# rotary-encoder - a generic driver for GPIO connected devices

**Author:** Daniel Mack <daniel@caiaq.de>, Feb 2009

## Function

Rotary encoders are devices which are connected to the CPU or other peripherals with two wires. The outputs are phase-shifted by 90 degrees and by triggering on falling and rising edges, the turn direction can be determined.

Some encoders have both outputs low in stable states, others also have a stable state with both outputs high (half-period mode) and some have a stable state in all steps (quarter-period mode).

The phase diagram of these two outputs look like this:

```
                 _____       _____       _____
                |     |     |     |     |     |
Channel A  ____|     |_____|     |_____|     |____

               :  :  :  :  :  :  :  :  :  :  :  :
          __       _____       _____       _____
            |     |     |     |     |     |     |
Channel B   |_____|     |_____|     |_____|     |__

               :  :  :  :  :  :  :  :  :  :  :  :
Event          a  b  c  d  a  b  c  d  a  b  c  d

              |<-------->|
                one step

              |<-->|
                one step (half-period mode)

              |<>|
                one step (quarter-period mode)
```

For more information, please see
https://en.wikipedia.org/wiki/Rotary_encoder

## Events / state machine

In half-period mode, state a) and c) above are used to determine the rotational direction based on the last stable state. Events are reported in states b) and d) given that the new stable state is different from the last (i.e. the rotation was not reversed half-way).

Otherwise, the following apply:

a. Rising edge on channel A, channel B in low state
   This state is used to recognize a clockwise turn
b. Rising edge on channel B, channel A in high state
   When entering this state, the encoder is put into 'armed' state, meaning that there it has seen half the way of a one-step transition.
c. Falling edge on channel A, channel B in high state
   This state is used to recognize a counter-clockwise turn
d. Falling edge on channel B, channel A in low state
   Parking position. If the encoder enters this state, a full transition should have happened, unless it flipped back on half the way. The 'armed' state tells us about that.

## Platform requirements

As there is no hardware dependent call in this driver, the platform it is used with must support gpiolib. Another requirement is that IRQs must be able to fire on both edges.

## Board integration

To use this driver in your system, register a platform_device with the name 'rotary-encoder' and associate the IRQs and some specific platform data with it. Because the driver uses generic device properties, this can be done either via device tree, ACPI, or using static board files, like in example below:

```
/* board support file example */

#include <linux/input.h>
#include <linux/gpio/machine.h>
#include <linux/property.h>

#define GPIO_ROTARY_A 1
#define GPIO_ROTARY_B 2

static struct gpiod_lookup_table rotary_encoder_gpios = {
        .dev_id = "rotary-encoder.0",
        .table = {
                GPIO_LOOKUP_IDX("gpio-0",
                                GPIO_ROTARY_A, NULL, 0, GPIO_ACTIVE_LOW),
                GPIO_LOOKUP_IDX("gpio-0",
                                GPIO_ROTARY_B, NULL, 1, GPIO_ACTIVE_HIGH),
                { },
        },
};

static const struct property_entry rotary_encoder_properties[] = {
        PROPERTY_ENTRY_U32("rotary-encoder,steps-per-period", 24),
        PROPERTY_ENTRY_U32("linux,axis",                      ABS_X),
        PROPERTY_ENTRY_U32("rotary-encoder,relative_axis",    0),
        { },
};

static const struct software_node rotary_encoder_node = {
        .properties = rotary_encoder_properties,
};

static struct platform_device rotary_encoder_device = {
        .name           = "rotary-encoder",
        .id             = 0,
};

...

gpiod_add_lookup_table(&rotary_encoder_gpios);
device_add_software_node(&rotary_encoder_device.dev, &rotary_encoder_node);
platform_device_register(&rotary_encoder_device);

...
```

Please consult device tree binding documentation to see all properties supported by the driver.