# ISR to task communication in FreeRTOS

## Part 1 – Activity indicator with a binary semaphore

Write a program that creates two tasks: one for reading characters from the serial port and the other for indicating received characters on the serial port. Use a **binary semaphore** to notify serial port activity to the indicator task. Note that a single blink sequence (200 ms) takes much longer than transmission time of one character (0.1 ms) and only one blink after last character is allowed.

### Task 1

Task reads characters from debug serial port using getchar_timeout_us and echoes them back to the serial port. When a character is received the task sends an indication (= gives the binary semaphore) to blinker task.

### Task 2

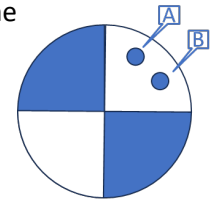
This task blinks the led once (100 ms on, 100 ms off) when it receives activity indication (= takes the binary semaphore).

## Part 2 - GPIO interrupts and queue

RP2040 has two GPIO banks that can trigger interrupts. Second GPIO bank is dedicated to external flash interface so user programs can use only GPIO bank 0. Each pin in the bank can be programmed to trigger an interrupt on HIGH/LOW level of rising or falling edge. It is possible to configure more than one trigger per pin, but the most common case is to configure only one type of trigger.

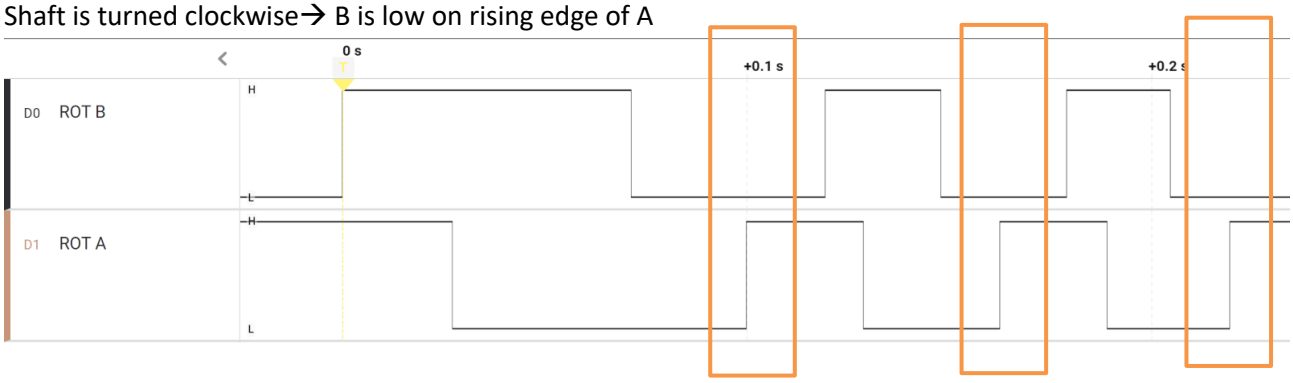
Study https://www.raspberrypi.com/documentation/pico-sdk/hardware.html#hardware_gpio. Pay attention to assigning callbacks to GPIO pins.

Rotary encoder has two outputs that change their state when the encoder is turned. The outputs are typically named A and B. The encoder is built so that only one of the outputs changes at a time when the shaft is turned. Depending on the rotation either A or B makes contact first. To decode direction of turn we can choose one of the signals as clock and monitor the change of the other on one edge of the clock.

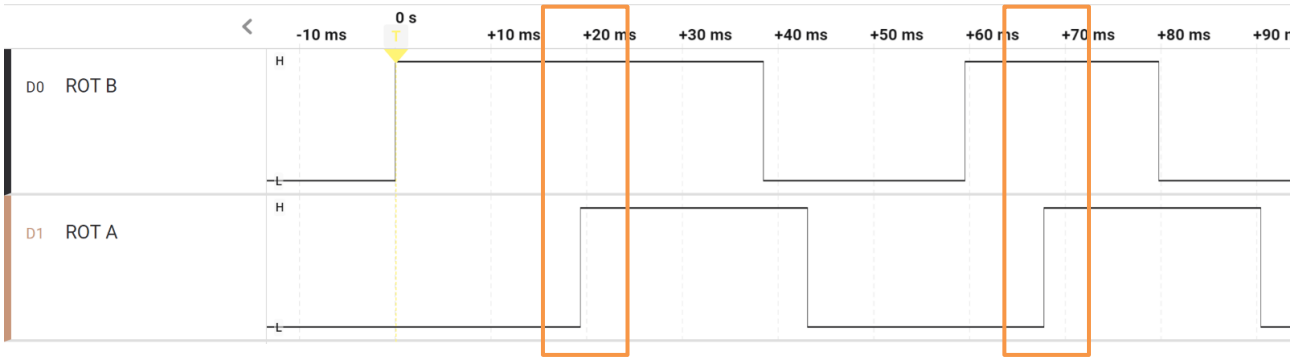


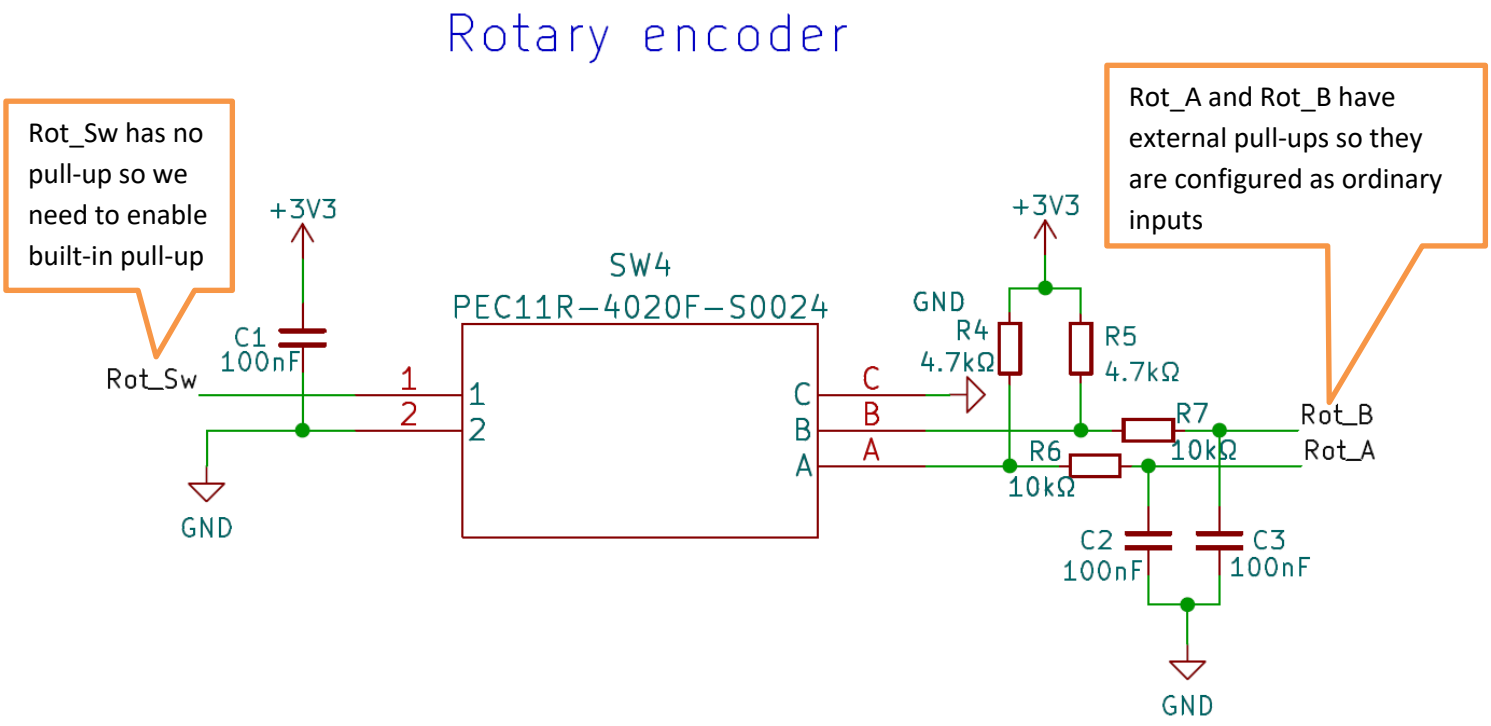

For example, if we choose A as the clock, we check the state of B on every rising edge of A. To ensure that we don’t miss any of the edges we’ll configure an interrupt on the rising edge of the pin A is wired to.

Shaft is turned clockwise→ B is low on rising edge of A




Shaft is turned counter-clockwise: B is high on rising edge of A




The rotary encoder is connected to three GPIO pins:

- Rot_A to 10 – configure as an input **without** pull-up/pull-down
- Rot_B to 11 – configure as an input **without** pull-up/pull-down

- Rot_Sw to 12 – configure as an input **with** pull-up

Implement a program for switching a LED on/off and changing the blinking frequency. The program should work as follows:

- Rot_Sw, the push button on the rotary encoder shaft is the on/off button. When button is pressed the state of LEDs is toggled. Program must require that button presses that are closer than 250 ms are ignored.
- Rotary encoder is used to control blinking frequency of the LED. Turning the knob clockwise increases frequency and turning counterclockwise reduces frequency. If the LED is in OFF state turning the knob has no effect. Minimum frequency is 2 Hz and maximum frequency is 200 Hz. When frequency is changed it must be printed
- When LED state is toggled to ON the program must use the frequency at which it was switched off.

You must use GPIO interrupts for detecting the encoder turns and button presses and send the button and encoder events to a queue.

All queues must be registered to queue registry.

Hint: create two tasks: one for receiving and filtering gpio events from the queue and other for blinking the LED.