ŠIAULIŲ VALSTYBINĖ KOLEGIJA
VERSLO IR TECHNOLOGIJŲ FAKULTETAS
TRANSPORTO INŽINERIJOS KATEDRA

MIKROPROCESORIAI IR JŲ PROGRAMAVIMAS

LABORATORINIS DARBAS NR. 4

EEPROM IR DINAMINIS ATVAIZDAVIMAS

DARBO TIKSLAS: Išmokti panaudoti išorinę EEPROM atmintį prijungtą per I2C sąsają ir išvesti informaciją dinaminio atvaizdavimo būdu

DARBO UŽDUOTIS: Pagal savo eilės Nr. sudaryti ir sujungti schemą bei užprogramuoti mikrovaldiklį taip, kad, prie jo prijungti du 7 segmentų LED indikatoriai dinaminiu būdu atvaizduotų iš EEPROM atminties nuskaitytą informaciją. Jei EEPROM atmintis tuščia, mikrovaldiklis į ją įrašo autoriaus gimimo dieną.

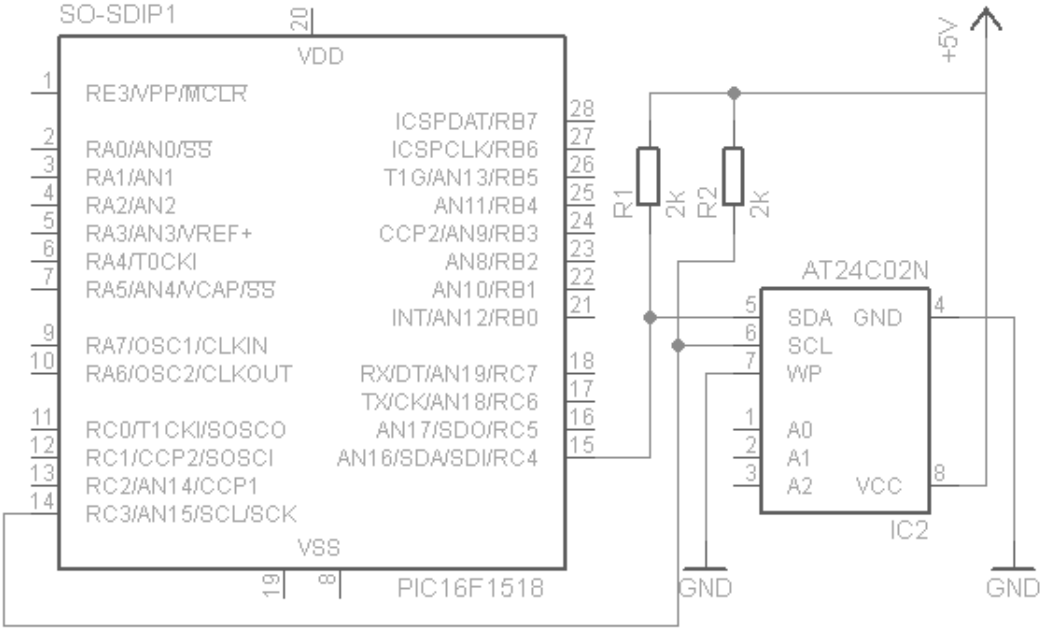
ATASKAITOS TURINYS:

Darbo tikslas;
Varianto duomenys;
Laboratorinio darbo schema pagal užduoties variantą;
Programos tekstas su komentarais (pagal užduoties variantą);
Programos algoritmas pagal užduoties variantą;
Išvados.

VARIANTAI PAGAL GRUPĖS SĄRAŠĄ:

Nr.	**7segm. duomenys**	**7segm. bendri anodai/katodai**	**7segm. indikatorių perjungimo periodas (~ms)**	**Duomenų pradžios EEPROM atmintyje adr.**
1.	PORTB	PORTC.0 ir PORTC.1	5	0x00
2.	PORTB	PORTC.1 ir PORTC.2	6	0x01
3.	PORTB	PORTC.2 ir PORTC.5	7	0x02
4.	PORTB	PORTC.5 ir PORTC.6	8	0x03
5.	PORTB	PORTC.6 ir PORTC.7	9	0x04
6.	PORTB	PORTC.7 ir PORTC.0	10	0x05
7.	PORTB	PORTC.0 ir PORTC.2	11	0x06
8.	PORTB	PORTC.5 ir PORTC.7	12	0x07
9.	PORTB	PORTC.6 ir PORTC.1	13	0x08
10.	PORTB	PORTC.7 ir PORTC.1	14	0x09
11.	PORTB	PORTC.6 ir PORTC.2	15	0x0A
12.	PORTB	PORTC.5 ir PORTC.0	16	0x0B
13.	PORTB	PORTC.0 ir PORTC.5	17	0x0C
14.	PORTB	PORTC.0 ir PORTC.6	18	0x0D
15.	PORTB	PORTC.0 ir PORTC.7	19	0x0E
16.	PORTB	PORTC.1 ir PORTC.0	20	0x0F
17.	PORTB	PORTC.1 ir PORTC.5	21	0x10
18.	PORTB	PORTC.1 ir PORTC.6	22	0x11
19.	PORTB	PORTC.1 ir PORTC.7	23	0x12
20.	PORTB	PORTC.0 ir PORTC.1	24	0x13
21.	PORTB	PORTC.2 ir PORTC.0	25	0x14
22.	PORTB	PORTC.2 ir PORTC.1	26	0x15
23.	PORTB	PORTC.2 ir PORTC.6	27	0x16
24.	PORTB	PORTC.2 ir PORTC.7	28	0x17
25.	PORTB	PORTC.5 ir PORTC.0	29	0x18
26.	PORTB	PORTC.5 ir PORTC.1	30	0x19
27.	PORTB	PORTC.5 ir PORTC.2	31	0x1A
28.	PORTB	PORTC.6 ir PORTC.0	32	0x1B
29.	PORTB	PORTC.6 ir PORTC.5	33	0x1C
30.	PORTB	PORTC.7 ir PORTC.2	34	0x1D
31.	PORTB	PORTC.7 ir PORTC.5	35	0x1E
32.	PORTB	PORTC.7 ir PORTC.6	36	0x1F

SCHEMOTECHNINIŲ SPRENDIMŲ PAVYZDŽIAI:



PROGRAMOS KODO PAVYZDŽIAI:

1 PVZ. Daliklio vertė nustatyta 1:16, t.y. TMR0 reikšmė bus didinama vienetu kas 16µs. Pradinė TMR0 reikšmė yra 130. Vadinasi pertraukimas, persipildžius TMR0 reikšmei įvyks kas (255 - 130) · 16 µs = 2000µs. Kintamojo k reikšmė didinama kiekvieno pertraukimo metu. Kai ši reikšmė tampa lygi 25 gaunamas laiko intervalas 2000µs · 25 = 50 000µs = 50ms.

```
<...>
unsigned k; //sukuriamas globalus kintamasis k

void interrupt isr (void) { //pertraukimo paprogramė
  k=k+1; //su kiekvienu pertraukimu k reikšmė padidinama vienetu
  TMR0 = 130; // Atstatoma pradinė TMR0 reikšmė
  INTCON = 0b10100000; // Nustatoma GIE, T0IE, išvaloma T0IF
  if (k == 25) { // Jeigu k reikšmė pasiekė 196, t.y. praėjo 1s
    PORTB = ~PORTB; // Invertuojama PORTB reikšmė
    k = 0; // skaičiuosime laiką iš naujo iki 1s.
  }
}

void main() {
  OSCCON=0b01101010; //Nustatome, jog mikrovaldiklis dirbs 4MHz dažniu
  OPTION_REG = 0b000000011; //parenkame skaitikliui TMR0 vidinį taktinių impulsų šaltinį (OPTION_REG
//5 bitas T0CS →0), daliklį priskiriame skaitikliui TMR0 (OPTION_REG 3 bitas PSA →0), nustatome dalikio
//vertę 1:16 (OPTION_REG 2-0 bitai PS2 →0, PS1→1, PS0→1), t.y. TMR0 reikšmė bus padidinama vienetu
//kas 16 µs.
  ANSELA = 0;
  ANSELB = 0; // Visi I/O išvadai yra skaitmeniniai
  TRISB = 0; // PORTB skirtas išvedimui
  PORTB = 0xFF; //Į PORTB nusiunčiama 0xFF
  TMR0 = 130; // Nustatoma pradinė TMR0 reikšmė
  INTCON = 0b10100000; // Aktyvuojami pertraukimai
  k = 0; // Inicializuojamas k
  while(1);
```

```
}
```

2. PVZ. Į EEPROM atmintį adresu 0xAA įrašomi duomenys 0x55, o 0xAB – duomenys 0x11. Po to, į kintamąjį *duom*, nuskaitomi duomenys iš adresų 0x05 ir 0x06.

```
<...>
#include „ee.h“; //nepamirškite į skyrių header files įdėti ee.h, o į skyrių source files ee.c bylas

#define SYS_FREQ        4000000L   //4MHz
#define FCY             SYS_FREQ/4 //1MIPS (1 mega instrukcija per sekundę)
#define  delay_ms(x)   _delay((unsigned long)((x)*(FCY/1000.0)))

void main() {
  char duom;
  ANSELA = 0; //
  ANSELB = 0; // Visi I/O išvadai yra skaitmeniniai
  TRISC = 0b00011000;
  ee_byte_to_ee(0xAA, 0x55); //įrašome 0x55 į EEPROM atmintį adresu 0xAA
  delay_ms(10); //delsiame 10ms – tai reikalinga, kad EEPROM atmintinė spėtų pabaigti įrašymo operaciją.
  ee_byte_to_ee(0xAB, 0x11);
  delay_ms(10);
  ee_byte_from_ee(0x05, &duom); //į duom nuskaitome duomenis iš EEPROM atminties adreso 0x05
  ee_byte_from_ee(0x06, &duom);
  while(1);
}
```