18 Ottobre 2023

**2 Lezione Embedded**

Simulatore -> vedi se programma funziona

Sia B che A -> esite sia port che tris

Tris -> piedino I/O -> direzione filo

Port -> H/L -> valutare stato filo

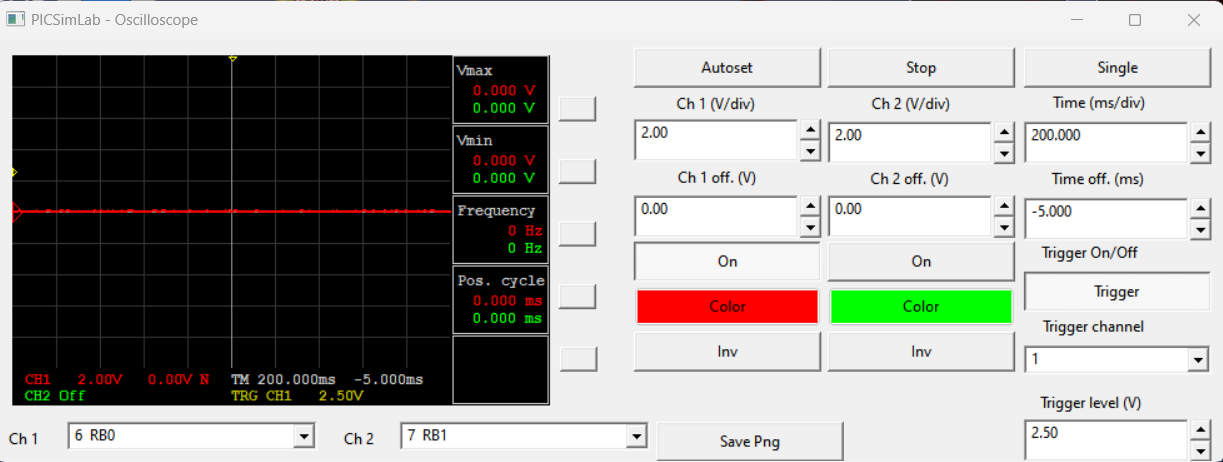
Tris -> 1 input

Tris -> 0 output

Port -> 1 High

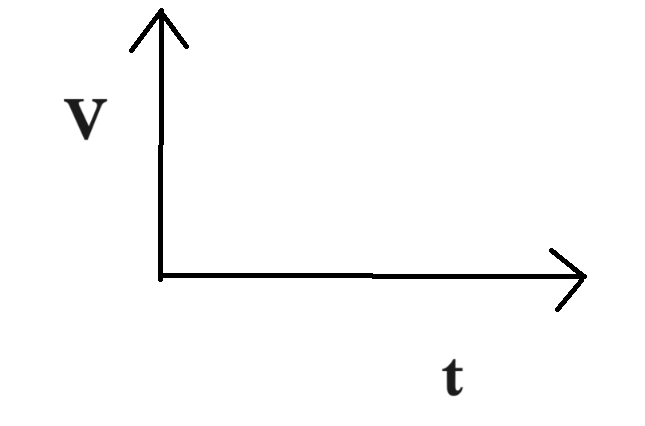
Port -> 0 Low

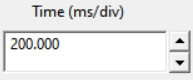
Oscilloscopio -> visualizza su schermo , proietta su schermo andamento di monitor in un determinato tempo -> sonda tensione si vende nel tempo -> dove cambia e come si capisce che cambia



Asse cartesiano

* x -> tempo
* y -> tensione V





Qui scala 200 ms

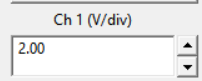
Oscilloscopio -> fa vedere 2 segnali alla volta

Dal menù a tendina -> scelgo cosa guardare

Colori segnali

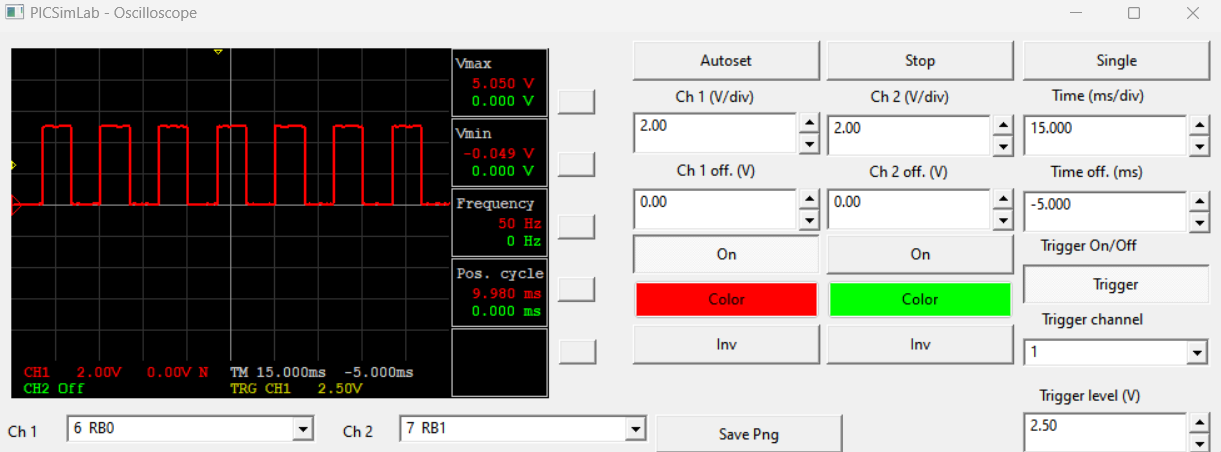


Una casella quel dire alta 2V

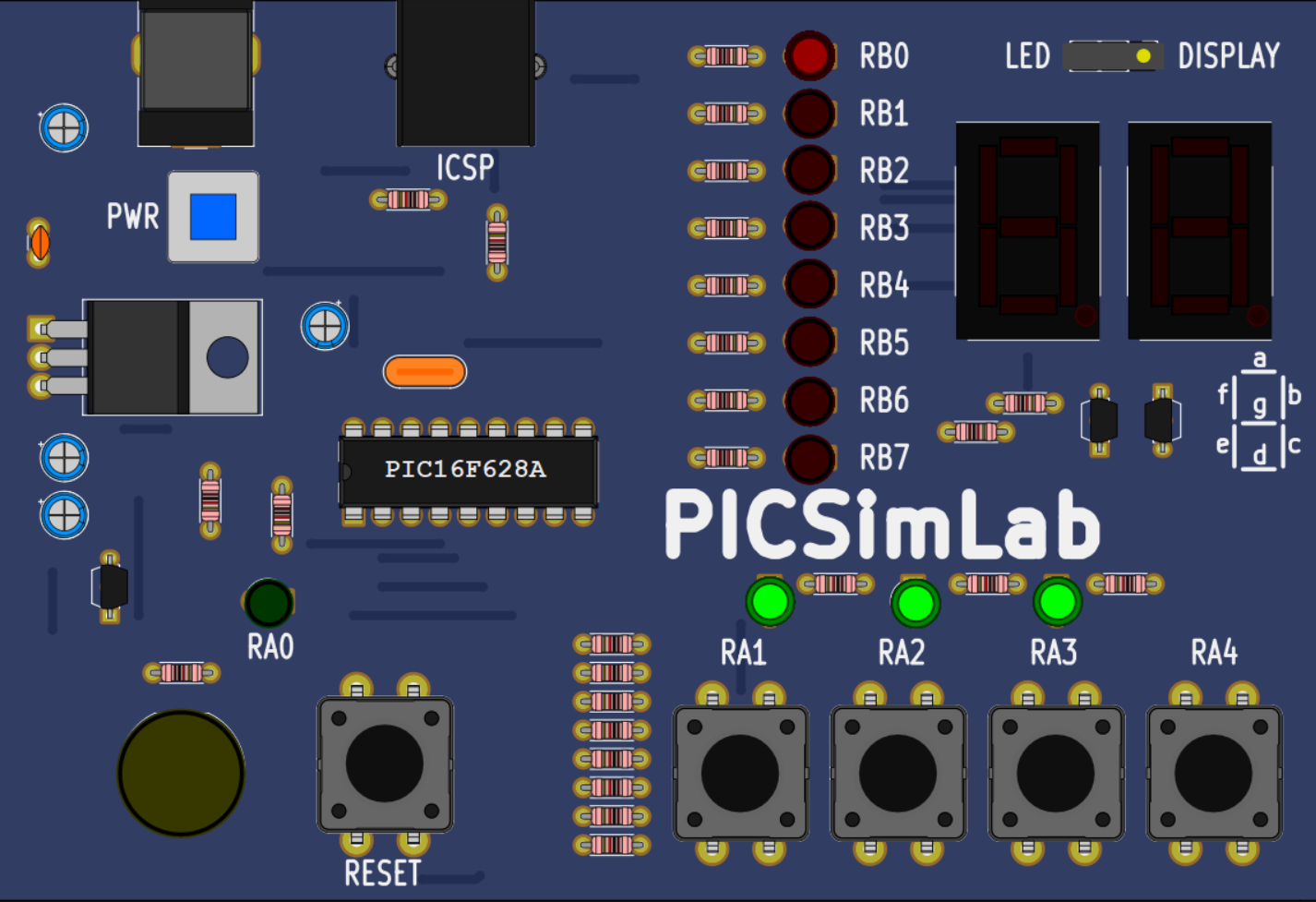


Offset -> traslare immagine verso l’alto o verso il basso -> sale di quanto mi interessa, confronto in verticale

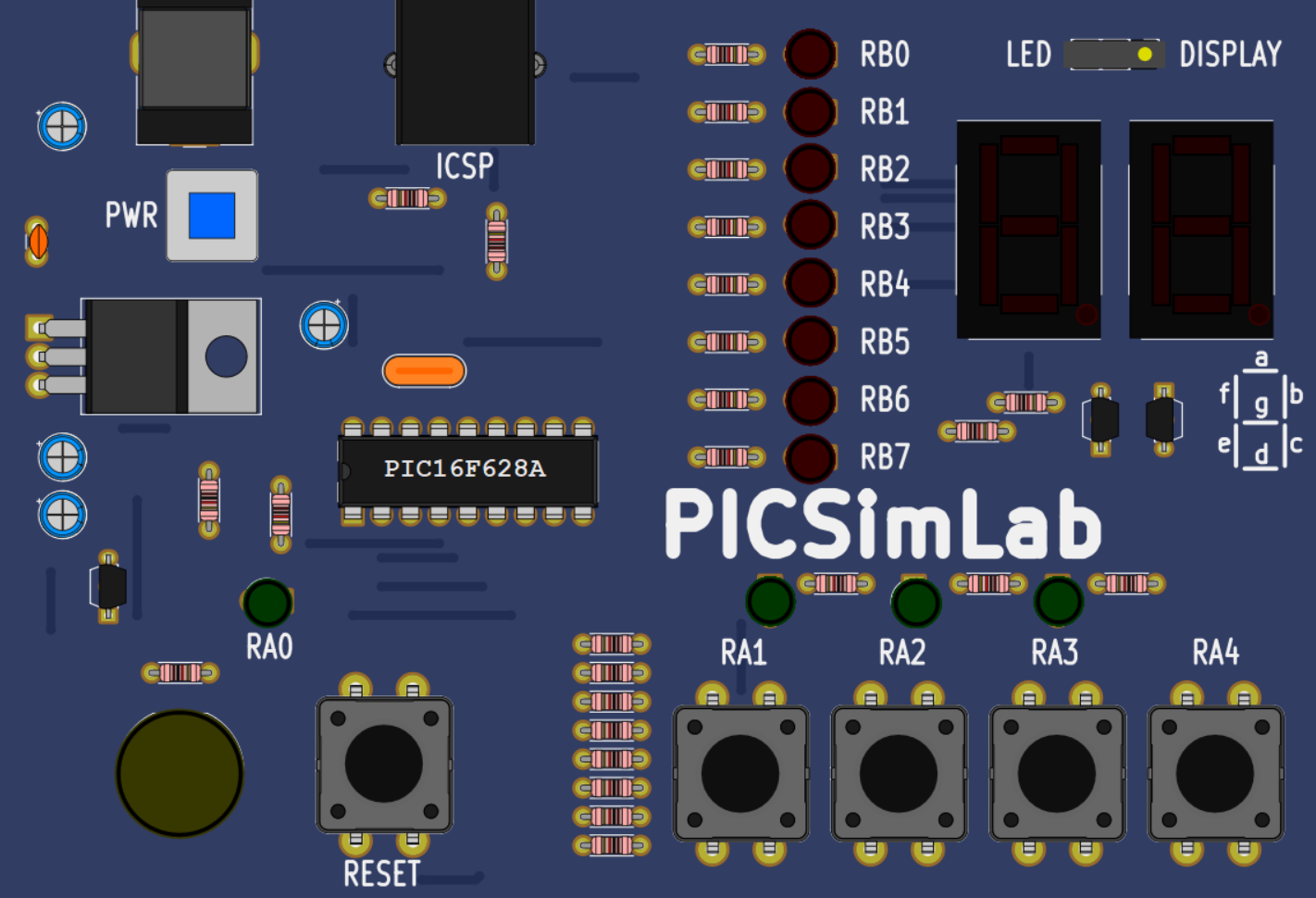
Clock lampeggiante -> oscilloscopio



Scheda con led acceso



Scheda con led spento

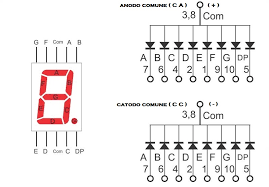


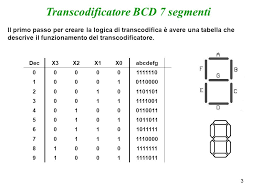
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

~ -> not controllo ogni bit -> dove c’è 0 diventa 1 dove è 1 diventa 0 -> controlla tutti i bit e inverte tutti i bit da 0 a 1 oppure da 1 a 0

! -> controlla tutto il risultato byte e lo nega

Display a 7 segmenti





**Conteggio con led a 7 segmenti da 0 a 9**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 |
| e | D | C |  | B | A | F | G |
| 0x04 | 0x08 | 0x20 |  | 0x40 | 0x80 | 0x02 | 0x01 |

#include <xc.h>

#define \_XTAL\_FREQ 8000000

char d[10] =

{

0xEE,0x28,0xCD, 0x6D, 0x2B, 0x67, 0xE7, 0x2C, 0xEF, 0x6F

};

void main(void) {

TRISB = 0x00;

while(1){

for(int i = 0; i < 10; i++)

{

PORTB = d[i];

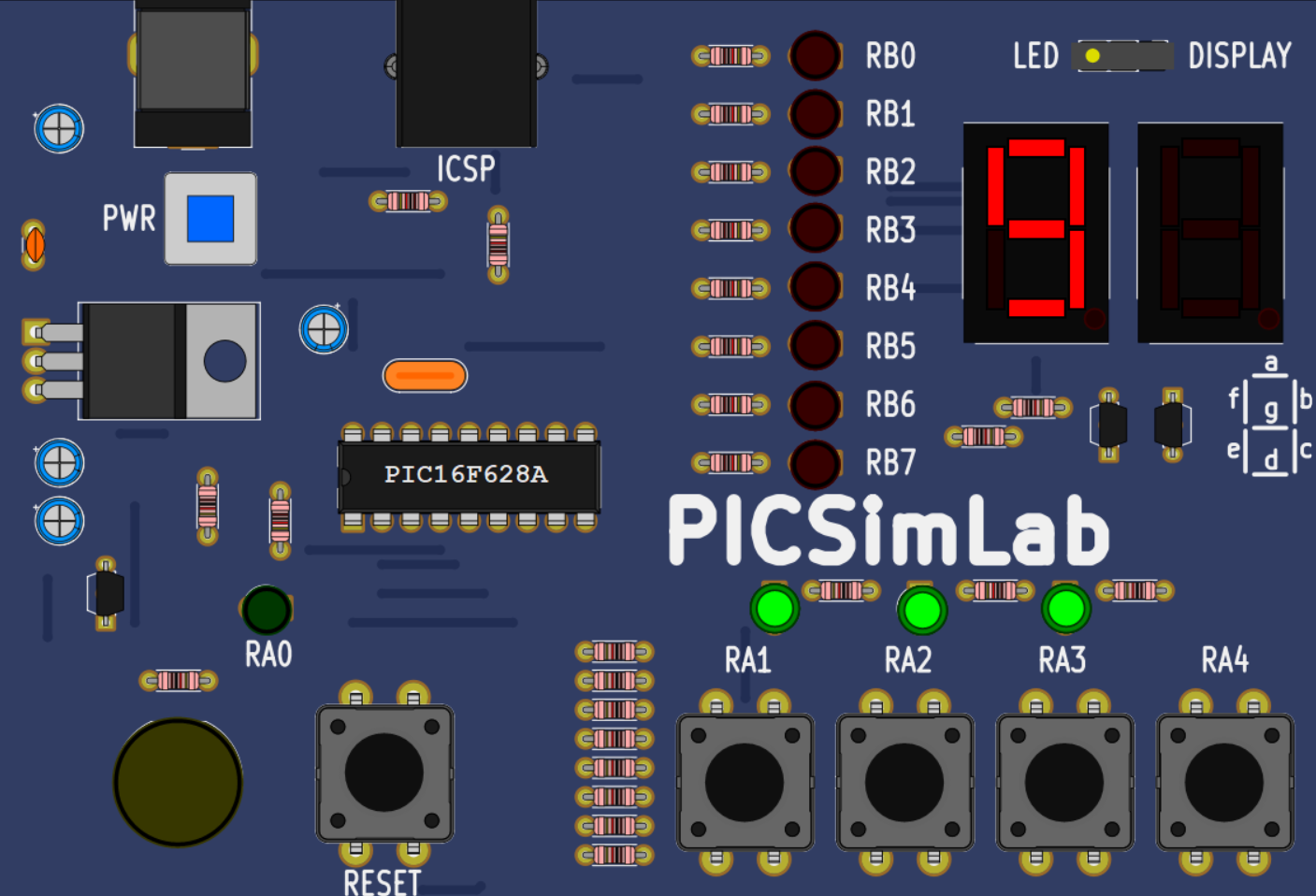
\_\_delay\_ms(1000);

}

}

}

Visualizzazione sulla scheda



Passare interrupt -> ciclo while più volte al posto di delay (rallenta programma) -> evento legato timer -> contatore-> riceve impulso ogni impulso incrementa di 1 il contatore -> numero max può contenere -> sopra il numero esce dal suo ciclo

1 -> dopo segnale di interrupt -> 9 bit -> massimo, torno a zero

Interrupt -> blocca esecuzione codice che fa eseguire

Nome funzione tipo \_\_interrupt() nome funzione ()

{

}

GIE -> genral interrupt enable -> impostare a 1 -> se no intrupt non li vedo mai

T0IF -> timer zero interrupt enable, ignora interrupt prima

Registro int com -> GIE , T0IF -> macchina sensibile interrupt timer zero -> primo interrupt

INTCON = ob10100000; -> in questo modo scatena interrupt in quella posizione

di GIE e T0IF

INTCON = 0xA0;

Bit 7 e bit 5 -> a 1 rappresentano GIE e T0IF;

Perscaller -> contatore divide la frequenza per 2 -> divide velocità contatore -> aumenta prestazione

T =

Interupt -> ogni 128000 ms o 128 micros

Interrupt -> più lento possibile -> f più lenta possibile -> mettono perscaller -> ogni 32,768 ms -> con più di 32,768 ms di tempo (t) -> nel codice bloccato circa un 1k volte quindi 1 s -> con 8MHz -> alla 33 volte è 1 s

Più veloce macchina -> più velocemente si blocca

Caso migliore -> interruzione ogni 8 s

Caso migliori -> utilizzare algoritmi che mi servono per quelle opzioni