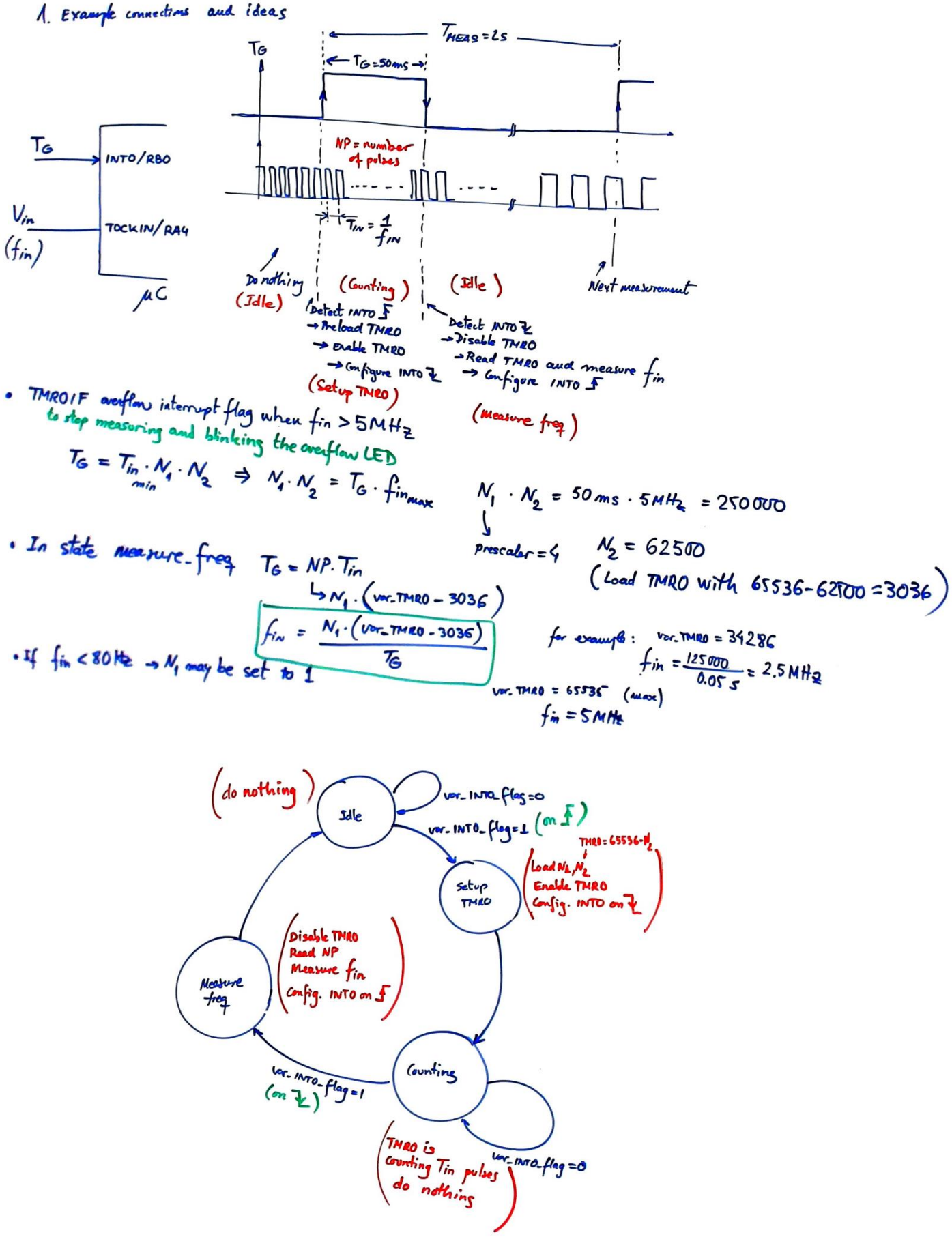
EXAM 2. Solution ideas and references

1. Frequency meter basic diagram. Example project [1-digit BCD counter with LCD and TMR0] on counting external pulses.



2. P11 on using the LCD as a design phase #2. Two design steps are possible: (1) printing simple static ASCII text, (2) printing numerical dynamic data

3. How to use TMR2 as timer counting internal pulses from the time base T_{OSC} derived from the microcontroller oscillator.

TMR2 hardware can count up to 16 · 256 · 16 = 65536 pulses before overflow and generate interrupt TMR2IF.

Because F_{OSC} = 12 MHz, we need another post-scaler to reach TG = 50 ms;

For example N1 = 16; N2 = 125 ; N3 = 5 (Interrupts every 3.33 ms) ; N4 = 15 (*var_soft_postscaler_N4* char type). In this way we can replace *var_INT0_flag* by for instance *var_COUNT_flag* to enable TMR0 counter (after T_G = 50 ms), and also to disable it after the same timing period.

If the external signal T_G has to be replaced by timings from TMR2, the 2 s measurement period (T_{MEAS}) can also be obtained counting TMR2IF interrupts: T_{MEAS} = 2 s = 40 · T_G

This frequency meter is a good example project for developing and testing because the three design phases are specified.

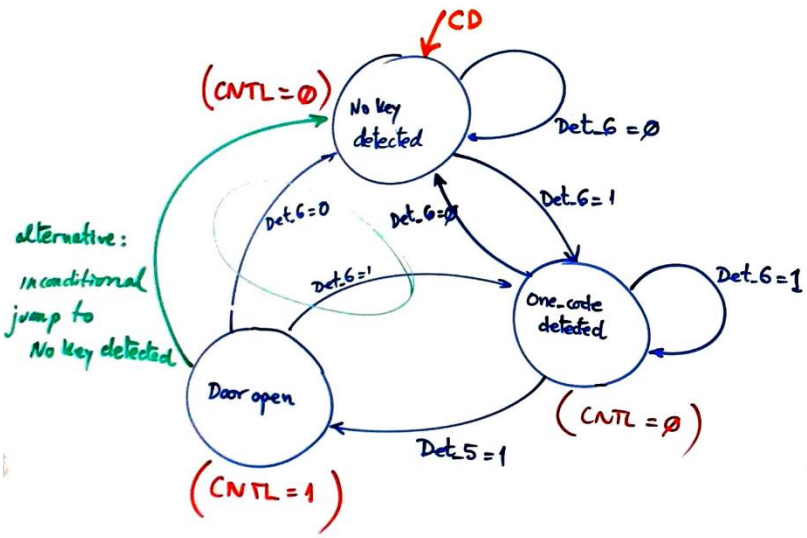
4. How to poll switches is explained in this lecture L9.3. This is the task assigned to the function *read_inputs()* executed in the main loop.

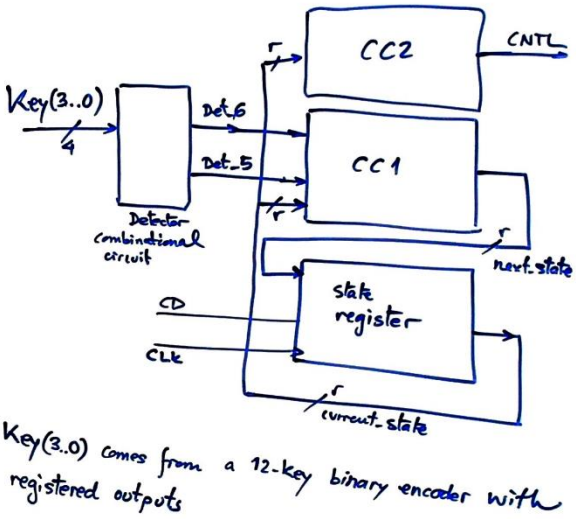
5. How to write output pins is explained in this lecture L9.4. This is the task assigned to the function *write_outputs()* executed in the main loop.

6. The BCD code, why is different from binary radix-2. Symbol. Example timing diagram. Truncating counters and expanding counters are concepts developed in this lecture L7.3, and proposed for instance in the highlighted project P7 on the *Hour_counter* (or up/down counter BCD modulo 24).

7. Two possible alternative designs. Examining the timing diagram, we see that the Key(3..0) vector is generated from a 12-key binary encoder with registered outputs that keep the last key pressed down (for example in this D2.15).

A *hardwired* standard P6 FSM application. Changing the secret code represents resynthesizing the full project. The state diagram is similar to the D2.4 pattern generator example.





A standard P8 dedicated processors including a datapath for solving comparison applications. The secret code is *information* easily saved and modified using switches. In the datapath two chained 4-bit data registers save the last two *Key* sampled values, and a comparator with “65” generates a status signal to control state transitions. This system is easy upgradable to 4-digit or longer secret key. This is an example project D2.8.

8. CLK_generator circuits are explained in lecture L8.2. Adapt the frequency dividers to the desired values.

9. Chaining shift registers is explained in this P7 unit.

10. This is the circuit modelled in Proteus (analysis method 2). Asynchronous_circuit.pdsprj. And how to solve such circuits in paper (analysis method 1) is explained in the many examples in P5 of flip-flops. The key idea is to determine the sampled values at the flip-flop control inputs on the CLK rising edges, to be able to infer the outputs using the corresponding function table.