Лабораторна робота № 5 РІДКОКРИСТАЛІЧНИЙ ІНДИКАТОР

Мета роботи: ознайомитись з базовими типами мікроконтролерів AVR. Набути навиків роботи з рідкокристалічними індикаторами, способів виведення текстової інформації на табло, робота з рядками.

Обладнання: навчально-відлагоджувальна плата AVR-Easy; мікроконтролери ATmega16; середовища програмування AVR Studio 4.19; внутрішньосхемний програматор.

Теоретичний матеріал

Алфавітно-цифрові рідкокристалічні індикатори

У даній роботі розглядаються алфавітно-цифрові рідкокристалічні індикатори (LCD) на основі контролера, сумісного з HD44780 (WH1602 від Winstar'a). Вони відображають символи ASCII з кодами від 32 до 122 та деякі інші, залежно від виробника та моделі. Також можна запрограмувати свої символи.

На навчальній платі встановлено індикатор розміру 16x2. LCD під'єднаний по 4-бітному інтерфейсу до порту A.

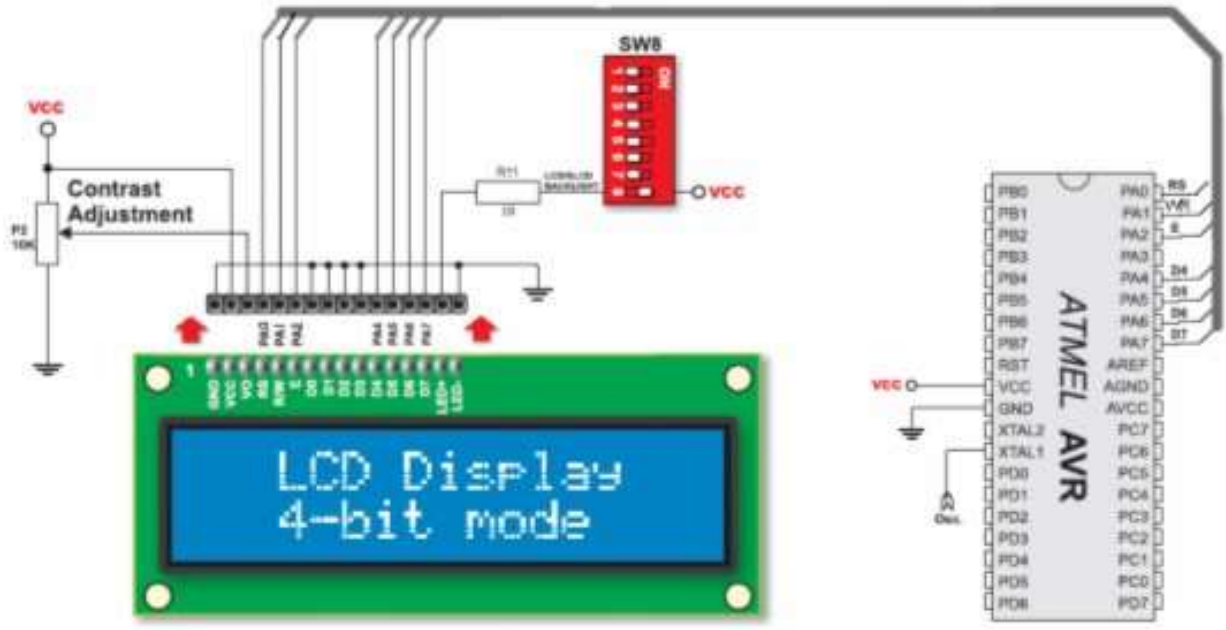

Рис. 2.9. Схема підключення алфавітно-цифрового рідкокристалічного індикатора до мікроконтролера на навчально-відлагоджувальній платі AVR- Easy-Kit

Входи/виходи дисплея

Керуючі входи

RS - Register Select За допомогою цього виводу ми повідомляємо дисплею який тип даних буде записаний / прочитаний. RS = 0 , працюємо з регістром команд (Instruction Register) , RS = 1 , працюємо з регістром даних (Data Register)

RW - Read / WriteЗа допомогою цього висновку, ми перемикаємо дисплей в режим запису або читання. RW = 0 , записуємо дані / команди в дісплей RW = 1 , читаємо дані / змінні з дісплей

E - Enable За допомогою цього висновку, активізується виконання операції запису / читання команд / даних. Або іншими словами, на цей висновок подається "стробірующій сигнал", без якого не може бути виконана жодна операція.Виконання операцій дисплеєм, починаються при спадающем фронті

У більшості випадків читати дані / параметри не приходиться - так що висновок RW можна сміливо підключати до Vss (земля), тобто дисплей весь час буде працювати в режимі запису (Write mode).

Шина даних / адрес

DB7 .. DB0 - Data Bus Символьний дисплей може працювати як з 8-бітної шиною даних / адрес, так і з 4-бітної шиною даних / адрес - що дозволяє заощадити дорогоцінні висновки мікроконтролера. Шина підключається безпосередньо до мікроконтролера, без жодних додаткових перетворень логічних рівнів, вона толерантна як 5В так і до 3.3В -. DB7 - найбільш значущий біт. DB0 - найменш значущий біт.

Якщо використовується 4-бітна шина даних, то в цьому випадку використовуються останні (старші) чотири біта:DB4 .. DB7 , а перші чотири підключаються до землі.

Пам'ять індикатора ділиться на три складові частини:DDR RAM (Display Data RAM), призначена для зберігання 8-бітних символів (в основному ASCII), які ми хочемо відображати на екрані.

Ємність цієї пам'яті становить 80 символів, по 40 на кожен рядок. Решта символи приховані.

Щоб їх відобразити, слід призначити іншу ділянку DDRAM пам'яті (2 × 16 символів) як видиму, за замовчуванням видима пам'ять починається з адреси 0 × 00.

CGROM (Character Generator ROM), тут зберігається розшифровка записаних в DDRAM символів. Тобто, коли ми записуємо в DDRAM комірку, скажімо, символ 0 × 41, то на екрані з'явиться символ зберігається в комірці CGROM пам'яті з адресою 0 × 41 - "A". Як і ASCII таблиці для різних країн, символи зберігаються в CGROM пам'яті відрізняються, так що вибирайте індикатор з потрібною вам ASCII таблицею (CGROM пам'яттю).

CGRAM (Character Generator RAM), в загальному це маленький ділянка CGROM-пам'яті в якій немає ніяких символом і яку можна змінювати - перші 64 байта CGROM пам'яті. Так що користувач може намалювати свої символи. Для того щоб пам'ятати адресу останньої клітинки, до якої ми зверталися, є спеціальний регістр - address counter, за замовчуванням він вказує на комірку 0 × 00, DDRAM пам'яті. Після кожного звернення до пам'яті він автоінкрементіруется або декрементируется в залежності від налаштувань режиму введення.системи команд.

Дисплей розпізнає всього 11 команд, в які входять і команди ініціалізації

дисплея.

Бібліотека для роботи з РК-дисплеєм

Для зручності написання програм з використанням РК індикаторі - доречно винести всі команди і процедури для роботи з індикатором в окрему бібліотеку. Для функціонування даної бібліотеки потрібно щоб в основній програмі була підключена бібліотека затримок “util/delay.h”

Особливості

- Працює з компіляторами IAR AVR, CodeVision AVR, GNU GCC,
- Підтримує lcd контролери HD44780 і KS0066,
- Підтримує підключення lcd до довільних виводів мікроконтролера,
- Підтримує 4-х і 8-ми розрядний інтерфейс,
- Має функції виведення рядків з ОЗУ і флеш,
- Має функції додавання призначених для користувача символів.

Склад бібліотеки

compilers_4.h - файл для підтримки трьох компіляторів

port_macros.h - макроси віртуальних портів

lcd_lib_2.h - заголовки LCD бібліотеки з прототипами функцій і настройками

lcd_lib_2.c - файл реалізації функцій LCD бібліотеки

Підключення до проекту

1. Переписуємо всі файли бібліотеки в папку проекту.
2. Підключаємо lcd_lib_2.c до проекту всередині середовища розробки.
3. Вставляємо заголовок lcd_lib_2.h до Сі файлу, в якому будуть використовуватися lcd функції.
4. Налаштовуємо конфігурацію lcd в заголовки lcd_lib_2.h
5. Прописуємо в код виклик функцій lcd бібліотеки.

Налаштування конфігурації

Налаштування конфігурації у файлі lcd_lib_2.h включає в себе наступні кроки.

1. Налаштування віртуального або реального порту, до якого підключається LCD

Синтаксис оголошення віртуального порту докладно описаний у файлі port_macros.h. У заголовки lcd_lib_2.h вже оголошений порт, в цих оголошення потрібно міняти тільки букви порту (A, B, C ..), номери виводів(0, 1, 2, 3 ...), тип порту (_REAL, _VIRT) , активний рівень (_HI, _NONE). Все інше (ім'я порту та імена висновків) чіпати не треба.

Приклад оголошення віртуального порту для 8-ми бітної шини і реального порту для 4-х бітної шини.

```
// виртуальный порт

//шина данных LCD
#define LCD_PORT  LCD_DATA, F, _VIRT

#define LCD_DATA_0  D, 0, _HI
#define LCD_DATA_1  D, 1, _HI
#define LCD_DATA_2  D, 2, _HI
#define LCD_DATA_3  C, 5, _HI
#define LCD_DATA_4  C, 6, _HI
#define LCD_DATA_5  C, 7, _HI
#define LCD_DATA_6  B, 4, _HI
#define LCD_DATA_7  B, 5, _HI

//управляющие выводы LCD
#define LCD_RS  C, 0, _HI
#define LCD_RW  C, 1, _HI
#define LCD_EN  C, 2, _HI
```

```
// реальный порт

//шина данных LCD
#define LCD_PORT  LCD_DATA, A, _REAL

#define LCD_DATA_0  A, 0, _NONE
#define LCD_DATA_1  A, 1, _NONE
#define LCD_DATA_2  A, 2, _NONE
#define LCD_DATA_3  A, 3, _NONE
#define LCD_DATA_4  A, 4, _HI
#define LCD_DATA_5  A, 5, _HI
#define LCD_DATA_6  A, 6, _HI
#define LCD_DATA_7  A, 7, _HI

//управляющие выводы LCD
#define LCD_RS  A, 0, _HI
#define LCD_RW  A, 1, _HI
#define LCD_EN  A, 2, _HI
```

Рамками виділені ті частини коду, які потрібно налаштовувати під свій проект.

2. Глобальні налаштування драйвера

LCD_CHECK_FL_BF - перевіряти прапор BF або використовувати програмну затримку. 0 - затримка, 1 - перевірка прапора.

LCD_BUS_4_8_BIT - використовувана шина даних. 0 - 4 розрядна шина, 1 - 8-ми розрядна

3. Налаштування ініціалізації дисплея

Ці установки визначають стан дисплей після виклику функції LCD_Init ().

LCD_ONE_TWO_LINE - кількість відображуваних рядків. 0 - 1 рядок; 1 - 2 рядки.

LCD_FONT58_FONT511 - тип шрифту. 0 - 5x8 точок; 1 - 5x11 точок.

LCD_DEC_INC_DDRAM - зміни адреси ОЗУ при виведенні на дисплей. 0 - курсор рухається вліво, адреса зменшується на 1 (текст виходить задом наперед); 1 - курсор рухається вправо, адреса збільшується на 1.

LCD_SHIFT_RIGHT_LEFT - зсув всього дисплея. 0 - при читанні ОЗУ зрушення не виконується, 1 - під час запису в ОЗУ зрушення дисплея виконується згідно з настановою LCD_DEC_INC_DDRAM (0 - зсув вправо, 1 - зсув вліво)

LCD_DISPLAY_OFF_ON - включення / вимикання дисплея. 0 - дисплей вимкнений, але дані в ОЗУ залишаються; 1 - дисплей включений.

LCD_CURSOR_OFF_ON - відображення підкреслює курсору. 0 - курсор не відображається, 1 - курсор відображається.

LCD_CURSOR_BLINK_OFF_ON - відображення миготливого курсору. 0 - миготливий курсор не відображається; 1 - миготливий курсор відображається.

LCD_CURSOR_DISPLAY_SHIFT - команда зсуву вправо / вліво курсора або дисплея без запису на дисплей.

Призначені для користувача макроси і функції

LCD_Clear () - очищення дисплея.

LCD_ReturnHome () - повернення курсору в початкове положення.

LCD_Goto (x, y) - позиціонування курсору. x - номер знакоместа, y - номер рядка.

void LCD_Init (void) - ініціалізація дисплея.

void LCD_WriteCom (uint8_t data) - запис команди

void LCD_WriteData (char data) - вивід одного символу

void LCD_SendStr (char * str) - вивід рядка з ОЗП.

void LCD_SendStrFl (char __flash * str) - вивід рядка з флеш пам'яті.

void LCD_SetUserChar (uint8_t __flash * sym, uint8_t adr) - завантаження споживацького символу в ОЗУ дисплея..

```
#include <util/delay.h>
#include <avr/io.h>
#include "lcd_lib_2.h"
char text Compiler [] = "xz";
__flash uint8_t quarter Note [] = {4,4,4,4,4,4,28,28};
int main (void)
{// Инициализируем дисплей
LCD_Init (); // Завантажуємо користувальницький символ в нульову комірку ОЗП дисплея
LCD_SetUserChar (quarterNote, 0);
// Встановлюємо курсор в 8-ме знакомісце
LCD_Goto (8,0);
// Виводимо рядок на дисплей
LCD_SendStr (textCompiler);
while (1);
return 1;
}.
```

Виведення тексту на LCD

Дана програма ініціалізує LCD і виводить текст “Hello, LCD!!! 16 chars, 2 lines.”

```
#include <util/delay.h>
#include <avr/io.h>
#include "lcd_lib_2.h"
{
DDRB = PORTB = 0xFF;
unsigned char text[] = "—Hello, LCD!!=-16 chars,2 lines.";
LCD_Init ();  //Ініціалізація РК-екрану

LCD_WriteCom (0x80);

//Переведення курсору на початок першого рядка

 for(unsigned char i=0; i<32; i++)
{ if(i == 16)
```

```
        LCD_WriteCom (0xC0);
//Переведення курсору на другий рядок, якщо текст виходить за межі першого
        LCD_WriteData (text[i]); //Виведеня і-го символу з масиву

        }


        for(;;);

        }
```

У нескінченному циклі процесор нічого не робить, але вилучити цикл не можна, щоб МК не скидався і не починав виконувати усю програму спочатку.

В кінці функції main() обов'язково повинен бути нескінченний цикл.

Символи кирилиці

Записувати символи як рядок у програмі дуже зручно, проте це працює лише для символів з кодами ASCII від 32 до 122. Решта символів, зокрема, букви кирилиці, закодовані не так, як у комп'ютері. Щоб правильно вивести їх на екран LCD, необхідна програма, що перекодовує символи. Її можна написати самому. Результатом програми є масив, який необхідно скопіювати у свою програму.

Рядок, що біжить

Програма показує у першому рядку заголовок “-== Планети ==-”, а другий рядок біжить: “Сонячна система містить 8 планет: Меркурій, Венера, Земля, Марс, Юпітер, Сатурн, Уран, Нептун.”.

```
        #include <util/delay.h>
        #include <avr/io.h>
        #include "lcd_lib_2.h"

        unsigned char text[ ] = { '-', 0x3D, 0x3D, 0x20, 0xA8, 0xBB, 0x61, 0xBD,
0x65, 0xBF, 0xB8, 0x20, 0x20, 0x3D, 0x3D, '-' };
        unsigned char text1[] = { 0x20, 0x20, 0x43, 0x6F, 0xBD, 0xC7, 0xC0, 0xBD,
0x61, 0x20, 0x63, 0xB8, 0x63, 0xBF, 0x65, 0xBC, 0x61, 0x20, 0xBC, 0x69, 0x63,
0xBF, 0xB8, 0xBF, 0xC4, 0x20, '8', 0x20, 0xBE, 0xBB, 0x61, 0xBD, 0x65, 0xBF,
0x3A, 0x20, 0x4D, 0x65, 0x70, 0xBA, 0x79, 0x70, 0x69, 0xB9, 0x2C, 0x20, 0x42,
0x65, 0xBD, 0x65, 0x70, 0x61, 0x2C, 0x20, 0xA4, 0x65, 0xBC, 0xBB, 0xC7, 0x2C,
0x20, 0x4D, 0x61, 0x70, 0x63, 0x2C, 0x20, 0xB0, 0xBE, 0x69, 0xBF, 0x65, 0x70,
0x2C, 0x20, 0x43, 0x61, 0xBF, 0x79, 0x70, 0xBD, 0x2C, 0x20, 0xA9, 0x70, 0x61,
0xBD, 0x2C, 0x20, 0x48, 0x65, 0xBE, 0xBF, 0x79, 0xBD,'.' };

        int main(void)
        {
        DDRB = PORTB = 0xFF;
        LCD_Init ();          //Ініціалізація РК-екрану
        LCD_WriteCom (0x80); //Переведення курсору на початок першого рядка
```

```
for(unsigned char i=0; i<16; i++)
        LCD_WriteData (text[i]); //Виведення першого напису
        for(;;)
        { for(unsigned char offset=0; offset<=sizeof(text1)-16; offset++)
           { LCD_WriteCom (0xC0);//Переведення курсору на початок другого рядка
for(unsigned char i=0; i<16; i++) LCD_WriteCom (text1[offset+i]);
           //Виведення 16- ти символів
           for(unsigned char i=0; i<40; i++) _delay_ms(10); //delay 400 ms
           }
        }

        }
```

Оскільки значення першого рядка не змінюються, він передається у LCD один раз на початку програми. У нескінченному циклі передається лише значення другого рядка.

Оператор sizeof(text) визначає довжину масиву text. Якби масив був описаний у вигляді рядка (рядок в Сі — це масив, що закінчується нуль- символом — символом з кодом 0), його довжина виявилась би на одиницю більша, тому що нуль-символ в кінці також враховується в довжину рядка.

Хід роботи

1. Знайти на навчальній платі рідкокристалічний індикатор.
2. Розібратися з принципом роботи тестових програм та призначенням кожного оператора у програмі.
3. Створити свою власну програму, яка робить рядок, що біжить. Текст, який біжить, має складатися з Вашого прізвища та будь-якого речення кирилицею. Назва тексту відображається на першому рядку. Скомпілювати програму.
4. Запрограмувати МК, перевірити правильність виконання.

Контрольні запитання

1. Якими можливостями володіють алфавітно-цифрові рідкокристалічні індикатори? Які ще є рідкокристалічні індикатори?
2. Привести існуючі схемні рішення для підключення алфавітно-цифрових рідкокристалічних індикаторів.
3. Яке призначення кожного з пінів алфавітно-цифрових рідкокристалічних індикаторів.
4. Який алгоритм роботи з алфавітно-цифровим рідкокристалічним індикатором?
5. Пояснити кожну з процедур для роботи з алфавітно-цифровим рідкокристалічним індикатором.

Зміст звіту

1. Тема та мета роботи.

2. Перелік використаного обладнання.
3. Стислий зміст теоретичних відомостей.
4. Лістинг власної програми з детальним поясненням кожного рядка.
5. Відповіді на контрольні запитання.
6. Висновки

Розшифровка команд для LCD

Команда LCD	HEX - код	Виконувані дії	Час виконання., мкс
Очистка дисплею	0x01	Порожній екран, очистка пам'яті, курсор в лівій верхній позиції	1640
Повернення курсору на початок	0x02	Курсор в лівій верхній позиції, пам'ять не очищається	1640
Здвиг курсору вліво	0x04	Після виводу чергового символу курсор автоматично здвигається на одне знакомісце вліво	40
Здвиг курсору вправо	0x06	Після виводу чергового символу курсор автоматично здвигається на одне знакомісце вправо	40
Вимкнення дисплею	0x08	Повна відсутність зображення на екрані LCD	40
Вимкнення курсору	0x0C	Дозволено виведення зображення, проте курсор не видно	40
Прямокутна форма курсору	0x0D	Дозволено виведення зображення, курсор у вигляді блимаю чого чорного прямокутника	40
Лінійна форма курсору	0x0E	Дозволено виведення зображення, курсор у вигляді нижньої підрядкової немигаючої лінії	40
Комплексна форма курсору	0x0F	Дозволено виведення зображення, курсор у вигляді блимаю чого чорного прямокутника з підкресленням	40
Інтерфейс 4 біта, 1 рядок	0x20	Зв'язок з однорядковим LCD через 4 лінії шини даних	40
Інтерфейс 4 біта, 2 рядки	0x28	Зв'язок з дворядковим LCD через 4 лінії шини даних	40
Інтерфейс 8 біт, 1 рядок	0x30	Зв'язок з однорядковим LCD через 8 ліній шини даних	40
Інтерфейс 8 біт, 2 рядки	0x38	Зв'язок з дворядковим LCD через 8 ліній шини даних	40
Доступ до ОЗП знакогенератора	0x40 - 0x7F	Запис даних по цим адресам дозволяє створити 16 власних символів	40
Установка позиції курсору	0x80 - 0xCF	Курсор встановлюється в позицію згідно рис. 2	40

Розподіл адрес на рядках екрану

Верхній рядок LCD

0x80	**0x81**	**0x32**	**0x33**	**0x84**	**0x85**	**0x86**	**0x37**	0x88	**0x89**	**0x8A**	**Ox3B**	**Ox3C**	**0x80**	**0x8 E**	**0x8F**
OxCO	**0xC1**	**0xC2**	**OxC3**	**0xC4**	**0xC5**	**OxCб**	0xC7	**0xC8**	**0xC9**	**OxCA**	**OxCB**	**OxCC**	**OxCD**	**OxCE**	**OxCF**

Нижній рядок LCD

Відповідність символів знакогенератора до адрес в пам’яті ПЗП
ПРИКЛАД: код 0x34 відповідає цифрі “ 4 код 0xB9 - букві “ й ”