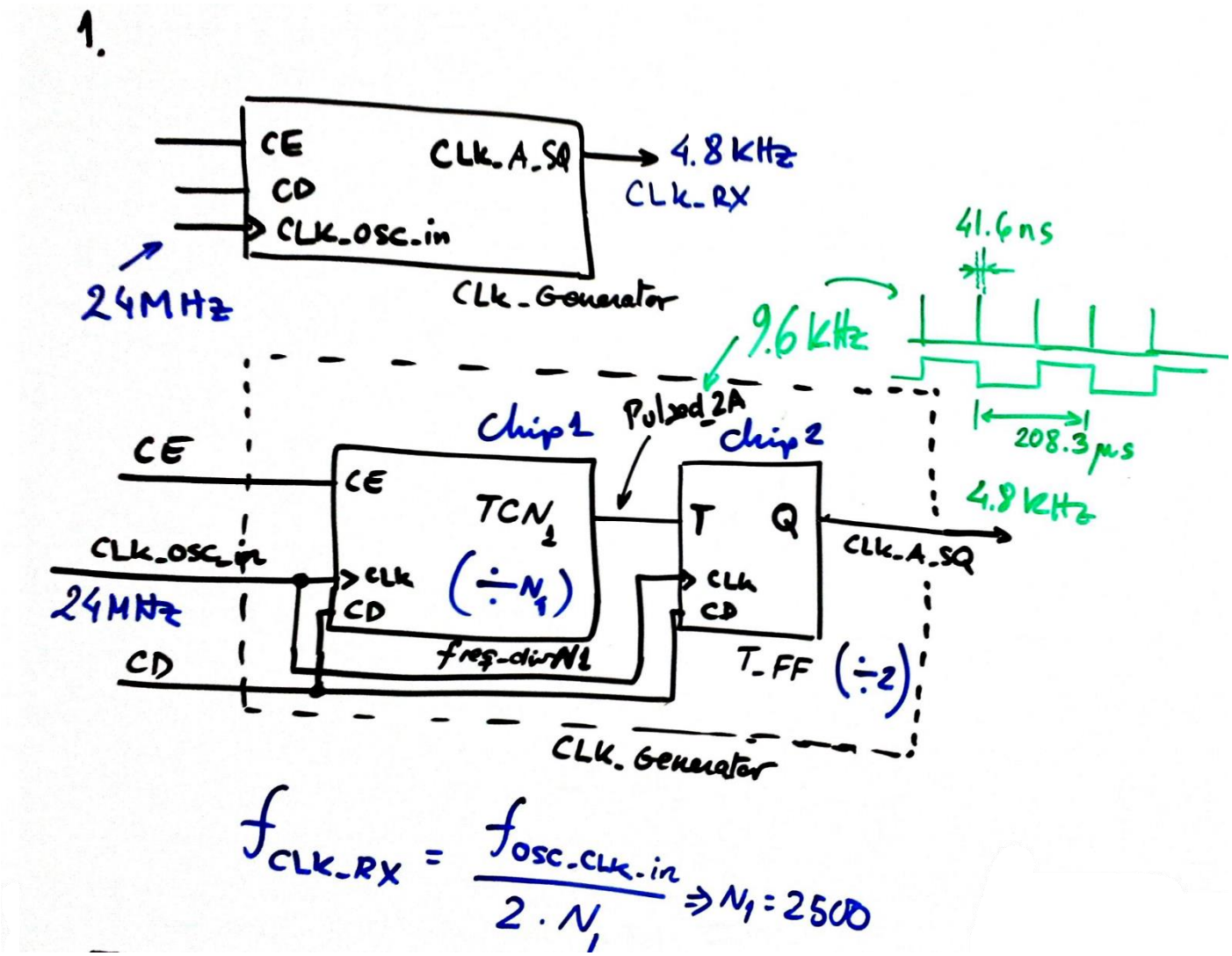
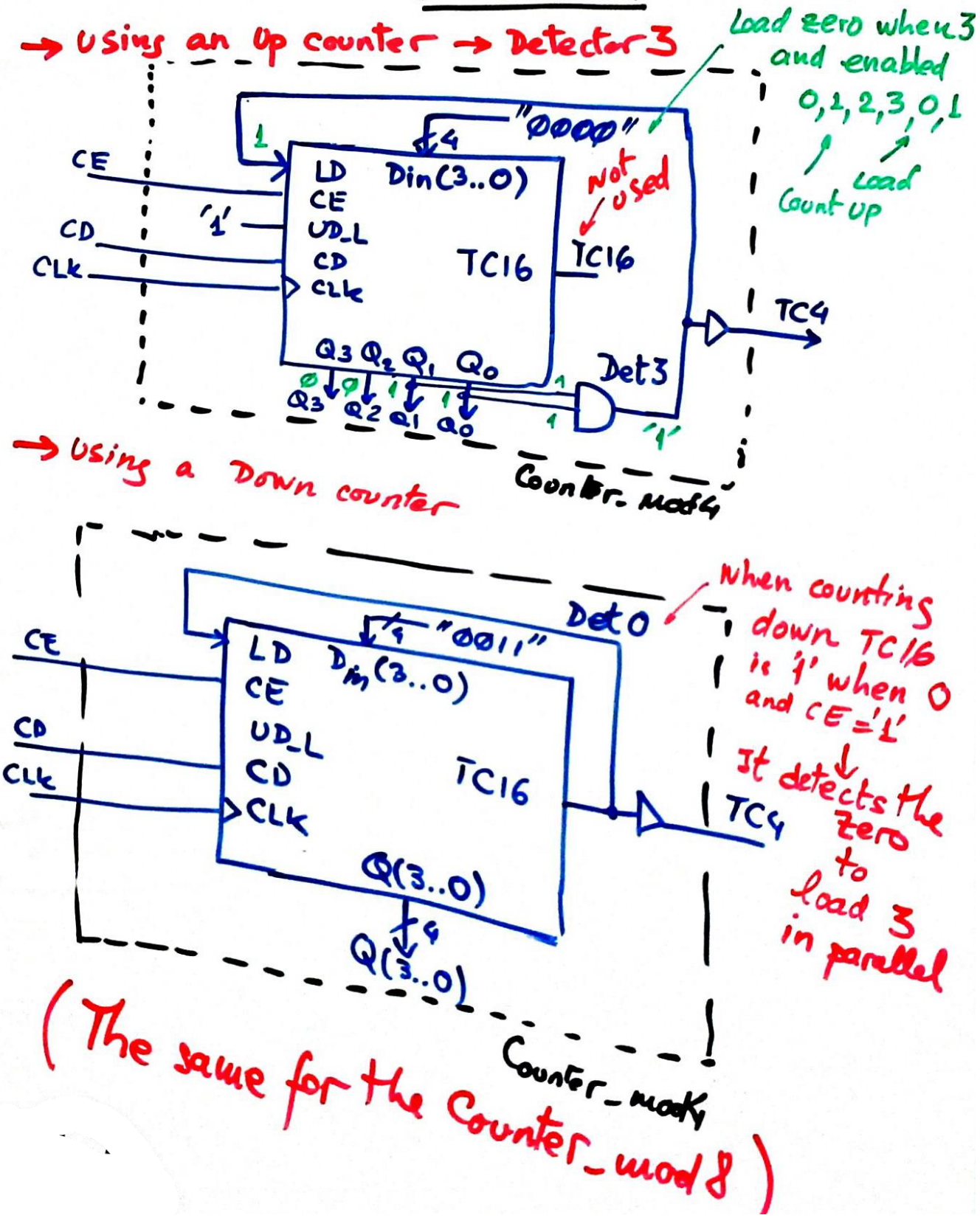
Example of EXAM solution

1.

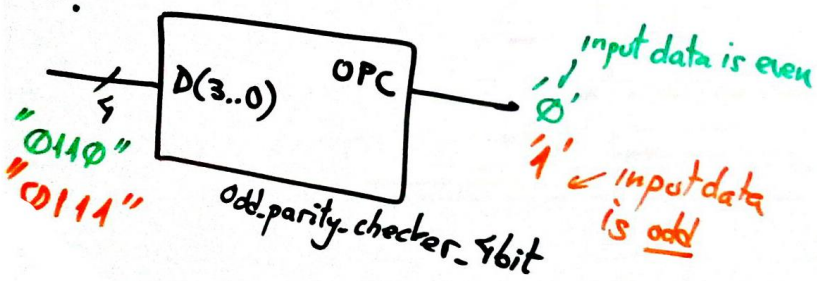


$$f_{CLK\_RX} = \frac{f_{OSC\_CLK\_in}}{2 \cdot N_1} \Rightarrow N_1 = 2500$$

2. Inventing a Counter_mod4 using the plan C2 and a standard Counter_mod16

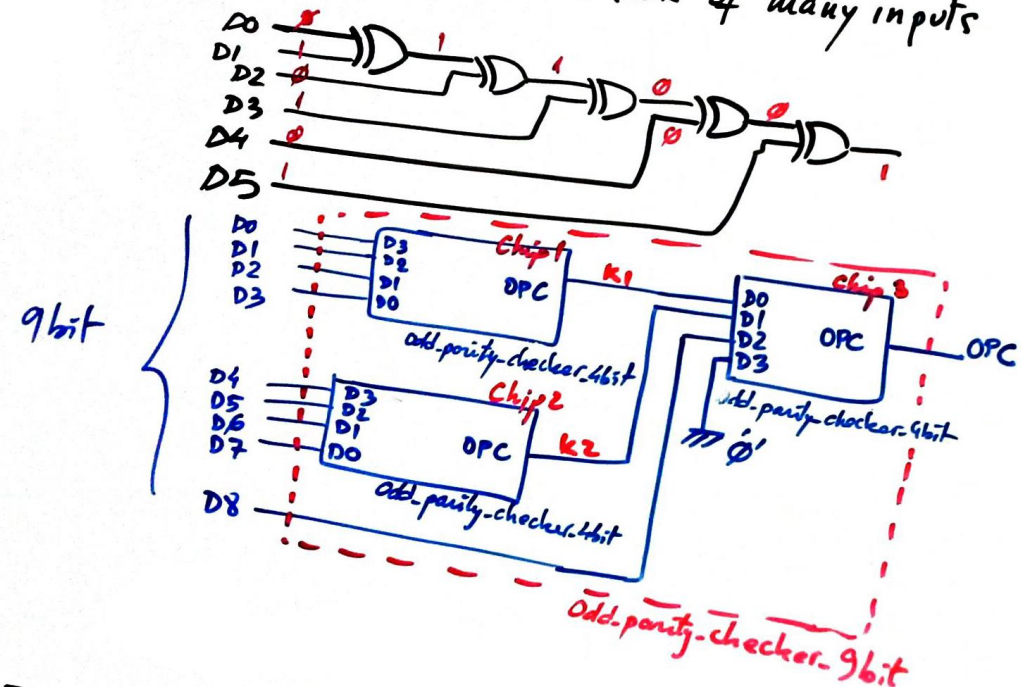


3.

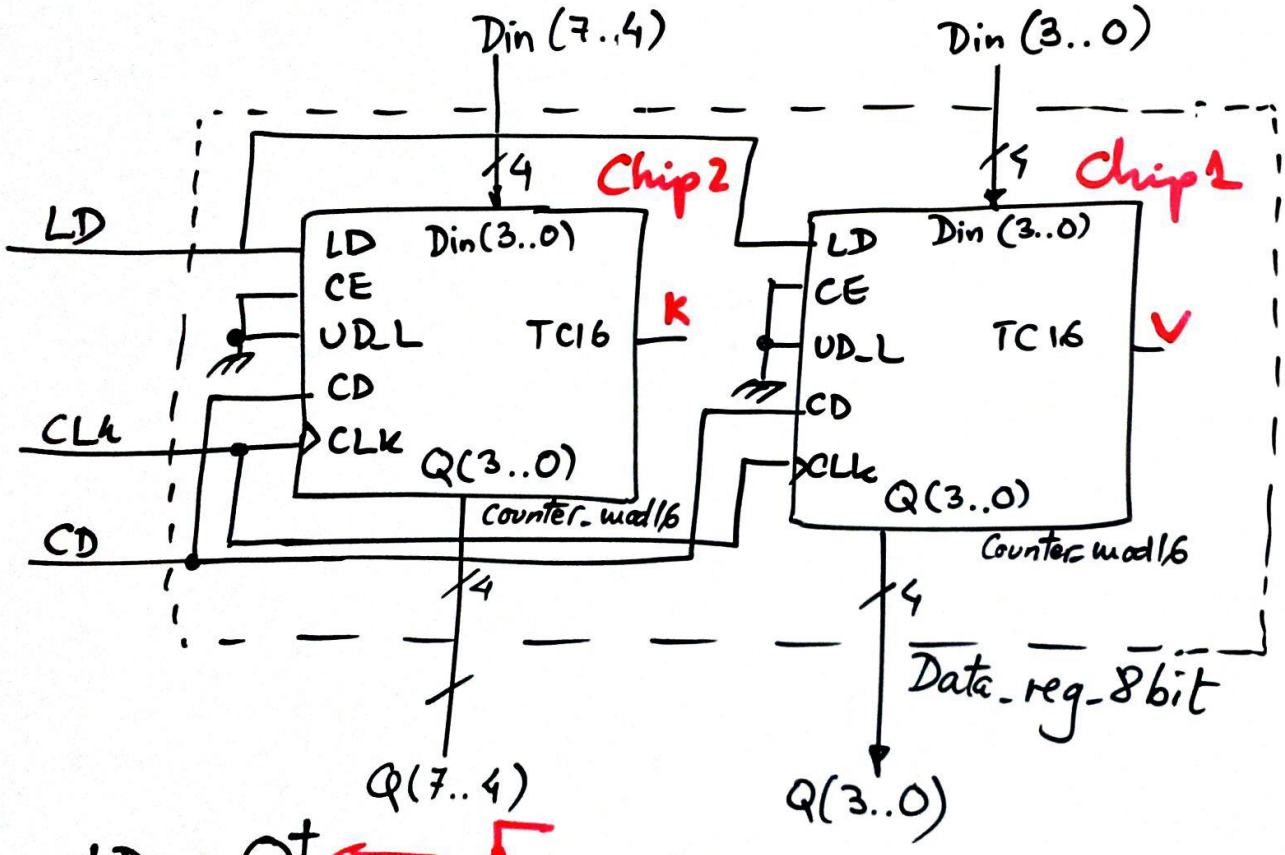


D3	D2	D1	D0	OPC
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
⋮				
1	1	1	0	1 ← odd detected
1	1	1	1	0 ← even

The OPC table is like a XOR of many inputs

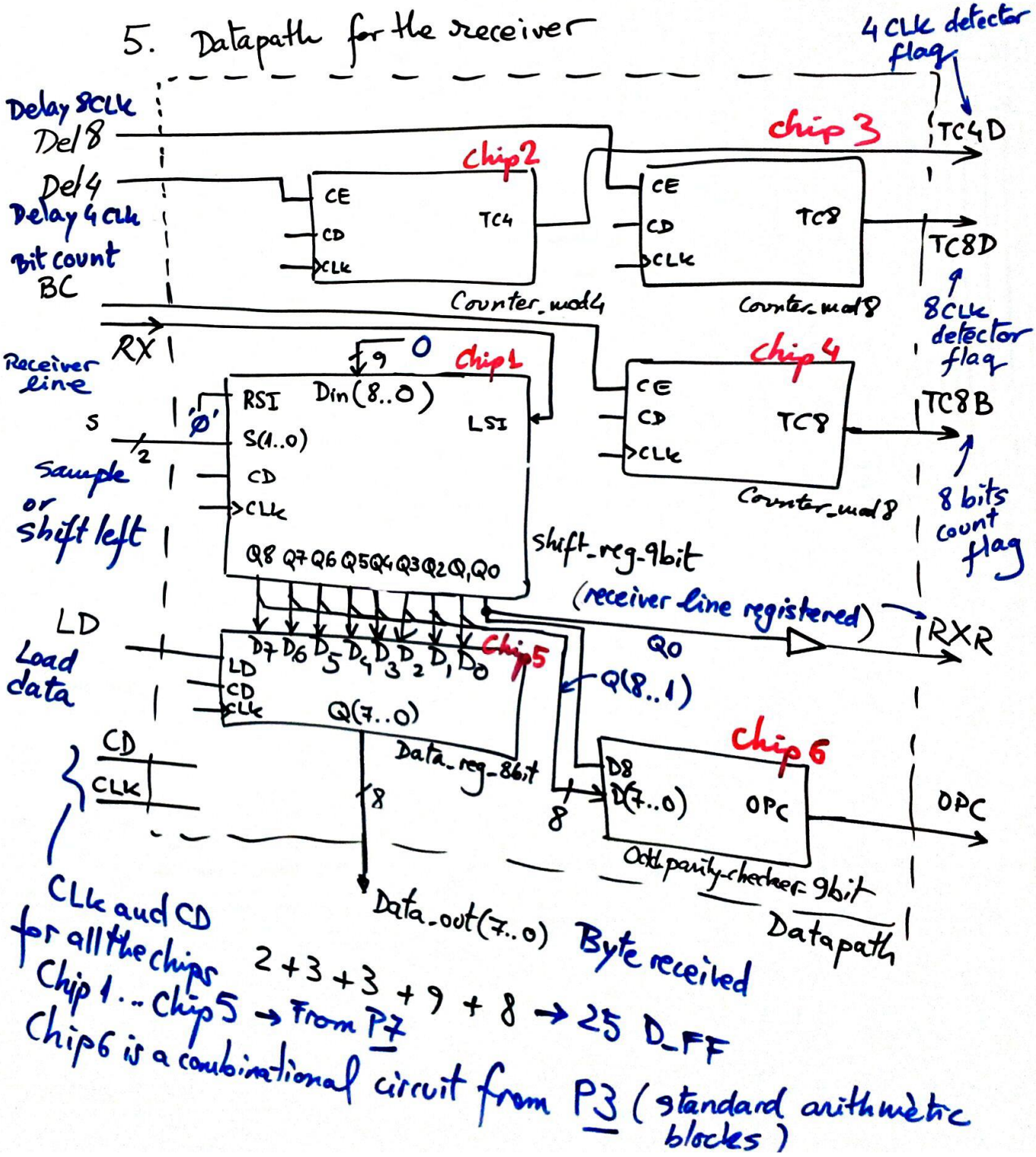


4. 8-bit data register using the plan C2 and components counter_mod16

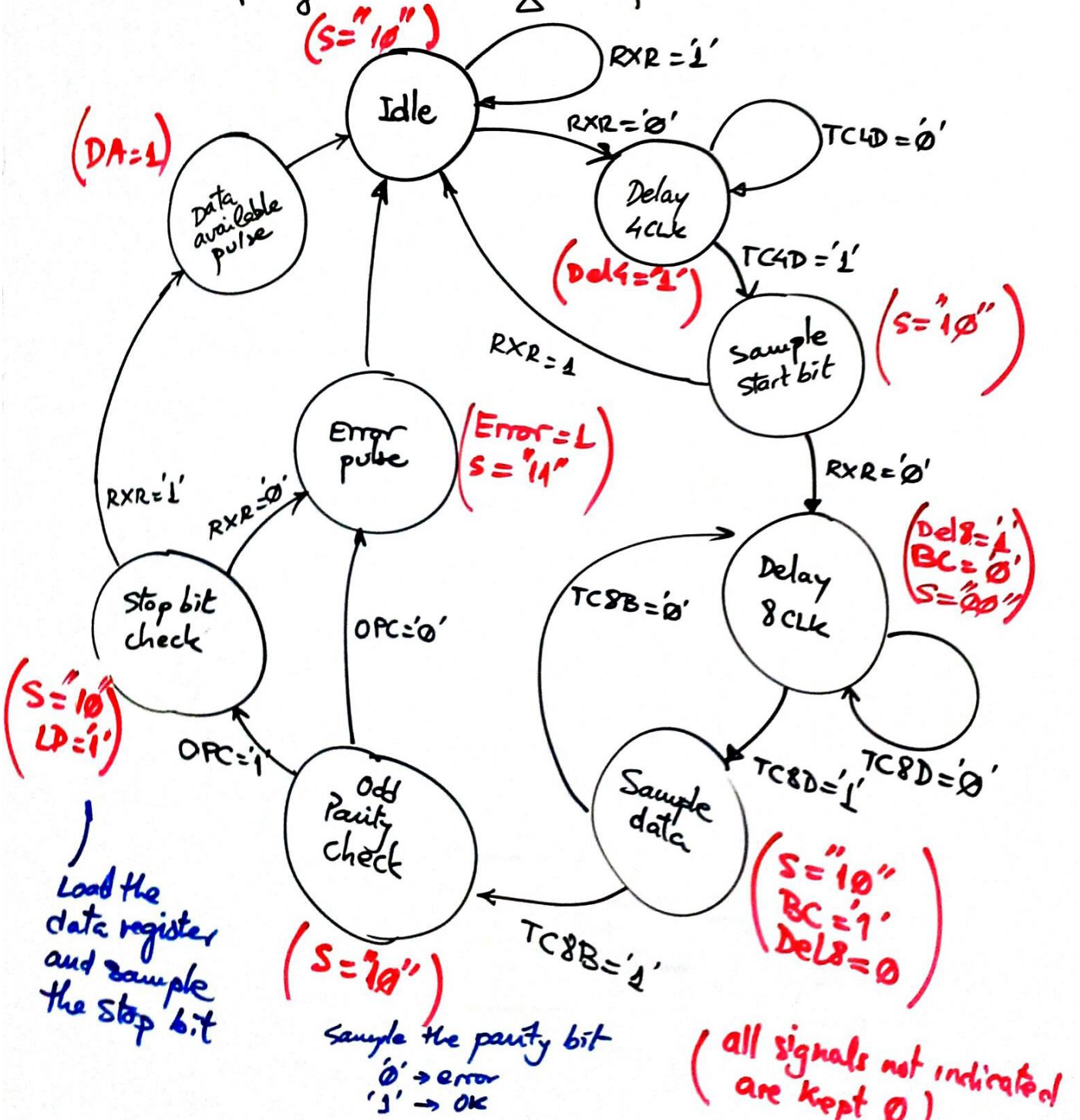


LD	Q^+	$\leftarrow$ CLK
0	Q	Do nothing because the CE = 0 (keep data)
1	Din	Parallel Load is write data

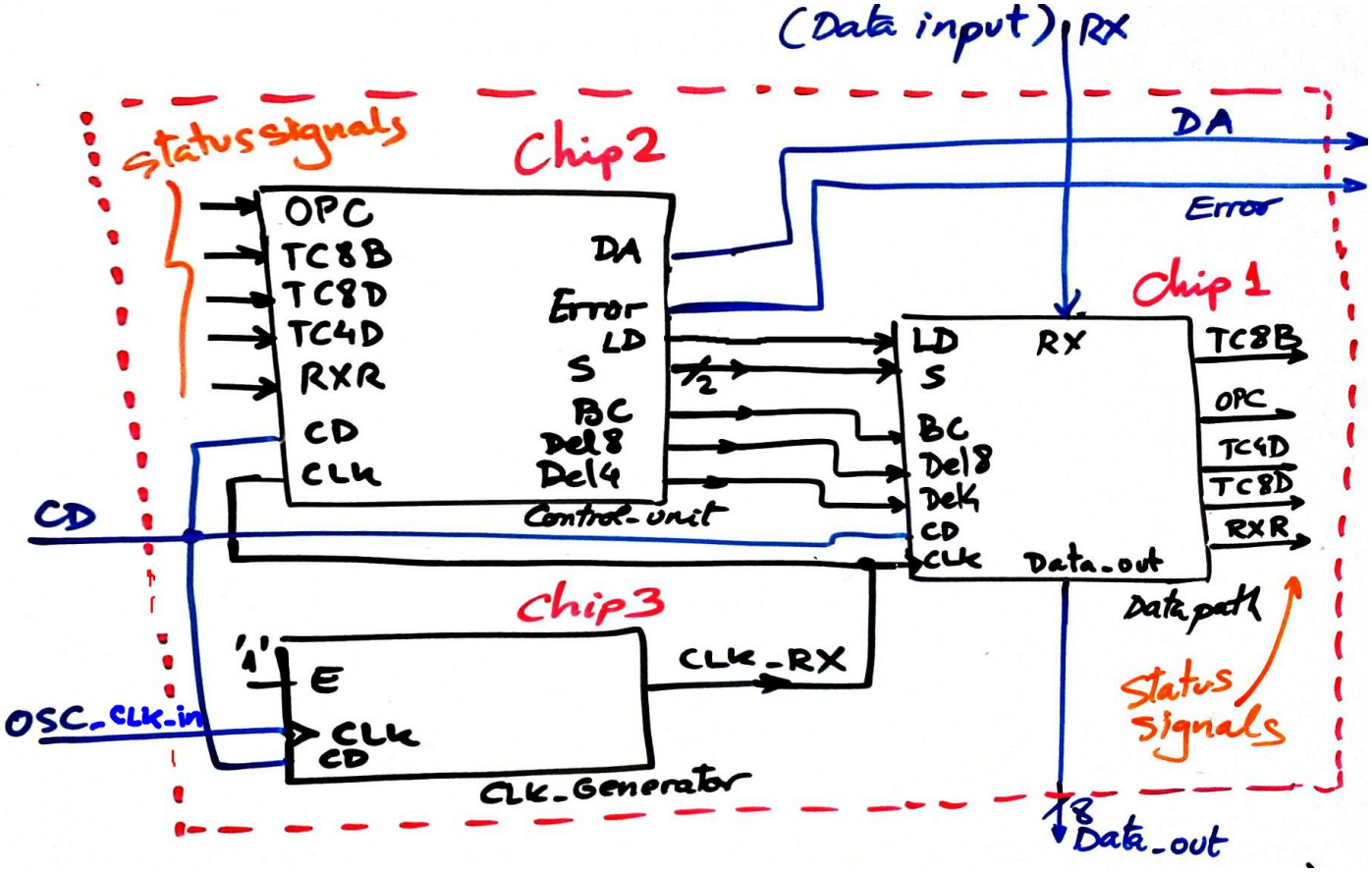
5. Datapath for the receiver



6. Example of state diagram for the control unit



This is an initial state diagram that can be modified on testing design stage



7.

Control_unit → 4D_FF (Coding in binary)
Datapath → 25 D_FF
CLK_Generator → 13 D_FF

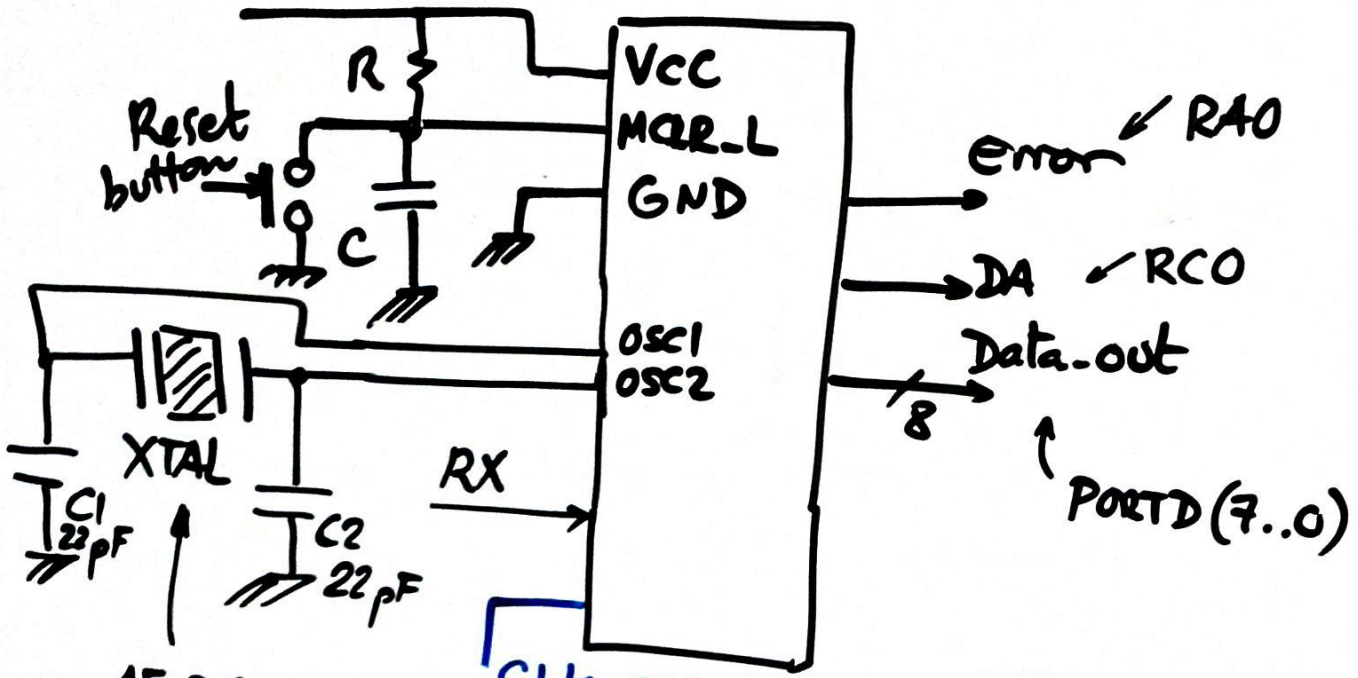
42 D_FF

→ It is possible to use a 10-bit shift register so that the stop bit check and the data write can be solved in two states

→ using components Countermod16 the Counter_mod4 can be 4 D_FF instead of 2 depending on the VHDL synthesiser options

8. Solving the project using a μC

Hardware circuit



Convenient value to generate standard USART transmitter and receiver frequencies using timer peripherals

RX → If an external interrupt is used → RB1/INT1

If it is polled in the main loop → any port pin → RA0 (or the same RB1 used as simple digital input)

RB0/INT0 to generate the baud rate

TRISA: Ø Ø Ø Ø Ø Ø Ø Ø (Error)

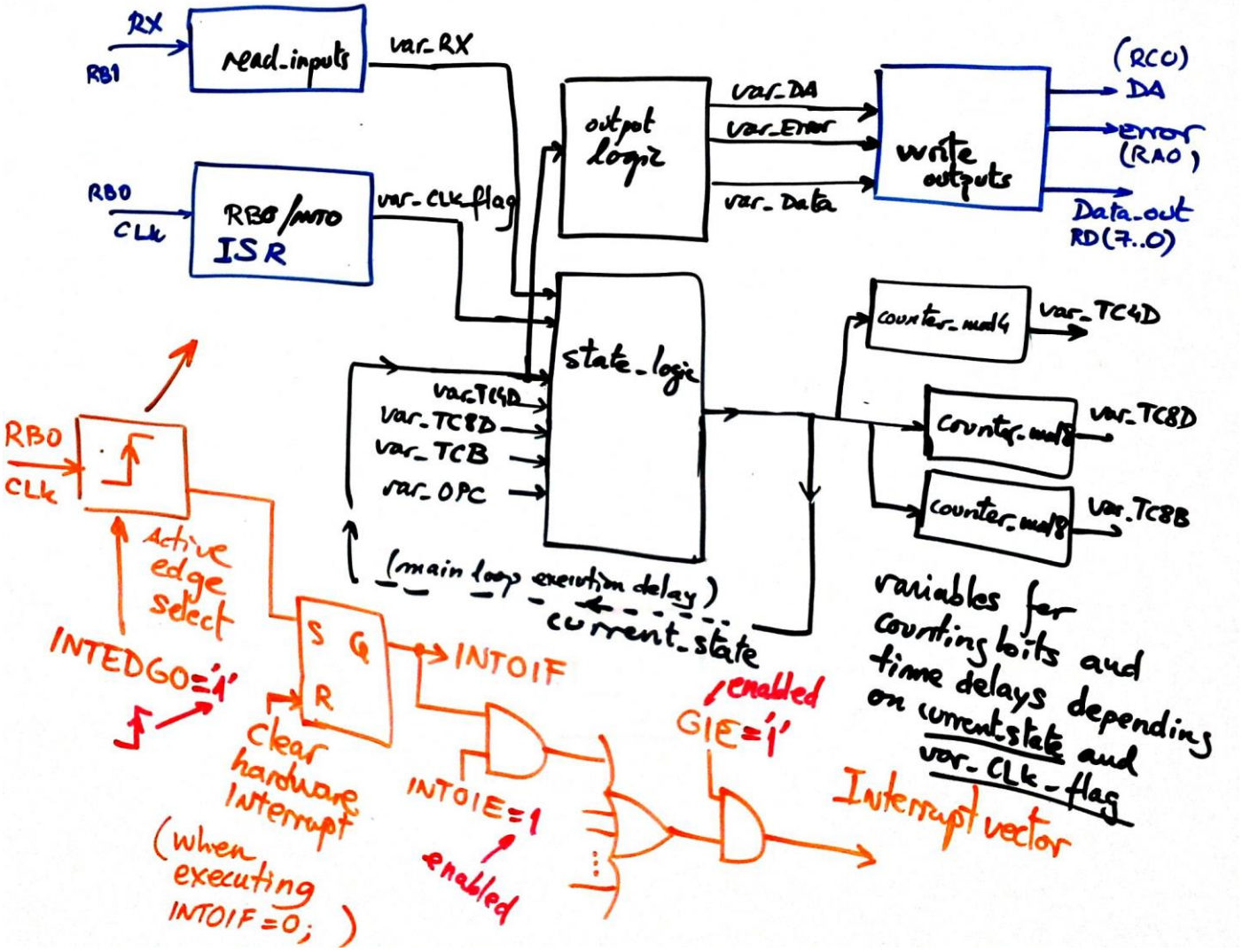
TRISB: Ø Ø Ø Ø Ø Ø 1 1 (CLK_RX)

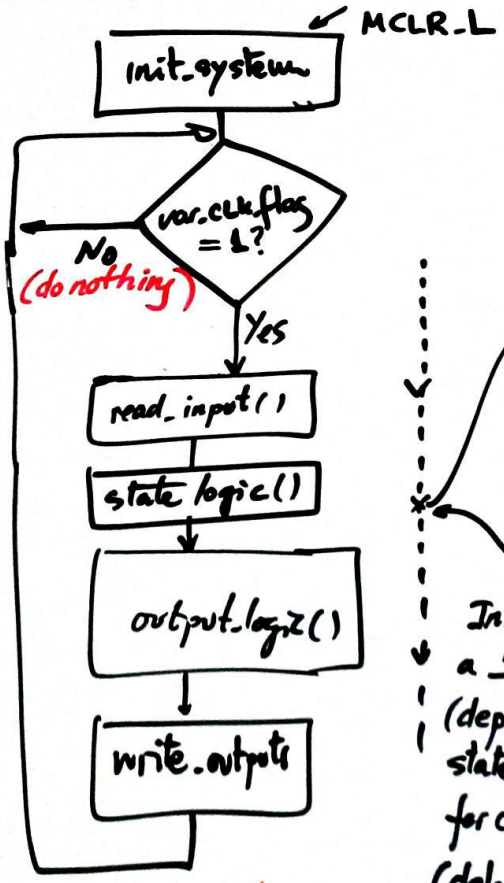
TRISC: Ø Ø Ø Ø Ø Ø Ø Ø (DA)

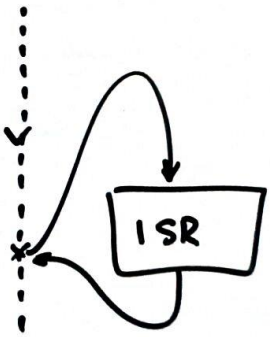
TRISD: Ø Ø Ø Ø Ø Ø Ø Ø

Not used pins → 'Ø' outputs

9. Software-hardware diagram and RAM variable of interest

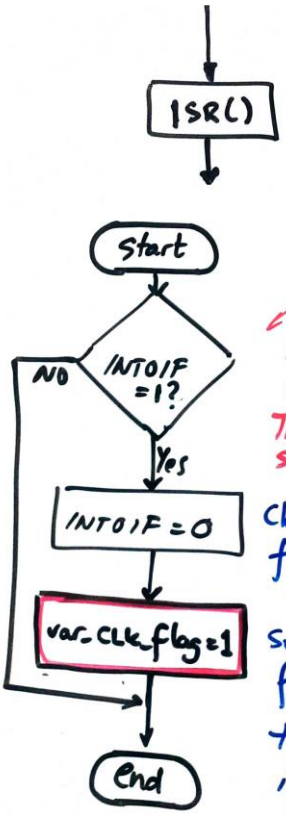






Interrupt when a ↓ is detected (depending on the state, the variables for counting interrupts (delays) and bits will be updated

(Functions can be optimised when executed only after a var. CLK_flag is set)



This external interrupt will be replaced by the TMR0IF to set the same RAM flag

Clear the hardware flag (RS_FF)

Set the RAM variable flag to execute the FSM functions in the main program when an INT0 is detected

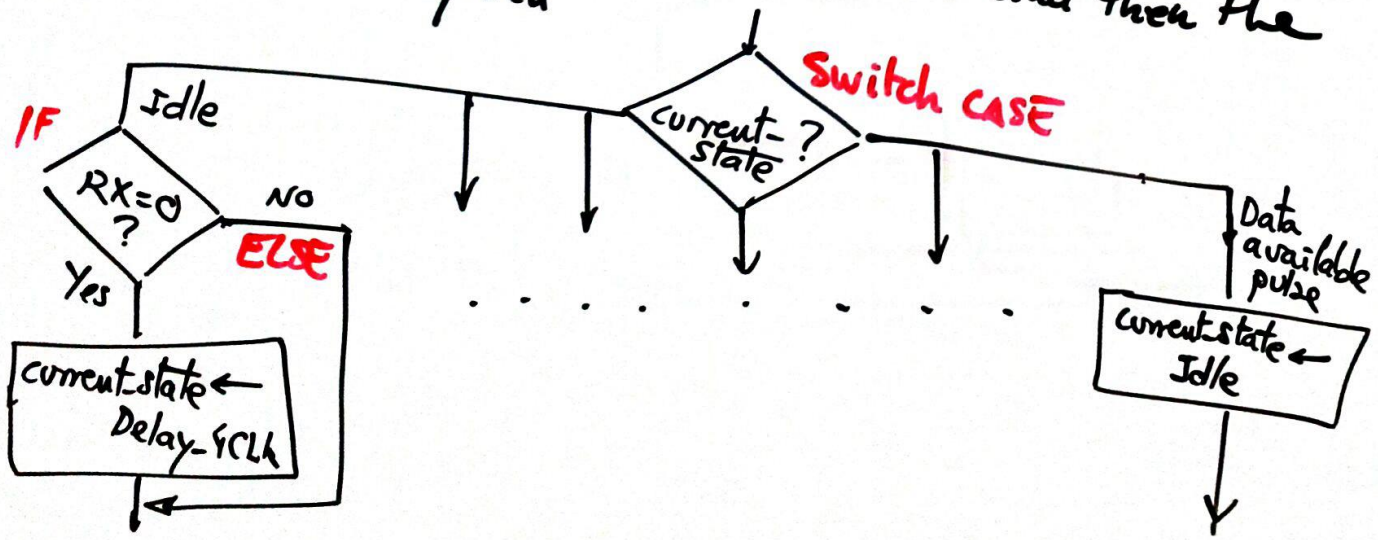
($T_s = \frac{1}{4.8\,kHz}$)

10. state logic truth table and flow chart

(RAM variables) (var_CLK_flag = 1 when ↑ is deleted) (next value)

RX	TC4D	TC8D	TC8B	OPC	current_state	current_state+
1	x	x	x	x	Idle	Idle
0	x	x	x	x	Idle	Delay_4CLK
x	0	x	x	x	Delay_4CLK	Delay_4CLK
x	1	x	x	x	Delay_4CLK	Sample_start_bit
0	x	x	x	x	Sample_start_bit	Delay_8CLK
1	x	x	x	x	Sample_start_bit	Idle
x	x	0	x	x	Delay_8CLK	Delay_8CLK
x	x	1	x	x	Delay_8CLK	Sample_data
					etc.	

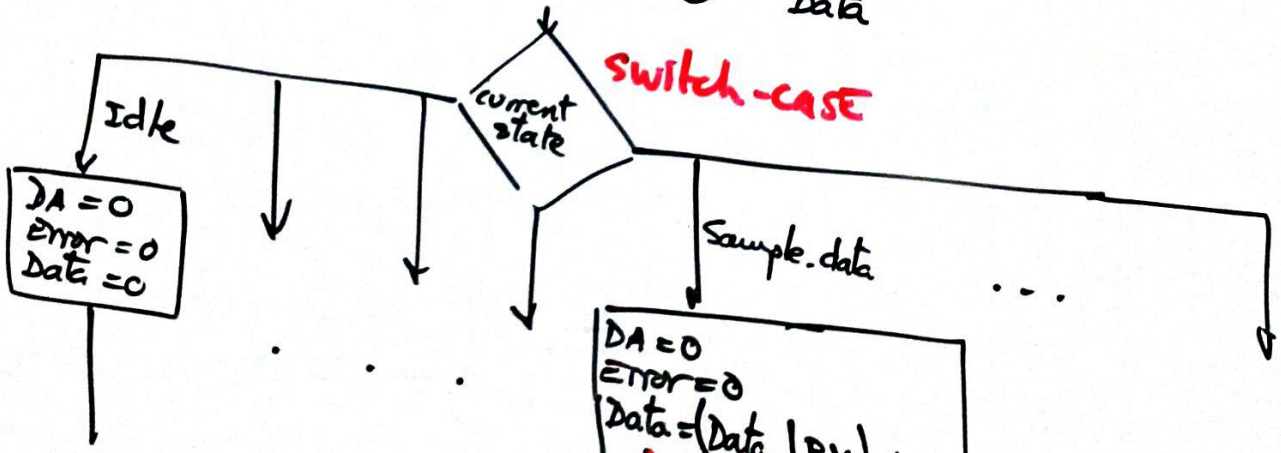
Thus, the flow chart is using switch-case and then the if-else when required



11. output logic truth table and flowchart

Depends on current_state and RX (only in some states to be able to shift data left)

RX	current_state	(RAM variables) DA	ERROR	Data	
x	Idle	0	0	0	
x	Delay_4CLK	0	0	0	
x	Sample_startbit	0	0	0	
x	Delay_8CLK	0	0	0	
RX	Sample_Data	0	0		Load RX, shift
RX	Odd parity_check	0	0		Load RX, shift, calculate OPC
RX	Stop_bit check	0	0		Data
x	Error_pulse	0	1	0	
x	Data_available_pulse	1	0		Data

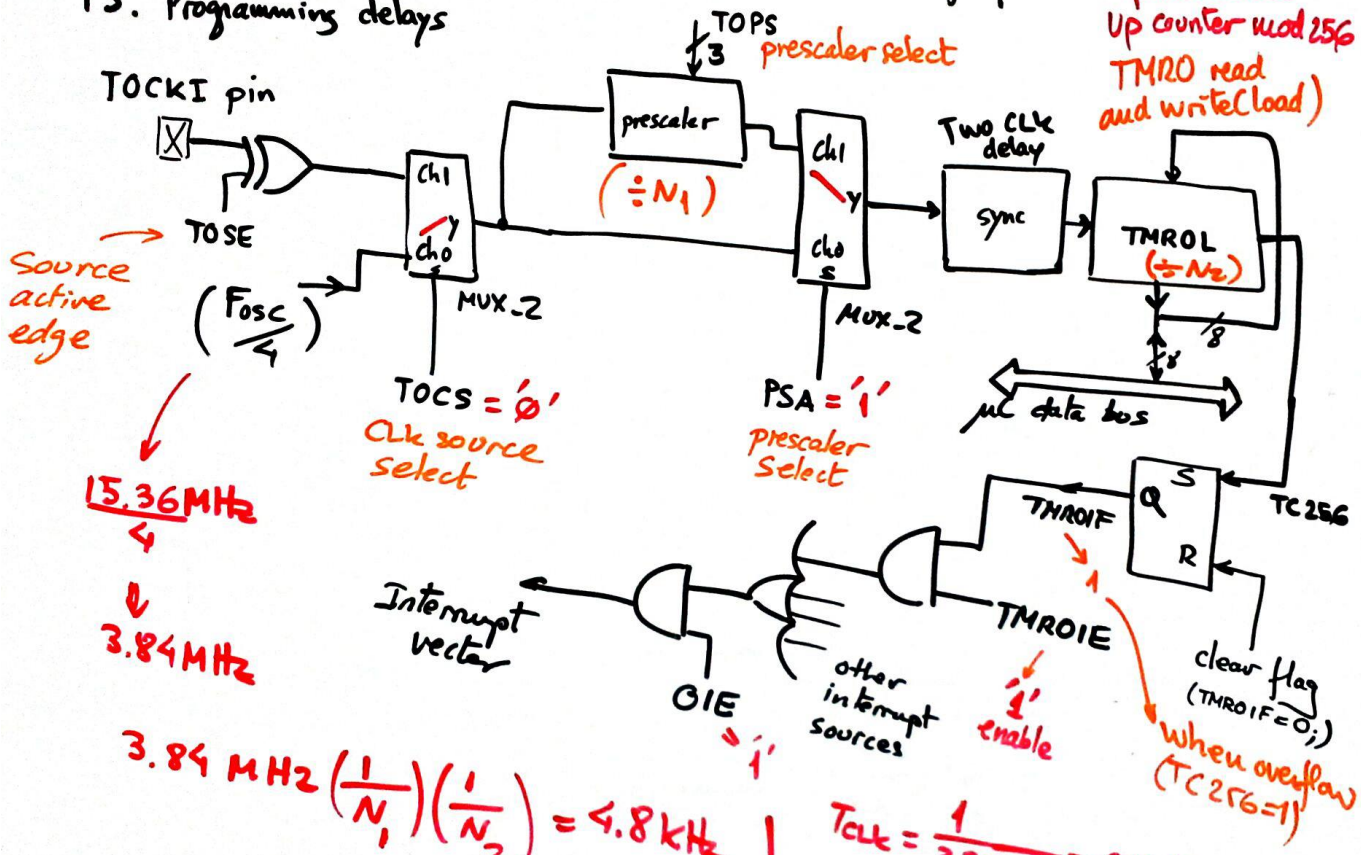


⇒ use the watch window and step by step

Load and shift left in C language

12. Using the Timer0 to generate baud rate frequencies

13. Programming delays



$3.84\,MHz\left(\frac{1}{N_1}\right)\left(\frac{1}{N_2}\right) = 4.8\,kHz$

$N_1 \cdot N_2 = 800$

For instance: $N_1 = 16$; $N_2 = 50$

⇨ This peripheral is acting as the datapath and the CLK generator in this application (timing delays)

$T_{CLK} = \frac{1}{3.84MHz} = 260.417\,ns$

$TC4D = 833.3\,ms = N_1 \cdot N_2 \cdot T_{CLK}$

$4 \cdot \left(\frac{1}{4.8kHz}\right)$

3200

$N_1 = 64$; $N2 = 50$

$TC8D = 1.67\,ms = N_1 \cdot N_2 \cdot T_{CLK}$

$N_1 = 64$

$N2 = 100$

6400