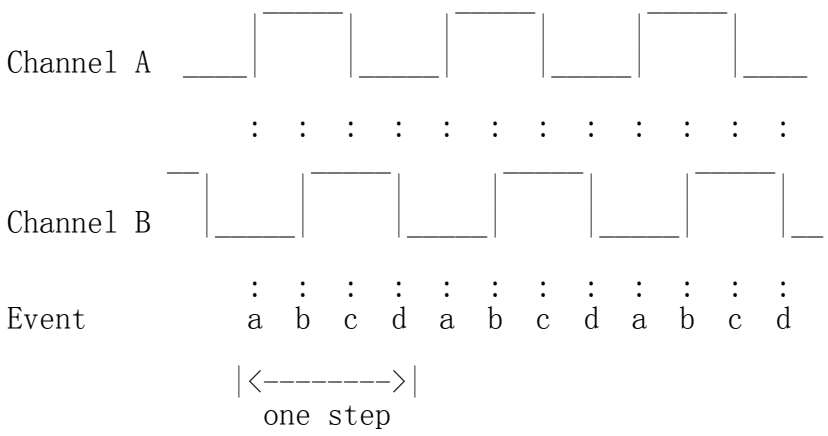
rotary-encoder - a generic driver for GPIO connected devices
Daniel Mack <daniel@caiaq.de>, Feb 2009

0. 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.

The phase diagram of these two outputs look like this:



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

1. Events / state machine

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 happend, unless it flipped back on half the way. The 'armed' state tells us about that.

2. 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.

3. 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.

struct rotary_encoder_platform_data is declared in
include/linux/rotary-encoder.h and needs to be filled with the number of
steps the encoder has and can carry information about externally inverted
signals (because of an inverting buffer or other reasons). The encoder
can be set up to deliver input information as either an absolute or relative
axes. For relative axes the input event returns +/-1 for each step. For
absolute axes the position of the encoder can either roll over between zero
and the number of steps or will clamp at the maximum and zero depending on
the configuration.

Because GPIO to IRQ mapping is platform specific, this information must
be given in separately to the driver. See the example below.

---------<snip>---------

```
/* board support file example */

#include <linux/input.h>
#include <linux/rotary_encoder.h>

#define GPIO_ROTARY_A 1
#define GPIO_ROTARY_B 2

static struct rotary_encoder_platform_data my_rotary_encoder_info = {
        .steps          = 24,
        .axis           = ABS_X,
        .relative_axis  = false,
        .rollover       = false,
        .gpio_a         = GPIO_ROTARY_A,
        .gpio_b         = GPIO_ROTARY_B,
        .inverted_a     = 0,
        .inverted_b     = 0,
};

static struct platform_device rotary_encoder_device = {
        .name           = "rotary-encoder",
        .id             = 0,
        .dev            = {
                .platform_data = &my_rotary_encoder_info,
        }
};
```