МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Кафедра АТ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисциплін

«ОБЧИСЛЮВАЛЬНА ТЕХНІКА І МІКРОПРОЦЕСОРИ»,

«МІКРОПРОЦЕСОРНІ СИСТЕМИ»

для студентів напрямів

6.050903 “Телекомунікаційні системи та мережі”

6.050201 “Системна інженерія”

усіх форм навчання

Розглянуто на засіданні кафедри
«Автоматика і телекомунікації»
протокол № 9 від 30.08.2010р.

Затверджені на засіданні
Навчально-видавничої ради ДонНТУ
Протокол № 4 від 07.10.2010р. (р.№339)

ДОНЕЦЬК – 2010

Методичні вказівки до виконання лабораторних робіт з дисциплін «Обчислювальна техніка і мікропроцесори» для студентів напряму 6.050903 “Телекомунікаційні системи та мережі” і «Мікропроцесорні системи» для студентів напряму 6.050201 “Системна інженерія” денної і заочної форм навчання. / Суков С.Ф, В.Я., Яремко І.М., Долгіх І.П., Батир С. С. – Донецьк, ДонНТУ, 2010. - 30 с.

Укладачі:

Суков С.Ф, В.Я., Яремко І.М., Долгіх І.П.,Батир С. С.

Відповідальний за випуск:
Зав. кафедрою «Автоматика і телекомунікації» к.т.н., доцент
Бессараб В.І.

Рецензент: к.т.н., доцент кафедри «Автоматизовані системи управління» П.О. Шатохін

ЗАГАЛЬНІ ПОЛОЖЕННЯ.

Вивчення кожної теми теоретичного матеріалу дисципліни «Обчислювальна техніка і мікропроцесори» завершується проведенням практичних і лабораторних робіт.

Мета лабораторних робіт - формування у студентів знань, методики та прийомів розробки і налагодження програмного і апаратного забезпечення мікропроцесорної системи.

Лабораторні роботи виконуються на спеціально розробленому стенді, схема якого представлена на рис.1, з використанням графічного середовища розробки програмного забезпечення для мікроконтролерів з архітектурою AVR “AlgorithmBuilder”.

Середовище забезпечує повний цикл розробки, починаючи від введення алгоритму, включаючи налагодження, і закінчуючи внутрішньосхемним програмуванням кристала. Студент має можливість розробляти програми як на рівні асемблера, так і на макро-рівні, при якому можлива робота зі знакозмінними величинами довільної довжини. Це наближає можливості програмування до мови високого рівня.

В результаті виконання лабораторних робіт студент повинен знати структуру апаратних засобів мікропроцесорної системи, структуру системи команд мікроконтролера, способи і методи управління складовими мікропроцесорної системи.

Для досягнення цієї мети студенти повинні виконати загальне завдання, як приклад, і видані викладачем індивідуальні завдання.

Порядок виконання індивідуального завдання (по заняттях):

1. Ознайомлення із завданням.
2. Розробка алгоритму завдання і тестових наборів даних.
3. Введення і відлагодження програми. Отримання і оцінка результатів.
4. Захист лабораторної роботи.

Самостійна робота студента включає:

1. Складання програми.
2. Оформлення звіту.

У звіті приводяться постановка завдання, блок-схема алгоритму і, при необхідності, її опис, лістинг програми, висновки по роботі.

За наведеним звітом викладач проводить співбесіду для контролю знань по результатам виконання робіт.

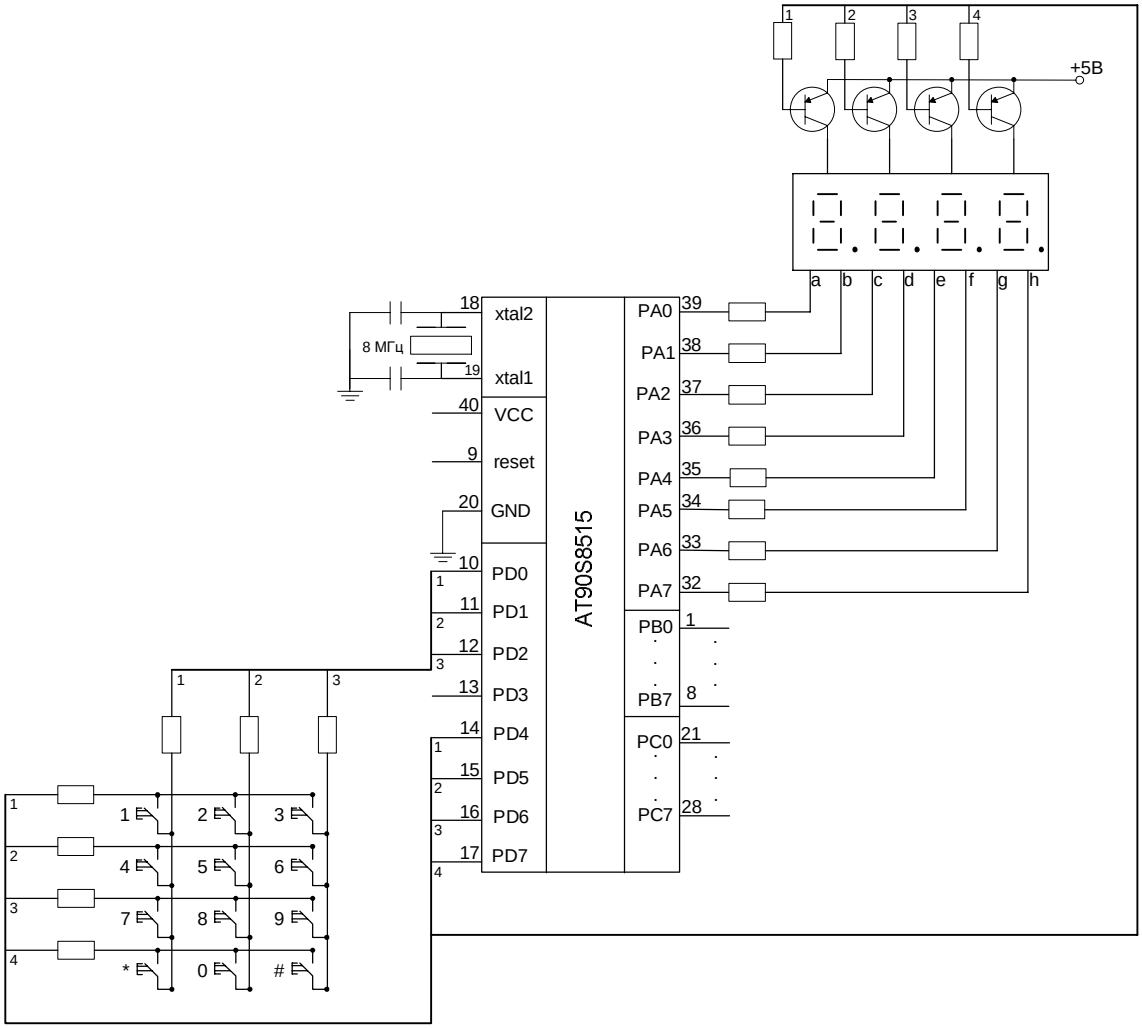

Рисунок 1. Схема стенда на основі мікроконтролера AT90S8515

ЛАБОРАТОРНА РОБОТА №1

ВИВЧЕННЯ МЕТОДИКИ ПІДГОТОВКИ І ВІДЛАГОДЖЕННЯ ПРОГРАМ В СЕРЕДОВИЩІ ALGORITHM BUILDER

Мета роботи:отримання навичок в складанні і відлагодженні простих програм в середовищі AlgorithmBuilder, ознайомлення з лабораторним стендом.

1. Короткі відомості з теми.

В лабораторній роботе необхідно реалізувати програмне забезпечення з використанням мови програмування AlgorithmBuilder для управління лінійкою світлодіодів.

Результати роботи програми: при натисканні на кнопку лінійка світлодіодів світить, інакше (в ненатиснутому стані) - не світить.

Аналіз стану натискання кнопки виконується по логічним нулем, при налаштуванні необхідного контакта порта (порта PD3) на введення і активізації внутрішніх підтягуючих резисторів. Лінійка світлодіодів управляється логічним нулем (при налаштуванні порта PC на виведення): якщо біт регістра PortC встановлено в 0 –відповідний діод світить, 1 - діод не горить.

Блок-схема роботи програми представлена на рис. 1.1.

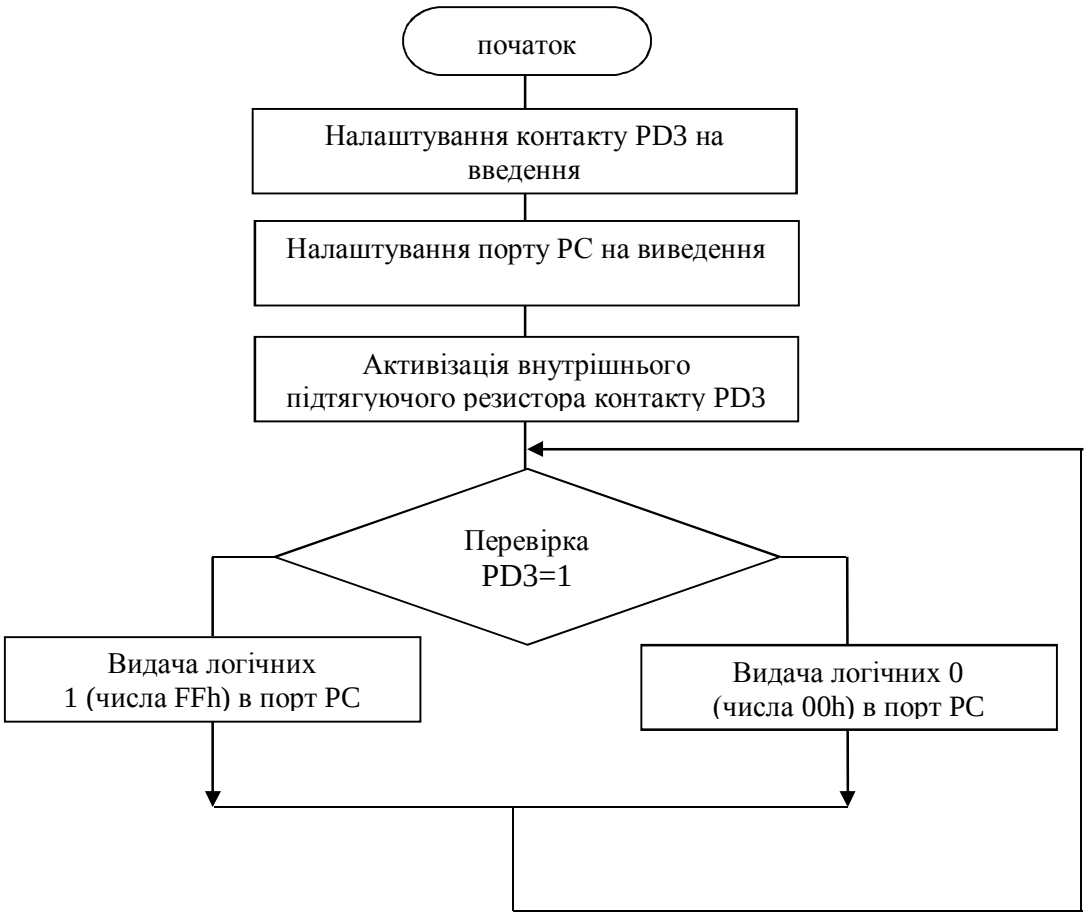

Рисунок 1.1 - Блок схема до виконання лабораторної роботи.

Налаштування контакту порту **PD** на введення виконується шляхом завдання значення 0 відповідному біту регістра **DDRD**, активізація внутрішнього підтягуючого резистора - завданням значення 1 відповідному біті регістра **PortD** (при налаштуванні порту на введення - див. опис МК).
Перевірка наявності логічного рівня - аналіз регістра **PinD**.
Налаштування контактів порту **PC** на вивід виконується завданням значення 1(одиниці) бітам регістра **DDRC.**

2. Порядок виконання роботи.

2.1 Ознайомитися з принципами роботи в середовищі AlgorithmBuilder, описом робочого стенду, керівництвом з використання МК AT90S8515.
2.2 Відповідно до завдання скласти алгоритм обробки натискання кнопки і управління світінням лінійки світлодіодів.
2.3 Відлагодити програму в покроковому режимі в симуляторі і здійснити прошивку кристала.
2.4 Перевірити працездатність програми на лабораторному стенді.
2.5 Виконати індивідуальне завдання, згідно варіанта (табл.1.1).

Таблиця 1.1 Варіанти індивідуальних завдань завдань.

№ варіанта	Завдання	
	При натисканні та утриманні кнопки світять світлодіоди	При ненатиснутій кнопці світять світлодіоди
1.	1, 8	2,6
2.	2, 7	1,5
3.	5, 6	3,4,8
4.	4, 8	4,3,2,1
5.	7, 5	1,2,3
6.	3, 8	2,7,6
7.	6,3	5
8.	1,3,4	2,8
9.	1,2,8	5,6
10.	1,5,8	3,7
11.	2,3,6,7	1,6
12.	3,5,7	1,8
13.	5,6,7,8	6,7,8
14.	1,3,5,7	1,2,5
15.	2,4,6,8	2,4,6
16.	1,8	3,5,7
17.	3,7,8	2,7
18.	4,5,6	2,6
19.	1,2,3	1
20.	4,5,6	8
21.	3,8	2,3
22.	1,4,5	4,5,6
23.	2,5,7	2,7
24.	3,4,8	1,8
25.	1,6	2

ЛАБОРАТОРНА РОБОТА №2

ВИКОРИСТАННЯ ПІДПРОГРАМ ПРИ ПРОГРАМУВАННІ В СЕРЕДОВИЩІ ALGORITHM BUILDER

Мета роботи: отримати навички написання і застосування підпрограм в загальній структурі створюваного програмного забезпечення за допомогою графічного середовища AlgorithmBuilder.

Завдання: реалізувати крапку, що біжить, на лінійці світлодіодів, міняючи напрям руху при натисненні і наступному відпусканні кнопки.

1. Короткі відомості по роботі

Для реалізації точки, що біжить, необхідно через деякі проміжки часу змінювати комбінацію біт, що виводяться в порт С. Оскільки система команд мікроконтролера не допускає можливість безпосередньої зміни, наприклад зсув, даних в регістрах портів, для цього необхідно використовувати допоміжний регістр (один з робочих регістрів), вміст якого потім повинен виводитися в порт С. Для зручності назвемо цей регістр регістром "крапки". Змінювати стан цього регістра можна, використовуючи логічний зсув одиниці, заздалегідь записаної в нього. Проте оскільки світлодіоди активуються логічним нулем, перед виведенням регістра "крапки" в порт **С** його необхідно інвертувати (щоб світився тільки один світлодіод). Після виводу регістр "рядка" наново інвертують і коли значення регістра "крапки" досягає нуля, залежно від напряму руху наново записується одиниця або в сьомій, або в нульовій біт цього регістра.

Оскільки напрям рядка, що біжить, міняється по відтисканню кнопки, то для фіксації цієї події потрібно два прапорці. Один прапорець - це факт натиснення кнопки, назвемо його *Button*. Другий прапорець - це прапорець напряму руху рядка, назвемо його *Direct*. Якщо *Direct* в нулі, то рядок, наприклад, рухається вгору (праворуч), якщо в одиниці (ліворуч). Ці прапорці можна реалізувати, використовуючи два біта будь-якого робочого регістра.

Аналіз і установка цих прапорців здійснюється таким чином. Аналізується контакт PD3, якщо на ньому логічний нуль, то кнопка натиснута і встановлюється прапор Button. При повторному аналізі контакту PD3, якщо на нім логічна одиниця (кнопка відтиснута) і прапор Button встановлений, то міняється значення прапора Direct. Значення прапора Direct міняється за допомогою інвертування використовуваного регістра, при цьому необхідно обнулити прапор Button.

Інтервал між операціями зсуву зручно реалізувати за допомогою підпрограми паузи. Пауза потрібна для того, щоб користувач візуально спостерігав крапку, що біжить, інакше частота виведення дуже велика і рух крапки буде непомітним (світитиметься уся лінійка світлодіодів). Для написання цієї підпрограми необхідно використовувати три лічильники (три вільних робочі регістри), в які заноситься певні значення, що формують тривалість паузи.

Використання підпрограм при створенні програмного забезпечення сприяє поліпшенню наочності (читаності) програми, роблячи її менш громіздкою. Також

це зменшує витрату ресурсів пам'яті самого контролера. У загальному випадку при виконанні програми процесор виконує команди в послідовності їх написання. При виявленні команди виклику підпрограми, він зберігає адресу поточної команди в стеку і переходить до виконання команд підпрограми. Після виходу з підпрограми процесор починає виконувати команди з адреси збереженого раніше в стеку. Тому на самому початку програми необхідно ініціалізувати стек SP, тобто вказати адресу в ОЗП контролера, де знаходитиметься вершина стека. Оскільки стек заповнюється в напрямі знизу-вгору, то вершиною стека зазвичай вказують адресу останньої комірки ОЗП.

Для мікроконтроллера AT90S8515 остання комірка ОЗП має адресу $025F, а покажчик стека складається з двох регістрів: SPL (молодший байт адреси) і SPH (старший байт адреси). При цьому AlgorithmBuilder допускає різні варіанти синтаксису для ініціалізації стека :

$025F -> SP
або
$5F -> SPL
$02 -> SPH.

Крім того, ініціалізація стека може бути виконана з використанням настроювача "SETTER": налаштування виконується встановленням опції Loadendaddressof SRAM. (рис. 2.1):

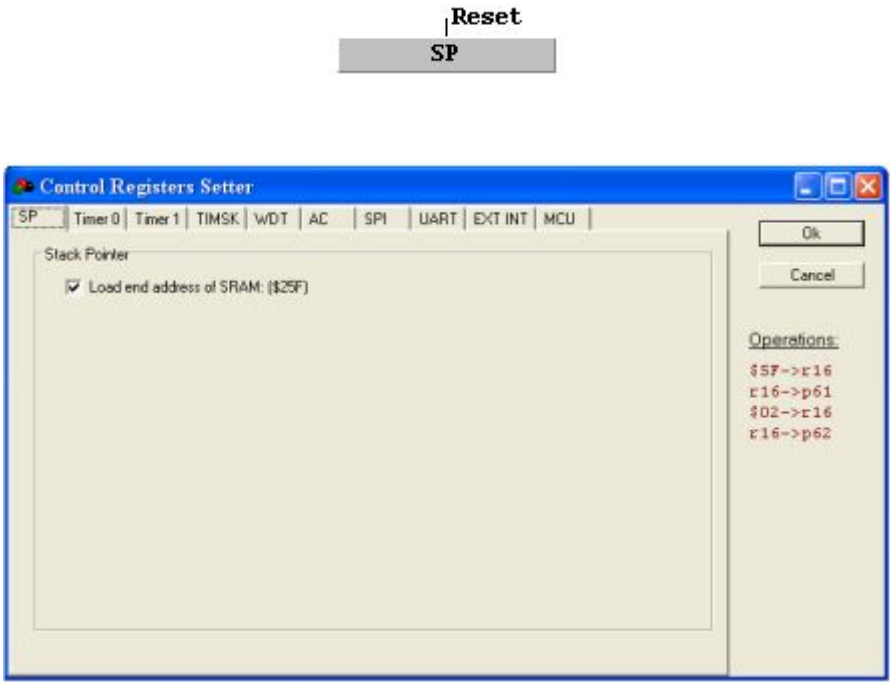

Рисунок 2.1 - Ініціалізація стека за допомогою настроювача "SETTER"

При цьому усі перераховані варіанти ініціалізації стека є макрооператорами і використовують регістр-посередник *R16* як показано на рис. 2.1.

Алгоритми підпрограми і основної програми і представлені на рис. 2.2 і 2.3 відповідно.

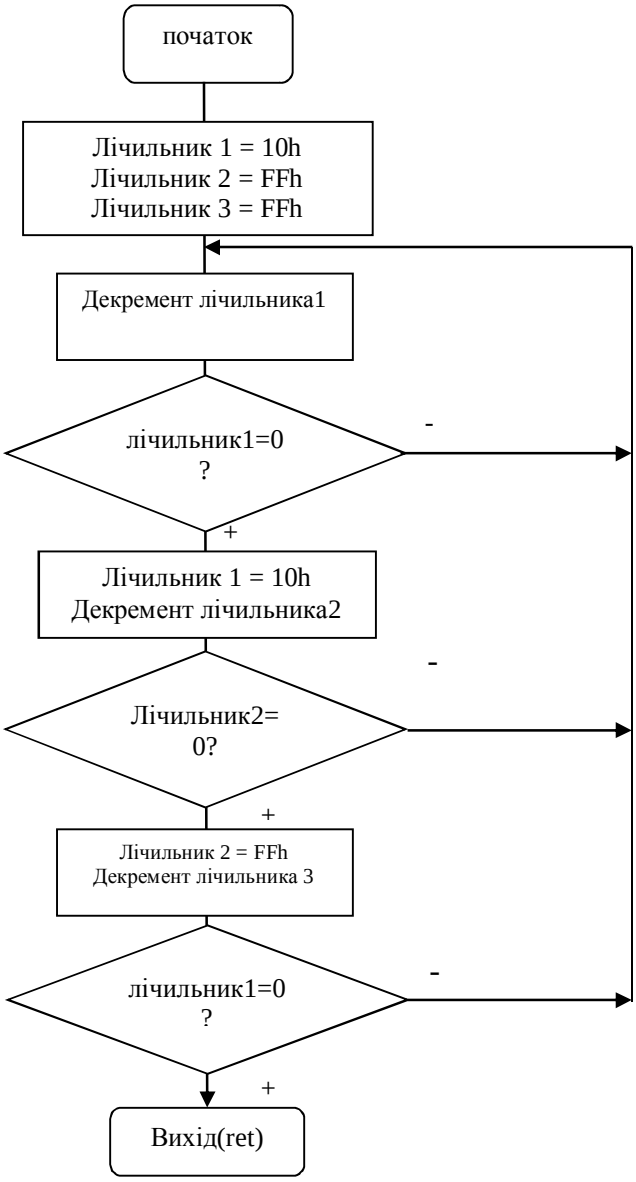

Рисунок 2.2 - Блок-схема підпрограми паузи

2. Порядок виконання роботи.

2.1 Ознайомитися з принципом написання підпрограм в графічному середовищі, їх викликом в основній програмі.
2.2 Написати і відлагодити програму в покроковому режимі в симуляторі.
2.3 Перевірити працездатність програми на лабораторному стенді.
2.4 Виконати індивідуальне завдання, згідно варіанта (табл.2.1).

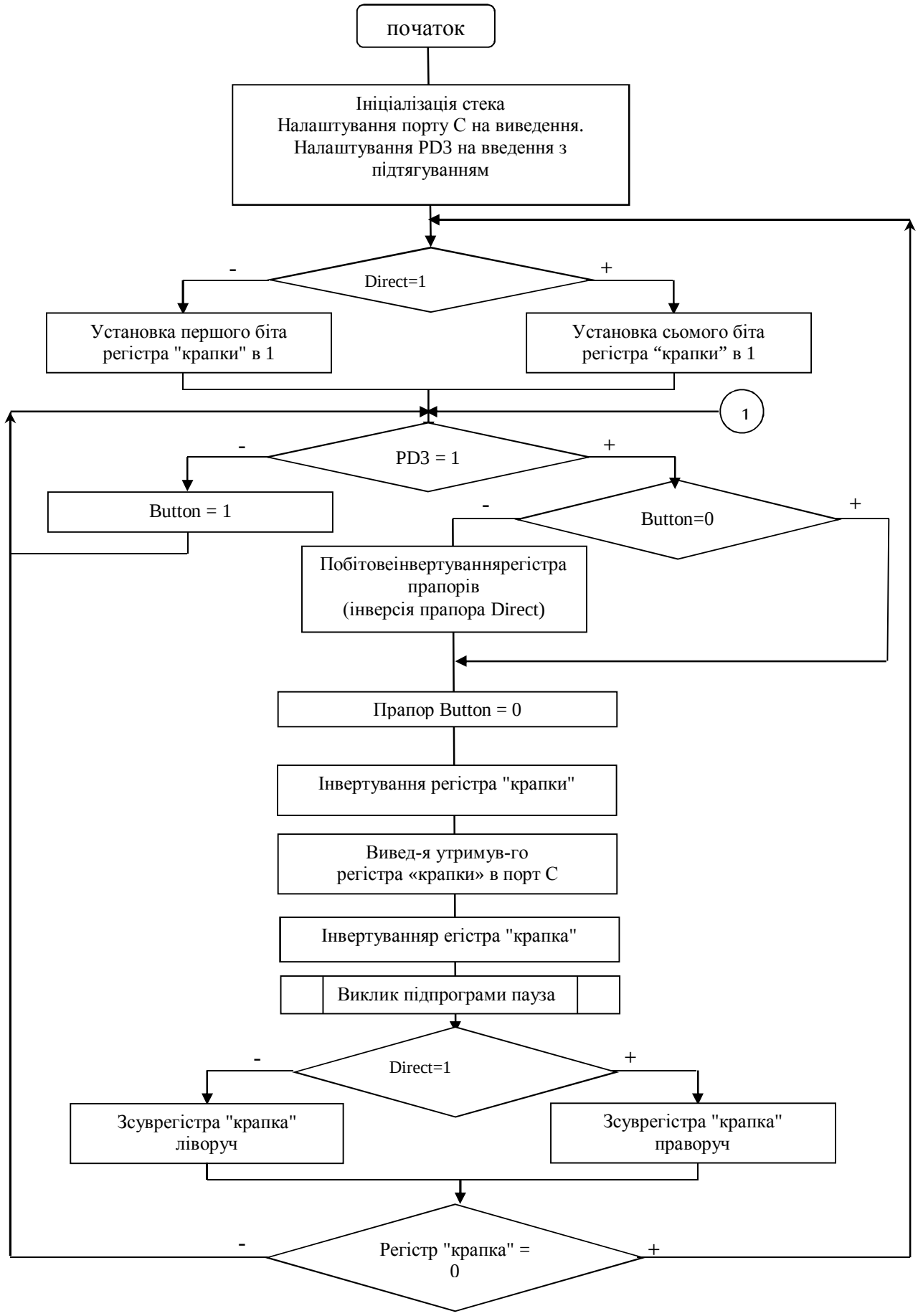

Рисунок 2.3 - Блок-схема основної програми

Таблиця 2.1 Варіанти індивідуальних завдань.

№	Дії при парному натисненні	Дії при непарному натисненні
1.	Біжуча крапка вліво	Рух крапки вліво-вправо. Напрямок змінюється при досягненні крайніх світло діодів.
2.	Біжуча крапка вправо	Дві крапки, які рухаються назустріч.
3.	Дві біжучі крапки вліво	Біжуча крапка вправо
4.	Біжуча крапка вліво по непарних світлодіодах	Крапка вправо по непарних світлодіодах.
5.	Біжуча крапка вправо по 4 менших світлодіодах	Біжуча крапка вліво по 4 старших світлодіодах
6.	Біжуча крапка вправо	Світяться непарні світлодіоди
7.	Світяться парні світлодіоди	Рух крапки вліво
8.	Рух крапки вліво	Блимають усі світлодіоди
9.	Рух крапки вліво з двократним блиманням	Біжуча крапка вправо
10.	Біжуча крапка вліво	Рух крапки вправо з трикратним блиманням
11.	Рух крапки вліво-вправо. Напрямок змінюється при досягненні крайніх світлодіодів.	Біжуча крапка вправо
12.	Почергово блимають 2 і 7 світлодіоди	Біжуча крапка вліво
13.	Біжуча крапка вправо	Світяться половина світлодіодів
14.	Біжуча крапка вліво	Крапка вправо по парних світлодіодах.
15.	Крапка вліво по непарних світлодіодах.	Біжуча крапка вправо
16.	Рух двох крапок від середини до крайніх світлодіодів	Біжуча крапка вліво
17.	Біжуча крапка вправо	Рух крапки вліво-вправо. Напрямок змінюється при досягненні крайніх світлодіодів.
18.	Біжуча крапка вліво	Блимання світлодіодної лінійки
19.	Біжуча крапка вправо	Блимають почергово тетради світлодіодів
20.	Рух двох крапок від крайніх світло діодів до середини.	Біжуча крапка вправо
21.	Рух крапки вліво-вправо. Напрямок змінюється при досягненні крайніх світло діодів.	Біжуча крапка вліво
22.	Біжуча крапка вправо	Рух крапки по непарних світло діодах вліво з двократним блиманням
23.	Біжуча крапка вліво	Рух двох крапок від середини до крайніх світло діодів і назад
24.	Рух двох крапок від середини до крайніх світлодіодів	Біжуча крапка вправо
25.	Блимають почергово парні та непарні світлодіоди	Біжуча крапка вліво

ЛАБОРАТОРНА РОБОТА №3

ОБРОБКА ПЕРЕРИВАНЬ

Ціль роботи: отримати навички організації обробки внутрішніх і зовнішніх переривань мікроконтроллера за допомогою графічного середовища AlgorithmBuilder.

Завдання: реалізувати рядок, що біжить, точку на лінійці світлодіодів, міняючи напрям руху по зовнішньому перериванню. Період між виводами реалізувати за перериванням від таймера/лічильника контролера.

3.1 Короткі відомості по роботі

Використання переривань при написанні програми дозволяє зменшити її об'єм, а також збільшити оперативність виконуваних операцій. Переривання дозволяютьпаралельно виконуваним командам програми вести спостереження за фактом появи деяких подій (зміна рівня сигналу на виведенні кристала, тривалість деякого процесу і так далі).

Для зручності програмування AlgorithmBuilder підтримує спеціальний вид міток - мітки обслуговування переривань. Для обслуговування переривання звичайним шляхом потрібне розміщення за адресою вектора переривання команди безумовного переходу на відповідну підпрограму. При використанні спеціального виду компілятор реалізує це автоматично. Для цього необхідно дати мітці (вершині) стандартне ім'я переривання, і помітити її як макрооператор, натиснувши клавішу "F2", при цьому ім'я відображуватиметься жирним шрифтом. Зустрівши хоч би одну таку мітку в алгоритмі, компілятор заповнить вільний простір векторів переривання кодом повернення з підпрограми обслуговування переривання ("reti"), а по відповідному перериванню помістить код безумовного переходу на цю мітку.

Для того, щоб програма могла нормально стартувати, початком алгоритму має бути макромітка "Reset". Це забезпечить завантаження в нульову адресу безумовного переходу на початок алгоритму.

У виконуваній роботі використовується зовнішнє переривання *Int1*, яке дозволяє відстежувати зміну сигналу на виведенні PD.3. Стандартне ім'я переривання - ***External_1***. Для дозволу цього переривання необхідно встановити в "1" сьомий біт (*"Int1"*) регістра маски переривання GIMSK . Біти управління видом сигналу переривання 1(*InterruptSenseControl - ISC11 і ISC10*) в регістрі управління процесора MCUCR визначають, чи активізується зовнішнє переривання по зрізу(фронту) імпульсу або по рівню на контакті PD.3. У цій роботі переривання активізується по фронту сигналу, тобто біти *ISC11* і *ISC10* встановлені в "1" (так фіксується факт відтискання кнопки). При виникненні події, що викликає переривання, в регістрі основних прапорців переривання GIFR встановлюється в логічну одиницю біт *INTF1*. Процесор відстежує цей регістр і при виявленні встановленого прапора, якщо переривання дозволене, переходить до підпрограми обробки переривання. Біт *INTF1* при цьому апаратно очищається.

Виведення сигналу на лінійку світлодіодів у виконуваній роботі запропоновано виконувати по перериванню таймера/лічильникаа1 (16-бітового регістра **TCNT1** таймера). При цьому, коли регістр заповнюється повністю, то встановлюється прапор **TOV1** в регістрі прапорів переривань таймерів/лічильників **TIFR**. Регістр **TCNT1** може заповнюватися з різною швидкістю, залежно від джерела, що тактує таймер/лічильник1. Цей чинник визначає тривалість паузи між виводами. У цій роботі зручно прийняти тактування, як CK/64, при цьому необхідно встановити біти CS12, CS11 і CS10 регістра управління таймера/лічильника1 **TCCR1B** відповідно в - 0,1,1. Для дозволу переривання необхідно встановити сьомий біт **TOIE1** регістра маски переривань TIMSK. При цьому, якщо процесор виявляє встановлення прапора **TOV1**, він переходить до підпрограми обробки цього переривання. Після виходу з підпрограми біт **TOV1** очищується апаратно.

Зарезервоване ім'я підпрограми обробки переривання по переповнюванню таймера/лічильника 1 - **Timer_1_Overflow**.

Для глобального дозволу обробки переривань необхідно встановити в "1" біт **I** регістра стану **SREG.**

Як альтернативний варіант можливе використання настроювачів "SETTER" (рис. 3.1).

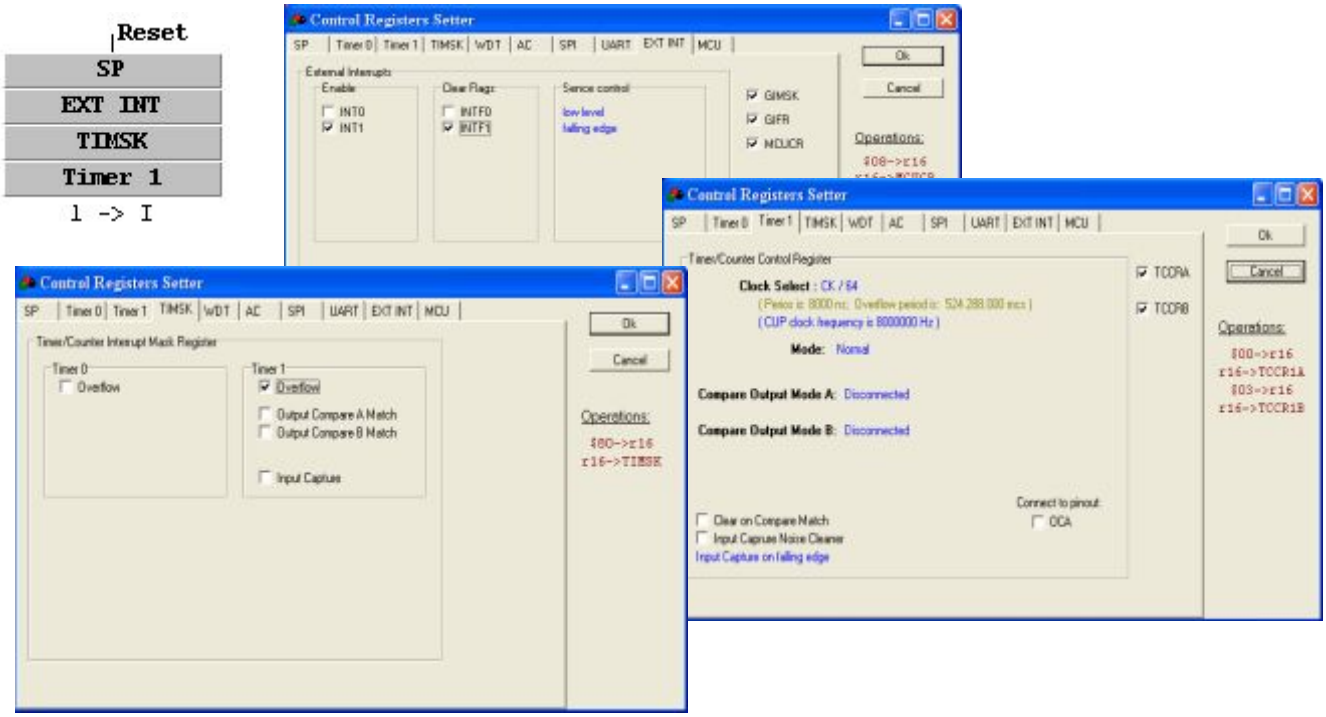

Рисунок 3.1 - Налаштування стека, таймера/счетчика1 і зовнішнього переривання INT1

У цій роботі крапка, що біжить, реалізується так само, як і в попередній роботі. У підпрограмі обробки переривання по таймеру реалізується формування утримуваного регістра "крапки" і виведення його в порт С. При цьому врахування напряму руху рядка ведеться прапорцем Direct. Цей прапор міняє своє значення в підпрограмі обробки зовнішнього переривання інвертуванням.

На рис.3.2 наведений алгоритм основної програми. На рис.3.3 зображений алгоритм підпрограми обробки переривання по таймеру. На рис.3.4 зображений алгоритм підпрограми обробки зовнішнього переривання.

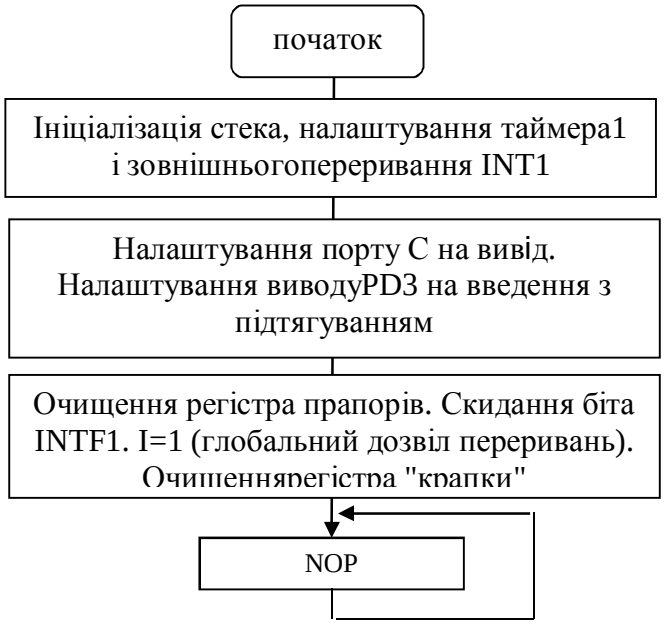

Рисунок 3.2 – Блок-схема основної програми

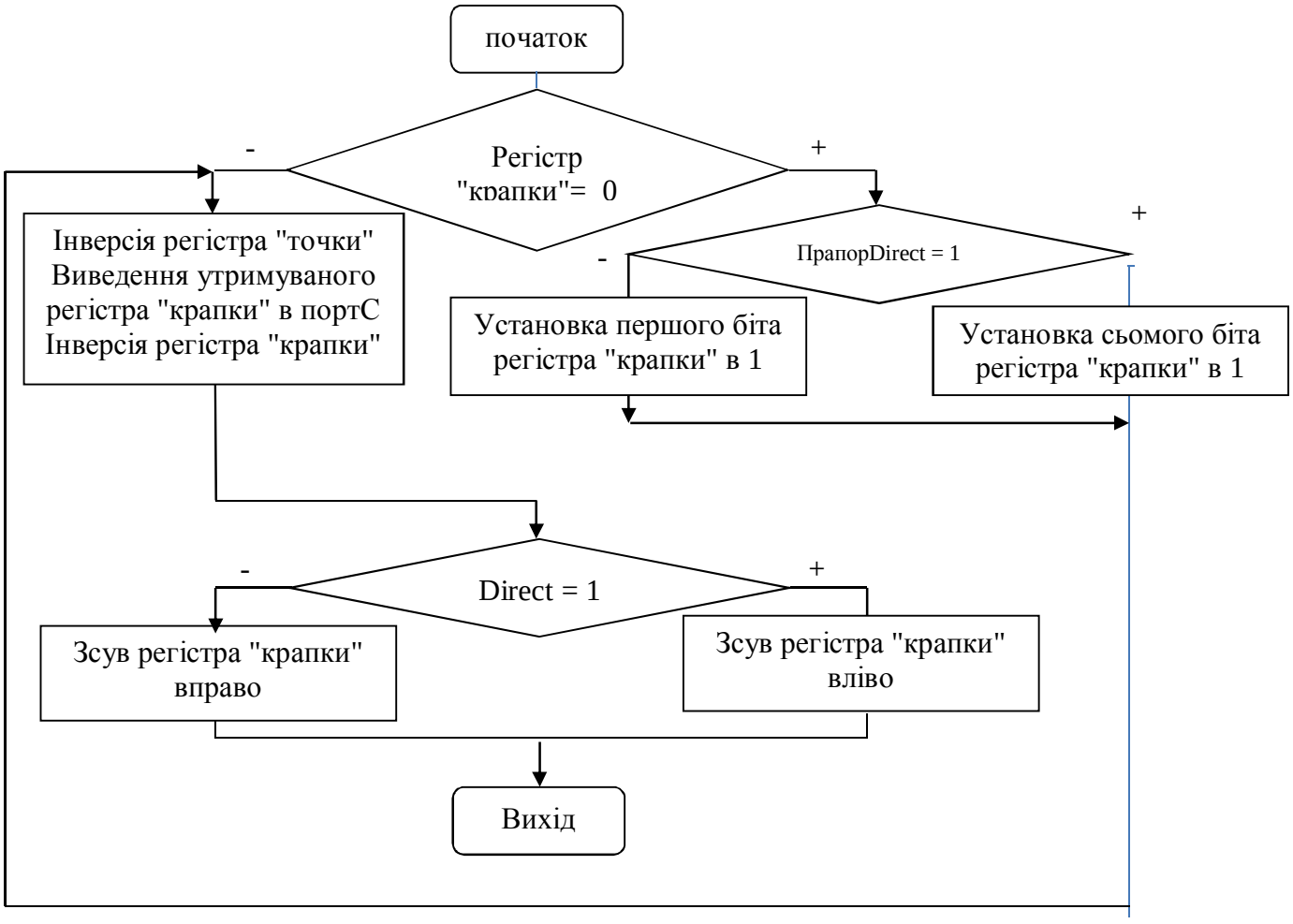

Рисунок 3.3 - Блок-схема підпрограми обробки переривань по таймеру

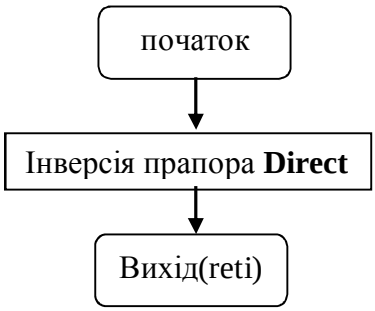

Рисунок 3.4 - Блок-схема підпрограми обробки зовнішнього переривання

2. Порядок виконання роботи.

2.1 Ознайомитися з принципом написання підпрограм обробки переривань в графічному середовищі, їх викликом в основній програмі.
2.2 Написати і відлагодити програму в покроковому режимі в симуляторі.
2.3 Перевірити працездатність програми на лабораторному стенді.
2.4 Виконати індивідуальне завдання, згідно варіанта (табл.2.1).

ЛАБОРАТОРНА РОБОТА №4

ОРГАНІЗАЦІЯ ЗВ'ЯЗКУ МІКРОКОНТРОЛЛЕРА З МАТРИЧНОЮ КЛАВІАТУРОЮ І СЕМИСЕГМЕНТНИМ СВІТЛОДІОДНИМ ІНДИКАТОРОМ

Ціль роботы: отримання навичок в складанні і відлагодженні програми в графічному середовищі AlgorithmBuilder для аналізу матричної клавіатури і виведення інформації на семисегментні індикатори.

4.1. Короткі відомості по темі.

Клавіатура є одним з широко поширених пристроїв введення в мікро-ЕОМ. При організації введення інформації з клавіатури в мікро-ЕОМ перед розробником ставиться ряд завдань, основними з яких є :

- визначення факту натиснення клавіші на клавіатурі:
- знаходження номера (кода) натиснутої клавіші;
- здійснення передачі управління на відповідну підпрограму.

У цій лабораторній роботі необхідно написати програму для аналізу матричної клавіатури і виведення інформації (цифр) на семи сегментний світлодіодний індикатор з використанням мови програмування AlgorithmBuilder. Результати роботи програми повинні зводитися до наступного: при натисненні на кнопку матричної клавіатури ("0" - "9") відповідна цифра повинні виводиться на світлодіодний індикатор. При цьому при натисненні на кнопки "*" і "#" не повинне відбуватися спрацьовування і на індикаторі повинне залишитися попереднє число. На стенді клавіатура організована у вигляді матриці кнопок 4х3.

Як пристрій виведення інформації використовується дисплей. Одним з найширше вживаних на практиці пристроїв виведення інформації є семи сегментний світлодіодний знаковий індикатор (ССЗІ). Кожен ССЗІ має сім випромінюючих сегментів A - G і крапку H (рис. 4.1).

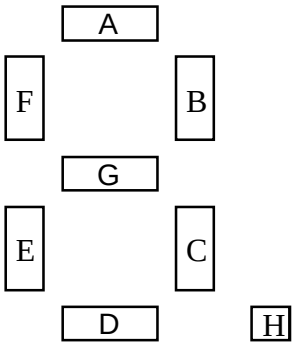

Рисунок 4.1 - Сегменти ССЗІ

Для реалізації виводу на одиничний ССЗІ в цьому стенді використовується порт С, вихідні буфери якого забезпечують вхідний струм до 20мА, що дозволяє використовувати його для безпосереднього управління світлодіодними індикаторами. Сегментам, що світяться, відповідають біти встановлені в 0, погашеним - 1. Коди для цифр "0"-"9" наведені в таблиці 4.1.

Таблиця 4.1 – Коди для ССЗІ

	H	**G**	**F**	**E**	**D**	**C**	**B**	**A**	**код**
1	1	1	1	1	1	0	0	1	$F9
2	1	0	1	0	0	1	0	0	$A4
3	1	0	1	1	0	0	0	0	$B0
4	1	0	0	1	1	0	0	1	$99
5	1	0	0	1	0	0	1	0	$92
6	1	0	0	0	0	0	1	0	$82
7	1	1	1	1	1	0	0	0	$F8
8	1	0	0	0	0	0	0	0	$80
9	1	0	0	1	0	0	0	0	$90
0	1	1	0	0	0	0	0	0	$C0

Аналіз стану натиснення кнопки матричної клавіатури виконується в наступному порядку:

1) Сканування по рядках. При цьому на контакти PD4 - PD7 видаються комбінації 0111, 1011, 1101, 1110 (старші біти регістра даних порту D). Таким чином, нульовий біт на PD4 вибирає кнопки 1...3, на PD5 - кнопки 4...6, на PD6 - кнопки 7...9, на PD7 - кнопки *…#.

2) Сканування по стовпцях. При цьому для кожної комбінації аналізується стан контактів PD0 - PD2 (молодші біти регістра вхідних контактів порту D) тим самим визначається позиція натиснутої кнопки (якщо така є).Якщо немає натиснутих кнопок, процес повторюється.

Цю лабораторну роботу доцільно реалізувати основною програмою з двома підпрограмами: підпрограми сканування по рядках і підпрограми визначення номера натиснутої кнопки і виведення інформації на ССЗІ. У основній програмі ініціалізується стек, настроюються порти: порт C і біти PD4 - PD7 порту D налаштовується на вивід, а біти PD0 - PD2 - на введення з підтяжкою. Для визначення номера натиснутої клавіші (у шістнадцятирічному форматі - $00-$0B) в цій роботі використовуються два лічильники комбінацій: перший лічильник, назвемо його *row*, набуває значень 0-3 залежно від вибраного рядка, тобто від поточної комбінації на контактах PD4 - PD7. Другий лічильник, назвемо його *column*, набуває значень 0-2 залежно від стану контактів PD0 - PD2, тобто від номера стовпця. Реалізовуються ці лічильники за допомогою двох будь-яких робочих регістрів, в які заносяться нулі на початку основної програми.

Як вже було сказано, процедура сканування по рядках реалізується за допомогою підпрограми. У цій підпрограмі залежно від значення лічильника row змінюються комбінації на контактах PD4 - PD7. Таким чином, кожному ряду кнопок 1...3, 4...6, 7...9, *…# відповідає значення лічильника *row* - 0, 1, 2, 3 відповідно. Інкремент і обнуління лічильника row для наступного циклу сканування проводиться в основній програмі.

Потім в основній програмі аналізується стан контактів PD0 - PD2. Якщо на якому-небудь з контактів з'являється логічний 0 (що відповідає натисненню кнопки), то в лічильник column заноситься значення, відповідне номеру контакту. Тобто, якщо з'являється активний рівень на PD0, то column набуває значення 0. Аналогічно для PD1 - column набуває значення 1, для PD2 - 2.

Блок-схема основної програми представлена на рис. 4.2. Блок-схема підпрограми сканування по рядках зображена на рис 4.3.

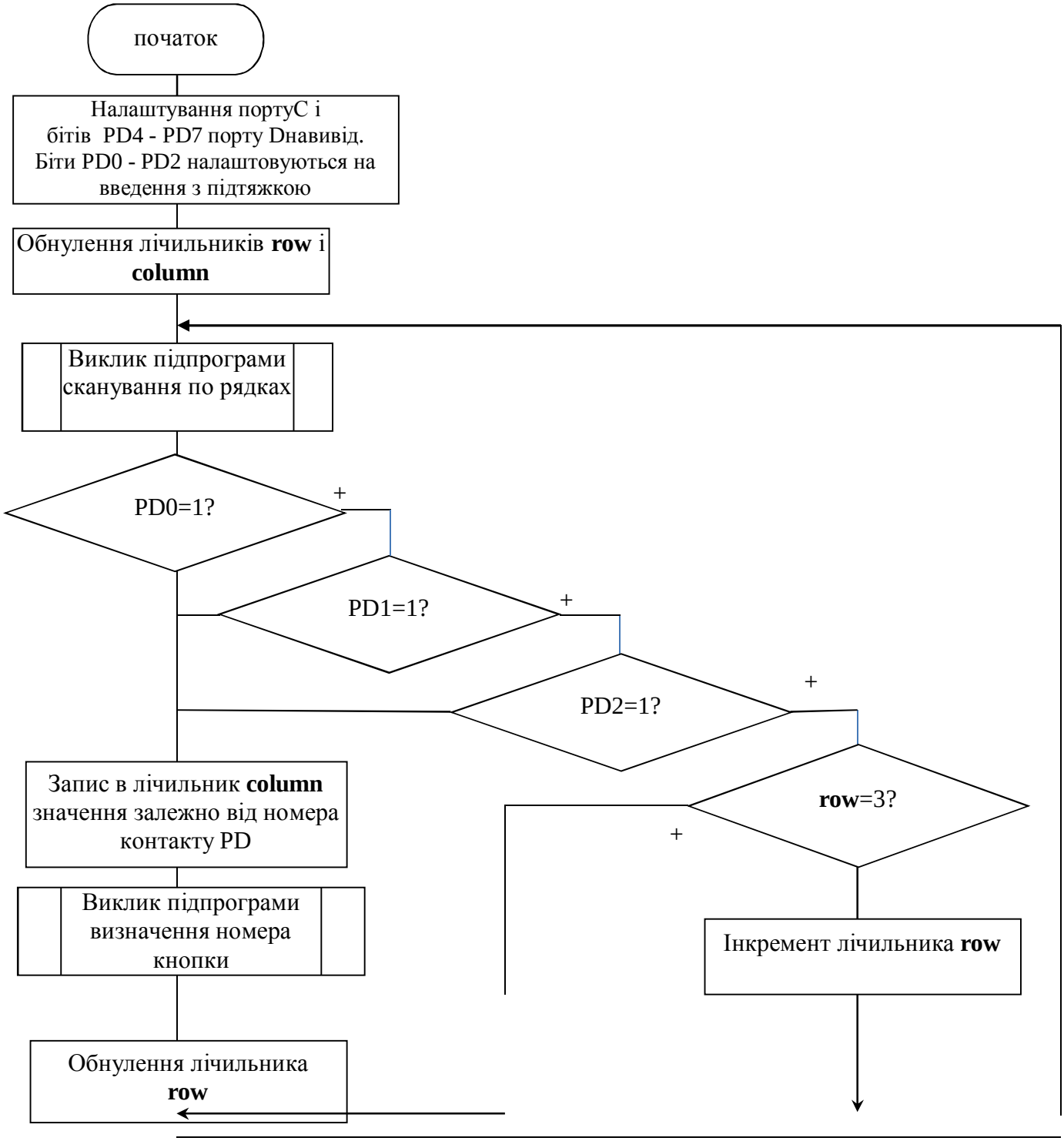

Рисунок 4.2 - Блок-схема основної програми

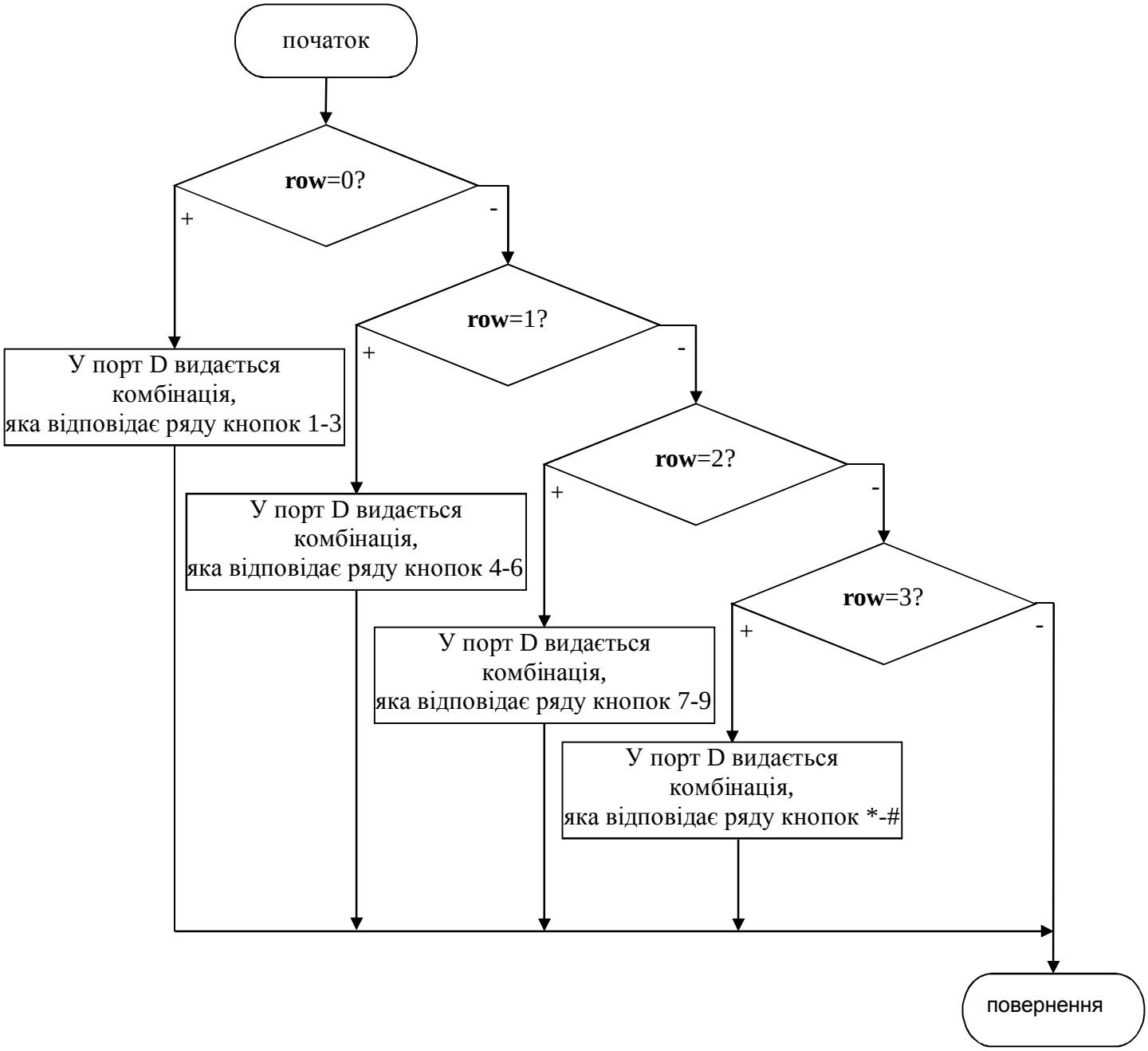

Рисунок 4.3 - Блок-схема підпрограми сканування по рядках

Визначення номера натиснутої клавіші і виведення інформації на ССЗІ здійснюється в підпрограмі, зображеній рис. 4.4 і 4.5. У цій підпрограмі спочатку визначається номер натиснутої клавіші. Для цього здійснюється множення на 3 значення лічильника row, внаслідок чого буде отриманий номер першого елементу у вибраному рядку (множення на 3 може бути реалізоване як множення на 2 шляхом зсуву числа вліво з наступним додаванням до результату початкового числа). Потім до набутого значення необхідно додати значення лічильника column, тобто номер стовпця. Наприклад, при натисненні на клавіатурі цифри "5", лічильник row дорівнює 1. Значення лічильника column також дорівнює 1. Номер натиснутої клавіші : 1*3+1=4. Тепер необхідно отримати код числа, відповідний номеру натиснутої клавіші.

Завдання кодів цифр, що виводяться на ССЗІ, здійснюється за допомогою масиву, який розташовується в програмній пам'яті мікроконтроллера:

```
                    ¦codetable
DB: $f9, $a4, $b0, $99, $92, $82, $f8, $80, $90, $c0
```

DB - формат даний (8-бітове число);
$F9, $A4, . - елементи масиву (числа в шістнадцятирічному форматі, відповідні цифрам 1, 2, .).Робота з масивом здійснюється за допомогою наступної послідовності команд:

```
CODETABLE*2 -> Z
Z+R3
LPM
```

Спочатку визначаться адреса пам'яті, по якій розташований масив. Для цього використовується регістр Z, який виступає індексом масиву. Потім знаходиться індекс необхідного елементу з масиву шляхом складання з регістром Z робочого регістра, в якому записаний номер натиснутої клавіші. У прикладі з натиснутою цифрою "5" до регістра Z додасться регістр, в якому записане $04. За допомогою команди lpm в регістр r0 завантажиться байт з програмної пам'яті за адресою Z. Тобто в r0 запишеться елемент масиву, відповідно до індексу. Таким чином, в прикладі після виконання команди lpm в регістр r0 буде занесено число з масиву з індексом 4, тобто $92, що по таблиці 1 відповідає "5" Після чого утримуване регістра r0 може бути виведено на ССЗІ.

У підпрограмі також необхідно проводити перевірку у разі натиснення клавіш "0", "*" або "#". Якщо після визначення номера натиснутої клавіші, він дорівнює $09 або $0B, що відповідає натиснутим клавішам "*" або "#", то відбувається повернення в основну програму без виконання яких-небудь дій. Якщо номер натиснутої клавіші рівний $0A (тобто натиснута клавіша "0"), то в робочий регістр, який складається з Z, записується індекс елементу масиву, який відповідає цифрі "0", тобто $09.

Після відображення числа на ССЗІ, відбувається повернення в основну програму, і увесь процес повторюється наново.

2 Порядок виконання роботи.

2.1 Ознайомиться з системою команд мови програмування AlgorithmBuilder, описом робочого стенду, керівництвом по використанню МК AT90S8515.
2.2 Відповідно до завдання скласти алгоритм і написати програму опитування матричної клавіатури і виводу на світлодіодний індикатор.
2.3 Відлагодити програму в покроковому режимі в симуляторі і прошивку кристала.
2.4 Перевірити працездатність програми на лабораторному стенді.
2.5 Виконати індивідуальне завдання, видане викладачем.

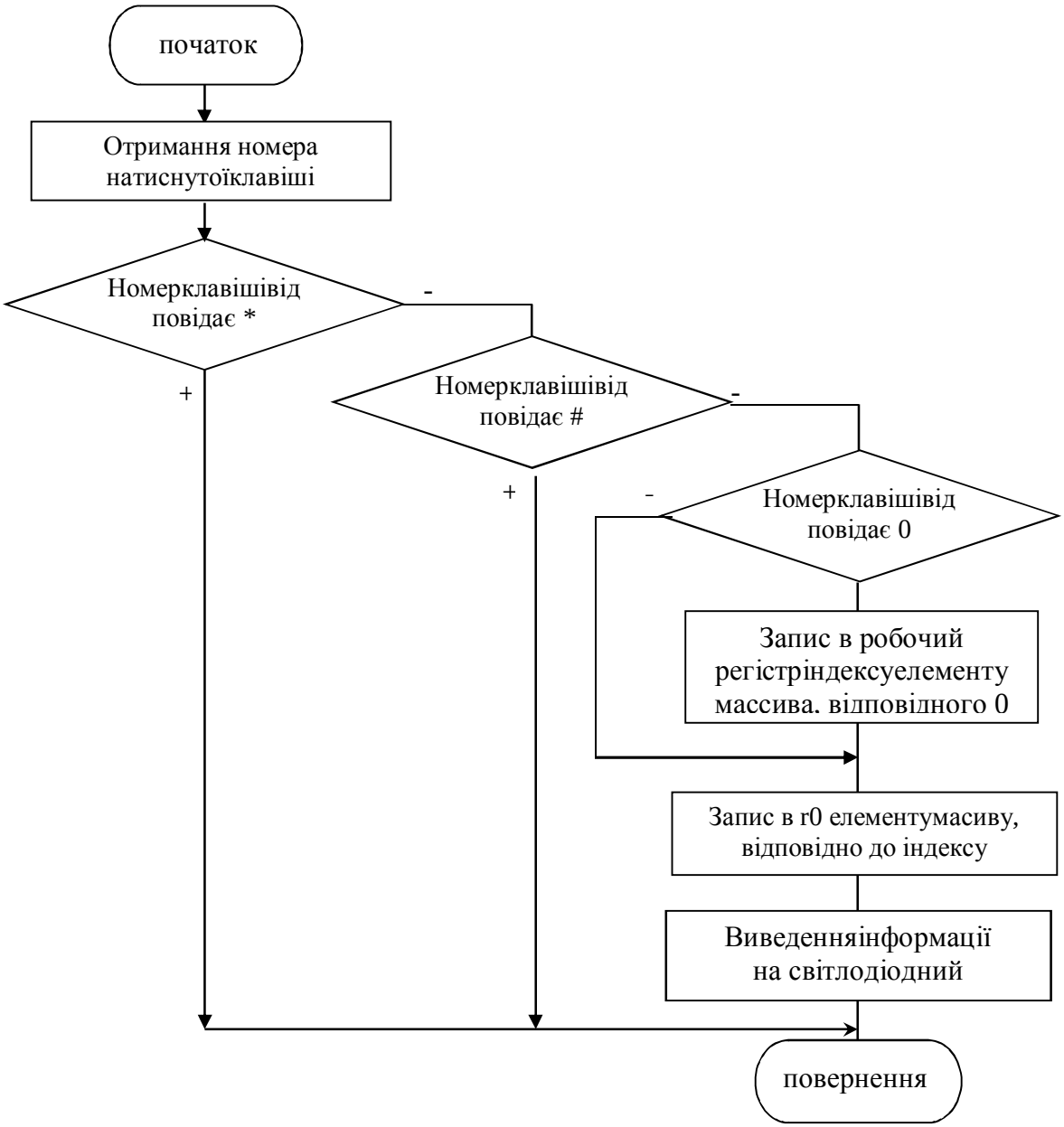

Рисунок 4.4 - Блок-схема підпрограми визначення номера натиснутої клавіші і виведення інформації наССЗІ

ЛАБОРАТОРНА РОБОТА №5

ОРГАНІЗАЦІЯ ДИНАМІЧНОЇ ІНДИКАЦІЇ

Ціль роботи: отримання навичок в складанні і відлагодженні програми в середовищі AlgorithmBuilder для аналізу матричної клавіатури і виведення цифрової інформації на багаторозрядний індикатор в режимі динамічної індикації; отримання навичок розподілу ресурсів і роботи з непрямою адресацією.

1. Короткі відомості з теми.

У цій лабораторній роботі необхідно реалізувати програмне забезпечення з використанням мови програмування AlgorithmBuilder для аналізу матричної клавіатури і реалізації динамічної індикації.

Результати роботи програми повинні зводитися до наступного: при натисненні на кнопку матричної клавіатури ("0" - "9") відповідна цифра повинні виводитися на поточне знакомісце індикатора (початковим знакомісцем є крайнє ліве). При цьому при натисненні на кнопки "*" і "#" не повинне відбуватися спрацьовування і на індикаторі повинне залишитися попереднє число. Після цього поточним стає знакомісце, розташоване праворуч. Після виведення інформації на 4 знакомісце (крайнє праве), поточним стає 1 знакомісце (крайнє ліве). Виведення інформації необхідно виробляти за допомогою динамічної індикації. Динамічна індикація реалізується таким чином: спочатку в порт А виводиться код, який викликає свічення сегментів індикатора, що формують необхідну цифру (0, 1, 2 .), а потім відкривається перший транзисторний ключ, що включає перше знакомісце, шляхом виводу 0 в PORTD.4. При цьому відображуватиметься тільки перше знакомісце індикатора. Через 5 мс необхідно закрити перший транзисторний ключ, вивести в порт А код, який відповідає цифрі, що виводиться на другий розряд, і відкрити другий транзисторний ключ. При цьому відображуватиметься тільки друге знакомісце індикатора. Далі ця процедура повторюється для кожного розряду і після відображення четвертого розряду цикл повторюється. Таким чином, час оновлення кожного розряду складає 5мс, а усього індикатора - 5 * 4 = 20 мс. При такій частоті оновлення людське око не помічає мерехтіння і сприймає індикацію як статичну.

Оскільки сканування клавіатури і індикація виконується за допомогою одних і тих же ліній, опитування клавіатури буде пов'язано з індикацією. Наприклад, якщо в даний момент на індикаторі відображується перший розряд (присутній 0 на лінії PORTD.4), то одночасно подається 0 і на перший рядок клавіатури, тому можна рахувати стан трьох кнопок цього рядка. Таким чином, номер розряду індикатора, який відображується в даний момент, буде номером рядка клавіатури, яку можна сканувати в даний момент.

При обробці клавіатури для запобігання багатократній реакції на одну і ту ж натиснуту в даний момент кнопку, після фіксації факту натиснення необхідно блокувати наступне введення даних з клавіатури до моменту відпуску цієї кнопки, тобто фактично до того моменту, коли не буде натиснута жодна з кнопок клавіатури. Крім того, при роботі з клавіатурою необхідно передбачити захист від

брязкоту, як при натисненні, так і при відпуску кнопки. Тому, зокрема, зняття блокування слід виробляти після закінчення деякого часу, наприклад 40 тактів спрацьовування таймера, що відповідає затримці в 200 мс.
Взаємна ув'язка підпрограм опитування клавіатури і виведення інформації на індикатор повинна здійснюватися в основному елементі програми - в підпрограмі обробки переривання по переповнюванню таймера, що викликається кожні 5 мс. Як таймер в даному випадку може бути використаний наявний в мікроконтроллері AT90S8515 таймер_0. При виборі частоти того, що тактує СК/256, де СК - частота тактового генератора рівна 8 Мгц, період переповнювання дорівнює 8.192 мс. Для отримання часу переповнювання в 5 мс необхідно в регістр **TCNT0** занести число $64 (100), виходячи з таких міркувань:

$T_{переп}$= 256·(256-N)·=256·(256-100) ·≈5 мс.

Перш ніж приступати до написання програми слід визначитися з потребою і розподілом регістрів мікроконтроллера під зберігання необхідних змінних. В даному випадку доцільно передбачити використання в програмі наступних змінних :

Лічильник знакомісць індикатора;

Лічильник рядків клавіатури;

Регістр для зберігання номера (індексу) натиснутої кнопки в рядку;

Регістр-лічильник паузи для реалізації антибрязкоту (чи два такі регістри, окремо для натиснення і відпуску кнопки);

Прапор блокування клавіатури;

4 регістри зберігання даних для виводу на індикатор, по одному для кожного знакомісця.

Для зберігання даних, таких, що виводяться на індикатор можна використовувати оперативну пам'ять мікроконтроллера. Для цього необхідно задати змінну типу масив однобайтових чисел в оперативній пам'яті мікроконтроллера. Робиться це таким чином. У середовищі AlgorithmBuilder треба натиснути клавішу F12, після чого з'явиться вікно таблиці даних. У розділі SRAM треба заповнити відповідні поля, а саме: в полі Name задати ім'я змінної, по якому потім можна буде до неї звернеться; поле адресу можна залишити порожньою, в цьому випадку змінна автоматично розміститься в пам'яті за першою доступною адресою; поле Format також можна залишити порожнім, оскільки за умовчанням тип змінної відповідатиме цілій величині в 1 байт; у полі Count заноситься кількість елементів масиву, в нашому випадку 4; у полі Commentary можна вказати коментар до змінної.
Звернення до такої змінної здійснюється за допомогою непрямої адресації. Щоб прочитати вміст елементу масиву, треба виконати наступну команду:

z_mesto -> r1,

де z_mesto - ім'я змінної. Проте при цьому в регістр r1 буде переписано значення, що зберігається в нульовому осередку масиву. Щоб дістати доступ до інших осередків, спочатку треба отримати адресу, відповідну потрібному осередку, а

потім набути значення, що зберігається за цією адресою. Тому, щоб прочитати вміст, наприклад, першого осередку (індексація осередків починається з нуля) необхідно виконати наступну команду:

```
[@z_mesto+1] -> r1.
```

Знак @ означає отримання адреси змінною, він ставиться перед її ім'ям. Оскільки після імені змінної стоїть +1, то буде отримана адреса не нульової, а першоїкомірки масиву. Квадратні дужки означають набуття значення, що зберігається за адресою, вказаною в них. Таким чином, розглянута команда перепише вміст першоїкомірки масиву в регістр r1. Аналогічно, попередня команда може бути переписана таким чином:

```
[@z_mesto] -> r1.
```

Запис в комірки масиву аналогічним чином:

```
$FF -> [@z_mesto+2].
```

Після виконання цієї команди число $FF буде записано в другукомірку масиву.

Використання непрямої адресації дозволяє легко визначити число, яке необхідно вивести в потрібне знакомісце індикатора. Для цього обчислюється зміщення відносно нульового індексу, а по ньому визначається потрібне значення. Нехай, наприклад, в даний момент сканується третій рядок клавіатури і відповідно відкритий третій транзисторний ключ. Отже, необхідно в *PORTA* вивести код, відповідний цифрі, яка повинна відображуватися в третьому знакомісці індикатора. Маючи регістр, в якому зберігається значення поточного сканованого рядка це легко зробити, підставивши його значення замість зміщення. Проте наступна команда неприпустима:

```
[@z_mesto+r2] -> PORTA,
```

де r2 - регістр, в якому зберігається значення поточного сканованого рядка. Щоб реалізувати таку команду слід використовувати проміжний двобайтовий регістр X, Y або Z. Необхідно в один з цих регістрів, наприклад X, переписати адресу нульовоїкомірки масиву, потім додати до цього регістра зміщення і, нарешті, переписати значення за отриманою адресою в PORTA. Реалізується це наступною послідовністю команд:

```
@z_mesto -> X
X+r2
[X] -> PORTA
```

При розробці алгоритму необхідно передбачити відсутність повторного виводу на індикатор цифри при затиснутій кнопці, тобто при натисненні на кнопку відповідна цифра повинна з'явитися на індикаторі один раз навіть при

утриманні кнопки. Слід пам'ятати, що при скануванні клавіатури в певний момент часу відкритий лише один з транзисторних ключів. Тому відсутність низького логічного рівня на відповідному виводі порту D (PIND.0, PIND.1 або PIND.2) зовсім не означає, що кнопка відтиснута. Наприклад, нехай був зафіксований факт натиснення кнопки відповідній цифрі "2", яка розташована в першому рядку клавіатури. При обробці наступного переривання через 5 мс скануватиметься вже другий рядок клавіатури. Навіть якщо при цьому кнопка, відповідна цифрі "2", не буде відтиснута, ні на одному з виводів порту D (PIND.0, PIND.1 або PIND.2) не буде низького логічного рівня. Тому необхідно враховувати цю особливість при розробці алгоритму.

Блок-схема основної програми представлена на рис.5.1., блок-схема обробки переривання таймера на рис. 5.2, підпрограма індикації на рис. 5.3, підпрограма оновлення даних для виводу на індикатор на рис. 5.4, підпрограма сканування клавіатури ні рис. 5.5.

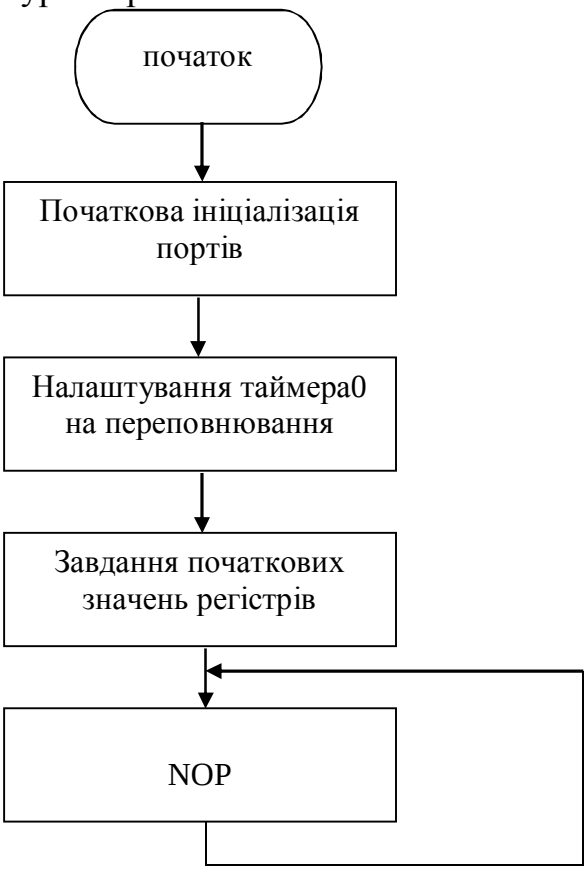

Рисунок 5.1 - Блок схема опитування матричної клавіатури і динамічної індикації

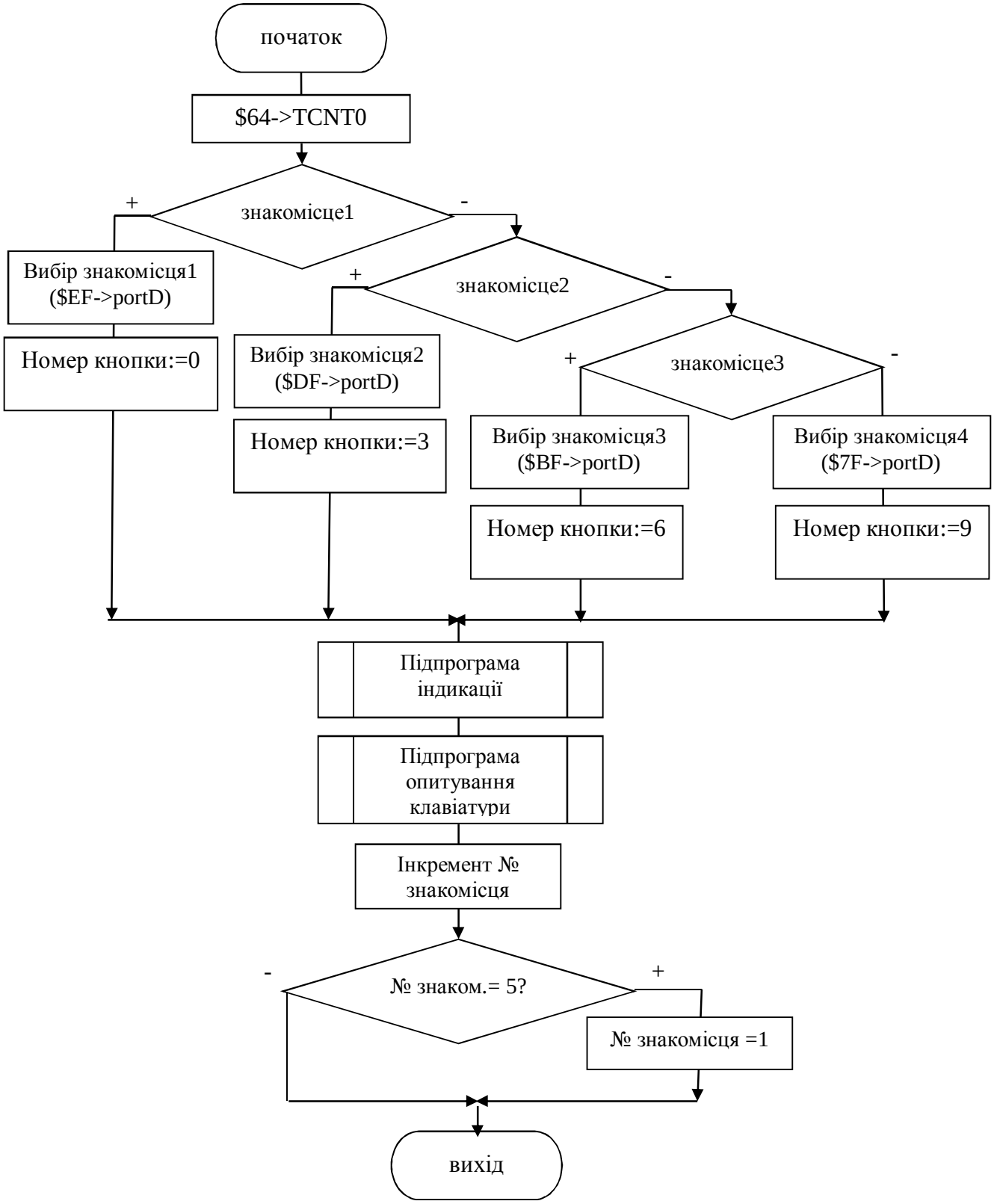

Рисунок 5.2 – Блок-схема підпрограми обробки переривання таймера

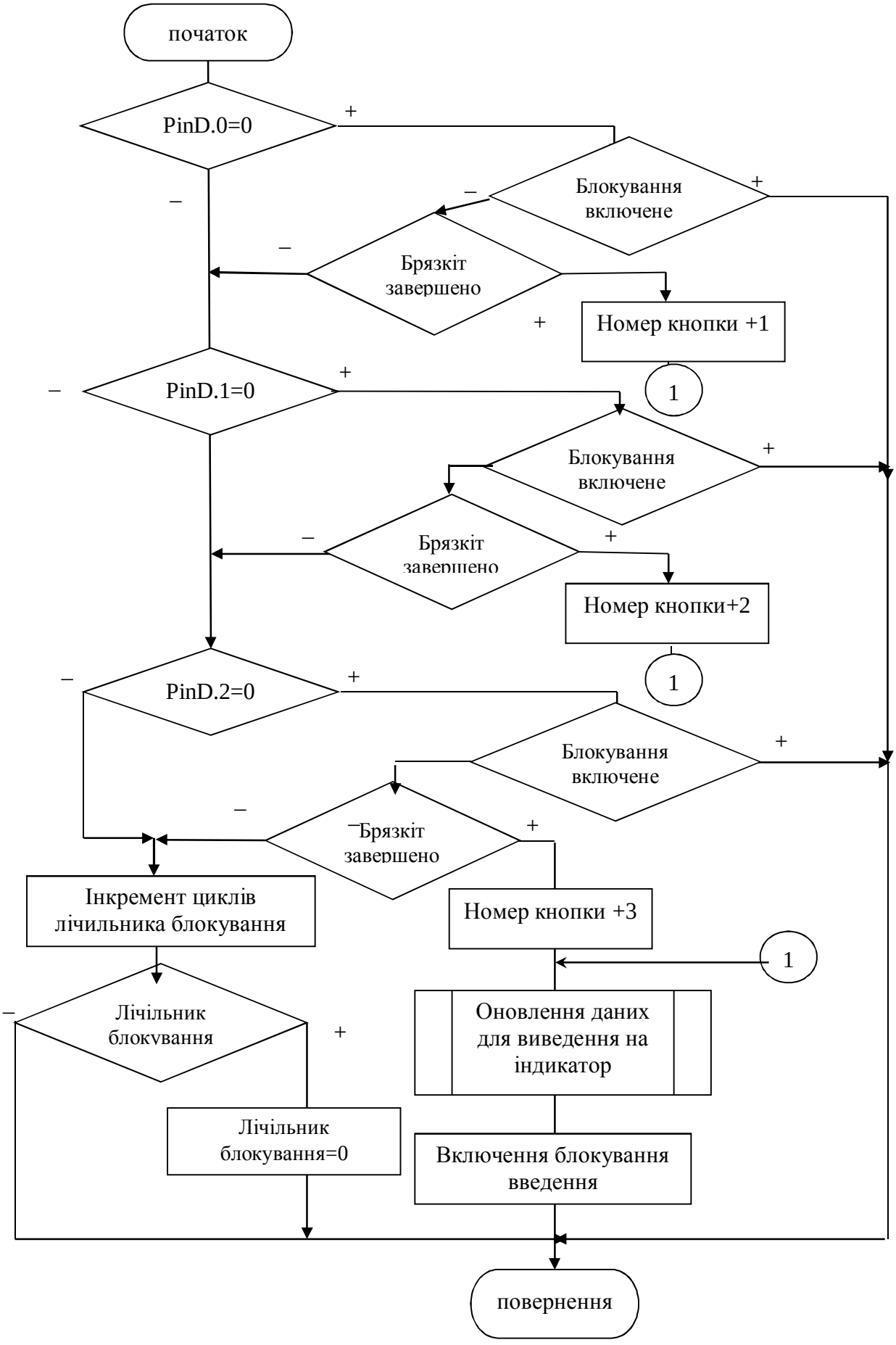

Рисунок 5.3 - Блок-схема підпрограми сканування клавіатури

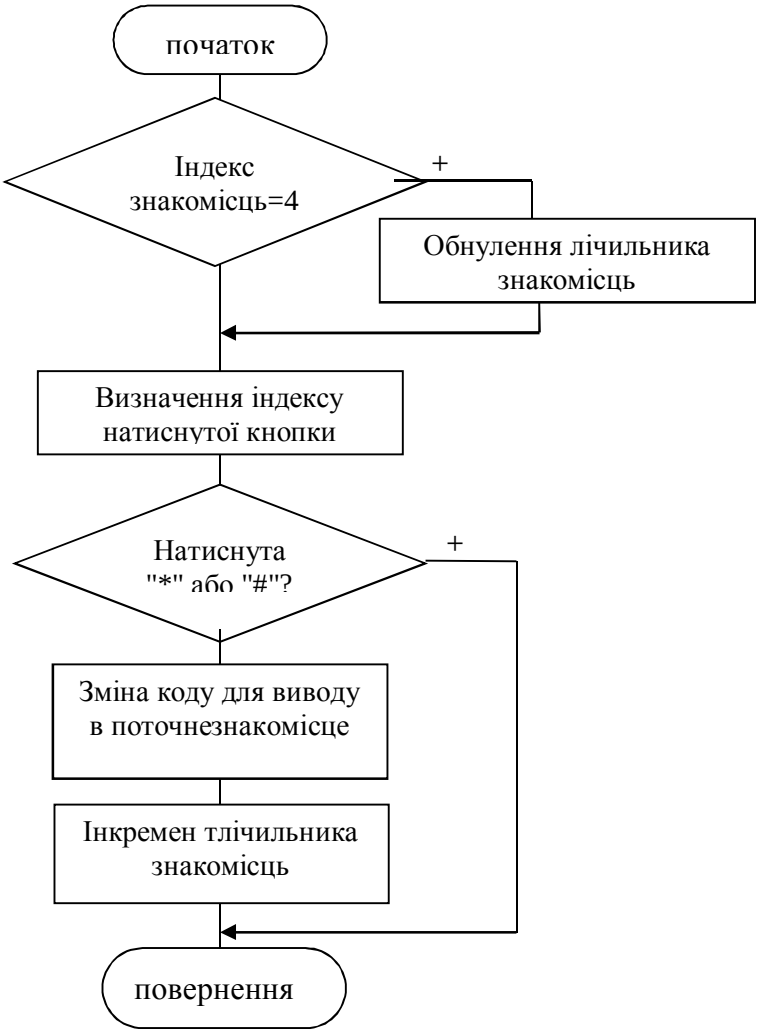

Рисунок 5.4 - Блок-схема підпрограми оновлення даних для виводу на індикатор

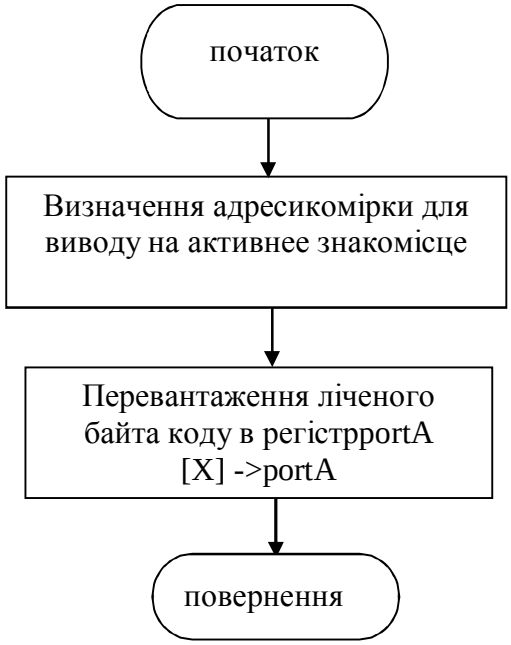

Рисунок 5.5 - Блок-схема підпрограми індикації

2. Порядок виконання роботи.

2.1 Ознайомиться з системою команд мови програмування AlgorithmBuilder, описом робочого стенду, керівництвом по використанню МК AT90S8515.
2.2 Відповідно до завдання скласти алгоритм і написати програму опитування матричної клавіатури і динамічної індикації.
2.3 Відлагодити програму в покроковому режимі в симуляторі і прошити кристал.
2.4 Перевірити працездатність програми на лабораторному стенді.
2.5 Виконати індивідуальне завдання, видане викладачем.

ЗМІСТ

Навчальне видання

Методичні вказівки до виконання лабораторних робіт з дисципліни
«Обчислювальна техніка і мікропроцесори»
для студентів напряму 6.050903 “Телекомунікаційні системи та мережі”
і «Мікропроцесорні системи»
для студентів напряму 6.050201 “Системна інженерія”
(усіх форм навчання)

Укладачі:

Суков Сергій Феліксович, к.т.н., доцент
Яремко Ігор Миколайович, ст.викладач
Долгіх Ірина Петрівна, ст.викладач
Батир Семен Сергійович, асистент

Затверджені на засіданні Навчально-видавничої ради ДонНТУ
Протокол № 4 від 07.10.2010 Р.№339 1,88 друк.арк.