

5. Елементи індикації

Практично кожна мікропроцесорна система управління містить елементи індикації. Як індикатори в даний час найчастіше застосовуються світлодіоди. На ринку є величезний вибір світлодіодів найрізноманітніших видів і розмірів.

У мікропроцесорній системі управління LED індикатори можуть служити для відображення різних режимів роботи: попередження про критичні ситуації, відображення ходу прийому сигналів керування тощо. Підключити одиночний світлодіодний індикатор до МК дуже просто. На рис. 5.1 наведена схема підключення світлодіода безпосередньо до виводу порту МК.

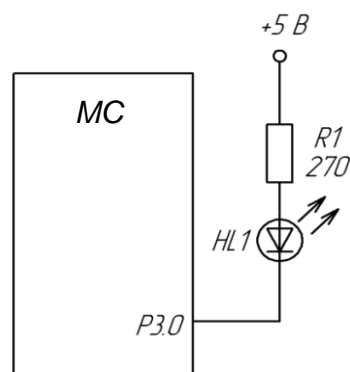


Рис. 5.1. Схема підключення світлодіодного індикатора

Усі вихідні каскади МК мають достатню навантажувальну здатність для того, щоб витримати підключення одного світлодіодного індикатора зі споживаним струмом у робочому режимі не більше 20 мА. Для керування двома світлодіодами одним виходом у МК передбачено активні вихідні каскади, і для перемикання режиму роботи (введення або виведення) слугує спеціальний регістр. Таким чином, сигнал кожного виходу будь-якого порту може мати 3 значення – "0", "1" і високоімпедансний ("Z") стан. Це дозволяє керувати двома світлодіодами за допомогою одного виводу (рис. 5.2).

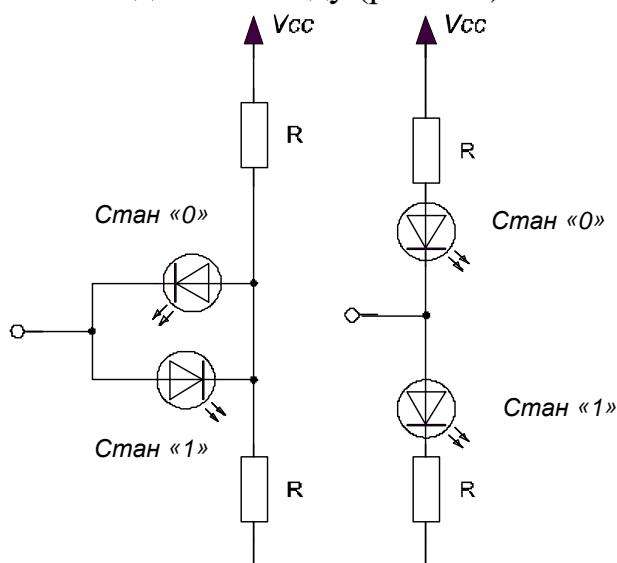


Рис. 5.2. Керування двома світлодіодами одним виходом МК

При роботі порту в режимі виходу, залежно від стану "0" або "1" горить відповідно верхній або нижній світлодіод. При перемиканні в Z-стан, і при відповідному виборі резисторів струм через світлодіоди дуже малий і їх світіння майже непомітно.

Семисегментний світлодіодний індикатор складається з семи елементів індикації (сегментів) (рис. 5.3), що включаються і виключаються окремо. Включаючи їх в різних комбінаціях, з них можна скласти спрощені зображення арабських цифр. Семисегментні світлодіодні індикатори бувають різних кольорів, зазвичай це білий, червоний, зелений, жовтий і блакитний кольори. Крім того, вони можуть бути різних розмірів.

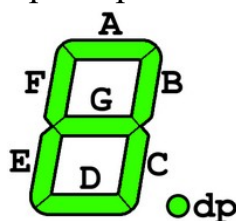


Рис. 5.3. Позначення сегментів на індикаторі

Сегменти позначаються буквами від А до G; восьмий сегмент - десяткова точка (decimal point, DP), призначена для відображення дрібних чисел. Інколи на семисегментному індикаторі відображають літери.

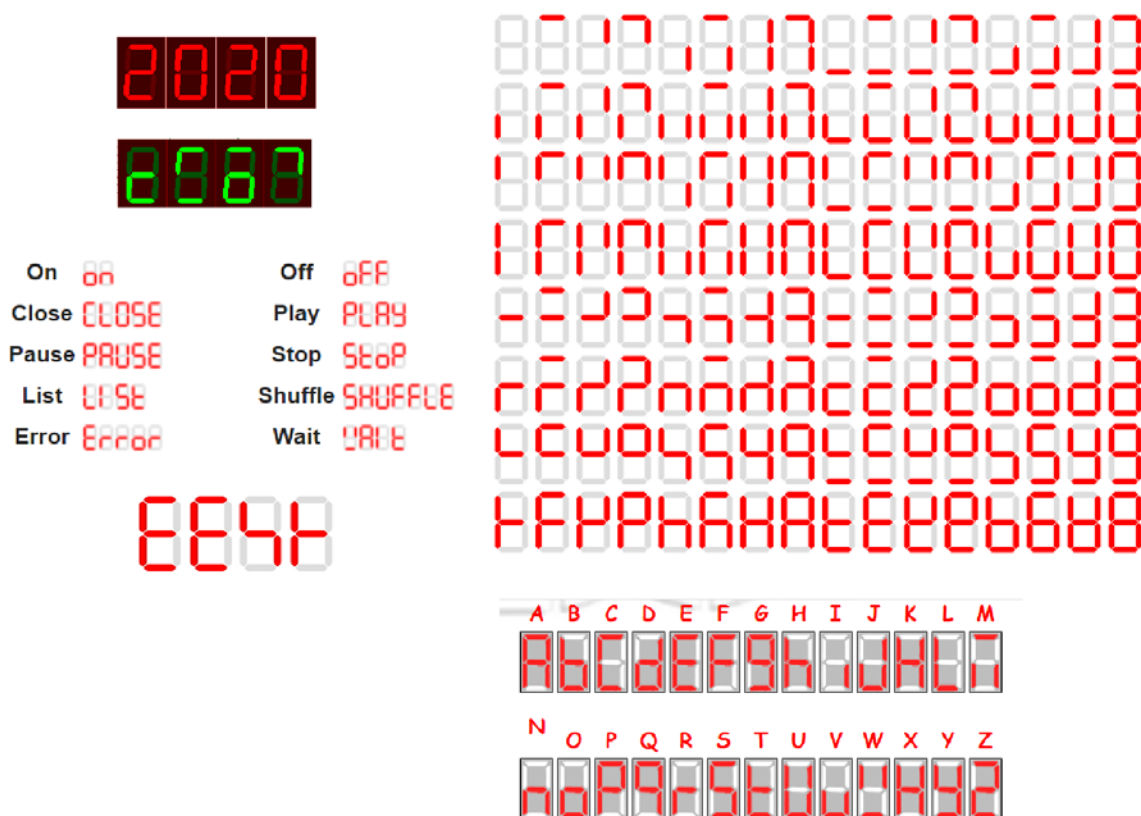


Рис. 5.4. Відображення літер на семисегментному індикаторі

У звичайному світлодіодному індикаторі дев'ять виводів: один йде до катодів усіх сегментів, а решта вісім – до анода кожного з сегментів. Ця схема називається «схема із загальним катодом», існують також схеми з загальним анодом (тоді все навпаки) (рис. 5.5).

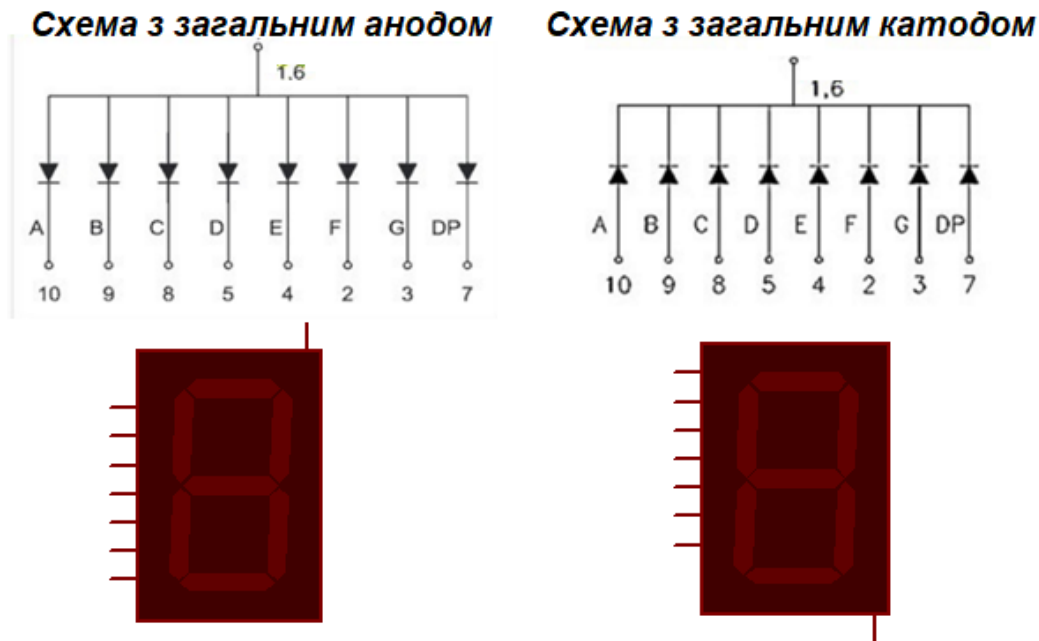
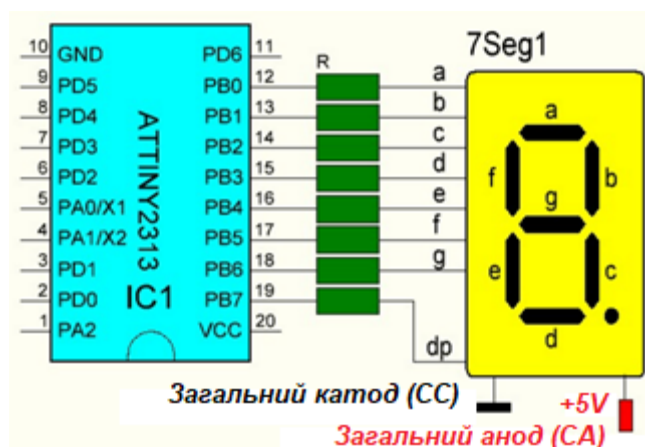


Рис. 5.5. Умовне позначення та схема 7 сегментного індикатора

На рис. 5.6 показано як підключається однорозрядний семисегментний індикатор до мікроконтролера.

При цьому слід враховувати, що якщо індикатор із загальним катодом, то його загальний вивід підключається до «землі», а керування сегментами відбувається подачею логічної одиниці на вивід порту.

Якщо індикатор з загальним анодом, то на його загальний провід подають «плюс» напруги, а керування сегментами відбувається подачею логічного нуля на вивід порту.



Характеристики індикатора

- робоча напруга – 2В
- робочий струм ток – 10 мА
- напруга живлення 5В

Формула розрахунку:

$$R = U/I = (5-2)/0,01 = 300 \text{ Ом}$$

Рис. 5.6. Підключається однорозрядного 7 сегментного індикатора до мікроконтролера

Лістинг програми 1. Вивід інформації на 7 сегментний індикатор

```
void setup() {  
  //виставляємо всі біти порти D як вихід  
  DDRD = 0xFF;  
}  
void loop() {  
  PORTD = 0x4F; //виводимо 3 СС  
  //PORTD = 0xB0; //виводимо 3 СА  
  delay (1000);  
}
```

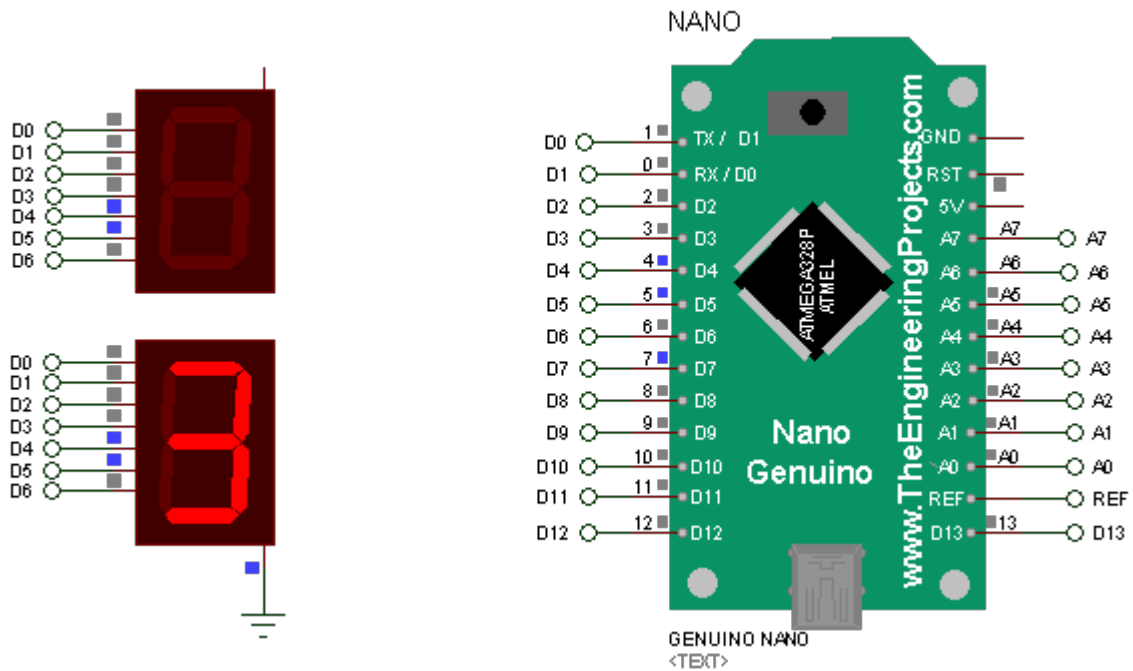


Рис. 5.7. Результат роботи програми (лістинг 1)

Лістинг програми 2. Вивід інформації на 7 сегментний індикатор

```
byte text[]={0x58, 0x54, 0x06, 0x7F};  
//From CA{0xA7,0xAB,0xF9, 0x80}  
void setup() {  
  //виставляємо всі біти порти D як вихід  
  DDRD = 0xFF;  
}  
void loop() {  
  for (int i=0; i<4; i++){  
    PORTD = text[i];  
    delay (500);  
    PORTD = 0x00;  
    delay (500);  
  }  
}
```

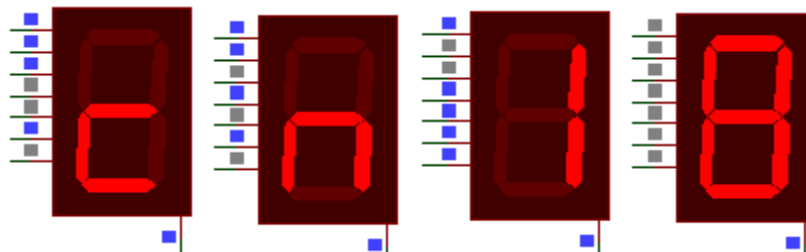


Рис. 5.8. Результат роботи програми (лістинг 2)

	dp	G	F	E	D	C	B	A	h
	7	6	5	4	3	2	1	0	
	0	0	0	0	0	0	0	0	
0		0	1	1	1	1	1	1	3F
1		0	0	0	0	1	1	0	6
2		1	0	1	1	0	1	1	5B
3		1	0	0	1	1	1	1	4F
4		1	1	0	0	1	1	0	66
5		1	1	0	1	1	0	1	6D
6		1	1	1	1	1	0	1	7D
7		0	0	0	0	1	1	1	7
8		1	1	1	1	1	1	1	7F
9		1	1	0	1	1	1	1	6F

Рис. 5.9. Перекодування цифри у код семисегментного індикатора

Багаторозрядні світлодіодні індикатори часто працюють за динамічним принципом: виводи однойменних сегментів всіх розрядів з'єднані разом. Щоб виводити інформацію на такий індикатор, мікроконтролер повинний циклічно подавати сигнал на загальні виводи всіх розрядів, в той час як на виводи сегментів сигнал подається в залежності від того, який сегмент в даному розряді має світитися.

Схема підключення багаторозрядного семисегментного світлодіодного індикатора аналогічна схемі підключення одnorозрядного індикатора. Єдине, додаються керуючі транзистори в катодах (анодах) індикаторів (рис. 5.10).

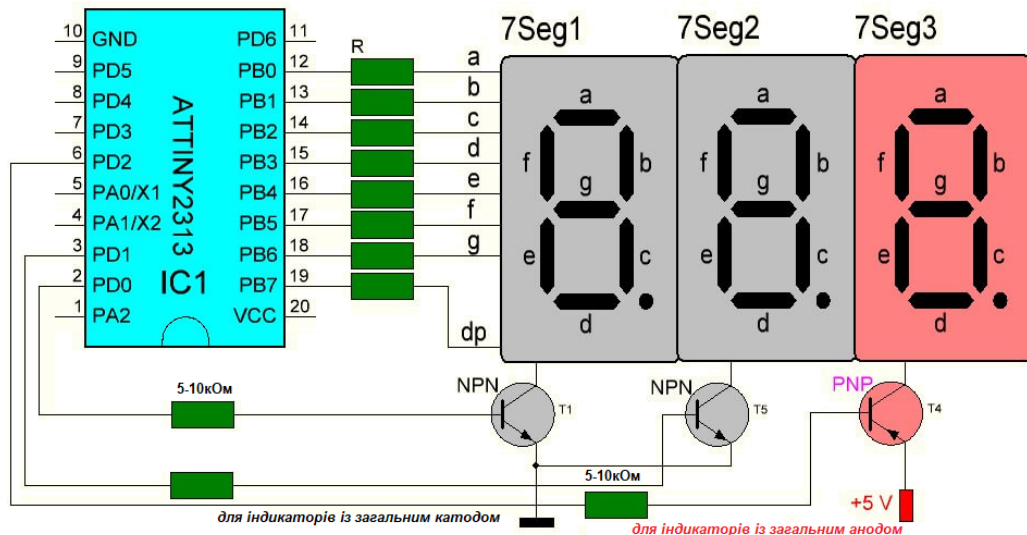


Рис. 5.10. Схема підключення багаторозрядного семисегментного світлодіодного індикатора

Здійснення індикації розрядами динамічним способом:

- виставляється двійковий код відповідної цифри на виходах порту РВ для 1 розряду, потім подається логічний рівень на керуючий транзистор першого розряду;
- виставляється двійковий код відповідної цифри на виходах порту РВ для 2 розряду, потім подається логічний рівень на керуючий транзистор другого розряду;
- виставляється двійковий код відповідної цифри на виходах порту РВ для 3 розряду, потім подається логічний рівень на керуючий транзистор третього розряду;
- усе повторяємо.

Треба враховувати:

- для індикаторів з ЗК (СС) застосовується керуючий транзистор структури NPN (управляється логічною одиницею);
- для індикатора з ЗА (СА) - транзистор структури PNP (управляється логічним нулем).

Лістинг програми 3. Вивід інформації на 4-х розрядний 7 сегментний індикатор

```
int pinA = 2;
int pinB = 3;
int pinC = 4;
int pinD = 5;
int pinE = 6;
int pinF = 7;
int pinG = 8;
int D1 = 9;

int D2 = 10;
int D3 = 11;
int D4 = 12;

void setup() {
    pinMode(pinA, OUTPUT);
    pinMode(pinB, OUTPUT);
    pinMode(pinC, OUTPUT);
```

```

pinMode(pinD, OUTPUT);
pinMode(pinE, OUTPUT);
pinMode(pinF, OUTPUT);
pinMode(pinG, OUTPUT);
pinMode(D1, OUTPUT);
pinMode(D2, OUTPUT);
pinMode(D3, OUTPUT);
pinMode(D4, OUTPUT);
}

void loop() {
    digitalWrite(D1, HIGH);
    digitalWrite(D2, LOW);
    digitalWrite(D3, LOW);
    digitalWrite(D4, LOW);
    //0
    digitalWrite(pinA, HIGH);
    digitalWrite(pinB, HIGH);
    digitalWrite(pinC, HIGH);
    digitalWrite(pinD, HIGH);
    digitalWrite(pinE, HIGH);
    digitalWrite(pinF, HIGH);
    digitalWrite(pinG, LOW);
    delay(10);

    digitalWrite(D1, LOW);
    digitalWrite(D2, HIGH);
    digitalWrite(D3, LOW);
    digitalWrite(D4, LOW);
    //1
    digitalWrite(pinA, LOW);
    digitalWrite(pinB, HIGH);
    digitalWrite(pinC, HIGH);
    digitalWrite(pinD, LOW);
    digitalWrite(pinE, LOW);
    digitalWrite(pinF, LOW);
    digitalWrite(pinG, HIGH);
    delay(10);

    digitalWrite(D1, LOW);
    digitalWrite(D2, LOW);
    digitalWrite(D3, LOW);
    digitalWrite(D4, HIGH);
    //2
    digitalWrite(pinA, HIGH);
    digitalWrite(pinB, HIGH);
    digitalWrite(pinC, HIGH);
    digitalWrite(pinD, HIGH);
    digitalWrite(pinE, LOW);
    digitalWrite(pinF, LOW);
    digitalWrite(pinG, HIGH);
    delay(45);
}

```

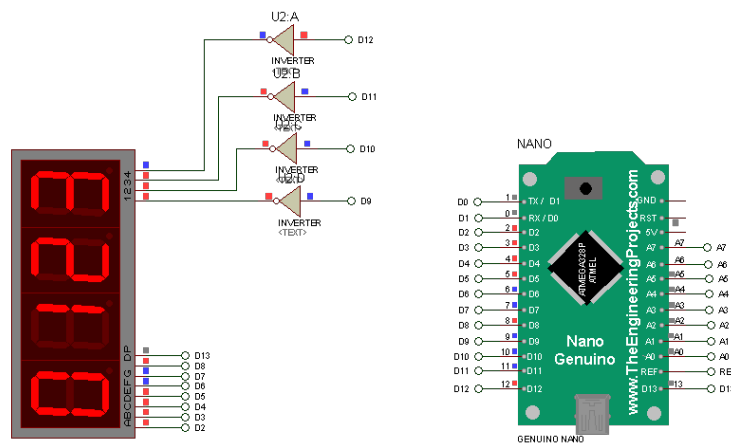


Рис. 5.11. Результат роботи програми (лістинг 3)

Лістинг програми 5. Вивід інформації на 4-х розрядний 7
сегментний індикатор

```
void setup() {
  DDRD=0xFF;
  DDRB=0xFF;
  PORTD =0x00;
  PORTB = 0<<0;
}
void loop() {
  PORTD=B11111100; //0
  PORTB=P0;
  delay (40);

  PORTD=B00011000;//1
  PORTB=P1;
  delay (40);

  PORTD=B01101100;//2
  PORTB=P2;
  PORTB|=1;
  delay (40);

  PORTD=B00111100;//3
  PORTB=P3;
  PORTB|=1;
  delay (40);
}
```

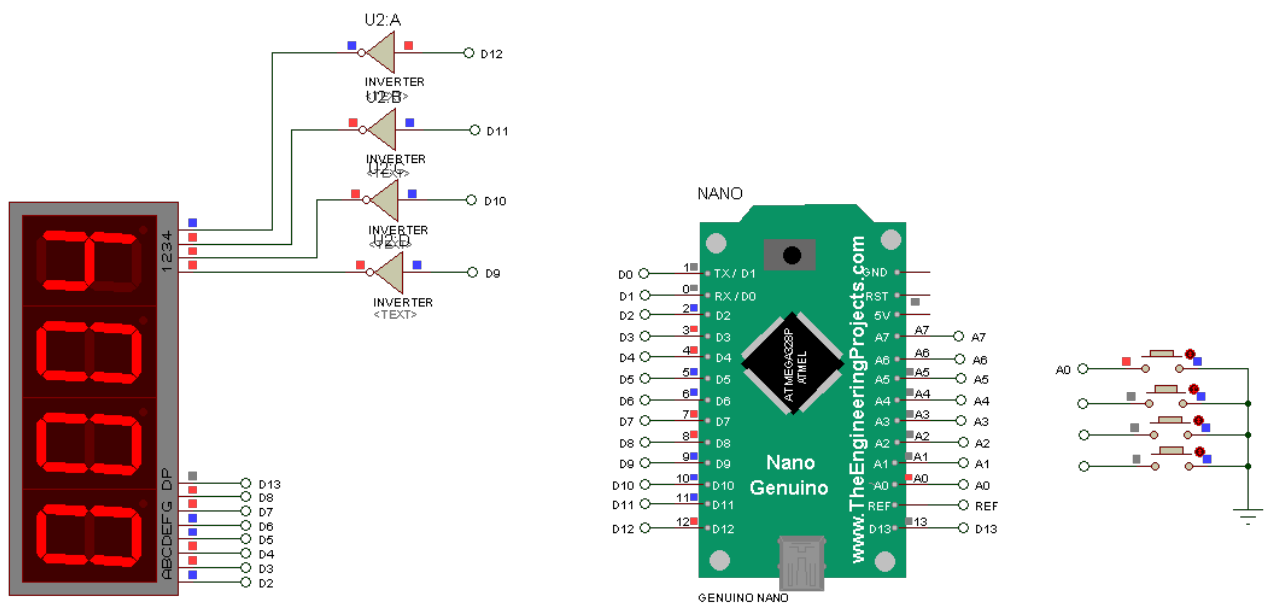


Рис. 5.11. Результат роботи програми (лістинг 6)

Лістинг програми 6. Вивід інформації на 4-х розрядний 7
сегментний індикатор з використанням переривання

```
//Counter 0-9999 with switch - A0
#define button A0

// segment pin definitions
#define SegA 2
#define SegB 3
#define SegC 4
#define SegD 5
#define SegE 6
#define SegF 7
#define SegG 8

// common pins of the four digits definitions
#define Dig1 9
#define Dig2 10
#define Dig3 11
#define Dig4 12

// variable declarations
byte current_digit;
int count = 0;

void setup()
{
    pinMode(button, INPUT_PULLUP);
    pinMode(SegA, OUTPUT);
    pinMode(SegB, OUTPUT);
    pinMode(SegC, OUTPUT);
    pinMode(SegD, OUTPUT);
    pinMode(SegE, OUTPUT);
    pinMode(SegF, OUTPUT);
    pinMode(SegG, OUTPUT);
    pinMode(Dig1, OUTPUT);
    pinMode(Dig2, OUTPUT);
    pinMode(Dig3, OUTPUT);
    pinMode(Dig4, OUTPUT);

    disp_off(); // turn off the display

    // Timer1 module overflow interrupt configuration
    TCCR1A = 0;
    TCCR1B = 1; //enable Timer1 with prescaler=1(16 ticks each 1 µs)
    TCNT1 = 0; // set Timer1 preload value to 0 (reset)
    TIMSK1 = 1; // enable Timer1 overflow interrupt
```

```

}

ISR(TIMER1_OVF_vect)    // Timer1 interrupt service routine (ISR)
{
    disp_off();    // turn off the display

    switch (current_digit)
    {
        case 1:
            disp(count / 1000);    // prepare to display digit 1 (most left)
            digitalWrite(Dig1, HIGH);    // turn on digit 1
            break;

        case 2:
            disp( (count / 100) % 10);    // prepare to display digit 2
            digitalWrite(Dig2, HIGH);    // turn on digit 2
            break;

        case 3:
            disp( (count / 10) % 10);    // prepare to display digit 3
            digitalWrite(Dig3, HIGH);    // turn on digit 3
            break;

        case 4:
            disp(count % 10);    // prepare to display digit 4 (most right)
            digitalWrite(Dig4, HIGH);    // turn on digit 4
    }

    current_digit = (current_digit % 4) + 1;
}

void loop()
{
    if(digitalRead(button) == 0)
    {
        count++;    // increment 'count' by 1
        if(count == 9999)
            count = 0;
        delay(200);
    }
}

void disp(byte number)
{
    switch (number)
    {
        case 0:    // print 0
            digitalWrite(SegA, HIGH);
    }
}

```

```
        digitalWrite(SegB, HIGH);
        digitalWrite(SegC, HIGH);
        digitalWrite(SegD, HIGH);
        digitalWrite(SegE, HIGH);
        digitalWrite(SegF, HIGH);
        digitalWrite(SegG, LOW);
        break;

case 1: // print 1
    digitalWrite(SegA, LOW);
    digitalWrite(SegB, HIGH);
    digitalWrite(SegC, HIGH);
    digitalWrite(SegD, LOW);
    digitalWrite(SegE, LOW);
    digitalWrite(SegF, LOW);
    digitalWrite(SegG, LOW);
    break;

case 2: // print 2
    digitalWrite(SegA, HIGH);
    digitalWrite(SegB, HIGH);
    digitalWrite(SegC, LOW);
    digitalWrite(SegD, HIGH);
    digitalWrite(SegE, HIGH);
    digitalWrite(SegF, LOW);
    digitalWrite(SegG, HIGH);
    break;

case 3: // print 3
    digitalWrite(SegA, HIGH);
    digitalWrite(SegB, HIGH);
    digitalWrite(SegC, HIGH);
    digitalWrite(SegD, HIGH);
    digitalWrite(SegE, LOW);
    digitalWrite(SegF, LOW);
    digitalWrite(SegG, HIGH);
    break;

case 4: // print 4
    digitalWrite(SegA, LOW);
    digitalWrite(SegB, HIGH);
    digitalWrite(SegC, HIGH);
    digitalWrite(SegD, LOW);
    digitalWrite(SegE, LOW);
    digitalWrite(SegF, HIGH);
    digitalWrite(SegG, HIGH);
    break;
```

```
case 5: // print 5
    digitalWrite(SegA, HIGH);
    digitalWrite(SegB, LOW);
    digitalWrite(SegC, HIGH);
    digitalWrite(SegD, HIGH);
    digitalWrite(SegE, LOW);
    digitalWrite(SegF, HIGH);
    digitalWrite(SegG, HIGH);
    break;
```

```
case 6: // print 6
    digitalWrite(SegA, HIGH);
    digitalWrite(SegB, LOW);
    digitalWrite(SegC, HIGH);
    digitalWrite(SegD, HIGH);
    digitalWrite(SegE, HIGH);
    digitalWrite(SegF, HIGH);
    digitalWrite(SegG, HIGH);
    break;
```

```
case 7: // print 7
    digitalWrite(SegA, HIGH);
    digitalWrite(SegB, HIGH);
    digitalWrite(SegC, HIGH);
    digitalWrite(SegD, LOW);
    digitalWrite(SegE, LOW);
    digitalWrite(SegF, LOW);
    digitalWrite(SegG, LOW);
    break;
```

```
case 8: // print 8
    digitalWrite(SegA, HIGH);
    digitalWrite(SegB, HIGH);
    digitalWrite(SegC, HIGH);
    digitalWrite(SegD, HIGH);
    digitalWrite(SegE, HIGH);
    digitalWrite(SegF, HIGH);
    digitalWrite(SegG, HIGH);
    break;
```

```
case 9: // print 9
    digitalWrite(SegA, HIGH);
    digitalWrite(SegB, HIGH);
    digitalWrite(SegC, HIGH);
    digitalWrite(SegD, HIGH);
    digitalWrite(SegE, LOW);
    digitalWrite(SegF, HIGH);
    digitalWrite(SegG, HIGH);
```

```

    }
}
void disp_off()
{
    digitalWrite(Dig1, LOW);
    digitalWrite(Dig2, LOW);
    digitalWrite(Dig3, LOW);
    digitalWrite(Dig4, LOW);
}

```

Схема реалізації динамічної індикація без додаткових елементів наведена на рис. 5.12. До порту В МК підключені катоди всіх світлодіодів матриці, а до порту А – аноди кожного з індикаторів, що створюють матрицю. На лініях порту А організовується одиниця, що «біжить». На лінії порту В при кожному положенні одиниці, що біжить, виводиться семисегментний код того символу, який повинен горіти в даному знакомісці. Для індикаторів із загальним катодом замість одиниці, що біжить, використовується нуль, що біжить. Перевага такого способу індикації — у відсутності яких-небудь додаткових компонентів (окрім самих світлодіодних індикаторів), головний недолік — значна перевитрата ліній портів. Таке рішення для МК може забезпечити роботу не більше 5 семисегментних індикаторів одночасно. При використанні інших МК з великою кількістю ніжок вказана проблема знімається.

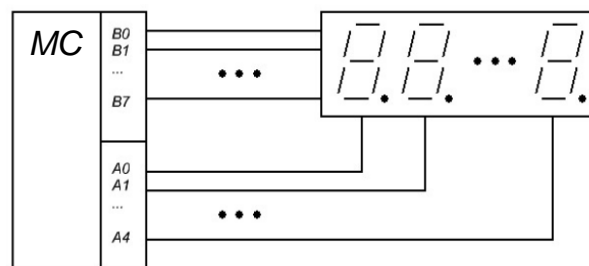


Рис. 5.12. Схема реалізації динамічної індикація без додаткових елементів

Схема реалізації динамічної індикації з одним додатковим елементом наведена на рис. 5.13. У цьому варіанті одиниця, що біжить, реалізується за допомогою регістра зсуву. Порт В у цій схемі також використовується в режимі часового мультиплексування, як для видачі символу, так і для занесення чергового біта до регістра зсуву. Схема також вимагає всього одну додаткову

лінію (крім порту В).

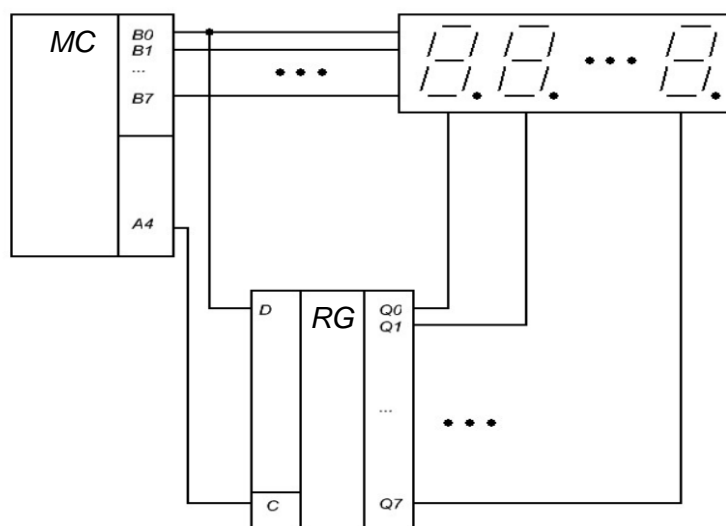


Рис. 5.13. Схема реалізації динамічної індикації з регістром зсуву

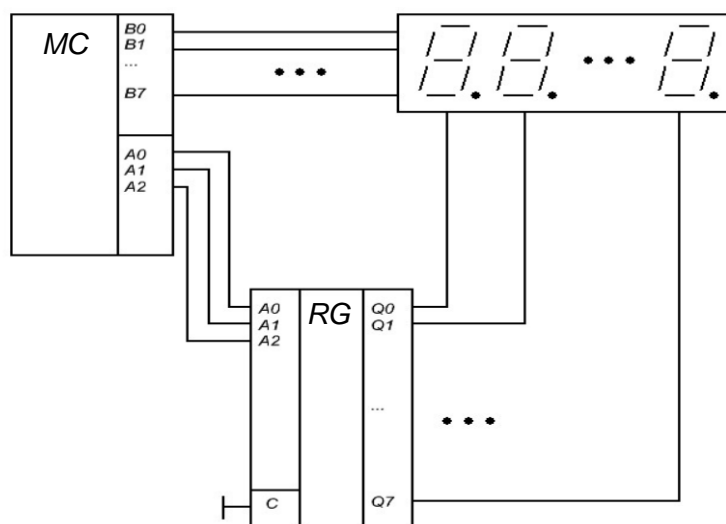


Рис.5.14. Схема реалізації динамічної індикації з дешифратором

Ще один варіант реалізації схеми з одним додатковим елементом наведений на рис. 5.14. Така схема придатна тільки для індикаторів із загальним катодом, оскільки для організації нуля, що біжить, в ній використовується дешифратор. Схема вимагає три додаткові лінії (крім порту В), проте у багатьох випадках вона може виявитися корисною.

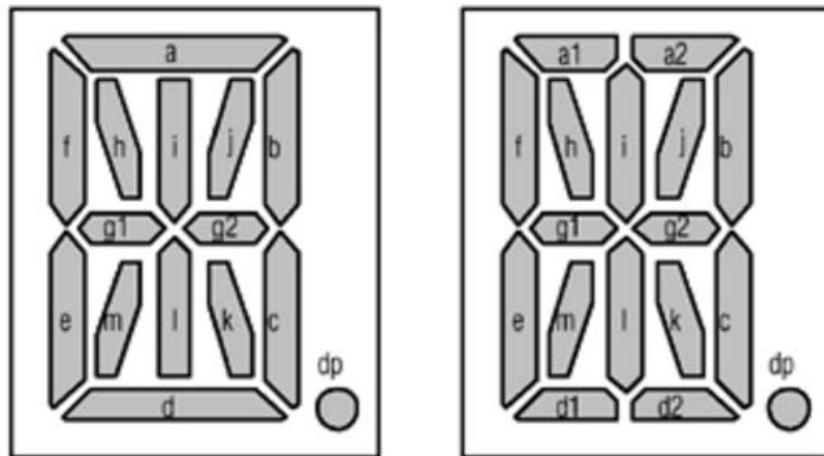


Рис.5.15. Багатосегментні символні індикатори