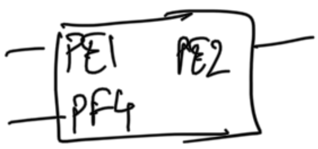
Things to go over:

1. Input Output ports
2. Clock frequency
3. Objective of the lab
4. Delay function basics
5. Stage 2: Implementing Duty cycle
6. Stage 3: Implementing PE1 button
7. Stage 4: Build Circuit
8. Stage 5: Breathing LED

1.



- PE2 is positive logic out
- PE1 is pos logic in
- PF4 is neg logic in

2.- normal clock freq is 16 MHz
- with Texas_Init is 80 MHz
- So, work with 80 MHz when calculating delay

3. Objective of the lab:
- Implement duty cycle operation
- Change duty cycle wrt button press
- Make circuit to implement button-led
- Implement breathing LED

2 Hz led flash

4. To calculate delay:
- with 800→R0: ≈ 0.05ms) x10
- Target delay >= 0.5 ms

8000 ≱ 0.5ms   >8000

$\boxed{\text{Delay 0.5ms}}$

~~10,000~~ ⟷ 8000

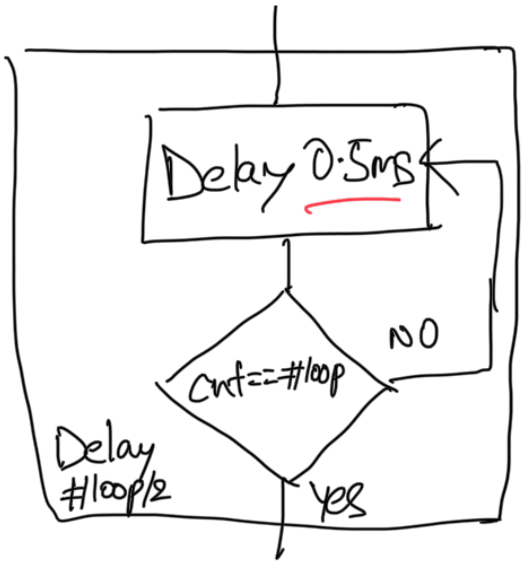
On the TM4C123 the default bus clock is 16 MHz ±1%. However, using TExaS_Init engages the Phase-Lock-loop (PLL) and runs the TM4C123 at 80 MHz. At 80 MHz one clock-cycle takes 1/80,000,000 seconds or 12.5 nanoseconds. The following is a portion of a listing file (dis-assembly) with a simple delay loop. The SUBS and BNE instructions are executed 800 times. The SUBS takes 1 cycle and the BNE takes (1+P) where P can vary between 1 and 3 cycles. In simulation P is 2 making the wait loop be of 4 cycles. On the real-board P can vary because of optimization using a pipeline. The minimum time to execute this code is 800*(1+(1+1))*12.5 ns= 30 us. The maximum time to execute this code is 800*(1+(1+3))*12.5 ns= 50 us. Since it is impossible to get an accurate time value using the cycle counting method, we will need another way to estimate execution speed.

```
0x00000158 F44F7016        MOV  R0,#800
0x0000015C 3801       wait SUBS R0,R0,#0x01
0x0000015E D1FD            BNE  wait
```

*(note: the **BNE** instruction executes in 3 cycles on the simulator, but an indeterminate number of cycles on the real board)*

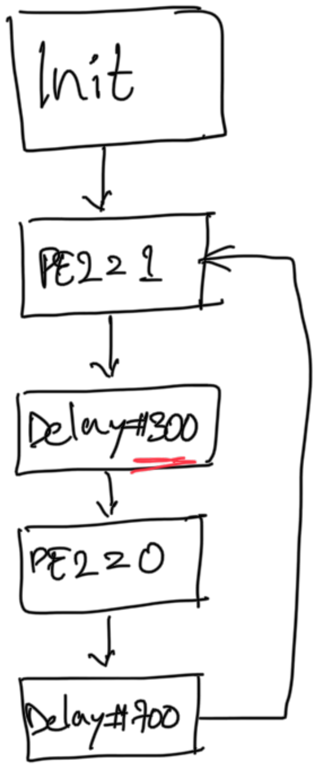
- To get Delay of
  150 ms ⟺ 300 loops
  350 ms ⟺ 700 loops

To check → look at logic analyzer



5. Stage2:

- Texas Init
- Init Port E clock (0x10)
- Init Port E (DIR = 0x04, DEN = 0x04)
- Enable Texas Oscilloscope
- loop
  make PE2 high
  Delay 150 ms
  make PE2 low
  Delay 350 ms



6. Stage3:

- Texas Init
- Init Port E clock (0x01)
- Init PortE (DIR = 0x04, DEN = 0x06)
- Enable Texas Oscilloscope
- Loop
  check PE1:

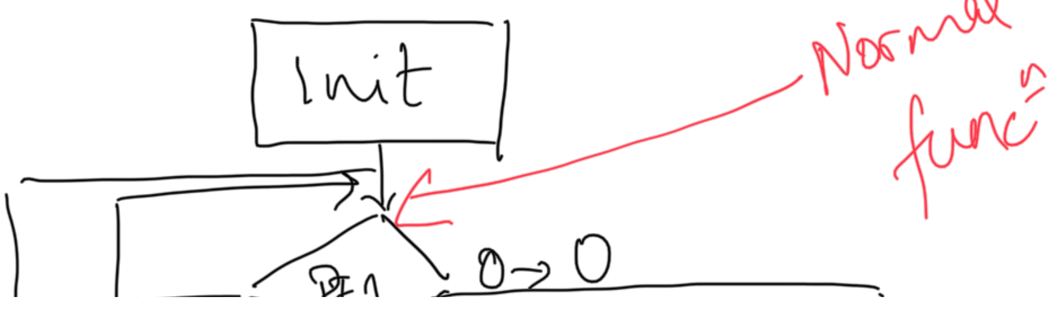
→ If high: loop till high, Inc duty cycle
  If duty cycle = 90%, set to 10%

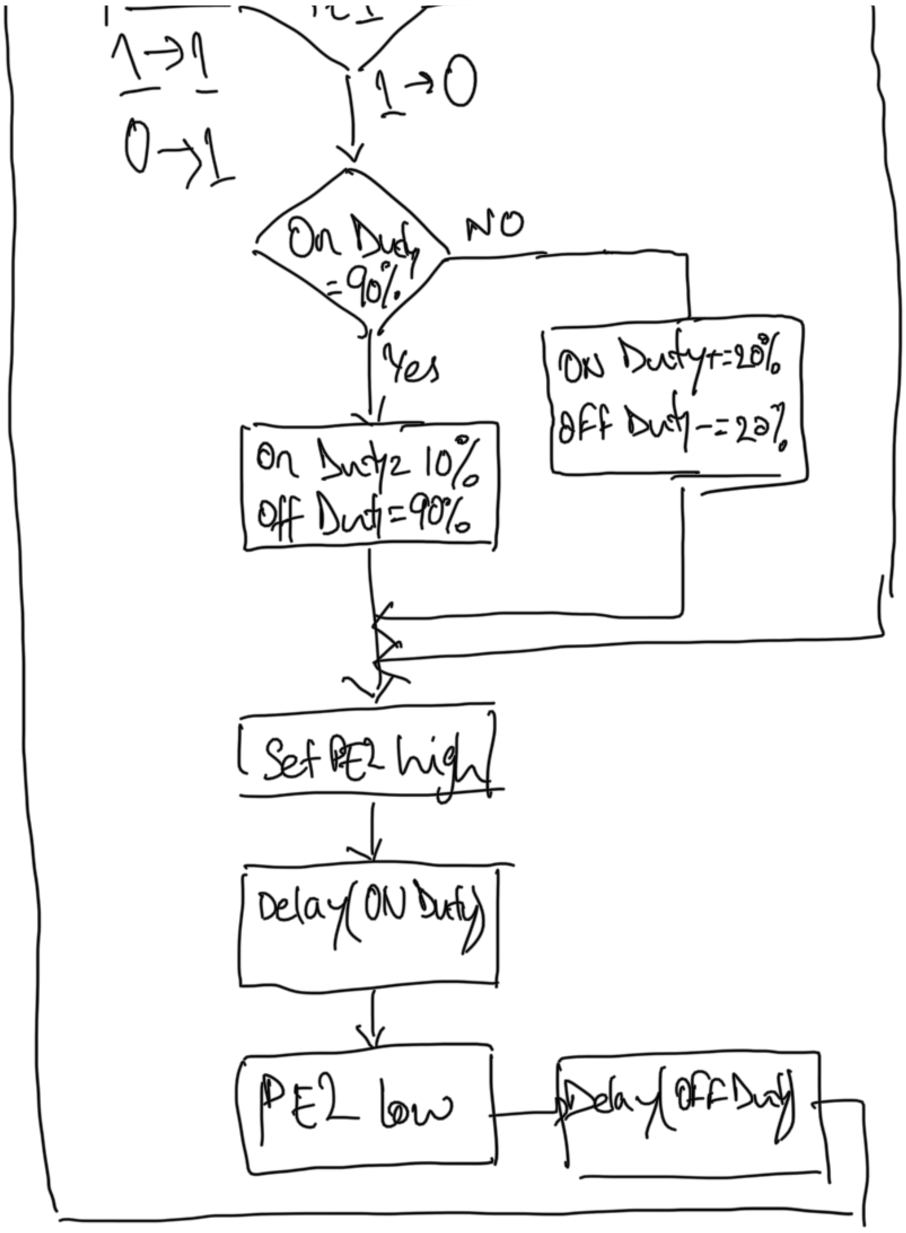
→ if low: make PE2 high
  Delay for ON Duty cycle
  make PE2 low
  Delay fo OFF Duty cycle
  loop back

- Init start Duty ON and Duty OFF (30%, 50%)

#900

#100





7. Stage 4: Build Circuit

8. Breathing LED:

- Req freq = 100 Hz ≈ 10 ms
- Req Duty = 10% ⟶ 90%
  ≈ 1ms ⟶ 9ms

- Basic delay block ≈ 0.5ms

  (from topic 4)

  (aka BDutyON BDutyOFF)

⇒ 1ms ≈ 2 * 0.5ms
 #loops Delay block

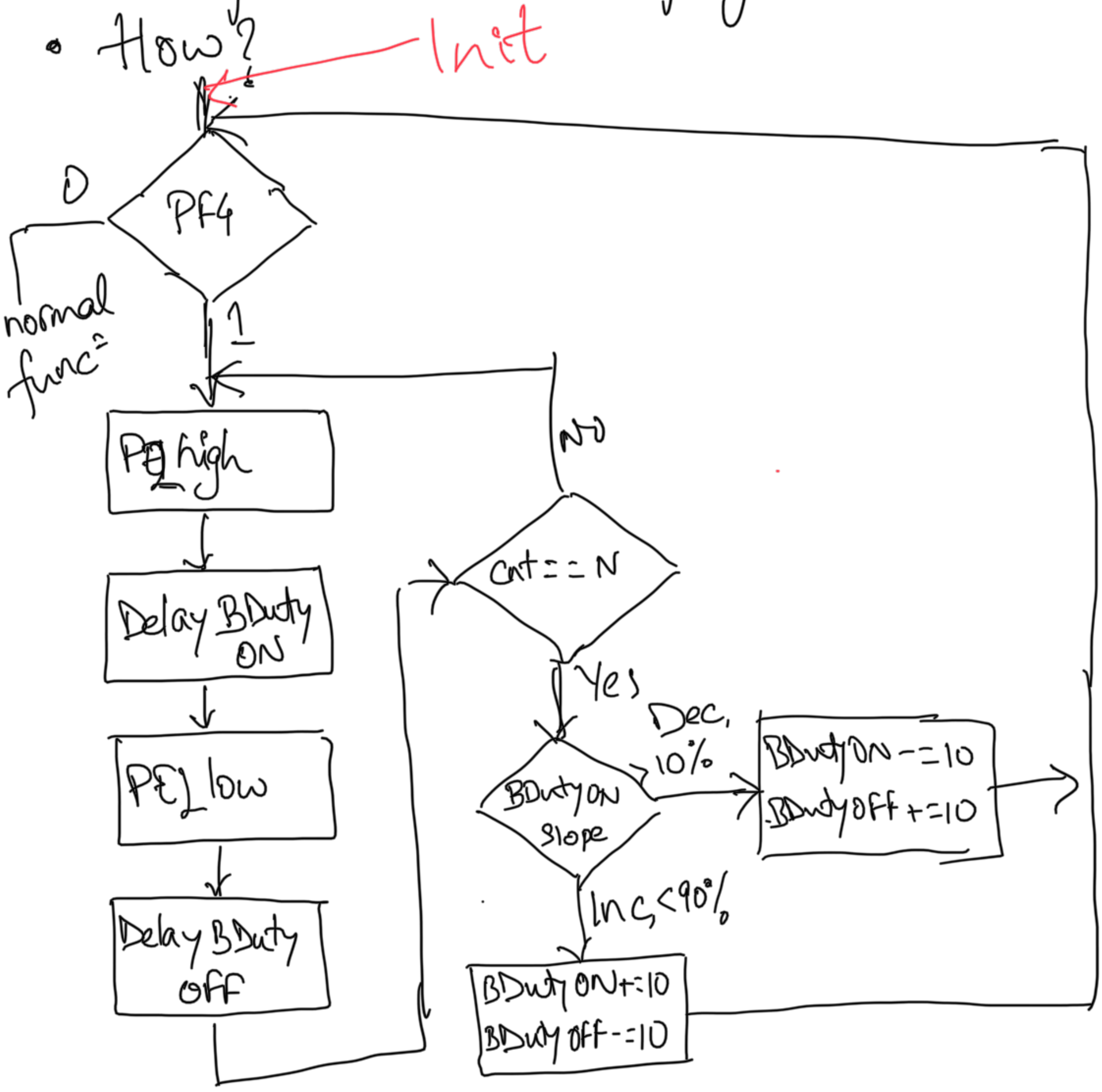
9ms ≈ 18 * 0.5ms
 #loops Delay block

- Can be initialized with 1ms BON Duty and 9ms BOFF Duty.

Stage 5: At this point you implemented 90% of the requirement of this lab. Now you will add the breathing feature which is enabled when PF4 (SW1) is pressed and disabled when released. We want you to be creative in devising a solution to implement this feature. However, here are some ideas:

- A breathing LED increases in brightness gradually and once it reaches its full brightness it decreases its brightness gradually till it reaches zero brightness. At which point it again repeats the increase.
- 2 Hz is too slow and will be visible to the naked eye as distinct on and off. We need the toggle the LED at a higher frequency (say 100 Hz) to be able to see the desired effect of duty-cycle impacting brightness.
- Varying brightness is achieved by varying duty-cycles. You may need more than 5 levels of duty-cycle for better breathing feel.
- Consider changing the duty-cycle at a programmable rate. That is, if your current duty-cycle is x%, stay at this duty-cycle for N iterations before changing to (x ± d)%.
- Remember, you can play with both the frequency and the duty-cycle.

- Create a delay func^n with
  1. Changable duty cycle (from Stage 3)
  2. Programmable number of cycles.
- Objective: Keep duty cycle at same value for N number of cycles before changing
- How?



Pseudo Code:

- Initialize Clock (Port E and Port F)
- Initialize Port F (PF4 - Input - digital)
- Initialize Port E (PE1 - Input - digital, PE2 - Output - digital)
- Initialize DutyON and DutyOFF for normal function (30%, 70%)
- Initialize DutyON and DutyOFF for breathing func^n (10%, 90%)
- Loop
  → Check PF4:
  if high, implement breating func^n, loop back
  if low, proceede to normal functionality

  → Normal functionality
  { Implement ... }

implement functionality from Stage 3