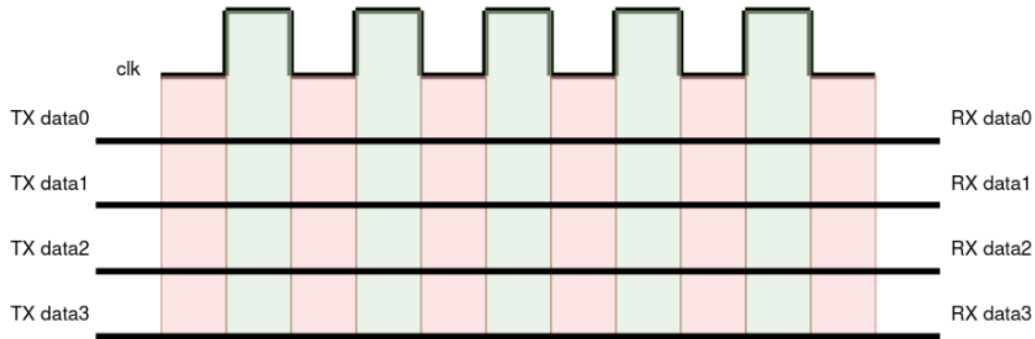


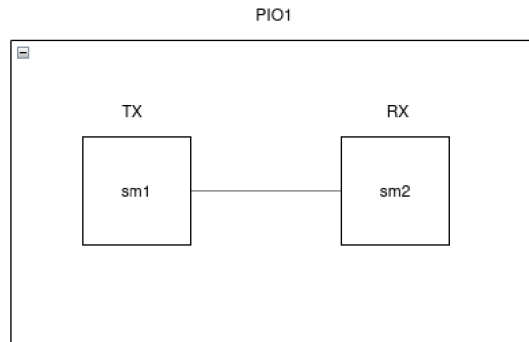
Projet personnel BUS COMMUNICATION Personnalisé

Dans mon projet, j'ai mis en place un protocole reposant sur la synchronisation entre une horloge et la transmission de données. À chaque front montant de l'horloge, les données stockées dans un registre temporaire (OSR) sont envoyées sur les pins de transmission. La lecture des données est effectuée uniquement lorsque l'horloge est à 1, assurant ainsi une synchronisation précise entre la transmission et la réception.

J'ai illustré les moments où il est possible de lire et d'écrire en les coloriant en vert, tandis que les périodes où aucune opération n'est autorisée sont indiquées en rouge.

Cependant, j'ai rencontré des difficultés liées à la configuration des pins sur ma carte, car les sorties et les entrées ne sont pas consécutives, ce qui a engendré des retards dans l'avancement de mon projet. Pour surmonter cela, j'ai décidé d'adopter une nouvelle approche en utilisant un simulateur en ligne, ce qui m'a permis de progresser plus rapidement.





```

import rp2
from rp2 import PIO
from machine import Pin
import utime

@rp2.asm_pio(out_shift_dir=PIO.SHIFT_RIGHT, out_init=(PIO.OUT_LOW,) * 4,
             sideset_init=PIO.OUT_LOW, autopull=True, pull_thresh=4)
def machine1():
    wrap_target()
    pull(block).side(0).delay(2)
    out(pins, 4).side(1)
    nop().side(0).delay(2)
    out(pins, 4).side(1)
    wrap()

@rp2.asm_pio(in_shift_dir=PIO.SHIFT_LEFT, autopush=True, push_thresh=4)
def machine2():
    wrap_target()
    wait(1, pins, 0)
    in_(pins, 4)
    push()
    wrap()

sm2 = rp2.StateMachine(1, machine2, freq=2000, in_base=Pin(16))
sm1 = rp2.StateMachine(0, machine1, freq=2000, sideset_base=Pin(0), out_base=Pin(1))

sm2.active(1)
sm1.active(0)

val = 0b0000001
sm1.put(val)
sm1.active(1)
  
```

```
print("Transmission d'erreur!")

while True:
    if sm2.rx_fifo() != 0:
        data = sm2.get() & 0xF
        print(f"Donnée reçue : {bin(data)} ({data})")
```

Ce projet n'est pas encore fini comme je le voudrais .