jamon / **pi-pico-pio-quadrature-encoder** (Public)

Raspberry Pi Pico PIO Quadrature Encoder

⚖ View license

☆ **44** stars   � **7** forks   Branches   Tags ⟋ Activity

| ☆ Star ▾ | 🔔 Notifications |

<> **Code**   ⊙ Issues 3   ⇅ Pull requests   ▷ Actions   ⊞ Projects   ⚠ Security   ∿ Insights

main ▾   1 Branch   0 Tags | Go to file | Go to file   Code ⋯

ardlc-1212 Update README.md •••		7 months ago ••• ⟲
📁 examples	Delete main.c	7 months ago
📁 include	initial commit	2 years ago
📁 lib	initial commit	2 years ago
📁 python	Add files via upload	7 months ago
📁 src	Update quadrature.pio	7 months ago
📁 test	initial commit	2 years ago
📄 .gitignore	initial commit	2 years ago
📄 LICENSE	Update LICENSE	7 months ago
📄 README.md	Update README.md	7 months ago
📄 library.json	Add files via upload	7 months ago
📄 platformio.ini	initial commit	2 years ago

📖 **README**   ⚖ License   ☰

Raspberry Pi Pico PIO Quadrature Encoder

Works with C/C++ via pico-sdk, arduino, and micropython. Examples are provided for all but arduino.

Pass A and B pins to PIO program and it will keeps track of quadrature position in state machine's "x" register.

To read the current value, force exec an "in" command from "x" register, then read the X value (see example in main.c).

The value of x is an absolute value of the encoder (initialized to 0 on startup). You can do relative measurements for a menu by capturing the value when the menu starts, but it doesn't support limits currently, so probably the best workaround would be to take the absolute value of the encoder modulo the length of your menu to select determine the position.

How to use

- Include quadrature.pio in your project

- Include headers

```
#include "hardware/pio.h"
#include "quadrature.pio.h"
```

- Define your QUADRATURE Encoder pins

```
#define QUADRATURE_A_PIN 2
#define QUADRATURE_B_PIN 3
```

- Initialize a PIO state machine

```
PIO pio = pio0;
uint offset = pio_add_program(pio, &quadratureA_program);
uint sm = pio_claim_unused_sm(pio, true);
quadratureA_program_init(pio, sm, offset, QUADRATURE_A_PIN, QUADRATURE_B_PIN);
```

- Read the current value of the encoder from the state machine

```
pio_sm_exec_wait_blocking(pio, sm, pio_encode_in(pio_x, 32));
uint x = pio_sm_get_blocking(pio, sm);
```

Zeroing/resetting position

```
pio_sm_exec(pio, sm, pio_encode_set(pio_x, 0));
```

Switch/Button

Not really applicable for this kind of library, but I'm going to drop this here in case it's useful. If you want to handle button presses via IRQ:

```
#define QUADRATURE_SW_PIN 4
PIO pio = pio0;
uint offset, sm;
void quadrature_sw_callback(uint gpio, uint32_t events) {
    if(gpio == QUADRATURE_SW_PIN) pio_sm_exec(pio, sm, pio_encode_set(pio_x, 0));
}
int main() {
    stdio_init_all();

    offset = pio_add_program(pio, &quadratureA_program);
    sm = pio_claim_unused_sm(pio, true);

    quadratureA_program_init(pio, sm, offset, QUADRATURE_A_PIN, QUADRATURE_B_PIN);

    gpio_set_irq_enabled_with_callback(QUADRATURE_SW_PIN, GPIO_IRQ_EDGE_FALL, true, &quadrature_sw_callback);

    // rest of your logic
}
```

Python Example

See python/quadrature.py.

Example

See the examples folder to see both 2 sub-state and 4 sub-state versions that write the value every 1 second.

License

Licensed under the MIT License

Releases

No releases published

Packages

No packages published

Contributors   3

ardlc-1212

hippo5329 Thomas Chou

jamon

Languages

● **C** 64.8%   ● **Python** 35.2%