



## **Вивід інформації на LCD-дисплей**

**Інструкція до лабораторної роботи № 4**  
з курсу “Програмування мікроконтролерів систем автоматики”

для студентів базового напрямку  
050201 “Системна інженерія”

Затверджено  
на засіданні кафедри  
“Комп’ютеризовані  
системи автоматики”  
Протокол № 3 від 10.10.2012

**Львів 2012**

Вивід інформації на LCD-дисплей: Інструкція до лабораторної роботи № 4 з курсу “Програмування мікроконтролерів систем автоматики” для студентів базового напрямку 050201 “Системна інженерія” / Укл.: А.Г. Павельчак, В.В. Самотий – Львів: Львівська політехніка. – 2012. – 28 с.

Укладачі: А.Г. Павельчак, к.т.н., доцент  
В.В. Самотий, д.т.н., професор

[bilyj@ukr.net](mailto:bilyj@ukr.net)

Відповідальний за випуск:  
А.Й. Наконечний, д.т.н., професор

Рецензент: З.Р. Мичуда, д.т.н., професор

**Мета роботи:** ознайомитися зі структурою LCD-дисплеїв на основі контролера HD44780, способами підключення дисплеїв до мікроконтролерів та програмними алгоритмами їхньої роботи, а також отримати навички програмування пристроїв, що мають ввід інформації через клавіатуру 4x4 та вивід на LCD-дисплей.

## Вступ

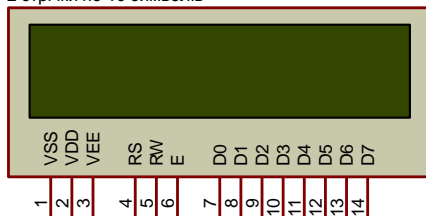
Контролер HD44780 фірми Hitachi фактично є промисловим стандартом і широко застосовується при виробництві алфавітно-цифрових LCD-модулів. Аналоги цього контролера, або сумісні з ним по інтерфейсу і командній мові мікросхеми, випускають безліч фірм, серед яких: Epson, Toshiba, Sanyo, Samsung, Philips. Ще більша кількість фірм виробляють LCD-модулі на базі даних контролерів. Ці модулі можна зустріти в найрізноманітніших пристроях: вимірювальних приладах, медичному обладнанні, примисловому і технологічному обладнанні, офісній техніці – принтерах, телефонах, факсимільних і копіювальних апаратах.

Алфавітно-цифрові LCD-модулі являють собою недороге і зручне рішення, що дозволяє заощадити час і ресурси при розробці нових виробів, при цьому забезпечують відображення великого обсягу інформації при гарному візуальному сприйнятті і низькому енергоспоживанні. Можливість оснащення LCD-модулів заднім підсвічуванням дозволяє експлуатувати їх в умовах зі зниженою або нульовою освітленістю, а виконання з розширеним діапазоном температур ( $-20^{\circ}\text{C}$  ...  $+70^{\circ}\text{C}$ ) – в складних експлуатаційних умовах, в тому числі в переносній, польовий і навіть, іноді, в бортовій апаратурі.

Контролер HD44780 потенційно може керувати 2-ма рядками по 40 символів в кожному (для модулів з 4-ма рядками по 40 символів використовуються два однотипних контролера), при матриці символу 5x8 точок. Контролер також підтримує символи з матрицею 5x10 точок, але зараз LCD-модулі з такою матрицею практично не зустрічаються, тому можна вважати, що фактично бувають тільки символи 5x8 точок.

## 1. Підключення до AVR мікроконтролерів LCD-дисплею HD44780

2 стрічки по 16 символів



### Підключення.

LCD на базі HD44780 підключається до AVR мікроконтролера напряму до портів за допомогою 8-ми або 4-бітної синхронної шини даних/адрес та додаткових 3-х керуючих сигналів.

#### Значення виводів LCD-дисплею:

- Виводи D7...D0 – шина даних/адреси.
- E – стробуючий вхід. Імпульсом напруги (визначеної тривалості) на цій лінії ми даємо вказівку дисплею, що потрібно забирати/віддавати дані з/на шину даних.
- RW – визначає в якому напрямку рухаються дані. Якщо 1 – то на читання з дисплею, якщо 0 – то на запис у дисплей.
- RS – визначає що передається: команда (RS=0) або дані (RS=1). Дані будуть записані у пам'ять за поточною адресою, а команда виконається контролером.
- VSS (GND) – мінус, він же загальний.
- VDD – плюс живлення, як правило +5V.
- V0 (VEE) – вхід контрастності. Сюди подається напруга від нуля до напруги живлення за допомогою змінного резистора, тим самим встановлюючи контрастність зображення. Необхідно вибрати значення максимального контрасту, але щоб не був видимий знаоміст (сірий ореол із квадратів навколо символу). Якщо ж виставити занадто малий контраст, то символи будуть перемикатися повільно.
- A – це вхід Анода світодіодної підсвідки (плюс).
- K – Катод підсвідки (мінус). Потрібно додатково ще встановлювати зовнішній резистор для обмеження струму.

## Структура LCD контролера HD44780.

Контролер має вбудований блок керування, який обробляє команди та працює з пам'яттю. Пам'ять поділяється на три типи:

**DDRAM** (display data RAM) – пам'ять дисплею. Містить 80 байтів, тобто 80 однобайтних символів (по 40 у кожній стрічці). Видима частина LCD може відображати лише 2 стрічки по 16 символів з пам'яті. Адресація видимої частини пам'яті наведена нижче на рисунку. Однак за допомогою відповідних команд можна виконувати зсув видимої частини у пам'яті DDRAM.

Display position DDRAM address

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

2-Line by 16-Character Display

Все, що запишеться в DDRAM буде виведено на екран. Наприклад, якщо записати код 0x31 — на екрані з'явиться символ «1», оскільки 0x31 це ASCII код цифри 1.

**CGROM** (Character Generator ROM) – постійна таблиця символів. Коли ми записуємо в комірку DDRAM байт, то з таблиці береться символ і відображається на екрані. CGROM не можна змінити.

**CGRAM** (Character Generator RAM) – змінна таблиця символів (для створення своїх символів). Адресується лінійно, тобто, спочатку йде 8 байт одного символу (пострічково) – один біт рівний одній точці на екрані. Потім другий символ таким самим чином. Оскільки знакове місце в нас 5 на 8 точок, то старші три біти значення не мають. Усього в CGRAM може бути 8 символів, відповідно CGRAM має 64 байта пам'яті. Ці програмовані символи мають коди від 0x00 до 0x07. Так що, закинувши, наприклад, у перші 8 байт CGRAM (перший символ з кодом 00) які-небудь символи, і записавши в DDRAM нуль (код першого символу в CGRAM), ми побачимо на екрані наше зображення.

0	0	0	0	0	1	0	0
0	0	0	0	0	1	1	0
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0
0	0	0	1	1	1	1	0
0	0	0	0	0	0	0	0

Рис. 1. Формування символу в комірці CGRAM

Upper 4 bit Lower 4 bit	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0 CG RAM (1)			0	a	P	`	P				Б	Ю	Ч	.	Д	М
1 CG RAM (2)			!	1	Q	a	a				Г	Я	Ш	.	О	К
2 CG RAM (3)			"	2	B	R	b	r			Є	Є	ь	ь	Ш	К
3 CG RAM (4)			#	3	C	S	c	s			В	В	ь	ь	Ш	К
4 CG RAM (5)			\$	4	D	T	d	t			З	Г	ь	ь	Ш	К
5 CG RAM (6)			%	5	E	U	e	u			И	Є	ь	ь	Ш	К
6 CG RAM (7)			&	6	F	V	f	v			О	ь	ь	ь	Ш	К
7 CG RAM (8)			'	7	G	W	g	w			Л	Б	ь	ь	Ш	К
8 CG RAM (1)			(	8	H	X	h	x			П	ь	ь	ь	Ш	К
9 CG RAM (2)			)	9	I	Y	i	y			У	ь	ь	ь	Ш	К
A CG RAM (3)			*	:	J	Z	j	z			Ф	ь	ь	ь	Ш	К
B CG RAM (4)			+	;	K	O	k	o			Ч	ь	ь	ь	Ш	К
C CG RAM (5)			,	<	L	* <sup>1</sup>	l	*			Ш	ь	ь	ь	Ш	К
D CG RAM (6)			—	=	M	O	m	o			ь	ь	ь	ь	Ш	К
E CG RAM (7)			.	>	N	^	n	^			ь	ь	ь	ь	Ш	К
F CG RAM (8)			/	?	O	L	o	e			Э	ь	ь	ь	Ш	К

A=0x41

Рис. 2. Таблиця символів (CGROM)

### Доступ до пам'яті.

Ми командою вибираємо в яку саме пам'ять і, починаючи з якої адреси, будемо писати. А потім просто шлемо байти. Якщо зазначено, що записуємо в DDRAM, то на екран (або в сховану область) будуть записуватися символи, якщо в CGRAM, то байти запишуться вже у пам'ять знакогенератора. Головне потім не забути перемкнутися назад на область DDRAM.

### Система команд.

Система команд є простою. Про те, що передається команда контролеру дисплею, інформує вивід RS=0. Сама команда складається зі старшого біта, що визначає за що відповідає дана команда, й бітів параметрів, що вказують контролеру HD44780, що робити.

Таблиця команд:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Команда
0	0	0	0	0	0	0	0	0	1	Очищення екрана. Лічильник адреси на 0 позицію DDRAM
0	0	0	0	0	0	0	0	1	–	Адресація DDRAM, скидання зсувів, Лічильник адреси = 0
0	0	0	0	0	0	0	1	I/D	S	Настроювання зсувів екрана й курсору
0	0	0	0	0	0	1	D	C	B	Настроювання режиму відображення
0	0	0	0	0	1	S/C	R/L	–	–	Зсув курсора або екрана, залежно від бітів
0	0	0	0	1	DL	N	F	–	–	Вибір числа ліній, ширини шини й розміру символу
0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Перемкнути адресацію на SGRAM і задати адресу в CGRAM
0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Перемкнути адресацію на DDRAM і задати адресу в DDRAM
0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Якщо BF=1, то контро- лер зайнятий, а також читання поточ- ної адреси лічильника
1	0	D7	D6	D5	D4	D3	D2	D1	D0	Запис даних у внутрішню RAM (DDRAM/CGRAM)
1	1	D7	D6	D5	D4	D3	D2	D1	D0	Запис даних з внутрішньої RAM (DDRAM/CGRAM)

### Значення окремих бітів:

I/D	1 – збільшення, 0 – зменшення лічильника адреси.
S	0 – зсув екрану не відбувається, 1 – з кожним новим символом буде зсуватися вікно екрана аж до кінця DDRAM.
D	1 – включити дисплей, 0 – зображення вимкнене, і ми можемо у відеопам'яті робити зміни, й вони не будуть видимі.
C	1 – включити курсор у вигляді підкреслення, 0 – вимнений.
B	1 – мигаючий курсор у вигляді чорного квадрата, 0 – вимкн.
S/C	1 – зсув екрану, 0 – зсув курсору. По одному разу за команду.
R/L	1 – зсув вправо курсору чи екрану, 0 – зсув вліво.
D/L	ширина даних: 1 – 8-розрядна шина даних, 0 – 4-розрядна.
N	число рядків: 1 – два рядки, 0 – один рядок.
F	розмір символу: 0 – 5x8 точок, 1 – 5x10 (використов. рідко).
AC	адреса у пам'яті DDRAM/CGRAM

### Робота з LCD-дисплеєм.

У першу чергу необхідно провести ініціалізацію дисплею, враховуючи режим підключення шини даних: 8-ми чи 4-бітний.

У пакеті моделювання Proteus використовується стандартна модель дисплею з контролером HD44780. Модель RC1602A-GGN-CSX (Raystar), що використовується у макетах лабораторних робіт курсу, містить контролер KS0066, що сумісний з HD44780. Однак, він може бути проініціалізований у 4-бітному режимі за більш спрощеною процедурою. Тому ми наведемо повну ініціалізацію, яка коректно працюватиме як у пакеті Proteus, так і на реальному LCD-дисплею.

data – порт з виводами DB7...DB0

control – порт, що містить виводи RS, RW, E

## 8-РОЗРЯДНИЙ ІНТЕРФЕЙС (для рис. 5)

### Процедура ініціалізації LCD

подача живлення

затримка 40 мсек

Порт **data** та виводи **control** налаштовуємо на вихід

// шина 8 біт, 2 стрічки, 5x8 точок

RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	N <sup>1</sup>	F <sup>0</sup>	— <sup>0</sup>	— <sup>0</sup>



E = 1  
затримка 0.15 мксек  
E = 0

затримка 50 мксек

E = 1  
затримка 0.15 мксек  
E = 0

затримка 50 мксек

*// включити дисплей (курсор)*

RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D <sup>1</sup>	C <sup>0</sup>	B <sup>0</sup>

E = 1  
затримка 0.15 мксек  
E = 0

затримка 50 мксек

*// зсуви екрану та курсору*

RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	I/D <sup>0</sup>	S <sup>0</sup>

E = 1  
затримка 0.15 мксек  
E = 0

затримка 50 мксек

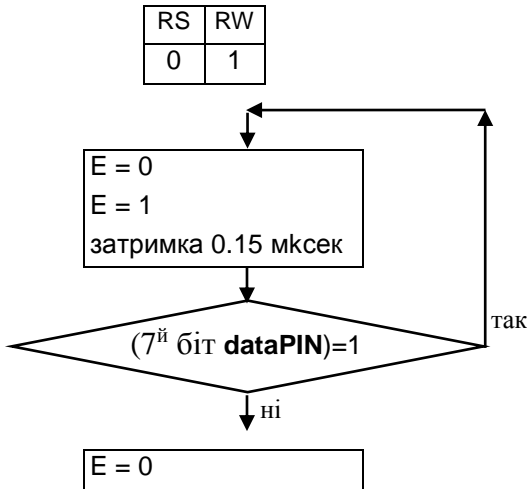
*// очистити дисплей*

RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

E = 1  
затримка 0.15 мксек  
E = 0

## Опитування прапорця зайнятості BF (busy flag)

Порт **data** налаштовуємо як високоімпедансний вхід



## Запис команди в LCD

**void** LCD\_WriteCommand(**unsigned char** Command)

Опитування прапорця BF

RS	RW
0	0

Порт **data** налаштовуємо на вихід

*// виводимо команду в порт даних*

dataPort = Command

E = 1  
затримка 0.15 мксек  
E = 0

## Вивід символу на LCD

**void** LCD\_WriteLetter(**unsigned char** Letter)

Опитування прапорця BF

RS	RW
1	0

Порт **data** налаштовуємо на вихід

*// виводимо команду в порт даних*

dataPort = Letter

E = 1

затримка 0.15 мсек

E = 0

## 4-РОЗРЯДНИЙ ІНТЕРФЕЙС (для рис. 6) Процедура ініціалізації LCD

подача живлення

затримка 40 мсек

Порт **data** та виводи **control** налаштовуємо на вихід

*// Функція установки*

RS	RW	DB7	DB6	DB5	DB4
0	0	0	0	1	1

E = 1

затримка 0.15 мсек

E = 0

затримка 2 мсек

E = 1  
затримка 0.15 мксек  
E = 0

затримка 50 мксек

E = 1  
затримка 0.15 мксек  
E = 0

затримка 50 мксек

*// Функція установки*

RS	RW	DB7	DB6	DB5	DB4
0	0	0	0	1	0

E = 1  
затримка 0.15 мксек  
E = 0

затримка 50 мксек

*// Функція установки*

RS	RW	DB7	DB6	DB5	DB4
0	0	0	0	1	0

E = 1  
затримка 0.15 мксек  
E = 0

RS	RW	DB7	DB6	DB5	DB4
0	0	N <sup>1</sup>	F <sup>0</sup>	0	0

E = 1  
затримка 0.15 мксек  
E = 0

затримка 50 мксек

*// включити дисплей (курсор)*

RS	RW	DB7	DB6	DB5	DB4
0	0	0	0	0	0

E = 1

затримка 0.15 мксек

E = 0

RS	RW	DB7	DB6	DB5	DB4
0	0	1	D <sup>1</sup>	C	B

E = 1

затримка 0.15 мксек

E = 0

затримка 50 мксек

*// зсуви екрану та курсору*

RS	RW	DB7	DB6	DB5	DB4
0	0	0	0	0	0

E = 1

затримка 0.15 мксек

E = 0

RS	RW	DB7	DB6	DB5	DB4
0	0	0	1	I/D <sup>0</sup>	S <sup>0</sup>

E = 1

затримка 0.15 мксек

E = 0

затримка 50 мксек

*// очистити дисплей*

RS	RW	DB7	DB6	DB5	DB4
0	0	0	0	0	0

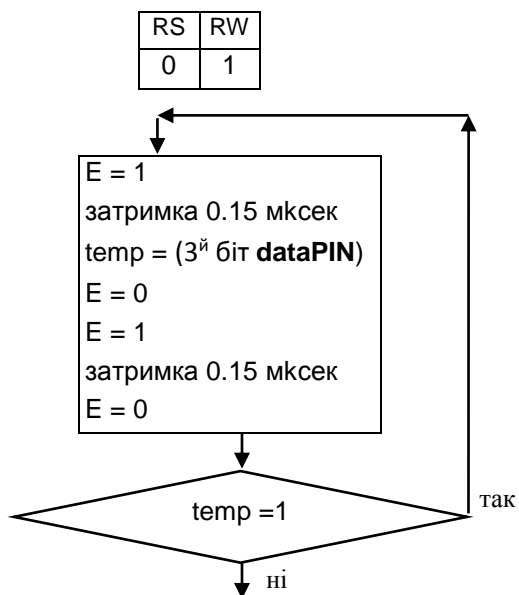
E = 1  
затримка 0.15 мксек  
E = 0

RS	RW	DB7	DB6	DB5	DB4
0	0	0	0	0	1

E = 1  
затримка 0.15 мксек  
E = 0

### Опитування прапорця зайнятості BF (busy flag)

виводи **data** 0...3 налаштовуємо як високоімпедансні входи



### Запис команди в LCD

**void** LCD\_WriteCommand(**unsigned char** Command)

Опитування прапорця BF

RS	RW
0	0

виводи **data** 0...3 налаштовуємо на вихід

*// виводимо старший байт команди на шину даних*

dataPort(0...3) = Command>>4

E = 1

затримка 0.15 мксек

E = 0

*// виводимо молодший байт команди на шину даних*

dataPort(0...3) = Command & 0b1111

E = 1

затримка 0.15 мксек

E = 0

### Вивід символу на LCD

**void** LCD\_WriteLetter(**unsigned char** Letter)

Опитування прапорця BF

RS	RW
1	0

виводи **data** 0...3 налаштовуємо на вихід

*// виводимо старший байт символу на шину даних*

dataPort(0...3) = Letter >>4

```
E = 1
затримка 0.15 мксек
E = 0
```

*// виводимо молодший байт символу на шину даних*

```
dataPort(0...3) = Letter & 0b1111
```

```
E = 1
затримка 0.15 мксек
E = 0
```

### **Приклади реалізації**

*однакові як для 8-ми, так і для 4-розрядної шини*

#### 1. Вивід символу на екран.

Вивід можемо виконувати, як за значенням коду у таблиці символів, а можемо і безпосередньо вказувати символ, оскільки його ASCII-код співпадає з кодом у таблиці. Однак, коди кирилических літер не співпадають, тому їх потрібно виводити згідно кодів у таблиці символів LCD.

```
LCD_WriteLetter(0x44);
LCD_WriteLetter('D');
```

#### 2. Функція виводу стрічки на екран LCD.

Для спрощення виводу стрічок на екран ми можемо написати таку додаткову функцію:

```
void LCD_WriteStr(volatile char *str)
{
    for(int i=0;i<255;i++)
        if (str[i]!='\0')    return;
        else                LCD_WriteLetter(str[i]);
}
```

Тоді вивід буде виглядати так:

```
LCD_WriteStr("1-Ware (Error)!");
```



### 3. Функція виводу стрічки, записаної у Flash, на екран LCD.

Нехай маємо проініціалізовану у Flash таку стрічку

```
PROGMEM unsigned char Hello[] ="Hello";
```

Функція виводу цієї стрічки на екран буде виглядати так:

```
void LCD_WriteStrPROGMEM(unsigned char *str)
{
    for(int i=0;i<255;i++)
        if (pgm_read_byte(& (str[i]) )=='\0')    return;
        else    LCD_WriteLetter(pgm_read_byte( &(str[i]) ));
}
```

Вивід буде виглядати так:

```
LCD_WriteStrPROGMEM(Hello);
```

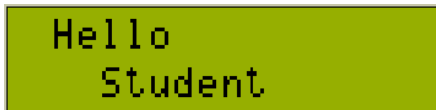
### 4. Перехід курсору на нову позицію на екрані LCD.

Для цього напишемо таку функцію

```
void LCD_GotoYX(unsigned char Y, unsigned char X)
{
    if(Y==1)    LCD_WriteCommand(((1<<7)|(X-1)));
    else        LCD_WriteCommand(((1<<7)|(0x40+X-1)));
}
```

Стрічку «Hello» запишемо у позиції [1,2], а стрічку «Student» у позиції [2,4]

```
LCD_GotoYX(1,2);
LCD_WriteStr("Hello");
LCD_GotoYX(2,4);
LCD_WriteStr("Student");
```



```
Hello
Student
```

## 2. Робота з клавіатурою 4×4

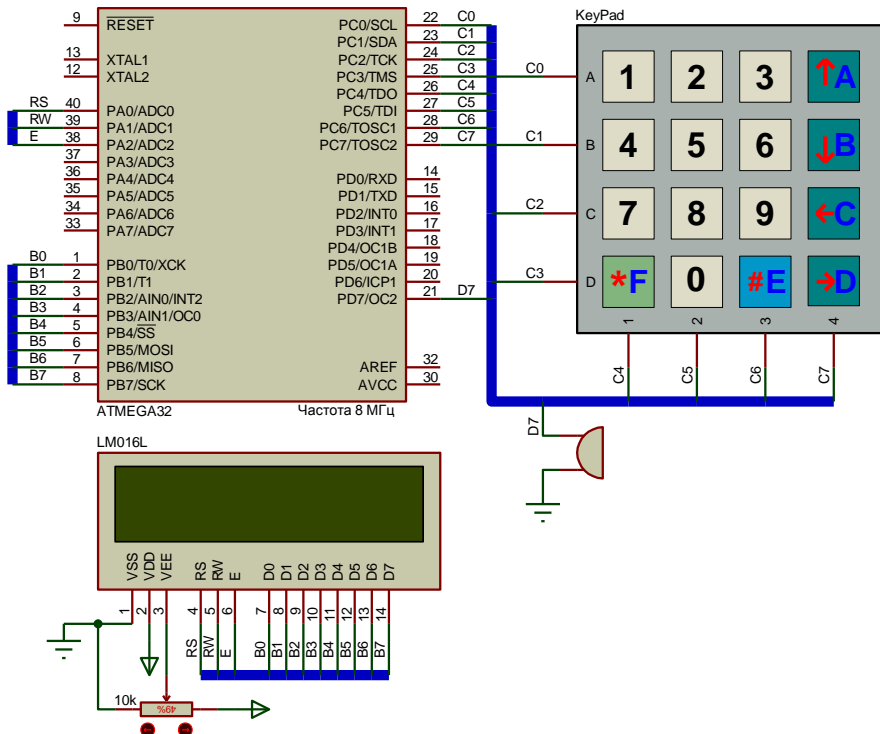
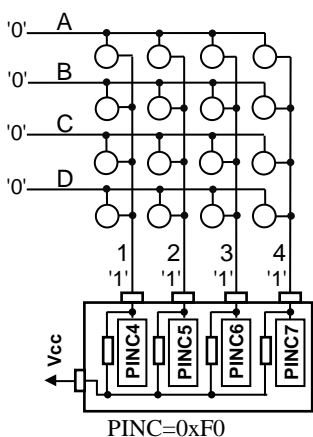


Рис. 3. Типова схема підключення клавіатури 4×4 до МК

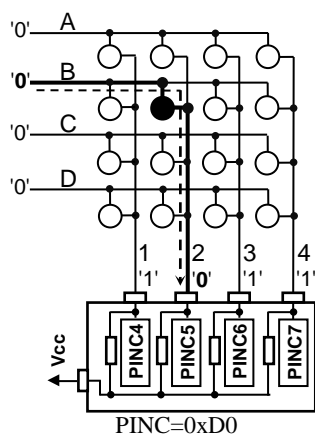
Клавіатура 4×4 підключається до виводів порту МК, при цьому одна половина виводів порту конфігурується на вихід та подається лог. «0», а інша половина конфігурується на вхід з внутрішніми підтягуючими резисторами. Нехай рядки (рис. 3) клавіатури будуть підключені до виходів МК, а стовпці до входів.

Якщо не натиснута жодна кнопка (рис. 4а), тоді на порті буде зчитуватися постійно число 0xF0 (F – лог. «1» за рахунок підтягуючих резисторів до напруги живлення, 0 – на виходах рядків присутній низький рівень). Як тільки буде натиснута довільна кнопка (рис. 4б), тоді через неї буде подано низький рівень напруги (лог. «0») на один з входів, і з порта буде зчитане число, відмінне від 0xF0. Після цього нам необхідно вчислити у якому рядку натиснута кнопка, при цьому стовпець вже відомий, бо у ньому з'явиться низький рівень (лог. «0»).

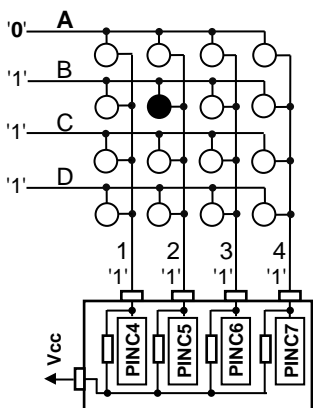
Для виявлення потрібного рядка з натиснутою кнопкою необхідно спершу подати низький рівень «0» на рядок А, а на решта рядків лог. «1» (рис. 4в). Якщо на одному з входів з'явиться «0», тоді місце розташування кнопки виявлено. Якщо на входах все таки присутні лише лог. «1», тоді переходимо до рядка В (рис. 4г) та подаємо тепер на нього лог. «0», і звіряємо значення входів. Сканування проводиться до моменту, поки не буде виявлений рядок, де натиснута кнопка. Після визначення необхідного рядка та стовпця ідентифікуємо нашу кнопку та виконуємо далі необхідні дії.



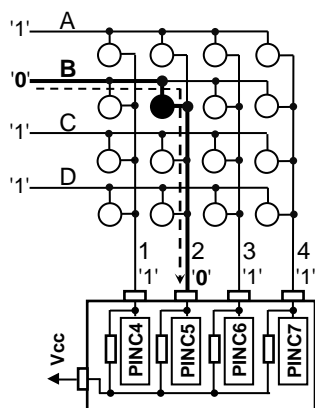
а) Жодна кнопка не натиснута



б) Виявлення натиску якоїсь кнопки



в) Сканування лінії А  
на присутність натиснутої кнопки



г) Сканування лінії В  
на присутність натиснутої кнопки

Рис. 4. Процедура сканування клавіатури 4×4

**Приклад.** Клавіатура 4×4 підключена до МК, як на рис. 3. Годинник реалізований на 16-ти розрядному таймері T1, який сконфігурований на переривання по співпадінню щосекунди. Значення годинника виводиться у 2-му рядку LCD. При натиску будь-якої кнопки на клавіатурі дисплей очищається, та виводиться символ натиснутої кнопки (годинник при цьому не відображається). Переключення режимів відображення дисплею реалізується за допомогою логічної змінної `display`. Повернення до відображення значення годинника здійснюється за допомогою натиску кнопки F (\*).

**Зауваження:** розміщений нижче код програми для наведеного прикладу не містить бібліотеки з ініціалізації LCD та стандартних функцій виводу команд та символів. Бібліотеку для LCD не подано, оскільки студенти повинні самі її реалізувати згідно завдання.

```
//===== LCD Define =====
#define LCDdataPORT PORTB // LCD Data Port
#define LCDdataPIN PINB
#define LCDdataDDR DDRB

#define LCDcontrolPORT PORTA // LCD Control Port
#define LCDcontrolPIN PINA
#define LCDcontrolDDR DDRA

#define RS 0
#define RW 1
#define E 2

//=====
#include <avr/io.h>
#include <util/delay.h>
#include <avr/pgmspace.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include "LCD_8.h"

#define Key_1 0b11101110
#define Key_2 0b11011110
#define Key_3 0b10111110
#define Key_A 0b01111110

#define Key_4 0b11101101
#define Key_5 0b11011101
#define Key_6 0b10111101
#define Key_B 0b01111101
```

```

#define Key_7    0b11101011
#define Key_8    0b11011011
#define Key_9    0b10111011
#define Key_C    0b01111011

#define Key_F    0b11100111
#define Key_0    0b11010111
#define Key_E    0b10110111
#define Key_D    0b01110111
PROGMEM unsigned char sixty[60][3] ={
    {"00"}, {"01"}, {"02"}, {"03"}, {"04"}, {"05"}, {"06"}, {"07"}, {"08"}, {"09"},
    {"10"}, {"11"}, {"12"}, {"13"}, {"14"}, {"15"}, {"16"}, {"17"}, {"18"}, {"19"},
    {"20"}, {"21"}, {"22"}, {"23"}, {"24"}, {"25"}, {"26"}, {"27"}, {"28"}, {"29"},
    {"30"}, {"31"}, {"32"}, {"33"}, {"34"}, {"35"}, {"36"}, {"37"}, {"38"}, {"39"},
    {"40"}, {"41"}, {"42"}, {"43"}, {"44"}, {"45"}, {"46"}, {"47"}, {"48"}, {"49"},
    {"50"}, {"51"}, {"52"}, {"53"}, {"54"}, {"55"}, {"56"}, {"57"}, {"58"}, {"59"} };

volatile unsigned char display = 0;
typedef struct
{
    unsigned char second, minute, hour;
} Time;
Time T2 = {0,0,0};

ISR(TIMER1_COMPA_vect)      // Таймер T1 по співпадінню A, кожної 1 сек.
{
    if(++T2.second == 60)
    {
        T2.second = 0;
        if(++T2.minute == 60)
        {
            T2.minute = 0;
            if(++T2.hour == 24)
                T2.hour = 0;
        }
    }
    if(display==0)
    {
        LCD_GotoYX(2,1);
        LCD_WriteStrPROGMEM(sixty[T2.hour],2);
        LCD_WriteLetter(':');
        LCD_WriteStrPROGMEM(sixty[T2.minute],2);
        LCD_WriteLetter(':');
        LCD_WriteStrPROGMEM(sixty[T2.second],2);
    }
}

```

```

void main()
{
    Initializer();
    sei();
    unsigned char freePIN=1, key=1;
    while (1) //основний робочий цикл
    {
        if(freePIN == 1) //перевірка чи була натиснута кнопка
        {
            //якщо =1, тоді ще не натискалася
            if(PINC != 0xF0)
            {
                _delay_ms(50);
                freePIN = 0;
                // Визначення натиснутої клавіші
                // почергова подача 0V на рядки клавіатури A,B,C,D
                PORTC = 0b11111110; // A-рядок
                asm("nop");
                if(PORTC == PINC)
                {
                    PORTC = 0b111111101; // B-рядок
                    asm("nop");
                    if(PORTC == PINC)
                    {
                        PORTC = 0b1111111011; // C-рядок
                        asm("nop");
                        if(PORTC == PINC)
                        {
                            PORTC = 0b11111110111; // D-рядок
                            asm("nop");
                            if(PORTC == PINC)
                                key=0; // жодна клавіша не натиснута
                        }
                    }
                }
            }
        }
        if(key==1) //визначення натиснутої клавіші
        {
            PORTD |= 1<<7; //включаємо бузер (звук)
            LCD_GotoYX(1,1);
            if(PINC == Key_1) LCD_WriteLetter('1');
            if(PINC == Key_2) LCD_WriteLetter('2');
            if(PINC == Key_3) LCD_WriteLetter('3');
            if(PINC == Key_4) LCD_WriteLetter('4');
            if(PINC == Key_5) LCD_WriteLetter('5');
            if(PINC == Key_6) LCD_WriteLetter('6');
        }
    }
}

```

```

        if(PINC == Key_7)    LCD_WriteLetter('7');
        if(PINC == Key_8)    LCD_WriteLetter('8');
        if(PINC == Key_9)    LCD_WriteLetter('9');
        if(PINC == Key_0)    LCD_WriteLetter('0');
        if(PINC == Key_A)    LCD_WriteLetter('A');
        if(PINC == Key_B)    LCD_WriteLetter('B');
        if(PINC == Key_C)    LCD_WriteLetter('C');
        if(PINC == Key_D)    LCD_WriteLetter('D');
        if(PINC == Key_E)    LCD_WriteLetter('E');
        if(PINC == Key_F)    LCD_WriteLetter('F');
        _delay_ms(300);      //затримка для бузера
        PORTD &= ~(1<<7);    //виключаємо бузер (звук)
    }
    key=1;
    PORTC = 0xF0;            //відновлюємо порт
}
}
else
    if(PINC == 0xF0)        //перевіряємо чи кнопка відпущена
    { _delay_ms(200); freePIN=1; }
}
}
void Initializer()          // Ініціалізація заліза
{
    // Ініціалізація портів
    // Порт A на вихід
    DDRA = 0xFF;
    PORTA = 0x00;
    // Порт B на вхід з підт. резистр.
    DDRB = 0x00;
    PORTB = 0xFF;
    // Порт C клавіатура: рядки на вихід, стовпці на вхід
    DDRC = 0x0F;
    PORTC = 0xF0;
    // Порт D на вихід
    DDRD = 0xFF;
    PORTD = 0x00;
    InitLCD();              // Ініціалізація LCD
    // Таймер#1: Скид при співпадинні OCR1A (1sec) + дільник=256
    OCR1AH = 0x7A;
    OCR1AL = 0x11;
    TCCR1A = 0x00;
    TCCR1B = 1<<WGM12 | 1<<CS02;
    TIMSK |= 1<<OCIE1A;    // дозвіл на переривання по співпадинню
}

```

