MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN Federation
Federal State Budgetary Educational Institution of Higher Education
"Kazan National Research Technical University named after A. N. Tupolev-KAI"
(KNRTU-KAI)
Institute of Computer Technologies and Information Security
Department of Computer Systems

Report № 6

«Proteus Virtual System Modeling (VSM)
PART I. TMR0 Application Counter Using TMR0.
PART II. EEPROM Memory Application »

«Architecture of embedded systems»

Student 4167 Edwin G. Carreno
(group number) (signature, data) (full name)

Associate professor of the computer systems' department Daria V. Shirshova

Grade _____

_____
(signature, data)

Kazan 2021

Content

1. Part I. TMR0 Application Counter Using TMR0

TMR0

a. Write an Assembly program to make a counter using TMR0; the counter should increment its value on every 2 pushbuttons on RA4.

```
;=====================================================================
; Main.asm file generated by New Project wizard
;
; Created:   Sun Apr 22 2021
; Processor: PIC16F84A
; Compiler:  MPASM (Proteus)
;=====================================================================

;=====================================================================
; DEFINITIONS
;=====================================================================

#include p16f84a.inc                ; Include register definition file

;=====================================================================
; VARIABLES
;=====================================================================

;=====================================================================
; RESET and INTERRUPT VECTORS
;=====================================================================

      ; Reset Vector
RST   code  0x0
      goto  Start

;=====================================================================
; CODE SEGMENT
;=====================================================================

PGM   code
Start
      BCF       STATUS, RP0   ; Selecting bank 1
      CLRF      PORTA         ; Clearing register PORTA
      CLRF      PORTB         ; Clearing register PORTB
      CLRF      TMR0          ; Clearing register TMR0

      BSF       STATUS,   RP0  ; Selecting bank 0
      MOVLW     0x10
      MOVWF     TRISA          ; Setting PORTA.4 as Input
      MOVLW     0xF0
      MOVWF     TRISB          ; Setting PORTA[3:0] as Output
      MOVLW     0x20
      MOVWF     OPTION_REG     ; Choosing a prescaler 1:2
      BCF       STATUS, RP0    ; Selecting bank 1
```

```
Loop
      MOVFW      TMR0
      MOVWF      PORTB            ; Transfering value of TMR0 to PORTB
      goto  Loop

;=====================================================================
END
```

b. Simulate the program using the circuit shown in figure via Proteus software. Verify it operates properly when simulated.

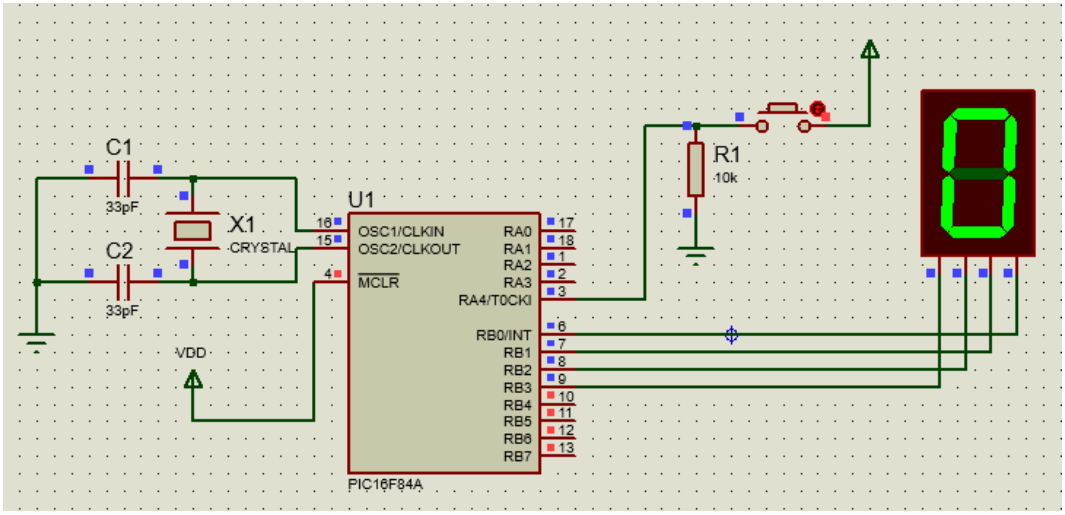


Figure 1. Counter using TMR0 and prescale 1:2. The pushbutton has been pressed 0 times.

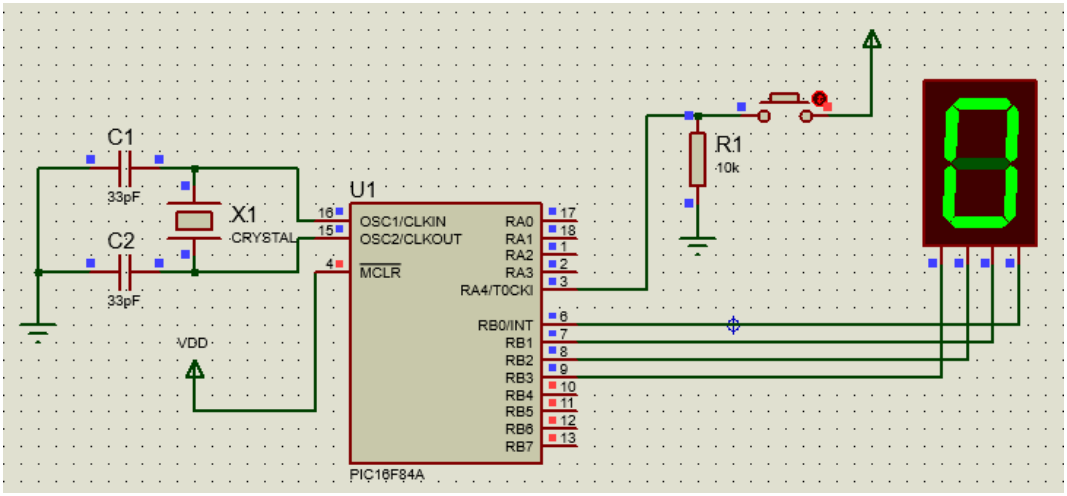


Figure 2. Counter using TMR0 and prescaler 1:2. The pushbutton has been pressed 1 time.

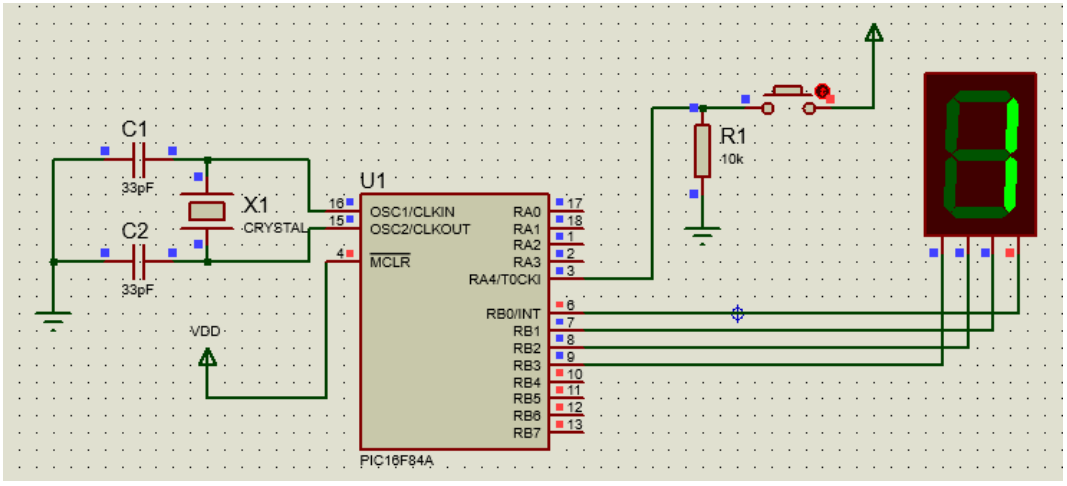


Figure 3.Counter using TMR0 and prescale 1:2. The button has been pressed 2 times.

c. Program a PIC16F84A using the QL2006 programmer.

d. Build the circuit using the programmed PIC16F84A and the observe its operation. Demonstrate the circuits operation to the instructor.

Watchdog Timer (WDT)

a. Write an Assembly program to make a counter using WDT; the counter should increment its value on every one single pushbutton on RA4, and also counts from 0 to 99.

```
;===================================================================
; Main.asm file generated by New Project wizard
;
; Created:   Sun Apr 22 2021
; Processor: PIC16F84A
; Compiler:  MPASM (Proteus)
;===================================================================

;===================================================================
; DEFINITIONS
;===================================================================

#include p16f84a.inc                ; Include register definition file

;===================================================================
; VARIABLES
;===================================================================
CBLOCK 0x20
      CACHETIMER
      TENS
 ENDC

;===================================================================
; RESET and INTERRUPT VECTORS
;===================================================================

      ; Reset Vector
RST   code  0x0
      goto  Start

;===================================================================
; CODE SEGMENT
;===================================================================

Start:
   BCF            STATUS, RP0       ; Select Bank 0
   CLRF           PORTA             ; Initialize PORTA
   CLRF           PORTB             ; Initialize PORTB
   CLRF           TMR0              ; Initialize TMR0

   BSF            STATUS, RP0       ; Select Bank 1
   MOVLW          0x10              ; Set RA[3:0] as outputs and RA4 as input
   MOVWF          TRISA

   MOVLW          0xF0              ; Set RB[3:0] as outputs
   MOVWF          TRISB
   MOVLW          0x78              ; Configure TMR0 to be used by the watchtimer
   MOVWF          OPTION_REG

   BCF            STATUS, RP0       ; Select Bank 0

Loop:

   MOVFW          TMR0              ; Capturing TMR0
   MOVWF          CACHETIMER        ; Save TMR0 to CACHETIMER
   MOVLW          0x00
   MOVWF          TENS

counting
   MOVLW          0x0A              ; Subtract 10 from 0x0C
   SUBWF          CACHETIMER, F
   BTFSS          STATUS, C  ; Display "tens" and "units" if "units" < 0
   GOTO           display
   INCF               TENS, F ; Increment "tens" and repeat
   GOTO           counting

display
   MOVLW          0x0A              ; Adjust "units" (units < 0)
   ADDWF          CACHETIMER, F
```

```
    MOVFW           TENS            ; Display "tens"
    MOVWF           PORTA
    MOVFW           CACHETIMER      ; Display "units"
    MOVWF           PORTB
    GOTO Loop
;*******************************************************************
END
```

b. Simulate the program using the circuit shown in figure via Proteus software. Verify it operates properly when simulated.

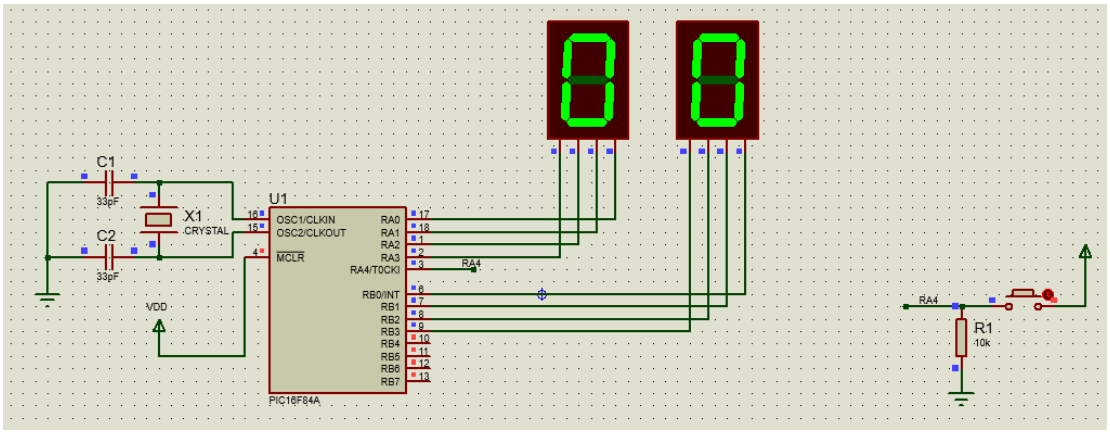


Figure 4. Counter using WDT. The pushbutton has been pressed 0 times.

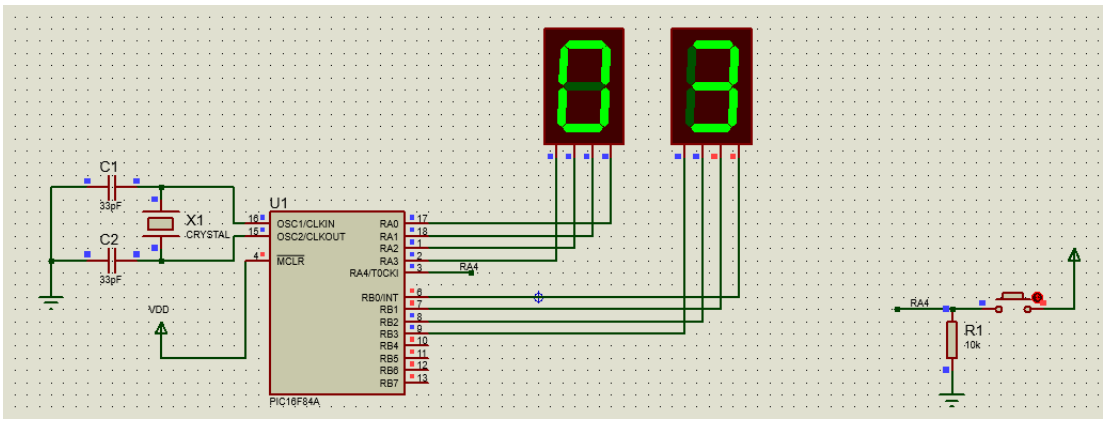


Figure 5. Counter using WDT. The pushbutton has been pressed 3 times.

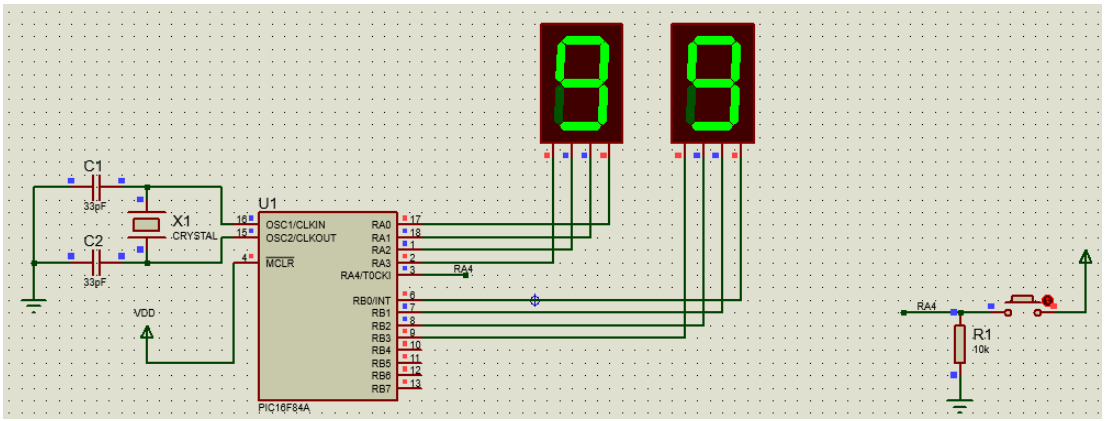


Figure 6. Counter using WDT. The pushbutton has been pressed 99 times.

c. Program a PIC16F84A using the QL2006 programmer.

d. Build the circuit using the programmed PIC16F8A and then observe its operation. Demonstrate the circuits operation to the instructor. Present your results in a lab report including a copy of the source codes.

2. EEPROM Memory Application

a. Write an assembly program to fill all the EEPROM Memory locations with 7. Hint: Build an external Macro called EEPROM_WRITE and EEPROM_READ takes two parameters the data and the address to achieve the writing and reading ; then call it in the main program.

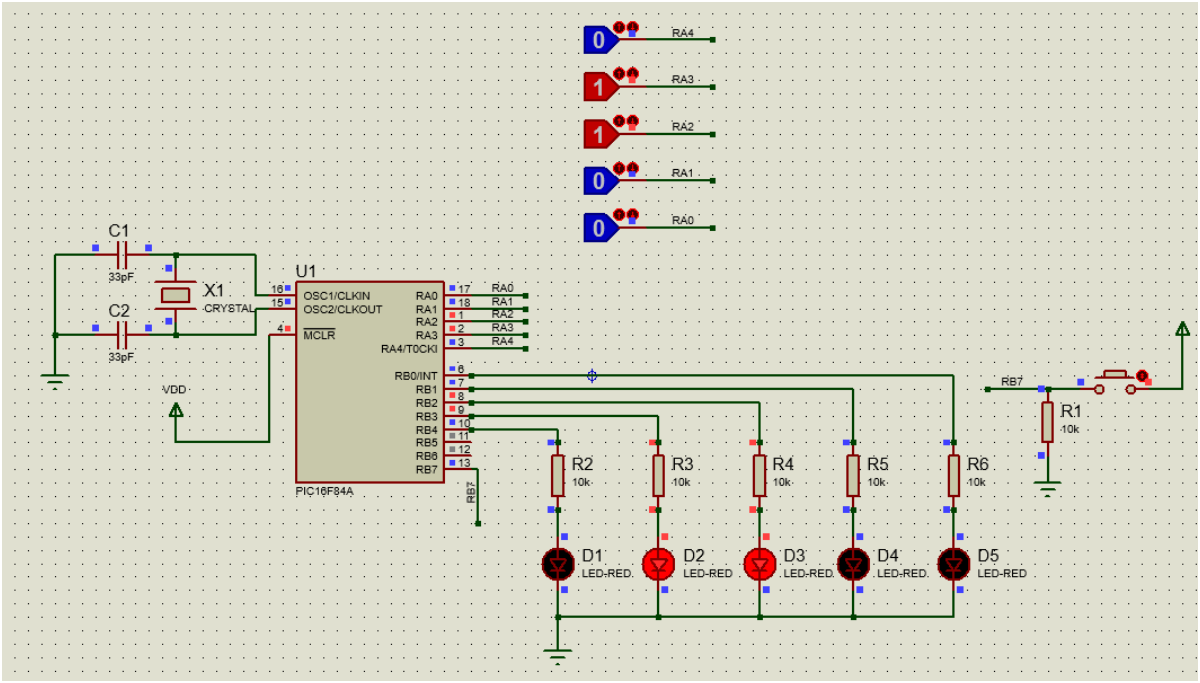
```
;=====================================================================
; Main.asm file generated by New Project wizard
;
; Created:   Sun Apr 22 2021
; Processor: PIC16F84A
; Compiler:  MPASM (Proteus)
;=====================================================================

;=====================================================================
; DEFINITIONS
;=====================================================================

#include p16f84a.inc                ; Include register definition file

;=====================================================================
; VARIABLES
;=====================================================================
 CBLOCK 0x20
      MEMADDRESS
      MEMDATA
 ENDC

;=====================================================================
; RESET and INTERRUPT VECTORS
;=====================================================================

      ; Reset Vector
RST   code  0x0
      goto  Start

;=====================================================================
; CODE SEGMENT
;=====================================================================

Start:
                    BSF STATUS, RP0           ; Selecting Bank 1
                    MOVLW      0x1F           ; Setting RA[4:0] as inputs
                    MOVWF      TRISA
                    MOVLW      0xE0           ; Setting RB[4:0] as outputs
                    MOVWF      TRISB

                    BCF STATUS, RP0           ; Selecting Bank 0

EEPROM_READ MACRO ADDRESS, INFO
                    BCF        STATUS, RP0
                    MOVF       ADDRESS, W            ; Setting EEPROM Address
                    MOVWF      EEADR
                    BSF        STATUS, RP0
                    BSF        EECON1, RD
                    BCF        STATUS, RP0
                    MOVF       EEDATA, W
                    MOVWF      INFO
                    endm

EEPROM_WRITE MACRO ADDRESS, INFO
                    BCF        STATUS, RP0
                    MOVF       ADDRESS, W
                    MOVWF      EEADR
                    MOVF       INFO, W
                    MOVWF      EEDATA
                    BCF        STATUS, RP0
                    BCF        INTCON, GIE    ; Disabling interruptions
```

```
                    BSF         STATUS, RP0
                    BSF         EECON1,WREN    ; Enable Write
                    MOVLW       0x55           ; Write 55h
                    MOVWF       EECON2
                    MOVLW       0xAA           ; Write AAh
                    MOVWF       EECON2
                    BSF         EECON1, WR

WRITE
                    BTFSS       EECON1, EEIF
                    GOTO        WRITE

                    BCF         EECON1, EEIF
                    BCF         STATUS, RP0
                    BSF         INTCON, GIE           ; Enabling interruptions
                    endm

Loop:
                    BCF STATUS, RP0
                    MOVLW       0x10                  ; Setting memory address
                    MOVWF       MEMADDRESS

                    MOVF        PORTA, W              ; Capturing Data to store
                    MOVWF       MEMDATA

                    EEPROM_WRITE       MEMADDRESS, MEMDATA   ; Writing invoke
                    EEPROM_READ        MEMADDRESS, MEMDATA   ; Reading invoke
                    MOVFW       MEMDATA
                    MOVWF       PORTB

                    GOTO  Loop
;===============================================================================
END
```

b. Write an assembly program to take the data existed on PORT A and display it on PORT B; first, the data must be taken from PORT A and stored in the EEPROM address location 0x10, and then be taken again from EEPROM and be displayed on PORT B.

- Done in the step *a*.

c. Simulate the program using the circuit shown in figure Proteus software. Verify it operates properly when simulated.



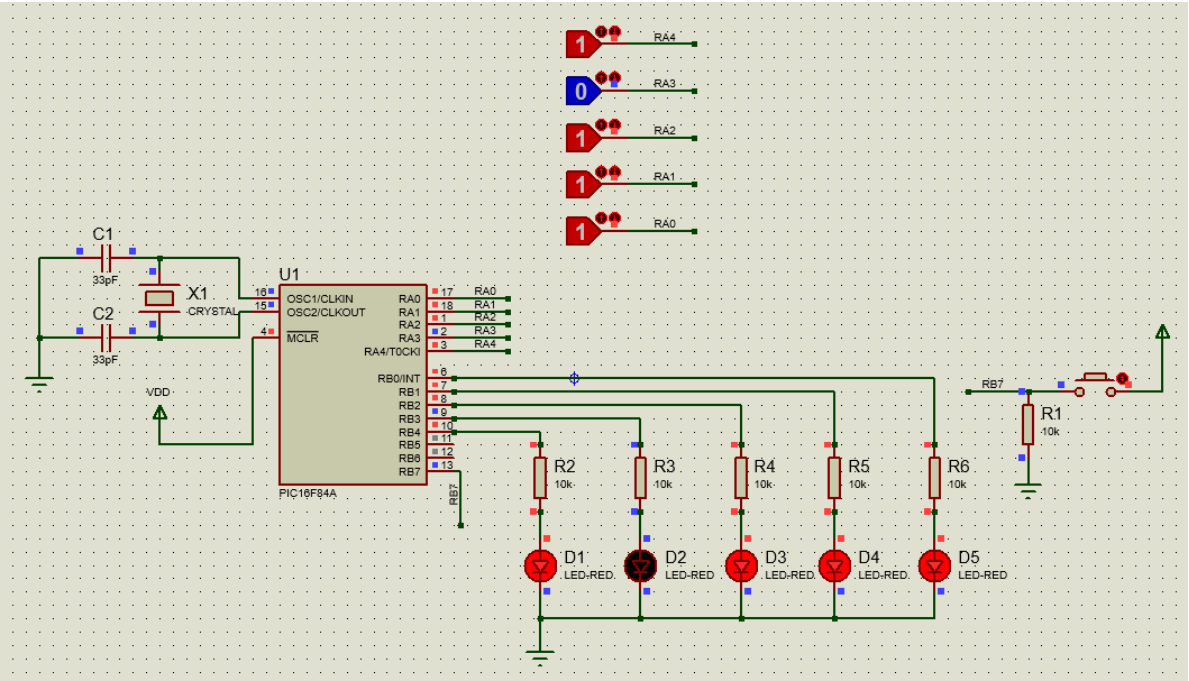


Figure 7. Performing R/W operations in the embedded EEPROM for the PIC16F84A.

3. Control Question

a. What is a timer?

It is a peripheral with capabilities of counting over time, restarting, and capturing rising/falling edges in external signals. Additionally, could interrupt the microcontroller CPU in order to perform actions that are a priority.

b. Why do we need timers?

In order to perform precise operations related to time. A timer allows to the microcontroller perform unattended operations and interrupt the processor in case it requires priority.

c. What is the Option Control Register?

It is the register that allows to the programmer to configure the TIMER embedded into the microcontroller. Some options you could find are prescaler selections bits and timer source.

d. What is EEPROM memory?

It is Non-volatile memory that allows to the microcontroller, in this case, to store data that could be critical for some applications such as settings in a medical device or the temperature in a vaccine refrigeration system that could suffer of constant electrical outages.

4. Summary

In this practice we continue with the usage of assembly language as a programming tool to configure microcontroller peripherals such as a timer. On the other hand, it is important to be careful with the appropriate usage of the Instruction Set for the microcontroller, sometimes Proteus do not inform errors and you as engineer do not has an easy starting point for debugging

your design. Always, you must have the datasheet as a reference.