**CODIFICADOR DE CUADRATURA RASPBERRY PI PICO PIO**

Funciona con C/C++ a través de pico-sdk, arduino y micropython. Se proporcionan ejemplos para todos menos para arduino.

Pase los pines A y B al programa PIO y realizará un seguimiento de la posición de cuadratura en el registro "x" de la máquina de estado.

Para leer el valor actual, fuerce la ejecución de un comando "in" desde el registro "x", luego lea el valor X (vea el ejemplo en main.c).

El valor de x es un valor absoluto del codificador (inicializado en 0 al inicio). Puede realizar mediciones relativas para un menú capturando el valor cuando se inicia el menú, pero actualmente no admite límites, por lo que probablemente la mejor solución sería tomar el valor absoluto del módulo del codificador de la longitud de su menú para seleccionar determinar la posición.

**Cómo utilizar**

* Incluye quadrature.pio en tu proyecto
* Incluir encabezados

**En Lenguaje C:**

#include "hardware/pio.h"

#include "quadrature.pio.h"

**Defina los pines del codificador de CUADRATURA:**

#define QUADRATURE\_A\_PIN 2

#define QUADRATURE\_B\_PIN 3

**Inicializar una máquina de estado PIO:**

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);

**Leer el valor actual del codificador de la máquina de estados:**

pio\_sm\_exec\_wait\_blocking(pio, sm, pio\_encode\_in(pio\_x, 32));

uint x = pio\_sm\_get\_blocking(pio, sm);

**Posición de puesta a cero/reposición:**

pio\_sm\_exec(pio, sm, pio\_encode\_set(pio\_x, 0));

**Botón/Interruptor**

Realmente no es aplicable para este tipo de biblioteca, pero lo dejaré aquí en caso de que sea útil. Si desea manejar las pulsaciones de botones a través de 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

}

**Ejemplo de Python**

# SPDX-FileCopyrightText: 2022 Jamon Terrell <github@jamonterrell.com>

# SPDX-License-Identifier: MIT

from rp2 import PIO, StateMachine, asm\_pio

from machine import Pin

import utime

@asm\_pio(autopush=True, push\_thresh=32)

def encoder():

label("start")

wait(0, pin, 0) # Wait for CLK to go low

jmp(pin, "WAIT\_HIGH") # if Data is low

mov(x, invert(x)) # Increment X

jmp(x\_dec, "nop1")

label("nop1")

mov(x, invert(x))

label("WAIT\_HIGH") # else

jmp(x\_dec, "nop2") # Decrement X

label("nop2")

wait(1, pin, 0) # Wait for CLK to go high

jmp(pin, "WAIT\_LOW") # if Data is low

jmp(x\_dec, "nop3") # Decrement X

label("nop3")

label("WAIT\_LOW") # else

mov(x, invert(x)) # Increment X

jmp(x\_dec, "nop4")

label("nop4")

mov(x, invert(x))

wrap()

sm1 = StateMachine(1, encoder, freq=125\_000\_000, in\_base=Pin(3), jmp\_pin=Pin(2))

sm1.active(1)

while(True):

utime.sleep(1)

sm1.exec("in\_(x, 32)")

x = sm1.get()

print(x)