**Лабораторна робота №2**

**Виведення інформації за допомогою LCD індикатору**

**Тема**: виведення статичної та динамічної інформації на LCD дисплей.

**Мета**: дослідження особливостей виведення інформації на LCD дисплей.

Рідиннокристалічний дисплей (англ. liquid crystal display (LCD)) — електронний пристрій візуального відображення інформації (дисплей), принцип дії якого ґрунтується на явищі електричного переходу Фредерікса в рідких кристалах. Дисплей складається з довільної кількості кольорових або монохромних точок (пікселів), і джерела світла або відбивача (рефлектора).

Кожна з кольорових точок рідкокристалічного дисплея складається з кількох комірок (як правило, з трьох), попереду яких встановлюються світлові фільтри (найчастіше — червоний, синій і зелений). Тобто колір певної точки і її яскравість визначається інтенсивностями світіння комірок, з яких вона складається.

Керування кожною рідкокристалічною коміркою здійснюється з допомогою напруги, яку подає на комірку один з тонкоплівкових транзисторів (TFT — абревіатура від англ. Thin-Film Transistor).

Рідкокристалічні дисплеї мають низьке енергоспоживання, тому вони знайшли широке застосування, як у кишенькових пристроях (годинниках, мобільних телефонах, кишенькових комп'ютера), так і в комп'ютерних моніторах, телевізорах тощо.

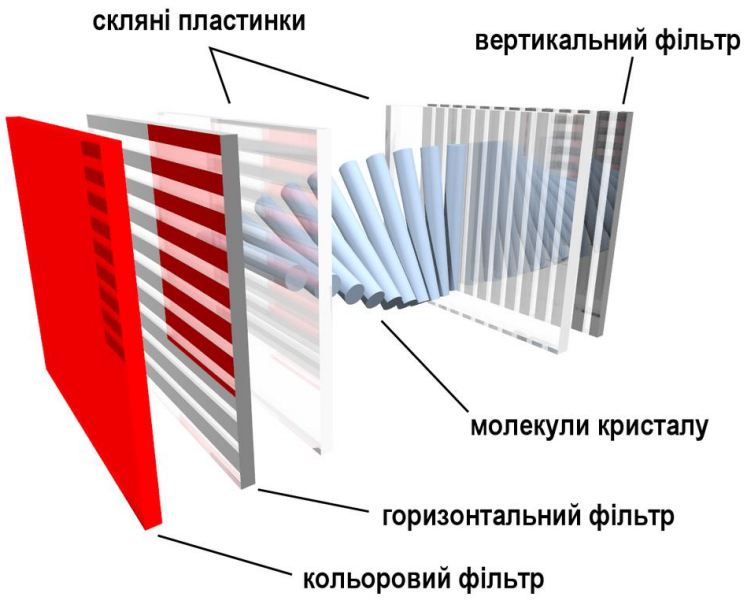
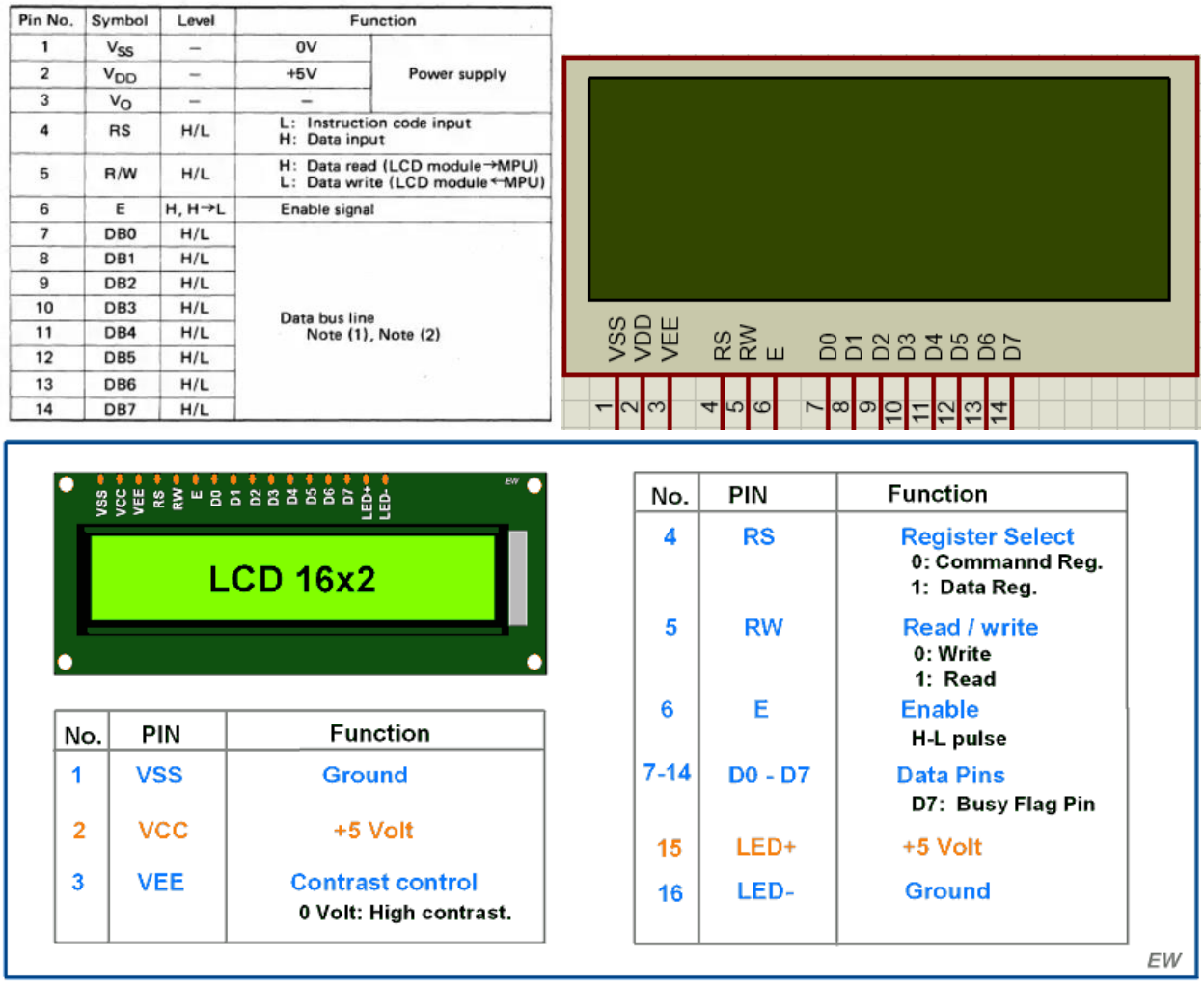



Рис.2.1 – Будова пікселя рідкокристалічного індикатора.

Екран LCD є масивом маленьких сегментів (пікселів), котрими можна маніпулювати для відображення інформації. LCD має кілька шарів, де ключову роль грають дві панелі, зроблені з вільного від натрію і дуже чистого скляного матеріалу, який називають субстратом або підкладкою. Проміжок між шарами заповнений тонким шаром рідкого кристалу. На панелях є борозенки, що надають їм спеціальної орієнтації. Борозенки

розташовані паралельно між собою в межах кожної панелі, але борозенки однієї панелі перпендикулярні до борозенок іншої. Поздовжні борозенки утворюються внаслідок нанесення на скляну поверхню тонких плівок прозорого пластику, що потім спеціальним чином обробляється.

Борозенки орієнтують молекули рідкого кристалу однаково у всіх комірках. Молекули одного з типів рідких кристалів (нематиків) при відсутності напруги повертають вектори електричного (і магнітного) полів світлової хвилі на деякий кут у площині, перпендикулярній до напрямку поширення світлового променя. Нанесення борозенок на поверхню скла дозволяє забезпечити однаковий кут повороту площини поляризації для всіх комірок. Проміжок між панелями дуже тонкий.



| Pin No. | Symbol | Level | Function | |
|---|---|---|---|---|
| 1 | $V_{SS}$ | – | 0V | Power supply |
| 2 | $V_{DD}$ | – | +5V | |
| 3 | $V_O$ | – | – | |
| 4 | RS | H/L | L: Instruction code input<br>H: Data input | |
| 5 | R/W | H/L | H: Data read (LCD module→MPU)<br>L: Data write (LCD module←MPU) | |
| 6 | E | H, H→L | Enable signal | |
| 7 | DB0 | H/L | Data bus line<br>Note (1), Note (2) | |
| 8 | DB1 | H/L | | |
| 9 | DB2 | H/L | | |
| 10 | DB3 | H/L | | |
| 11 | DB4 | H/L | | |
| 12 | DB5 | H/L | | |
| 13 | DB6 | H/L | | |
| 14 | DB7 | H/L | | |



| No. | PIN | Function |
|---|---|---|
| 1 | VSS | Ground |
| 2 | VCC | +5 Volt |
| 3 | VEE | Contrast control<br>0 Volt: High contrast. |

| No. | PIN | Function |
|---|---|---|
| 4 | RS | Register Select<br>0: Commannd Reg.<br>1: Data Reg. |
| 5 | RW | Read / write<br>0: Write<br>1: Read |
| 6 | E | Enable<br>H-L pulse |
| 7-14 | D0 - D7 | Data Pins<br>D7: Busy Flag Pin |
| 15 | LED+ | +5 Volt |
| 16 | LED- | Ground |

Рис.2.2 - Таблиця призначення виводів.

Дисплей LM044L — це текстовий рідкокристалічний дисплей (LCD), який використовується для відображення символів. Він належить до серії дисплеїв з контролерами, які зазвичай підтримують стандартний інтерфейс HD44780. Ось основні характеристики LM044L.

Підтримує як 8-бітний, так і 4-бітний інтерфейс паралельного з'єднання з мікроконтролером, що дає можливість вибирати між швидкістю передачі даних і кількістю необхідних з'єднань. Працює на напрузі 5 В, а також вимагає підключення змінного резистора для регулювання контрастності екрану. В більшості випадків має світлодіодне

підсвічування, яке може бути жовто-зеленого або синього кольору, залежно від версії дисплея. Керування символами, положенням курсора та іншими функціями здійснюється шляхом передачі відповідних команд контролеру дисплея (HD44780 або аналогічному). Це дає змогу очищати екран, переміщувати курсор, вмикати або вимикати підсвітку та інші функції. Підтримує відображення стандартних ASCII символів та спеціальних символів, включаючи символи, що можуть бути запрограмовані користувачем (приблизно 8 користувацьких символів). Для керування LM016L зазвичай потрібно підключити кілька контактів для живлення, даних, а також сигналів керування (RS, RW, E), що дозволяє взаємодіяти з мікроконтролером, наприклад, через шини даних або SPI/I2C за наявності відповідних перетворювачів.

| Character No | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1st Line | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 8A | 8B | 8C | 8D | 8E | 8F | 90 | 91 | 92 | 93 |
| 2nd Line | C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | CA | CB | CC | CD | CE | CF | D0 | D1 | D2 | D3 |
| 3rd Line | 94 | 95 | 96 | 97 | 98 | 99 | 9A | 9B | 9C | 9D | 9E | 9F | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 |
| 4th Line | D4 | D5 | D6 | D7 | D8 | D9 | DA | DB | DC | DD | DE | DF | E0 | E1 | E2 | E3 | E4 | E5 | E6 | E7 |

Рис.2.2 – HEX значення для роботи з позиціонуванням символів.

4-бітний і 8-бітний режими роботи рідкокристалічного дисплея (LCD), наприклад LM044L, визначають спосіб передачі даних між мікроконтролером і дисплеєм.

**8-бітний режим:**

У цьому режимі передача даних відбувається паралельно через 8 ліній даних (D0-D7). Мікроконтролер відправляє одразу всі 8 біт інформації (1 байт) на дисплей за один такт передачі, що робить передачу швидшою. Потрібно використовувати 8 ліній даних, що вимагає більше пінів на мікроконтролері.

**4-бітний режим:**

У цьому режимі передача даних відбувається через 4 лінії даних (D4-D7). Для відправки 1 байта (8 біт) інформації мікроконтролер спочатку передає старші 4 біти, а потім — молодші 4 біти. Тобто для передачі одного байта потрібно дві операції. Використовуються лише 4 лінії даних замість 8, що дозволяє підключити дисплей до мікроконтролера з меншою кількістю доступних пінів. Передача даних займає більше часу, оскільки кожен байт інформації ділиться на дві частини і передається за два цикли.

Порівняння:

8-бітний режим забезпечує швидший обмін даними, але вимагає більше ліній для з’єднання.

4-бітний режим більш економний з точки зору використання пінів, але передача даних відбувається повільніше, оскільки кожен байт ділиться на дві частини.

Незалежно від обраного режиму, функціонал дисплея залишається однаковим. Вибір режиму залежить від доступних ресурсів мікроконтролера.

| No | HEX Value | COMMAND TO LCD |
|---|---|---|
| 1 | 0x01 | Clear Display Screen |
| 2 | 0x30 | Function Set: 8-bit, 1 Line, 5x7 Dots |
| 3 | 0x38 | Function Set: 8-bit, 2 Line, 5x7 Dots |
| 4 | 0x20 | Function Set: 4-bit, 1 Line, 5x7 Dots |
| 5 | 0x28 | Function Set: 4-bit, 2 Line, 5x7 Dots |
| 6 | 0x06 | Entry Mode |
| 7 | 0x08 | Display off, Cursor off |
| 8 | 0x0E | Display on, Cursor on |
| 9 | 0x0C | Display on, Cursor off |
| 10 | 0x0F | Display on, Cursor blinking |
| 11 | 0x18 | Shift entire display left |
| 12 | 0x1C | Shift entire display right |
| 13 | 0x10 | Move cursor left by one character |
| 14 | 0x14 | Move cursor right by one character |
| 15 | 0x80 | Force cursor to beginning of 1st row |
| 16 | 0xC0 | Force cursor to beginning of 2nd row |

Рис.2.3 – HEX значення для роботи з LED дисплеєм.

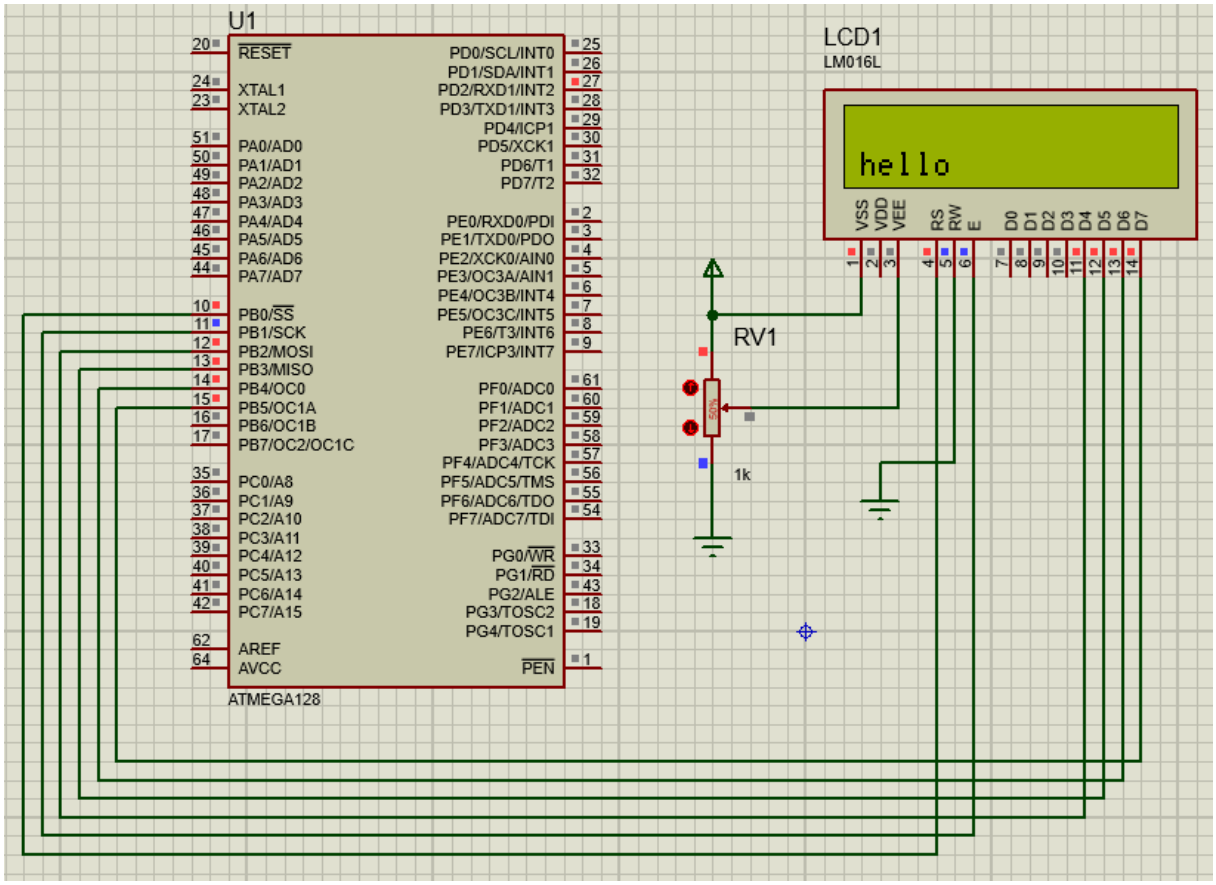



Рис.2.4 – Підключення за допомогою 4х бітів.(Додаток 1)

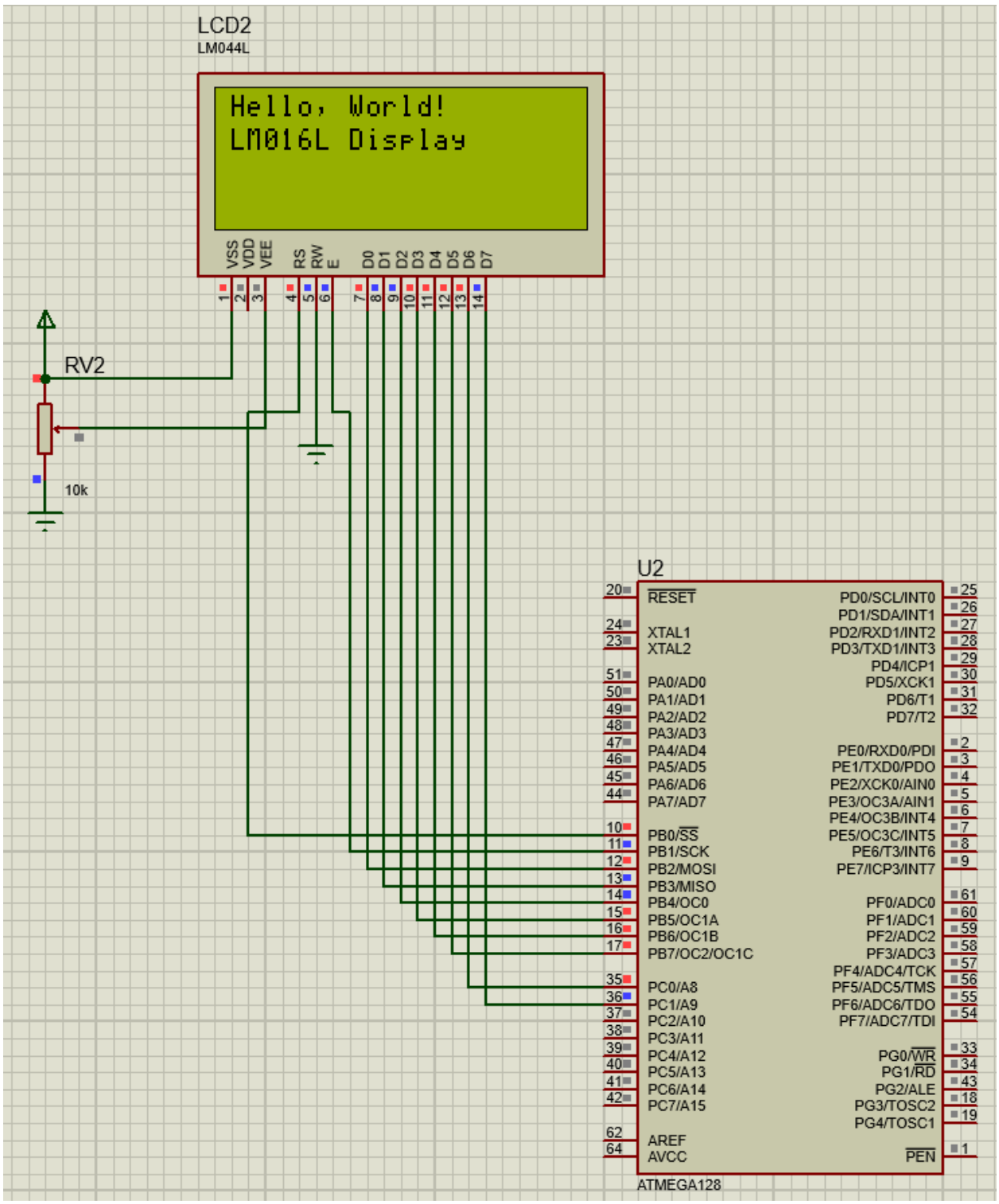



Рис.2.5 – Підключення за допомогою 8 бітів.(Додаток 2)

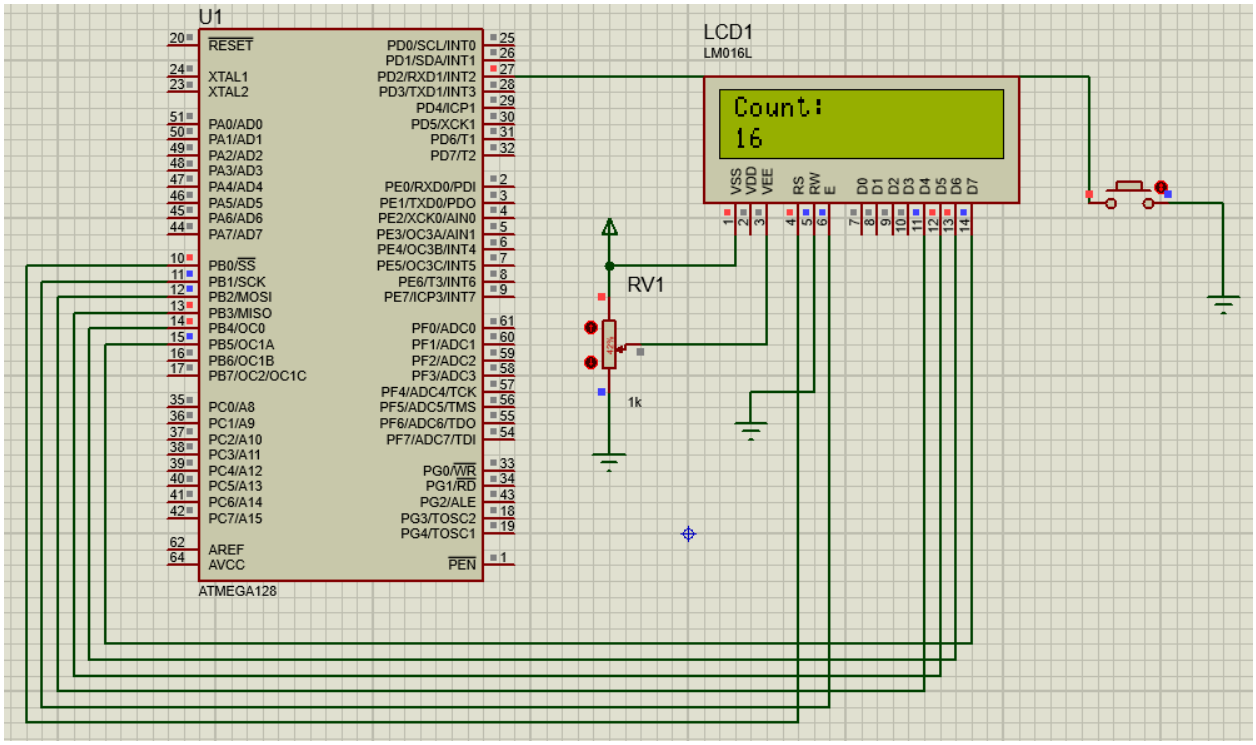



Рис.2.6 – Схема відображення на LCD дисплеї кількості натискань кнопки (Додаток 3).

## 2.2. Порядок виконання лабораторної роботи

1. Ознайомитись з теоретичними відомостями.
2. Зібрати схему підключення на рисунку 2.4 та відобразити (додаток 1) на дисплеї своє прізвище та ім'я положення виводу обрати наступне:
   - Непарний варіант в 1му рядку;
   - Парний варіант в 2му рядку.
3. Зібрати схему підключення на рисунку 2.5 та відобразити (додаток 2) на дисплеї своє прізвище та ім'я положення обрати наступне:
   - 1,5,9,13,17,21 варіант виводить інформацію в 1му рядку з 5го знакомісця та заміняє пробіли → «*» на початку;
   - 2,6,10,14,18,22 варіант виводить інформацію в 2му рядку з 4го знакомісця та заміняє пробіли → «#» на початку;
   - 3,7,11,15,19,23 варіант виводить інформацію в 3му рядку з 3го знакомісця та заміняє пробіли → «@» на початку;
   - 4,8,12,16,20,24 варіант виводить інформацію в 4му рядку з 2го знакомісця та заміняє пробіли → «&» на початку.
4. Зібрати схему підключення на рисунку 2.6 та відобразити на дисплеї своє прізвище та кількість натискань на кнопку та доповнити код (додаток 3).
5. Зробити висновки

## 2.3 Зміст звіту

1. Найменування і мета роботи.
2. Програма та схема виводу інформації на LM016L дисплей.
3. Програма та схема виводу інформації на LM044L дисплей.
4. Програма та схема виводу інформації на дисплей.

5. Висновки по роботі.

## 2.4 Контрольні запитання

1. Які основні характеристики рідкокристалічних дисплеїв LM016L та LM044L? Чим вони відрізняються?

2. Як підключити рідкокристалічний дисплей LM016L до мікроконтролера ATmega128? Опишіть схему підключення.

3. Чим відрізняється 4-бітний та 8-бітний режими роботи дисплея? Який з них ефективніше використовувати, і чому?

4. Які методи можна використовувати для контролю яскравості підсвітки рідкокристалічного дисплея LM016L?

Які команди використовуються для очищення дисплея LM016L і скидання курсора на початок? Яка їхня роль у програмі?

```c
#define F_CPU 1000000UL // Частота мікроконтролера 1 МГц
#include <avr/io.h>
#include <util/delay.h>

// Піни для підключення дисплея
#define RS PB0
#define EN PB1
#define D4 PB2
#define D5 PB3
#define D6 PB4
#define D7 PB5

// Функції для керування дисплеєм
void LCD_Command(unsigned char cmd);
void LCD_Char(unsigned char data);
void LCD_Init(void);
void LCD_String(const char *str);
void LCD_Clear(void);

// Встановлення порту і піна для дисплея
void LCD_Port(unsigned char data) {
	if (data & 1) PORTB |= (1<<D4);
	else PORTB &= ~(1<<D4);

	if (data & 2) PORTB |= (1<<D5);
	else PORTB &= ~(1<<D5);

	if (data & 4) PORTB |= (1<<D6);
	else PORTB &= ~(1<<D6);

	if (data & 8) PORTB |= (1<<D7);
	else PORTB &= ~(1<<D7);
}

void LCD_Command(unsigned char cmd) {
	// Відправляємо команду (RS=0)
	PORTB &= ~(1<<RS);

	// Передаємо старші 4 біти
	LCD_Port(cmd >> 4);
	PORTB |= (1<<EN);
	_delay_ms(1);
	PORTB &= ~(1<<EN);

	// Передаємо молодші 4 біти
	LCD_Port(cmd);
	PORTB |= (1<<EN);
	_delay_ms(1);
	PORTB &= ~(1<<EN);
}

void LCD_Char(unsigned char data) {
	// Відправляємо дані (RS=1)
	PORTB |= (1<<RS);
```

```
        // Передаємо старші 4 біти
        LCD_Port(data >> 4);
        PORTB |= (1<<EN);
        _delay_ms(1);
        PORTB &= ~(1<<EN);

        // Передаємо молодші 4 біти
        LCD_Port(data);
        PORTB |= (1<<EN);
        _delay_ms(1);
        PORTB &= ~(1<<EN);
}

void LCD_Init(void) {
        // Налаштовуємо пін як вихід
        DDRB |= (1<<RS) | (1<<EN) | (1<<D4) | (1<<D5) | (1<<D6) | (1<<D7);

        // Початкова ініціалізація
        LCD_Command(0x02);  // Перемикання в 4-бітний режим
        LCD_Command(0x28);  // 2 рядки, 5x8 точок
        LCD_Command(0x0C);  // Увімкнути дисплей, без курсора
        LCD_Command(0x06);  // Інкрементувати курсор
        LCD_Command(0x01);  // Очистити дисплей
        _delay_ms(2);
}

void LCD_String(const char *str) {
        while (*str) {
                LCD_Char(*str++);
        }
}

void LCD_Clear(void) {
        LCD_Command(0x01);  // Команда очищення дисплея
        _delay_ms(2);
}


int main(void) {
        // Ініціалізація дисплея
        LCD_Init();

        // Виведення рядків на дисплей
        LCD_String("Hello, World!");  // Виводимо текст у першому рядку
        LCD_Command(0xC0);  // Перехід на другий рядок
        LCD_String("LM016L Display");

        while (1) {
                // Бездовільний цикл
        }
                                                }
```

```c
#define F_CPU 1000000UL // Частота мікроконтролера 1 МГц
#include <avr/io.h>
#include <util/delay.h>

#define RS PB0
#define EN PB1
#define D0 PB2
#define D1 PB3
#define D2 PB4
#define D3 PB5
#define D4 PB6
#define D5 PB7
#define D6 PC0
#define D7 PC1

// Функція для передачі 8-бітних даних на порти
void LCD_Port(unsigned char data) {
	// Передаємо перші 6 біти через PORTB (D0-D5)
	if (data & 1) PORTB |= (1<<D0);
	else PORTB &= ~(1<<D0);

	if (data & 2) PORTB |= (1<<D1);
	else PORTB &= ~(1<<D1);

	if (data & 4) PORTB |= (1<<D2);
	else PORTB &= ~(1<<D2);

	if (data & 8) PORTB |= (1<<D3);
	else PORTB &= ~(1<<D3);

	if (data & 16) PORTB |= (1<<D4);
	else PORTB &= ~(1<<D4);

	if (data & 32) PORTB |= (1<<D5);
	else PORTB &= ~(1<<D5);

	// Передаємо біти D6 і D7 через PORTC
	if (data & 64) PORTC |= (1<<D6);
	else PORTC &= ~(1<<D6);

	if (data & 128) PORTC |= (1<<D7);
	else PORTC &= ~(1<<D7);
}

// Функція для відправки команди
void LCD_Command(unsigned char cmd) {
	PORTB &= ~(1<<RS); // RS = 0 для команд

	// Відправляємо 8-бітні дані за один раз
	LCD_Port(cmd);

	PORTB |= (1<<EN);  // Встановлюємо Enable
	_delay_ms(1);
	PORTB &= ~(1<<EN); // Вимикаємо Enable
}
```

```
// Функція для відправки символу
void LCD_Char(unsigned char data) {
	PORTB |= (1<<RS); // RS = 1 для передачі даних

	// Відправляємо 8-бітні дані за один раз
	LCD_Port(data);

	PORTB |= (1<<EN);  // Встановлюємо Enable
	_delay_ms(1);
	PORTB &= ~(1<<EN); // Вимикаємо Enable
}

// Ініціалізація дисплея
void LCD_Init(void) {
	// Встановлюємо пін RS, EN і всі дані як вихід
	DDRB |= (1<<RS) | (1<<EN) | (1<<D0) | (1<<D1) | (1<<D2) | (1<<D3) | (1<<D4) |
(1<<D5);
	DDRC |= (1<<D6) | (1<<D7);  // Налаштовуємо D6 і D7 через PORTC як виходи

	// Початкова ініціалізація дисплея
	LCD_Command(0x38);  // Вибір 8-бітного режиму, 2 рядки, 5x8 точок
	LCD_Command(0x0C);  // Увімкнути дисплей, вимкнути курсор
	LCD_Command(0x06);  // Інкрементувати курсор
	LCD_Command(0x01);  // Очистити дисплей
	_delay_ms(2);
}

// Виведення рядка на дисплей
void LCD_String(const char *str) {
	while (*str) {
		LCD_Char(*str++);
	}
}

// Очищення дисплея
void LCD_Clear(void) {
	LCD_Command(0x01);  // Очищення дисплея
	_delay_ms(2);
}

int main(void) {
	LCD_Init();  // Ініціалізація дисплея

	// Виведення рядків на дисплей
	LCD_String("Hello, World!");  // Виводимо текст у першому рядку
	LCD_Command(0xC0);  // Перехід на другий рядок
	LCD_String("LM044L Display");

	while (1) {
		// Безкінечний цикл, програма тут чекає
	}
                                        }
```

```
#define F_CPU 1000000UL // Частота мікроконтролера 1 МГц
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h> // Для форматування чисел в рядок

// Піни для підключення дисплея
#define RS PB0
#define EN PB1
#define D4 PB2
#define D5 PB3
#define D6 PB4
#define D7 PB5

// Функції для керування дисплеєм
void LCD_Command(unsigned char cmd);
void LCD_Char(unsigned char data);
void LCD_Init(void);
void LCD_String(const char *str);
void LCD_Clear(void);

// Встановлення порту і піна для дисплея
void LCD_Port(unsigned char data) {
	if (data & 1) PORTB |= (1<<D4);
	else PORTB &= ~(1<<D4);

	if (data & 2) PORTB |= (1<<D5);
	else PORTB &= ~(1<<D5);

	if (data & 4) PORTB |= (1<<D6);
	else PORTB &= ~(1<<D6);

	if (data & 8) PORTB |= (1<<D7);
	else PORTB &= ~(1<<D7);
}

void LCD_Command(unsigned char cmd) {
	// Відправляємо команду (RS=0)
	PORTB &= ~(1<<RS);

	// Передаємо старші 4 біти
	LCD_Port(cmd >> 4);
	PORTB |= (1<<EN);
	_delay_ms(1);
	PORTB &= ~(1<<EN);

	// Передаємо молодші 4 біти
	LCD_Port(cmd);
	PORTB |= (1<<EN);
	_delay_ms(1);
	PORTB &= ~(1<<EN);
}

void LCD_Char(unsigned char data) {
	// Відправляємо дані (RS=1)
	PORTB |= (1<<RS);
```

```
        // Передаємо старші 4 біти
        LCD_Port(data >> 4);
        PORTB |= (1<<EN);
        _delay_ms(1);
        PORTB &= ~(1<<EN);

        // Передаємо молодші 4 біти
        LCD_Port(data);
        PORTB |= (1<<EN);
        _delay_ms(1);
        PORTB &= ~(1<<EN);
}

void LCD_Init(void) {
        // Налаштовуємо пін як вихід
        DDRB |= (1<<RS) | (1<<EN) | (1<<D4) | (1<<D5) | (1<<D6) | (1<<D7);

        // Початкова ініціалізація
        LCD_Command(0x02);  // Перемикання в 4-бітний режим
        LCD_Command(0x28);  // 2 рядки, 5x8 точок
        LCD_Command(0x0C);  // Увімкнути дисплей, без курсора
        LCD_Command(0x06);  // Інкрементувати курсор
        LCD_Command(0x01);  // Очистити дисплей
        _delay_ms(2);
}

void LCD_String(const char *str) {
        while (*str) {
                LCD_Char(*str++);
        }
}

void LCD_Clear(void) {
        LCD_Command(0x01);  // Команда очищення дисплея
        _delay_ms(2);
}

int main(void) {
        // Додати код для відображення LCD скільки разів було натиснуто кнопку
}
```