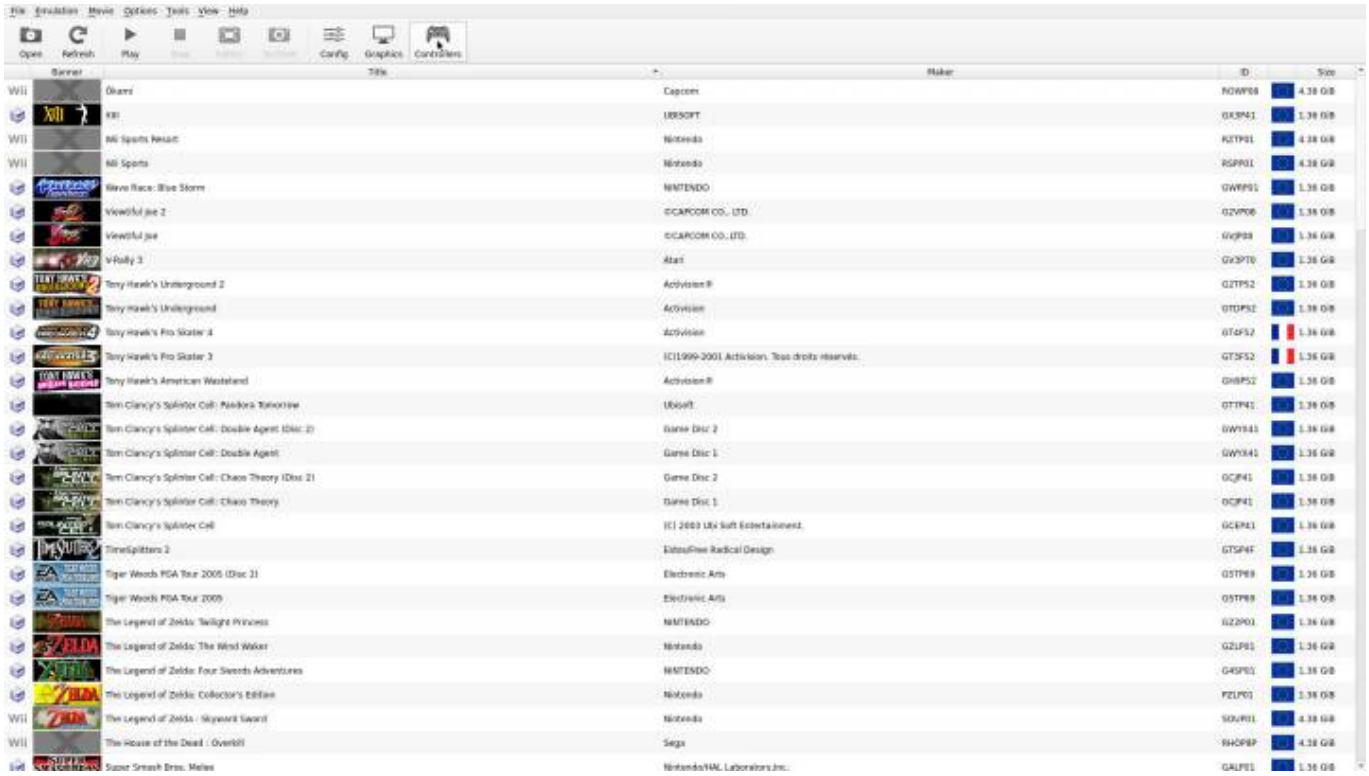
The simplest solution to playing Wii games is to use a real Wiimote connected to your system via a Mayflash adapter bar; this can be set up externally.

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.

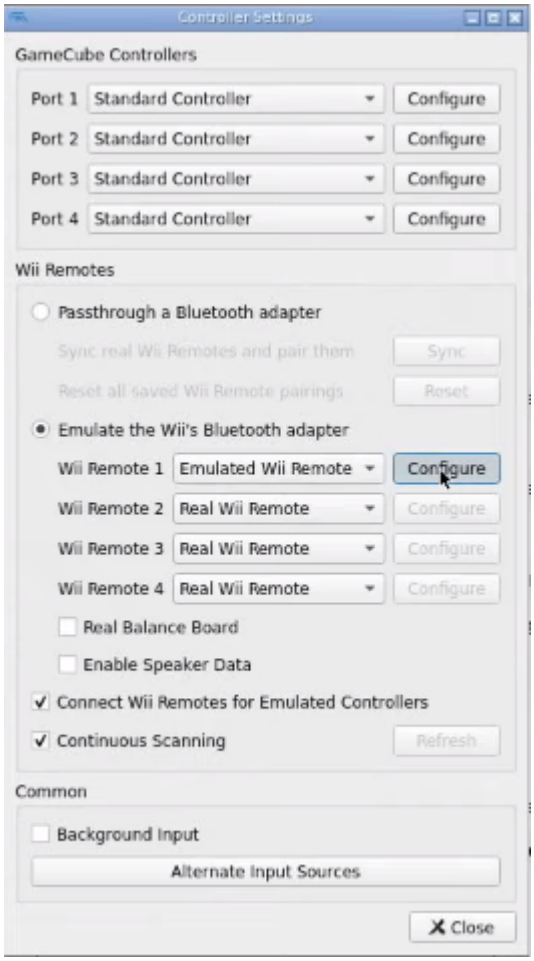
If you prefer to use Dolphin's interface for configuring controls instead of Batocera's preset configurations, read on.

A mouse and keyboard will be required to navigate these menus.

On the system menu, open Files with `[F1]` on the keyboard and navigate to **Applications** → **dolphin-config**. This will open Dolphin's menu. Navigate to **Controllers** in the toolbar (or press `[F1]` on the keyboard).

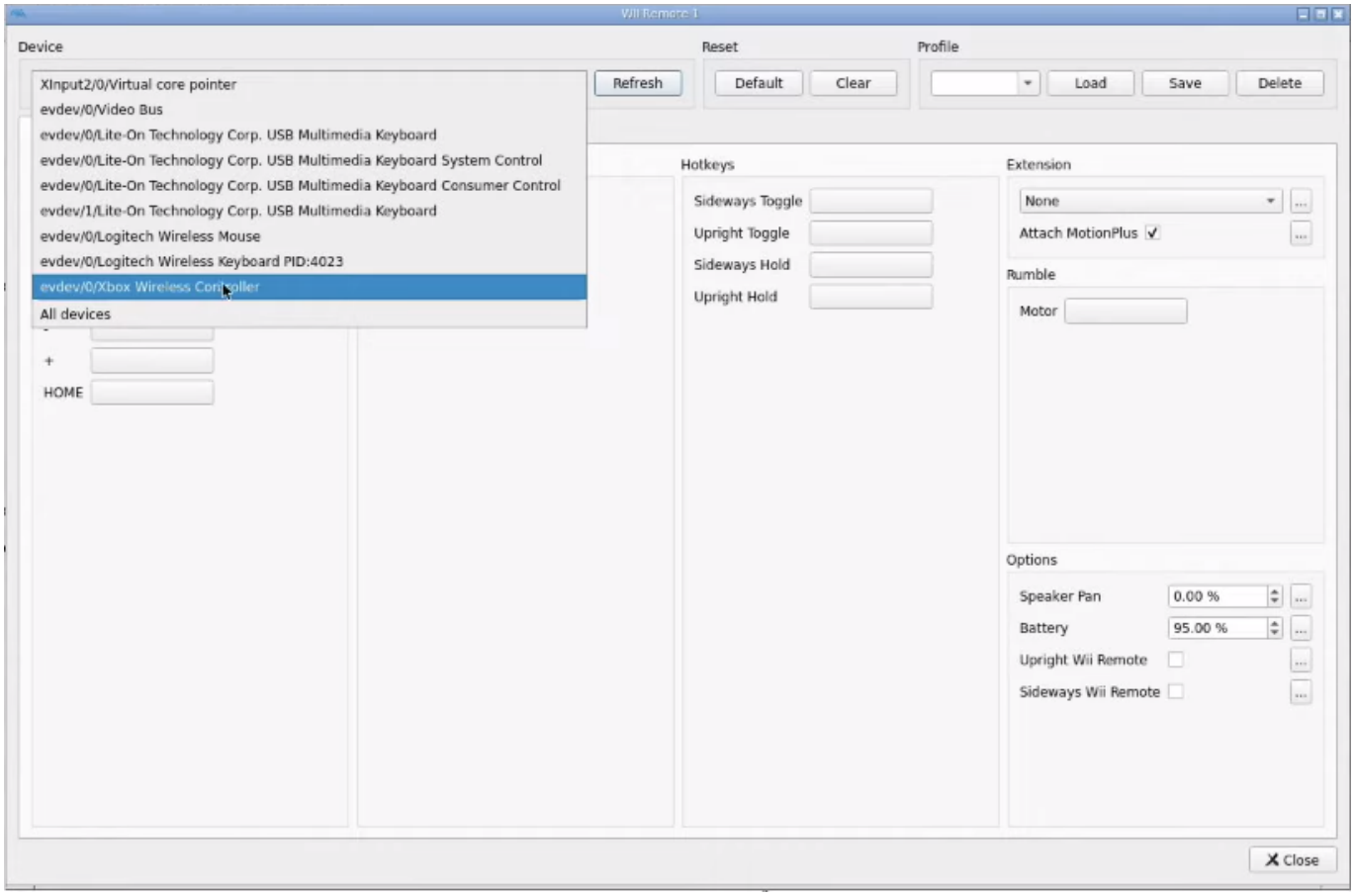


Ensure “Emulate the Wii's Bluetooth adapter” is checked, and that the first Wii Remote is set to “Emulated Wii Remote”, then click **Configure**.

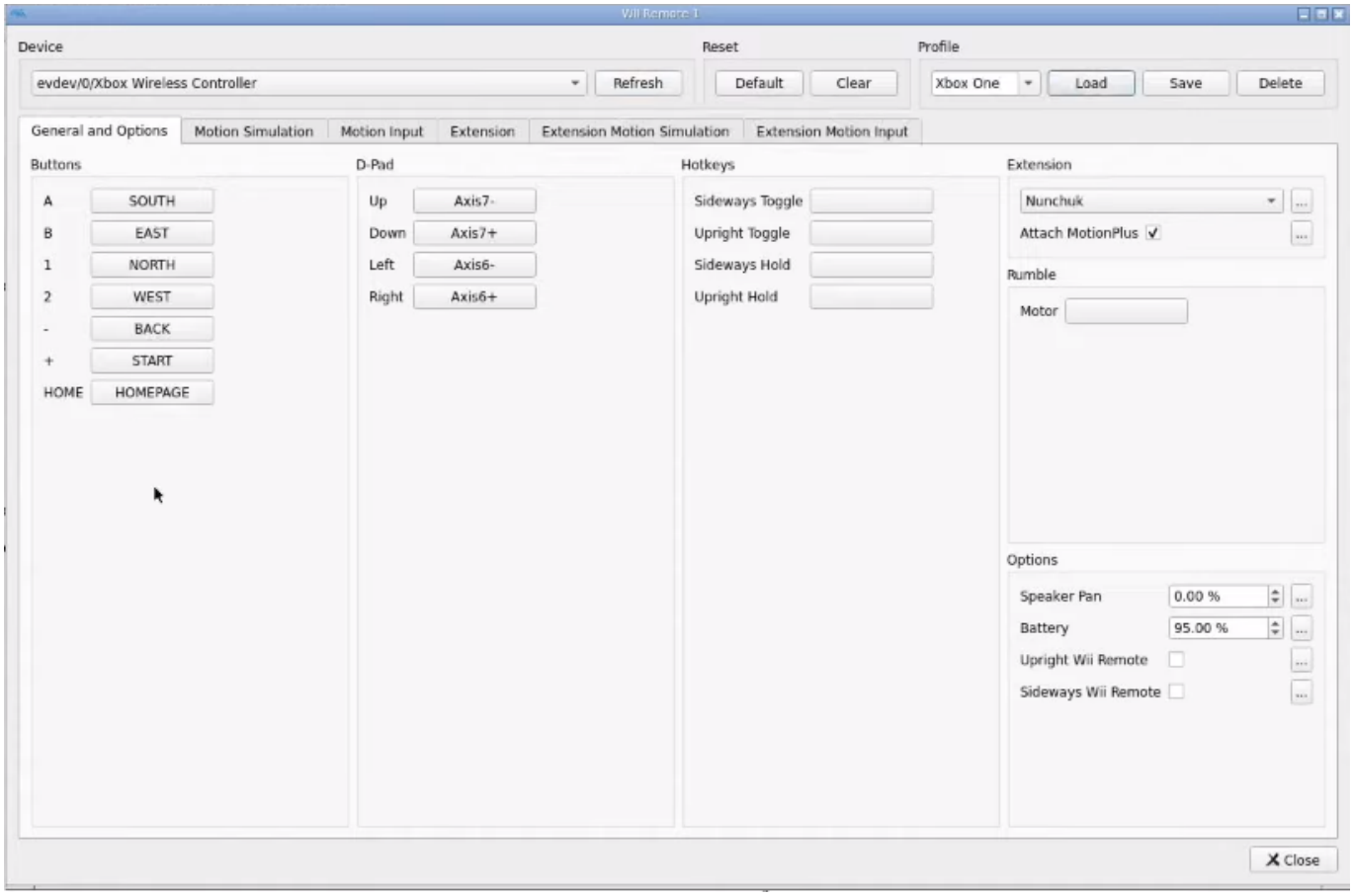



This will open the controller configuration menu. Make sure the controller you intend to use is selected at the top-left of the window. For instance, here the Xbox controller is being selected to

emulate the Wiimote:




Set up your emulated Wiimote as you'd like, and then save your profile in the top-right of the window.

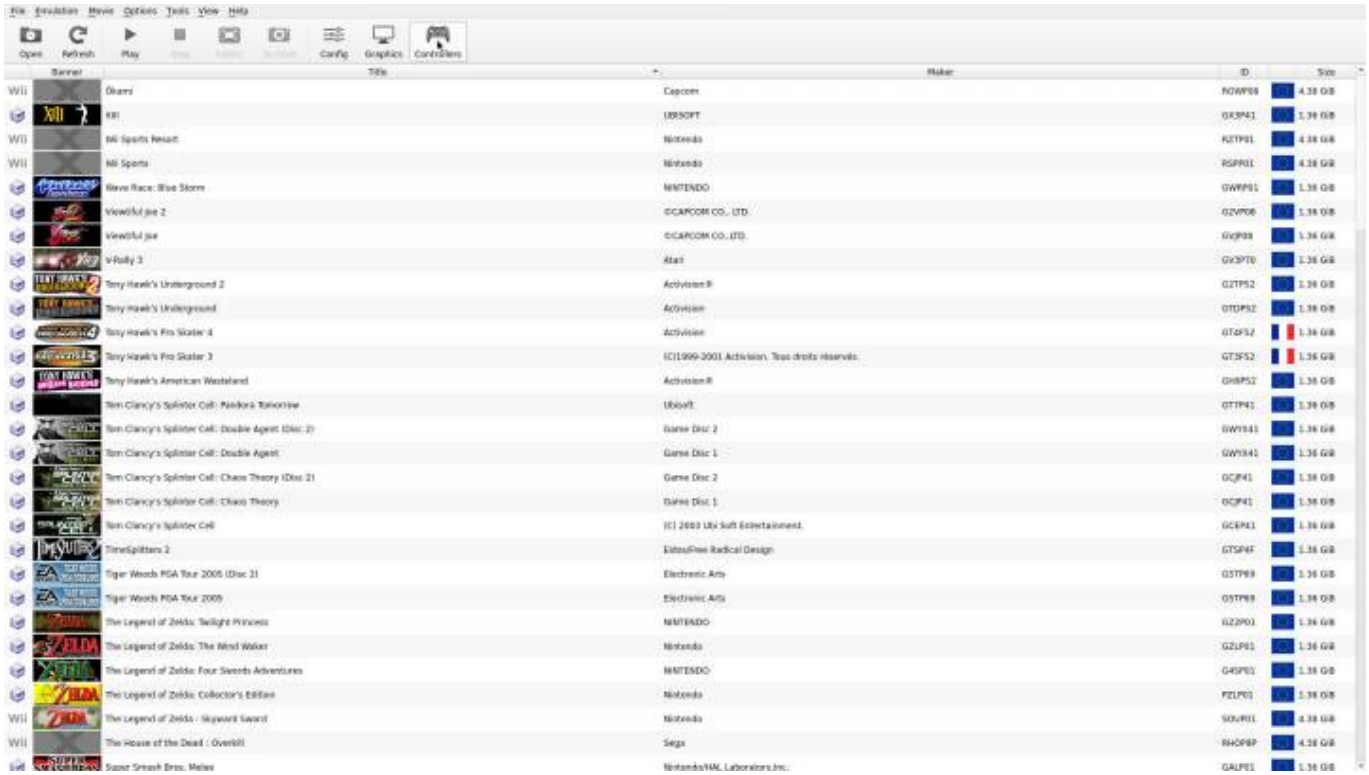


The control panel itself is intuitive and easy to follow, but has extremely powerful scripting features. Further documentation available at https://dolphin-emu.org/docs/guides/configuring-controllers/#Emulated_Wii_Remote.

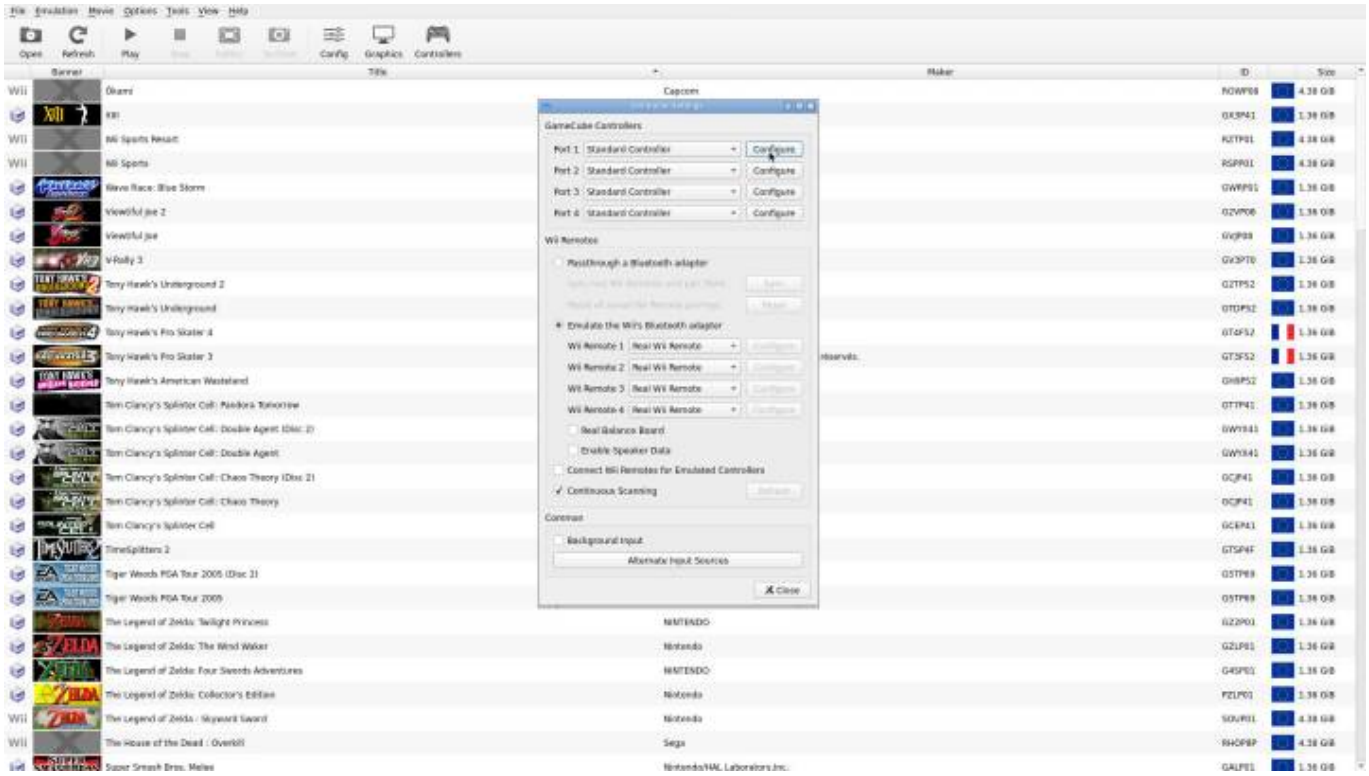
And now your controller is remapped in Dolphin. The profile used will be the last selected profile in Dolphin's controller configuration menu for that controller when no game-specific profile specifies otherwise.

## GameCube

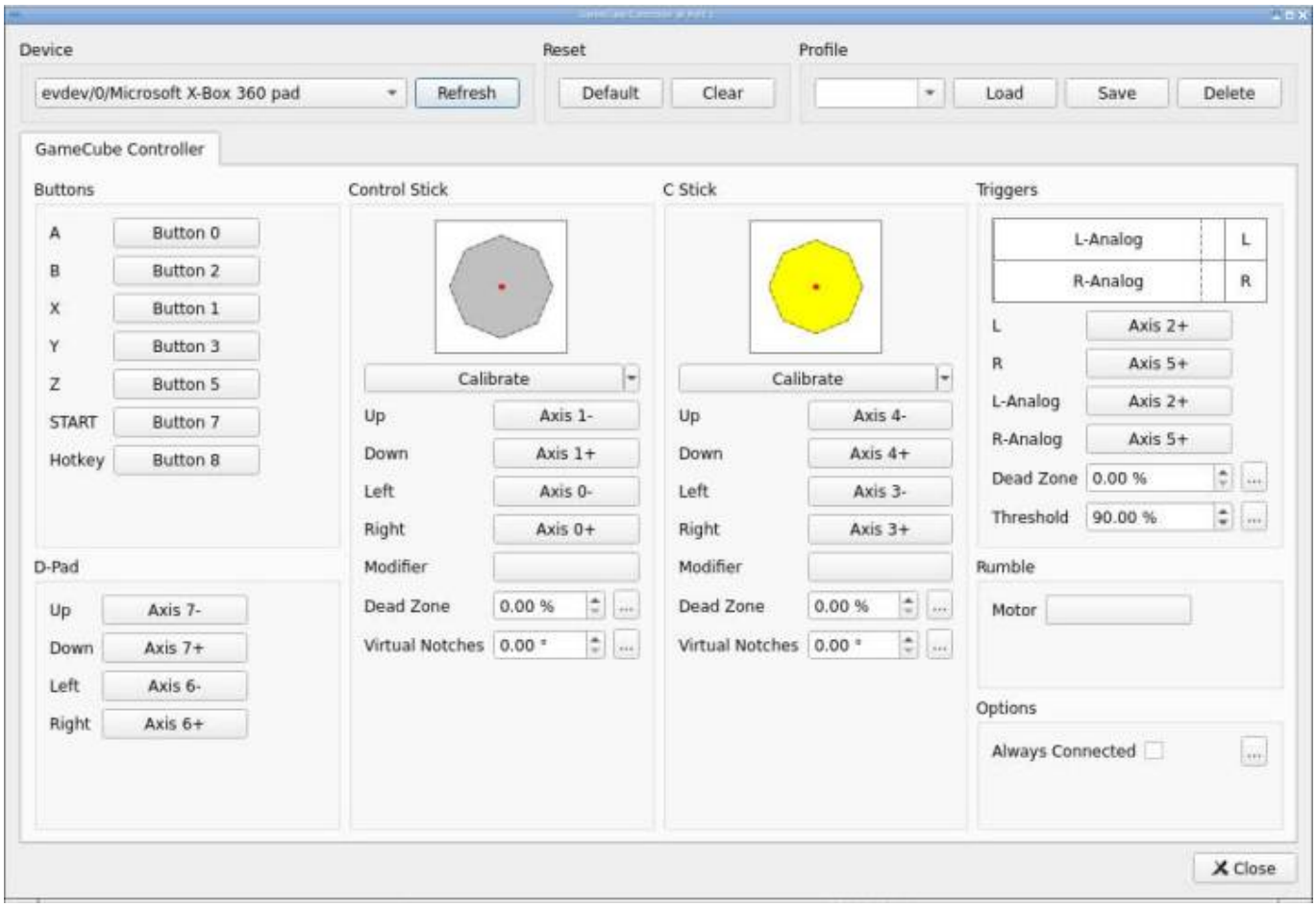
You can configure the controls for this system within its own controller configuration utility (mouse and keyboard required). On the system menu, open Files with [F1] on the keyboard and navigate to **Applications → dolphin-config**. This will open Dolphin's menu.



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

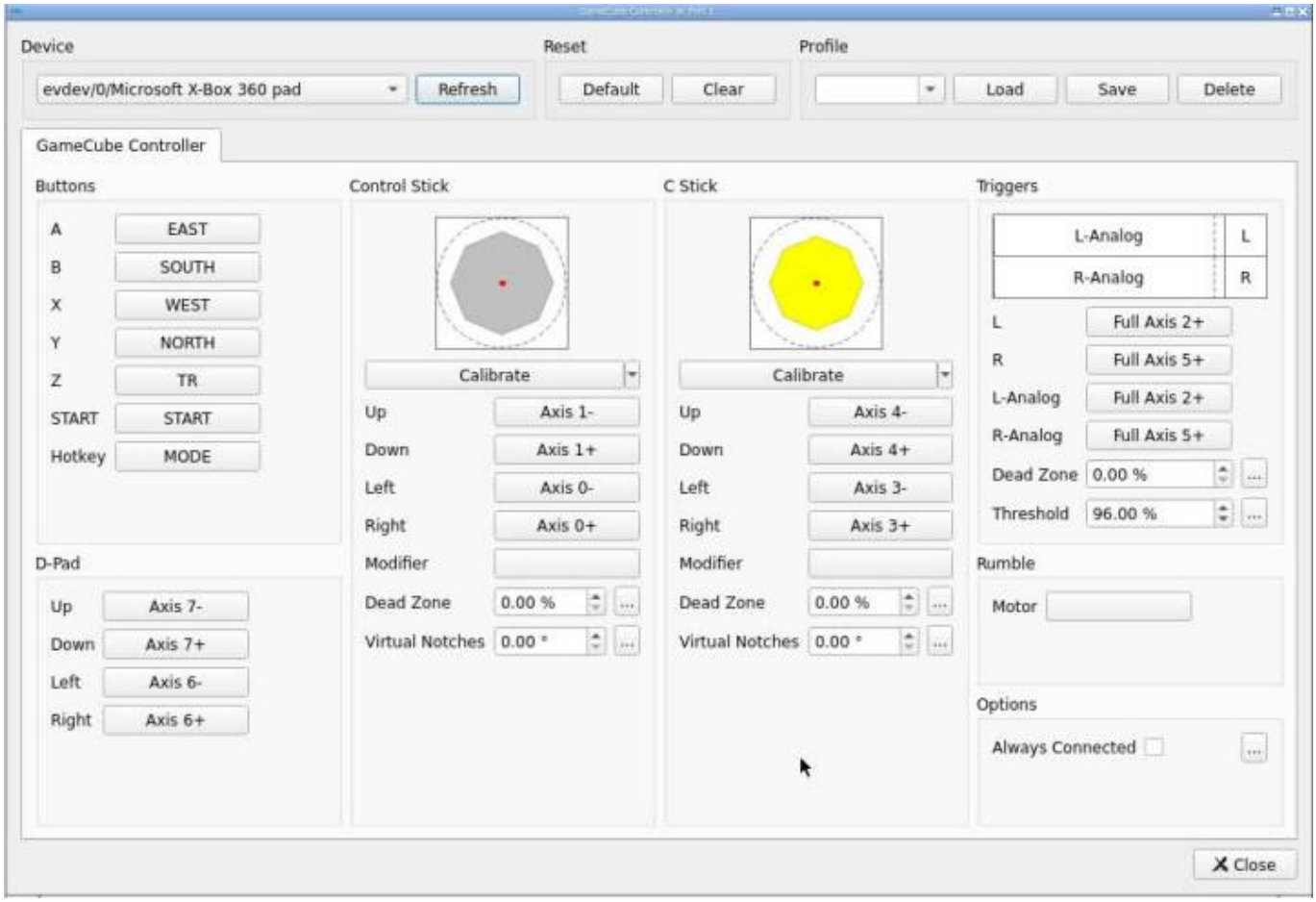


Click on the **Configure** button for Port 1 of the GameCube Controllers. You'll notice that it has a bunch of “Button #” already assigned; this is the default configuration of Dolphin and is unrelated to your current controller.



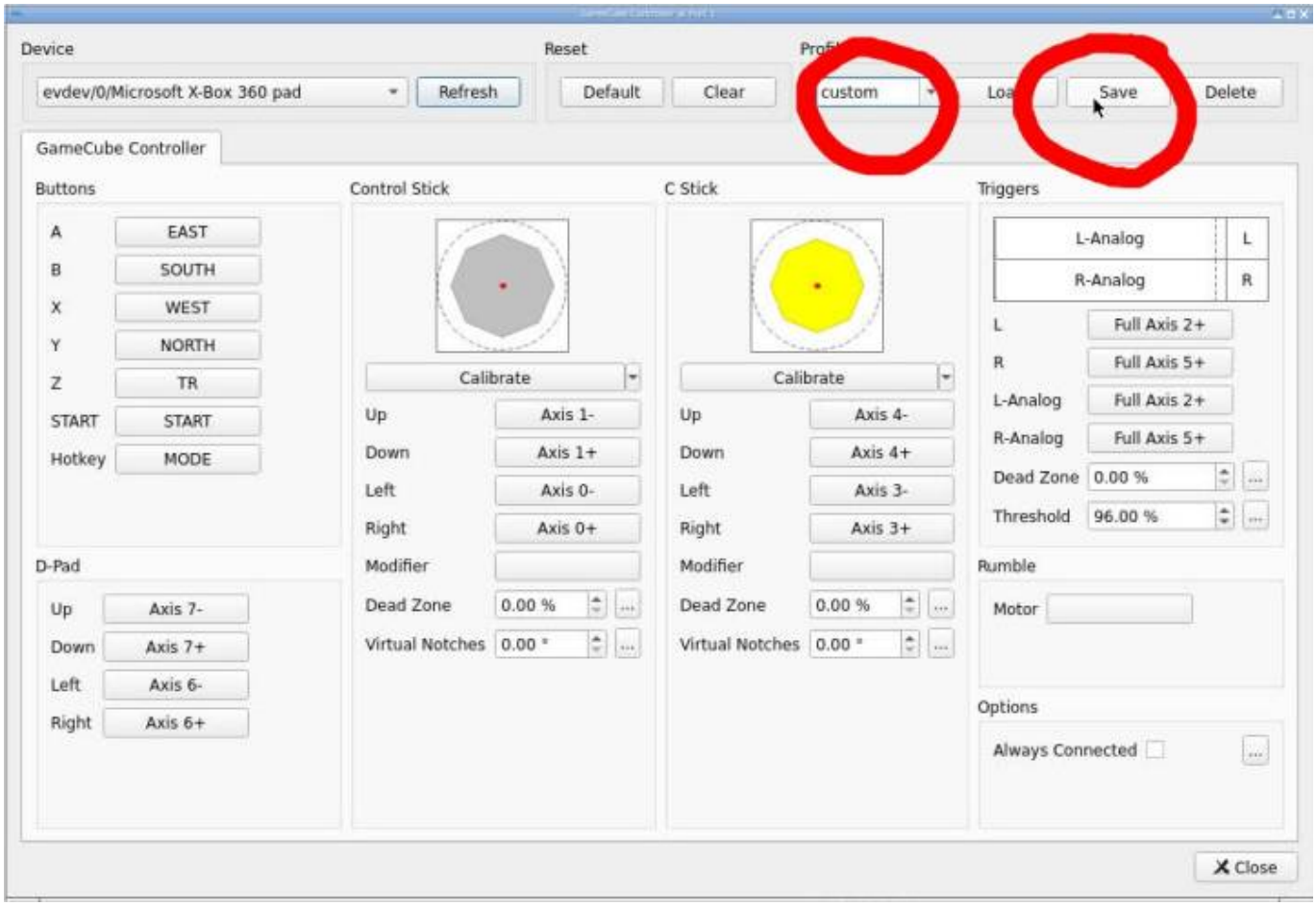

You can configure *system controls* by clicking on them and pressing a button on your *controller*. Let's say I had a SNES-style pad and wanted the labels to match the game's inputs, I would reassign it like

so:



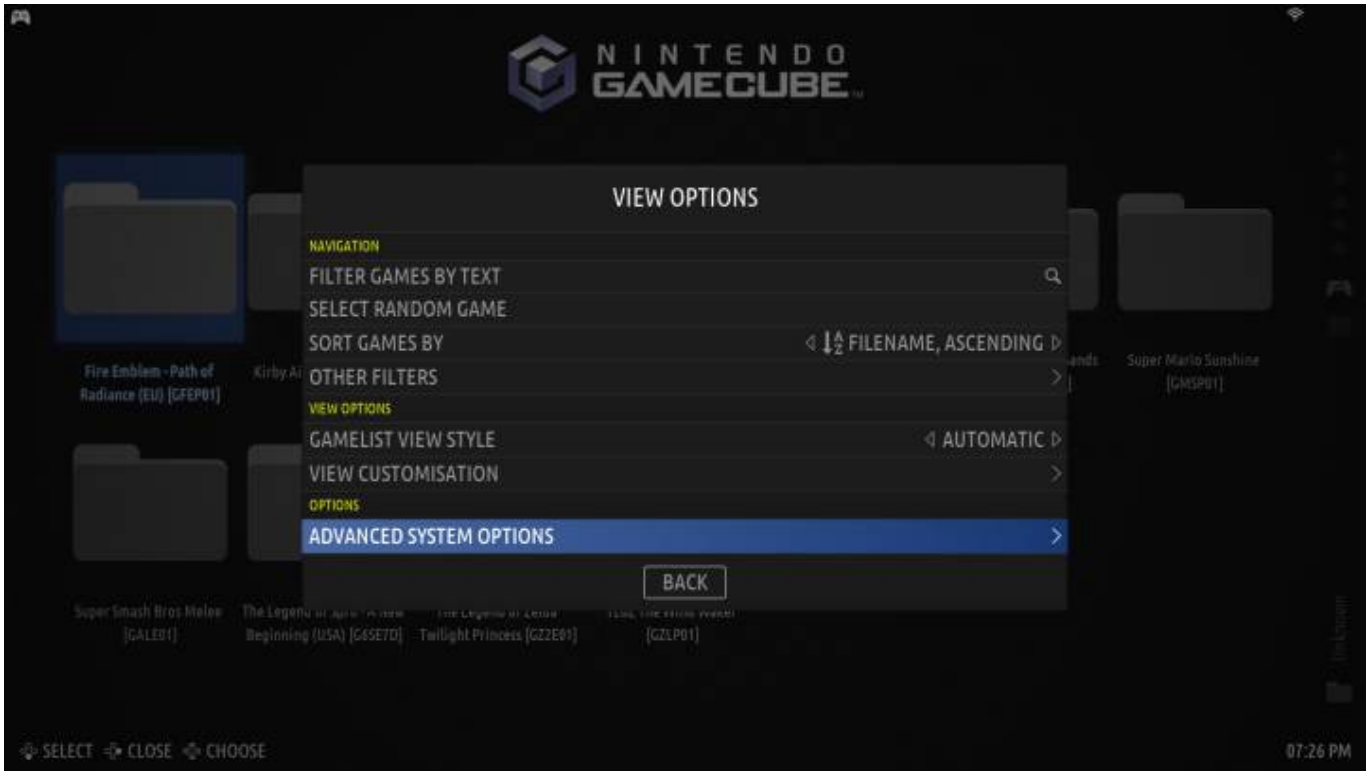

You'll notice that the buttons on your *controller* will turn into cardinal directions. This is the lingo used by Batocera to unify all the various controllers you can connect to it. You might also want to calibrate your sticks, as I've done here, and maybe increase the trigger threshold (good controllers 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**.

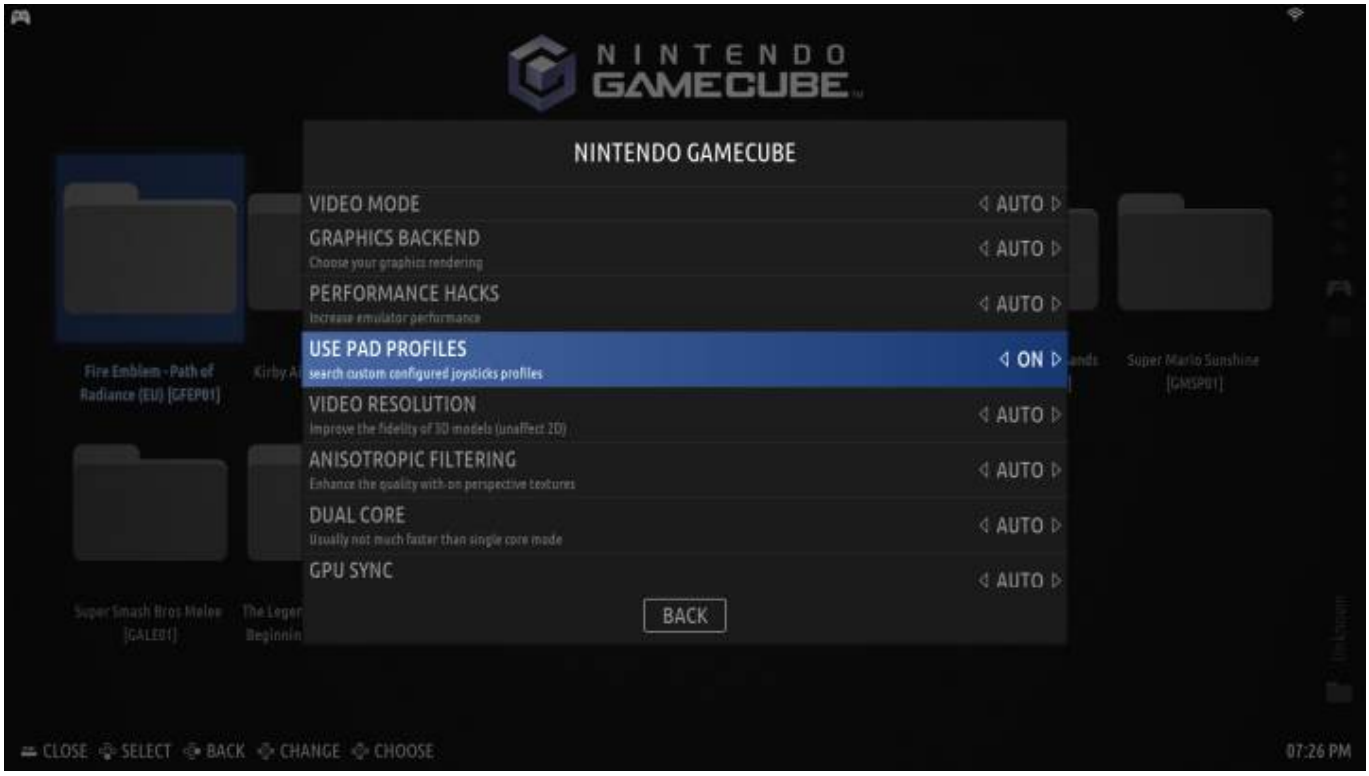



Click **Close** on all the windows and then **File** → **Quit** to leave Dolphin.

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.



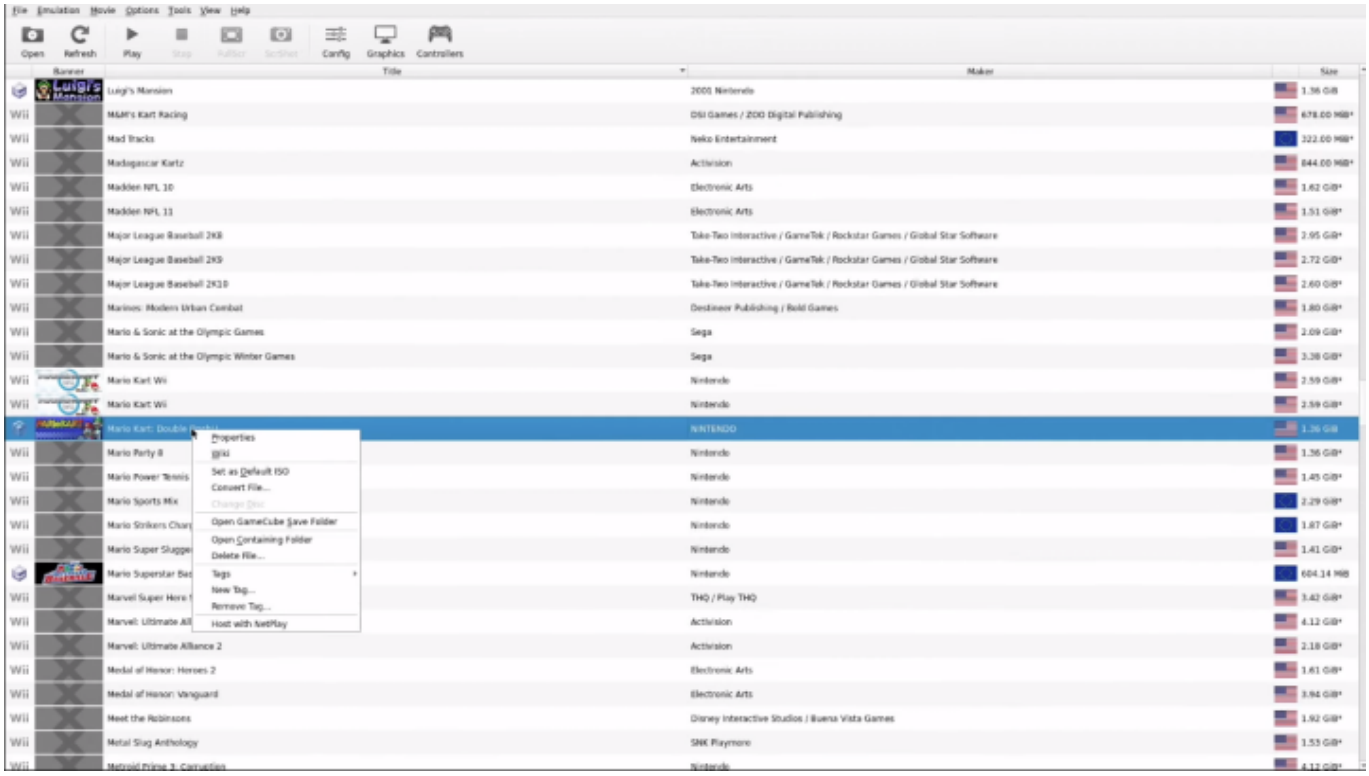

And now your controller is remapped in Dolphin. The profile used will be the last selected profile in Dolphin's controller configuration menu for that controller when no game-specific profile specifies otherwise.

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.

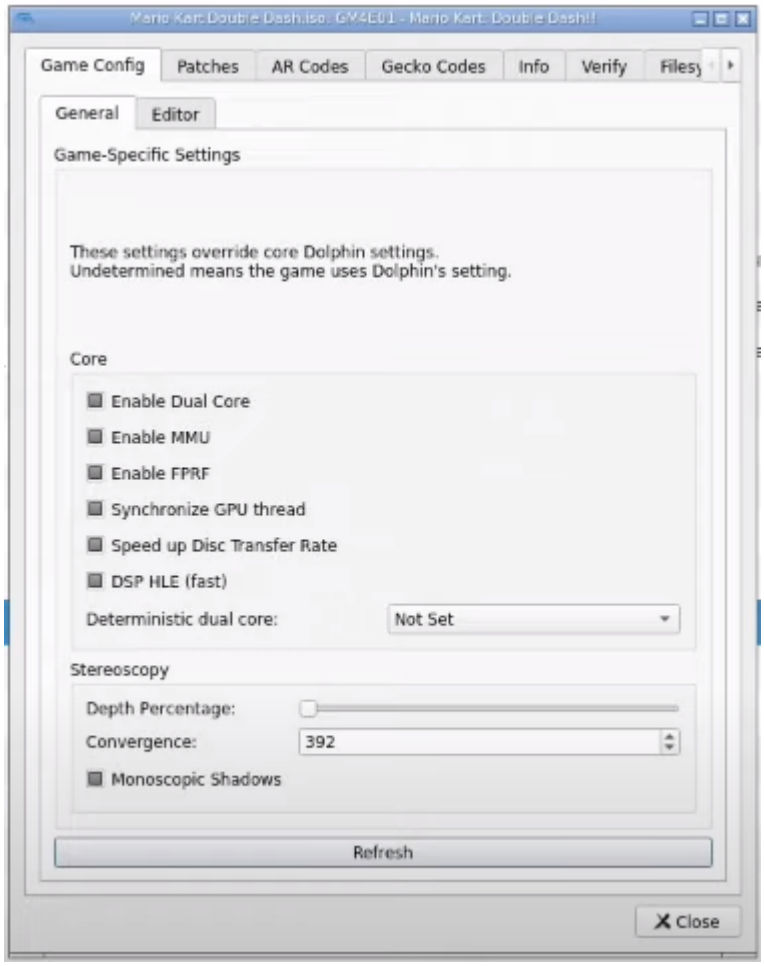
## Save per-game remap profiles

Controller remaps can be applied to specific games via `dolphin-emu-config` as well.

After creating and saving a custom profile, go back to Dolphin's main menu and right click on the game you'd like to use the custom profile for.




Go into **Properties**.




Go into **Editor**.


Then in the “User Config” text box, ensure the dropdown menu is set to “Presets” and type the following into the window:

```
[Controls]
PadProfile1 = <profile>
```

where `<profile>` is the name of the custom profile you'd like to use for that game.

For setting this profile for more than just player 1, append the following:

```
PadProfile2 = <profile>
PadProfile3 = <profile>
PadProfile4 = <profile>
```

Not all players have to use the same profile. An example user config might look like this:

```
[Controls]
PadProfile1 = triggers on L2 R2
PadProfile2 = triggers on L2 R2
PadProfile3 = triggers on L1 R1
```

```
PadProfile4 = Peters super cool custom config
```

## 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.

## Default light gun mappings

Here are the default light gun mappings, in case you're configuring an emulator which does not support “press the button to map” remapping functionality:

| Global keyboard/mouse code | Wiigun button | Wiigun code | Sinden button | Sinden code | GUN4IR button | GUN4IR code | AimTrack button | AimTrack code | AElightgun button | AElightgun code | Dolphinbar button | Dolphinbar code | GunCon2 button | GunCon2 code |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BTN_LEFT | B | BTN_EAST | Trigger | BTN_LEFT | Trigger | BTN_LEFT | Trigger | BTN_LEFT | Trigger | BTN_LEFT | B | BTN_LEFT | Trigger | BTN_LEFT |
| BTN_RIGHT | A | BTN_SOUTH | Pump/Front left | BTN_RIGHT | Offscreen/A/C | BTN_RIGHT | Right side | BTN_RIGHT | Left mouse | BTN_RIGHT | A | BTN_RIGHT | C | BTN_RIGHT |
| BTN_MIDDLE | PLUS | KEY_NEXT | Back left | BTN_MIDDLE | B | BTN_MIDDLE | Left side | BTN_MIDDLE | Middle mouse | BTN_MIDDLE | – | – | START | BTN_MIDDLE |
| BTN_1 | MINUS | KEY_PREVIOUS | Front right | KEY_1 | START | KEY_1 | – | – | 1 | BTN_1 | – | – | SELECT | BTN_1 |
| BTN_2 | 1 | BTN_1 | Back right | KEY_2 | SELECT | KEY_5 | – | – | 2 | BTN_2 | – | – | A | BTN_2 |
| BTN_3 | 2 | BTN_2 | – | – | – | – | – | – | 3 | BTN_3 | – | – | B | BTN_3 |
| BTN_4 | HOME | BTN_MODE | – | – | – | – | – | – | 4 | BTN_4 | – | – | – | – |
| BTN_5 | ↑ | KEY_UP | ↑ | KEY_UP | ↑ | KEY_UP | – | – | Up | KEY_UP | – | – | ↑ | 5 |
| BTN_6 | ↓ | KEY_DOWN | ↓ | KEY_DOWN | ↓ | KEY_DOWN | – | – | Down | KEY_DOWN | – | – | ↓ | 6 |
| BTN_7 | ← | KEY_LEFT | ← | KEY_LEFT | ← | KEY_LEFT | – | – | – | – | – | – | ← | 7 |
| BTN_8 | → | KEY_RIGHT | → | KEY_RIGHT | → | KEY_RIGHT | – | – | – | – | – | – | → | 8 |