

ControllerRemap for Mame

© 2011 *drventure Enterprises*

Contents

Quick and Dirty	2
Introduction	6
ControllerRemap to the Rescue.....	8
How Does it Work?	9
Setting up a Controller File	10
What's in a Controller File?.....	11
The Details	12
Listing Controllers	13
Determining Your Controller Mappings.....	15
What about Dual Controllers with the Same Name	16
Controller Aliases	17
Aliases and Instances	18
A few final notes about aliases	18
“Begins With” Controller Matching	19
Prerequisites	20
Executables, DLLs	21
Other files.....	21
Version Notes.....	22
Version 0.0.7	23
Version 0.0.6	23

Quick and Dirty

This section is intended to get you started quickly.

There are a few prerequisites though:

- An installation of MAME
- Ability to use a text editor (notepad will do, but NotePad++ works much better)
- Ability to copy files around
- A basic understanding of the format of an XML file and how to edit one without mangling it

This is a very bare-bones command line utility at this point. There's no "point and click" UI for any of this configuration. It's all text file editing.

With that out of the way, here's a stripped down example of a standard Mame controller CFG file, with the extra elements in place that let ControllerRemap work its magic.

Pay particular attention to the <controlleralias> and <controller> elements (the parts in **RED**). You'll need to edit and/or create new instances of those elements to define your own controllers.

```
<!-- Configuration Remap file
-->
<mameconfig version="10">
  <!-- #####

      Define controller aliases, This just makes the controller names
      a little easier to deal with for the rest of the file.

      #####
  -->
  <controlleralias>
    <id>HID#Vid_1241_Pid_1111#b_2eabd86_1_0000#</id>
    <alias>Trackball</alias>
  </controlleralias>
  <controlleralias>
    <id>Ultimarc Ultra-Stik Player 1</id>
    <alias>U360 Player1</alias>
  </controlleralias>
  <controlleralias>
    <id>WingMan Extreme Digital 3D</id>
    <alias>Flightstick</alias>
  </controlleralias>
  <controlleralias>
    <id>WingMan Extreme Digital 3D</id>
    <alias>Flightstick</alias>
  </controlleralias>
  <controlleralias>
    <id>HID#VID_061C_PID_AA00#7_35df86d5_0_0000#</id>
    <alias>Lightgun1</alias>
  </controlleralias>

  <!-- #####

      This is the System Default section
      It generally should be the FIRST system section in the cfg file

      #####
  -->
  <system name="default">
    <!-- put a controller element here to have it copied into ONLY this particular
      system element
    -->
    <controller id="U360 Player1">
      <input>
```

```

        <port type="P1_JOYSTICK_UP">
            <newseq type="standard">
                JOYCODE_YAXIS_UP_SWITCH
            </newseq>
        </port>
        <port type="P1_JOYSTICK_DOWN">
            <newseq type="standard">
                JOYCODE_YAXIS_DOWN_SWITCH
            </newseq>
        </port>
    </input>
</controller>
<controller id="Flightstick">
    <input>
        <port type="P1_SELECT">
            <newseq type="standard">
                JOYCODE_BUTTON_7
            </newseq>
        </port>
        <port type="START1">
            <newseq type="standard">
                JOYCODE_BUTTON8_SWITCH
            </newseq>
        </port>
    </input>
</controller>
<controller id="Lightgun1">
    <input>
        <port type="P1_LIGHTGUN_X">
            <newseq type="standard">
                GUNCODE_XAXIS
            </newseq>
        </port>
        <port type="P1_LIGHTGUN_Y">
            <newseq type="standard">
                GUNCODE_YAXIS
            </newseq>
        </port>
    </input>
</controller>

```

<!-- this keyboard section provides a way to specify keyboard input mappings that should ALSO be made for the specific port types. -->

```

<controller id="keyboard">
    <input>
        <port type="P1_JOYSTICK_UP">
            <newseq type="standard">
                KEYCODE_UP
            </newseq>
        </port>
        <port type="P1_JOYSTICK_DOWN">
            <newseq type="standard">
                KEYCODE_DOWN
            </newseq>
        </port>
        <port type="P1_JOYSTICK_LEFT">
            <newseq type="standard">
                KEYCODE_LEFT
            </newseq>
        </port>
        <port type="P1_JOYSTICK_RIGHT">
            <newseq type="standard">
                KEYCODE_RIGHT
            </newseq>
        </port>
    </input>
</controller>
<controller id="Trackball">
    <input>
        <port type="P1_TRACKBALL_X">

```

```

        <newseq type="standard">
            MOUSECODE_XAXIS
        </newseq>
    </port>
    <port type="P1_TRACKBALL_Y">
        <newseq type="standard">
            MOUSECODE_YAXIS
        </newseq>
    </port>
</input>
</controller>

<!-- #####

        ACTUAL INPUT MAPPING STARTS HERE
        DO NOT actually put anything in this section.
        It is completely cleared and regenerated!

        #####
-->
<input>
</system>

<!-- #####

        This is a GAME specific section
        Add additional <controller> sections here
        to define how those controllers should map for this
        particular game

        #####
-->
<system name="ribbit">
    <!-- This would be stupid to actually use, but as an example
        For this game ("Ribbit"), reverse the UP and DOWN directions
        on the joystick -->
    <controller id="U360 Player1">
        <input>
            <port type="P1_JOYSTICK_UP">
                <newseq type="standard">
                    JOYCODE_YAXIS_DOWN_SWITCH
                </newseq>
            </port>
            <port type="P1_JOYSTICK_DOWN">
                <newseq type="standard">
                    JOYCODE_YAXIS_UP_SWITCH
                </newseq>
            </port>
        </input>
    </controller>
    <!-- #####

        ACTUAL INPUT MAPPING STARTS HERE
        DO NOT actually put anything in this section.
        It is completely cleared and regenerated!

        #####
-->
<input></input>
</system>
</mameconfig>

```

Save the above CFG file into your MAME “ctrlr” folder, name it MyArcade.cfg.

Assuming you’ve already extracted the ControllerRemap.ZIP file, run:

```
ControllerRemap /list
```

This will print a list of the joystick and RawMouse controllers available in your system, their enumeration order (that Mame uses) and their unique IDs.

Print or copy this list for future reference. You'll need those ID values to update the above CFG file with your own Controller IDs as appropriate (in the <controlleralias> elements), then update and add additional <port> elements to configure how your controllers should map to mame inputs, just like how you normally map ports in Mame.

Once that's done, BACKUP your CFG file (just in case) and perform a remap by running:

```
ControllerRemap /Remap:c:\mame\ctrlr\MyArcade.cfg
```

Load the newly remapped MyArcade.CFG file back into your text editor and look for the new expanded <input> elements. They should contain all the appropriate <port> elements for your various controllers that are currently installed. If they don't, check your CFG file's XML format and try again.

Keep in mind that ControllerRemap will automatically CLEAR OUT anything and everything in the <input> elements, so don't put anything in them to begin with. And when you edit the CFG file, you won't need to bother with cleaning out already remapped entries. ControllerRemap will clean them out and start fresh.

Once you're satisfied with the resulting CFG file, try loading it in Mame:

```
Mame ribbit -ctrlr MyArcade.cfg
```

If there's anything wrong with the XML formatting, Mame won't load it. Test your controllers to make sure they function as expected.

Now, plug in any additional controllers you want to map. Run *ControllerRemap /list* again to determine the new controller IDs.

Finally, add new <controlleralias> and <controller> elements to the CFG file to define the new controllers, and test again.

Keep in mind that ControllerRemap ALWAYS combines controller port mapping with OR keywords. For instance, if you define a <controller> element for a Joystick and map the P1_JOYSTICK_UP port to JOYCODE_YAXIS_UP_SWITCH, and then create another <controller> element for Keyboard and map the P1_JOYSTICK_UP port to KEYCODE_UP, you should end up with the P1_JOYSTICK_UP port in the applicable <input> element defined as

```
JOYCODE_YAXIS_UP_SWITCH OR KEYCODE_UP
```

The key is to define ONLY the input sequences that are applicable for the particular controller you're mapping, and let ControllerRemap combine everything as appropriate depending on what is actually installed.

Introduction

Why? Well, if you've ever set up a MAME cabinet that has external USB ports, you may have noticed that, on occasion, when you plug other USB devices in, your Mame control configuration becomes completely messed up, particularly if you use USB joysticks like Ultimarc's u360.

The reason is because Mame references USB Game controller devices ONLY by their enumeration number. And USB devices are enumerated based on a hex "VendorID" that is assigned to the vendor on request. The numbers are more or less random, so, depending on what the vendorID is on that game controller you just plugged in, it might end up enumerating above or below the devices that you already have in your system.

As an example...

You've just built your great new Mame machine. You're using several USB devices in it, including a Wingman Flightstick and 2 U360's. The devices are enumerated by Windows like so:

- 1) Wingman stick
- 2) Ultimarc U360 Player 1
- 3) Ultimarc U360 Player 2

You spend a good chunk of time configuring Mame to use all the right controllers, keyboard buttons, etc and get it working perfectly.

Then, you decide it'd be nice to add a PS2-USB adapter and external ports for some PS2 devices (say, dance pads or D-pad type controllers).

You pick up an adapter off eBay, plug it into a spare USB port and away you go. UNTIL.... You run Mame and discover some of your device mappings don't work anymore. That's because that new PS2 adapter changes the enumeration of some of your existing devices

- 1) Wingman stick
- 2) PS2 USB Adapter
- 3) Ultimarc U360 Player 1
- 4) Ultimarc U360 Player 2

Fine, you curse under your breath, remap everything, and you're good again.

A few weeks later, you pick up some nice Xbox Controllers you'd like to hook up. No big deal, you run a couple extender cables to external ports on your cab, and connect them up to 2 more spare USB ports inside.

Connect up the new controllers and....scrambled again. All your Mame controllers you'd just reconfigured don't work yet again. And worse, the flight stick doesn't work, now, either.

Looking at the enumerated list of devices, you see this:

- 1) Microsoft XBOX Controller
- 2) Microsoft XBOX Controller
- 3) Wingman stick

- 4) PS2 USB Adapter
- 5) Ultimarc U360 Player 1
- 6) Ultimarc U360 Player 2

Then, you come to a sinking realization. Regardless of how you fix the controller mapping, since you won't typically leave the Xbox controllers plugged in all the time, you're going to be constantly remapping controllers if you want to use them.

Not a good scene.

ControllerRemap to the Rescue

ControllerRemap exists for precisely this reason. Ideally, the Mame team (or someone more well versed in the intricacies of the Mame input model than I) will rework the Input.c code to deal with USB Game controller devices (and multiple Raw Mouse devices) via their built in JoystickID number or some other unique identifier, which doesn't change regardless of whether other devices are added or removed. FoFix and many games make use of the JoystickID, and the excellent little utility JoystickIDs (http://www.wingmanteam.com/latest_software/gadgets.htm#JoyIDs_Utility;) allows you to easily set those IDs to anything you want.

But, for now, Mame doesn't use JoystickIDs.

That's where ControllerRemap comes in.

ControllerRemap is a little .net utility that essentially regenerates the input port mapping sections in a Mame controller CFG file dynamically to match the current installed state of your USB devices.

With ControllerRemap, you just set your front end up to run ControllerRemap right before running Mame, for example:

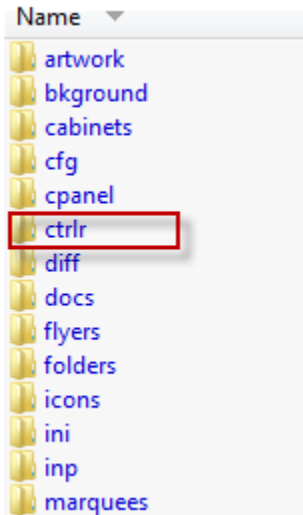
```
.\utilities\ControllerRemap.exe /remap:\games\emulators\mame\ctrlr\MyArcade.cfg
```

It alters the CFG file as necessary and when Mame starts, everything's gold.

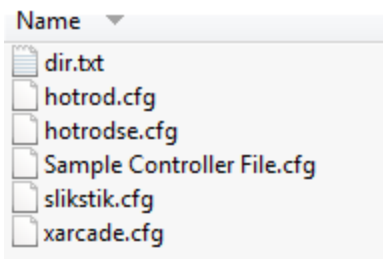
One Caveat: Since it must run *before* Mame starts, and actually only alters the CFG file for Mame, if you insert or remove devices *while Mame is running*, things will still not work properly. However, most games appear to work this way, so this shouldn't be too big an issue.

How Does it Work?

ControllerRemap takes advantage of an often neglected feature of Mame called “Controller files”. In your Mame installation folder, they’ll be stored in `.\ctrlr`



In that folder, you’ll usually find a few poor, lonely CFG files, that haven’t been touched in ages.



That dir.txt file says it all....

```
Place customized controller *.cfg files here.  
See respective controllers' websites for Mame  
controller \ctrlr\*.cfg files.
```

You can tell Mame to use one of these files when it starts by passing the “`-ctrlr filename.cfg`” command line argument, like this:

```
Mame romname -ctrlr MyArcade.cfg
```

The nice thing about these `ctrlr` files as opposed to the `CFG` files that live in the `.\cfg` folder, is that these controller files NEVER get rewritten or altered in any way by Mame. They can also contain comments, additional XML information, etc, and Mame just happily reads what it recognizes and ignores the rest.

`CFG` files in the `.\cfg` folder, on the other hand, will fail to load if they contain anything that Mame doesn’t recognize, and Mame rewrites those files, obliterating any data in them that Mame isn’t specifically aware of.

Setting up a Controller File

Sounds hard, it's actually quite easy.

- 1) Start Mame
- 2) Get into the admin menus and go through your normal controller configuration process
- 3) Quit Mame
- 4) Use Explorer to locate the \Mame\cfg folder
- 5) Find the DEFAULT.CFG file in that folder. It should have a modified date of just a few minutes ago (if you just remapped any controls).
- 6) Move the DEFAULT.CFG to the \Mame\ctrlr folder, and name it whatever you want, ie MyArcade.cfg
- 7) Done

Now, in order to actually make use of this new MyArcade.cfg file, you'll need to start Mame with an additional command line parameter as described previous, like this:

```
Mame romname -ctrlr MyArcade.cfg
```

If you need more information on Mame Controller files, google "Mame Controller Config files" or check out:

<http://mameworld.info/easyemu/mameguidenew/mameguide-controlini.htm>

http://wiki.arcadecontrols.com/wiki/Ctrlr_file

<http://www.scribd.com/doc/38150866/CTRLR>

What's in a Controller File?

Here's a shortened version of "Sample Controller File.cfg" that comes with Mame:

```
<?xml version="1.0"?>

<mameconfig version="10">
  <system name="default">
    <input>
      <port type="P1_JOYSTICK_UP">
        <newseq type="standard">
          KEYCODE_UP
        </newseq>
      </port>
      <port type="P1_JOYSTICK_DOWN">
        <newseq type="standard">
          KEYCODE_DOWN
        </newseq>
      </port>
      <port type="P1_JOYSTICK_LEFT">
        <newseq type="standard">
          KEYCODE_LEFT
        </newseq>
      </port>
      <port type="P1_JOYSTICK_RIGHT">
        <newseq type="standard">
          KEYCODE_RIGHT
        </newseq>
      </port>
    </input>
  </system>
</mameconfig>
```

Now, say we wanted to add a U360 to the inputs for Player-1-Joystick-up. The cfg file snippet might look like this:

```
<?xml version="1.0"?>

<mameconfig version="10">
  <system name="default">
    <input>
      <port type="P1_JOYSTICK_LEFT ">
        <newseq type="standard">
          JOYCODE_2_XAXIS_LEFT_SWITCH OR KEYCODE_LEFT
        </newseq>
      </port>
    </input>
  </system>
</mameconfig>
```

Note that JOYCODE_2_XAXIS_LEFT_SWITCH refers to moving the #2 joystick in the system (which happens to be a U360 right now), in the left direction. I've also configured it to accept using the LEFT ARROW KEY on the keyboard for player-one-joystick-left as well.

Easy enough.

Unfortunately, this is where the problem is. That JOYCODE_2_XAXIS_LEFT_SWITCH refers specifically to the joystick that enumerates as the second joystick in the system. And that can change easily when you add or remove devices.

The Details

ControllerRemap basically reads new <controller> elements in the CFG file and rebuilds the <input> elements by combining all the input element definitions for all the controllers that it can actually find currently installed in the system.

```
<?xml version="1.0"?>

<mameconfig version="10">
  <system name="default">
    <controller id="Ultimarc Ultra-Stik Player 1">
      <input>
        <port type="P1_JOYSTICK_LEFT">
          <newseq type="standard">
            JOYCODE_XAXIS_LEFT_SWITCH
          </newseq>
        </port>
      </input>
    </controller>
  </system>
</mameconfig>
```

Notice the <controller> element? That’s specific to ControllerRemap. Mame will simply ignore it.

The basic idea is as follows.

1. Create a <system> element for each system that you might want to generate a specific mapping for. You normally create a “default” system element (as shown above), but you might also create <system> elements for specific games, specific BIOS’s, or specific MAME drivers. The details of when and why you might want to create additional <system> elements for drivers, or bios’s is beyond the scope of this document but information is readily available online.
2. Within each <system> element, you would normally create an <input> element. However, ControllerRemap will be responsible for creating the <input> element for you, so you DO NOT create it yourself (and if you do, ControllerRemap simply deletes it, so don’t bother).
3. You DO, however, create a <controller> element within the <system> element, one for each controller that you want to provide mappings for. For instance, in the above snippet, I’ve created a controller element for “Ultimarc Ultra-Stik Player 1” in the “default” <system> element. But how do you know what id to use, you ask? See *Listing Controllers* elsewhere in this document.
4. Repeat this process for each <system> element you need, and each <controller> element that you want to map, even if it isn’t currently plugged in.

Now, you’ll need to define those <controller> elements you’ve created.

In each <controller> element, you create an <input> element, and any number of <port> elements, exactly as you would see in a normal <system> element. The only significant difference is that you DO NOT need to specify the joystick or mouse number in the port mapping.

For instance, in a normal mapping, you might specify:

```
<port type="P1_JOYSTICK_LEFT">
  <newseq type="standard">
    JOYCODE_5_XAXIS_LEFT_SWITCH
  </newseq>
</port>
```

Notice the `_5_` denoting to use the 5th joystick in the system.

In `<controller>` elements, all JOYCODE and MOUSECODE values *only* refer to the particular controller you're mapping, so there's no need for the number. You would leave out the 5, so you have something like this:

```
<port type="P1_JOYSTICK_LEFT">
  <newseq type="standard">
    JOYCODE_XAXIS_LEFT_SWITCH
  </newseq>
</port>
```

As mentioned above though, ControllerRemap will automatically take out the number if you leave it in, so feel free to do so if you'd prefer.

KeyCode, JoyCode, MouseCode and GunCode

By now, you may have noticed that the various controller files contain mentions of KEYCODE, MOUSECODE, JOYCODE and GUNCODE entries.

These are MAME controller keywords that identify the type of input device you're mapping.

However, they're all handled more or less the same, with a few variations.

- 1) KEYCODE: If I understand the Mame config files correctly, there is multikeyboard support in Mame, and I'm guessing, some way for Mame to distinguish between an UP ARROW keypress on keyboard 1, vs an UP ARROW on keyboard 2. However, I have not yet setup a multikeyboard Mame machine, so there is currently no support for dynamically remapping multiple keyboards in ControllerRemap.
- 2) JOYCODE: All USB joysticks are "hot swappable", meaning you can unplug and replug them at any time, in any USB port connected to the PC. This is precisely the purpose of ControllerRemap.
- 3) MOUSECODE: Mice typically don't get unplugged and replugged in most Mame machines. But, for machines with swappable control panels, it's very easy to imagine needing to remap mice as they are disconnected and reconnected. NOTE: as far as I know, old style serial mice ARE NOT hot swappable. Stick with USB Mice.
- 4) GUNCODE: From what I can tell, most Mame compatible gun controllers essentially act like mice and they are enumerated the same way. Mame, however, has a different set of input codes (the GUNCODE_# codes) to represent gun inputs. ControllerRemap handles this the same as with MOUSECODE.

Listing Controllers

ControllerRemap supports a `/LIST` command line parameter to cause it to list (to the console window) all the joysticks and RawMouse (including most typical gun) devices currently attached to the system.

```
ControllerRemap /list
```

For joysticks, it lists the Enumeration number of the stick, the Name of the stick, and its JoystickID.

For RawMouse devices (and most gun devices), it lists each device with its enumeration number, and it's raw USB Name (unfortunately, RawMouse devices don't have anything like the JoystickID, and the raw USB device name is the only name available, but it's not a particularly user friendly name. For example:

```
HID#VID_046D_PID_C012#8_248b2298_0_0000#
```

You use these names as the value of the "id" attribute of the <controller> elements to refer to those devices. For instance, if I wanted to specify the above RawMouse device, I might create a <controller> element like this:

```
<controller id=" HID#VID_046D_PID_C012#8_248b2298_0_0000#" name="TrackBall P1">
  <input>
    <port type="P1_TRACKBALL_X">
      <newseq type="standard">
        MOUSECODE_XAXIS
      </newseq>
    </port>
    <port type="P1_TRACKBALL_Y">
      <newseq type="standard">
        MOUSECODE_YAXIS
      </newseq>
    </port>
  </input>
</controller>
```

Notice that in the above snippet, I've identified the mouse I want to map via the "id" attribute, and given it a readable name with the "name" attribute. However, only the "id" attribute is used by ControllerRemap. I just used a "name" attribute to help identify which controller is which in the CFG file.

Determining Your Controller Mappings

The easiest way to determine what controller mappings you need to use is to do the following:

1. Connect the controller in question to the system.
2. Run "ControllerRemap.exe /list" to find out its enumeration number and its name. Copy this list, save it to a file and print it for reference later.
3. Start Mame, and map the controller via the configuration screens in Mame as you normally would.
4. Quit Mame
5. Open the "default.cfg" file from the .\cfg folder in Mame in your favorite text editor.
6. Find the applicable controller mapping entries, and copy them over to your custom mapping controller file stored in the .\ctrlr folder, placing them into the appropriate <system>/<controller> element
7. Add the "id" attribute to the <controller> element to identify which controller this element corresponds to
8. And finally, remove any JOYCODE_#, MOUSECODE_#, or GUNCODE_# numbers, since they aren't necessary and will likely just be confusing (note, however, that ControllerRemap will ignore any of those numbers so this step isn't technically necessary).

What about Dual Controllers with the Same Name

Many USB controllers are specifically designed to provide unique names when enumerated. The LEDWiz/GP and Ultimarc UltraSticks are perfect examples. However, there are plenty that don't.

```
Joystick Remap Utility - For Mame
(c) 2011 DrVenture Enterprises

The specified configuration folder doesn't exist:
'',
Defaulting to use the application folder:
'C:\Dev\Darin\ControllerRemap\bin\Release'

Joystick Device List (Devices are in enumeration order) ...

1. 'Controller (XUSB Gamepad)', ID=0
2. 'Controller (XUSB Gamepad)', ID=1

Mouse Device List (Devices are in enumeration order) ...

1. 'HID#VID_046D_PID_C012#8_248b2298_0_0000#'

End of list
```

Note that there are 2 XBOX USB Game controllers installed, but they both have the same name (although they do have different JoystickIDs).

In this situation, you can identify the specific joystick in 2 ways:

1. Specify just the JoystickID, ie "1", or "0", etc, for the name of the controller. When ControllerRemap sees a <controller> element with an id of a simple integer, it *assumes* you mean a joystick who's ID is that integer. Note that since RawMouse devices don't have IDs, this won't work for them. For example:

```
<controller id="2" name="Left side Xbox Stick">
  <input>
    <port type="P1_JOYSTICK_LEFT">
      <newseq type="standard">
        JOYCODE_XAXIS_LEFT_SWITCH
      </newseq>
    </port>
  </input>
</controller>
```

2. Specify an instance number by placing a ":" and the number at the end of the controller name. For instance, using the above example, you would refer to the 2'nd XUSB Game controller as "Controller (XUSB Gamepad):2".

```
<controller id="Controller (XUSB Gamepad):2" name="Left side Xbox Stick">
  <input>
    <port type="P1_JOYSTICK_LEFT">
      <newseq type="standard">
        JOYCODE_XAXIS_LEFT_SWITCH
      </newseq>
    </port>
  </input>
</controller>
```

If you use the second notation, ControllerRemap will look at the installed controllers in the system. If there are, indeed, 2 controllers with the given name, it will map the first to one instance and the second to the second instance, as you'd expect.

However, if there is only one controller actually in the system, it will be mapped to ALL controllers of that id, regardless of whether an instance is specified.

Controller Aliases

For some controllers, (for instance, Ultimarc U360's), their actual USB provided name is pretty clear and concise. But for others (for instance any RawMouse device), the USB name can be difficult to work with.

For that reason, ControllerRemap supports controller aliases.

Basically, a controller alias is just another name for a specific controller ID.

For instance, in one of the above examples, I mention a Trackball controller mapping entry, looking something like this.

```
<controller id="HID#VID_046D_PID_C012#8_248b2298_0_0000#" name="TrackBall P1">
  <input>
    <port type="P1_TRACKBALL_X">
      <newseq type="standard">
        MOUSECODE_XAXIS
      </newseq>
    </port>
    <port type="P1_TRACKBALL_Y">
      <newseq type="standard">
        MOUSECODE_YAXIS
      </newseq>
    </port>
  </input>
</controller>
```

Not too difficult to deal with, and you can always use the *name* attribute to better describe whatever the device “HID#VID_046D_PID_C012#8_248b2298_0_0000#” refers to.

Still, copying around that long hex-numeric ID string isn't particularly friendly.

Instead, you can create an alias for that specific device. Do this by creating a <controlleralias> element directly under the root <mameconfig> element. Within <controlleralias>, define two additional elements, <id> which contains the actual id of the device (as described above), and <alias> which contains the alias you want to use for the device. For example:

```
<mameconfig version="10">
  <controlleralias>
    <id>Ultimarc Ultra-Stik Player 1</id>
    <alias>U360 Player1</alias>
  </controlleralias>
  ...
```

Here, I've created a controlleralias for the device with id = “Ultimarc Ultra-Stik Player 1”. The alias is “U360 Player1”. Not a particularly stellar improvement here. But what about...

```
<mameconfig version="10">
  <controlleralias>
    <id>HID#VID_046D_PID_C012#8_248b2298_0_0000#</id>
    <alias>Trackball</alias>
  </controlleralias>
  ...
```

Ah! Now that's more like it! That terrible looking device ID is actually my Trackball.

Now, I can use that alias anywhere that I'd normally use the ID, for instance:

```

<controller id="Trackball">
  <input>
    <port type="P1_TRACKBALL_X">
      <newseq type="standard">
        MOUSECODE_XAXIS
      </newseq>
    </port>
    <port type="P1_TRACKBALL_Y">
      <newseq type="standard">
        MOUSECODE_YAXIS
      </newseq>
    </port>
  </input>
</controller>

```

Aliases and Instances

Aliases make use of instance numbers exactly like the normal IDs. So if you have two devices installed with EXACTLY the same ID, you can create an alias for the base device name:

```

<mameconfig version="10">
  <controlleralias>
    <id>XBOX Game Controller</id>
    <alias>GamePad</alias>
  </controlleralias>
  ...

```

In this case, we have a “GamePad” alias for an XBOX Game controller device. In your <controller> elements, you can refer to the id “XBOX Game Controller” or “GamePad” to reference this device. And you can also use instance numbers, such as:

```

<controller id="GamePad:2">
  <input>
    <port type="P1_UP">
      <newseq type="standard">
        JOYCODE_BUTTON_5
      </newseq>
    </port>
  </input>
</controller>

```

However, you can also create an alias for a specific instance of the device

```

<mameconfig version="10">
  <controlleralias>
    <id>XBOX Game Controller:2</id>
    <alias>GamePad P2</alias>
  </controlleralias>
  ...

```

In this case, the alias ONLY applies to instance 2 of an XBOX Game Controller device. If there is less than 2 of these devices connected, this alias won’t apply to anything and any mappings using it will be ignored.

A few final notes about aliases

- They are NOT case sensitive
- A controller can have any number of aliases (I haven’t found a particularly good use for that capability yet, but still)
- They don’t do dishes.

“Begins With” Controller Matching

As of version 0.0.6, ControllerRemap supports a “Begins with” match algorithm to accommodate situations where a controller is unplugged and plugged into a different USB port.

In that case, the controller ID might very well change slightly, since often, a number is appended to the end of the ID to signal what port the controller is connected to.

Here's an example:

This is a controller element with the FULL, ENTIRE controller ID specified:

```
<controlleralias>
  <id>HID#Vid_1241_Pid_1111#b_2eabd86_0_00000#</id>
  <alias>Top Trackball</alias>
</controlleralias>
```

Notice that "_0_00000#" at the end? That identifies which USB port the controller is plugged into. Since that port number can change if you move the device to a different port, ControllerRemap might not find it the next time.

Instead, you can now specify any amount of the ID, beginning at the start. So, I might set up the configuration like this instead:

```
<controlleralias>
  <id>HID#Vid_1241_Pid_1111#b_2eabd86</id>
  <alias>Top Trackball</alias>
</controlleralias>
```

When ControllerRemap sees the shortened version and attempts to match it against the complete ID of the installed device, it won't match.

Once it's searched all devices in that manner, it goes back and checks each device ID to see if it *begins with* the specified ID. If it finds a match this way, the device is matched up just like before.

Technically, this does mean you could specify something silly like:

```
<controlleralias>
  <id>HID#Vid_12
  <alias>Top Trackball</alias>
</controlleralias>
```

In this case, the specified ID is likely too short to uniquely identify a single device, so your result will probably not be what you want.

Just use as much of the device ID as what looks to uniquely identify the device and you should be good.

And finally, the example above specifies the ID as part of an alias, but you can use a partial controller ID anywhere an ID is expected in the configuration file.

Command Line Reference

/h, /?, /help	Displays a list of applicable commands to the screen
/list	list out all joystick and mouse devices to the screen
/save	generate a ControllerRemap.map file. This file can then be used for testing purposes to define a virtual set of controllers available to remapping purposes.
/testmap	indicates that the ControllerRemap should read in the contents of the ControllerRemap.map file (in the same folder as the ControllerRemap.exe) and use it as the definition of what controllers exist on the machine for remapping purposes (instead of reading the list of actual controllers on the current machine). Mainly for testing uses.
/testmap:name of map file	same purpose as /testmap above, but you can specify a specific MAP file.
/test	instead of updating the specified CFG file with the remapping info, ControllerRemap will create a new file of the same name, but ending in ".TEST". Mainly used for testing purposes.
/mame:name of mame.exe	specifies where to find and launch the mame executable. Then, launches mame when remapping is complete. Use this option to automatically start mame after remapping it's configuration.
/mame	same as above, but assumes that mame.exe is in the same path as ControllerRemap.exe. If it's not, ControllerRemap will not launch mame but will otherwise update the cfg file with remapping information.

Prerequisites

Executables, Dlls

Microsoft .net Framework 3.5

Microsoft.DirectX.DirectInput.dll

Microsoft.DirectX.dll

Other files

A Mame CFG file, usually located in the Mame ctrlr folder. This file must be read/writable, because it is altered when you run ControllerRemap.

It's also a good idea to keep a backup of your CFG file, just in case.

Version Release Notes

Version 0.0.11

Chipywiny reported a problem with the alias feature when devices were no longer actually present in the machine. In that case, ControllerRemap is supposed to simply ignore the device and not map any of the inputs to it. That wasn't happening.

This release corrects that problem.

Version 0.0.10

MESS supports KEYPAD type controllers for emulation of old console type games, and ControllerRemap wasn't taking that into consideration when generating its output file.

So, for instance, setting up a KEYPAD controller in your CTRLR file

```
<controller id="Keypad1">
  <input>
    <port tag=":SAC_KEYPAD1" type="KEYPAD" mask="2" defvalue="2">
      <newseq type="standard">
        JOYCODE_1_BUTTON9
      </newseq>
    </port>

    <port tag=":SAC_KEYPAD2" type="KEYPAD" mask="1" defvalue="1">
      <newseq type="standard">
        JOYCODE_2_BUTTON1
      </newseq>
    </port>
  </input>
</controller>
```

Would cause you to end up with an mapped file that looked something like this

```
<port type="KEYPAD" tag=":SAC_KEYPAD1" mask="1" defvalue="1">
  <newseq type="standard">
    JOYCODE_1_BUTTON19
  OR
  . . .
  OR
    JOYCODE_1_BUTTON20
  OR
    JOYCODE_1_BUTTON18
  OR
    JOYCODE_2_BUTTON19
  OR
    JOYCODE_2_BUTTON9
  OR
  . . .
```

Note that all end buttons end up lumped into a single PORT entry.

This is because ControllerRemap didn't consider the MASK and TAG attributes when determining the uniqueness of a port element for a KEYPAD type port.

It now does.

Many thanks to nick3092 for pointing this out.

Version 0.0.9

For controllers with ports that contain multiple “newseq” entries (a good example being a steering wheel with INCREMENT and DECREMENT newseq elements). Before, ControllerRemap would just “loose” those elements.

It now enumerates all of them and remaps them appropriately.

Further, additional attributes on the PORT element and each NEWSEQ element weren’t being copied over to the new mapped entry. This is fixed now, too.

Here’s an example (thanks to ArcadeBliss for pointing this out):

```
<controller id="Logitech MOMO Racing USB">
  <input>
    <port type="P1_PADDLE" mask="255" defvalue="128">
      <newseq type="standard">
        JOYCODE_1_XAXIS
      </newseq>
    </port>
    <port type="P1_PEDAL" mask="255" defvalue="0" reverse="yes">
      <newseq type="decrement">
        JOYCODE_ZAXIS_POS_SWITCH
      </newseq>
      <newseq type="increment">
        JOYCODE_ZAXIS_POS_SWITCH
      </newseq>
    </port>
  </input>
</controller>
```

Notice the newseq elements (in blue) and the additional port attributes (in red). These should all be copied to the newly mapped entry in the resulting output CFG file.

Version 0.0.8

The /SAVE command line option didn’t properly save MAP files when used.

Version 0.0.7

Added support for GUNCODE_# input types.

Thanks to Damonator for pointing this one out!

Version 0.0.6

First version released for public consumption.