UPC. EETAC. Bachelor's degree. 2A. Digital Circuits and Systems (CSD). **Exam 2.** Grades available by 18th January.
Questions and revision: January 20th, 10:00 – 12:00, office C4-103P. **14th January**

(Problem 1) and (Problem 2) and [(Problem 3) or (Problem 4) or (Problem 5)]

NOTE for all problems and questions: firstly, **draw your circuits**, sketches and diagrams; explain your plan and what you do. Justify your results.

Problem 1. (3p)

Analyse the circuit in Fig. 1 by means of a timing diagram, naming all the signals of interest and indicating sampled values on the CLKs' rising edges of interest.

a) Find the binary codes (numbers) generated at the output vector W(2..0). Use several steps to deduce the solution, for instance, you can start solving only the outputs from Chip0.

b) Discuss what will be the circuit's CLK maximum frequency of operation if *flip-flop* components are implemented internally as standard FSM. Suppose that the typical CLK to output propagation time (t_{CO}) for the state register *D_FF* is 4.3 ns. Typical propagation delay (t_P) of a generic logic gate is 2.8 ns.

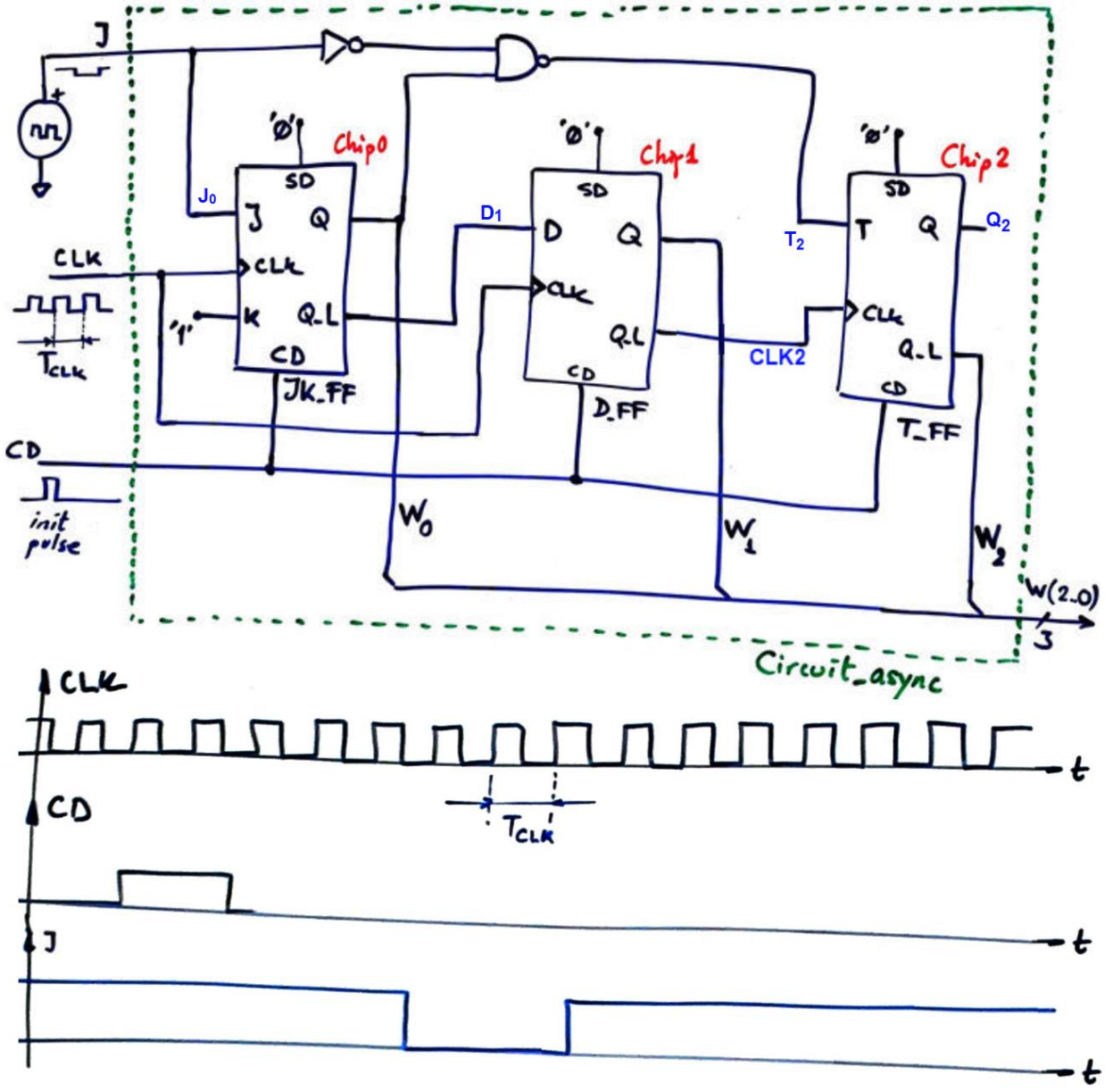


Fig. 1. Circuit based on flip-flops and example stimulus waveforms for fixing a test bench. At the end of the exam (Fig. 7) you can find flip-flop types and their function tables.

Problem 2. (3p)

Invent a synchronous binary radix-2 down counter module 21 (*Counter_mod21*) with count enable (**CE**) and terminal count (TC21) using **plan C2** and standard components such *Counter_mod16* (Fig. 2b) and other circuits if necessary. Use count expansion and count truncation techniques.

The circuit has to generate the 21 following binary radix-2 codes at **Q**(4..0) outputs when counting down continuously (with **CE** = '1'):

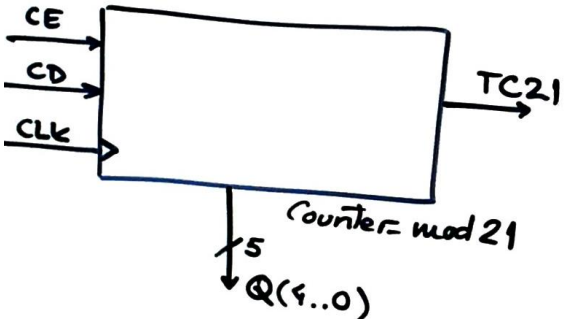
…25, 24, 23, **11**, **10**, **9**, **8**, **7**, **6**, **5**, **4**, **3**, **2**, **1**, **0**, **31**, **30**, **29**, **28**, **27**, **26**, **25**, **24**, **23**, 11, 10, …

down counting →

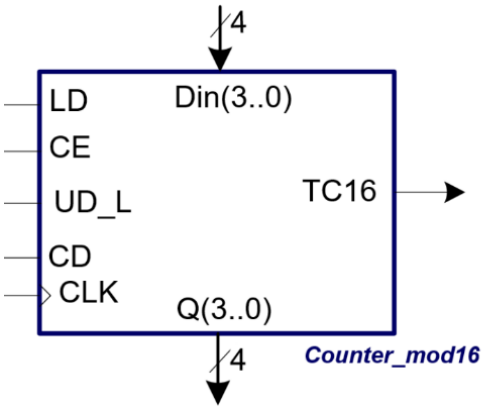
Terminal count ↑

a) Specifications: symbol (Fig. 2a), function table, timing diagram, state diagram.

b) Plan and develop the circuit in steps. How many *D_FF* will include the *Counter_mod21* design? How many VHDL files will include the project?



a)



LD	CE	UD_L	Q⁺	Synchronous operation after the CLK's rising edge
1	x	x	Din	Parallel load (register data)
0	0	x	Q	Do nothing (inhibit)
0	1	1	$(Q+1)_{mod16}$	Up counting in binary
0	1	0	$(Q-1)_{mod16}$	Down counting in binary

TC16 = '**1**' when CE = '1' *and* [(Q = 15 *and* UD_L = '1') *or* (Q = 0 *and* UD_L = '0')]; '0' otherwise

b)

Fig. 2a) Symbol of the counter to be designed. b) Symbol of the 4-bit standard universal binary radix-2 counter component (*Counter_mod16*) and its function table.

Problem 3. Water tank controller (4p)

We want to design a water tank controller (*Tank_cntl*) as an application of FSM to drive two pumps independently, as represented in Fig. 3. The tank has level sensors D_1, D_2, and D_3 attached to the wall, so that a '1' is generated when the sensor is sunk into water. The controller works as follows: when the tank is empty, below D_1, both pumps work simultaneously; when the water level is above D_2 pump P_1 stops; when the water is above D_3, meaning that the tank is full, pump P_2 stops; and finally, the pumps do not switch on until the water level is again below D_1.

Our goal is to design the circuit using two different technologies and thus be able to compare how they perform:

(1) FPGA and VHDL (2) Microcontroller and C language.

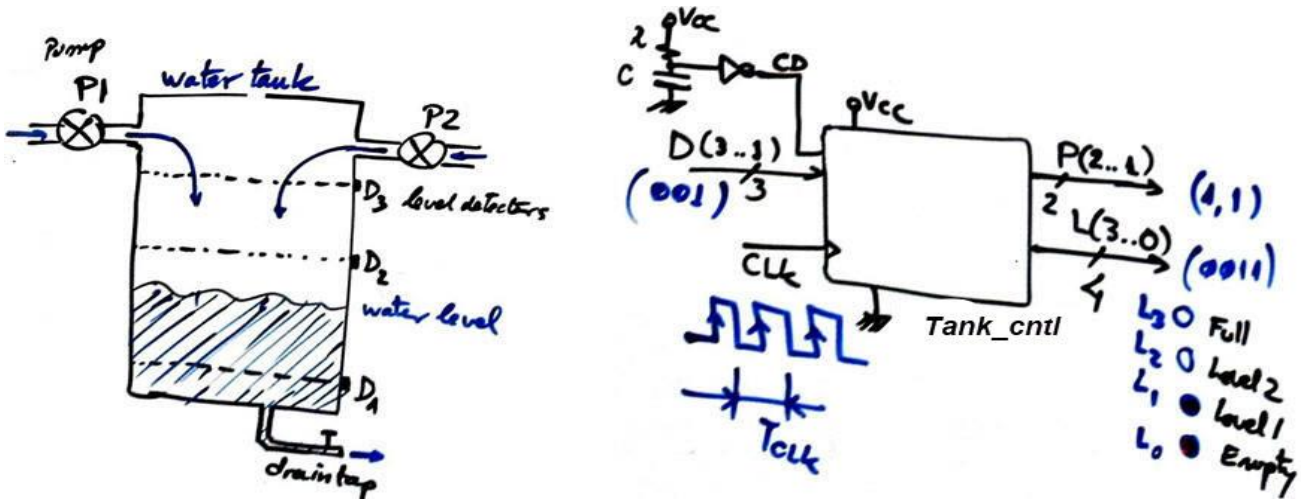


Fig. 3. Symbol and idea of the water tank controller. We will use a 1.25 kHz CLK.

Section 1. Hardware design (FPGA, VHDL).

a) Propose a FSM state diagram indicating both, state transitions and outputs.

b) Draw the FSM structure. Draw the state register schematic using *D_FF*. Code the internal states in binary radix-2.

c) Propose the CC1 and CC2 truth tables and their flowcharts ready for VHDL translation.

d) Design and add to the tank controller a *CLK_Generator* circuit to obtain the 1.25 kHz system CLK from an external 16 MHz crystal oscillator. How many *D_FF* will contain the full design *Tank_cntl_top*?

Section 2. Microcontroller design.

e) Draw a schematic to implement the application using a PIC18F4520 microcontroller indicating what port pins to use. Draw the power-on reset (MCLR_L) circuit and explain how it works. Draw the oscillator circuit using a 16 MHz quartz crystal. An external f_{CLK} = 1.25 kHz will be used to sample the sensors.

f) Adapt the state diagram so that it can be driven by interrupt flags. Organise the main program as a FSM and draw the hardware-software diagram indicating the required RAM variables.

g) Explain how to configure *init_system*() and the *ISR*().

h) Replace the external CLK signal by the internal TMR2 (diagram from datasheet in Fig. 6 at the end of exam).

Problem 4. LED rotator (4p)

Our aim is to implement a LED rotator like the one represented in Fig. 4, which is a FSM acting as an 8-bit one-hot shift register that runs continuously. The system operates by means of clicking several times the same push-button PB:

- First click: shift and rotate right at 5 Hz
- Second click: shift and rotate left at 5 Hz
- Third click: all LED off

Our goal is to design the circuit using two different technologies and thus be able to compare how they perform:

(1) FPGA and VHDL (2) Microcontroller and C language.

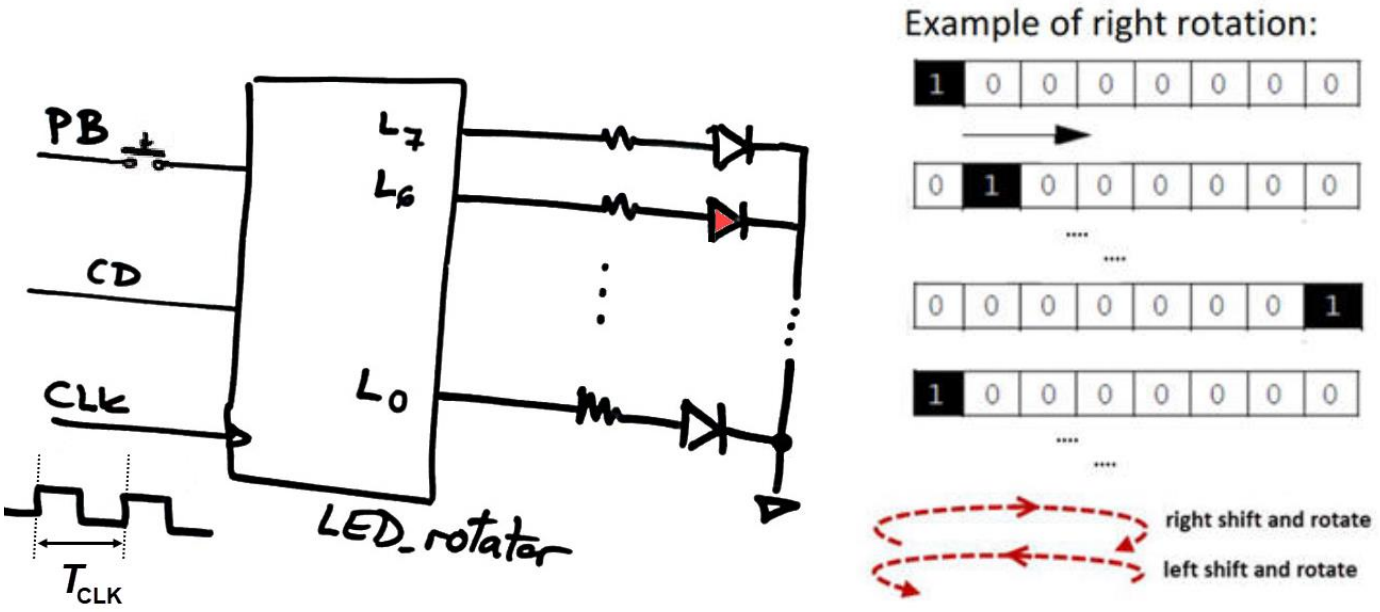


Fig. 4. Basic LED rotator symbol and idea.

Section 1. Hardware design (FPGA, VHDL).

a) Propose a plan C2 architecture for the circuit using components such FSM, *Shift_reg_8bit* and *CLK_Generator*. How many VHDL files are required to solve this project?

b) Draw the FSM state diagram and its internal architecture. Draw its state register schematic using *D_FF*. Encode the internal states in one-hot.

c) Propose the CC1 and CC2 truth tables and their flowcharts.

d) Design the *CLK_Generator* circuit to obtain from a 13 MHz crystal oscillator both, the f_{CLK} = 250 Hz signal for running the FSM and the 5 Hz square waveform for shifting operations. How many *D_FF* will contain the complete *LED_rotator_top*?

Section 2. Microcontroller design.

e) Draw a schematic to implement the *LED_rotator* using a PIC18F4520 microcontroller indicating what port pins to use. Represent the power-on reset (MCLR_L) circuit and explain how it works. Draw the oscillator circuit using a 13 MHz quartz crystal.

f) Adapt the state diagram so that it can be driven by interrupt flags. Do the interrupts have to be enabled all the time? Explain the *ISR*().

g) Draw the hardware-software diagram indicating the required RAM variables and how the FSM is solved in software. Organise the main program as a FSM in our CSD way.

h) Replace the external CLK signal by the internal TMR2 so that the FSM works at the same 250 Hz frequency. How the 5 Hz flag variable can be generated using the same TMR2? Datasheet in Fig. 6.

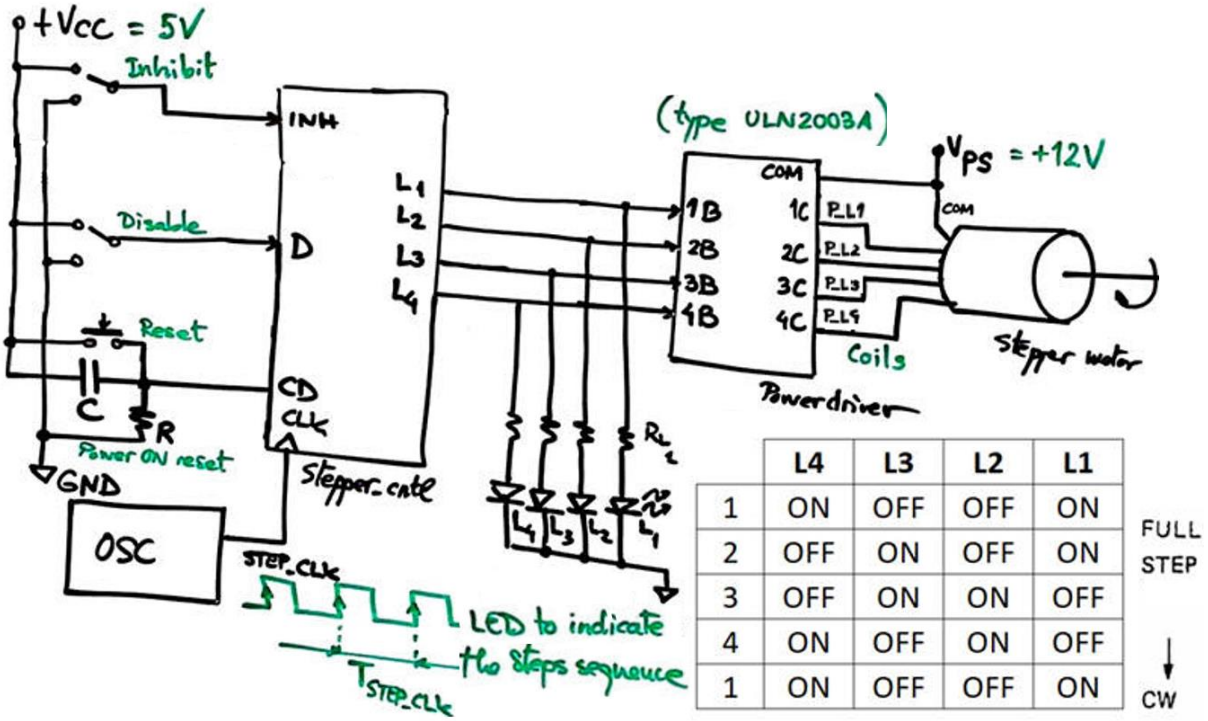
Problem 5. Stepper motor controller (4p)

We have to design the digital control unit (*Stepper_cntl*) for the "9904 112 31004" stepping motor from Premotec shown in Fig. 5 following our FSM strategies.

Step or stride angle is 7.5 degree for this model, thus 48 CLK periods are required for a full revolution. We can start supposing an external CLK frequency of 96 Hz, thus, when running it rotates at two revolutions per second.

We can imagine a "disabled" state when switching D = '1' at which all coils are OFF. The motor is not energised and thus the shaft moves freely.

On the other hand, inhibit control input, when activated INH = '1', acts like a count disable preventing the motor moving from its current state.



	L4	L3	L2	L1	
1	ON	OFF	OFF	ON	FULL STEP
2	OFF	ON	OFF	ON	
3	OFF	ON	ON	OFF	
4	ON	OFF	ON	OFF	
1	ON	OFF	OFF	ON	CW

Fig. 5. Example of two-phase stepper motor: characteristics, connections, full wave steeping sequence and unipolar winding. A power driver chip is required to energise the coils with nominal currents and voltages.

Our goal is to design the circuit using two different technologies and thus be able to compare how they perform:

(1) FPGA and VHDL (2) Microcontroller and C language.

Section 1. Hardware design (FPGA, VHDL).

a) Propose the FSM state diagram indicating both, state transitions and outputs.
b) Draw the FSM structure. Draw the state register schematic using *D_FF*. Code the internal states in Gray.
c) Propose the CC1 and CC2 truth tables and their flowcharts ready for VHDL translation.
d) Design and add to the controller a *CLK_Generator* circuit to obtain a step CLK of 96 Hz from an external 12 MHz crystal oscillator. How many *D_FF* will contain the full design *Stepper_cntl_top*?

Section 2. Microcontroller design.

e) Draw a schematic to implement the *Stepper_cntl* using a PIC18F4520 microcontroller indicating what port pins to use. Draw the power-on reset (MCLR_L) circuit and explain how it works. Draw the oscillator circuit using a 12 MHz quartz crystal.

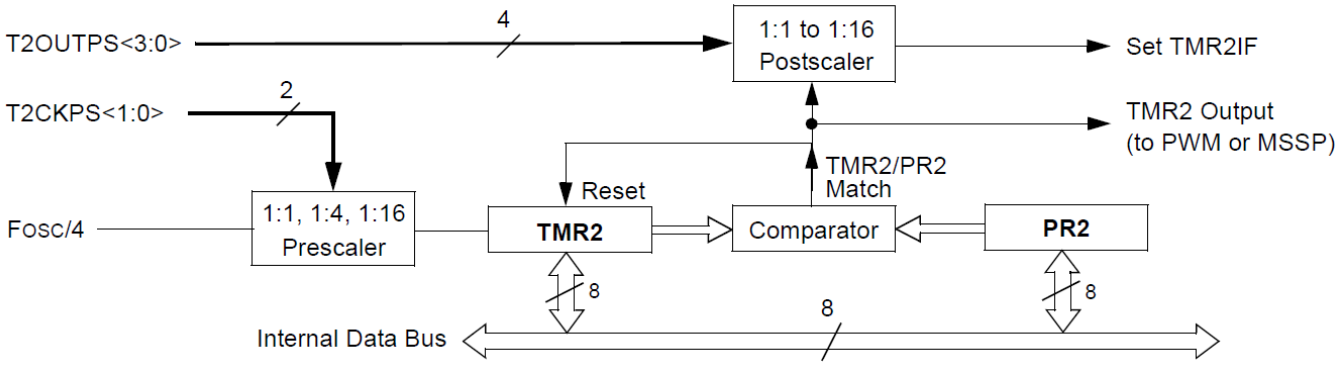
f) Adapt the state diagram so that it can be driven by interrupt flags. Explain the *ISR*().

g) Draw the hardware-software diagram indicating the required RAM variables and how the FSM is solved in software. Organise the main program as a FSM in our CSD way.

h) Replace the external CLK signal by the internal TMR2 so that the machine works in the same way and configure its parameters (datasheet in Fig. 6).

DATASHEETS

TMR2 diagram from Microchip datasheet:



REGISTER 13-1: T2CON: TIMER2 CONTROL REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **Unimplemented:** Read as '0'

bit 6-3 **T2OUTPS<3:0>:** Timer2 Output Postscale Select bits

0000 = 1:1 Postscale
0001 = 1:2 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON:** Timer2 On bit

1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS<1:0>:** Timer2 Clock Prescale Select bits

00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

Fig. 6. TMR2 block diagram from Microchip datasheet.

Flip-flops

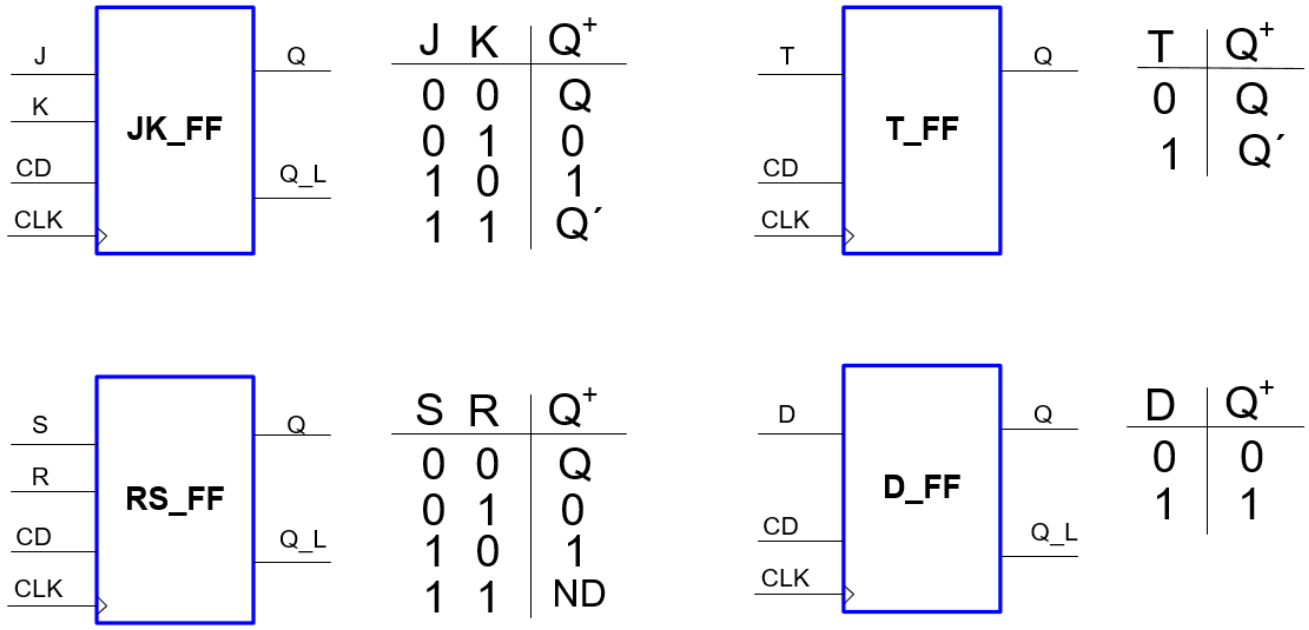


Fig. 7. *Flip-flop* symbols and their functions tables.