Joystick Mapper for Linux – Configuration

Alexandre Hardy

August 20, 2016

Introduction

This document describes configuration files for the joystick mapper for Linux. The joystick mapper allows multiple joystick devices to be combined into one or more joystick devices. Joystick events can also be mapped to mouse events or keyboards events. The system is well suited to applications that have limited configurability with regard to joysticks. The joystick mapper kernel drivers must be loaded before any joystick drivers to ensure a consistent and sensible mapping.

Configuration file format

The configuration file consists of several lines. Each line is a mapping from some joystick event to another event. Blank lines are ignored. Comments are indicated by #. The software ignores everything from a # until the end of a line.
Each line begins with a keyword describing the kind of mapping that will be performed. A number of key-value pairs follows the keyword to describe the mapping.
All axes and button values begin at 0. That is the first button is button 0, the second button is button 2 etc.

1 Mapping directives

1.1 Common keys

- **id** – Identifier of joystick for which the mapping is specified. This identifier corresponds to the mapper device allocated to this joystick (see `/proc/bus/input/devices`).
- **vendor** – Vendor identification of the device for which the mapping is specified (see `/proc/bus/input/devices`).
- **product** – Product identification of the device for which the mapping is specified (see `/proc/bus/input/devices`).

Values are decimal/hexadecimal numbers, predefined keywords or strings. Strings are indicated by placing the value in double quotation marks, eg. "string".

1.2 `axis`

The `axis` mapping directive maps a joystick axis event to another event. The allowed keys are

- **id**
- **vendor**
- **product**
- **src** – This key specifies which axis of the joystick is to be mapped.
- **target** – This key specifies which device event will be triggered by an axis event on the specified axis. Valid options are `mouse`, `joyaxis`, `joybtn` or `kbd`.
- **device** – If the target event is a joystick, then the joystick number can be specified with the `device` key. The joystick number corresponds to 0 if js0 is selected, 1 if js1 is selected etc.
- **axis** – If the target has an axis, then the `axis` key specifies which axis the event must be mapped to.
- **plus**
- **minus** – If the target has a button, then the `plus` (`minus`) keys indicate which buttons/keys must be triggered if the axis is moved in a positive (negative) direction.
- **flags** – The `invert` flag can be used to reverse the sense of the axis. i.e. positive becomes negative and negative becomes positive. The `binary` flag can be used to trigger events only when the axis is moved completely in the other direction. The `trinary` flag is similar to the binary flag but also triggers events when the axis is centered (the 0 input value resets state).
- **min** – The reported minimum value on the axis. Used for determining trigger levels.
- **max** – The reported maximum value on the axis. Used for determining trigger levels.
- **deadzone** – The deadzone for the axis.

Either the `id` or the `vendor` and `product` keys must be specified to identify the joystick. The `kbd` target only supports button presses.

If `min` and `max` are specified, then the input will be remapped into the range $[-32767, 32767]$ using the formula

$$o = \frac{65536 \times (x - min)}{max - min} - 32767$$

where x is the input and o is the reported axis position. The value will be clamped if necessary. Note that if $min > max$ then the axis reporting is switched around by this formula.

If a deadzone is specified, then the deadzone is enforced **after** mapping using the formula above so that any input x such that $-deadzone \leq x \leq deadzone$ will be reported as 0.

Example.

```
axis vendor=0x068e product=0x00f1 src=0 target=mouse axis=0
```

This mapping maps from a CH Products PRO Throttle axis 0 to the x-axis of the mouse.

1.3 `button`

The `button` mapping directive maps a joystick button event to another event. The allowed keys are

- **id**
- **vendor**
- **product**
- **src** – This key specifies which button of the joystick is to be mapped.
- **target** – This key specifies which device event will be triggered by an axis event on the specified axis. Valid options are `mouse`, `joyaxis`, `joybtn` or `kbd`.
- **device** – If the target event is a joystick, then the joystick number can be specified with the `device` key. The joystick number corresponds to 0 if js0 is selected, 1 if js1 is selected etc.
- **button** – If the target has a button, then the `button` key specifies which button the event must be mapped to.
- **axis** – If the target has an axis, then the `axis` key specifies which axis the event must be mapped to. The button will trigger a move in the positive direction on the axis.

- **flags** – The `invert` flag can be used to reverse the sense of the axis. i.e. the button will trigger a move in the negative direction on the axis.

  The `autorelease` flag specifies that the pressed button or key must be released immediately after pressing, even if the joystick button is held in.

  The `release` flag specifies that the mapping only applies to a release of the joystick button.

  The `press` flag specifies that the mapping only applies to a button being pressed, and not to a button being released.

  If neither `press` nor `release` is specified then two mapping entries are created, one for press and one for release so that pressing the button on the joystick will press the target button which will only be released when the joystick button is released.

  The `shift` key specifies that this mapping is only applied when in a shifted state (see the `shift` statement below).

Either the `id` or the `vendor` and `product` keys must be specified to identify the joystick. The `kbd` target only supports button presses.
If the target is `kbd`, then a string can be provided which specifies a series of keyboard events that must be executed. **Example.**

```
button vendor=0x068e product=0x00f4 src=2 target=kbd button="b REL
b a REL a n REL n g REL g leftshift 1 REL 1 REL leftshift"
```

This mapping maps from a CH Products Combatstick, button 2, to the key sequence *bang!* (see the file `keys.map` for recognized keys). `REL` indicates that the next key must be released.

1.4 `shift`

The joystick mapper supports a *shifted* mode in which the operation of buttons and axes are modified. A single button on a joystick can be designated the shift button. If this button is pressed then the joystick is in *shifted* mode. Allowed keys are

- **id**
- **vendor**
- **product**
- **src** – This key specifies which button of the joystick is to be the shift button.

Example.

```
shift vendor=0x068e product=0x00f1 src=5
```

This mapping designates button 5 on the CH Products PRO Throttle as the shift button.

1.5 `code`

A program can be specified that is executed at regular intervals and can generate joystick and keyboard events. The joystick events generated by the program can be remapped through the joystick mapper device. The code joystick is identified by vendor 0x00ff and produce 0x0000. The syntax of programs is described in a separate document.
Example.

```
code "myprogram"
```

The joystick mapper will read and attempt to compile the program in the file `myprogram`. If all mapping statements are correct, then the program will be communicated to the kernel driver along with the other mappings.

1.6 `script`

The program refers to existing joysticks in the system. To prevent the program from failing to function correctly due to changes in the joystick allocation (allocation to js0, js1, mapper0, mapper1 etc.), for example if the joysticks have been unplugged and reinserted, joysticks can be numbered based on vendor and product identifiers. Allowed keys are

- **id**
- **vendor**
- **product**
- **device** – Specify the joystick number in the program file.

Example.

```
script vendor=0x068e product=0x00f4 device=0
```

This mapping specifies that the CH Products Combatstick will be referred to as joystick 0 within the program specified by the `code` statement.