Problem 1

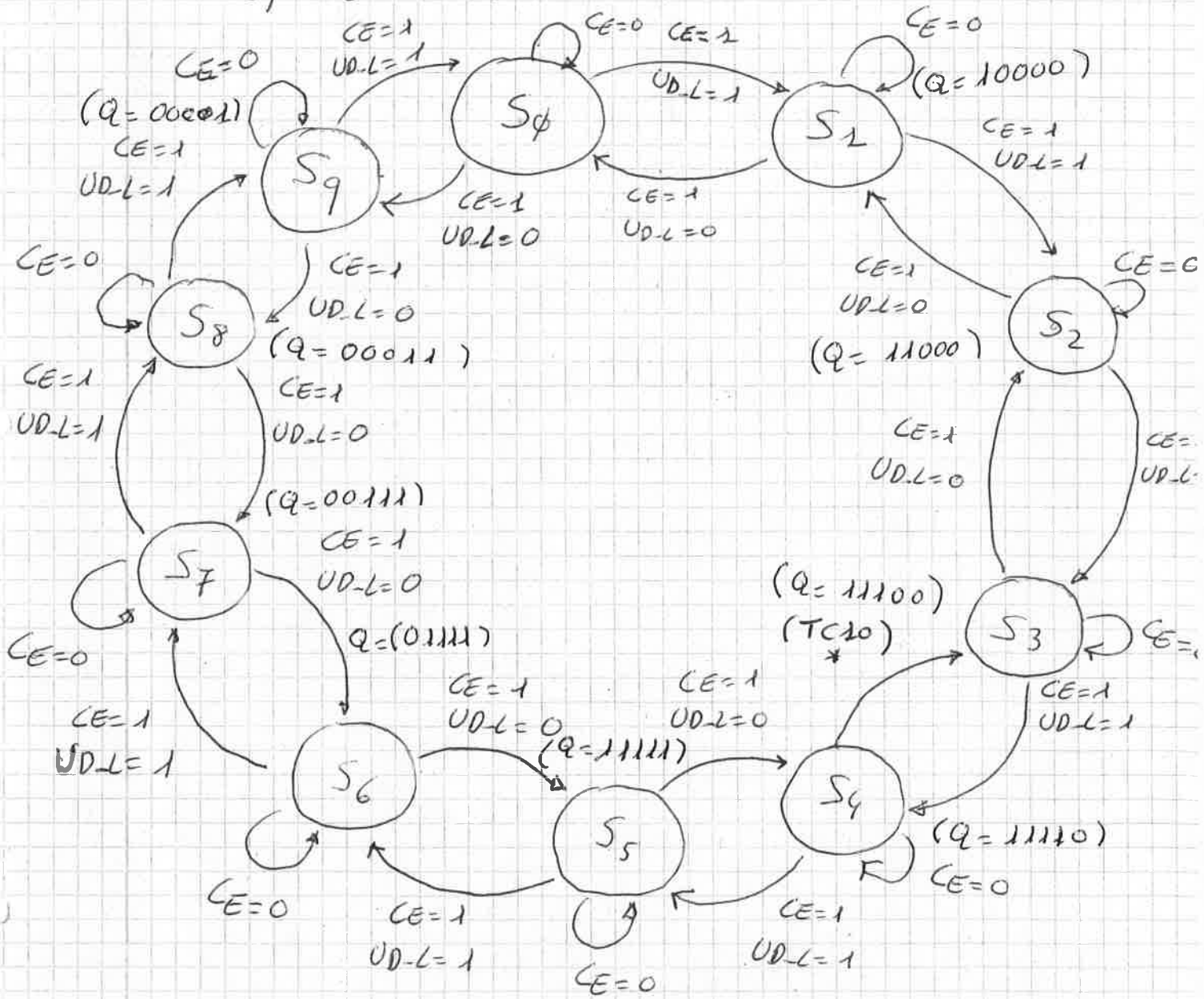
a)

	Q_4	Q_3	Q_2	Q_1	Q_0
S_ϕ	0	0	0	0	0
S_1	1	0	0	0	0
S_2	1	1	0	0	0
S_3	1	1	1	0	0
S_4	1	1	1	1	0
S_5	1	1	1	1	1
S_6	0	1	1	1	1
S_7	0	0	1	1	1
S_8	0	0	0	1	1
S_9	0	0	0	0	1

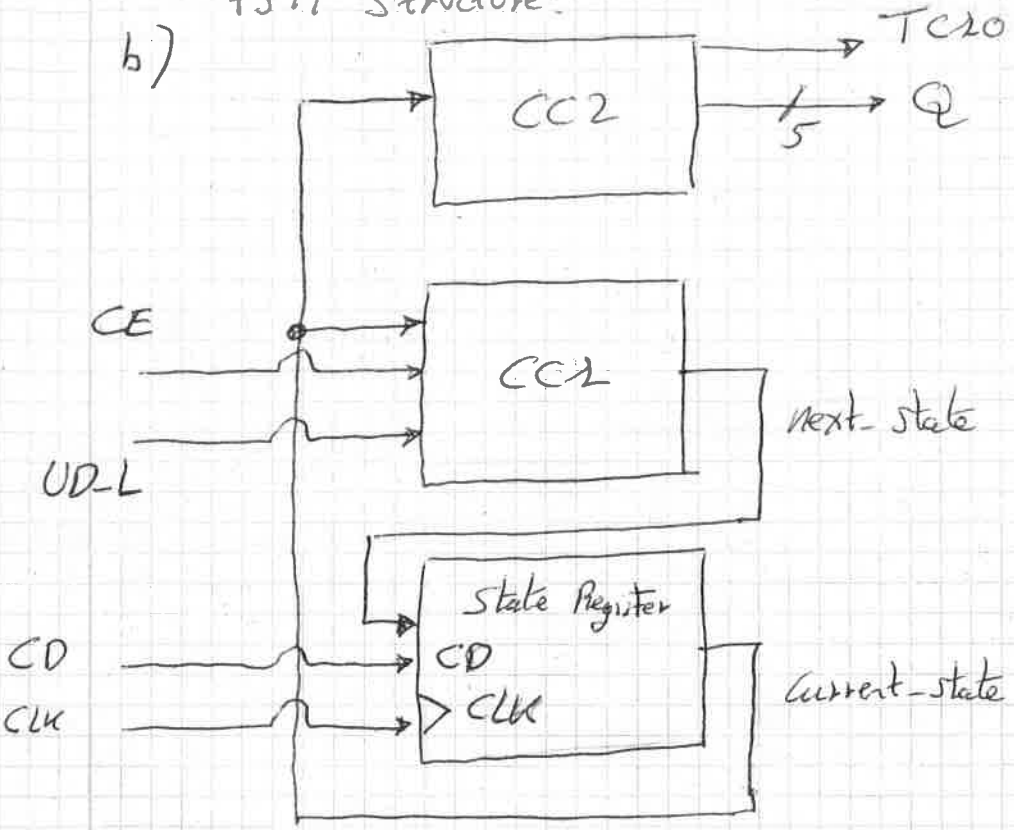
TC10 (*)

TC10 = 1 in S9 and Up and CE = 1

or in S0 and Down and CE = 1



b) FSM structure:

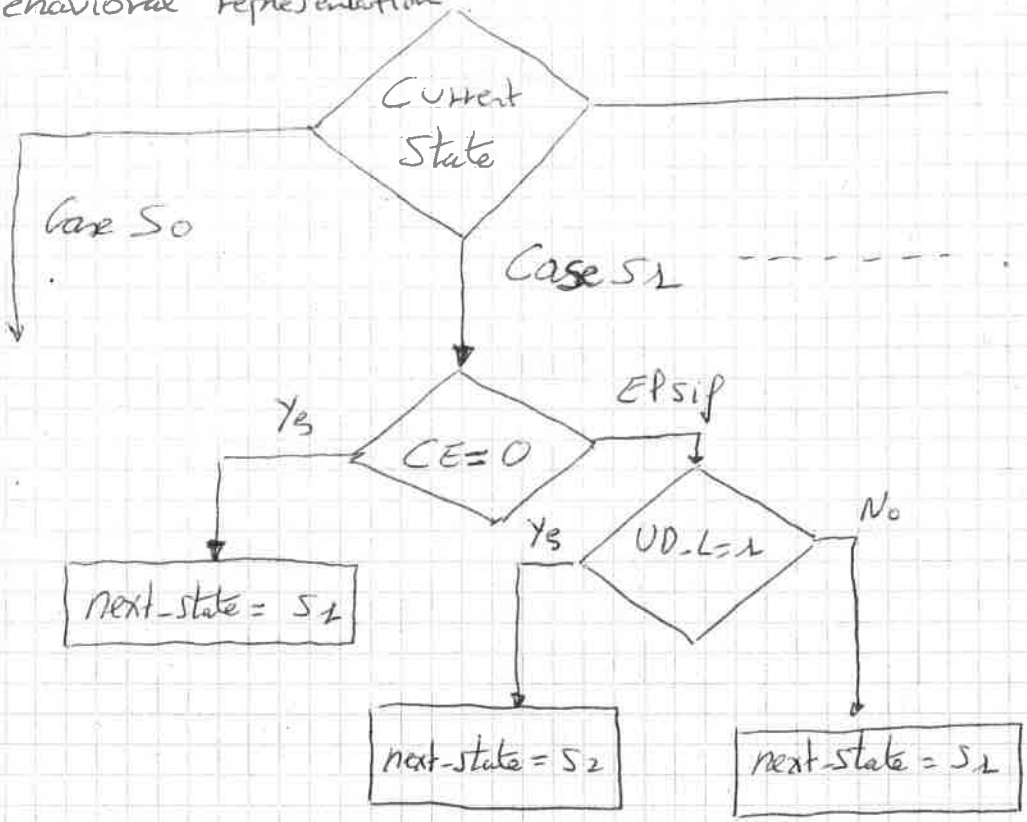


There are 10 states, then if coded in one-hot it is required 10 D-FF and if coded in binary it is required 4 D-FF.

c)

CE	UD_L	current_state	next_state
0	X	S_0	S_0
1	1	S_0	S_1
1	0	S_0	S_9
0	X	S_1	S_1
1	1	S_1	S_2
1	0	S_1	S_0
0	X	S_2	S_2
1	1	S_2	S_3
1	0	S_2	S_1
⋮	⋮	⋮	⋮
0	X	S_9	S_9
1	1	S_9	S_0
1	0	S_9	S_8

Behavioral representation



d) Main VHDL sentences of CC2.

```
CC2: Process ( current_state, CE) UD_L )
    BEGIN
      If ((current_state = S9 ) AND (CE= '1') AND (UD_L=1))
          OR (current_state = S0) AND (CE='1') AND (UD-L=0)  Then
           TC10<='1';
      ELSE
           TC10 <='0';
      END IF

      CASE current_state is
            When S0 =>
                 Q <= "00000";
            when S1 =>
                 Q <= "10000";
            when S2 =>
                 Q <= "11000";
                 .
                 .
                 .
            when S9 =>
                 Q <= "00001";
    END CASE
    END PROCESS
```

e) The functional simulation is ideal
In this case the maximum frequency can be ∞

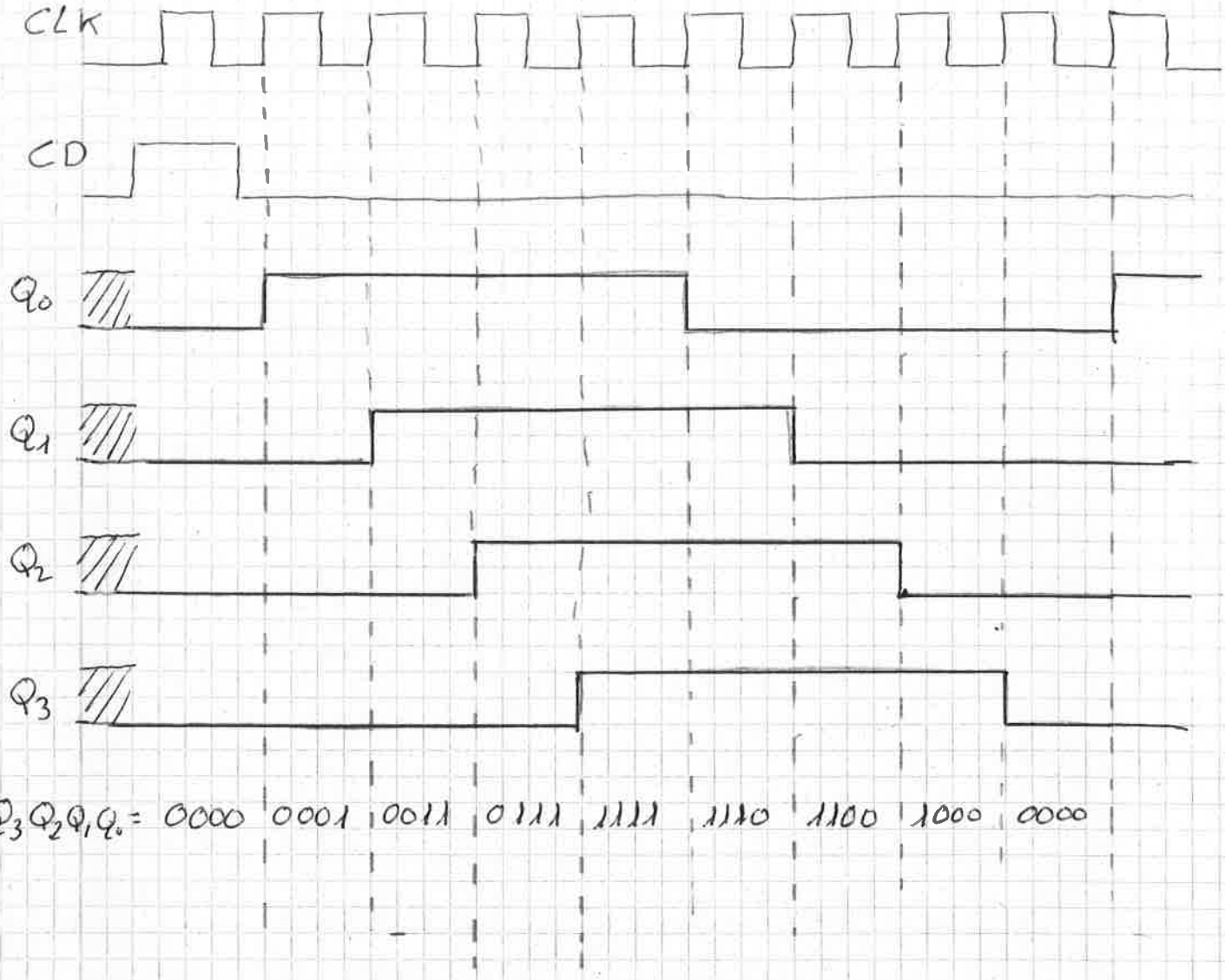
f)

$$f_{max} = \frac{1}{t_{CO} + N \cdot t_{PD}}$$

If CC2 has three levels of gate

$$f_{max} = \frac{1}{4'6ns + 3 \cdot 1'7ns} = \boxed{203'09\ MHz}$$

Problem 2



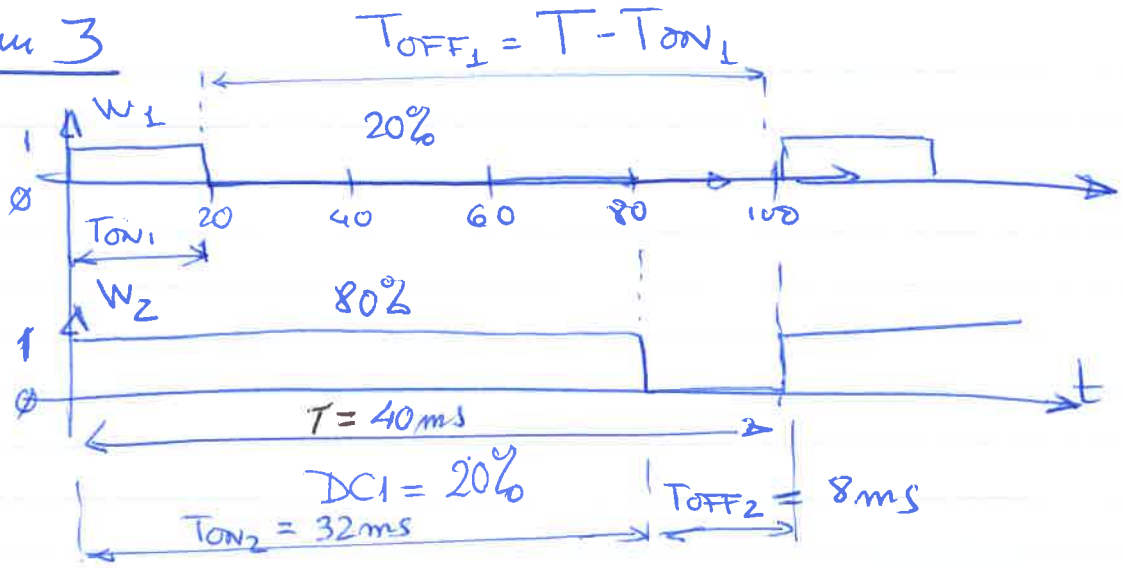
This circuit can be used to implement a Johnson counter of 4 bits

Two VHDL files will be required

$$T = (25\,Hz)^{-1} \rightarrow 40\,ms$$

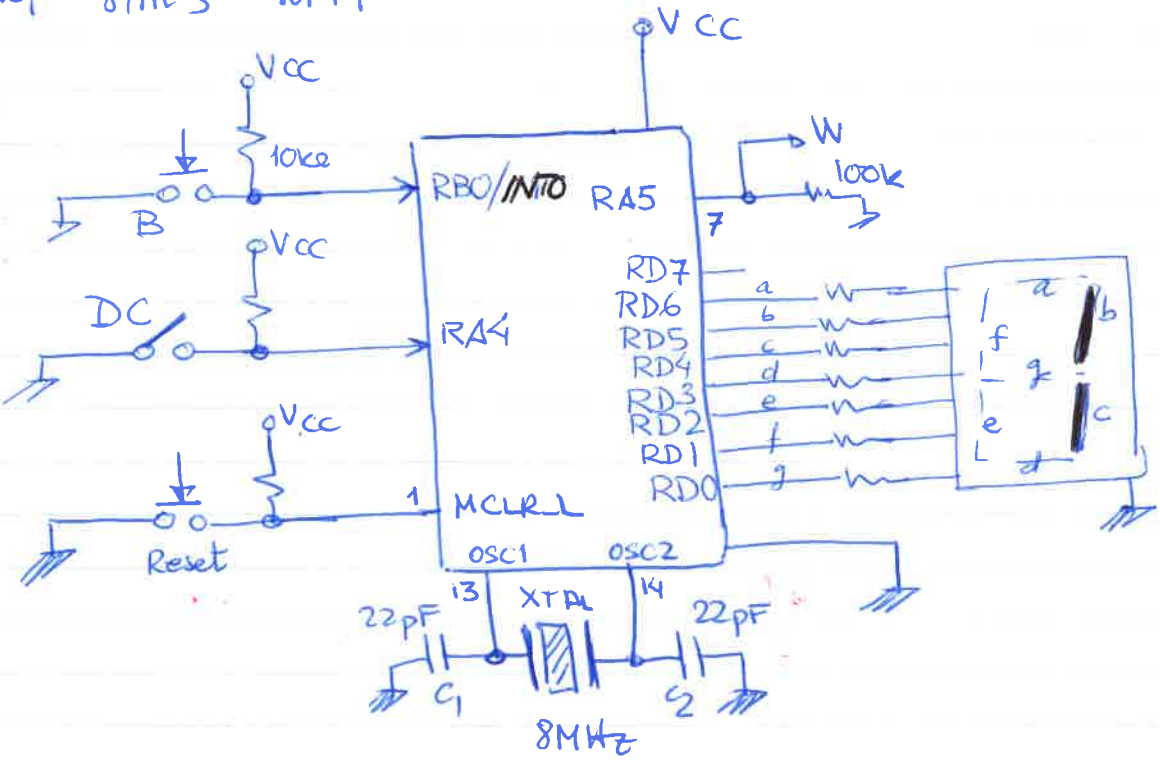
Problem 3

(a)



$T_{ON_1} = 8\,ms$ $T_{OFF_1} = 32\,ms$

(b)



Accordingly to the port pins selected,

TRISA: 7 6 5 4 3 2 1 0 — Ø Ø Ø **1** Ø Ø Ø Ø

TRISB: 7 6 5 4 3 2 1 0 — Ø Ø Ø Ø Ø Ø Ø **1**

PORTA: x x x x x x x x (DC ↓, W ↑)

PORTB: x x x x x x x x (B ↑)

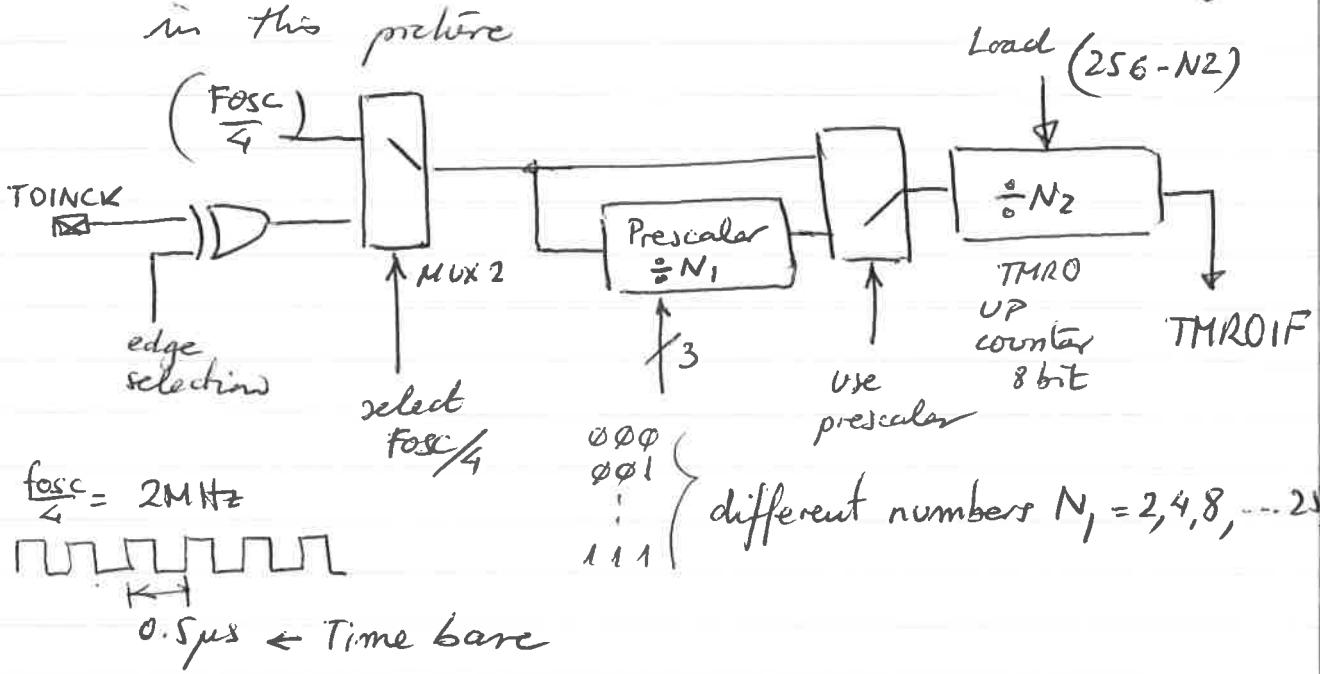
TRISD: 7 6 5 4 3 2 1 0 — Ø Ø Ø Ø Ø Ø Ø Ø

PORTD: x x x x x x x x (a b c d e f g ↓) — Segments — all outputs

'Ø' → output is the default value

* allow interrupts from INT0 and the Timer0 (TMR0)
{ INT0IE = 1
TMR0IE = 1
GIE = 1

(c) The TMR0 hardware is as represented approximately in this picture



TMR0IF when the system overflows

$$Timing\text{-}period = \left(\frac{4}{fosc}\right) \cdot N_1 \cdot N_2$$

$$T_{ON_1} = 8ms = 8000\,\mu s = (0{,}5\,\mu s) \cdot N_1 \cdot N_2$$

(T_{OFF_2}) ($N_1 \rightarrow 64$, $N_2 \rightarrow 250$)

$$T_{OFF_1} = 32ms = 32000\,\mu s = (0.5\,\mu s) \cdot N_1 \cdot N_2$$

(T_{ON_2}) ($N_1 \rightarrow 256$, $N_2 \rightarrow 250$)

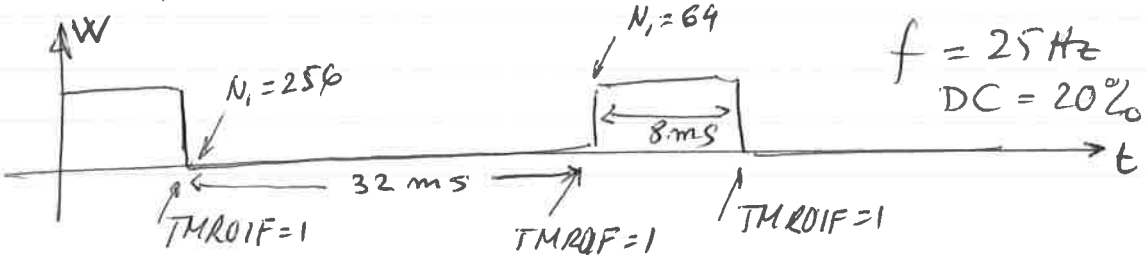
In this way we can load the TMR0 with the same value

$N_2 = 250 \rightarrow TMR0 = (256 - 250) = 6$

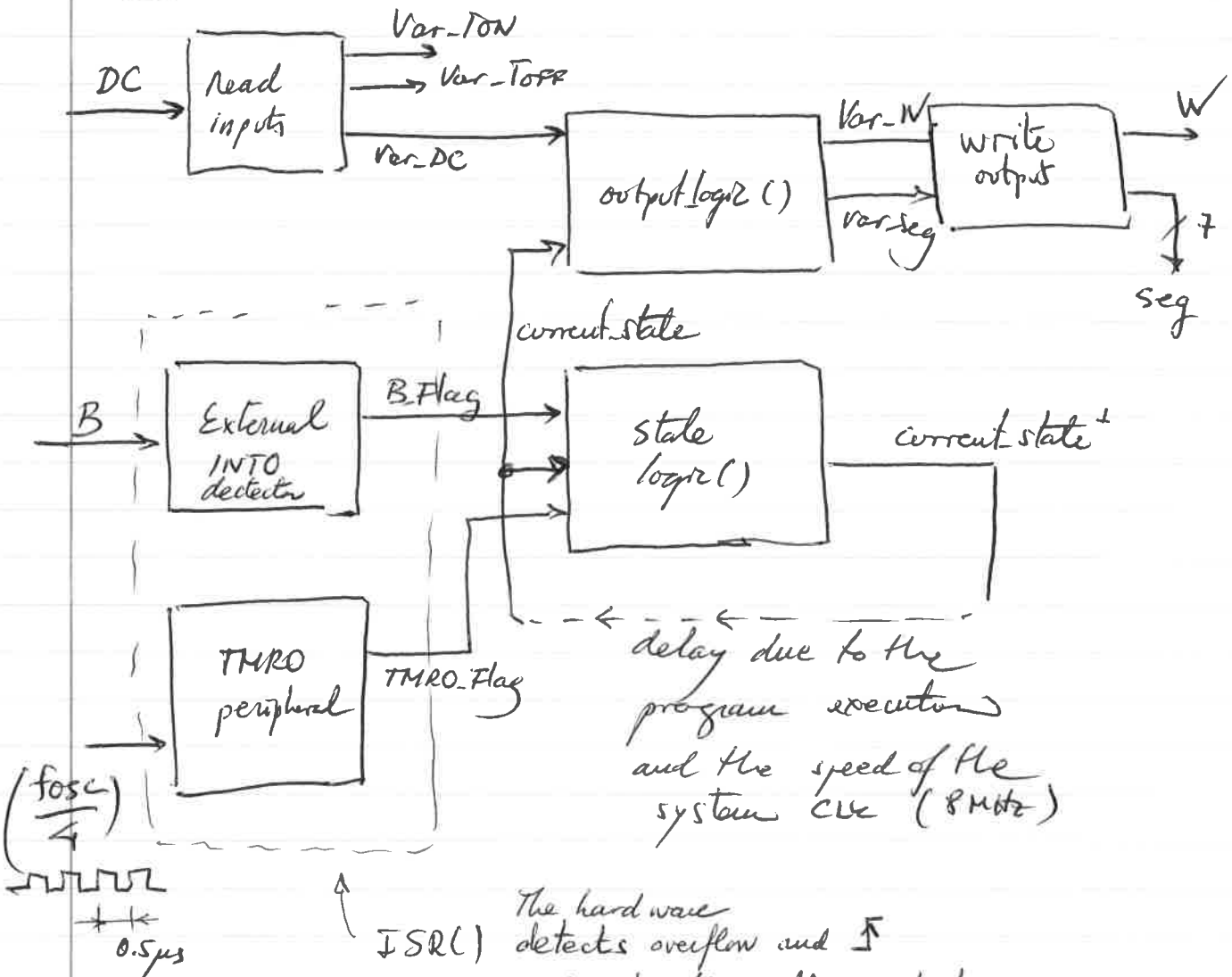
and depending on the prescaler, the 2 periods are possible. Swaping $N_1 \rightarrow N_1 = 64 \rightarrow$ TMR0IF every 8 ms

$N_1 = 256 \rightarrow$ TMR0IF every 32 ms

In this way the mechanism is



d



The hardware detects overflow and ↑ and sets the flags which are transformed into convenient variables to run the FSM

Char → Current_state variable → any ASCII value, like A, B, C, D, E

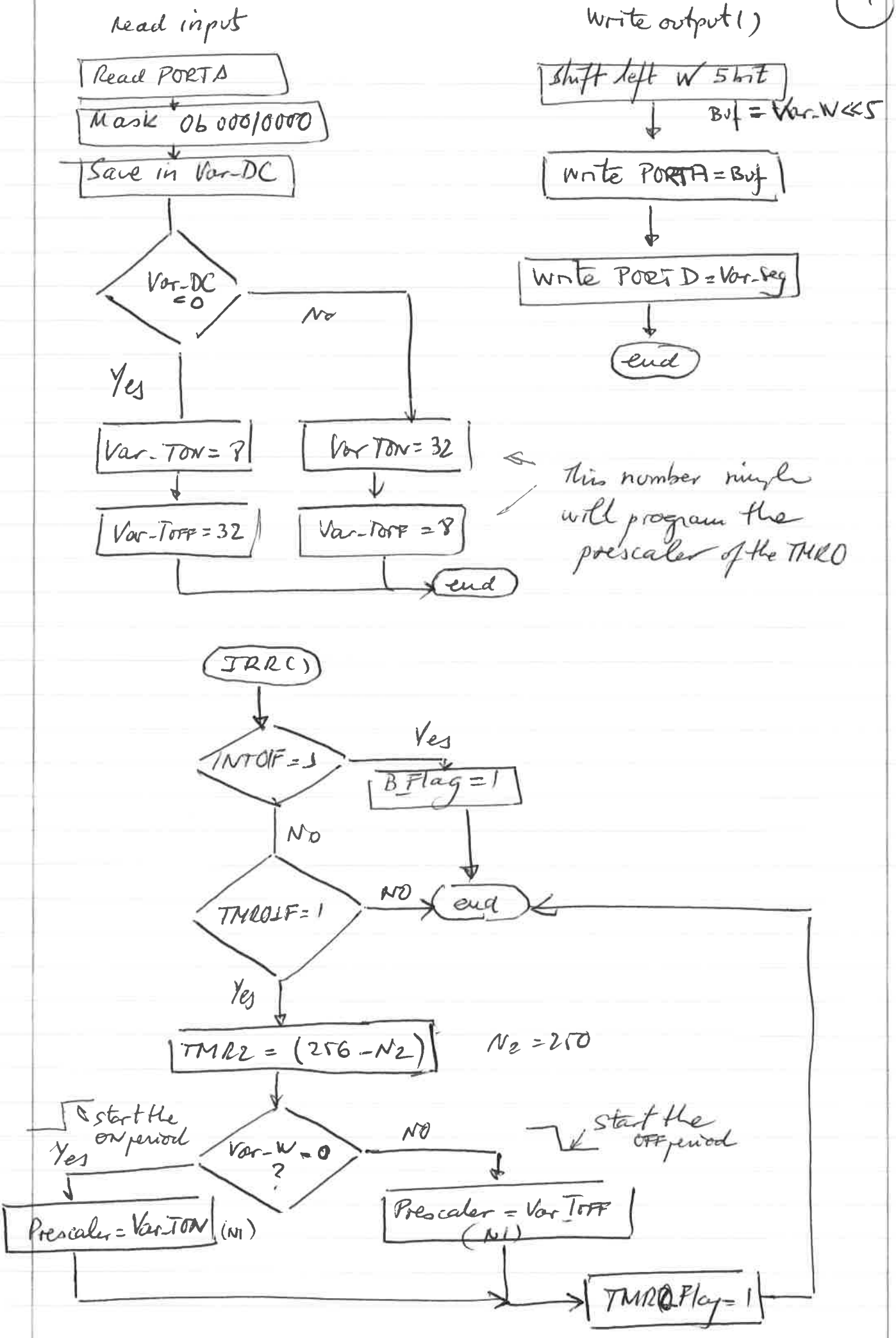
	7	6	5	4	3	2	1	0	
Char →	Ø	Ø	Ø	Ø	Ø	Ø	Ø	·	B_Flag
" →	Ø	Ø	Ø	Ø	Ø	Ø	Ø	·	TMR0_Flag
" →	Ø	Ø	Ø	Ø	Ø	Ø	Ø	·	Var_DC { Ø → DC1, 1 → DC2 }
" →	Ø	Ø	Ø	Ø	Ø	Ø	Ø	·	Var_W { Ø → OFF, 1 → ON }
" →	Ø	a	b	c	d	e	f	g	Var_seg

This value will set the N_1 ↑ 64

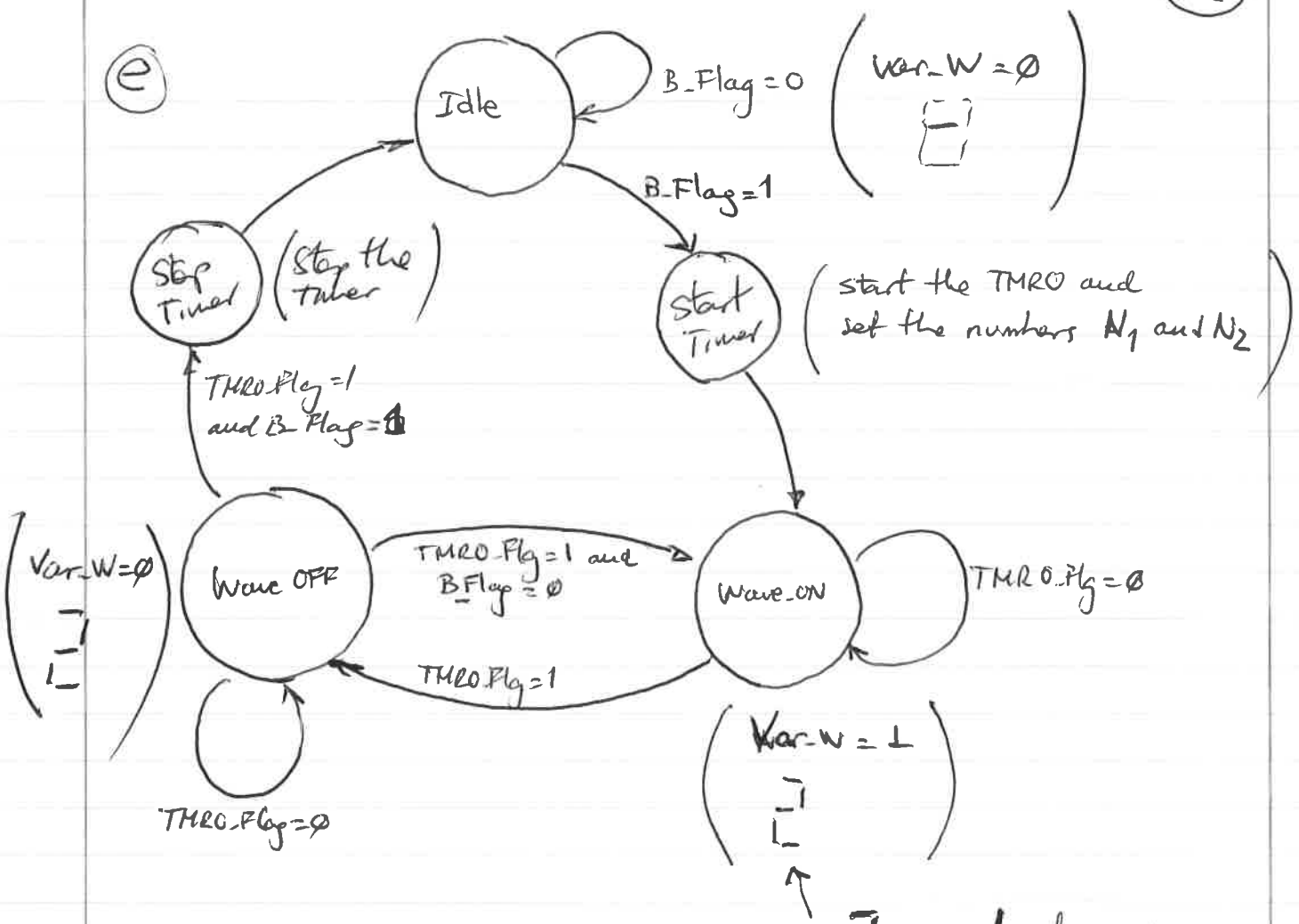
→ This value will set the N_1 ↑ 256

in the ISR() when reloading the TMR0

Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Blank	0x00
Ø	Ø	Ø	Ø	Ø	Ø	Ø	1	(-)	0x01
Ø	Ø	1	1	Ø	Ø	Ø	Ø	(!)	0x30
Ø	1	1	Ø	1	1	Ø	1	(É)	0x6D


Read input
Read PORTA
Mask 0b 000/0000
Save in Var_DC
Var_DC = 0
No
Yes
Var_TON = 8
Var_TOFF = 32
Var TON = 32
Var_TOFF = 8
end
This number simply will program the prescaler of the TMR0
Write output()
Shift left W 5 bit
Buf = Var_W<<5
Write PORTA = Buf
Write PORTD = Var_Seg
end
IRRC()
INT0IF = 1
Yes
B_Flag = 1
No
TMR0IF = 1
NO
end
Yes
TMR2 = (256 - N2)
N2 = 250
Var_W = 0 ?
Yes
start the ON period
NO
start the OFF period
Prescaler = Var_TON (N1)
Prescaler = Var_TOFF (N1)
TMR0Flag = 1

e)



Output logic truth table

Var_DC	Current_state	Var_reg	Var_W	Var_TON	Var_TOFF
X	Idle	0x01	0	x	x
0	Start Timer	0x00	1	T_{ON1}	T_{OFF1}
1	Start_Timer	0x00	1	T_{ON2}	T_{OFF2}
0	Wave ON	0x30	1	x	x
1	Wave ON	0x6D	1	x	x
0	Wave_OFF	0x30	0	x	x
1	Wave_OFF	0x6D	0	x	x
x	Stop_Timer	0x00	0	x	x

state_logic() truth table

each time the loop is executed ↓ +

B_Flag	TMR0_Flag	Current_state	Current_state+
0	X	Idle	Idle
1	X	Idle	start_Timer
X	X	start_Timer	Wave_ON
X	0	Wave_ON	Wave_ON
X	1	Wave_ON	Wave_OFF
X	0	Wave_OFF	Wave_OFF
0	1	Wave_OFF	Wave_ON
1	1	Wave_OFF	Stop_Timer
X	X	Stop_Timer	Idle

This table is for setting all the state transitions

This is the way the main program is organised → FSM