Do-It-Yourself Arcade Controls

Under construction.

Introduction

Some controllers, mainly arcade panels, use IPAC or JAMMASD controllers.

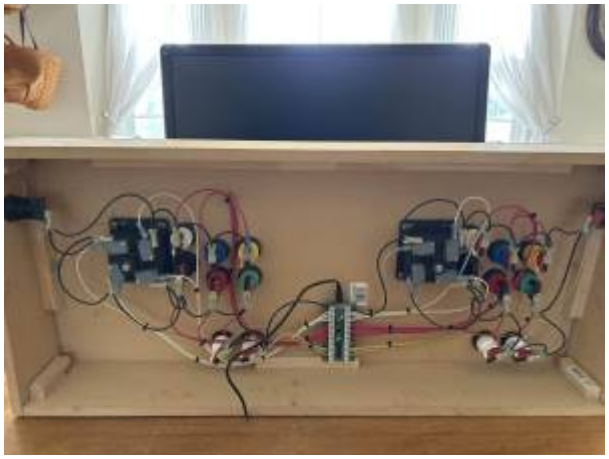
This is the case for custom made controllers or for industrial arcade machines.

These controllers are seen as keyboards and should be converted to virtual controllers to give the best results.

Since Batocera 41, we support two main new features :

- `hotkeygen` : a way to handle hotkeys like “exit game”, “go in menu”, even if you use regular pads
- keyboards to pads/hotkeys: to convert keyboards to pads and hotkeys. This feature was partially done using Arcade2Joystick

Gallery




ARCADE GAMES

Arcade machine seller respecting licences

Most premade arcade machines sellers are not respecting batocera and emulators licences.

Here is a list of 2 sellers respecting licences :

- https://iconicarcade.com/
- https://recroommasters.com/

Contextual hotkeys

Contextual hotkey is a Batocera way to map hotkeys to actions in emulators. For each emulator, the emulator hotkeys are referenced and mapped to Batocera hotkeys.

The current list of Batocera hotkeys is:

Batocera hotkey	Keyboard key equivalent
exit	KEY_EXIT
coin	KEY_EURO
menu	KEY_MENU
pause	KEY_PAUSE
files	KEY_FILE
save_state	KEY_SAVE
restore_state	KEY_SEND
next_slot	KEY_NEXT
previous_slot	KEY_PREVIOUS
screenshot	KEY_SYSRQ
volumeup	KEY_VOLUMEUP
volumedown	KEY_VOLUMEDOWN
volumemute	KEY_MUTE
brightness-cycle	KEY_BRIGHTNESS_CYCLE

For example, while Emulationstation is running, the `hotkeygen –list` command will return something similar to :

```
$ hotkeygen --list
Context [emulationstation]:
  exit------------> KEY_ESC
  menu------------> KEY_SPACE
  files-----------> KEY_F1
# device /dev/input/event5 [Ultimarc I-PAC 4] (no Ultimarc_IPAC_4-
d209-430.mapping file found)
  KEY_PAUSE-------> pause
# device /dev/input/event3 [AT Translated Set 2 keyboard] (no
AT_Translated_Set_2_keyboard-01-01.mapping file found)
  KEY_PAUSE-------> pause
# device /dev/input/event23 [Xtension hotkeys] (no
```

```
Xtension_hotkeys-00-00.mapping file found)
  KEY_EXIT--------> exit------------> KEY_ESC
  KEY_MENU--------> menu------------> KEY_SPACE
  KEY_PAUSE-------> pause
```

This means:

- the current context is `emulationstation`
- the hotkey “exit” will send the equivalent of the keyboard KEY_ESC (escape key)
- the hotkey “menu” will send the equivalent of the keyboard KEY_SPACE (space bar)
- the hotkey “files” will send the equivalent of the keyboard KEY_F1 (F1 key)
- 3 devices are able to generate these events (via KEY_EXIT, KEY_MENU, and so on)
- there is no custom mapping file found for any of the 3 devices. The standard hotkeys will be applied.

Another example of context while running `ppsspp`:

```
$ hotkeygen --list
Context [ppsspp]:
  exit------------> ['KEY_LEFTALT', 'KEY_F4']
  save_state------> KEY_F3
  restore_state---> KEY_F4
  menu------------> KEY_F9
  pause-----------> KEY_F9
  next_slot-------> KEY_F6
  previous_slot---> KEY_F5
$
```

You can also simulate the hotkeys from the command line. Within any context, for example, the hotkey “exit” will quit the game. It can be launched via the command:

```
hotkeygen --send exit
```

And when available, the following command will pause the emulator. With most emulators, the hotkeys menu and pause do the same:

```
hotkeygen --send pause
```

If you add the parameter –debug, you can get more details. For example, the command `hotkeygen –list –debug` will give you the paths used to customize your contexts.

Extend hotkeygen config

Defaut hotkeygen config is available in /etc/hotkeygen. Files common_context.conf and default_context.conf can be created in the folder /userdata/system/configs/hotkeygen. If any, they will be merged with the default config.

- common_context.conf is used to define hotkeys always available, whatever the context, like taking a screenshot
- default_mapping.conf is used to define which keys set which action. In theory, you don't need to

customize this file.

You need to restart the hotkeygen service so that any update on these file is taken.

Arcade Controllers (keyboadToPads)

Some controllers are marked to use keyboardToPads, so that they can be converted to one or multiple pads.

To check whether your pad is eligible, just run:

```
$ keyboardToPads --search
device /dev/input/event3 : "Ultimarc I-PAC 4"
  config file name : UltimarcIPAC4.vd209.p0430.yml
  system config found at
/usr/share/keyboardToPads/inputs/UltimarcIPAC4.vd209.p0430.yml
  you can create a custom config at
/userdata/system/configs/keyboardToPads/inputs/UltimarcIPAC4.vd209.p0430.yml
. Take example on files in /usr/share/keyboardToPads/inputs.
```

It will display your pads if any, and the file you need to create in order to customize it.

Files are in yaml format.

There are some file samples in `/usr/share/keyboardToPads/inputs`.

Example :

```
# Xtension 2 Player Plus
target_devices:
  - name: Xtension 2P Player 1
    type: joystick
    mapping:
      "key:up":        "abs:hat0y:-1"
      "key:down":      "abs:hat0y:1"
      "key:left":      "abs:hat0x:-1"
      "key:right":     "abs:hat0x:1"
      "key:leftshift": "btn:south"
      "key:enter":     "btn:south"
      "key:z":         "btn:east"
      "key:leftctrl":  "btn:west"
      "key:leftalt":   "btn:north"
      "key:space":     "btn:tl"
      "key:x":         "btn:tr"
      "key:c":         "btn:start"
      "key:1":         "btn:start"
      "key:v":         "btn:select"
      "key:5":         "btn:select"
  - name: Xtension 2P Player 2
    type: joystick
```

```
    mapping:
      "key:r": "abs:hat0y:-1"
      "key:f": "abs:hat0y:1"
      "key:d": "abs:hat0x:-1"
      "key:g": "abs:hat0x:1"
      "key:w": "btn:south"
      "key:i": "btn:east"
      "key:a": "btn:west"
      "key:s": "btn:north"
      "key:q": "btn:tl"
      "key:k": "btn:tr"
      "key:j": "btn:start"
      "key:2": "btn:start"
      "key:1": "btn:select"
      "key:6": "btn:select"
  - name: Xtension hotkeys
    type: hotkeys
    mapping:
      "key:tab": "key:menu"
      "key:p":   "key:pause"
      "key:esc": "key:exit"
```

In that example, the controller is split into 3 virtual devices: 2 pads and 1 hotkey device.

Note that once you've created this file, you just need to unplug and replug the device so that it is taken into account.

Note that the naming and mapping of the device just create a virtual pad, but you will have to configure it via Emulationstation as usual for any controller (as it wouldn't be in our database yet).

The name of the first virtual pad is `Xtension 2P Player 1`. Rename Xtension 2P Player 1 & 2 to the name of your controller. Its type is “joystick”. Possible types are “joystick” for pads, and “hotkeys” for hotkeys.

`mapping` is a key-value association table to convert each key from the real device into the virtual pad. You can assign two or more physical keys to a single pad button.

You can use the following command to find the input keys :

```
evsieve --input /dev/input/event3 --print
```

You may also want to use `btn:tl2` and `btn:tr2` (left and right second triggers) depending on your context.

Troubleshooting

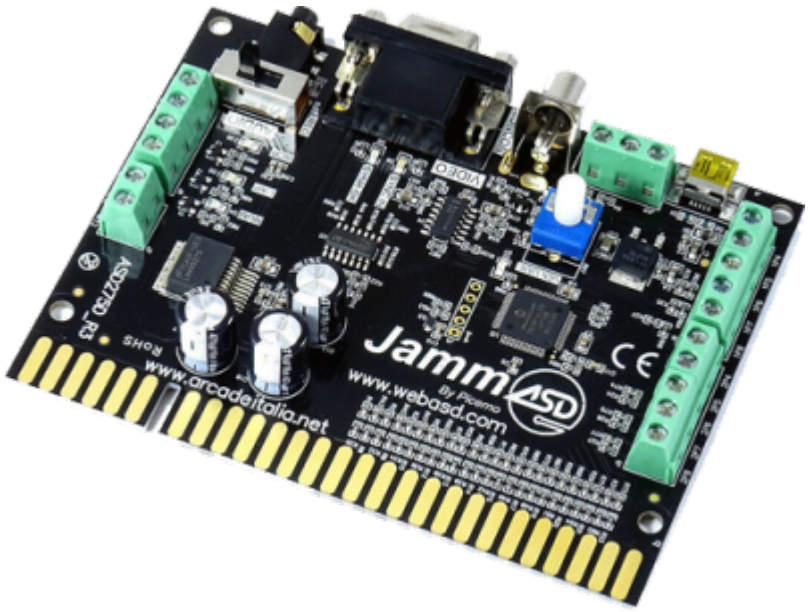
In case of error, you can try to run this command (with the event number corresponding to your setup):

```
keyboardToPadsLauncher /dev/input/event3 run
```

Before batocera 41

JammASD

This guide will help you to use your original arcade cabinet controls (sticks and buttons) under Batocera. The simplest way actually, is to use a **JammASD controller**.



For the moment, this controller (used to send VGA signal, sound, and control information to your JAMMA connector) is not known by EmulationStation, but a few tips can help you to use Batocera in your original cabinet.

JammASD and Emulation Station

Under EmulationStation, JammASD's controls are considered as a keyboard, and it's not the easiest way to configure 2 players controllers.

So, you have to change your JammASD drivers into X-Arcade drivers to use all your arcade panel.

1. Connect to Batocera using SSH

2. Type:

```
ls /dev/input/by-id
```

You will see this interface : *usb-ASD_JammASD_Interface_ASD275D-if01-event-kbd*

```
touch /userdata/system/configs/xarcade2jstick/usb-
ASD_JammASD_Interface_ASD275D-if01-event-kbd
```

3. Reboot your Batocera

Now, you only have to edit your controls under EmulationStation menu. Then, your arcade panel will

be known as *Xarcade-to-Gamepad Device 1* and *Xarcade-to-Gamepad Device 2*

```
Note: With this tip, it's possible now to have START P1 + A to credit your arcade games ^^
```

Ultimarc

This instructions are for Batocera 5.24 or later, they are dedicated to Ultimarc's line of control interfaces (including I-PAC, J-PAC, A-PAC, Mini-PAC, Opti-PAC, U-HID and more) and X-Arcade USB Controllers.

TL;DR: If your encoder is any model of I-PAC or Mini-PAC, press and hold the buttons corresponding to Start1+P1SW2 for ten seconds. If you're lucky, your controller is now a plug 'n play USB controller you can configure like any other controller in batocera. That's it. You're done. Go play some games 😎

If nothing happens or you want to know more, keep reading.

The above will only work with Ultimarc's encoders (they call them control interfaces) made from 2015 on and equipped with Firmware version 50 or higher. You can check what you have with the software they offer on their website. **This is important as flashing your pre-2015 encoder with the new firmware will brick it!** If your board is new enough, but your firmware isn't, you can update with the same tool. Read the information on their page under **“Multi-Mode”** to learn about the different modes and what you can do with them. The instructions above will make your controller run in Mode 2, which should be fine for most users. What this means is that instead or being recognised as a USB keyboard, your controller will now be recognized as a D-input game controller and you won't need to do any of the steps below.

If your Ultimarc encoder doesn't offer this functionality, you don't want to upgrade the firmware or you are using an X-Arcade controller, read on.

Xarcade2jstick has been patched to support keyboard encoders. You may have to reconfigure a few keys of your encoder as, sadly, x-gaming X-Arcade devices didn't map as usual mame controllers.

Do not change the key mappings of your I-PAC or Mini-PAC as it is unnecessary, but the following procedure only applies if your encoder is in keyboard mode, which is the default mode when they leave the factory.

Configuring your keyboard encoder

Follow these steps :

- login to your Batocera box locally or through SSH
- find your encoder's device name with `ls /dev/input/by-id`. Usually, there is a trailing kbd in the event name. For example : `usb-Ultimarc_IPAC_2_Ultimarc_IPAC_2_9-if01-event-kbd`

PLEASE NOTE: a single encoder can have multiple possible names, so try all of them
For exemple: `usb-Cypress_I-PAC_Arcade_Control_Interface-event-kbd` → works
`usb-Cypress_I-PAC_Arcade_Control_Interface-if01-event-kbd` → doesn't work

- Now remount / as read-write `mount -o remount,rw /`
- Create an empty file that has the same name of your keyboard device found 2 steps above `touch /usr/share/batocera/datainit/system/configs/xarcade2jstick/devicename`. With the previous example : `touch /usr/share/batocera/datainit/system/configs/xarcade2jstick/usb-Ultimarc_IPAC_2_Ultimarc_IPAC_2_9-if01-event-kbd`
- Edit `batocera.conf` and set `controllers.xarcade.enabled=1` (it should already be set by default but check it)
- Save your modifications through SSH `batocera-save-overlay`
- Reboot by typing `reboot`

Now all your keys should be usable, that is for each player : 4 directions joystick, select, start, hotkey, A, B, X, Y, L1 and R1.
For now it is not possible to attribute more buttons like L2, R2, L3 or R3

You must now configure both the controllers in the Controller Configuration menu and also associate the right controller to player 1 and 2.

Since Batocera **v34**, Ultimarc has been included. Read more here.

Troubleshooting

The entire article is under construction but this part is *especially* under construction.

I use Dragonrise encoders and player 1 and player 2's inputs are swapped

For Batocera 40 and newer, the fix has been reported as not being necessary any longer, but this was an issue in previous versions.

Make sure they are all wired exactly the same, each button going to the same USB encoder.

If that's not working or is impossible to do due to your setup's requirements, it's possible to work around by using a USB quirk. Add the following onto the end of your boot line (on Raspberry Pi images, this is at `/boot/uEnv.txt`, x86_64 users can refer to here):

```
usbhid.quirks=0×0079:0×0006:0×040
```

It's added onto the end of the line, not on a new line. For instance, your boot line might end up looking like this:

```
    APPEND=label=BATOCERA rootwait quiet loglevel=0 console=ttyAML0,115200n8
console=tty3 vt.global_cursor_default=0 usbhid.quirks=0×0079:0×0006:0×040
```

I have a two devices that are recognized as mice and I have to reconfigure them every launch

Arcade trackballs and spinners are handled as mouse devices.

The problem comes from the way newer versions of RetroArch recognize mouse devices, by ID instead of name. The ID that they can have is thus unpredictable, resulting in configs (such as `retroarchConfig['input_player1_mouse_index'] = "Barcode Reader Mouse"`) no longer working.

For now, this can be worked around using a script to automatically log and change the ID as appropriate. Save it to `/userdata/system/configs/emulationstation/scripts/game-start/mouse-fix.sh` and mark it as executable with `chmod +x /userdata/system/configs/emulationstation/scripts/game-start/mouse-fix.sh`:

mouse-fix.sh

```
#!/bin/bash

# NAME OF DESIRED MOUSE INPUT
# Can be found via the RetroArch log file or by running 'evtest'
mouse_name="Combined Analog Arcade Controls"

# RetroArch log file must be enabled for this to work
batocera-settings-set global.retroarch.log_dir
"/userdata/system/logs/retroarch"
batocera-settings-set global.retroarch.log_to_file true
batocera-settings-set global.retroarch.log_to_file_timestamp false

# Read the mouse index values from the last RetroArch log file
# and update the config for the next time RetroArch is run
# NOTE: Using '~' as a sed delimiter as some device names include the
traditional '/' delimiter
# NOTE: Pipe to 'head -1' to return the index of the first matching
device, as some devices expose multiple inputs
mouse_index=$(sed -En "s~.*Mouse.* #(.*): \"$mouse_name\".*~\1~p"
/userdata/system/logs/retroarch/retroarch.log | head -1)
if [[ -z "$mouse_index" ]]; then
    mouse_index=0
fi
batocera-settings-set global.retroarch.input_player1_mouse_index
$mouse_index
```

You may need to launch a game twice in order for it to have an effect.

Keep updated by watching this space: https://github.com/libretro/RetroArch/issues/7638. Original forum post describing this issue and its workaround: https://forum.batocera.org/d/6652-being-able-to-use-trackball-and-spinner-using-per-mouse-index

Batocera may make the workaround process easier in the future by the inclusion of `evsieve`.

Standalone MAME has its own mechanism for enabling spinner, trackball, and other mouse devices. In **ADVANCED GAME OPTIONS**, select **MAME** as the emulator, then choose **ENABLE MOUSE** > **ENABLED**.