МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
КАФЕДРА ЕЛЕКТРИЧНОЇ ІНЖЕНЕРІЇ

# МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт

з дисципліни «Електроніка та мікропроцесорна техніка»

для здобувачів першого (бакалаврського) рівня вищої освіти

спеціальності 141 Електроенергетика, електротехніка та електромеханіка

освітньо-професійної програми

Електроенергетика, електротехніка та електромеханіка

всіх форм навчання

Частина 2

Кривий Ріг
2025 р.

Укладачі: Федотов В. О., канд. техн. наук, доцент;
Сьомочкин А. Б., канд. техн. наук, доцент;
Осадчук Ю. Г., канд. техн. наук, доцент.

Відповідальний за випуск: Федотов В. О., канд. техн. наук, доцент.

Рецензент: Сінчук О. М., д-р. техн. наук, професор.

Методичні вказівки містять короткі теоретичні відомості, опис експериментальної частини, схеми досліджень, алгоритми, програми, часові діаграми роботи схем, питання для самоконтролю, список рекомендованої літератури.

**ЗМІСТ**

# ВСТУП

Електронні пристрої широко використовуються як в промисловості так і побутовій техніці. Без них неможливо уявити сучасне обладнання для автоматики та телемеханіки, засобів зв'язку, метрології, вимірювальній техніки, медицини побутових приладів та інше

Методичні вказівки до виконання лабораторних робіт з дисципліни «Електроніка та мікропроцесорна техніка» містять короткі теоретичні відомості, опис експериментальних частин для виконання лабораторних робіт, схеми досліджень, часові діаграми роботи схем, алгоритми та програми роботи мікропроцесорної системи, питання для самоконтролю та список рекомендованої літератури.

При виконанні лабораторних робіт здобувачі вищої освіти повинні скласти технічний документ у вигляді зошита зі звітами по лабораторним роботам, які складаються з тем та цілей лабораторних робіт, креслень досліджуваних схем, таблиць вимірюваних параметрів, побудованих графіків необхідних залежностей, розрахунків, часових діаграм роботи, алгоритмів та програм роботи з відповідними поясненнями.

Метою виконання лабораторних робіт є поглиблення теоретичних знань здобувачів вищої освіти, набуття навичок в дослідженнях роботи електронних схем та схем мікропроцесорних пристроїв, виконанні розрахунків елементів електроніки, побудові характеристик та часових діаграм їх роботи, складанню алгоритмів та написанню програм роботи, набуття досвіду роботи із спеціалізованими приладами.

**Лабораторна робота № 1**

**Тема: Засоби для розробки схем та програм мікропроцесорних систем**

**Мета: Ознайомитись із засобами для розробки схем та програм мікропроцесорних систем та отримати навички роботи з ними.**

1. MCU 8051 IDE – інтегроване середовище розробки мікроконтролерів на базі Intel 8051 (рис. 1.1).

Середовище підтримує мови програмування С та асемблер, має свій власний симулятор та власний асемблер, підтримує два інші зовнішні асемблери. Для мови С використовується компілятор SDCC.

Симулятор MCU має велику кількість функцій налагодження: запуск програми в режимі «крок за кроком», засіб перегляду переривань, перегляд зовнішньої пам'яті, засіб перегляду пам'яті коду та багато інших.

Середовище MCU 8051 IDE має симулятор для електронних периферійних пристроїв, таких як світлодіоди, світлодіодні дисплеї, світлодіодні матриці, РК-дисплеї, тощо.

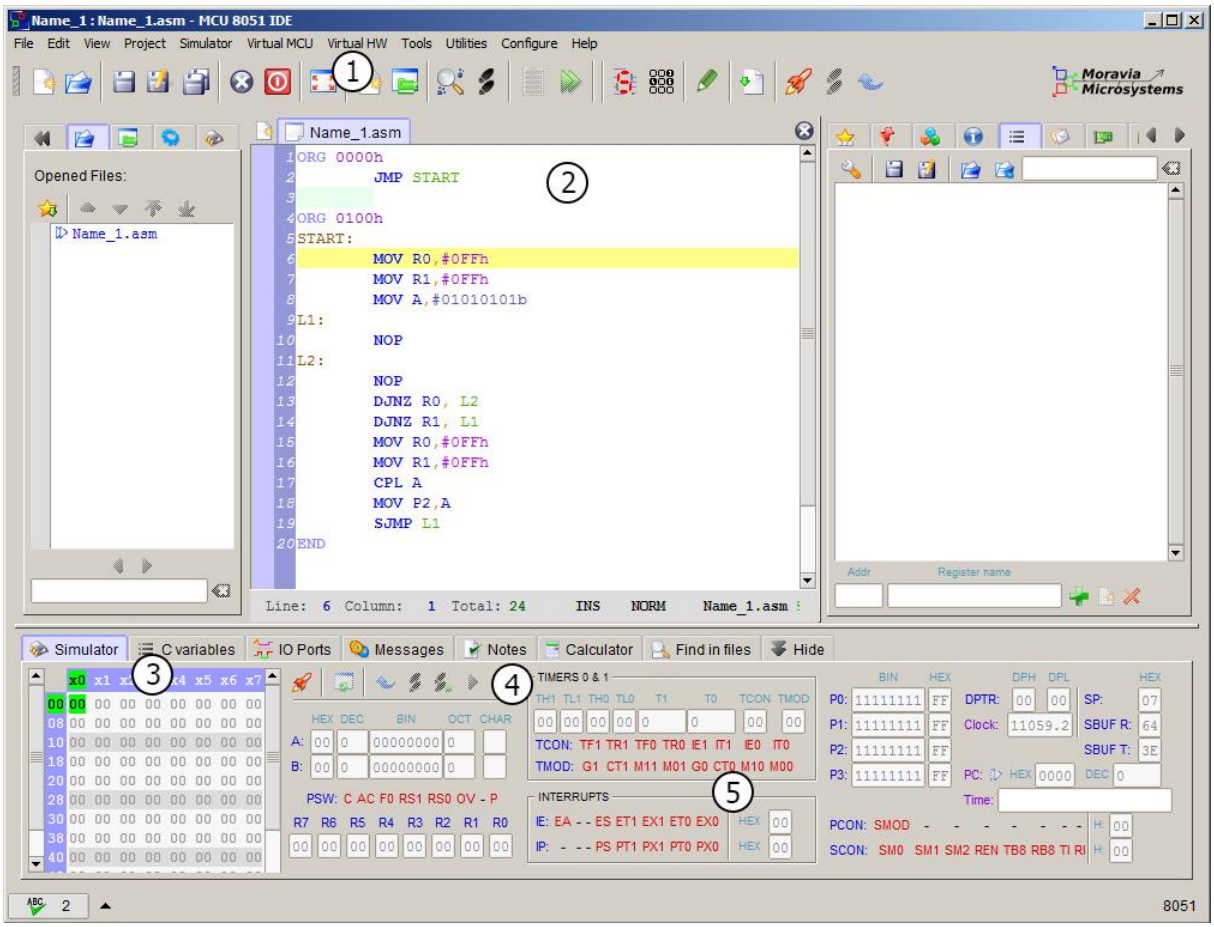



Рисунок 1.1 – Світлина утиліти для програмування мікроконтролерів STC-ISP

Створити проєкт та виконати відповідні налагодження можна через меню

(1). Програми проєкту записуються у відповідній світлині (2). Результати симуляції отримуються в світлинах (3). Запускати компіляцію проєкту та симуляцію роботи мікроконтролера можна використовуючи кнопки (4). У світлині (5) відображається вміст регістрів мікроконтролера та час роботи.

2. STC-ISP – утиліта для програмування мікроконтролерів (рис. 1.2).

Утиліта дозволяє обирати тип мікроконтролера, обирати COM порт та налагоджувати швидкість порту, обирати програмний код та записувати програму в пам'ять мікроконтролера.

Після отримання файлу із кодом в середовищі MCU 8051 IDE із розширенням hex або bin необхідно записати цей код в пам’ять мікроконтролера. Для чого потрібно обрати потрібний тип мікроконтролера (1) та обрати COM порт, через який навчальний стенд А2 під’єднаний до комп’ютера (2). Відкрити файл із кодом, натиснувши на кнопку Open Code File (3). Перевірити під’єднання мікроконтролера натиснувши кнопку Check MCU (4). Запустити програмування Re-Program натиснувши кнопку (5) та натиснути кнопку подачі живлення на навчальний стенд А2. Дочекатись кінця процедури програмування мікроконтролера.

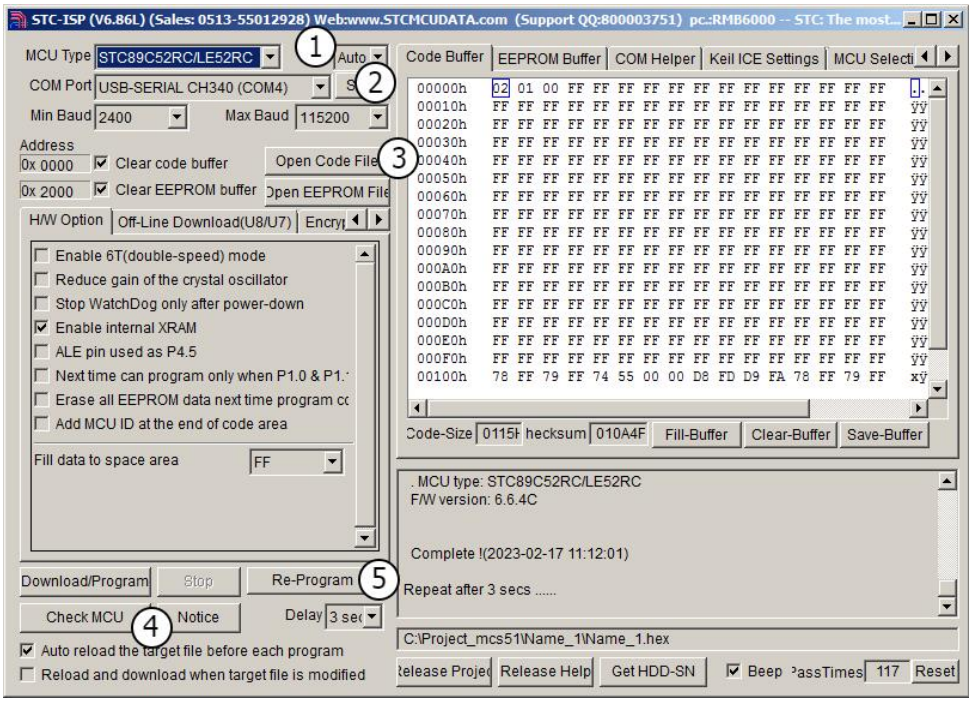



Рисунок 1.2 – Світлина утиліти для програмування мікроконтролерів STC-ISP

3. PROTEUS DESIGN – пакет програм для автоматизованого проєктування електронних схем.

Пакет базується на основі моделей електронних компонентів прийнятих в PSpice. PROTEUS DESIGN дозволяє моделювати роботу програмованих

пристроїв: мікроконтролерів, мікропроцесорів, DSP та ін. Пакет складається з двох програм: ISIS – програма синтезу та моделювання електронних схем та ARES – програма розробки друкованих плат.

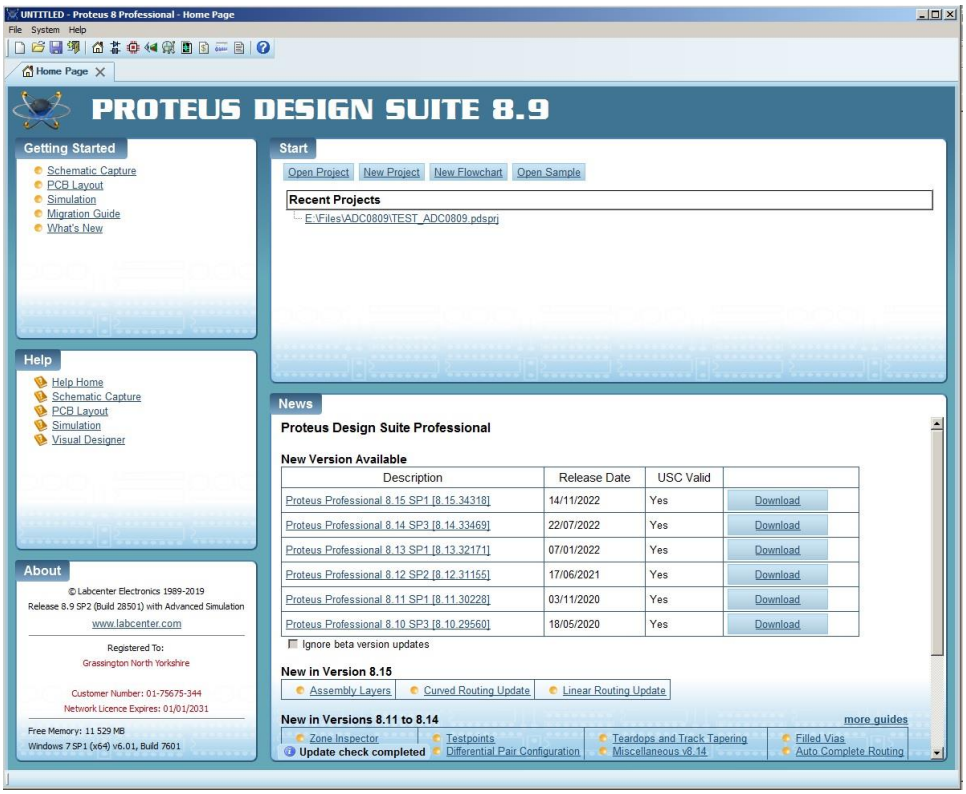



Рисунок 1.3 – Зовнішній вигляд світлини пакету PROTEUS DESIGN

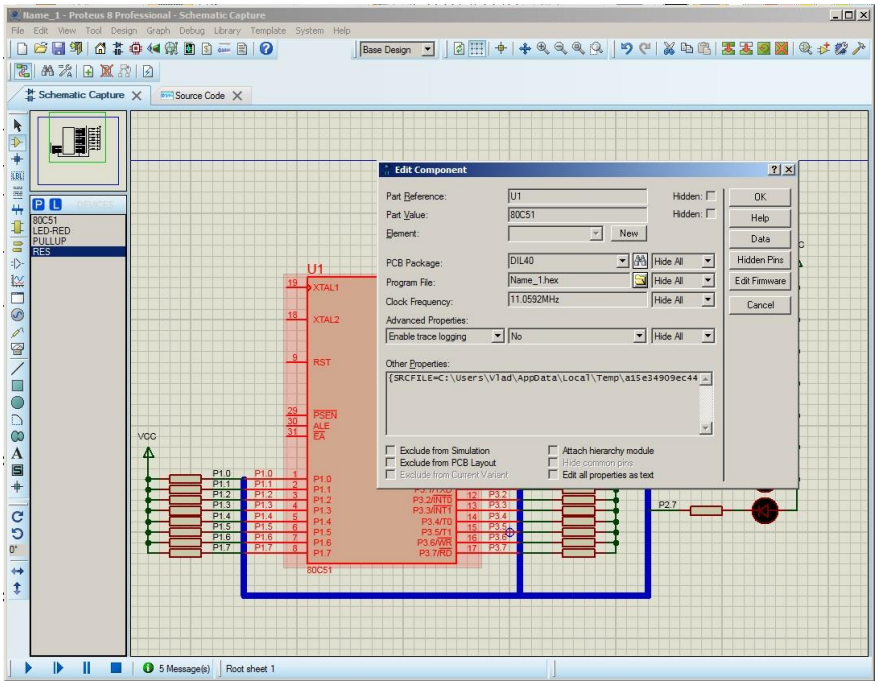



Рисунок 1.4 – Підключення до мікроконтролера файлу з програмою

Новий проєкт в пакеті PROTEUS DESIGN створюється натисканням на «New Project» (рис. 1.3). Потім в світлинах, що відкриваються, робляться

необхідні налаштування.

Після створення нового проєкту потрібно обрати необхідні компоненти та зробити з'єднання між ними. Для моделювання роботи мікроконтролера підключається файл з кодом програми (рис. 1.4), який був отриманий після компіляції проєкту в середовищі MCU 8051 IDE.

Хід роботи:

1. В інтегрованому середовищі MCU 8051 IDE створити новий проєкт та виконати необхідні налаштування.

2. Ввести код програми прикладу.

```
ORG 0000h
        JMP START
ORG 0100h
START:
        MOV R0,#0FFh
        MOV R1,#0FFh
        MOV R2,#010h
        MOV A,#01010101b
L3:
        NOP
L1:
        NOP
L2:
        NOP
        DJNZ R0, L2
        MOV R0,#0FFh
        DJNZ R1, L1
        MOV R1,#0FFh
        DJNZ R2, L3
        MOV R0,#0FFh
        MOV R1,#0FFh
        MOV R2,#010h
        CPL A
        MOV P2,A
        SJMP L1
END
```

3. Виконати симуляцію роботи мікроконтролера з програмою прикладом в режимі «крок за кроком».

4. Виконати налаштування утиліти для програмування мікроконтролерів STC-ISP.

5. Відкрити в утиліті STC-ISP файл відкомпільованої програми прикладу, під’єднати до комп’ютера навчальний стенд А2, записати програму приклад в пам’ять мікроконтролера та перевірити її роботу.

6. В пакеті PROTEUS DESIGN створити проєкт схеми мікроконтролера із світлодіодами, перевірити її роботу.

7. Зробити звіт з лабораторної роботи.

Контрольні питання:

1. Які мови підтримує середовище MCU 8051 IDE?
2. Які мікроконтролери дозволяє програмувати утиліта для програмування STC-ISP?
3. Як в пакеті PROTEUS DESIGN підключається файл з кодом програми до обраного мікроконтролера?
4. Чи дозволяє пакет PROTEUS DESIGN розробляти друковані плати для електронних схем?
5. Як виконувати налагодження роботи мікроконтролера в середовище MCU 8051 IDE «крок за кроком»?

## Лабораторна робота № 2

## Тема: Архітектура та система команд мікроконтролера

**Мета: Ознайомитись із архітектурою та системою команд мікроконтролерів сімейства MCS-51.**

1. Структурна схема мікроконтролера MCS-51 показана на рис. 2.1. Мікроконтролер містить резидентну пам'ять програм (РПП) та резидентну пам'ять даних (РПД); пристрій управління і синхронізації, до складу якого входить лічильник команд, регістр команд і регістр ознак; арифметико-логічний пристрій, до складу якого входить АЛБ, акумулятор і регістри; блок таймерів-лічильників та блок послідовного інтерфейсу і переривань. Обмін даними здійснюється через чотири порти P0, P1, P2, P3, або через послідовний порт.

Резидентна пам’ять програм, має ємність 4Кб. Призначена для зберігання команд, констант, управляючих слів ініціалізації, таблиць кодування вхідних і вихідних змінних. Резидентна пам’ять даних підключена до шістнадцятибітної шини адреси, що надходить з лічильника команд, або регістру покажчика даних.

Резидентна пам’ять даних призначена для зберігання змінних у процесі виконання програми, адресується одним байтом і має ємність 128 байт. До адресного простору резидентної пам’яті даних належать регістри спеціальних функцій.

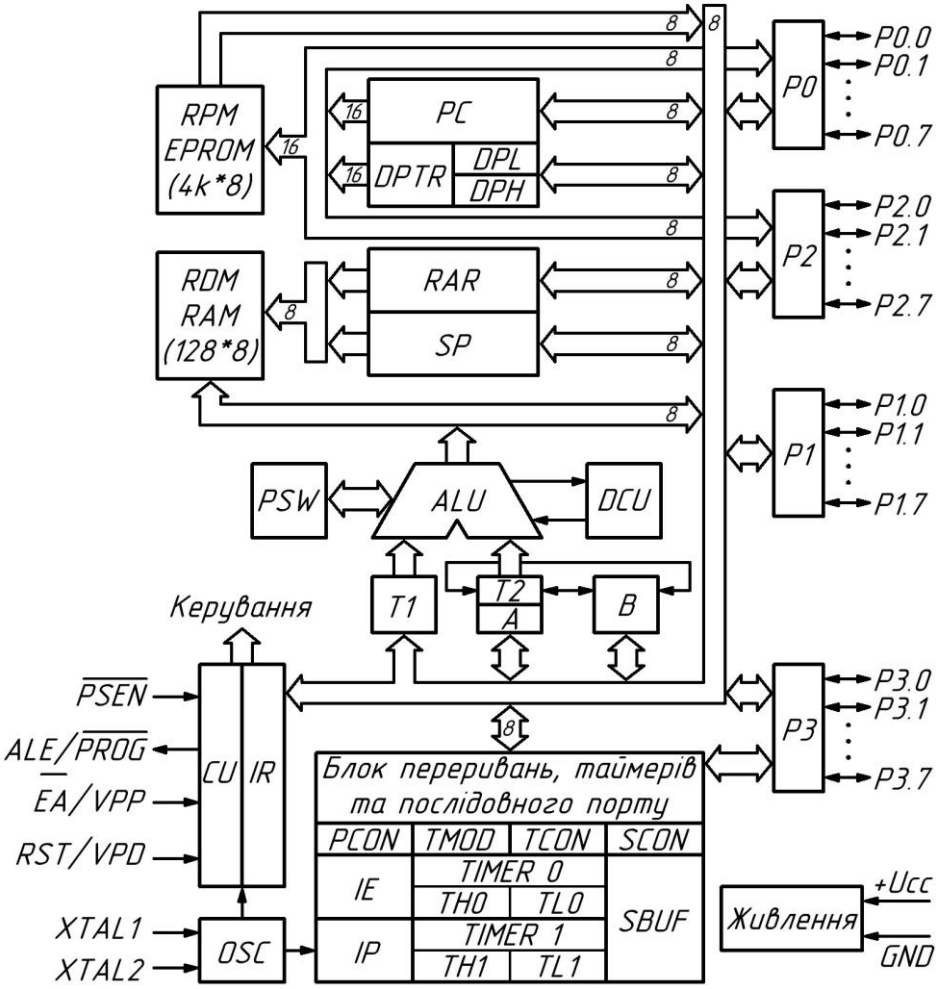



Рисунок 2.1 – Структурна схема MCS-51

Резидентна пам’ять даних призначена для зберігання змінних у процесі виконання програми, адресується одним байтом і має ємність 128 байт. До адресного простору резидентної пам’яті даних належать регістри спеціальних функцій.

Восьмибітний арифметико-логічний пристрій (АЛП) виконує арифметичні операції додавання, віднімання, множення, ділення, логічні операції – «І», «АБО» та «ВИКЛЮЧНЕ АБО», операції циклічного зсуву, скидання, інвертування, інкременту, декременту, тощо.

Важливою особливістю мікроконтролера MCS-51 є можливість оперувати не тільки байтами, але і бітами. Окремі програмно доступні біти можуть бути

встановлені, скинуті, передані, інвертовані, проаналізовані, використані в логічних операціях.

Під час виконання багатьох команд а АЛП формуються ряд ознак, які фіксуються у регістрі слова стану програми, що належить до регістрів спеціальних функції. Ознака переносу С приймає участь і модифікується в процесі виконання великої кількості операції в АЛП, таких як додавання, віднімання, зсув, тощо. Окрім того ознака переносу виконує дії своєрідного акумулятора під час виконання операцій з бітами. Ознака переповнення OV фіксує автоматичне переповнення під час виконання операцій зі знаками, і дає можливість реалізації арифметики в доповнювальних кодах.

Таким чином АЛП може оперувати з чотирма типами інформаційних об'єктів: булевськими (1 біт), цифровими (4 біти), байтовими (8 біт), і адресними (16 біт). В АЛП виконується 51 операція пересилки та перетворювання даних. Застосовуються одинадцять режимів адресації – сім для даних, чотири для адресів.

Одинадцять регістрів спеціальних функцій допускають як байтову, так і побітову адресацію.

Акумулятор ACC – це восьмирозрядний регістр, який використовується як приймач або джерело операнду під час виконання арифметичних і логічних операцій та ряду операцій передачі даних. За застосування акумулятора виконуються операції зсувів, перевірки на нуль, формування ознаки парності.

Регістр В використовується під час виконання операцій множення і ділення. Під час виконання інших операцій може застосовуватись як допоміжний регістр.

Регістр слова стану програми PSW призначений для зберігання інформації про ознаки, що формуються в АЛП під час виконання обчислень.

Восьмибітний покажчик стека SP може адресувати будь-яку область РПД і призначений для зберігання адреси комірки стека, до якого було останнє звернення. В мікроконтролері реалізований передінкрементний/ постдекрементний спосіб адресації стеку. Вміст регістру SP інкрементується перш ніж дані будуть запам'ятовуватись у стеку під час виконання команд PUSH і CALL, та декрементується після виконання команд POP і RET.

В процесі ініціалізації мікроконтролера в регістр SP автоматично завантажується код 07h. Якщо прикладна програма не перевизначає стек, то перший елемент даних у стеку буде розміщуватись у РПД за адресою 08h.

Двобайтний регістр-покажчик даних DPTR застосовується для фіксації

шістнадцятибітної адреси під час виконання операцій звернення до зовнішньої пам’яті. Командами мікроконтролера регістр-покажчик даних може бути застосований і як шістнадцятибітний регістр і як два незалежних восьмибітних регістра.

У склад мікроконтролера входять регістрові пари TH0, TL0, TH1, TL1, на основі яких функціонують два незалежних шістнадцятибітних таймера/лічильника подій.

Буфер приймача / передавача SBUF складається з двох незалежних регістрів – буферу приймача і буферу передатчика. Завантаження байту в регістр викликає початок процесу передачі через послідовний порт. Коли байт зчитується із регістру, його джерелом є приймач послідовного порту.

Пристрій управління та синхронізації з кварцовим резонатором, що підключається до зовнішніх виводів XTAL1 та XTAL2 мікросхеми управляє роботою внутрішнього генератора, який в свою чергу формує сигнали синхронізації.

Пристрій управління на основі сигналів синхронізації формує машинний цикл фіксованої довжини, що дорівнює дванадцяти періодам резонатора/

2. Кількість команд мікроконтролера – сто одинадцять. Команди відносно функціональних ознак класифікуються за наступними групами:

– команди передачі даних;

– команди виконання арифметичних операцій;

– команди виконання логічних операцій;

– команди виконання операцій з бітами;

– команди передачі управління.

Команди мають довжину один, два або три байти і виконуються відповідно за один, два або чотири машинні цикли. Якщо частоти генератора 12 МГц, то тривалість циклу складає 1 мкс.

При розгляді команд використовуються наступні позначення:

Rn (n = 0, 1, …, 7) – регістр загального призначення в обраному банку регістрів;

Ri (i = 0, 1) – регістр загального призначення в обраному банку регістрів, що використовується як регістр посередньої адреси;

ad – пряма адреса байту;

ads – пряма адреса байту-джерела;

add – пряма адреса байту-одержувача;

ad11 – 11-розрядна абсолютна адреса переходу;

ad16 – 16-розрядна абсолютна адреса переходу;

rel – відносна адреса переходу;

#d – безпосереднє чисельне значення операнду;

#d16 – безпосереднє значення двобайтного операнду;

bit – пряма адреса біта;

/bit – пряма адреса біта, який інвертується;

A – акумулятор;

PC – лічильник команд;

DPTR – регістр-покажчик даних;

( ) – вміст елемента пам’яті чи регістра.

Група команд пересилання даних містить 28 команд, короткі відомості про які наведені в табл. 2.1, де зазначені також тип команди (Т), її довжина в байтах (Б) та час виконання в машинних циклах (Ц).

Таблиця 2.1

| Мнемокод | | КОП | Т | Б | Ц | Коментар |
|---|---|---|---|---|---|---|
| MOV | A,Rn | 11101rrr | 1 | 1 | 1 | Пересилання в акумулятор з регістра (n=0÷7) (A)←(Rn). |
| MOV | A,ad | 11100101 | 3 | 2 | 1 | Пересилання в акумулятор байта з адресою ad (A)←(ad). |
| MOV | A,@Ri | 1110011i | 1 | 1 | 1 | Пересилання в акумулятор байта з РПД (i=0,1) (A)←((Ri)). |
| MOV | A,#d | 01110100 | 2 | 2 | 1 | Завантаження в акумулятор константи (A)←#d. |
| MOV | Rn,A | 11111rrr | 1 | 1 | 1 | Пересилання в регістр з акумулятора (Rn)←(A). |
| MOV | Rn,ad | 10101rrr | 3 | 2 | 2 | Пересилання в регістр байта з адресою ad (Rn)←(ad). |
| MOV | Rn,#d | 01111rrr | 2 | 2 | 1 | Завантаження в регістр константи (Rn)←#d. |
| MOV | ad,A | 11110101 | 3 | 2 | 1 | Пересилання за прямою адресою акумулятора (ad)←(A). |
| MOV | ad,Rn | 10001rrr | 3 | 2 | 2 | Пересилання за прямою адресою регістру (ad)←(Rn). |
| MOV | add,ads | 10000101 | 9 | 3 | 2 | Пересилання о байта з адресою ads за прямою адресою add (add)←(ads). |
| MOV | ad,@Ri | 1000011i | 3 | 2 | 2 | Пересилання байта з РПД за прямою адресою (ad)←((Ri)). |
| MOV | ad,#d | 01110101 | 7 | 3 | 2 | Пересилання за прямою адресою константи (ad)←#d. |
| MOV | @Ri,A | 1111011i | 1 | 1 | 1 | Пересилання в РПД байту з акумулятора ((Ri))←(A). |
| MOV | @Ri,ad | 1010011i | 3 | 2 | 2 | Пересилання в РПД байту з адресою ad ((Ri))←(ad). |
| MOV | @Ri,#d | 0111011i | 2 | 2 | 1 | Пересилання в РПД константи ((Ri))←#d. |
| MOV | DPTR,#d16 | 10010000 | 13 | 3 | 2 | Завантаження покажчика даних (DPTR)←#d16. |
| MOVC | A,@A+DPTR | 10010011 | 1 | 1 | 2 | Пересилання в акумулятор байта з ПП (A)←((A)+(DPTR)). |
| MOVC | A,@A+PC | 10000011 | 1 | 1 | 2 | Пересилання в акумулятор байта з ПП (PC)←(PC)+1; (A)←((A)+(PC)). |
| MOVX | A,@Ri | 1110001i | 1 | 1 | 2 | Пересилання в акумулятор байта з ЗПД (A)←((Ri)). |
| MOVX | A,@DPTR | 11100000 | 1 | 1 | 2 | Пересилання в акумулятор байта з розширеного ЗПД (A)←((DPTR)). |
| MOVX | @Ri,A | 1111001i | 1 | 1 | 2 | Пересилання у ЗПД з акумулятора ((Ri))←(A). |
| MOVX | @DPTR,A | 11110000 | 1 | 1 | 2 | Пересилання до розширеного ЗПД з акумулятора ((DPTR))←(A). |
| PUSH | ad | 11000000 | 3 | 2 | 2 | Завантаження у стек (SP)←(SP)+1; ((SP))←(ad). |
| POP | ad | 11010000 | 3 | 2 | 2 | Витяг із стеку (ad)←((SP)); (SP)←(SP)-1. |
| XCH | A,Rn | 11001rrr | 1 | 1 | 1 | Обмін акумулятора з регістром (A)↔(Rn). |
| XCH | A,ad | 11000101 | 3 | 2 | 1 | Обмін акумулятора з байтом з адресою ad (A)↔(ad). |

| | | | | | | |
|---|---|---|---|---|---|---|
| XCH | A,@Ri | 1100011i | 1 | 1 | 1 | Обмін акумулятора з байтом із РПД (A)↔((Ri)). |
| XCHD | A,@Ri | 1101011i | 1 | 1 | 1 | Обмін молодших тетрад акумулятора та байту РПД ($A_{3-0}$)↔(($Ri_{3-0}$)). |

До групи команд арифметичних операцій належить 24 команди (табл. 2.2). Мікроконтролер виконує досить широкий набір команд для організації обробки цілочисельних даних, у тому числі множення та ділення. За результатом виконання команд ADD, ADDC, SUBB, MUL і DIV модифікуються прапорці (ознаки) байту стану програми PSW.

Таблиця 2.2

| **Мнемокод** | | **КОП** | **Т** | **Б** | **Ц** | **Коментар** |
|---|---|---|---|---|---|---|
| ADD | A,Rn | 00101rrr | 1 | 1 | 1 | Додавання акумулятора з регістром (n=0÷7) (A)←(A)+(Rn). |
| ADD | A,ad | 00100101 | 3 | 2 | 1 | Додавання акумулятора з прямоадресованим байтом (A)←(A)+(ad). |
| ADD | A,@Ri | 0010011i | 1 | 1 | 1 | Додавання акумулятора з байтом з РПД (i = 0,1) (A)←(A)+((Ri)). |
| ADD | A,#d | 00100100 | 2 | 2 | 1 | Додавання акумулятора з константою (A)←(A)+#d. |
| ADDC | A,Rn | 00111rrr | 1 | 1 | 1 | Додавання акумулятора з регістром та переносом (A)←(A)+(Rn)+(C). |
| ADDC | A,ad | 00110101 | 3 | 2 | 1 | Додавання акумулятора з байтом з адресою ad та переносом (A)←(A)+(ad)+(C). |
| ADDC | A,@Ri | 0011011i | 1 | 1 | 1 | Додавання акумулятора з байтом з РПД та переносом (A)←(A)+((Ri))+(C). |
| ADDC | A,#d | 00110100 | 2 | 2 | 1 | Додавання акумулятора з константою та переносом (A)←(A)+#d+(C). |
| DA | A | 11010100 | 1 | 1 | 1 | Десяткова корекція акумулятора. |
| SUBB | A,Rn | 10011rrr | 1 | 1 | 1 | Віднімання з акумулятора регістру та позики (A)←(A)-(Rn)-(C). |
| SUBB | A,ad | 10010101 | 3 | 2 | 1 | Віднімання з акумулятора байта з адресою ad та позики (A)←(A)-(ad)-(C). |
| SUBB | A,@Ri | 1001011i | 1 | 1 | 1 | Віднімання з акумулятора байта РПД та позики (A)←(A)-((Ri))-(C). |
| SUBB | A,#d | 10010100 | 2 | 2 | 1 | Віднімання з акумулятора константи та позики (A)←(A)-#d-(C). |
| INC | A | 00000100 | 1 | 1 | 1 | Інкремент акумулятора (A)←(A)+1. |
| INC | Rn | 00001rrr | 1 | 1 | 1 | Інкремент регістру (Rn)←(Rn)+1. |
| INC | ad | 00000101 | 3 | 2 | 1 | Інкремент байту з адресою ad (ad)←(ad)+1. |
| INC | @Ri | 0000011i | 1 | 1 | 1 | Інкремент байту в РПД ((Ri))←((Ri))+1. |
| INC | DPTR | 10100011 | 1 | 1 | 2 | Інкремент покажчика даних (DPTR)←(DPTR)+1. |
| DEC | A | 00010100 | 1 | 1 | 1 | Декремент акумулятора (A)←(A)-1. |
| DEC | Rn | 00011rrr | 1 | 1 | 1 | Декремент регістру (Rn)←(Rn)-1. |
| DEC | ad | 00010101 | 3 | 2 | 1 | Декремент байту з адресою ad (ad)←(ad)-1. |
| DEC | @Ri | 0001011i | 1 | 1 | 1 | Декремент байту в РПД ((Ri))←((Ri))-1. |
| MUL | AB | 10100100 | 1 | 1 | 4 | Помноження акумулятора на регістр (B)(A)←(A)·(B). |
| DIV | AB | 10000100 | 1 | 1 | 4 | Ділення акумулятора на регістр (B).(A)←(A)/(B). |

Структура байту стану програми PSW має наступний вигляд:

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| C | AC | F0 | RS1 | RS0 | OV | – | P |

C (PSW.7) – ознака переносу, встановлюється і скидається апаратно або

програмно при виконанні арифметичних та логічних операцій, якщо в старшому розряді результату виникає перенос або позика.

AC (PSW.6) – ознака додаткового переносу встановлюється і скидається апаратно при виконанні додавання і віднімання, якщо в третьому розряді результату D3 виникає перенос або позика, також служить для реалізації десяткової арифметики (ознака використовується командою десяткової корекції DAA).

F0 (PSW.5) – ознака нульового результату операції, встановлюється і скидається програмно, специфікується користувачем.

RS1 (PSW.4), RS2 (PSW.3) – вибір банку регістрів, встановлюється і скидається програмно для вибору банку регістрів.

OV (PSW.2) – ознака переповнювання, встановлюється і скидається апаратно, встановлюється при перенесенні із розряду D6, тобто тоді, коли результат арифметичної дії не вміщується в 7 розрядів і старший (восьмий) біт акумулятора не може бути інтерпретований як знаковий, використовується для роботи з числами зі знаком.

PSW.1 – не використовується.

P (PSW.0) – ознака парності, встановлюється і скидається апаратно в кожному циклі команди і фіксує парну/непарну кількість одиниць в акумуляторі, встановлюється в стан 1, якщо кількість одиничних біт в акумуляторі парна, а коли непарна – скидається в стан 0.

Група команд логічних операцій містить 25 команд (табл. 2.3), які дають змогу виконувати операції над байтами: логічне множення (AND, $\wedge$ ), логічне додавання (OR, $\vee$ ), додавання за модулем 2 (XOR, $\oplus$), інверсію (NOT), скидання в нульовий стан і зсув.

Таблиця 2.3

| Мнемокод | | КОП | Т | Б | Ц | Коментар |
|---|---|---|---|---|---|---|
| ANL | A,Rn | 01011rrr | 1 | 1 | 1 | Логічне «І» акумулятора та регістра (A)←(A)∧(Rn). |
| ANL | A,ad | 01010101 | 3 | 2 | 1 | Логічне «І» акумулятора та байту з адресою ad (A)←(A)∧(ad). |
| ANL | A,@Ri | 0101011i | 1 | 1 | 1 | Логічне «І» акумулятора та байту з РПД (A)←(A)∧((Ri)). |
| ANL | A,#d | 01010100 | 2 | 2 | 1 | Логічне «І» акумулятора та константи (A)←(A)∧#d. |
| ANL | ad,A | 01010010 | 3 | 2 | 1 | Логічне «І» байту з адресою ad та акумулятора (ad)←(ad)∧(A). |
| ANL | ad,#d | 01010011 | 7 | 3 | 2 | Логічне «І» байту з адресою ad та константи (ad)←(ad)∧#d. |
| ORL | A,Rn | 01001rrr | 1 | 1 | 1 | Логічне «АБО» акумулятора та регістра (A)←(A)∨(Rn). |
| ORL | A,ad | 01000101 | 3 | 2 | 1 | Логічне «АБО» акумулятора та байту з адресою ad (A)←(A)∨(ad). |
| ORL | A,@Ri | 0100011i | 1 | 1 | 1 | Логічне «АБО» акумулятора та байту з РПД (A)←(A)∨((Ri)). |

| | | | | | | |
|---|---|---|---|---|---|---|
| ORL | A,#d | 01000100 | 2 | 2 | 1 | Логічне «АБО» акумулятора та константи (A)←(A)∨#d. |
| ORL | ad,A | 01000010 | 3 | 2 | 1 | Логічне «АБО» байту з адресою ad та акумулятора (ad)←(ad)∨(A). |
| ORL | ad,#d | 01000011 | 7 | 3 | 2 | Логічне «АБО» байту з адресою ad та константи (ad)←(ad)∨#d. |
| XRL | A,Rn | 01101rrr | 1 | 1 | 1 | Виключне «АБО» акумулятора та регістра (A)←(A)⊕(Rn). |
| XRL | A,ad | 01100101 | 3 | 2 | 1 | Виключне «АБО» акумулятора та байту з адресою ad (A)←(A)⊕(ad). |
| XRL | A,@Ri | 0110011i | 1 | 1 | 1 | Виключне «АБО» акумулятора та байту з РПД (A)←(A)⊕((Ri)). |
| XRL | A,#d | 01100100 | 2 | 2 | 1 | Виключне «АБО» акумулятора та константи (A)←(A)⊕#d. |
| XRL | ad,A | 01100010 | 3 | 2 | 1 | Виключне «АБО» байту з адресою ad та акумулятора (ad)←(ad)⊕(A). |
| XRL | ad,#d | 01100011 | 7 | 3 | 2 | Виключає «АБО» байту з адресою ad та константи (ad)←(ad)⊕#d. |
| CLR | A | 11100100 | 1 | 1 | 1 | Скидання акумулятора (A)←0. |
| CPL | A | 11110100 | 1 | 1 | 1 | Інверсія акумулятора (A)←NOT(A). |
| SWAP | A | 11000100 | 1 | 1 | 1 | Обмін місцями тетрад в акумуляторі ($A_{7-4}$)↔($A_{3-0}$). |
| RL | A | 00100011 | 1 | 1 | 1 | Зсув ліворуч. |
| RLC | A | 00110011 | 1 | 1 | 1 | Зсув ліворуч через перенос. |
| RR | A | 00000011 | 1 | 1 | 1 | Зсув праворуч. |
| RRC | A | 00010011 | 1 | 1 | 1 | Зсув праворуч через перенос. |

Група команд операцій над бітами містить 12 команд (табл. 2.4). Команди операцій над бітами дозволяють встановлювати в одиницю, скидати в нуль, інвертувати окремі біти, а також виконувати логічне множення (AND, ∧ ) та додавання (OR, ∨ ) біт тощо.

Таблиця 2.4

| Мнемокод | | КОП | Т | Б | Ц | Коментар |
|---|---|---|---|---|---|---|
| CLR | C | 11000011 | 1 | 1 | 1 | Скидання переносу (C)←0. |
| CLR | bit | 11000010 | 4 | 2 | 1 | Скидання біта (bit)←0. |
| SETB | C | 11010011 | 1 | 1 | 1 | Встановлення переносу (C)←1. |
| SETB | bit | 11010010 | 4 | 2 | 1 | Встановлення біта (bit)←1. |
| CPL | C | 10110011 | 1 | 1 | 1 | Інверсія переносу (C)←NOT(C). |
| CPL | bit | 10110010 | 4 | 2 | 1 | Інверсія біта (bit)←NOT(bit). |
| ANL | C,bit | 10000010 | 4 | 2 | 2 | Логічне «І» біта та переносу (C)←(C)∧(bit). |
| ANL | C,/bit | 10110000 | 4 | 2 | 2 | Логічне «І» інверсії біта та перенесення (C)←(C)∧NOT(bit). |
| ORL | C,bit | 01110010 | 4 | 2 | 2 | Логічне «АБО» біта та переносу(C)←(C)∨(bit). |
| ORL | C,/bit | 10100000 | 4 | 2 | 2 | Логічне «АБО» інверсії біта та переносу (C)←(C)∨NOT(bit). |
| MOV | C,bit | 10100010 | 4 | 2 | 2 | Пересилання біта у перенос (C)←(bit). |
| MOV | bit,C | 10010010 | 4 | 2 | 2 | Пересилання переносу в біт (bit)←(C). |

При виконанні операцій над бітами як «логічний акумулятор» виступає прапорець ознаки перенесення C (розряд D7 байту стану програми PSW). Він бере участь в усіх операціях над двома бітами. Ознака перенесення C є бітом-джерелом операнду і бітом-одержувачем результату. У ролі операндів можуть

використовуватися 128 біт із резидентної (внутрішньої) пам’яті даних і біти деяких регістрів спеціальних функцій.

Група команди передачі керування містить команди безумовного і умовного переходів, виклику підпрограм та повернення із підпрограм (табл. 2.5).

Таблиця 2.5

| Мнемокод | | КОП | Т | Б | Ц | Коментар |
|---|---|---|---|---|---|---|
| LJMP | adl6 | 00000010 | 12 | 3 | 2 | Довгий безумовний перехід на будь-яку адресу пам’яті. |
| AJMP | adll | $a_{10}a_9a_8$00001 | 6 | 2 | 2 | Абсолютний перехід у межах сторінки 2 Кб. |
| SJMP | rel | 10000000 | 5 | 2 | 2 | Короткий перехід у межах сторінки 256 байт. |
| JMP | @A+DPTR | 01110011 | 1 | 1 | 2 | Безумовний перехід за непрямою адресою. |
| JZ | rel | 01100000 | 5 | 2 | 2 | Перехід, якщо акумулятор дорівнює нулю. |
| JNZ | rel | 01110000 | 5 | 2 | 2 | Перехід, якщо акумулятор не дорівнює нулю. |
| JC | rel | 01000000 | 5 | 2 | 2 | Перехід, якщо (C)=1. |
| JNC | rel | 01010000 | 5 | 2 | 2 | Перехід, якщо (C)=0. |
| JB | bit,rel | 00100000 | 11 | 3 | 2 | Перехід, якщо (bit)=1. |
| JNB | bit,rel | 00110000 | 11 | 3 | 2 | Перехід, якщо (bit)=0. |
| JBC | bit,rel | 00010000 | 11 | 3 | 2 | Перехід, якщо (bit)=1 із скиданням біта (bit)←0. |
| DJNZ | Rn,rel | 11011rrr | 5 | 2 | 2 | Декремент регістру та перехід, якщо не нуль. |
| DJNZ | ad,rel | 11010101 | 8 | 3 | 2 | Декремент байту з адресою ad та перехід, якщо не нуль. |
| CJNE | A,ad,rel | 10110101 | 8 | 3 | 2 | Порівняння акумулятора з байтом з адресою ad та перехід, якщо не дорівнює. |
| CJNE | A,#d,rel | 10110100 | 10 | 3 | 2 | Порівняння акумулятора з константою та перехід, якщо не дорівнює. |
| CJNE | Rn,#d,rel | 10111rrr | 10 | 3 | 2 | Порівняння регістру з константою та перехід, якщо не дорівнює. |
| CJNE | @Ri,#d,rel | 1011011i | 10 | 3 | 2 | Порівняння байта в РПД з константою та перехід, якщо не дорівнює. |
| LCALL | ad16 | 00010010 | 12 | 3 | 2 | Довгий виклик під-програми із всього ПЗП. |
| ACALL | ad11 | $a_{10}a_9a_8$10001 | 6 | 2 | 2 | Абсолютний виклик підпрограми у межах сторінки 2 Кб. |
| RET | | 00100010 | 1 | 1 | 2 | Повернення із підпрограми. |
| RETI | | 00110010 | 1 | 1 | 2 | Повернення із підпрограми обробки переривання. |
| NOP | | 00000000 | 1 | 1 | 1 | Пуста операція. |

Для можливості управління процесом трансляції програми використовуються директиви мови Асемблера.

Директива ORG задає асемблеру адресу комірки пам'яті, в якій повинна бути розташована наступна команда прикладної програми.

```
ORG 0000h
```

Директива EQU дає можливість будь-якому символічному імені, що використовується в програмі, поставити у відповідність певний операнд.

```
PORT_X EQU P0
```

Директива SET визначає символічні імена операндів, що перевизначаються в процесі виконання програми.

```
X SET 3
```

```
X SET X + 1
```

Директива BIT дозволяє визначити символічне ім'я як адресу біта.

```
FLAG BIT 25h.3
```

Директива DB забезпечує занесення в пам'ять програм однобайтні константи.

```
DB 75h, 80h
DB ‘PORT’
DB ‘P’
```

Директива DW дозволяє заносити у пам'ять програм числа довжиною два байти.

```
DW 03FAh, 5C20h
DW ‘XY’
```

Директива END дає асемблеру вказівку про закінчення трансляції.

**Приклад 1.**

Записати в резидентну пам’ять даних за адресами 30h та 31h число довжиною два байти 235Ah.

```
MOV R0,#30h          ;Завантаження в регістр R0 покажчика даних.
MOV @R0,#23h         ;Завантаження в комірку РПД з адресою 30h  числа 23h.
INC R0               ;Збільшення на 1 покажчика даних.
MOV @R0,#5Ah         ;Завантаження в комірку РПД з адресою 31h числа 5Ah.
END
```

**Приклад 2.**

Додати два числа довжиною один байт кожне, які розташовані в резидентній пам’яті даних (РПД) за адресою 40h та в регістрі R5 банку пам’яті 1. Результати записати в комірку РПД з адресою 50h.

```
SETB RS0             ;Вибір банка пам’яті 1.
MOV A,40h            ;Запис в акумулятор вмісту комірки РПД з адресою 40h.
ADD A, R5            ;Додавання до А вмісту регістра R5 банку пам’яті 1.
MOV 50h,A            ;Запис в комірку РПД з адресою 50h вмісту акумулятора.
END
```

**Приклад 3.**

Помножити ціле число довжиною один байт, яке розташоване в комірці РПД за адресою 40h на константу 55h. Результати записати в комірки РПД з адресами 41h старший байт та 42h молодший байт.

```
MOV R0,#40h          ;Запис в регістр R0 числа 40h.
MOV A,#00h           ;Запис в акумулятор числа 00h.
MOV DPTR,#CONS       ;Запис в DPTR адреси таблиці констант.
MOVC A, @A+DPTR      ;Запис в акумулятор А константи із РПП.
MOV B, @R0           ;Запис в В вмісту комірки РПД з адресою, яка в R0.
MUL AB               ;Множення вмісту акумулятора А на вміст регістра В.
```

```
        INC R0              ;Збільшення на 1 вмісту регістра R0.
        MOV @R0,B           ;Запис в комірку РПД, адреса якої знаходиться в
                            ;регістрі R0, вмісту регістру B.
        INC R0              ;Збільшення на 1 вмісту регістра R0.
        MOV @R0,A           ;Запис в комірку РПД, адреса якої в R0, вмісту A.
CONS:                       ;Розташування таблиці констант в РПП.
        DB 55h              ;Запис в РПП константи 55h.
END
```

**Приклад 4.**

Виконати логічне «І» між вмістом комірки РПД з адресою 20h та регістра R7 банку пам'яті 3. Результат записати в регістр R7 банку пам'яті 0.

```
        MOV A,20h           ;Завантаження в A вмісту комірки РПД з адресою 40h.
        SETB RS0            ;Вибір банку пам'яті 3.
        SETB RS1            ;
        ANL A, R7           ;Порозрядне логічне «І» вмісту A та вмісту регістра R7.
        MOV 06h,A           ;Завантаження в комірку РПД вмісту акумулятора (регістр
                            ;R7 банку пам'яті 0 має адресу 06h).
END
```

Хід роботи:

1. Ознайомитись із архітектурою мікроконтролера сімейства MCS-51.
2. В інтегрованому середовищі MCU 8051 IDE та на навчальному стенді А2 ознайомитись із роботою команд різних груп та прикладів.
3. Для завдання, яке ставить викладач, розробити алгоритми та скласти програми.
4. Промоделювати роботу програм в середовищі MCU 8051 IDE та на навчальному стенді А2.
5. Зробити звіт з лабораторної роботи.

Контрольні питання:

1. Які умови встановлення прапорців перенесення (C), допоміжного перенесення (AC), переповнення (OV) і паритету (P)?
2. Які функції виконують регістри R0 і R1?
3. Як вибирається необхідний банк регістрів загального призначення?
4. Яка різниця в діях команд RET та RETI?
5. Чи можливий перехід у будь-яку точку програмного простору за командами умовного переходу?

## Лабораторна робота № 3

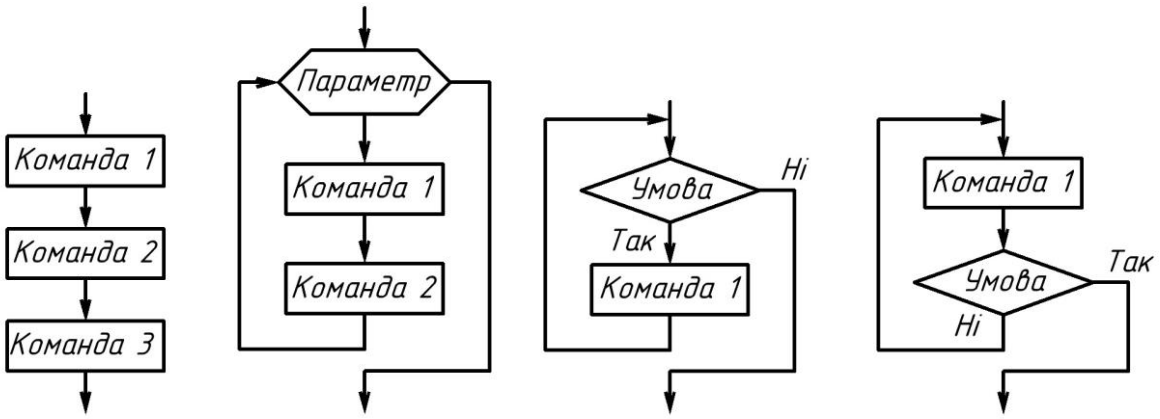
## Тема: Умовні алгоритми роботи мікроконтролера

**Мета: Ознайомитись із способами реалізації умовних алгоритмів роботи мікроконтролерів та отримати навички в написанні програм.**

Алгоритм – скінченний набір інструкцій, які описують послідовність дій (команд), виконання яких приводить до розв'язання поставленого задачі. Для візуалізації алгоритмів використовують блок-схеми.

Перед розробкою програми на основі даних завдання розробляється блок-схема алгоритму, кількість блоків якого під час розробки поступово збільшується, а самі блоки спрощуються до виконання простих дій.

Кожен алгоритм складається з елементарних кроків, які поєднуються у певні алгоритмічні конструкції: лінійну (послідовну), розгалужену, циклічну (рис. 3.1, 3.2).




а) б) в) г)

Рисунок 3.1 – Типи алгоритмів: а) лінійний; б) циклічний із параметром; в) циклічний із передумовою; г) циклічний із післяумовою

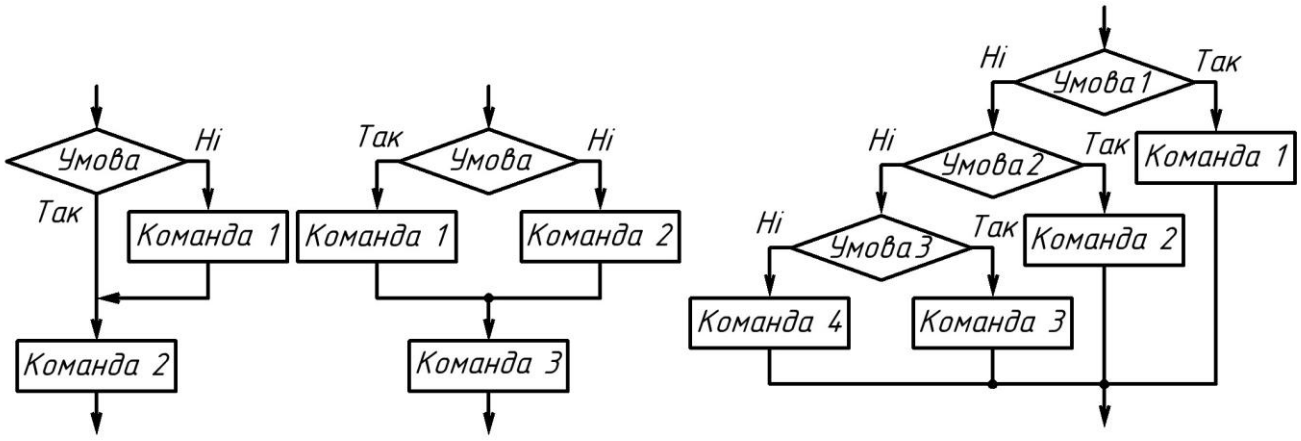
Лінійною називається конструкція алгоритму (рис. 3.1, а), реалізована у вигляді послідовності кроків (команд, дій), причому кожен крок виконується лише один раз поки алгоритм не досягне свого кінцевого блоку.

За допомогою лінійних алгоритмів описуються лінійні процеси. Такого типу алгоритми використовують при описі узагальненого розв'язання завдань як послідовностей модулів.

Розгалуженою називають таку алгоритмічну конструкцію (рис. 3.2), яка забезпечує вибір між двома варіантами рішень в залежності від значень вхідних даних.

Розгалуження бувають двох типів: неповне («якщо умова – то дія 1») і повне («якщо умова – то дія 1 – інакше дія 2»). За допомогою повного

розгалуження (рис. 3.2, б) можна організувати дві гілки в алгоритмі (те чи інше), кожна з яких призведе до загальної точки їх злиття, алгоритм виконуватиметься незалежно від того, яким шляхом пішло рішення. За наявності неповного розгалуження (рис. 3.2, а) передбачаються деякі дії алгоритму лише на одній гілці («то»), оскільки друга відсутня, для одного з результатів перевірки дії робити немає необхідності, управління відразу перейде до точки злиття. Дуже часто доводиться вибирати шлях вирішення задачі не з двох, а з кількох можливих. У програмуванні цей процес можна реалізувати, використовуючи кілька умовних операторів (рис. 3.2, в).




а) б) в)

Рисунок 3.2 – Типи розгалужених алгоритмів: а) розгалужений (неповне розгалуження); б) розгалужений (повне розгалуження); в) розгалужений з вибором

Циклічною (або циклом) називається конструкція алгоритму, в якій деяка група кроків (команд, дій), що йдуть підряд, виконується кілька разів залежно від умови завдання і вхідних даних. Таку групу повторюваних дій у кожному кроці циклу називають тілом циклу.

У будь-якій циклічній конструкції містяться елементи розгалуженої конструкції алгоритму.

Розрізняють три типи циклічних алгоритмів:

1) цикл із параметром (арифметичний цикл) (рис. 3.1, б);

2) цикл із передумовою (рис. 3.1, в);

3) цикл із післяумовою (рис. 3.1, г).

Цикли із передумовою та післяумовою також називають ітераційними.

У арифметичному циклі число кроків однозначно визначено правилом зміни параметра, що задається за допомогою початкових та кінцевих значень, а

також кроку його зміни. Тобто, на кожному кроці циклу значення параметра змінюється згідно з кроком циклу, поки не досягне значення, що дорівнює кінцевому значенню параметра.

У циклі із передумовою кількість кроків заздалегідь не визначена, вона залежить від вхідних даних. У цій циклічній структурі спочатку відбувається перевірка значення умови, що стоїть перед виконанням чергового етапу циклу. При істинному значенні умовного виразу виконуватиметься тіло циклу. Після чого знову виконуватиметься перевірка умови. Ці дії повторюватимуться до того часу, поки значення умови стане помилковим, тоді цикл завершиться.

Особливістю цього типу циклу є те, що при початковій помилковості значення умови тіло циклу не буде виконуватися зовсім.

У циклі із післяумовою, як і в попередньої, заздалегідь не визначене число повторень тіла циклу, воно залежить від вхідних параметрів. Відмінною рисою циклу з передумовою від циклу з післяумовою є те, що тіло у будь-якому випадку буде виконано хоча б один раз і лише після цього перевіриться умова. У цій конструкції (рис. 3.1, г) тіло циклу виконується до того часу, поки значення умовного висловлювання буде помилковим. Як тільки воно стане дійсним, виконання команд припиниться.

Для побудови програм на основі умовних алгоритмів використовується група команд команди передачі керування, які дозволяють виконувати вибір гілки алгоритму на основі ознака виконаної арифметичної або логічної операції.

Алгоритми вирішення прикладів 1-3 наведені на рис. 3.3.

**Приклад 1.**

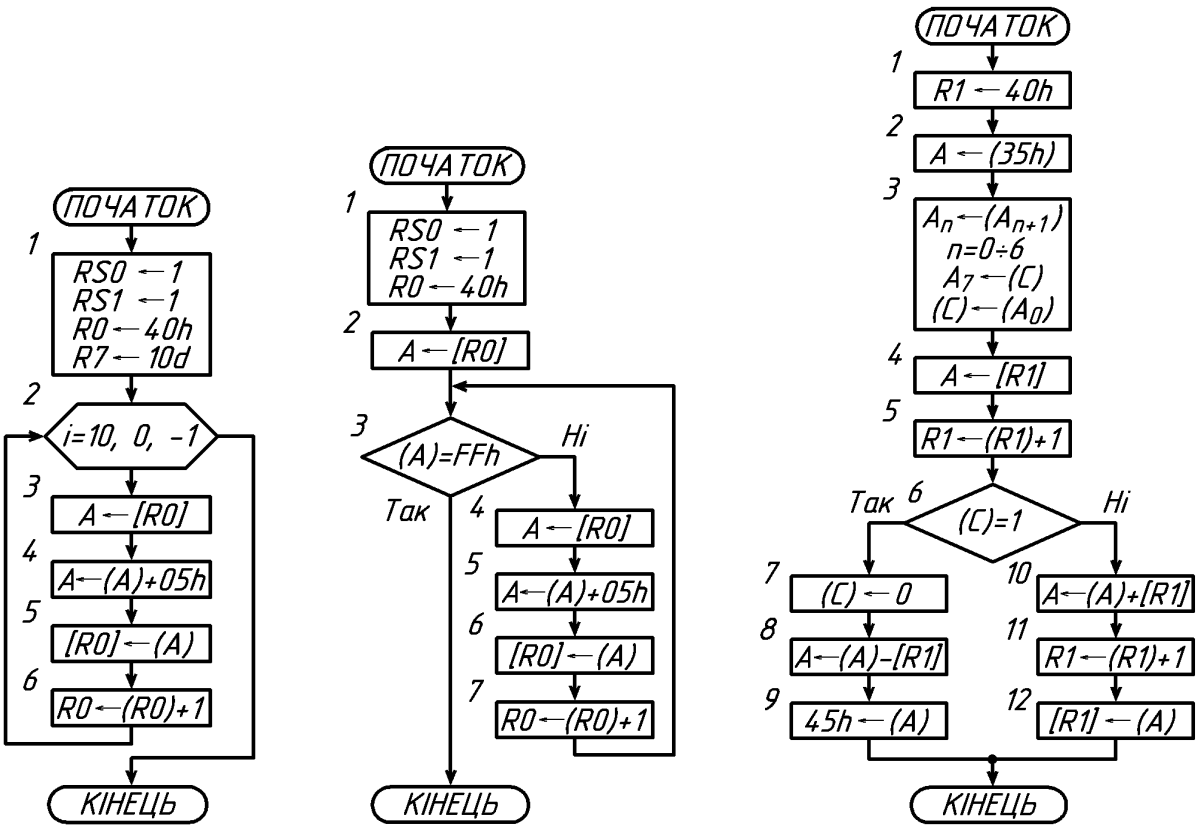
В РПД в комірках пам’яті починаючи за адреси 40h знаходиться масив із 10 однобайтових цілих чисел без знаку. Потрібно до всіх елементів масиву додати число 5h.

Для такої завдання кількість елементів визначена, тому можна використати алгоритм арифметичного циклу (рис. 3.1, б), для чого потрібно організувати лічильник, для зберігання параметру якого використаємо, наприклад, регістр R7 банку пам’яті 3. Для адресації комірок пам’яті зручно використовувати регістр R0.

Команда DJNZ дозволяє складати програму на основі алгоритму арифметичного циклу. При її використанні параметр повинен бути записаний до регістру R0-R7 або до комірки пам’яті з певною адресою.

```
SETB RS0                ;Вибір банку пам’яті через біти RS0 та RS1.
SETB RS1
MOV R0,#40h             ;Запис адреси 40h до регістру R0 обраного банку пам’яті.
```

```
        MOV R7,#10d         ;Запис кількості елементів масиву до регистру R0 обраного
                            ;банку пам’яті.
L1:
        MOV A,@R0           ;Запис до А вмісту комірки РПД, адреса якої в R0.
        ADD A,#05h          ;Додавання числа 5 до вмісту акумулятора А.
        MOV @R0,A           ;Запис вмісту А до комірки РПД, адреса якої в R0.
        INC R0              ;Збільшення на 1 вмісту регістру покажчика адреси R0.
        DJNZ R7,L1          ;Перевірка стану лічильника на регістрі R7, спочатку
                            ;зменшення вмісту регістру R7, а потім перевірка вмісту на
                            ;рівність 0, та перехід на мітку L1, якщо не 0.
        NOP
END
```




а) б) в)

Рисунок 3.3 – Алгоритми розв'язку прикладів: а) алгоритм прикладу 1; б) алгоритм прикладу 2; в) алгоритм прикладу 3

**Приклад 2.**

В РПД в комірках пам’яті починаючи за адреси 40h знаходиться масив із однобайтових цілих чисел без знаку, кінець масиву позначено міткою – числом FFh. Потрібно до всіх елементів масиву додати число 5h.

На відміну від попереднього прикладу кількість елементів в завданні не визначена, можна використати алгоритм із передумовою тому, що потрібно

перевіряти вміст комірок на мітку кінцю масиву FFh та якщо не кінець, тоді додавати до елементу число 5h. Лічильник у даному випадку не потрібен. Для адресації комірок пам’яті зручно використовувати адресацію через регістр R0, залишимо банк пам’яті 3.

За допомогою команди CJNE можна порівняти вміст акумулятору або комірки РПД із константою або вмістом комірки РПД та на основі цього зробити вибір потрібної дії.

```
        SETB RS0            ;Вибір банку пам’яті через біти RS0 та RS1.
        SETB RS1
        MOV R0,#40h         ;Запис адреси 40h до регістру R0 обраного банку пам’яті.
L1:
        MOV A,@R0           ;Запис вмісту комірки РПД з адресою 40h до A.
        CJNE A,#0FFh,L2     ;Перевірка вмісту комірки РПД, адреса якої в R0, якщо не
                            ;дорівнює FFh, то це кінець масиву, перехід на мітку L2.
        SJMP L3             ;Безумовний перехід на мітки L3, вихід із програми.
L2:
        MOV A,@R0           ;Запис до A вмісту комірки РПД, адреса якої в R0.
        ADD A,#05h          ;Додавання числа 5 до вмісту акумулятора A.
        MOV @R0,A           ;Запис вмісту A до комірки РПД, адреса якої в R0.
        INC R0              ;Збільшення на 1 вмісту регістру покажчика адреси R0.
        SJMP L1             ;Безумовний перехід на мітки L1.
L3:
        NOP
END
```

**Приклад 3.**

В комірці РПД з адресою 35h знаходиться число, якщо це число парне потрібно до числа, яке знаходиться в комірці із адресою 40h додати число, яке знаходиться в комірці із адресою 41h, результат записати в комірку РПД із адресою 42h. А якщо непарне, то видалити із вмісту комірки РПД із адресою 40h вміст комірки пам’яті із адресою 41h, а результати записати в комірку пам’яті із адресою 45h.

Так як на основі парності числа в комірці 35h необхідно обирати ти чи інші дії, то для вирішення цього завдання можна використати алгоритм із повним розгалуженням. Для адресації комірок пам’яті використаємо адресацію через допомогою регістру R1.

Проаналізувати парність або непарність числа можна за допомогою визначення значення 0 або 1 в нульовому розряді. Для цього можна виконати циклічний зсув числа вправо через ознаку переносу C, а потім проаналізувати її стан 0 чи 1 за допомогою команд JC або JNC.

```
        MOV R1,#40h         ;Запис адреси 40h до регістру R1 обраного банку пам’яті.
        MOV A,35h           ;Запис вмісту комірки РПД за адресою 35h до А.
        RRC A               ;Циклічний зсув вмісту А вправо через ознаку С.
        MOV A,@R1           ;Запис в А вмісту комірки РПД, адреса якої записана в R1.
        INC R1              ;Збільшення на 1 вмісту регістру R1 для адресації
                            ;наступної комірки РПД.
        JC L1               ;Якщо 1, то число непарне, перехід на мітку L1, якщо 0, то
                            ;наступна команда.
        ADD A,@R1           ;Додавання до А вмісту комірки РПД, адреса якої в R1.
        INC R1              ;Збільшення на 1 вмісту R1 для адресації наступної
комірки.
        MOV @R1,A           ;Запис вмісту А в комірку РПД, адреса якої в R1.
        SJMP L2             ;Безумовний перехід на L2, на кінець програми.
L1:
        CLR C               ;Скидання стану ознаки С.
        SUBB A,@R1          ;Видалення із А вмісту комірки РПД, адреса якої в R1.
        MOV 45h,A           ;Запис вмісту А в комірку РПД з адресою 45h.
L2:
        NOP
END
```

Хід роботи:

1. Ознайомитись із умовними алгоритмами роботи мікроконтролера сімейства MCS-51.
2. В інтегрованому середовищі MCU 8051 IDE та на навчальному стенді А2 ознайомитись із роботою команд різних груп та прикладів.
3. Для завдання, яке ставить викладач, розробити алгоритми та скласти програми.
4. Промоделювати роботу програм в середовищі MCU 8051 IDE та на навчальному стенді А2.
5. Зробити звіт з лабораторної роботи.

Контрольні питання:

1. Яка група команд мікроконтролера сімейства MCS-51 дозволяє реалізовувати умовні алгоритми?
2. Які ознаки операцій існують в мікроконтролері сімейства MCS-51?
3. Які існують типи циклічних алгоритмів?
4. Які існують типи розгалужених алгоритмів?
5. Чим відрізняються команди DJNZ та CJNE?

## Лабораторна робота № 4

## Тема: Порти введення-виведення мікроконтролера

**Мета: Отримати навички в роботі із портами введення-виведення мікроконтролерів та їх програмуванні.**

Мікроконтролер містить чотири порти вводу-виводу: P0, P1, P2 та P3, які можуть бути використані для організації вводу-виводу інформації за 32 лініями передачі. Кожен із портів містить восьмирозрядний регістр, що має байтову та бітову адресацію для встановлення або скидання розрядів цього регістру за допомогою програмного забезпечення. Виходи цих регістрів з'єднані із зовнішніми виводами мікроконтролера.

Окрім виконання функцій вводу-виводу порти мікроконтролера можуть виконувати додаткові функції:

– через порт P0 (рис. 4.1) видається молодший байт адреси і передається байт даних при роботі із зовнішньою пам'яттю програм і даних;

– через порт P2 видається старший байт адреси даних при роботі із зовнішньою пам'яттю програм і даних;

– кожна лінія порту P3 (рис. 4.2) має альтернативну функцію, яка реалізуються, якщо у відповідному розряді регістра-засувки записана 1.

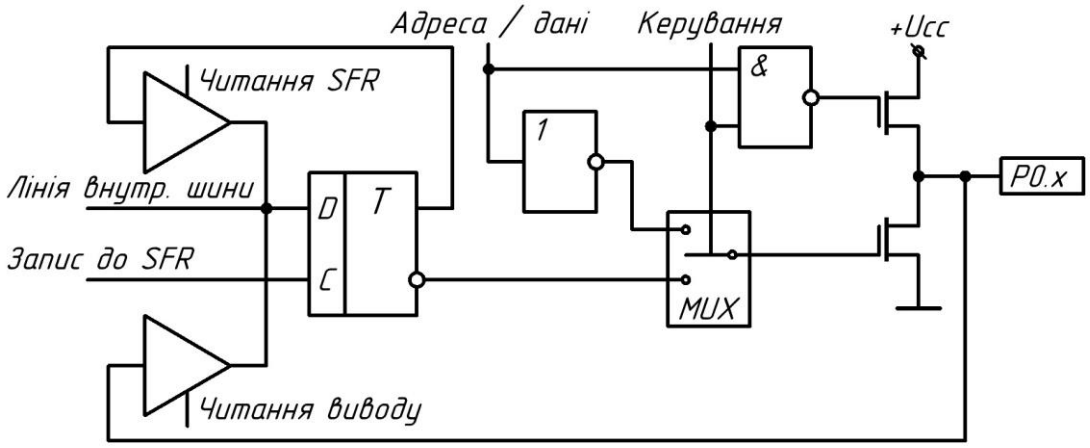



Рисунок 4.1 – Схема порту P0 мікроконтролера сімейства MCS-51

Один розряд регістра-засувки порту є D-тригер. Дані з внутрішньої шини мікроконтролера записуються в регістр-засувку за сигналом «Запис до SFR». Вихід Q D-тригера підключається до внутрішньої шини (зчитується) за сигналом «Читання SFR». Значення сигналу безпосередньо із зовнішнього виводу порту зчитується за сигналом «Читання виводу». Деякі команди читання порту використовують сигнал «Читання SFR», інші «Читання виводу».

Команди зчитування портів діляться на дві групи – команди, що зчитують

інформацію з регістрів-засувок, які використовують сигнал «Читання SFR», та команди, що зчитують інформацію із зовнішніх контактів виходів портів та використовують сигнал «Читання виводу». Перша група команд реалізує режим «зчитування-модифікація-запис», під час якого вміст регістра-засувки зчитується, за необхідності модифікується і знов записується в регістр-засувку. До таких команд належать команди ANL, ORL, XRL, JBC, CPL, INC, DEC, DJNZ, а також команди типу MOV Px.x,C, CLR Px.x, SETB Px.x.

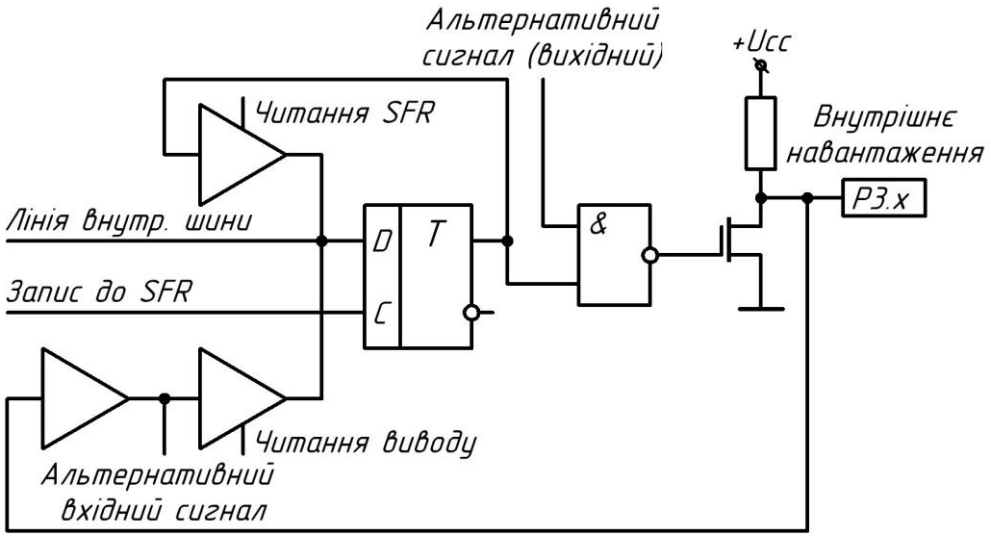



Рисунок 4.2 – Схема порту P3 мікроконтролера сімейства MCS-51

При записі числа в порт для зміни потенціали на виводах мікросхеми мікроконтролера можна використовувати командами з байтовою адресацією:

```
MOV P2,#01001110b     ;Встановити на виводах 1, 2, 3 та 6 порту P2 одиничні
                      ;рівні, а 0, 4 та 5 – нульові.
MOV P1,#7Bh           ;Встановити на виводах порту P1 число 7Bh.
MOV P2,A              ;Встановити на виводах порту P2 вміст акумулятора.
MOV P3,R2             ;Встановити на виводах порту P3 регістру R2.
```

Використання команди логічного «І» дозволяє встановлювати нульовий рівень на обраних лініях портів однією командою:

```
ANL P2,#11011110b     ;Встановити на виводах 0 та 5 порту P2 нульові рівні.
```

За допомогою команди логічного «АБО» можна встановлювати одиничні рівні на обраних лініях портів однією командою:

```
ORL P2,#11011110b     ;Встановити на виводах 1, 2, 3, 4, 6 та 7 порту P2 одиничні
                      ;рівні.
```

Команда «виключне АБО» дозволяє інвертувати стан ліній портів однією командою:

```
XRL P2,#11011110b     ;Інвертувати стан виводів 1, 2, 3, 4, 6 та 7 порту P2.
```

Наведені команди дозволяють змінювати стан декількох виводів

мікроконтролера одразу. Також для зміни стану на виводах мікроконтролера можна використовувати команди з бітовою адресацією

```
MOV P3.1,C      ;Запис вмісту біту переносу C до 1 біту порту P3.
CPL P1.5        ;Інвертувати біт 5 порту P1.
SETB P2.0       ;Записати 1 в 0 біт порту P2.
CLR P3.3        ;Записати 0 в 3 біт порту P3.
```

Таким самим чином можлива зміна станів окремих бітів регістрів спеціальних функцій.

При записі в розряд порту (у тригер Т) логічного 0 вихідний транзистор відкривається і на виводі мікросхеми з'являється низький потенціал, який змінити ззовні неможливо. Тому при опитуванні виводу мікросхеми вхідна інформація в цьому випадку завжди сприйматиметься як логічний 0 незалежно від стану виходів зовнішніх пристроїв. Якщо зазначений розряд записати логічну 1, то вихідний транзистор закривається і на виводі мікросхеми з'являється високий потенціал за рахунок генератора струму. Він може зовні бути змінений на нульовий потенціал при під’єднанні цього виводу мікросхеми до загального. У цьому випадку інформація, що зчитується мікроконтролером, буде відповідати інформації на виході зовнішнього пристрою. Тому, перед тим як здійснити введення інформації з якогось порту вводу-виводу, відповідний розряд необхідно налаштувати на введення, записавши в нього логічну 1.

З тієї ж причини при налаштуванні виводів порту на роботу з альтернативними функціями у відповідні розряди потрібно записати логічні 1.

Порт P0 використовується для організації шин адреси та даних при роботі мікроконтролера із зовнішньою пам'яттю даних або програм, та під час доступу до зовнішньої пам'яті в усі тригери-засувки порту P0 апаратно записуються 1 (тобто вміст порту втрачається), інформація у P2 при цьому залишається незмінною.

До ліній портів вводу-виводу мікроконтролерів можна підключати різні зовнішні пристрої, такі як кнопки керування рис. 4.3 а, малопотужні світлодіоди рис. 4.3 б, котушки реле (для індуктивного навантаження потрібно паралельно навантаженню встановлювати діод VD1) рис. 4.3 в, потужного навантаження рис. 4.3 г. При управлінні потужним навантаженням можна використовувати ключові схеми на основі біполярних чи польових транзисторів або спеціалізовані мікросхеми, які складаються із потужних складових ключів.

У мікропроцесорних пристроях та системах управління об'єктами події фіксуються з використанням різноманітних сенсорів цифрового та аналогового типів. Найбільш поширеними є двійкові сенсори типу «Так/Ні», прикладами

яких є також кнопки, що під’єднані до портів вводу-виводу.

Типова процедура очікування зовнішніх події з двійкового сенсора складається з таких дій як: введення сигналу від сенсора, аналіз значення сигналу та передача управління залежно від стану сенсора. Конкретна програмна реалізація процедури залежить від того, яким чином сенсор підключений до мікроконтролера. Наприклад, при підключенні сенсора (або кнопки) до лінії 0 порту P2 програма очікування розмикання контакту матиме наступний вигляд:

```
WAIT0:
        JNB P2.0,WAIT0          ;Очікування розмикання контакту сенсора.
```

Процедура очікування замикання контакту сенсора:

```
WAIT1:
        JB P2.0,WAIT1           ;Очікування замикання контакту сенсора.
```

Для опитування особливо важливих сенсорів з метою зменшення часу реакції на виняткову ситуацію в об'єкті доцільно використовувати режим переривання.

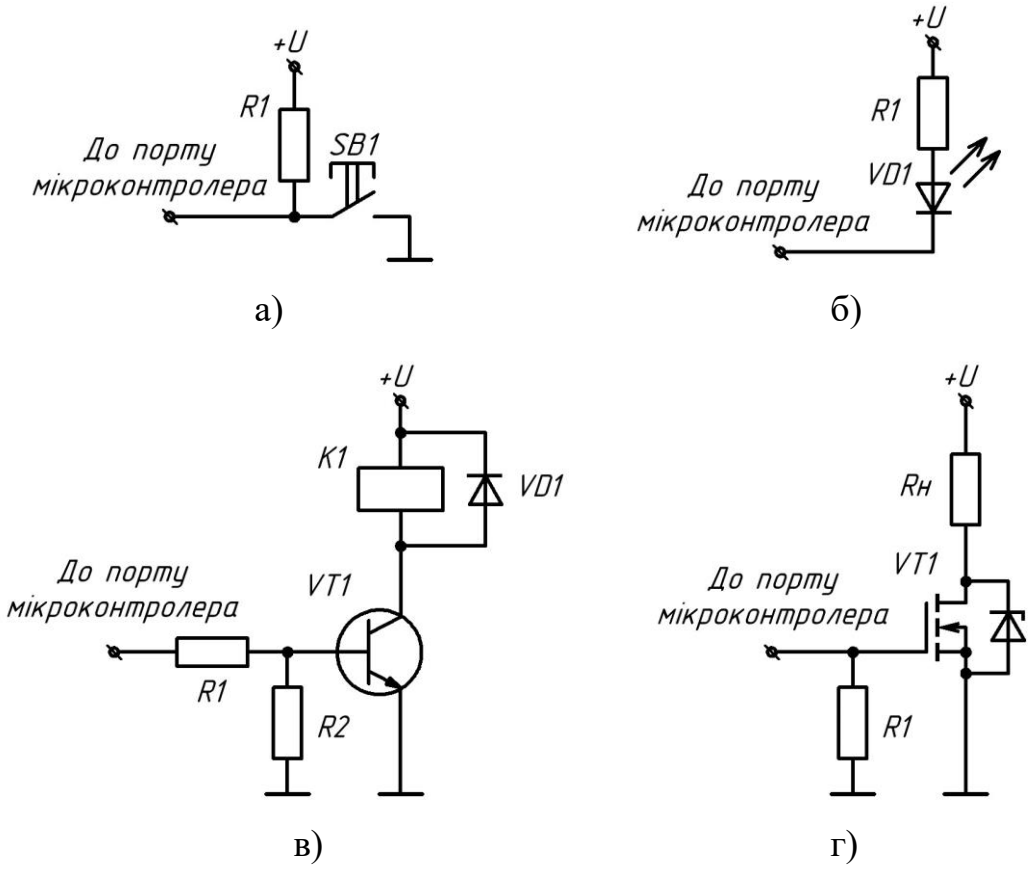



Рисунок 4.3 – Схеми підключення зовнішніх пристроїв до портів мікроконтролерів: а) кнопки; б) малопотужного світлодіода; в) котушки реле; г) потужного навантаження

Особливість процедури очікування імпульсного сигналу полягає в тому, що мікроконтролер повинен виявити не тільки факт появи, але і факт закінчення сигналу. Для програмування цієї процедури зручно скористатися прикладами з процедурами WAIT0 та WAIT1.

Послідовність процедур WAIT0 та WAIT1 залежить від форми імпульсу. Для «негативного» імпульсу (1→0→1) процедура WAIT1 передує процедурі WAIT0, для «позитивного» (0→1→0) слідує за нею.

Приклад програмної реалізації процедури очікування «негативного» імпульсного сигналу при підключенні сенсора до лінії 7 порту P1 за умови, що початковий стан входу одиничний.

```
WAIT1:
        JB P1.7, WAIT1          ;Очікування нульового рівня на лінії P1.7.
WAIT0:
        JNB P1.7, WAIT0         ;Очікування одиничного рівня на лінії P1.7.
```

Програмна реалізація циклу очікування накладає обмеження на тривалість імпульсу – імпульси тривалістю менше часу виконання циклу очікування можуть бути «не помічені» мікроконтролером. Для виявлення короткочасних імпульсів зазвичай використовують спосіб фіксації імпульсу на зовнішньому тригері прапора. На вхід у цьому випадку надходить не короткочасний сигнал із сенсору, а прапор, який формується тригером. Тригер встановлюється фронтом імпульсу, а скидається програмним шляхом – подачею спеціального керуючого впливу. Тривалість імпульсу при цьому буде обмежена знизу лише швидкодією тригера.

При роботі з двійковими сенсорами, які мають механічні або електромеханічні контакти (кнопки, клавіші, реле і клавіатури), виникає явище, що називається брязкотом. Він полягає в тому, що при замиканні контактів можлива поява відскоку контактів, що призводить до перехідного процесу. При цьому сигнал з контакту може бути прочитаний мікроконтролером як випадкова послідовність нулів та одиниць. Придушити це небажане явище можна схемотехнічними засобами, але найчастіше це робиться програмним шляхом.

Найбільшого поширення набули два програмних способи очікування встановленого значення.

1. Підрахунок заданого числа відповідних значень сигналу, суть якого полягає в багаторазовому зчитуванні сигналу з контакту. Підрахунок успішних опитувань, які виявили, що контакт стабільно замкнутий, ведеться лічильником. Якщо після серії успішних опитувань зустрічається невдалий, то

підрахунок починається спочатку. Контакт вважається стабільно замкнутим, якщо було N вдалих опитувань. Число N підбирається експериментально для кожного типу використовуваних сенсорів та лежить в межах від 5 до 50. Приклад програмного придушення брязкоту контакту наводиться для випадку, коли сенсор імпульсного сигналу підключений до лінії P3.4, рахунок вдалих опитувань ведеться в регістрі R3, N = 20.

```
L1:
        MOV R3,#20d             ;Ініціалізація лічильника.
L2:
        JB P3.4,L1              ;Якщо контакт розімкнутий, то почати відлік опитувань спочатку.
        DJNZ R3,L2              ;Повторювати, поки R3 не стане рівним 0.
```

2. Спосіб усунення брязкоту контакту шляхом введення часової затримки, при якому програма виявляє замикання контакту, забороняє опитування його стан на певний час, який більше тривалості перехідного процесу. Приклад програми написаний для випадку підключення сенсора до лінії P3.4 та програмної реалізації часової затримки:

```
L1:
        JB P3.4,L1              ;Очікування нульового рівня на лінії P3.4.
        CALL DELAY              ;Виклик підпрограми затримки.
EXIT:                           ;Вихід із процедури.
```

Час роботи часової затримки підбирається для кожного типу сенсора та реалізується підпрограмою DELAY.

**Приклад 1.**

Розробити програму, яка при натисканні кнопки SB19 (рис. 4.4) буде запалювати світлодіоди HL1-HL4 та гасити HL5-HL8, а при натисканні кнопки SB20 навпаки, запалювати діоди HL5-HL8 та гасити HL1-HL4.

Згідно із схемою (рис. 4.4) кнопки під’єднані до ліній 2 та 3 порту P3, а світлодіоди до ліній 0-7 порту P2. При роботі програми використовується програмний антибрязкіт. Для того, щоб запалити діод потрібно на відповідну лінію порту P2 подати нульовий рівень.

На регістрі R7 та за допомогою команди DJNZ зроблено програмний лічильник.

Алгоритм роботи програми представлено на рис. 4.5.

```
        SETB P3.2               ;Налагодження лінії 2 порту P3 на введення.
        SETB P3.3               ;Налагодження лінії 3 порту P3 на введення.
L1:
        JB P3.2,L2              ;Перевірка стану лінії 2 порту P3, якщо 1, то на L2.
```

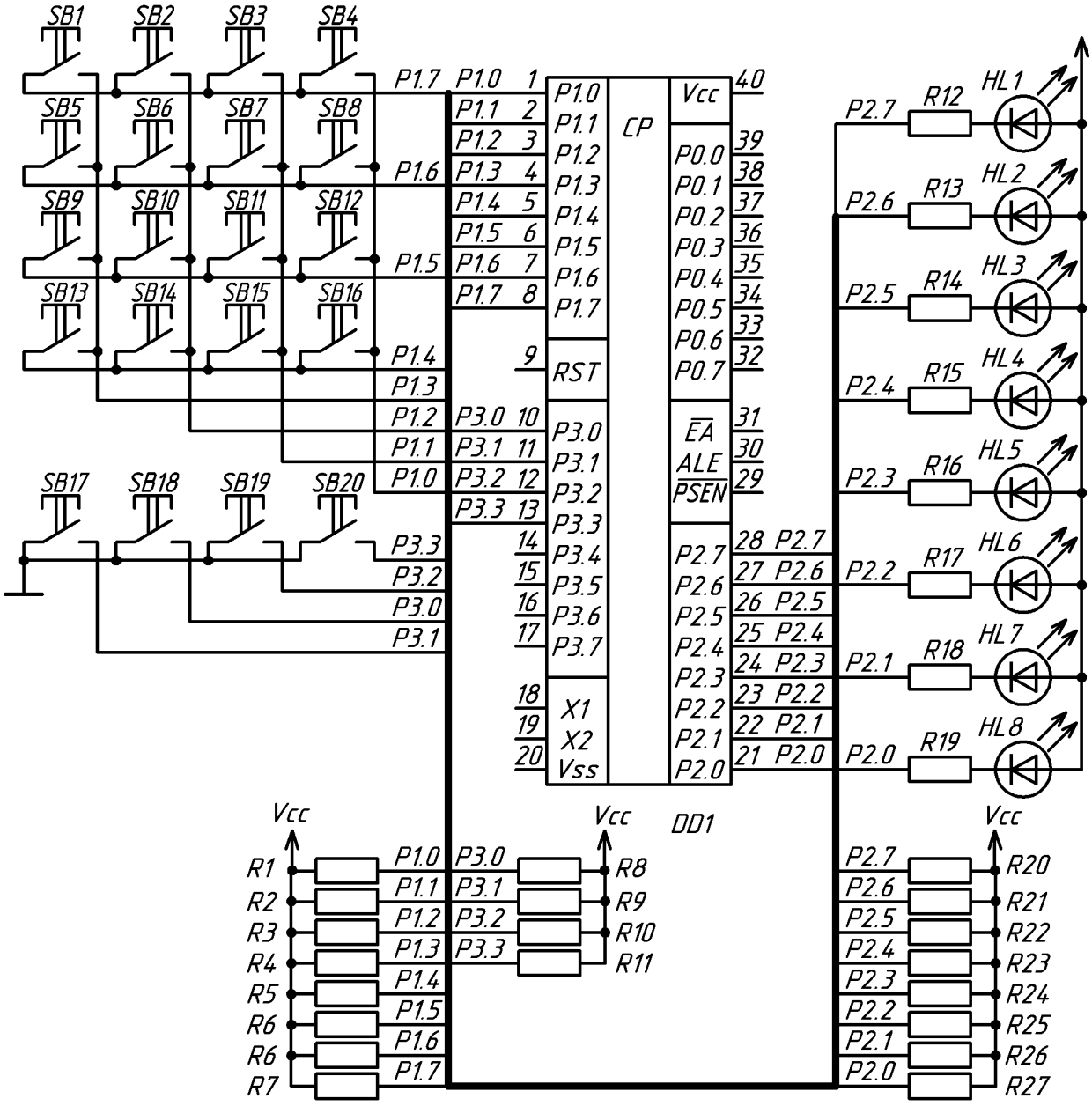



Рисунок 4.4 – Схема підключення кнопок та світлодіодів стенда А2

```
        ACALL DELAY         ;Виклик підпрограми часової затримки.
        JB P3.2,L2          ;Перевірка стану лінії 2 порту Р3, якщо 1, то на L2.
        MOV P2,#00001111b   ;Запис у порт Р2 числа для запалювання світлодіодів.
        JNB P3.2,$          ;Перевірка стану лінії 2 порту Р3 та очікування якщо 1.
        ACALL DELAY         ;Виклик підпрограми часової затримки.
        JNB P3.2,$          ;Перевірка стану лінії 2 порту Р3 та очікування якщо 0.
L2:
        JB P3.3,L3          ;Перевірка стану лінії 3 порту Р3, якщо 1, то на L2.
        ACALL DELAY         ;Виклик підпрограми часової затримки.
        JB P3.3,L3          ;Перевірка стану лінії 3 порту Р3, якщо 1, то на L2.
        MOV P2,#11110000b   ;Запис у порт Р2 числа для запалювання світлодіодів.
        JNB P3.3,$          ;Перевірка стану лінії 3 порту Р3 та очікування якщо 1.
```

```
        ACALL DELAY         ;Виклик підпрограми часової затримки.
        JNB P3.2,$          ;Перевірка стану лінії 3 порту P3 та очікування якщо 0.
L3:
        SJMP L1             ;Безумовний перехід на мітку L1.

DELAY:                      ;Підпрограма часової затримки.
        MOV R7,#250d        ;Запис в R7 числа, яке визначає часову затримку.
        DJNZ R7,$           ;Зменшення на 1 вмісту R7, та очікування якщо не 0.
        RET                 ;Повернення із підпрограми.
END
```

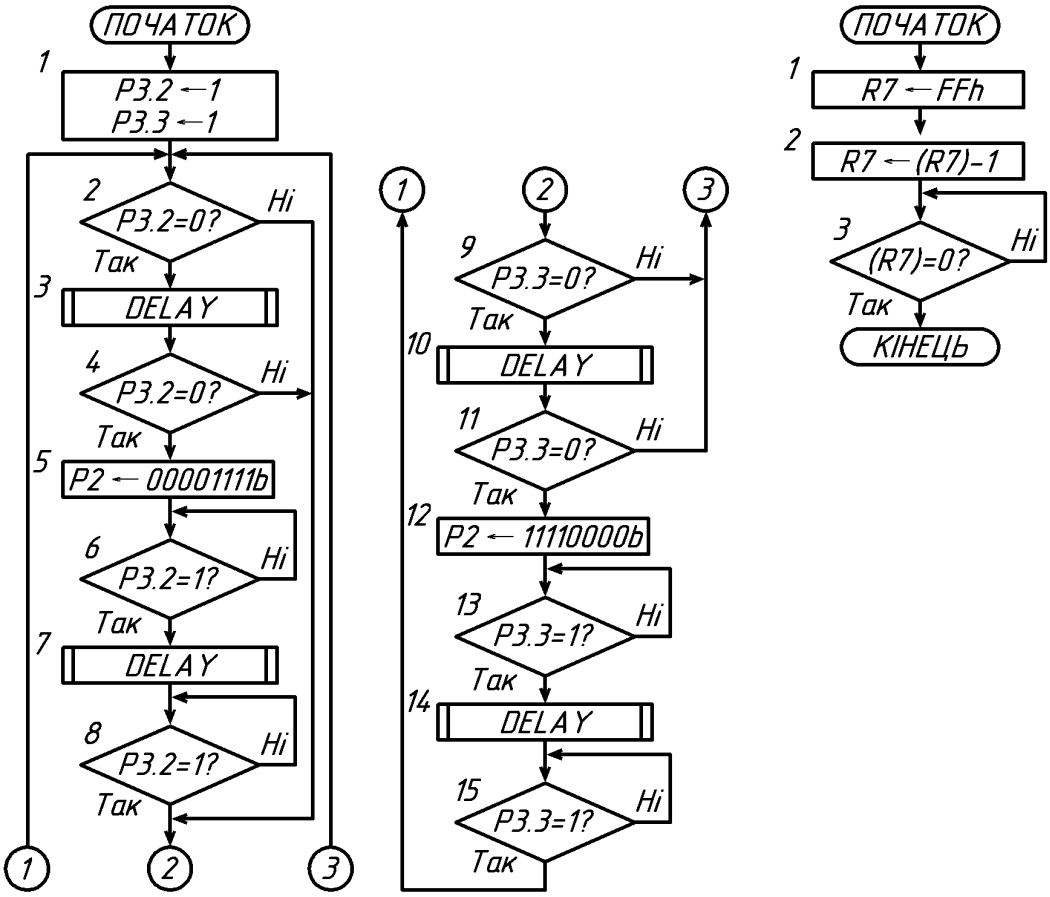



Рисунок 4.5 – Алгоритм виконання прикладу 1

**Приклад 2.**

Розробити програму для пристрою на мікроконтролері, який при виявленні перешкоди інфрачервоним сенсором перешкод FC-51 (рис. 4.6) буде зупиняти рух електричного мікродвигуна постійного струму, а при відсутності перешкоди буде запускати мікродвигун.

Для керування мікродвигуном використовується мікросхема з потужними складовими ключами, яка при подачі на вхід високого рівня напруги включає електричний двигун підключений до відповідного виходу.

Для керування електричним двигуном використовується лінія 0 порту P1. Вихід інфрачервоного сенсору перешкод приєднаний до лінії 7 порту P3.

```
        MOV P1,#0           ;Початкове налаштування ліній порту P1.
        SETB P3.7           ;Налаштування лінії 7 порту P3 на введення інформації.
L1:
        JB P3.7,L2          ;Перевірка спрацювання сенсора перешкоди.
        CLR P1.0            ;Якщо на виході сенсора 0, то зупиняється двигун
        SJMP L1             ;Безумовний перехід на мітку L1.
L2:
        SETB P1.0           ;Якщо на виході сенсора 1, то включається двигун.
        SJMP L1             ;Безумовний перехід на мітку L1.
END
```

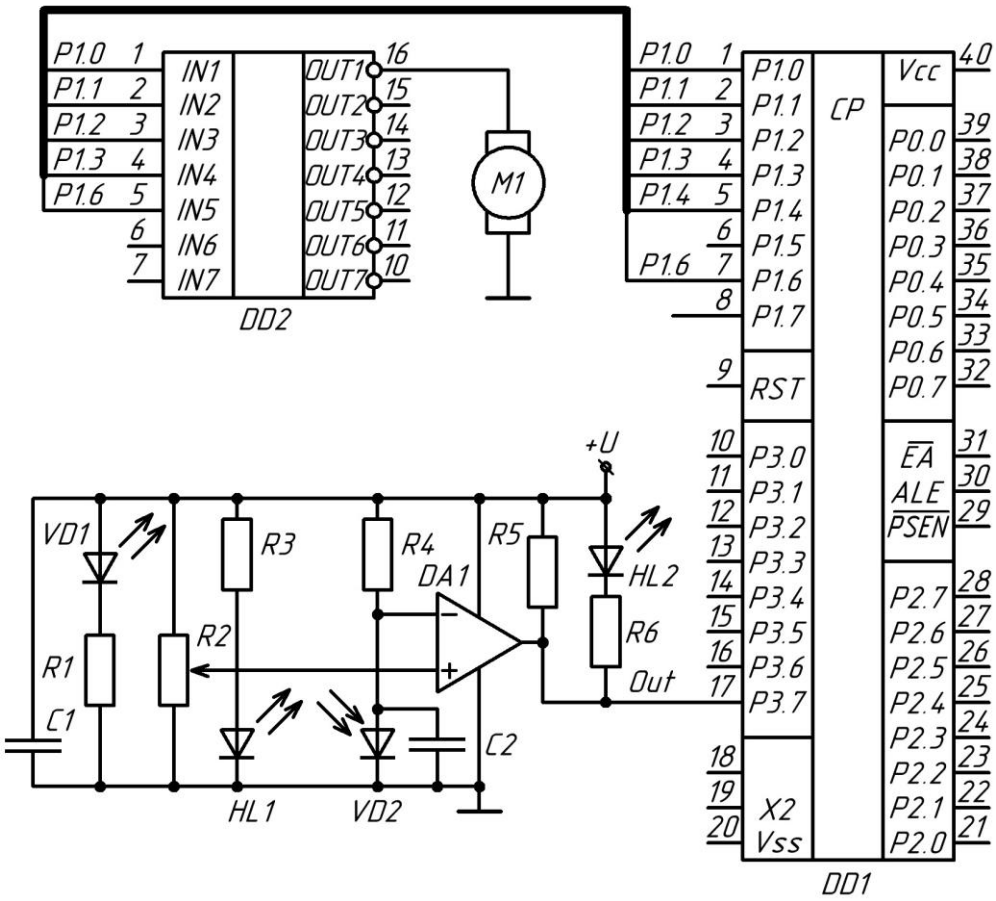



Рисунок 4.6 – Схема підключення сенсора FC-51 та двигуна для прикладу 2

Хід роботи:

1. Ознайомитись із будовою портів введення-виведення мікроконтролера сімейства MCS-51.

2. В інтегрованому середовищі MCU 8051 IDE та на навчальному стенді А2 ознайомитись із роботою команд різних груп та прикладів.

3. Для завдання, яке ставить викладач, розробити алгоритми та скласти

програми.

4. Промоделювати роботу програм в середовищі MCU 8051 IDE та на навчальному стенді А2.

5. Зробити звіт з лабораторної роботи.

Контрольні питання:

1. Чи відрізняються схеми портів P0-P3 мікроконтролера сімейства MCS-51?

2. Які функції виконують порти вводу-виводу мікроконтролера сімейства MCS-51?

3. Як до портів вводу-виводу мікроконтролера під'єднуються зовнішні пристрої?

4. Які команди реалізують режим «зчитування-модифікація-запис»?

5. Як можна встановити на лініях 1, 2, 4 та 6 порту P2 одиниці?

## Лабораторна робота № 5

### Тема: Системи відображення інформації мікроконтролера

**Мета: Отримати навички в роботі із системами відображення інформації мікроконтролерів та їх програмуванні.**

При роботі мікропроцесорних систем потрібно якимось чином передавати інформацію людини. Для простого відображення інформації можуть використовуватись набори світлодіодів, семисегментні світлодіодні та рідкокристалічні індикатори, світлодіодні матриці, графічні дисплеї. Найпростіший варіант – це лінійки світлодіодів. Але інформація, яку вони надають представлена двійковим кодом, що дуже незручно. Якщо потрібно лише відображати числа, то можливе використання світлодіодних семисегментних індикаторів, які прості в управлінні та можуть використовуватися спільно з будь-яким мікроконтролером із достатньою кількістю виводів. Рідкокристалічні семисегментні індикатори мають наднизьке енергоспоживання, що дуже важливо при використанні батарейок та акумуляторів в якості елементів живлення.

Семисегментні індикатори мають просту конструкцію, дешеві, надійні, забезпечують високу яскравість і контрастність інформації, що відображається. Існує велика різноманітність індикаторів: з різним кольором світіння сегментів, різного розміру, різними схемами підключення світлодіодів (із загальним

катодом або загальним анодом). За необхідності відображення кількох розрядів можна встановити кілька однорозрядних індикаторів чи вибрати потрібний варіант багаторозрядного індикатора.

Назву семисегментні індикатори отримали у зв'язку з тим, що зображення символу формується за допомогою семи елементів – сегментів, які управляються окремо. Ці елементи дозволяють відобразити будь-яку цифру від 0 до 9, а також деякі символи. Це дозволяє використовувати індикатор для виведення чисел різних систем числення та текстових повідомлень. Зазвичай індикатор має також восьмий елемент – точку. Сегменти індикатора позначають літерами a, b, c, d, e, f, g (a – верхній елемент, далі літери присвоюються сегментам за годинниковою стрілкою; g – центральний сегмент; dp – точка).

Вісім елементів, кожен з яких може перебувати в одному з двох станів – світиться або не світиться, дають 256 можливих комбінацій або 128 комбінацій з точкою.

Існує два варіанти семисегментних індикаторів (рис. 5.1): із загальними катодами або загальними анодами. Усього для підключення використовується 9 виводів – 1 загальний та 8 окремих виводів світлодіодів.

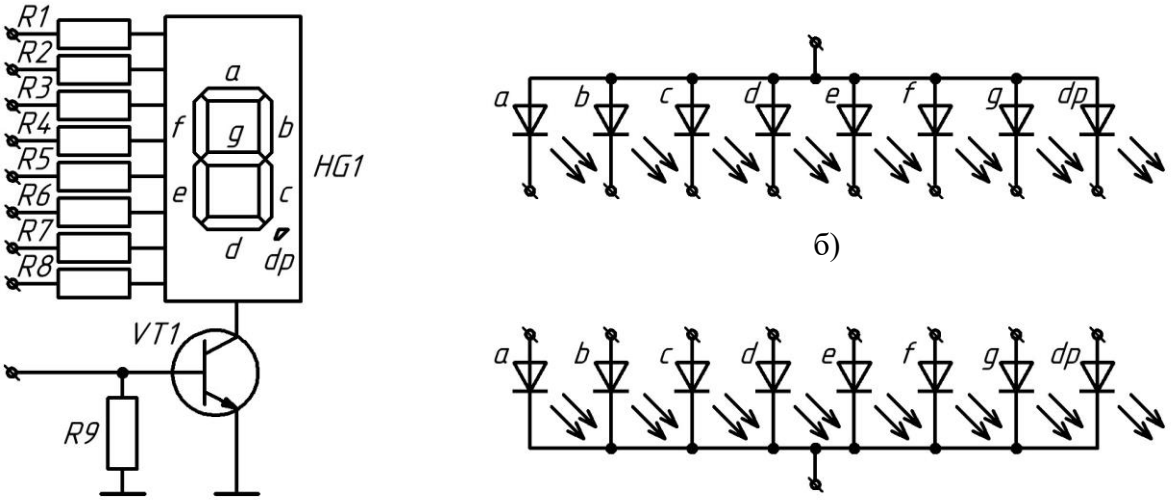



Рисунок 5.1 – Семисегментний індикатор: а) схема підключення; б) схема з загальним анодом; в) схема з загальним катодом

У разі, якщо світлодіоди в індикаторі мають з'єднані разом аноди (схема із загальним анодом), загальний анод підключається до позитивного виводу джерела напруги, а катоди світлодіодів – сегментів підключаються до схеми управління (рис. 5.1, б). Запалюються сегменти низьким рівнем (логічний 0) на виводах схеми управління. Змінюючи величину живлячої напруги індикатора,

можна регулювати яскравість свічення.

Якщо в індикаторі з'єднані разом катоди (схема із загальним катодом), то загальний катод підключається до загального провідника схеми, а аноди світлодіодів підключаються до схеми управління (рис. 5.1, в). У цьому випадку сегмент запалюється високим рівнем на виході схеми управління.

При використанні мікроконтролера з малим вихідним струмом виходів або індикатора з великим струмом, підключення здійснюється через драйвер – інтегральну мікросхему, що містить набір повторювачів або інверторів з потужними виходами. Також можна використовувати транзистори в якості ключів для управління індикатором.

При використанні світлодіодів, які мають різку залежність струму від напруги на світлодіоді, потрібна стабілізація струму світлодіодів для забезпечення роботи у номінальному режимі. Зазвичай використовується найпростіший спосіб – послідовне включення резисторів, що задають струм (рис. 5.1, а).

Для формування зображення символу на індикаторі використовують дешифратор семисегментного індикатора або таблицю, яка ставить у відповідність коду символу набір сегментів, що відображаються. Набір сегментів, який формує символ, розглядається як двійкове число. Якщо біт числа дорівнює 0, відповідний сегмент не запалюється при відображенні символу, а якщо дорівнює 1, то запалюється. Приклад кодів чисел – цифрі 0 відповідає код 3Fh, цифрі 2 – 06h, цифрі 3 – 5Bh.

За способом включення індикаторів в електричну схему виділяються два режими: статична та динамічна індикація.

Спосіб статичної індикації полягає у постійному підсвічуванні індикатора від джерела інформації, тобто кожен із цифрових індикаторів блоку індикації постійно підключений до джерела інформації.

Якщо застосовується велика кількість індикаторів, тоді апаратна реалізація статичної індикації недоцільна, тому що для цього знадобиться велика кількість виводів джерела інформації та до великого споживання струму індикаторами.

Динамічна індикація, на відміну статичної, передбачає використання набагато меншої кількості контактів передачі інформації, у кожному наборі із семи сегментів, однойменні сегменти під’єднуються одного виводу.

Принцип дії динамічної індикації полягає в тому, що інформація відображається не у всіх розрядах індикатора відразу, а по черзі, у кожний

момент часу лише в одному розряді. У зв'язку з тим, що зір людини інерційний, необов'язково щоб усі елементи зображення світилися безперервно та одночасно. Якщо з високою частотою послідовно перемикатися від відображення одного розряду до наступного, а при досягненні останнього розряду індикатора знову переходити до відображення першого, то око людини це сприйматиме так, як якщо б кожен розряд відображав інформацію статично. Для цього у циклі виконуються такі дії:

1) гасяться всі розряди індикатора для запобігання появі артефактів на зображенні при зміні стану шини управління сегментами;

2) на шину управління сегментами видаються сигнали відображення символу в черговому розряді;

3) видається сигнал для запалення чергового розряду;

4) робиться пауза, протягом якої відбувається відображення інформації.

Індикатори, що містять кілька розрядів, випускаються в розрахунку на динамічну індикацію, та всі необхідні з'єднання виконані всередині пристрою (рис. 5.2). N-розрядний індикатор у цьому випадку має 8 виводів для керування сегментами та N виводів для керування включенням розрядів (загальний анод або катод розряду).

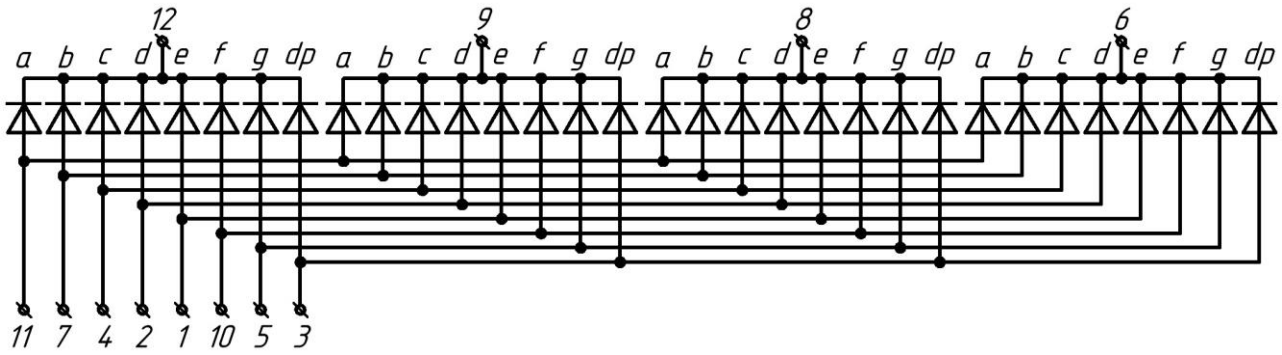



Рисунок 5.2 – Схема семисегментного індикатора 3462AS-1

Для того щоб не було помітно мерехтіння частота регенерації зображення повинна бути не менше ніж 100 Гц.

Кожен розряд горить протягом не більше 1/N від періоду регенерації. При швидкому перемиканні розрядів мерехтіння непомітне, але сприймається усереднена яскравість. Вона становитиме 1/N від величини у разі статичної індикації при тих самих струмах через світлодіоди. Тому силу струму під час імпульсів при динамічній індикації потрібно збільшувати проти сили струму при статичній. Призначені для динамічної індикації індикатори розраховані на це, вони мають досить великий максимально допустимий імпульсний струм, що

в кілька разів перевищує максимальний середній струм.

В навчальному стенді А2 має два семисегментних індикатора по чотири цифри (рис. 5.3) та матрицю світлодіодів (рис. 5.4), для роботи яких використовується динамічна індикація.

**Приклад 1.**

Використовуючи статичну індикацію вивести на семисегментний індикатор навчальному стенду А2 цифру 8.

В схемі підключення семисегментних індикаторів у стенді А2 застосовуються 2 індикатори типу 3462AS-1 з загальним катодом, восьмеричний приймач/передавач шини з неінвертуючими виходами, які мають 3 стани, з входами активації виходу і відправки-прийому 74HC245 та дешифратор, що містить 3 бінарні зважені входи та 3 входи активації 74HC138.

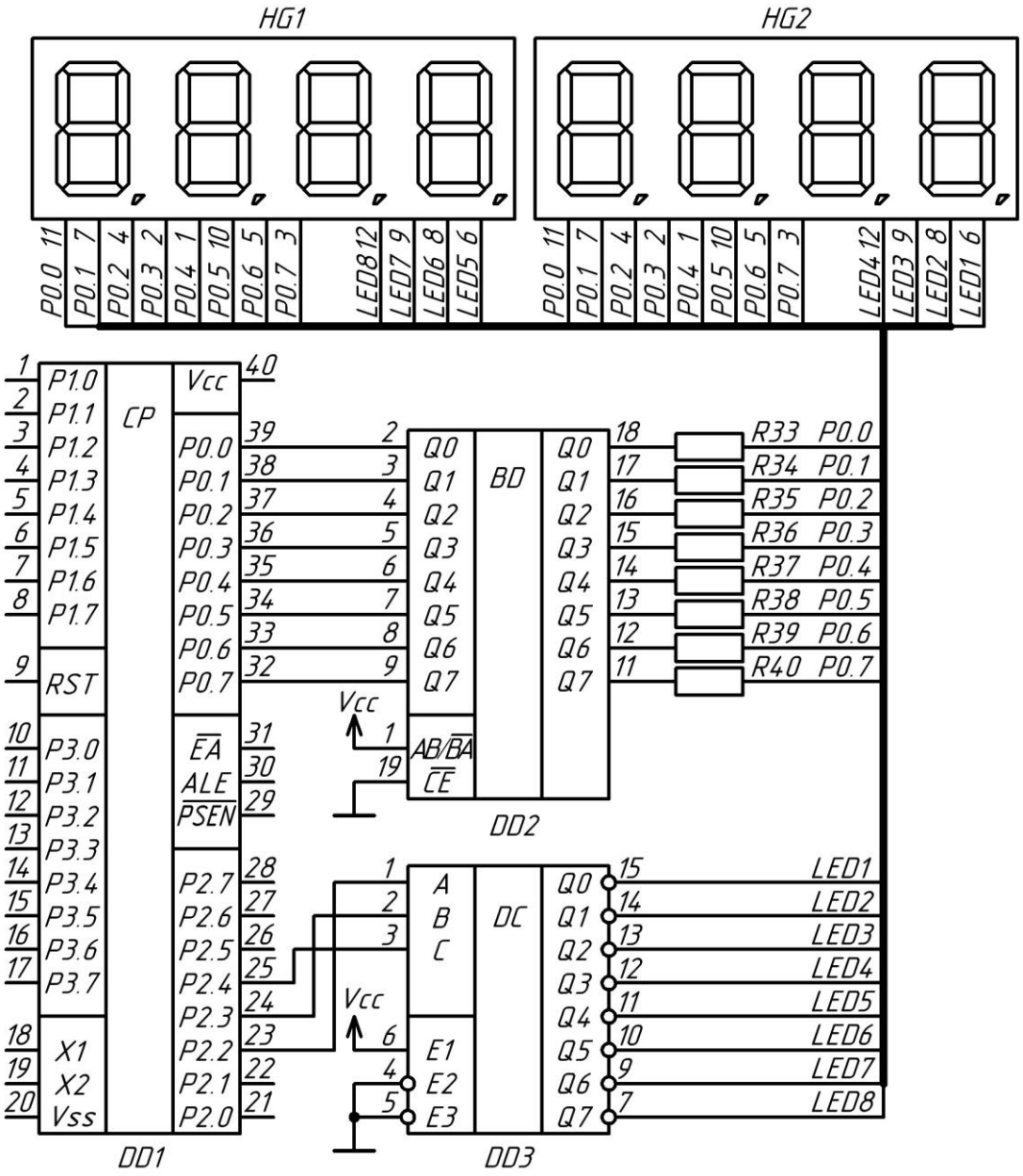


Рисунок 5.3 – Схема підключення семисегментних індикаторів стенда А2

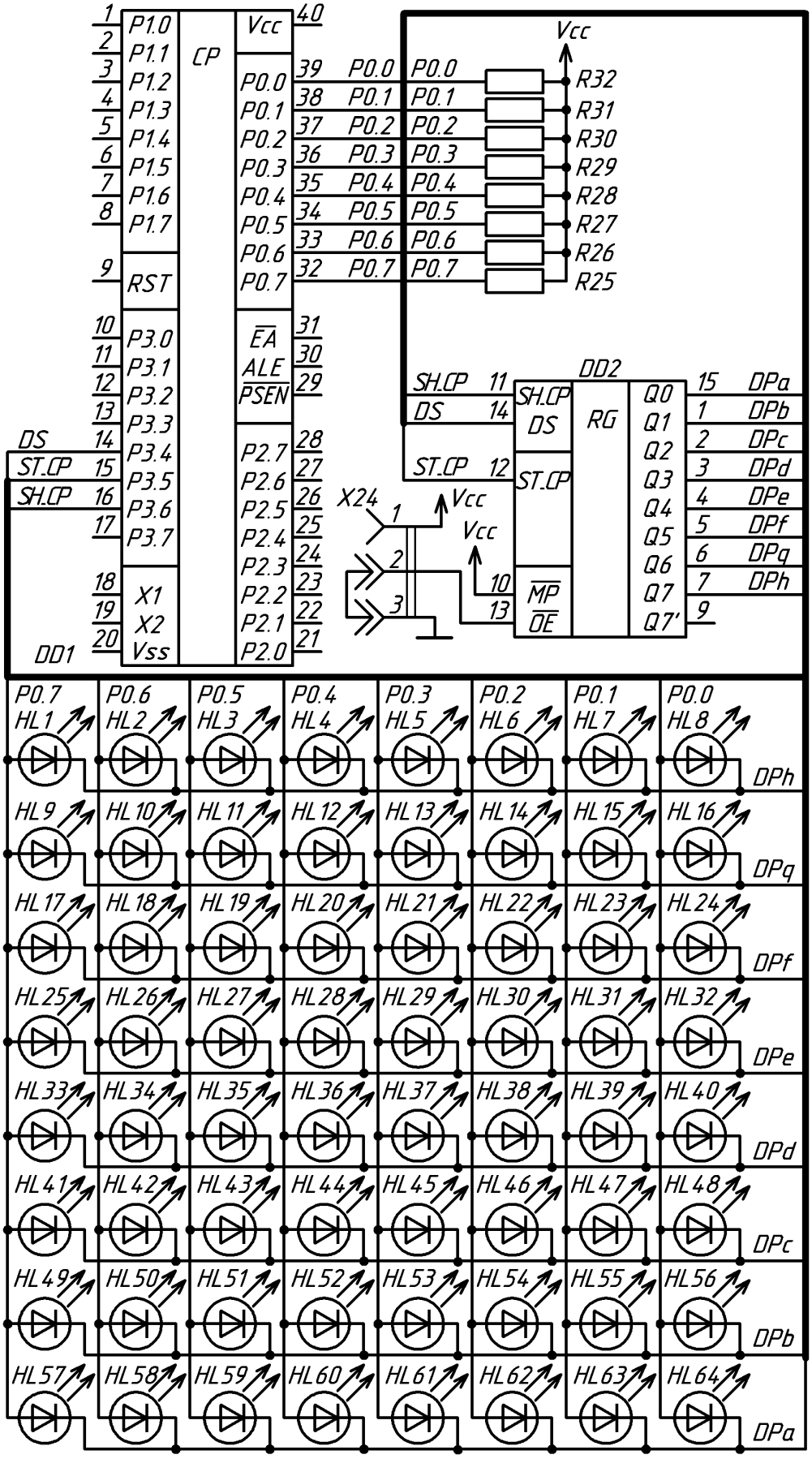


Рисунок 5.4 – Схема підключення матриці світлодіодів стенда А2

Індикатори під’єднані до портів Р0 та Р2 мікроконтролера. Лініями порту Р0 передається код символу, який запалюється, а лініями порту Р2 передається номер обраної позиції символу на дешифратор, до якого приєднанні загальні катоди індикаторів.

```
        MOV P2,#00000000b       ;Запис в порт Р2 комбінації, яка включає індикатор.
        MOV P0,#7Fh             ;Запис коду числа 8 в порт Р0.
END
```

**Приклад 2.**

Розробити програму для мікроконтролера, яка буде виводити числа з 0 по 7 на семисегментні індикатори навчального стенда А2. Індикація динамічна.

```
        MOV P0,#00h             ;Початкова установка значень 0 на лініях порту Р0
        MOV P2,#00h             ;Початкова установка значень 0 на лініях порту Р2
L1:
        MOV P0,#07h             ;Запис в порт Р0 коду числа 7.
        MOV P2,#00000000b       ;Запис в порт Р2 коду для запалювання позиції 1.
        ACALL DELAY             ;Виклик підпрограми затримки часу DELAY.
        MOV P0,#0               ;Запис в порт Р0 числа 0 для гасіння усіх індикаторів.
        MOV P2,#00000100b       ;Запис в порт Р2 коду для запалювання позиції 2.
        MOV P0,#7Dh             ;Запис в порт Р0 коду числа 6.
        ACALL DELAY             ;Виклик підпрограми затримки часу DELAY.
        MOV P0,#0               ;Запис в порт Р0 числа 0 для гасіння усіх індикаторів.
        MOV P2,#00001000b       ;Запис в порт Р2 коду для запалювання позиції 2.
        MOV P0,#6Dh             ;Запис в порт Р0 коду числа 5.
        ACALL DELAY             ;Виклик підпрограми затримки часу DELAY.
        MOV P0,#0               ;Запис в порт Р0 числа 0 для гасіння усіх індикаторів.
        MOV P2,#00001100b       ;Запис в порт Р2 коду для запалювання позиції 3.
        MOV P0,#66h             ;Запис в порт Р0 коду числа 4.
        ACALL DELAY             ;Виклик підпрограми затримки часу DELAY.
        MOV P0,#0               ;Запис в порт Р0 числа 0 для гасіння усіх індикаторів.
        MOV P2,#00010000b       ;Запис в порт Р2 коду для запалювання позиції 4.
        MOV P0,#4Fh             ;Запис в порт Р0 коду числа 3.
        ACALL DELAY             ;Виклик підпрограми затримки часу DELAY.
        MOV P0,#0               ;Запис в порт Р0 числа 0 для гасіння усіх індикаторів.
        MOV P2,#00010100b       ;Запис в порт Р2 коду для запалювання позиції 5.
        MOV P0,#5Bh             ;Запис в порт Р0 коду числа 2.
        ACALL DELAY             ;Виклик підпрограми затримки часу DELAY.
        MOV P0,#0               ;Запис в порт Р0 числа 0 для гасіння усіх індикаторів.
        MOV P2,#00011000b       ;Запис в порт Р2 коду для запалювання позиції 6.
        MOV P0,#06h             ;Запис в порт Р0 коду числа 1.
```

```
        ACALL DELAY             ;Виклик підпрограми затримки часу DELAY.
        MOV P0,#0               ;Запис в порт P0 числа 0 для гасіння усіх індикаторів.
        MOV P2,#00011100b       ;Запис в порт P2 коду для запалювання позиції 7.
        MOV P0,#3Fh             ;Запис в порт P0 коду числа 0.
        ACALL DELAY             ;Виклик підпрограми затримки часу DELAY.
        MOV P0,#0               ;Запис в порт P0 числа 0 для гасіння усіх індикаторів.
        AJMP L1

DELAY:                          ;Підпрограма часової затримки.
        MOV R7,#250             ;Запис в регістр R7 числа кількості ітерацій.
        DJNZ R7,$               ;Зменшення на 1 вмісту регістру R7 поки не досягне 0.
        RET                     ;Повернення із підпрограми.
END
```

**Приклад 3.**

Розробити програму для мікроконтролера, яка буде виводити числа на семисегментні індикатори навчального стенда А2, що записані до комірок пам’яті з адресами 30h-37h.

Для виконання цього завдання таблиця кодів символів індикаторів буде записана до РПП, а доступ до кодів символів для семисегментних індикаторів буде виконуватись через код числа та базову адресу таблиці (TABLE) за допомогою команди MOVC A,@A+DPTR, в якій в DPTR записано базову адресу TABLE, а в А число для відображення.

```
        MOV 30h,#01h            ;Початкове налаштування, запис у комірку РПД числа 1.
        MOV 31h,#02h            ;Початкове налаштування, запис у комірку РПД числа 2.
        MOV 32h,#03h            ;Початкове налаштування, запис у комірку РПД числа 3.
        MOV 33h,#04h            ;Початкове налаштування, запис у комірку РПД числа 4.
        MOV 34h,#05h            ;Початкове налаштування, запис у комірку РПД числа 5.
        MOV 35h,#06h            ;Початкове налаштування, запис у комірку РПД числа 6.
        MOV 36h,#07h            ;Початкове налаштування, запис у комірку РПД числа 7.
        MOV 37h,#08h            ;Початкове налаштування, запис у комірку РПД числа 8.
        MOV SP,#60h             ;Налаштування розташування стеку з адреси 60h.
        MOV DPTR,#TABLE         ;Запис до DPTR базової адреси таблиці кодів символів.
        MOV P0,#00h             ;Початкове налаштування, запис у порт P0 числа 00h.
        MOV P2,#00h             ;Початкове налаштування, запис у порт P2 числа 00h.
        MOV R1,#01h             ;Початкове налаштування, запис у регістр R1 числа 01h.
M1:
        ACALL DIN_IND           ;Виклик підпрограми динамічної індикації DIN_IND.
        ACALL DELAY             ; Виклик підпрограми часової затримки DELAY.
        NOP
        AJMP M1                 ;Безумовний перехід на мітку M1.
```

```
DIN_IND:
        MOV P0,#00h             ;Запис в порт P0 числа 00h для гасіння всіх розрядів.
        DJNZ R1,DI1             ;Зменшення вмісту R1 та перевірка на 0 (кількість циклів).
        MOV R1,#08h             ;Запис до регістру R1 числа 08h, налаштування кількості
                                ;циклів.
        MOV R0,#37h             ;Запис до регістру R0 числа 37h, налаштування регістрової
                                ;адресації через регістр R0, для вибору чисел.
        ANL P2,#11100011b       ;Логічне «І» між вмістом порту P2 та числом 11100011b,
                                ;підключення першого розряду індикатора.
        AJMP DI2                ;Безумовний перехід на мітку DI2.
DI1:
        MOV A,P2                ;Запис вмісту регістру порту P2 до акумулятора.
        ADD A,#04h              ;Збільшення A на 04h.
        MOV P2,A                ;Запис вмісту A в порт P2, включається наступний розряд.
DI2:
        MOV A,@R0               ;Запис в A вмісту комірки РПД, адреса якої в R0.
        MOVC A,@A+DPTR          ;Запис в A символу для індикатора.
        MOV P0,A                ;Запис в порт P0 вмісту акумулятора A.
        DEC R0                  ;Зменшення на 1 вмісту регістру R0.
        RET                     ;Повернення із підпрограми DIN_IND.

DELAY:                          ;Підпрограма часової затримки.
        MOV R7,#250d            ;Запис в R7 числа 250d, яке визначає часову затримку.
        DJNZ R7,$               ;Зменшення на 1 вмісту R7, очікування 0 в R7.
        RET                     ;Повернення із підпрограми DELAY.

TABLE:                          ;Таблиця кодів символів.
        DB 3Fh,06h,5Bh,4Fh,66h,6Dh,7Dh,07h      ;Запис в РПП кодів символів для індикатора.
        DB 7Fh,6Fh,77h,7Ch,39h,5Eh,79h,71h
END
```

Хід роботи:

1. Ознайомитись із способами відображення інформації мікроконтролера сімейства MCS-51.

2. В інтегрованому середовищі MCU 8051 IDE та на навчальному стенді А2 ознайомитись із роботою команд різних груп та прикладів.

3. Для завдання, яке ставить викладач, розробити алгоритми та скласти програми.

4. Промоделювати роботу програм в середовищі MCU 8051 IDE та на навчальному стенді А2.

5. Зробити звіт з лабораторної роботи.

Контрольні питання:

1. Яким чином інформація представляється в мікропроцесорних системах?
2. Чим відрізняється статична та динамічна індикація?
3. Яким алгоритмом можна описати роботу динамічної індикації?
4. Які міри приймаються при роботі мікропроцесорних систем для усунення брязкоту контактів?
5. Опишіть алгоритм роботи програмного антибрязкоту контактів.

## Лабораторна робота № 6

## Тема: Таймери-лічильники мікроконтролера

**Мета: Отримати навички в роботі із таймерами-лічильниками мікроконтролерів та їх програмуванні.**

Два шістнадцятирозрядні таймера/лічильника T/C0 та T/C1 призначені для отримання програмно керованих часових затримок і підрахунку зовнішніх подій. Кожний з них складається з двох восьмирозрядних регістрів: старшого – THx і молодшого – TLx.

Під час роботи в якості таймера, в кожному машинному циклі виконується інкремент вмісту таймера/лічильника з частотою $f_{РЕЗ}/12$, де $f_{РЕЗ}$ – частота тактового генератора, оскільки машинний цикл складається з дванадцяти періодів частоти синхронізації. Під час роботи в якості лічильнику вміст таймера/лічильника інкрементується за переходу з 1 в 0 зовнішнього сигналу, що подається на вхід лічильника зовнішніх подій T1 (T0) мікроконтролера. Максимальна частота зовнішнього вхідного сигналу складає $f_{РЕЗ}/24$.

**Режим 0 роботи таймерів/лічильників.**

Логіка роботи T/Cx в режимі 0 показана на рис. 6.1. Таймер/лічильник є тринадцятирозрядним лічильником, де послідовно з’єднані п’ятирозрядний регістр TLx і восьмирозрядний регістр THx. Залежно від значення розряду C/Tx регістра TMOD на вхід лічильника поступають зовнішні сигнали з входу Tx (лічильник) або сигнал $f_{РЕЗ}/12$ (таймер). Рахування розпочинається за встановлення біта TR регістра TCON. Управління рахуванням ззовні здійснюють за допомогою біту GATE регістра TMOD. При цьому рахування

дозволене за встановлення значення вхідного сигналу INT1 = 1 і заборонене за сигналом – INT0 = 0. Під час переповнювання T/Cx встановлюється ознака TFx.

**Режим 1 роботи таймерів/лічильників.**

Аналогічний режиму 0 (рис. 6.1) з тією лише різницею, що T/Cx є шістнадцятирозрядним лічильником, тобто регістр TLx – восьмирозрядний.

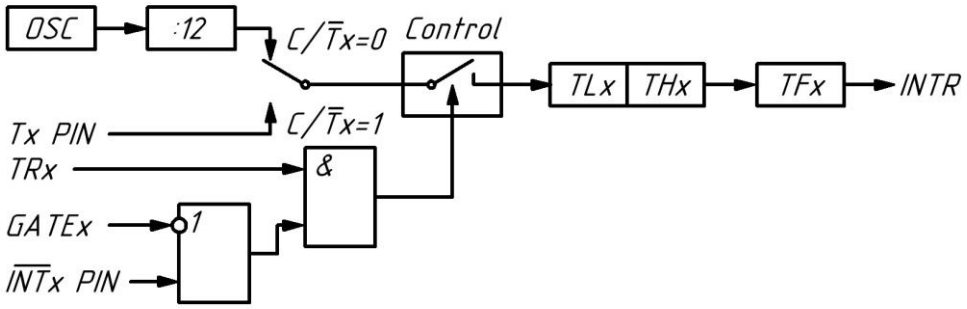



Рисунок 6.1 – Логіка роботи таймерів/лічильників T/C0 та T/C1 в режимах 0 і 1

**Режим 2 роботи таймерів/лічильників.**

Логіка роботи T/Cx в режимі 2 показаний на рис. 6.2. Таймер/лічильник в такому режимі є восьмирозрядним лічильником на основі регістру TLx. Під час кожного переповнювання регістру TLx відбувається завантаження вмісту регістру THx в регістр TLx. Вміст регістру THx завантажується програмно та в процесі рахування не змінюється.

Режими 0, 1, 2 ідентичні для обох таймерів/лічильників.

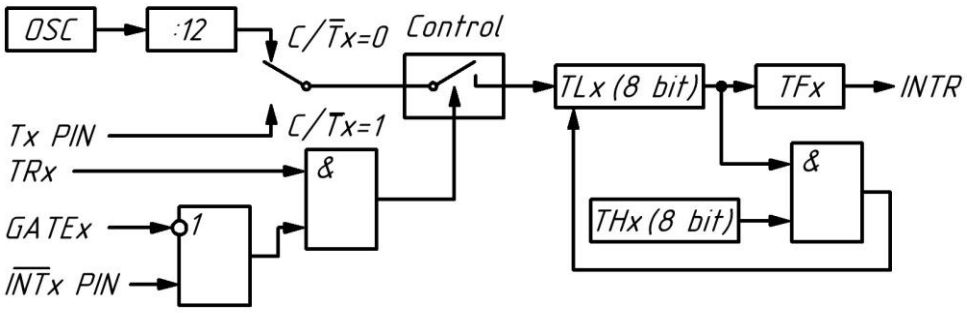



Рисунок 6.2 – Логіка роботи таймерів/лічильників T/C0 та T/C1 в режимі 2

**Режим 3 роботи таймерів/лічильників.**

У режимі 3 робота T/C0 та T/C1 відрізняється. Таймер/лічильник T/C0 являє собою два незалежних пристрої – на основі регістру TL0 може працювати і як таймер, і як лічильник (рис. 6.3); T/C1, на основі регістру TH0, працює тільки в режимі таймера. Для включення останнього використовується біт TR1, під час переповнювання встановлюється ознака TF1. Таймер/лічильник T/C1 включений постійно, його біт TR1 встановлений, і працює в режимах 0, 1 або 2, не виставляючи ознаки переповнювання. T/C1 може бути використаний в будь-

якому режимі, що не вимагає переривань. Наприклад, для роботи з послідовним інтерфейсом, який супроводжується сигналами переповнювання T/C1.

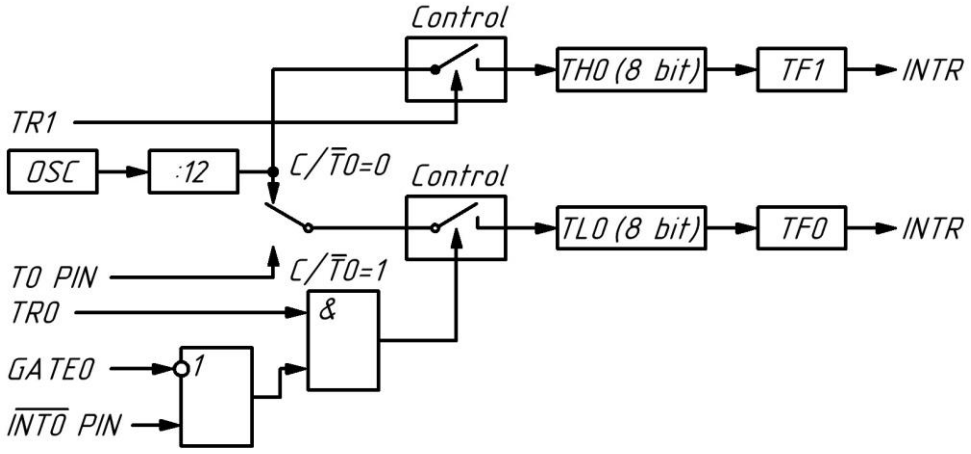



Рисунок 6.3 – Логіка роботи таймера/лічильника T/C0 в режимі 3

Для управління режимами роботи таймерів/лічильників та для організації їх взаємодії з системою переривань використовуються два регістри спеціальних функцій – регістр режимів таймера TMOD (табл. 6.1) і регістр управління/статусу таймера TCON (табл. 6.2).

Таблиця 6.1

<table>
<tr><th>Символ</th><th>Позиція</th><th>Ім’я та призначення</th></tr>
<tr><td>GATE</td><td>TMOD.7 T/C1<br>TMOD.3 T/C0</td><td>Управління блокуванням, за встановлення розряду GATEx = 1 дозволяється управляти таймером/лічильником x, якщо зовнішній управляючий сигнал INTx = 1 і біт управління TRx встановлений, інакше управління T/Cx дозволяється, тільки-но встановлюється біт управління TRx</td></tr>
<tr><td>C/T</td><td>TMOD.6 T/C1<br>TMOD.2 T/C0</td><td>Біт вибору режиму, якщо C/Tx = 0, визначає роботу в якості таймера від внутрішнього джерела сигналів синхронізації, якщо C/Tx = 1, працює як лічильник від зовнішніх сигналів на вході Tx</td></tr>
<tr><td>M1</td><td>TMOD.4 T/C1<br>TMOD.1 T/C0</td><td rowspan="2">Визначають режими 0-3 роботи таймера/лічильника
<table>
<tr><th>M0</th><th>M1</th><th>Режим роботи</th></tr>
<tr><td>0</td><td>0</td><td>TLx працює як 5-бітний дільник</td></tr>
<tr><td>0</td><td>1</td><td>16-бітний таймер/лічильник, THx та TLx вмикаються послідовно</td></tr>
<tr><td>1</td><td>0</td><td>8-бітний таймер/лічильник, що автоматично перезавантажується, THx зберігає значення, яке має бути перезавантажено в TLx щоразу після переповнення</td></tr>
<tr><td>1</td><td>1</td><td>TLO працює як 8-бітний таймер/лічильник, і його режим визначається керуючими бітами T/C0, TH0 працює тільки як 8 бітний таймер, і його режим визначається керуючими бітами T/C1</td></tr>
</table></td></tr>
<tr><td>M0</td><td>TMOD.3 T/C1<br>TMOD.0 T/C0</td></tr>
</table>

Таблиця 6.2

| Символ | Позиція | Ім’я та призначення |
|---|---|---|
| TF1 | TCON.7 | Ознака переповнювання таймер/лічильника 1, встановлюються програмно або апаратно під час переповнювання T/C1; якщо переривання від T/C1 дозволене, установка ознаки викличе переривання; ознаки скидаються програмно, або апаратно за обслуговування відповідного переривання |
| TR1 | TCON.6 | Біт управління таймер/лічильника 1, встановлюються та скидаються програмно для пуска і зупинки |

| | | |
|---|---|---|
| TF0 | TCON.5 | Ознака переповнювання таймер/лічильника 0, встановлюються програмно або апаратно під час переповнювання T/C0; якщо переривання від T/C0 дозволене, установка ознаки викличе переривання; ознаки скидаються програмно, або апаратно за обслуговування відповідного переривання |
| TR0 | TCON.4 | Біт управління таймер/лічильника 0, встановлюються та скидаються програмно для пуска і зупинки |

**Приклад 1.**

Налаштування таймера/лічильника T/C0 на роботу в якості таймера в режимі 1.

```
TIMER0_MODE_1:
        MOV TH0, #0FFh              ;Запис в TH0 числа FFh.
        MOV TL0, #0F0h              ;Запис в TL0 числа F0h.
        MOV TMOD,#00000001b         ;Налаштування T/C0 таймером, режим роботи 1.
        SETB TR0                    ;Дозвіл роботи T/C0.
M1:
        NOP
        LJMP M1                     ;Безумовний перехід на мітку M1.
END
```

**Приклад 2.**

Налаштування таймера/лічильника T/C0 на роботу в якості лічильника в режимі 1.

Для роботи за такою умовою потрібно налаштувати лінію 4 порту P3 на введення інформації.

```
COUNTER0_MODE_1:
        SETB P3.4                   ;Налаштування лінії P3.4 на введення інформації
        MOV TH0, #0FFh              ;Запис в TH0 числа FFh.
        MOV TL0, #0F0h              ;Запис в TL0 числа F0h.
        MOV TMOD,#00000101b         ;Налаштування T/C0 лічильником, режим роботи 1.
        SETB TR0                    ;Дозвіл роботи T/C0.
M1:
        NOP
        LJMP M1                     ;Безумовний перехід на мітку M1.

END
```

**Приклад 3.**

Налаштування таймера/лічильника T/C0 на роботу в якості таймера в режимі 2.

```
TIMER0_MODE_2:
        MOV TH0,#0F2h               ;Запис в TH0 числа F2h.
        MOV TL0,#0F0h               ;Запис в TL0 числа F0h.
        MOV TMOD,#00000010b         ;Налаштування T/C0 таймером, режим роботи 2.
```

```
        SETB TR0                ;Дозвіл роботи T/C0.
M1:
        NOP
        LJMP M1                 ;Безумовний перехід на мітку M1.
END
```

**Приклад 4.**

Налаштування таймерів/лічильників T/C0 та T/C1 на роботу в режимі 3.

```
TIMER_MODE_3:
        MOV TH0,#0F5h           ;Запис в TH0 числа F5h.
        MOV TL0,#0F0h           ;Запис в TL0 числа F0h.
        MOV TH1,#0FFh           ;Запис в TH1 числа FFh.
        MOV TL1, #0A0h          ;Запис в TL1 числа A0h.
        MOV TMOD,#00010011b     ;Налаштування T/C0 таймером, режим роботи 3,
                                ;T/C1 таймером, режим роботи 1.
        SETB TR0                ;Дозвіл роботи T/C0.
M1:
        NOP
        LJMP M1                 ;Безумовний перехід на мітку M1.
END
```

**Приклад 5.**

Для навчального стенда A2 розробити програму керування сервоприводом, яка дозволяє задавати кут повороту валу сервоприводу.

Для керування сервоприводом використовується сигнал, який має період 20 мс, тривалість від 1 мс до 2 мс (1,5 мс для нейтрального положення) та амплітуду 5 В (рис. Рисунок 6.4 ).

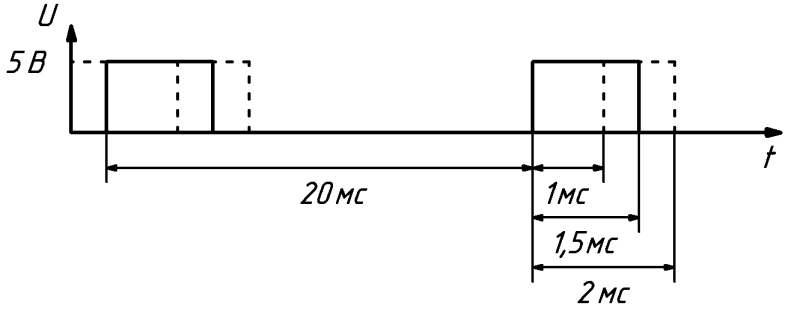



Рисунок 6.4 – Часова діаграма імпульсів керування сервоприводом

Для формування сигналу керування будуть використані два таймера/лічильника мікроконтролера, T/C1 – для формування періоду сигналу, а T/C0 – для формування тривалості імпульсу.

В навчальному стенді A2 для роботи мікроконтролера використовується кварцовий резонатор частотою 11,0592 МГц. Тоді таймери мікроконтролера

працюють на частоті $f_{РЕЗ}/12 = 1105920/12 = 921600$ Гц, а період сигналу тактової частоти – $1/92160 \approx 1{,}085$ мкс. Якщо використовувати шістнадцятирозрядні таймери, то максимальна часова затримка може бути $65536 \cdot 1{,}085 \cdot 10^{-6} \approx 0{,}071110656$ с. Цього достатньо для роботи сервоприводу, тому таймери/лічильники можуть тактуватись від внутрішнього тактового генератора та використовуватись в якості таймерів.

Період сигналу керування 20 мс, тому для завдання такого часового інтервалу потрібно $20 \cdot 10^{-3}/1{,}085 \cdot 10^{-6} \approx 18433$ імпульсів, або 4801h в шістнадцятковій системі числення. При переповненні таймера T/C1 комбінація його розрядів змінюються з FFFFh на 0000h та записується 1 в прапор ознаки переповнення. Тому, для завдання інтервалу в 4801h імпульсів потрібно, щоб таймер почав працювати не з комбінації в його регістрах 0000h, а з числа, яке розраховується наступним чином – до максимального числа таймера FFFFh додати 1 та відняти необхідну кількість імпульсів, $FFFFh + 1 - 4801h = B7FFh$.

Розрахувати число для запису в регістри таймера T/C0 для завдання тривалість імпульсу можна таким самим чином. Для тривалості 1 мс:

$$FFFFh + 1 - 0{,}001/1{,}085 \cdot 10^{-6} = FC67h.$$

Для тривалості 1,5 мс:

$$FFFFh + 1 - 0{,}0015/1{,}085 \cdot 10^{-6} = FA9Ah.$$

Для тривалості 2 мс:

$$FFFFh + 1 - 0{,}002/1{,}085 \cdot 10^{-6} = F8CDh.$$

Тобто число в регістрах таймера при змінах сигналу керування сервоприводом від 1 мс до 2 мс, буде змінюватись від FC67h до F8CDh.

Так як період тактових імпульсів має дробове число, то числа, які записуються в регістри таймерів повинні бути скориговані використовуючи виміри на осцилографі при налаштуванні пристрою.

Для роботи таймерів/лічильників не використовуються переривання, тому потрібно контролювати стан ознак переповнення таймерів/лічильників та своєчасно їх скидати.

Число, яке буде записуватись до регістрів таймеру знаходиться спочатку регістрах R6 та R7. Сигнал керування сервоприводом видається через лінію 7 порту P1.

```
SERVO:
    MOV R6,#0FAh          ;Запис числа, з якого почнеться відлік тривалості
    MOV R7,#9Ah           ;імпульсу до регістрів R6 та R7.
```

```
        MOV TMOD,#00010001b     ;Налаштування таймерів/лічильників T/C0 та T/C1
                                ;на роботу в режимі 1.
        MOV TH1,#0B7h           ;Запис в регістри T/C1 числа, яке задає період
        MOV TL1,#0FFh           ;повторення імпульсів керування.
        SETB TR1                ;Дозвіл роботи T/C1.
L1:
        JNB TF1,L2              ;Перевірка на 1 стану ознаки переповнення T/C1,
                                ;якщо 1, то переповнення, якщо ні, то перехід на L2.
        CLR TF1                 ;Скидання ознаки переповнення T/C1.
        SETB P1.7               ;Встановлення високого рівня на лінії P1.7 (+5 В).
        MOV TH1,#0B7h           ;Запис в регістри T/C1 числа, яке задає період
        MOV TL1,#0FFh           ;повторення імпульсів керування.
        MOV TH0,R6              ;Запис в регістри T/C0 числа з регістрів R6 та R7,
        MOV TL0,R7              ;завдання тривалості імпульсу керування.
        SETB TR0                ;Дозвіл роботи T/C0.
L2:
        JNB TF0,L3              ;Перевірка на 1 стану ознаки переповнення T/C0,
                                ;якщо 1, то переповнення, якщо ні, то перехід на L3.
        CLR TR0                 ;Якщо переповнення, то зупинка T/C0,
        CLR TF0                 ;Скидання ознаки переповнення T/C0.
        CLR P1.7                ;Встановлення низького рівня на лінії P1.7 (0 В).
L3:
        LJMP L1                 ;Безумовний перехід на L1.
END
```

**Приклад 6.**

Для навчального стенда А2 розробити програму, що підраховує кількість імпульсів, які надходять від периферійного пристрою, за інтервал часу 50 мс та виводить підраховану кількість на семисегментний індикатор.

Імпульси від зовнішнього пристрою подаються на лінію 4 порту P3.

Для завдання інтервалу часу для підрахування імпульсів буде використовуватись таймер/лічильник мікроконтролера T/C1, а таймер/лічильник T/C0 – для рахування кількості імпульсів.

Так як не використовується система переривань, то для фіксації переповнення таймерів/лічильників потрібно постійно перевіряти стан ознак переповнення.

Період інтервал часу вимірювання 50 мс, тому для його завдання потрібно $50 \cdot 10^{-3} / 1{,}085 \cdot 10^{-6} \approx 46083$ імпульсів, або B403h. Тоді таймера/лічильник повинен починати працювати з числа $\mathrm{FFFFh} + 1 - \mathrm{B403h} = \mathrm{4BFDh}$.

```
COUNTER:
```

```
        MOV 30h,#00h            ;Початкові налаштування вмісту комірок РПД.
        MOV 31h,#00h
        MOV 32h,#00h
        MOV 33h,#00h
        MOV 34h,#00h
        MOV 35h,#00h
        MOV 36h,#00h
        MOV 37h,#00h
        MOV R0,#37h             ;Запис в R0 адреси комірки 37h для адресації через R0.
        MOV R7,#08h             ;Запис до R7 числа 08h, налаштування кількості циклів.
        MOV SP,#60h             ;Перенесення верхівки стеку на комірку з адр. 60h.
        MOV DPTR,#TABLE         ;Запис до DPTR базової адреси таблиці TABLE.
        SETB P3.4               ;Налаштування лінії 4 порту P3 на введення.
        MOV TMOD,#00011101b     ;Налаштування T/C1 на роботу в режимі 1 в якості
                                ;таймера, а T/C0 в режимі 1 в якості лічильника.
        MOV TL1,#0FDh           ;Запис в регістри T/C1 числа 4BFDh, для
        MOV TH1,#4Bh            ;завдання часу вимірів 50 мс.
        ORL TCON,#01010000b     ;Дозвіл роботи T/C0 та T/C1.

MAIN1:                          ;Основна програма
        JNB TF1, MAIN2          ;Якщо ознака TF1 невстановлена, перехід на MAIN2.
        ANL TCON,#00101111b     ;Заборона роботи T/C0 і T/C1, скидання ознаки TF1.
        MOV TL1,#0FDh           ;Запис в регістри T/C1 числа 4BFDh, для
        MOV TH1,#4Bh            ;завдання часу вимірів 50 мс.
        MOV R5,TL0              ;Запис в регістр R5 вмісту TL0.
        MOV R4,TH0              ;Запис в регістр R4 вмісту TH0.
        MOV TL0,#0              ;Скидання вмісту регістрів TL0 та TH0.
        MOV TH0,#0
        ORL TCON,#01010000b     ;Дозвіл роботи T/C0 та T/C1.
MAIN2:
        LCALL UNPACKING         ;Виклик підпрограми UNPACKING.
        LCALL DIN_IND           ;Виклик підпрограми DIN_IND.
        LJMP MAIN1              ;Безумовний перехід на MAIN1.

DIN_IND:
        MOV P0,#00h             ;Запис в порт P0 числа 00h для гасіння всіх розрядів.
        DJNZ R7,DI1             ;Зменшення вмісту R7 та перевірка на 0 (кількість циклів).
        MOV R7,#08h             ;Запис до R7 числа 08h, налаштування кількості циклів.
        MOV R0,#37h             ;Запис до регістру R0 числа 37h, налаштування регістрової
                                ;адресації через регістр R0, для вибору чисел.
        ANL P2,#11100011b       ;Логічне «І» між вмістом порту P2 та числом 11100011b,
                                ;підключення першого розряду індикатора.
```

```
        AJMP DI2                ;Безумовний перехід на мітку DI2.
DI1:
        MOV A,P2                ;Запис вмісту регістру порту P2 до акумулятора.
        ADD A,#04h              ;Збільшення A на 04h.
        MOV P2,A                ;Запис вмісту A в порт P2, включається наступний розряд.
DI2:
        MOV A,@R0               ;Запис в A вмісту комірки РПД, адреса якої в R0.
        MOVC A,@A+DPTR          ;Запис в A символу для індикатора.
        MOV P0,A                ;Запис в порт P0 вмісту акумулятора A.
        DEC R0                  ;Зменшення на 1 вмісту регістру R0.
        RET                     ;Повернення із підпрограми DIN_IND.

DELAY:                          ;Підпрограма часової затримки.
        MOV R2,#250             ;Запис в регістр R2 числа кількості ітерацій.
        DJNZ R2,$               ;Зменшення на 1 вмісту регістру R2 поки не досягне 0.
        RET

UNPACKING:                      ;Підпрограма перетворення чисел в регістрах R4 та R5 для
                                ;можливості відображення на індикаторі.
        MOV R1,#37h             ;Запис в R1 адреси комірки 37h для адресації через R1.
        MOV A,R4                ;Запис до A числа із R4.
        ANL A,#0Fh              ;Логічне «І» над вмістом A та числом 0Fh, обнуління
                                ;старшої тетради.
        MOV @R1,A               ;Запис вмісту A в комірку РПД, адреса якої записана в R1.
        DEC R1                  ;Збільшення на 1 вмісту регістра R1.
        MOV A,R4                ;Запис до A числа із R4.
        SWAP A                  ;Зміна місцями старшої та молодшої тетрад числа в A.
        ANL A,#0Fh              ;Логічне «І» над вмістом A та числом 0Fh.
        MOV @R1,A               ;Запис вмісту A в комірку РПД, адреса якої записана в R1.
        DEC R1                  ;Збільшення на 1 вмісту регістра R1.
        MOV A,R5                ;Запис до A числа із R5.
        ANL A,#0Fh              ;Логічне «І» над вмістом A та числом 0Fh.
        MOV @R1,A               ;Запис вмісту A в комірку РПД, адреса якої записана в R1.
        DEC R1                  ;Збільшення на 1 вмісту регістра R1.
        MOV A,R5                ;Запис до A числа із R5.
        SWAP A                  ;Зміна місцями старшої та молодшої тетрад числа в A.
        ANL A,#0Fh              ;Логічне «І» над вмістом A та числом 0Fh.
        MOV @R1,A               ;Запис вмісту A в комірку РПД, адреса якої записана в R1.
        RET

TABLE:
        DB 3Fh,06h,5Bh,4Fh,66h,6Dh,7Dh,07h
```

```
        DB 7Fh,6Fh,77h,7Ch,39h,5Eh,79h,71h
        DB 80h
END
```

Хід роботи:

1. Ознайомитись із таймерами/лічильниками мікроконтролера сімейства MCS-51.
2. В інтегрованому середовищі MCU 8051 IDE та на навчальному стенді А2 ознайомитись із роботою прикладів.
3. За допомогою осцилографа зняти форму сигналів та привести відповідні креслення в звіті.
4. Для завдання, яке ставить викладач, розробити алгоритми та скласти програми.
5. Промоделювати роботу програм в середовищі MCU 8051 IDE та на навчальному стенді А2.
6. Зробити звіт з лабораторної роботи.

Контрольні питання:

1. Скільки таймерів/лічильників має мікроконтролер сімейства MCS-51?
2. Які режими роботи мають таймери/лічильники мікроконтролера сімейства MCS-51?
3. Чим відрізняються таймери від лічильників мікроконтролера?
4. Таймери/лічильники мікроконтролера сімейства MCS-51 можуть викликати переривання?
5. Поясніть роботи таймерів/лічильників в режимі 3.

## Лабораторна робота № 7

## Тема: Система переривань мікроконтролера

**Мета: Отримати навички в роботі із системою переривань мікроконтролерів та її програмуванні.**

Переривання в мікроконтролерах це механізм, який дозволяє мікроконтролеру реагувати на зовнішні події. Механізм переривань працює таким чином, що при настанні деякої події в мікроконтролері виникає сигнал, що змушує контролер перервати виконання поточної програми (виникло переривання). Після того, як виконання поточної програми перервано, мікроконтролер повинен перейти до виконання програмної процедури,

пов'язаної з цією подією (перериванням) – процедури обробки переривання. Однак, перш ніж перейти безпосередньо до процедури обробки переривання, мікроконтролер повинен виконати деякі попередні дії. Перш за все, щоб у майбутньому він зміг продовжити перервану програму, необхідно зберегти стан деяких внутрішніх регістрів (лічильника команд, словостану програми, внутрішніх регістрів і т. д.) на момент, що передує перериванню. Тобто, потрібно зберегти стан всіх тих ресурсів, які так чи інакше можуть бути змінені в процесі обробки переривання. Далі, якщо в системі є декілька можливих джерел переривань, мікроконтролер повинен визначити джерело запиту переривань. І потім перейти до виконання самої процедури переривань, конкретно для цього переривання. Після завершення обробки переривання процесор повинен відновити стан ресурсів, що відповідає перерваній програмі, після чого вона може бути продовжена. Для збереження всіх необхідних ресурсів, пошуку джерела переривання та переходу до процедури обробки переривання мікроконтролер повинен витратити певний час. Цей час називається прихованим часом переривання. Чим менший прихований час переривання, тим вища швидкість реакції системи на зовнішні події.

Система переривань – сукупність апаратних та програмних засобів, що реалізують механізм переривань у мікроконтролері. Схема системи переривань мікроконтролера MCS-51 представлена на рис. 7.1. Система переривань дозволяє автоматично реагувати на зовнішні і внутрішні події. Джерелами переривання в мікроконтролері MCS-51 є зовнішні сигнали INT0 і INT1, таймери/лічильники та послідовний порт.

Переривання від зовнішніх сигналів INT1 і INT0 можуть бути ініційовані за зрізом сигналу або за його низьким рівнем, що визначається бітами IT регістра TCON. Переривання викликає встановлення одного з бітів IE регістра TCON. Переривання від таймерів/лічильників викликаються встановленням бітів TF регістра TCON, під час переповнювання лічильників. Переривання від послідовного порту УАПП викликаються встановленням бітів переривання приймача RI або передавача TI регістра SCON.

Переривання від кожного джерела може бути дозволено (заборонено) встановленням (скиданням) відповідного біта в регістрі масок переривань IE (табл. 5.1).

Таблиця 5.1

| Символ | Позиція | Ім’я та призначення |
|---|---|---|
| EA | IE.7 | EA = 0 – забороняє всі переривання незалежно від стану бітів EX0, ET0, EX1, ET1, ES регістра IE; EA = 1 – переривання можуть бути дозволені, переривання |

| | | |
|---|---|---|
| | | дозволене якщо відповідний біт дорівнює одиниці, встановлюється і скидається програмно. |
| – | IE.6 | Не використовується. |
| – | IE.5 | Не використовується. |
| ES | IE.4 | Біт дозволу переривання, від приймача/передавача. Встановлюється/скидається програмою для дозволу/заборони переривань від прапорів TI або RI. |
| ET1 | IE.3 | Біт дозволу переривання від таймера/лічильника 1. Встановлюється/скидається програмою для дозволу/заборони переривань від таймера/лічильника 1. |
| EX1 | IE.2 | Біт дозволу зовнішнього переривання 1. Встановлюється/скидається програмою для дозволу/заборони зовнішнього переривання 1. |
| ET0 | IE.1 | Біт дозволу переривання від таймера/лічильника 0. Встановлюється/скидається програмою для дозволу/заборони переривань від таймера/лічильника 0. |
| EX0 | IE.0 | Біт дозволу зовнішнього переривання 0. Встановлюється/скидається програмою для дозволу/заборони зовнішнього переривання 0. |

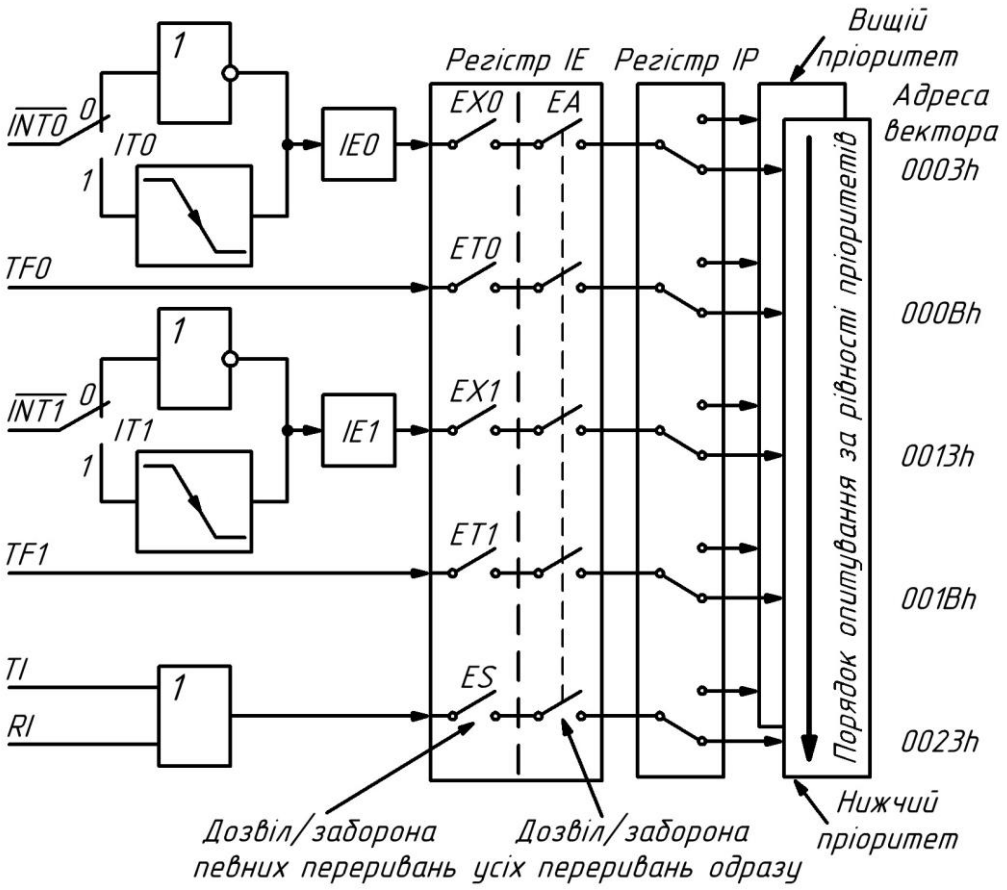



Рисунок 7.1 – Схема переривань мікроконтролера MCS-51

Система пріоритетів переривань є двоступінчастою. Кожному джерелу переривання може бути привласнений один з рівнів пріоритету – високий або низький, що визначається відповідним бітом регістра пріоритетів переривань IP (табл. 5.2). Наявність у відповідному розряді регістра IP одиниці встановлює для джерела високий рівень пріоритету, якщо до відповідного розряду регістра IP записано нуль – низького.

Таблиця 5.2

| Символ | Позиція | Ім’я та призначення |
|---|---|---|
| – | IP.7 | Не використовується. |
| – | IP.6 | Не використовується. |
| – | IP.5 | Не використовується. |
| PS | IP.4 | Біт пріоритету приймача. Встановлюється/скидається програмою для присвоєння перериванню від приймача вищого/нижчого пріоритету. |
| PT1 | IP.3 | Біт пріоритету таймера 1. Встановлюється/скидається програмою для присвоєння перериванню від таймера 1 вищого/нижчого пріоритету. |
| PX1 | IP.2 | Біт пріоритету зовнішнього переривання 1. Встановлюється/скидається програмою для присвоєння вищого/нижчого пріоритету зовнішньому перериванню INT1. |
| PT0 | IP.1 | Біт пріоритету таймера 0. Встановлюється/скидається програмою для присвоєння перериванню від таймера 0 вищого/нижчого пріоритету. |
| PX0 | IP.0 | Біт пріоритету зовнішнього переривання 0. Встановлюється/скидається програмою для присвоєння вищого/нижчого пріоритету зовнішньому перериванню INT0. |

Програма обробки переривання з низьким рівнем пріоритету може бути перервана запитом переривання з високим рівнем пріоритету, але не може бути перервана іншим запитом з низьким рівнем пріоритету. Програма обробки переривання з високим рівнем пріоритету не може бути перервана ніяким іншим запитом переривання. Під час одночасного надходження двох запитів спочатку буде обслуговуватись переривання з високим пріоритетом. Якщо надійшли декілька запитів з однаковим рівнем пріоритету, обробка їх проводиться в порядку, визначеному послідовністю опитувань ознак переривань.

Призначення розрядів регістра управління/статусу SCON, які відносяться до системи переривань представлені в табл. 5.3.

Таблиця 5.3

| Символ | Позиція | Ім’я та призначення |
|---|---|---|
| TI | SCON.1 | Біт ознаки переривання передавача; встановлюється апаратно за закінчення передачі байту; скидається програмно. |
| RI | SCON.0 | Біт ознаки переривання приймача; встановлюється апаратно за закінчення прийому байту; скидається програмно. |

Біти регістра управління/статусу таймера TCON призначені для управління перериваннями від зовнішніх сигналів INT1 і INT0 представлені в табл. 5.5.

Таблиця 5.3

| Символ | Позиція | Ім’я та призначення |
|---|---|---|
| IE1 | TCON.3 | Біт ознаки запиту (фронту) зовнішнього переривання 1, встановлюються апаратно за зрізом зовнішнього сигналу INT1 або програмно, скидається апаратно при обслуговуванні переривання, викликаного фронтом сигналу переривання. |
| IT1 | TCON.2 | Біт управління типом переривання 1 на вході INT1, встановлюється і скидається програмно для специфікації запиту INT1, якщо IT1 = 0, то дозволено переривання за низьким рівнем сигналу, якщо IT1 = 1, то дозволено переривання за зрізом сигналу. |

| | | |
|---|---|---|
| IE0 | TCON.1 | Біт ознаки запиту (фронту) зовнішнього переривання 0, встановлюються апаратно за зрізом зовнішнього сигналу INT0 або програмно, скидається апаратно при обслуговуванні переривання, викликаного фронтом сигналу переривання. |
| IT0 | TCON.0 | Біт управління типом переривання 0 на вході INT0, встановлюється і скидається програмно для специфікації запиту INT0, якщо IT0 = 0, то дозволено переривання за низьким рівнем сигналу, якщо IT0 = 1, то дозволено переривання за зрізом сигналу. |

**Приклад 1.**

Для навчального стенда А2 розробити програму, яка після приходу заднього фронту (спаду) запиту зовнішнього переривання по лінії INT0 запалює світлодіоди HL1-HL4 (рис. 4.4), а після приходу заднього фронту запиту зовнішнього переривання по лінії INT1 запалює світлодіоди HL5-HL8. Встановити пріоритет переривання INT0 вищім, а INT1 – нижчим.

Запити на переривання спрацьовують по задньому фронту, тому потрібно записати до бітів IT0 та IT1 управління/статусу таймера TCON одиничні значення.

Для встановлення вищого пріоритету перериванню INT0 потрібно записати в біт PX0 регістру пріоритетів переривань IP одиничне значення.

Для дозволу роботу зовнішніх переривань потрібно записати до бітів EX0 та EX1 одиничні значення і одиничне значення до біту дозволу всіх переривань EA регістру масок переривань IE.

```
ORG 0000h               ;Директива, яка задає асемблеру адресу комірки пам'яті, в
                        ;якій повинна бути розташована наступна команда.
    LJMP START          ;Безумовний перехід на мітку START.

ORG 0003h               ;Підпрограма обслуговування переривання INT0.
    MOV P2,#11110000b   ;Запис в порт P2 числа для запалювання HL1-HL4.
    RETI                ;Повернення з підпрограми обслуговування переривання.

ORG 0013h               ;Підпрограма обслуговування переривання INT1.
    MOV P2,#00001111b   ;Запис в порт P2 числа для запалювання HL5-HL8.
    RETI                ;Повернення з підпрограми обслуговування переривання.

ORG 0030h
START:                  ;Початок основної програми, блок налаштувань.
    SETB P3.2           ;Налагодження лінії P3.2 на введення інформації.
    SETB P3.3           ;Налагодження лінії P3.3 на введення інформації.
    SETB IT0            ;Налагодження переривання INT0 по спаду сигналу.
    SETB IT1            ;Налагодження переривання INT1 по спаду сигналу.
    SETB PX0            ;Налагодження вищого пріоритету для переривання INT0.
```

```
        CLR PX1             ;Налагодження нижчого пріоритету для переривання INT1.
        SETB EX0            ;Дозвіл роботи зовнішнього переривання INT0.
        SETB EX1            ;Дозвіл роботи зовнішнього переривання INT1.
        SETB EA             ;Дозвіл роботи всіх переривань.
MAIN1:                      ;Основна програма.
        NOP
        AJMP MAIN1          ;Безумовний перехід на мітку MAIN1, нескінченний цикл.
END
```

**Приклад 2.**

Для навчального стенда А2 розробити програму, що підраховує кількість імпульсів, які надходять від периферійного пристрою, за інтервал часу 50 мс та виводить підраховану кількість на семисегментний індикатор, використовуючи переривання від таймера/лічильника Т/С1 (умова аналогічна прикладу 4 лабораторної роботи № 6).

Розрахунки вмісту регістрів Т/С1 наведені в прикладі 4 лабораторної роботи № 6.

Імпульси від зовнішнього пристрою подаються на лінію 4 порту Р3.

Для завдання інтервалу часу для підрахування імпульсів буде використовуватись таймер/лічильник мікроконтролера Т/С1, а таймер/лічильник Т/С0 – для рахування кількості імпульсів.

```
ORG 0000h
        LJMP START          ;Безумовний перехід на мітку START.

ORG 001Bh                   ;Підпрограма обслуговування переривання від Т/С1.
        ANL TCON,#10101111b ;Заборона роботи Т/С0 і Т/С1.
        MOV TL1,#0FDh       ;Запис в регістри Т/С1 числа 4BFDh, для
        MOV TH1,#4Bh        ;завдання часу вимірів 50 мс.
        MOV R5,TL0          ;Запис в регістр R5 вмісту TL0.
        MOV R4,TH0          ;Запис в регістр R4 вмісту TH0.
        MOV TL0,#0          ;Скидання вмісту регістрів TL0 та TH0.
        MOV TH0,#0
        ORL TCON,#01010000b ;Дозвіл роботи Т/С0 та Т/С1.
        RETI                ;Повернення з підпрограми обслуговування переривання.

ORG 0050h
START:
        MOV 30h,#00h        ;Початкові налаштування вмісту комірок РПД.
        MOV 31h,#00h
        MOV 32h,#00h
        MOV 33h,#00h
        MOV 34h,#00h
```

```
        MOV 35h,#00h
        MOV 36h,#00h
        MOV 37h,#00h
        MOV R0,#37h             ;Запис в R0 адреси комірки 37h для адресації через R0.
        MOV R7,#08h             ;Запис до R7 числа 08h, налаштування кількості циклів.
        MOV SP,#60h             ;Перенесення верхівки стеку на комірку з адресою 60h.
        MOV DPTR,#TABLE         ;Запис до DPTR базової адреси таблиці TABLE.
        SETB P3.4               ;Налаштування лінії 4 порту P3 на введення.
        MOV TMOD,#00010101b     ;Налаштування T/C1 на роботу в режимі 1 в якості
                                ;таймера, а T/C0 в режимі 1 в якості лічильника.
        MOV TL1,#0FDh           ;Запис в регістри T/C1 числа 4BFDh, для
        MOV TH1,#4Bh            ;завдання часу вимірів 50 мс.
        SETB EX1                ;Дозвіл роботи переривання від T/C1.
        SETB EA                 ;Дозвіл роботи всіх переривань.
        ORL TCON,#01010000b     ;Дозвіл роботи T/C0 та T/C1.

MAIN1:                          ;Основна програма.
        LCALL UNPACKING         ;Виклик підпрограми перетворення чисел.
        LCALL DIN_IND           ;Виклик підпрограми динамічної індикації.
        LJMP MAIN1              ;Безумовний перехід на MAIN1.

DIN_IND:                        ;Підпрограма динамічної індикації.
        MOV P0,#00h             ;Запис в порт P0 числа 00h для гасіння всіх розрядів.
        DJNZ R7,DI1             ;Зменшення вмісту R7 та перевірка на 0 (кількість циклів).
        MOV R7,#08h             ;Запис до R7 числа 08h, налаштування кількості циклів.
        MOV R0,#37h             ;Запис до регістру R0 числа 37h, налаштування регістрової
                                ;адресації через регістр R0, для вибору чисел із РПД.
        ANL P2,#11100011b       ;Логічне «І» між вмістом порту P2 та числом 11100011b,
                                ;підключення першого розряду індикатора.
        AJMP DI2                ;Безумовний перехід на мітку DI2.
DI1:
        MOV A,P2                ;Запис вмісту регістру порту P2 до акумулятора.
        ADD A,#04h              ;Збільшення A на 04h.
        MOV P2,A                ;Запис вмісту A в порт P2, включається наступний розряд.
DI2:
        MOV A,@R0               ;Запис в A вмісту комірки РПД, адреса якої в R0.
        MOVC A,@A+DPTR          ;Запис в A коду символу для індикатора.
        MOV P0,A                ;Запис в порт P0 вмісту акумулятора A.
        DEC R0                  ;Зменшення на 1 вмісту регістру R0.
        LCALL DELAY             ;Виклик підпрограми часової затримки.
        RET                     ;Повернення із підпрограми DIN_IND.
```

```
DELAY:                  ;Підпрограма часової затримки.
        MOV R2,#250     ;Запис в регістр R2 числа кількості ітерацій.
        DJNZ R2,$       ;Зменшення на 1 вмісту регістру R2 поки не досягне 0.
        RET
UNPACKING:              ;Підпрограма перетворення чисел в регістрах R4 та R5 для
                        ;можливості відображення на індикаторі.
        MOV R1,#37h     ;Запис в R1 адреси комірки 37h для адресації через R1.
        MOV A,R4        ;Запис до A числа із R4.
        ANL A,#0Fh      ;Логічне «І» над вмістом A та числом 0Fh, запис нулів до
                        ;старшої тетради.
        MOV @R1,A       ;Запис вмісту A в комірку РПД, адреса якої записана в R1.
        DEC R1          ;Збільшення на 1 вмісту регістра R1.
        MOV A,R4        ;Запис до A числа із R4.
        SWAP A          ;Зміна місцями старшої та молодшої тетрад числа в A.
        ANL A,#0Fh      ;Логічне «І» над вмістом A та числом 0Fh.
        MOV @R1,A       ;Запис вмісту A в комірку РПД, адреса якої записана в R1.
        DEC R1          ;Збільшення на 1 вмісту регістра R1.
        MOV A,R5        ;Запис до A числа із R5.
        ANL A,#0Fh      ;Логічне «І» над вмістом A та числом 0Fh.
        MOV @R1,A       ;Запис вмісту A в комірку РПД, адреса якої записана в R1.
        DEC R1          ;Збільшення на 1 вмісту регістра R1.
        MOV A,R5        ;Запис до A числа із R5.
        SWAP A          ;Зміна місцями старшої та молодшої тетрад числа в A.
        ANL A,#0Fh      ;Логічне «І» над вмістом A та числом 0Fh.
        MOV @R1,A       ;Запис вмісту A в комірку РПД, адреса якої записана в R1.
        RET

TABLE:
        DB 3Fh,06h,5Bh,4Fh,66h,6Dh,7Dh,07h,7Fh,6Fh,77h,7Ch,39h,5Eh,79h,71h,80h
END
```

Як видно із прикладу 2, основна програма зменшилась, вона складається тільки із двох викликів підпрограм, а також в цьому прикладі вже не потрібно перевіряти стан ознаки переповнення таймера/лічильника T/C1 та скидати його після встановлення.

Хід роботи:

1. Ознайомитись із системою переривань мікроконтролера сімейства MCS-51.

2. В інтегрованому середовищі MCU 8051 IDE та на навчальному стенді A2 ознайомитись із роботою прикладів.

3. Для завдання, яке ставить викладач, розробити алгоритми та скласти

програми.

4. Промоделювати роботу програми в середовищі MCU 8051 IDE та на навчальному стенді А2.

5. Зробити звіт з лабораторної роботи.

Контрольні питання:

1. Скільки запитів та скільки рівнів має система переривань мікроконтролера сімейства MCS-51?

2. Які регістри використовуються для налагодження системи переривань мікроконтролера сімейства MCS-51?

3. Чим відрізняється обробка переривань за зрізом сигналу та за низьким рівнем сигналу?

4. Які використовуються адреси векторів переривань в мікроконтролерах сімейства MCS-51?

5. Які біти ознак переривань скидаються апаратно, а які програмно в мікроконтролерах сімейства MCS-51?

## Лабораторна робота № 8

## Тема: Послідовний порт мікроконтролера

**Мета: Отримати навички в роботі із послідовними портами мікроконтролерів та їх програмуванні.**

Блок послідовного інтерфейсу призначений для вводу і виводу послідовної інформації. До його склад входять універсальний асинхронний приймач/передавач (УАПП), буфер приймача/передавача SBUF, регістр управління/статусу УАПП SCON.

Універсальний асинхронний приймач/передавач, який називають також послідовним портом, призначений для обміну інформацією, поданою послідовним кодом, і може працювати в наступних чотирьох режимах.

**Режим 0 роботи УАПП.**

Інформація передається і приймається через зовнішній вхід приймача RXD. Через зовнішній вихід передавача видаються імпульси синхронізації, що стробують кожен біт інформації. Формат посилки – вісім біт, частота передачі і прийому:

$$f = f_{РЕЗ}/12, \qquad (8.1)$$

де $f_{РЕЗ}$ частота тактового генератора.

У режимах 1, 2 та 3 інформація приймається через вхід RXD, а передається через вихід TXD.

**Режим 1 роботи УАПП.**

Формат посилки – десять біт: старт-біт (0), вісім біт даних і стоп-біт (1). Частота прийому і передачі задається Т/С1. В цьому випадку необхідно заборонити переривання від Т/С1 і запустити його на рахування в режимах 0, 1 або 2. Найчастіше для цього використовується режим 2 таймера з автозавантаженням. При цьому частота обміну даними визначається як:

$$f = \frac{2^{SMOD} f_{РЕЗ}}{32 \cdot 12 \cdot (256 - TH1)}. \qquad (8.2)$$

**Режим 2 роботи УАПП.**

Формат посилки – одинадцять біт: старт-біт (0), вісім біт даних, програмований дев'ятий біт і стоп-біт (1). Дев'ятий біт приймає значення бита TB8 регістра SCON. Частота прийому і передачі задається програмно значенням бита SMOD регістра PCON дорівнює $f_{РЕЗ}/32$, за SMOD = 1 або $f_{РЕЗ}/64$ за SMOD = 0:

$$f = 2^{SMOD} f_{РЕЗ}/64. \qquad (8.3)$$

**Режим 3 роботи УАПП.**

Аналогічний режиму 2 за винятком того, що частота прийому і передачі задається Т/С1, як в режимі 1.

Налаштування таймера 1 для керування частотою роботи приймача/передавача наведено в табл. 7.1.

У всіх випадках передача ініціалізується інструкцією, в якій дані переміщуються SBUF. Прийом ініціалізується при виявленні перепаду з 1 на 0 на вході приймача. При цьому у режимі 0 цей перехід має супроводжуватися виконанням умов RI = 0 і REN = 1, а інших режимів – REN = 1.

Таблиця 7.1

| **Частота прийому/передачі (BAUD RATE)** | **Частота резонатора, МГц** | **Таймер/лічильник 1** | | | |
|---|---|---|---|---|---|
| | | **SMOD** | **C/T** | **Режим (MODE)** | **Число, яке перезавантажується** |
| Режим 0, макс: 1 МГц | 12 | X | X | X | X |
| Режим 2, макс: 375 КГц | 12 | 1 | X | X | X |
| Режим 1, 3: 62,2 КГц | 12 | 1 | 0 | 2 | 0FFh |
| 19,2 кГц | 11,059 | 1 | 0 | 2 | 0FDh |
| 9,6 кГц | 11,059 | 0 | 0 | 2 | 0FDh |
| 4,8 кГц | 11,059 | 0 | 0 | 2 | 0FAh |
| 2,4 кГц | 11,059 | 0 | 0 | 2 | 0F4h |
| 1,2 кГц | 11,059 | 0 | 0 | 2 | 0F4h |
| 137,5 Гц | 11,059 | 0 | 0 | 2 | 1Dh |

| | | | | | |
|---|---|---|---|---|---|
| 110 Гц | 6 | 0 | 0 | 2 | 72h |
| 110 Гц | 12 | 0 | 0 | 1 | 0FEEBh |

Управління роботою УАПП здійснюється за допомогою регістра SCON (табл. 7.2)

Таблиця 7.2

<table>
<tr><th>Символ</th><th>Позиція</th><th>Ім’я та призначення</th></tr>
<tr><td>SM0</td><td>SCON.7</td><td rowspan="2">Біти керування режимом роботи приймача/передавача, встановлюються та скидаються програмно.
<table>
<tr><th>SM0</th><th>SM1</th><th>Режим роботи</th></tr>
<tr><td>0</td><td>0</td><td>Зсувний регістр розширення вводу/виводу.</td></tr>
<tr><td>0</td><td>1</td><td>8 бітовий приймач/передавач, змінна швидкість передачі.</td></tr>
<tr><td>1</td><td>0</td><td>9 бітовий приймач/передавач. фіксована швидкість передачі.</td></tr>
<tr><td>1</td><td>1</td><td>9 бітовий приймач/передавач, змінна швидкість передачі.</td></tr>
</table></td></tr>
<tr><td>SM1</td><td>SCON.6</td></tr>
<tr><td>SM2</td><td>SCON.5</td><td>Біт управління режимом УАПП, встановлюється програмно для заборони прийому повідомлення, в якому дев’ятий біт має значення нуля.</td></tr>
<tr><td>REN</td><td>SCON.4</td><td>Біт дозволу прийому, встановлюється і скидається програмно для дозволу і заборони прийому даних.</td></tr>
<tr><td>TB8</td><td>SCON.3</td><td>Передача біта 8, дев'ятий біт даних, що передаються, в режимах 2, 3; встановлюється і скидається програмно.</td></tr>
<tr><td>RB8</td><td>SCON.2</td><td>Прийом біта 8, дев'ятий біт даних, що приймаються, в режимах 2, 3; у режимі 1 за встановлення SM2 = 0 є прийнятим стоп-бітом.</td></tr>
<tr><td>TI</td><td>SCON.1</td><td>Ознака переривання передавача; встановлюється апаратно за закінчення передачі байту; скидається програмно.</td></tr>
<tr><td>RI</td><td>SCON.0</td><td>Ознака переривання приймача; встановлюється апаратно за закінчення прийому байту; скидається програмно.</td></tr>
</table>

Розряди SM0 і SM1 регістра управління/статусу SCON визначають чотири режими роботи УАПП. За значення SM2 = 1 у режимах 2, 3 ознака RI регістра SCON не встановлюється, якщо прийнятий дев’ятий біт даних дорівнює нулю. Дев’ятий біт дозволяє вирішити задачу обміну інформацією між декількома мікроконтролерами, об'єднаними в локальну мережу за допомогою моноканалу. Наприклад, ведені (підлеглі) мікроконтролери встановлюють біти SM2 своїх регістрів SCON і продовжують виконувати програми. Головний мікроконтролер, якщо йому необхідно обмінятися інформацією з одним із ведених мікроконтролерів, посилає в моноканал байт-ідентифікатор абонента з одиничним дев’ятим бітом (встановлений біт TB8). Отримання такого байту викличе переривання ведених мікроконтролерів, які виконують підпрограму порівняння отриманого байту з кодом власної адреси. Мікроконтролер, який розпізнав адресу, скидає біт SM2 регістру SCON і готується до прийому даних. Решта мікроконтролерів не міняє значення цього біта і продовжує виконувати основну програму. Головний скидає біт TB8 і передає дані в моноканал. У режимі 1 біт SM2 використовується для контролю

істинності стоп-біта. Ознака RI не буде встановлена, якщо за встановлення SM2 = 1 не прийнятий стоп-біт, дорівнює одиниці.

Регістр управління потужністю PCON має тільки один біт SMOD, що управляє швидкістю передачі послідовного порту. Якщо зазначений біт встановлений, швидкість передачі подвоюється.

**Приклад 1.**

Скласти програму для циклічної передачі числа A7h через УАПП в режимі 0.

```
UART_1:
        CLR SM0             ;Вибір режиму 0 УАПП
        CLR SM1             ;SM0 = 0, SM1 = 0.
        SETB P3.0           ;Запис 1 в тригер ліній 0 та 1 порту P3, налаштування на
        SETB P3.1           ;роботу з альтернативними функціями, сигнали RXD TXD.
L0:
        MOV SBUF,#A7h       ;Запис числа A7h в SBUF для передачі.
        JNB TI,$            ;Очікування 1 в прапорі TI.
        CLR TI              ;Скидання прапора TI.
        LJMP L0
END
```

**Приклад 2.**

Скласти програму, яка дозволить циклічно приймати числа через УАПП в режимі 0 та записувати в комірку пам’яті з адресою 37h.

```
UART_2:
        CLR SM0             ;Вибір режиму 0 УАПП
        CLR SM1             ;SM0 = 0, SM1 = 0.
        SETB P3.0           ;Запис 1 в тригер ліній 0 та 1 порту P3, налаштування на
        SETB P3.1           ;роботу з альтернативними функціями, сигнали RXD TXD.
        SETB REN            ;Запис 1 до REN для дозволу прийому даних.
L0:
        SETB REN            ;Запис 1 до REN для дозволу прийому даних.
        JNB RI,$            ;Очікування 1 в прапорі RI прийому даних.
        MOV 37h,SBUF        ;Запис прийнятих даних в комірку із адресою в регістрі R0.
        CLR RI              ;Скидання прапора RI.
        LJMP L0             ;Безумовний перехід на мітку L0.
END
```

**Приклад 3.**

Скласти програму для навчального стенду А2 (рис. 8.1), яка дозволить в режимі роботи 1 УАПП на частоті 19,2 кГц приймати байт даних, який надходить із послідовного порту комп’ютера, відображати результати на лінійці світлодіодів, змінювати в отриманому числі місцями тетради та

відправляти результат по послідовному порту назад до комп'ютера.

Для налаштування частоти передачі порту використовуються данні із табл. 7.1. Таймер/лічильник налаштовується на роботу таймером в режимі 2 (з автоматичним перезавантаженням), в регістри таймеру записуються числа FDh.

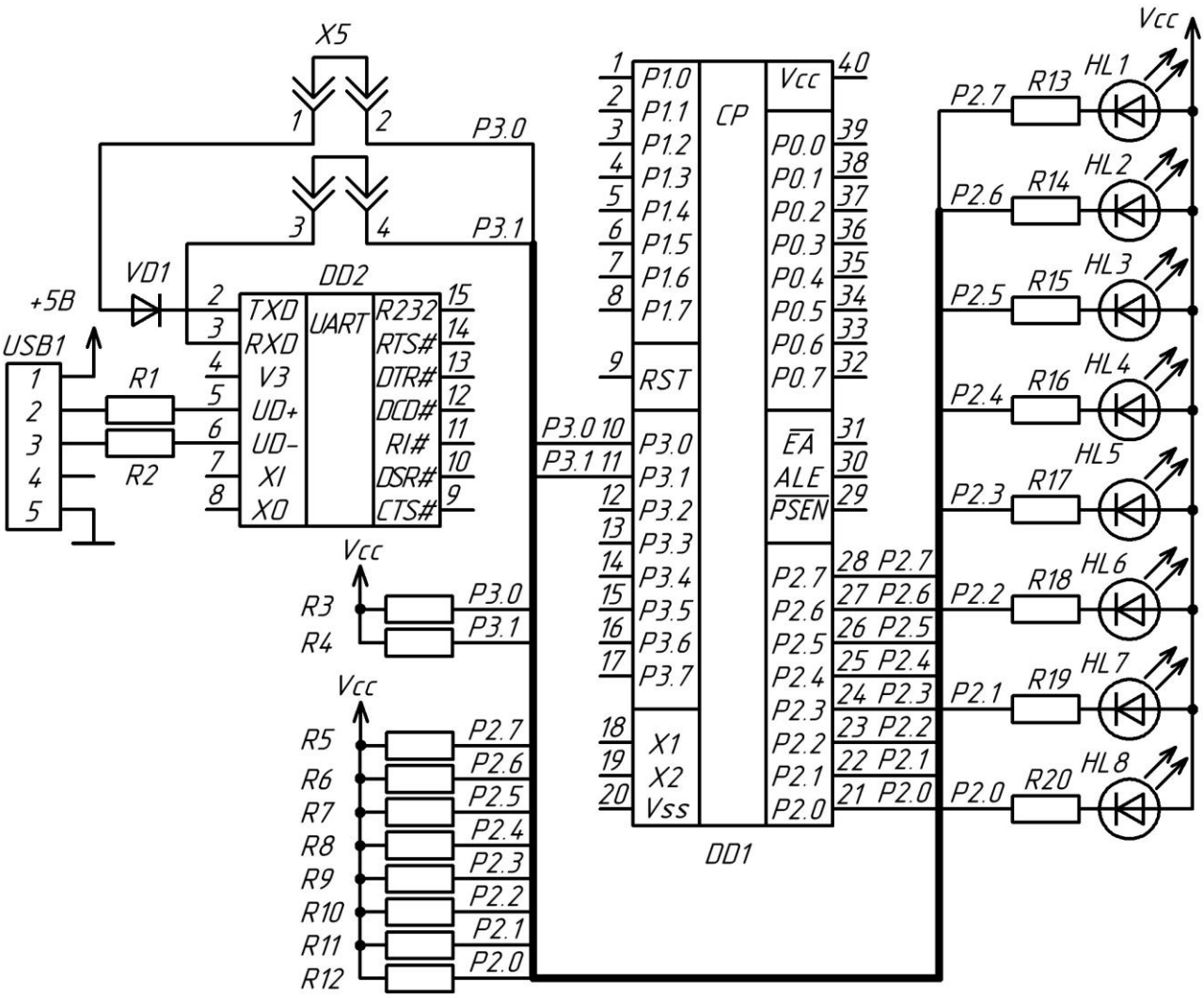



Рисунок 8.1 – Схема з'єднання перетворювача USB-UART CH340G до мікроконтролера у стенді А2

В біті SMOD регістра управління потужністю PCON встановлюється 1.

```
UART_3:
        MOV SCON,#01100000b     ;Вибір режиму 1 УАПП та дозвіл прийому.
        ANL PCON,#01111111b     ;Встановлюється 1 в біті SMOD.
        SETB P3.0               ;Налаштування ліній 0 та 1 порту P3 на роботу з
        SETB P3.1               ;альтернативними сигналами.
        MOV TH1,#0FDh           ;Запис числа FDh до регістру TH1.
        MOV TL1,#0FDh           ;Запис числа FDh до регістру TL1.
        MOV TMOD,#00100000b     ;Вибір режиму 1 таймера/лічильника T/C1.
        CLR ET1                 ;Заборона переривання від таймера/лічильника T/C1
```

```
        SETB TR1                ;Дозвіл роботи таймера/лічильника T/C1
L0:
        JNB RI, $               ;Перевірка значення прапору прийому байту УАПП, якщо
                                ;0, то мікроконтролер очікує передачу.
        MOV A,SBUF              ;Запис прийнятого байту до акумулятора.
        CLR RI                  ;Скидання прапору прийому байту RI.
        CPL A                   ;Інверсія вмісту акумулятора для відображення.
        MOV P2,A                ;Запис вмісту акумулятора до порту P2.
        CPL A                   ;Інверсія вмісту акумулятора для відновлення числа.
        SWAP A                  ;Зміна місцями старшої та молодшої тетрад числа в A.
        MOV SBUF,A              ;Запис вмісту A в SBUF для передачі в комп'ютер.
        JNB TI, $               ;Перевірка прапору передачі байту, якщо 0, то очікується
                                ;закінчення передачі.
        CLR TI                  ;Скидання прапору передачі байту TI.
        JMP L0                  ;Безумовний перехід на мітку L0.
END
```

Хід роботи:

1. Ознайомитись із послідовним портом мікроконтролера сімейства MCS-51.

2. В інтегрованому середовищі MCU 8051 IDE та на навчальному стенді А2 ознайомитись із роботою прикладів.

3. За допомогою осцилографа зняти форму сигналів та привести відповідні креслення в звіті.

4. Для завдання, яке ставить викладач, розробити алгоритми та скласти програми.

5. Промоделювати роботу програм в середовищі MCU 8051 IDE та на навчальному стенді А2.

6. Зробити звіт з лабораторної роботи.

Контрольні питання:

1. Які існують режими роботи універсального асинхронного приймача/передавача мікроконтролера сімейства MCS-51?

2. На якій частоті виконується передавання інформації в режимі 0 універсального асинхронного приймача/передавача?

3. Чим відрізняються режими роботи 2 та 3 універсального асинхронного приймача/передавача?

4. Які регістри спеціальних функцій використовуються для налаштування роботи універсального асинхронного приймача/передавача?

5. Чи має можливість універсальний асинхронний приймач/передавач викликати апаратні переривання?

## Лабораторна робота № 9

### Тема: Обмін інформацією мікроконтролера з зовнішніми пристроями

**Мета: Отримати навички в програмуванні роботи мікроконтролера з зовнішніми пристроями.**

Швидкості передавання та прийняття даних мікропроцесорною системою і зовнішнього пристрою можуть відрізнятись. Перед початком читання нової порції даних з порту вводу, необхідно переконатися, що зовнішній пристрій готовий надати або вже надав ці дані. Інакше операція зведеться до вводу недійсних чи старих даних. Аналогічна ситуація складається і під час виводу даних, коли потрібна перевірка готовності зовнішнього пристрою до прийому нових даних. В іншому випадку недозволений вивід з боку мікропроцесорної системи може призвести до втрати нового або попереднього елемента даних.

Типове вирішення проблеми синхронізації обміну полягає в використанні протоколу обміном службовою інформацією з квітуванням, при якому супроводжується операція умовного вводу/виводу спеціальним сигналом готовності RDY (Ready), що генерується зовнішнім пристроєм та служить для інформування мікропроцесорної системи про готовність зовнішнього пристрою прийняти чи передати нові дані (рис. 9.1).

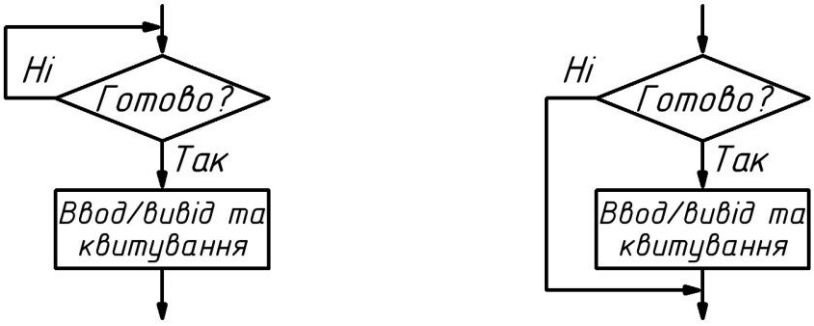

а) б)

Рисунок 9.1 – Часові діаграми умовного обміну: а) ввод; б) вивод

Після завершення операції вводу/виводу сигнал готовності RDY має бути знято і поставлено заново тільки при новій готовності до обміну. З цією метою зовнішній пристрій слід проінформувати про закінчення операції, для чого використовується включений в одне з керуючих слів сигнал підтвердження АСК (Acknowledgement).

Існують два типи умовного вводу/виводу із заняттям циклу (рис. 9.2, а) та суміщений (рис. 9.2, б). У першому випадку мікропроцесорна система очікує готовність, витрачаючи це весь машинний час. У другому випадку, якщо

зовнішній пристрій не готовий до обміну, мікропроцесорна система повертається до основного завдання без виконання операції вводу/виводу. Однак вона може знову перевірити готовність зовнішнього пристрою до обміну та за вдалого результату виконати його.




а) б)

Рисунок 9.2 – Алгоритми умовного вводу/виводу: а) із заняттям циклу; б) суміщеного

**Приклад 1.**

Скласти програми для реалізації програмно-керованого обміну інформацією між мікроконтролерами з квітуванням із заняттям циклу. Схема з'єднання мікроконтролерів зображена на рис. 9.3.

Мікроконтролер DD1 передає масив чисел Xi довжиною по 8 біт, які знаходяться в резидентній пам'яті програм починаючи з адреси 50h, до мікроконтролера DD2. Максимальна можлива кількість елементів масиву 20, кінець масиву позначається елементом FFh.

Мікроконтролер DD2 повинен підрахувати кількість елементів прийнятого масиву чисел та запалити відповідне число на світлодіодах HL1-HL8.

Програма мікроконтролера DD1.

Масив чисел, який потрібно передавати розташований в комірках РПП, починаючи із адреси 50h. Читати цифри з РПП можливо тільки в акумулятор за допомогою команди MOVC A,@A+DPTR.

```
ORG 0000h
DD1:
    CLR P3.0            ;Встановлення 0 на лінії 0 порту P3.
    SETB P3.1           ;Налаштування на лінії 1 порту P3 на введення інформації.
    MOV P2,#00h         ;Встановлення 0 на лініях порту P2.
    MOV DPTR,#TABLE     ; Запис до регістру DPTR базової адреси таблиці TABLE.
    MOV R0,#00h         ;Запис до регістру R0 зсуву відносно базової адреси
                        ;таблиці TABLE.
```

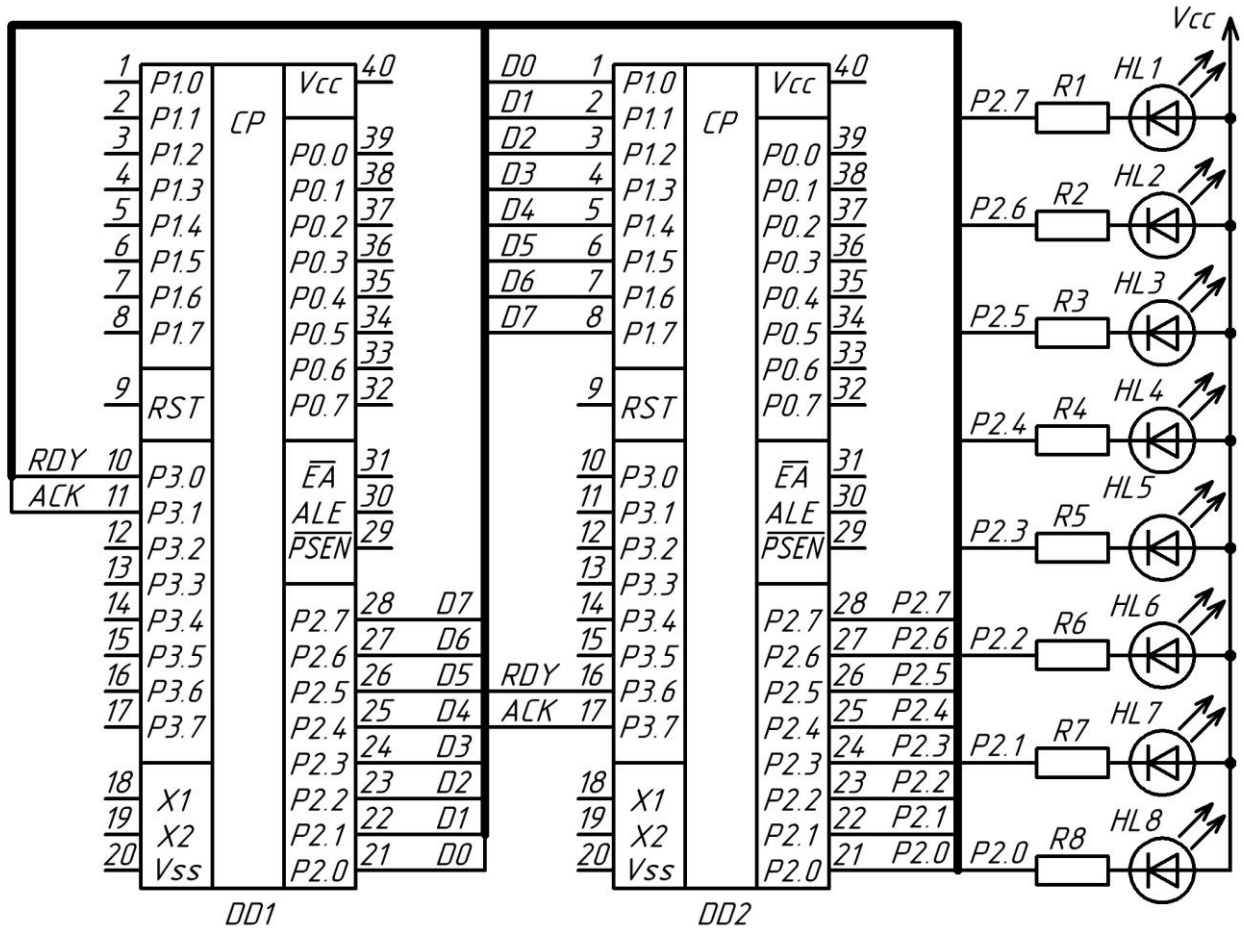



Рисунок 9.3 – Схема з’єднання мікроконтролерів

```
L1:
        JB P3.1,L1          ;Перевірка готовності DD2 приймати інформацію.
        MOV A,R0            ;Запис до A зсуву відносно базової адреси таблиці TABLE.
        MOVC A,@A+DPTR      ;Запис до A наступного числа з таблиці TABLE.
        MOV P2,A            ;Запис в порт P2 вмісту A.
        INC R0              ;Збільшення вмісту R0 на 1 для адресації наступного
                            ;числа в комірці РПП.
        SETB P3.0           ;Встановлення 1 на лінії 0 порту P3.
L2:
        JNB P3.1,L2         ;Перевірка того, що DD2 прийняв інформацію.
        CLR P3.0            ;Встановлення 0 на лінії 0 порту P3.
        CJNE A,#FFh,L3      ;Порівняння вмісту A із FFh, перехід на L3 якщо нерівно.
        SJMP L4             ;Закінчення передавання, безумовний перехід на L4.
L3:
        SJMP L1             ;Безумовний перехід на L1.
L4:
        NOP
        SJMP L4             ;Безумовний перехід на L4.
ORG 0050h
```

```
TABLE:
        DB 3Fh,06h,5Bh,4Fh,66h,6Dh,7Dh,07h,7Fh,0FFh
END
```

Програма мікроконтролера DD2.

Масив чисел, який приймає мікроконтролер буде розташовуватись в комірках РПД, починаючи із адреси 40h.

```
DD2:
        SETB P3.6           ;Налаштування на лінії 6 порту P3 на введення інформації.
        CLR P3.7            ;Встановлення 0 на лінії 7 порту P3.
        MOV P1,#0FFh        ;Налаштування ліній порту P1 на введення інформації.
        MOV P2,#0FFh        ;Всі світлодіоди на лініях порту P2 погашені.
        MOV R1,#40h         ;Запис до R1 адреси для першого елементу масиву.
        MOV R2,#00h         ;Початкове налаштування R2, в якому буде виконуватись
                            ;підрахунок кількості прийнятих елементів масиву.
L1:
        JNB P3.6,L1         ;Перевірка готовності DD1 передавати інформацію.
        MOV A,P1            ;Запис числа, що передається із порту P2 в A.
        SETB P3.7           ;Встановлення 1 на лінії 7 порту P3.
L2:
        JB P3.6,L2          ;Перевірка того, що DD1 закінчив передавання інформації.
        CLR P3.7            ;Встановлення 0 на лінії 7 порту P3.
        CJNE A,#FFh,L3      ;Порівняння отриманого числа із FFh, перехід на L3 якщо
                            ;нерівно.
        SJMP L4             ;Безумовний перехід на L4.
L3:
        MOV @R1,A           ;Запис отриманого числа в комірку РПД, адреса якої
                            ;знаходиться в регістрі R1.
        INC R1              ;Збільшення на 1 вмісту регістру R1, для можливості
                            ;адресації наступного елемента масиву
        INC R2              ;Збільшення на 1 вмісту регістру R2.
        SJMP L1             ;Безумовний перехід на L1.
L4:
        MOV A,R2            ;Запис в A вмісту регістру R2, кількості елементів.
        CPL A               ;Інвертування вмісту A для запалювання світлодіодів.
        MOV P2,A            ;Запис вмісту A в порт P2, запалювання світлодіодів.
L5:
        NOP
        SJMP L5             ;Безумовний перехід на L5
END
```

Хід роботи:

1. Ознайомитись зі способами обміну інформацією мікроконтролера із зовнішніми пристроями та програмами їх обслуговування.

2. В інтегрованому середовищі MCU 8051 IDE та на навчальному стенді А2 ознайомитись із роботою прикладів.

3. За допомогою осцилографа зняти форму сигналів та привести відповідні креслення в звіті.

4. Для завдання, яке ставить викладач, розробити алгоритми та скласти програми.

5. Промоделювати роботу програм в середовищі MCU 8051 IDE та на навчальному стенді А2.

6. Зробити звіт з лабораторної роботи.

Контрольні питання:

1. Які типи програмно-керованого обміну існують?

2. Чим відрізняється програмно-керований обмін від обміну по перериванню?

3. Який обмін інформацією мікропроцесорної системи з зовнішнім пристроєм називається програмно-керованим?

4. Які існують алгоритми умовного вводу/виводу?

5. Поясніть алгоритм роботи мікропроцесорної системи при умовному вводі/виводі?

## РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Колонтаєвський Ю. П., Сосков А. Г. Промислова електроніка : підручник / ред. А. Г. Сосков. Київ : Каравела, 2023. 536 с.

2. Матвієнко М. П. Промислова електроніка : підручник. Київ : Ліра-К, 2021. 423 с.

3. Дудикевич В. Б., Кеньо Г. В., Петрович І. В. Електроніка та мікросхемотехніка : навч. посіб. Львів : НУ «Львів. Політехніка», 2010. Ч. 1 : Електроніка. 204 с.

4. Колонтаєвський Ю. П., Сосков А. Г. Електроніка і мікросхемотехніка : підручник / ред. А. Г. Сосков. 2-ге вид. Київ : Каравела, 2009. 416 с.

5. Мілих В. І., Шавьолкін О. О. Електротехніка, електроніка та мікропроцесорна техніка : підручник / ред. В. І. Мілих. 2-ге вид. Київ : Каравела, 2008. 688 с.

6. Островерхов М. Я., Сенько В. І., Чибеліс В. І. Промислова електроніка. Напівпровідникові перетворювачі змінної напруги в постійну : навч. посіб. Київ : Ліра-К, 2021. 341 с.

7. Матвієнко М. П. Проектування цифрових пристроїв : підручник. Київ : Ліра-К, 2024. 362 с.

8. Мікропроцесорна техніка : підручник / Ю. І. Якименко та ін. ; ред. Т. О. Терещенко. 2-ге вид. перероб. та допов. Київ : Політехніка, Кондор, 2018. 440 с.

9. Огородник К. В., Книш Б. П. Мікропроцесорна техніка : навч. посіб. Вінниця : ВНТУ, 2018. 106 с.

10. Мікропроцесорна техніка : навч. посіб. / В. В. Ткачов та ін. Дніпропетровськ : Нац. гірн. ун-т, 2012. 188 с.

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт
з дисципліни «Електроніка та мікропроцесорна техніка»
для здобувачів першого (бакалаврського) рівня вищої освіти
спеціальності 141 Електроенергетика, електротехніка та електромеханіка
освітньо-професійної програми
Електроенергетика, електротехніка та електромеханіка
всіх форм навчання
Частина 2

Укладачі: Федотов В. О., канд. техн. наук, доцент;
Сьомочкин А. Б., канд. техн. наук, доцент;
Осадчук Ю. Г., канд. техн. наук, доцент.

Реєстраційний № _____

Підписано до друку «___» _____ 2025 р.

Формат А5
Обсяг 73 стор.

Видавничий центр КНУ, вул. Віталія Матусевича, 11,
м. Кривий Ріг