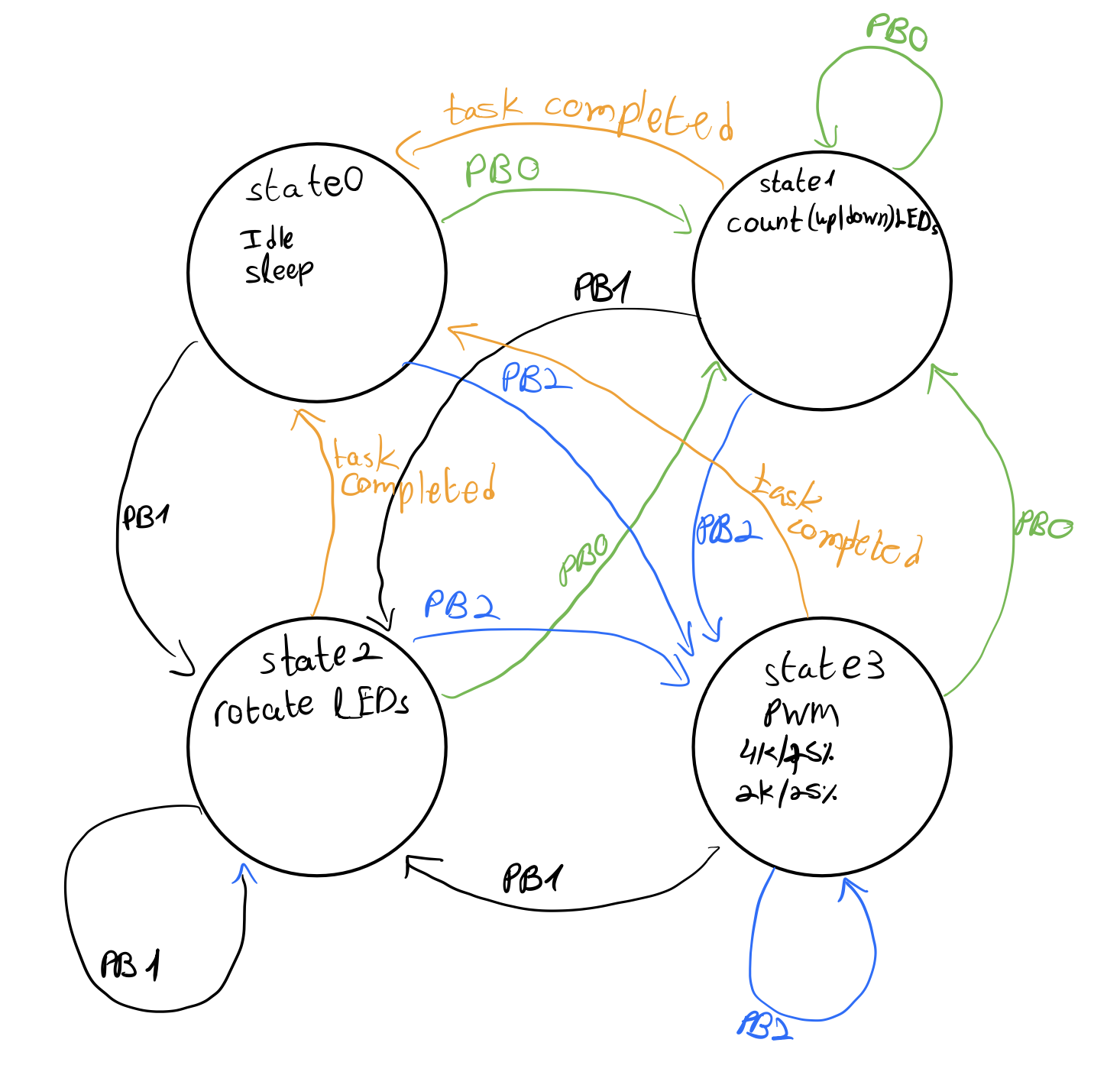
Preparation Report LAB1

DCS

Yarden Levy 206212276

Nachman Mimoun 321730558

# FSM diagram



Before writing the code, we first drew a state transition diagram for the different states required

## 1. explain the role of the registers:

Each port in the microcontroller (PORT1 to PORT10 for the series 4 kit) has dedicated registers that allow configuration and control of the port pins (input/output operations). Each pin in a port corresponds to a bit in these registers.

**PxDIR** determines the direction of the pin (input when the corresponding bit is set to 0 (**default**), and output if 1)

**PxSEL** selects whether the pin is used for GPIO or for a peripheral use ( 0 = GPIO (**default**), 1 = Peripheral module)

**PxIN** reads the logical value on the pin when it is configured as input (register value is corresponding to voltage level on pin)

**PxOUT** writes values to output pins when it is configured as output (register value is corresponding to voltage level on pin)

## 2. after RESET, what is the default state of the ports and why?

As mentioned above, the default value PxSEL=0 and PxDIR=0 defines all GPIO IO ports to behave as GPIO inputs.

## 3. steps to configure even-indexed pins as output and odd-indexed pins as Input:

P9SEL = 0  
P9DIR = 0x55 (8`b01010101)

## 4. to create a square wave with period of 1ms ,how many MCLK cycles are required for the ‘1’ in the wave?

Recall that MCLK is a 1MHz clock, meaning each clock cycle is apporx. 0.954us, thus we can calculate that it would require 500(ms) / 0.954 (us/clock) = **524 clock cycles** .

## 5. What is an interrupt

An interrupt is a signal that changes the normal sequential flow of the program execution to handle an event immediately.

After the ISR is executed, the program resumes execution from where it was interrupted (state register (SR) and program counter (PC) are stored in the stack until the end of the ISR)

It is required to handle hardware or external event that requires immediate attention, without polling continuously.

## 6. Explain the advantages of interrupt vs polling and when to combine them

* Interrupt allows the processor to perform other tasks and only respond when necessary.
* Polling wastes processor cycles by checking constantly for events.

The correct way to combine the two is to use interrupts for event detection and polling for checking multiple flags or conditions withing the ISRs

## 7. Explain the 3 types of interrupts and their uses

1. Non-maskable interrupt (NMI) – cannot be ignored, used for critical functions that cannot delay for ISR execution (such as manually pressing the RESET button)
2. Maskable interrupt (MI) – Generated by hardware peripherals or software and can be disabled \ enabled  
   the 3 types of MI are:
   1. External interrupts – triggered by hardware events
   2. Internal interrupts – triggered by specific instructions or software conditions
   3. Software interrupt – generated explicitly by software code

## 8. Explain the microcontroller operating modes and when to use them

There are two main approaches for operation:

* Infinite loop – if power consumption is not a concern
* Low power mode (LPM) – used to save energy.  
  different LPM levels disable various parts of the microcontroller (CPU , clocks etc..)  
  higher LPM level corresponds with lower energy consumption and lower device availability.

## 9. the steps to configure pin P2.0 to trigger an interrupt on falling edge

1. P2SEL[0] = 1 🡪 set as GPIO
2. P2DIR[0] = 1 🡪 set as input
3. P2IES[0] = 1 🡪 set interrupt on falling edge
4. P2IE[0] = 1 🡪 enable interrupts for P2.0 (local interrupt enable)

## 10. Explain the following programming paradigms

 **Event Driven Programming** - A programming approach where different parts of the system communicate by sending events (messages). The system reacts when an event occurs, allowing other tasks to run in the meantime.

 **Interrupt Driven Programming** - Execution is controlled by interrupts from hardware or software. When an interrupt occurs, the processor pauses the current task, runs a special function (ISR), then returns to normal operation.

 **Blocking Programming** - In this method, the code stops running (waits) until a certain action is completed. Simple to use, but inefficient if the processor could be doing other tasks meanwhile.

 **Non-Blocking Programming** - Here, the code continues to run without waiting for an action to finish. The result is handled later, allowing better use of processor time for multitasking.