Міністерство освіти і науки України

Національний університет водного господарства та природокористування

С. В. Василець, К. С. Василець, А. В. Килимчук

МІКРОПРОЦЕСОРНА ТЕХНІКА В СИСТЕМАХ ОБЛІКУ ЕНЕРГІЇ ТА РЕЛЕЙНОМУ ЗАХИСТІ

Навчальний посібник

Рівне 2023

УДК 621.311:004.031.6
В19

Рецензенти:

Квасніков В. П., доктор технічних наук, професор, Заслужений метролог України, завідувач кафедри комп'ютеризованих електротехнічних систем та технологій Національного авіаційного університету, м. Київ;

Мартинюк П. М., доктор технічних наук, професор, директор Навчально-наукового інституту автоматики, кібернетики та обчислювальної техніки Національного університету водного господарства та природокористування, м. Рівне.

Рекомендовано вченою радою Національного університету водного господарства та природокористування.
Протокол № 8 від 30 серпня 2023 р.

Василець С. В., Василець К. С., Килимчук А. В.

В19 Мікропроцесорна техніка в системах обліку енергії та релейному захисті : навч. посіб. [Електронне видання]. – Рівне : НУВГП, 2023. – 204 с.

ISBN 978-966-327-569-7

У навчальному посібнику викладено основи пробудови мікропроцесорних пристроїв, що використовуються в системах обліку енергії та релейному захисті: проаналізовано основи побудови апаратної частини, розглянуто принципи розроблення алгоритмів функціонування цифрових вимірювальних органів.

Посібник призначено для здобувачів вищої освіти першого (бакалаврського) рівня в галузі знань 14 «Електрична інженерія» за спеціальністю 141 «Електроенергетика, електротехніка та електромеханіка». Також може бути корисним під час самостійного освоєння мікропроцесорної техніки.

УДК 621.311:004.031.6

ISBN 978-966-327-569-7

ЗМІСТ

ВСТУП

Метою освітньої компоненти «Мікропроцесорна техніка в системах обліку енергії та релейному захисті» є формування у здобувачів вищої освіти компетентностей із використання мікропроцесорних пристроїв в релейному захисті систем електропостачання, електричних систем та мереж, а також для комерційного та технічного обліку електроенергії.

Завданнями освітньої компоненти є набуття знань щодо двійкової та шістнадцяткової систем числення, основ кодування інформації, вивчення структури мікропроцесора, мікроконтролера, організації мікропроцесорної системи, оволодіння програмуванням мікроконтролерів поширених типів, набуття знань щодо основних алгоритмів функціонування релейного захисту та засобів обліку електроенергії, уміння їх реалізовувати засобами мікропроцесорної техніки.

В результаті вивчення освітнього компоненту здобувач вищої освіти має оволодіти наступними компетентностями: здатність застосовувати знання у практичних ситуаціях; здатність до пошуку, оброблення та аналізу інформації з різних джерел; здатність оперативно вживати ефективні заходи в умовах надзвичайних (аварійних) ситуацій в електроенергетичних та електромеханічних системах; здатність комплексно аналізувати процеси генерації електричної енергії традиційними та відновлюваними джерелами, перетворення, розподілу та споживання електроенергії, з урахуванням засобів мікропроцесорного керування, в тому числі – електропостачання об’єктів водного господарства та технічних засобів природокористування.

Програмні результати навчання: застосовувати прикладне програмне забезпечення, мікроконтролери та мікропроцесорну техніку для вирішення практичних проблем у професійній діяльності; розуміти основні принципи і завдання технічної та екологічної безпеки об’єктів електротехніки та електромеханіки, враховувати їх при прийнятті рішень; розв’язувати складні спеціалізовані задачі з проектування і технічного обслуговування електромеханічних систем, електроустаткування електричних станцій, підстанцій, систем та мереж; застосовувати знання щодо нерозривності процесів генерації електроенергії традиційними та відновлюваними джерелами, перетворення, розподілу та споживання електроенергії під час побудови пристроїв та систем мікропроцесорного керування електроенергетичними об’єктами.

Навчальний посібник складається з двох розділів. Розділ 1 «Типові вузли мікропроцесорних пристроїв» розкриває улаштування апаратної частини мікропроцесорних пристроїв як елементів пристроїв релейного захисту та обліку електроенергії. В розділі 2 «Вимірювальні органи цифрового релейного захисту» розкриваються підходи до програмного оброблення миттєвих значень контрольованих параметрів електромережі (напруг та струмів) та розглядають способи побудови органів цифрового релейного захисту.

РОЗДІЛ 1. ТИПОВІ ВУЗЛИ МІКРОПРОЦЕСОРНИХ ПРИСТРОЇВ

1.1. Терміни та визначення. Системи числення. Алгебра логіки. Представлення символьних даних

Терміни та визначення. Системи числення. Алгебра логіки. Представлення символьних даних. Історія розвитку мікропроцесорних пристроїв. Основні терміни і визначення, що використовуються в мікропроцесорній техніці. Розвиток інструментів програмування. Представлення даних. Десяткова, двійкова і шістнадцяткова системи числення. Логічні операції. Типові логічні елементи. Представлення символьних даних. Таблиці ASCII, UNICODE.

1.1.1. Історія розвитку мікропроцесорних пристроїв

В V ст. до н.е. у Вавилоні було винайдено перший механічний рахунковий пристрій, що називався абак. Схожий пристрій використовувався у Римі, рис. 1.1. Загальний принцип розрахунку – значення кожного регістра відповідає кількості камінців або кісточок у визначеній смузі – був у широкому вжитку до недавнього часу у бухгалтерських рахівницях.

Рис. 1.1. Реконструкція римського абака

Перший механічний апарат, здатний здійснювати арифметичні обчислення, був сконструйований математиком Блезом Паскалем, рис. 1.2.

Рис. 1.2. Машина Паскаля «Паскаліна»

Обчислювальний пристрій Паскаля (машина «Паскаліна») використовував систему механічних редукторів. Такий принцип в наш час застосовується у одометрах автомобіля. «Паскаліна» була розміщена у скриньці. Для введення вихідних даних використовувалися коліщатка, які повертали у необхідну позицію. Кожна з коліщаток відповідало десятковому розряду числа. Після повного оберту, після цифри 9, коліщатко набувало значення 0, при цьому сусіднє коліщатко збільшувало значення на 1. Перші машини мали 5 зубчастих коліс, пізніше їх кількість збільшилася до 6 або 8. Це дозволяло працювати з числами до 9999999. Для відображення відповіді слугували віконечка у верхній частині передньої панелі. Машина дозволяла виконувати додавання. Множення виконувалося шляхом багаторазового додавання. Для віднімання використовувався метод доповнення до 9, для чого передбачене додаткове віконце.

Ідеї Паскаля були удосконалені Чарльзом Беббіджем в проекті різницевої машини (Difference Engine), рис. 1.3. Такий механічний апарат призначався для автоматизації обчислень за допомогою апроксимації функцій поліномами і обчислення кінцевих різниць. Машина могла оперувати логарифмами та тригонометричними функціями за умови їх подання поліномами. При житті винахідника машина повністю так і не була побудована.

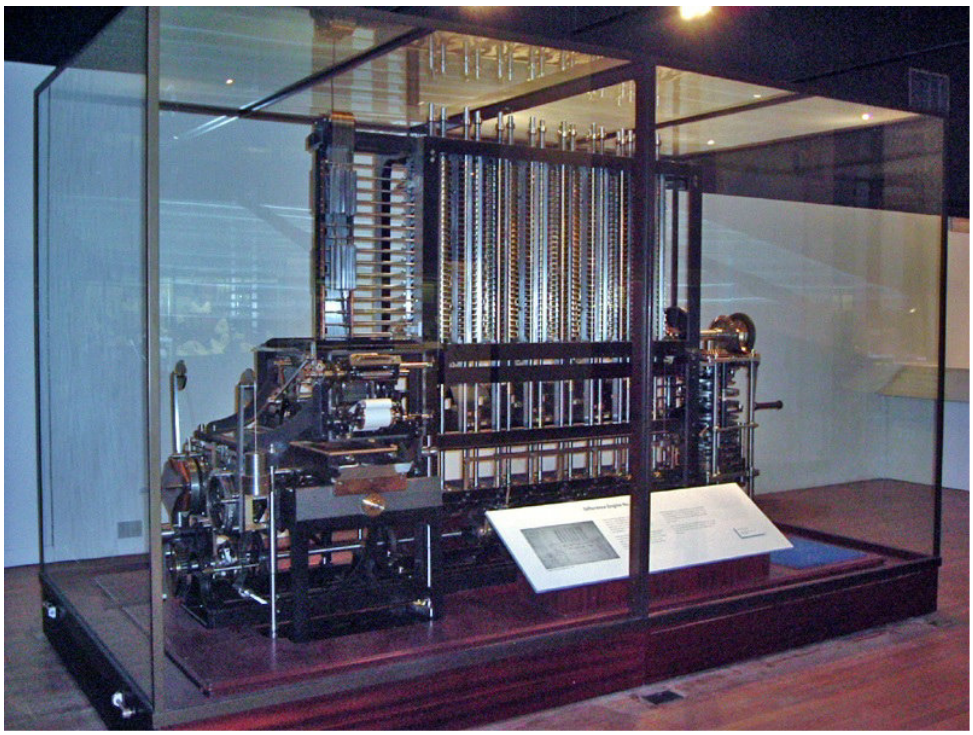


Рис. 1.3. Реконструкція лічильної машини Беббіджа

Механічна обчислювальна машина, обладнана електроприводом, була винайдена Германом Холлерітом у 1889 р., рис. 1.4. Машина призначена для оброблення та впорядкування інформації, що задана на перфокартах. Винахід Холлеріта зацікавив уряд США, йому доручили виконати оброблення даних перепису населення 1890 р. за допомогою табулятора. У 1896 р. Холлеріт започатковує компанія з розроблення обчислювальних машин з використанням

перфокарт. Після об'єднання компанії Холлеріта з іншими компаніями була створена корпорація IBM (International Business Machines), яка успішно функціонує до сьогоднішнього дня.

Рис. 1.4. Табулятор Холлеріта

Перша у світі електронна обчислювальна машина була розроблена німецьким винахідником Конрадом Цузе у 1941 році. Це був комп’ютер Z3 на основі електромеханічних реле, що працював на тактовій частоті 5,33 Гц, рис. 1.5. Він застосовувався німецькою армією для розроблення ракет та літаків під час Другої світової війни.

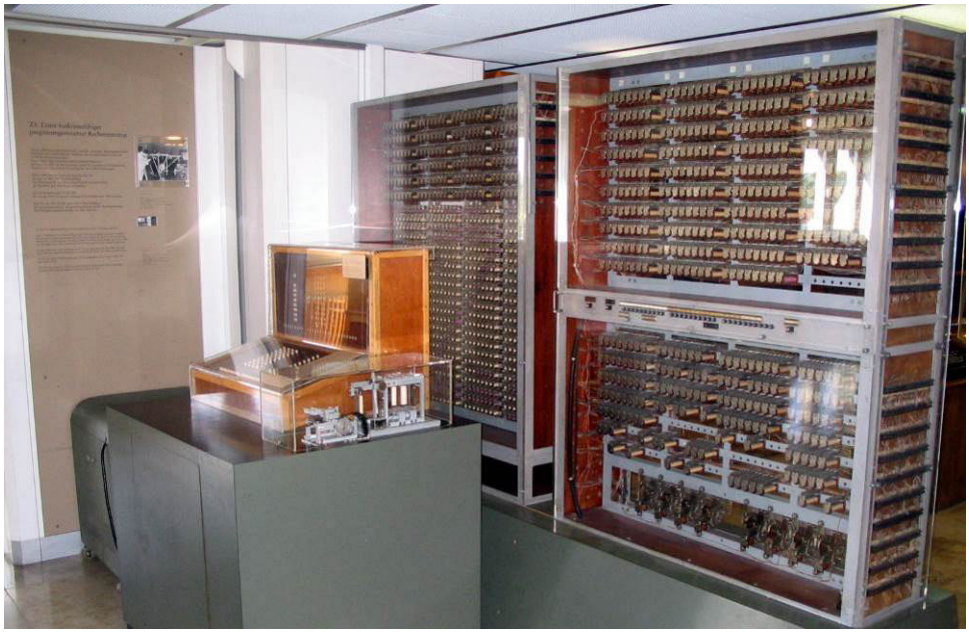


Рис. 1.5. Реконструкція Z3 у Німецькому музеї м. Мюнхен

Алан Тюрінг з групою винахідників у 1943 р. розробив перший комп'ютер на електронних лампах «Colossus», рис. 1.6. Він використовувався англійцями для розшифрування секретних повідомлень.

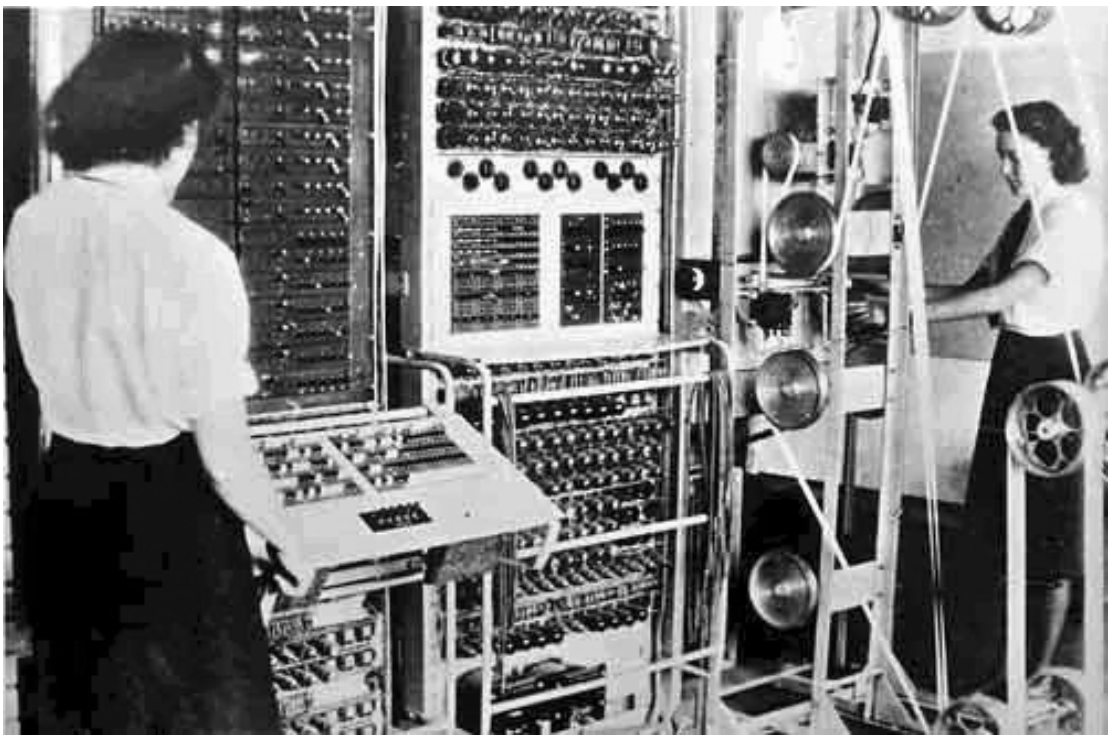


Рис. 1.6. Британський комп'ютер Colossus

У 1946 р. в університеті Пенсильванії (США) було виготовлено перший програмований електронний комп'ютер, що одержав назву Electronic Numerical Integrator and Computer (ENIAC), рис. 1.7. Комп’ютер включав більше 17000 електронних ламп, що були з’єднані 900 км проводів. Загальна маса машини становила 30 т. Комп’ютер був здатний виконувати близько 100000 операцій за секунду. ENIAC використовувався для розрахунку зброї, в тому числі – ядерної, проектування літаків, дослідження космічного випромінювання. Також за допомогою комп’ютера досліджували випадкові помилки округлення чисел. Задавання програми здійснювалося шляхом зміни електричної схеми.

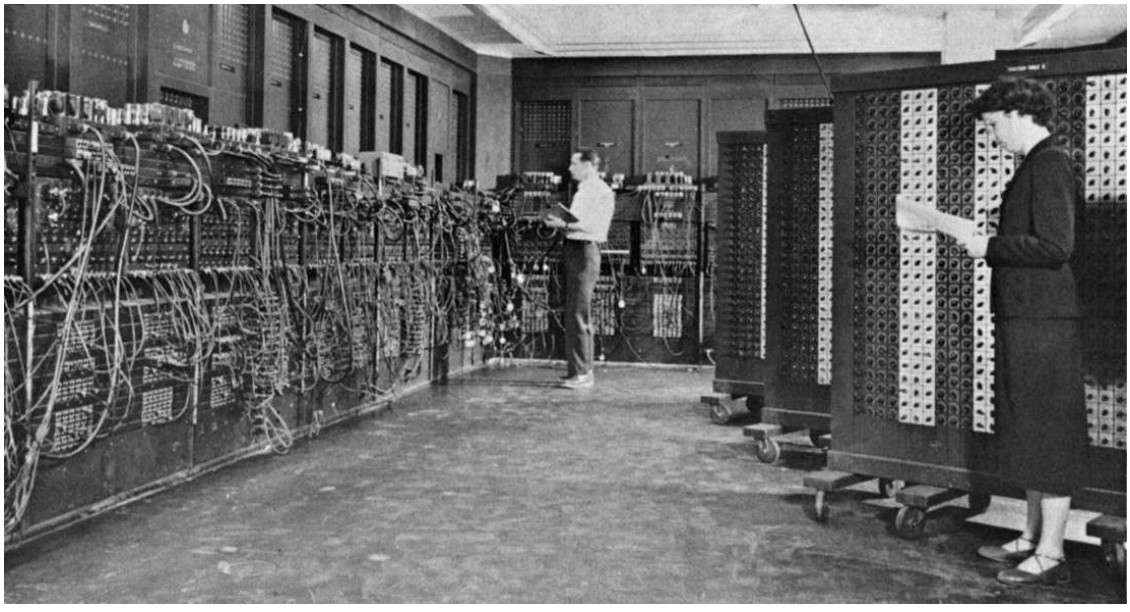


Рис. 1.7. Перший програмований електронний комп’ютер ENIAC

Електронні лампи були дуже ненадійні, споживали багато енергії, мали великі габарити. З винайденням фірмою Bell Laboratories першого транзистора у 1948 р. елементна база комп'ютерів кардинально змінилася. Транзистори займали значно менше місця, були більш надійними, ніж лампи. Подальше зменшення габаритів обчислювальних машин пов'язано з винайденням Джеком Кілбі з фірми Texas Instruments інтегральної мікросхеми у 1948 р. Це дозволило розробити логічні мікросхеми, що суттєво підвищило швидкість роботи та зменшило габарити комп'ютерів.

Бурхливий розвиток комп'ютерної техніки розпочався у 1971 р. з випуском фірмою Intel першого мікропроцесора Intel 4004, що виконаний як одна мікросхема, рис. 1.8.

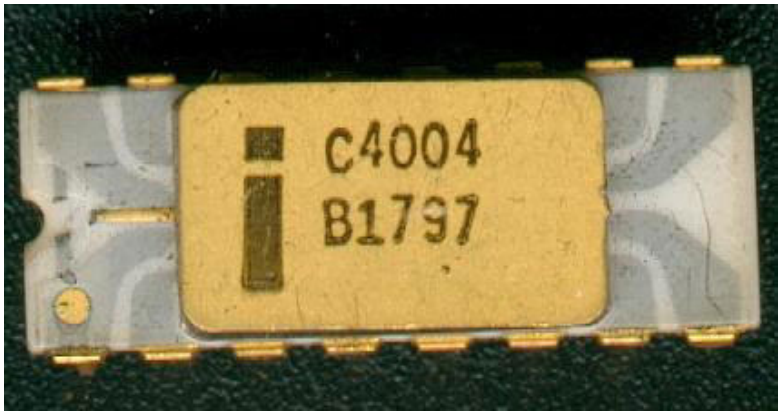


Рис. 1.8. 4-бітовий мікропроцесор Intel 4004, випущений корпорацією Intel у 1971 р.

1.1.2. Основні терміни і визначення, що використовуються в мікропроцесорній техніці

Мікропроцесор (МП) – пристрій, що забезпечує приймання, оброблення та видачу інформації. МП функціонує відповідно до програми, що зберігається у пам'яті. Конструктивно мікропроцесор являє собою одну або декілька інтегральних схем.

Мікропроцесорна система – обчислювальна, вимірювальна або управляюча система, оброблення інформації в якій забезпечується мікропроцесором.

Мультипроцесорна система – система, що включає об'єднані універсальні або спеціалізовані мікропроцесори, що реалізують функції паралельного оброблення інформації та розподіленого управління.

Класифікація мікропроцесорів

1. За призначенням:

1.1. Універсальні – мікропроцесори загального призначення, забезпечують розв'язання широкого класу завдань обчислення, оброблення та управління.

1.2. Спеціалізовані мікропроцесори:

– сигнальні процесори – забезпечують цифрове оброблення сигналів в реальному масштабі часу (зокрема: фільтрація сигналів, обрахування згортки,

обчислення кореляційної функції, перетворення сигналу, виконання перетворення Фур'є тощо);

– *медійні і мультимедійні процесори* – забезпечують для оброблення аудіосигналів, графічної інформації, відеозображень;

2. За кількістю великих інтегральних схем багатокристальні мікропроцесорні комплекти і однокристальні мікропроцесори. Однокристальний мікропроцесор – це конструктивно закінчений виріб у вигляді однієї великої інтегральної схеми.

3. За способом управління розрізняють мікропроцесори зі схемним та з мікропрограмним управлінням. Схемне управління передбачає наявність фіксованого набору команд, що розроблений виробником і не може змінюватися споживачем. У мікропроцесорах, що мають мікропрограмне управління, система команд розробляється при проектуванні конкретного пристрою на базі набору найпростіших мікрокоманд з урахуванням особливості завдань, для вирішення яких призначений мікропроцесорний пристрій.

4. За типом архітектури (за принципом побудови) розрізняють мікропроцесори з фоннеймановською архітектурою та з гарвардською архітектурою.

Архітектура фон Неймана передбачає спільне зберігання даних та машинних команд в комірках однієї пам'яті обчислювальної машини. Для вирізнення даних та команд виникає необхідність побудови складної файлової системи, до файли з даними та командами позначені по-різному. На сьогодні є головною архітектурою для комп'ютерів загального призначення.

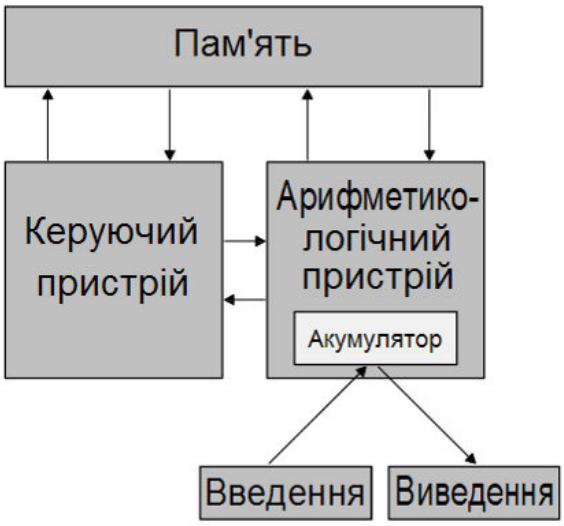


Рис. 1.9. Структурна схема обчислювальної машини,
що має архітектуру фон Неймана

Гарвардська архітектура передбачає наявність окремих запам'ятовуючих пристроїв для збереження даних та кодів програми, рис. 1.10. Вперше такий принцип побудови було використано у комп'ютері Mark I, який був розроблений у Гарварді. В такій машині для збереження команд використовувалися перфокарти, а дані зберігалися у запам'ятовуючому

пристрої на основі реле. На сьогодні відповідно до гарвардської архітектури побудовані однокристальні мікроконтролери, що мають окремі пам’яті для команд та даних. До переваг такого виду архітектури відноситься можливість одночасного завантаження команд і даних. Недоліком є деяка складність апаратної реалізації через наявність двох запам’ятовуючих пристроїв з різними шинами доступу.

Рис. 1.10. Гарвардська архітектура обчислювальної машини

5. За типом системи команд розрізняють:

– CISC (Complete Instruction Set Computing) – мікропроцесори з повним набором команд. У перших мікропроцесорах Intel, які використовувалися в комп’ютерах типу IBM-PC, використовувалася ідеологія повного набору команд CISC. Система команд включала надзвичайно багато складних команд, були передбачені розвинені методи адресації даних. Вважалося, що використання складних низькорівневих команд підвищить ефективність компіляції програм з високорівневих мов. Однак з’ясувалося, що декодування та виконання складних машинних команд потребує багато машинного часу. На початку 1970-х років недоліки CISC призвели до розуміння необхідності спрощення системи команд.

– RISC (Reduced Instruction Set Computing) – мікропроцесори із скороченим набором команд. Мають відносно невелике число коротких і простих команд. Такі команди можуть бути швидко декодовані за допомогою спеціальних електронних схем і виконані. У системі команд відсутні арифметико-логічні операції над операндами безпосередньо у пам’яті. Будь-яке оброблення даних передбачає їх завантаження до регістру (регістрів), виконання операції, збереження результату до пам’яті.

– MISC (Minimal Instruction Set Computer) – мікропроцесори з мінімальним набором команд. Процесори мають невелике число команд, однак, на відміну від RISC, реалізована можливість одночасного виконання взаємно несуперечливих команд. Це дозволяє максимально завантажити маршрутизатори, за якими проходять потоки команд та даних. Компоненти

MISC-процесора прості і функціонують на високих частотах.

– ZISC (Zero Instruction Set Computer) – мікропроцесори з нульовим набором команд. Передбачає використання технології зіставлення із зразком. У процесорах у звичайному розумінні мікрокоманди відсутні. Технологія ZISC заснована на ідеях, запозичених з нейромереж. Реалізується апаратне паралельне оброблення даних.

1.1.3. Розвиток інструментів програмування

Засоби програмування виникли і почали розвиватися одночасно з появою програмованих обчислювальних машин. Для програмування першого програмованого електронного комп'ютера ENIAC необхідно було безпосередньо з'єднувати електричні кола. Оскільки це було надзвичайно незручно, була здійснена можливість користуватися електронними ключами замість з'єднання кіл за допомогою проводів. Для керування такими ключами використовувалися нулі та одиниці. Мова програмування, що передбачає написання програми з використанням 0 та 1, називається машинна мова. Двійкові числа утворюють інструкції, які зберігаються в запам'ятовуючому пристрої. Перехід від збирання електричної схеми до написання програми у двійкових кодах був суттєвим кроком у напрямку спрощення програмування, проте для написання програми, що складалася тільки з 0 та 1 необхідно було багато часу.

В 1950-х роках було розроблено мову асемблера, рис. 1.11. Ця мова забезпечувала можливість заміняти двійковий код, що позначає команду, мнемонічною інструкцією, яка складається з декількох літер. Наприклад, інструкція 01000111 замінялася мнемонікою add.

```
01 DATA SEGMENT
02       MESSAGE DB "HELLO WORLD!!!$"
03 ENDS
04
05 CODE SEGMENT
06     ASSUME DS:DATA CS:CODE
07 START:
08       MOV AX,DATA
09       MOV DS,AX
10       LEA DX,MESSAGE
11       MOV AH,9
12       INT 21H
13       MOV AH,4CH
14       INT 21H
15 ENDS
16 END START
17
```

Рис. 1.11. Фрагмент програми на мові асемблера

Першою мовою високого рівня була Flow-Matic, 1957 р. У 1958 р. фірмою IBM була випущена мова Fortran (FORmula TRANslator), що дозволяла автоматизувати кодування формул, рис. 1.12. Мова Fortran розвивається і понині. Незабаром з'являється мова високого рівня Algol (ALGOrithmic Language), рис. 1.13.

```
      problem readrite(15000, 3000)  ! lorentzian series, a great model
        common /rr/ v(3), vc(3), pw50(3), time0(3),
     +    npoints, deltat, data(180), time(180), error(180)
        npoints = 180  : deltat = .9765625 * ( 512. / 180)
        call setup
        call initial( 1)
        FIND v, pw50, time0; in pulse; by Ajax(cntl); to match error

C   plot signal & data vs. time
        @aplot('~rr-plt')
      end

      model pulse
        common /rr/ v(3), vc(3), pw50(3), time0(3),
     +    npoints, deltat, data(180), time(180), error(180)
        sum = 0
        do 20 j = 1, npoints
          call alorentz( time(j), ampl)
          error(j) = ampl - data(j)
          sum = sum + error(j)**2
   20   continue
      end
```

Рис. 1.12. Фрагмент програми на мові Fortran

Beispiel ALGOL 60:

```
'begin'
   'comment'
      Druckt Zufallszahlen und deren Mittelwert;
    'integer' NN;
    NN := 20;
    'begin'
        'integer' i;
        'real'sum;
        vprint ("Zufallszahlen :");
        sum := 0;
        'for'i := 1 'step'1'until'NN 'do'
        'begin'
           'real'x;
          x := rand;
          sum := sum + x;
          vprint (i,x);
        'end';
        vprint ("Mittelwert :", sum/NN);
    'end'
'end'
```

Рис. 1.13. Фрагмент програми на мові Algol

З того часу з'явилася безліч інших мов програмування. Найбільш відомі з них – Basic, C/C++, Pascal, Ada, Python. Мова Basic була створена 1963 р. для навчання основам програмування студентів неінженерних спеціальностей, рис. 1.14. З тих пір протягом багатьох десятиліть використовувалася в освітньому процесі. Також використовується для розроблення програм для ОС Windows під назвою Visual Basic. VB.NET входить, нарівні з C++, до складу Microsoft Visual Studio, рис. 1.15.

```
 10 INPUT "What is your name: "; U$
 20 PRINT "Hello "; U$
 25 REM
 30 INPUT "How many stars do you want: "; N
 35 S$ = ""
 40 FOR I = 1 TO N
 50 S$ = S$ + "*"
 55 NEXT I
 60 PRINT S$
 65 REM
 70 INPUT "Do you want more stars? "; A$
 80 IF LEN(A$) = 0 THEN GOTO 70
 90 A$ = LEFT$(A$, 1)
100 IF (A$ = "Y") OR (A$ = "y") THEN GOTO 30
110 PRINT "Goodbye ";
120 FOR I = 1 TO 200
130 PRINT U$; " ";
140 NEXT I
150 PRINT
```

Рис. 1.14. Фрагменти програм на мові Basic

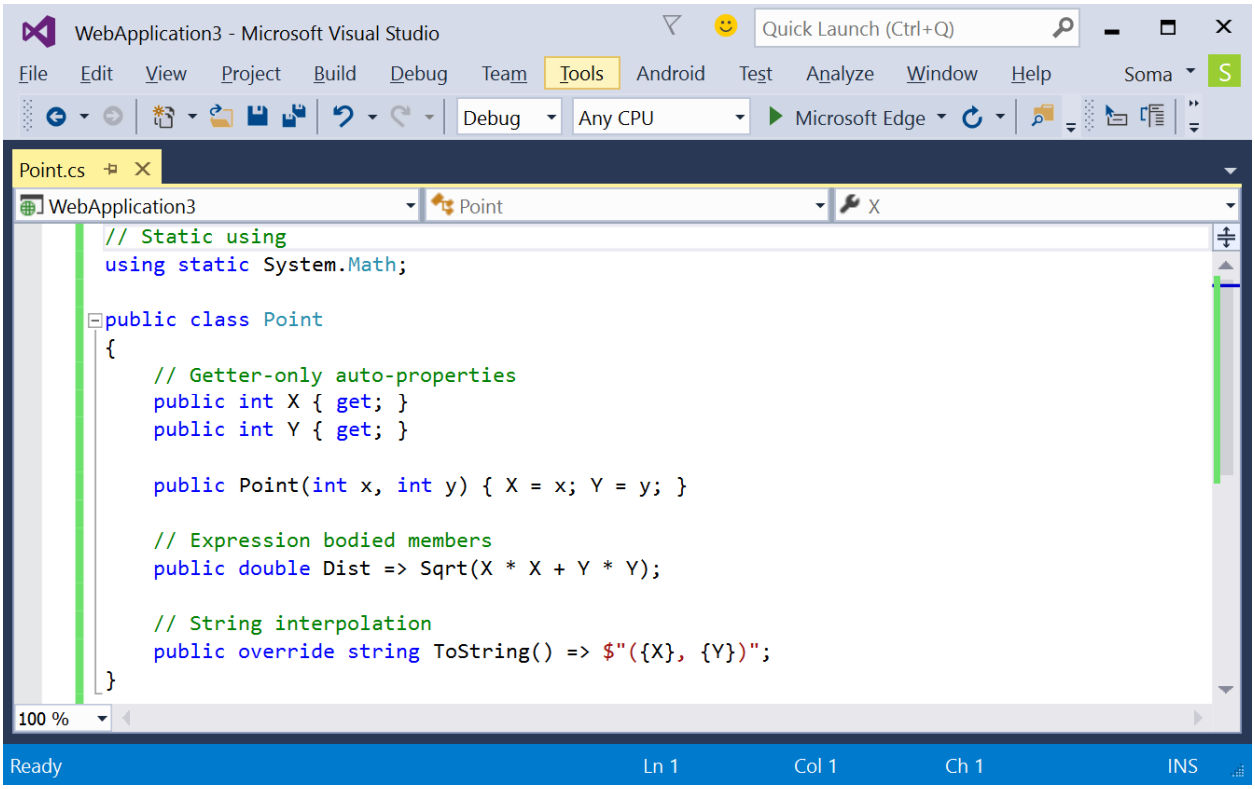


Рис. 1.15. Фрагменти програм на мові VB.NET у Microsoft Visual Studio

Мова Pascal розвивалася, починаючи з 1970-х років, рис. 1.16. Для розроблення наукових програм використовується мова C/C++, рис. 1.17. C/C++ забезпечує функції низького рівня, які раніше можна було реалізувати тільки за допомогою асемблера. Однак, асемблер як і раніше зберігає свої позиції у спеціальних застосунках. Мова Ada застосовується в системі Міністерства оборони США, рис. 1.18. Починаючи з 1990 р. інтенсивно розвивається об'єктно-орієнтована мова програмування високого рівня Python.

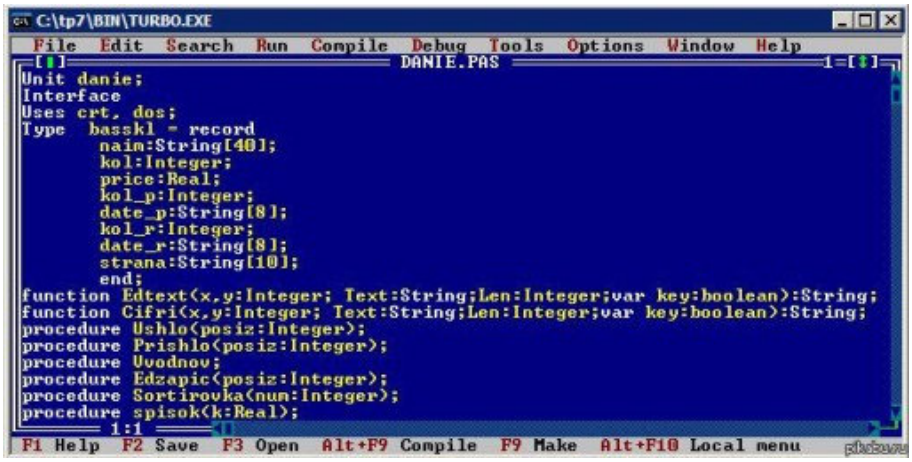


Рис. 1.16. Фрагменти програм на мові Pascal

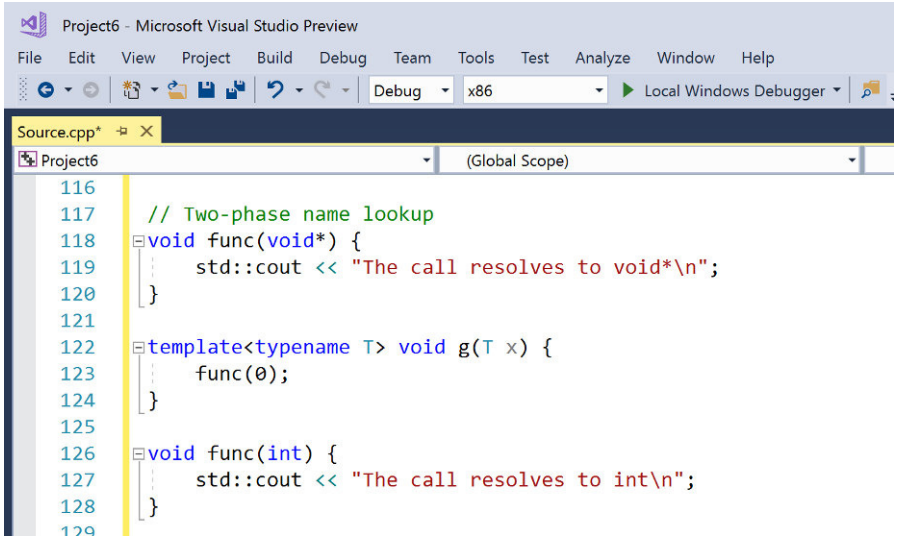


Рис. 1.17. Фрагменти програм на мові C++ у Microsoft Visual Studio

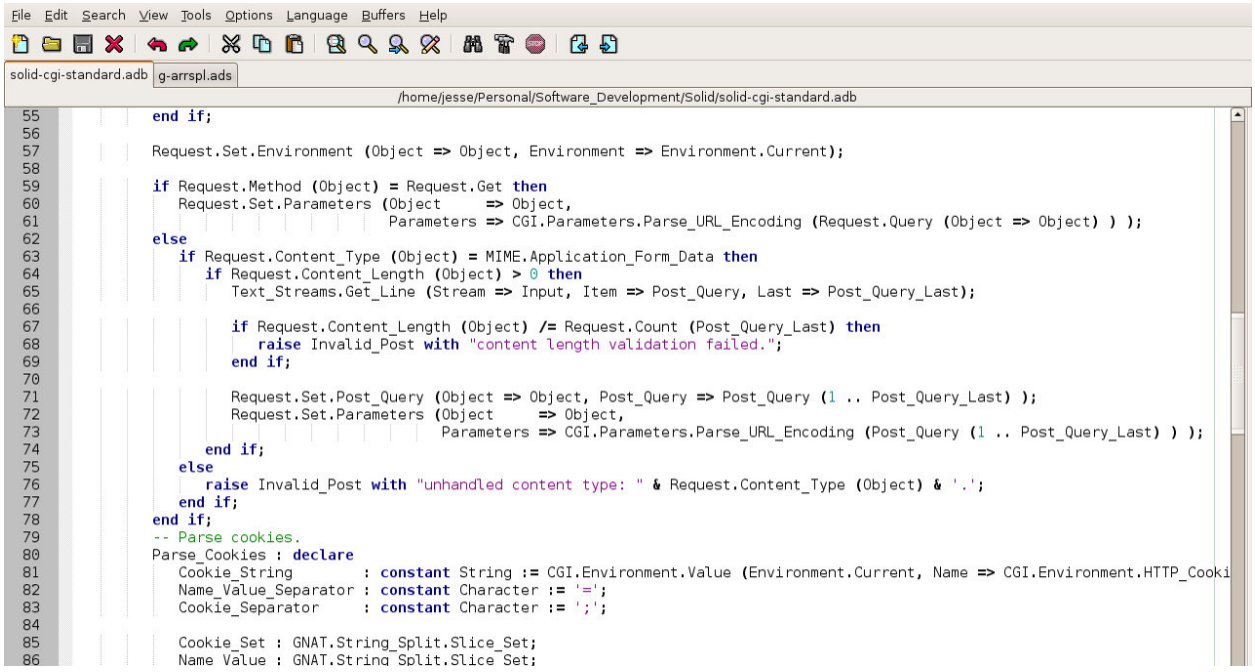


Рис. 1.18. Фрагмент програми на мові Ada

1.1.4. Представлення даних. Десяткова, двійкова і шістнадцяткова системи числення

Мікропроцесорний пристрій складається з набору цифрових електронних схем, в яких розрізняються тільки два логічні стани:

- ввімкнено (стан відповідає логічній одиниці, «1»);
- вимкнено (стан відповідає логічному нулю, «0»).

Тому всі мікропроцесори використовують двійкову систему числення.

При програмуванні використовуються: двійкова, десяткова і шістнадцяткова системи числення (табл. 1.1). У кожній системі числення використовується своє базове значення – максимальне значення числа, яке можна представити за допомогою одного розряду.

Таблиця 1.1

Цифри двійкової, десяткової та шістнадцяткової систем числення

Система числення	*База*	*Цифри, що використовуються*															
Двійкова (bin)	2	0	1														
Десяткова (dec)	10	0	1	2	3	4	5	6	7	8	9						
Шістнадцяткова (hex)	16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Для представлення розрядів шістнадцяткового числа використовуються цифри від 0 до 9 і літери від A до F, що відповідають десятковим числам від 10 до 15. Зазвичай для представлення вмісту пам’яті комп’ютера і відображення цифр машинного коду шістнадцяткові числа використовувати зручніше, рис. 1.19.

```
00000000: 31 20 D0 B4 2E D0 B0 2E|20 2D 20 34 30 20 D1 82
00000010: D0 B8 D1 81 2E 20 D0 B7|D0 BD 20 D0 B7 20 D0 BF
00000020: D1 80 D0 BE D0 B1 20 2D|20 32 34 20 D1 81 D1 82
00000030: D0 BE D1 80 2E 0D 0A 0D|0A D1 87 D0 B0 D1 81 D1
00000040: 82 D0 B8 D0 BD D0 B0 20|D0 A1 D0 92 3A 20 34 2C
00000050: 36 20 D0 B4 2E D0 B0 2E|0D 0A D1 87 D0 B0 D1 81
00000060: D1 82 D0 B8 D0 BD D0 B0|20 D0 9A D0 A1 3A 20 31
00000070: 2C 36 20 D0 B4 2E D0 B0|2E 0D 0A D1 87 D0 B0 D1
00000080: 81 D1 82 D0 B8 D0 BD D0|B0 20 D0 90 D0 92 3A 20
00000090: 31 2C 36 20 D0 B4 2E D0|B0 2E 20 0D 0A 0D 0A D0
000000A0: B4 D0 BB D1 8F 20 D1 96|D0 BD D0 B4 20 D0 BF D0
000000B0: BB D0 B0 D0 BD D1 83 3A|20 37 2E 38 20 D0 B4 2E
```

Рис. 1.19. Представлення вмісту текстового файлу шістнадцятковими кодами

Способи запису. У математиці основу системи числення прийнято вказувати в десятковій системі в нижньому індексі. Наприклад, десяткове число 1443 можна записати як 1443_{10} або як $5A3_{16}$. Двійкові числа зазвичай завершують символом b. Наприклад, 01011010b. Іноді десяткові числа завершуються символом d. Наприклад, 1443d. Шістнадцяткові числа можуть завершуватися символом h. Зокрема: 8Bh.

У різних мовах програмування для запису шістнадцяткових чисел використовують різний синтаксис:

Мова	*Спосіб запису*
Ada, VHDL	16#5A3#
C, Java	00x5A3
Асемблер	5A3h, якщо число починається не з десяткової цифри, то для відмінності від імен ідентифікаторів (наприклад, констант) попереду ставиться "0" (нуль) : "0FFh"
Асемблери (AT&T, Motorola), Паскаль, деякі версії Бейсика	$5A3
Асемблери ZX Spectrum	#5A3
Інші версії Бейсика	&h5A3

Двійкові числа. Комп'ютер зберігає команди і дані в пам'яті у вигляді набору електричних зарядів, кожен з яких відповідає одному біту.

Біт – мінімальна одиниця інформації. Біт приймає значення «0» або «1».

Байт – одиниця інформації розміром 8 біт.

Біти в байті нумерують справа наліво, від нуля до семи, нульовий біт часто називають молодшим бітом, а сьомий – старшим, табл. 1.2.

Таблиця 1.2

Розряди байта

Номер розряду байта	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
Значення розряду	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
	128	64	32	16	8	4	2	1	
Назви бітів	старший біт							молодший біт	
Назви тетрад	старша тетрада				молодша тетрада				
Приклад вмісту розряду	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**1**	
Значення	1·128+	0·64+	1·32+	0·16+	0·8+	1·4+	0·2+	1·1=	165

Слово **–** одиниця інформації розміром 16 біт (2 байти), табл. 1.3.

Подвійне слово – одиниця інформації розміром 32 біта (4 байти).

Збільшене учетверо слово – одиниця інформації розміром 64 біта.

Таблиця 1.3

Одиниці інформації

Одиниця інформації	*Кількість біт*	*Ступінь двійки*	*Діапазон значень*
Байт	8	$0…(2^8–1)$	0...255
Слово	16	$0…(2^{16}–1)$	0...65535
Подвійне слово	32	$0…(2^{32}–1)$	…
Збільшене учетверо слово	64	$0…(2^{64}–1)$	…

Двійкові цілі числа можуть мати знак або бути беззнакові.

Цілі двійкові числа: а) зі знаком (додатні, нуль, від'ємні); б) беззнакові (додатні, нуль).

При використані спеціальних схем кодування за допомогою двійкових чисел можуть бути представлені дійсні числа.

Беззнакові цілі двійкові числа виражаються у вигляді:

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

– значення розрядів двійкового числа

Для обрахування еквівалентного десяткового значення *n*-розрядного двійкового числа використовується зважено-позиційна форма запису:

$$DEC = (B_{n-1} \cdot 2^{n-1}) + (B_{n-2} \cdot 2^{n-2}) + ... + (B_1 \cdot 2^1) + (B_0 \cdot 2^0),$$

де n – кількість розрядів двійкового числа; B_i – значення i-го двійкового розряду.

Наприклад, переведення двійкового числа 10011010b у десяткову форму здійснюється наступним чином:

7	6	5	4	3	2	1	0	
1	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**b**
2^7			2^4	2^3		2^1		

128 + 16 + 8 + 2 = 154

Для перетворення беззнакового десяткового числа в двійкове, необхідно зробити послідовні операції цілочисленого ділення на 2, записуючи залишок від ділення в двійковий розряд.

Наприклад, перетворимо десяткове число 25d у двійкове:

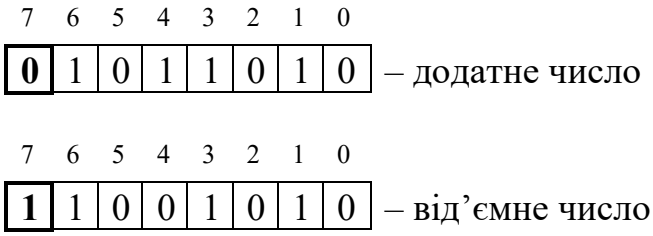
25	2				
24	12	2			
1	12	6	2		
	0	6	3	2	
		0	2	1	2
			1	0	0
				1	
1	0	0	1	1	

Записуємо залишки, що булі отримані, у зворотному порядку: 11001b.

Зазвичай доповнюють нулями в старших розрядах до байту: 00011001b.

Перевірка: 24 + 23 + 20 =16 + 8 + 1 = 25 d.

Цілі двійкові числа зі знаком. В якості знакового розряду використовується старший біт числа. Якщо його значення дорівнює нулю, число додатне, якщо одиниці, то від'ємне:



В двійковому цілому *n*-розрядному числі зі знаком для представлення абсолютного значення числа може використовувати тільки (*n*−1) бітів. Тобто 1 байт може зберігати десяткове число від (-2^7) до (2^7-1), що відповідає діапазону (−128)…(+127).

Шістнадцяткові числа. З двійковими числами дуже важко працювати напряму, тому в асемблері використовується шістнадцяткова форма запису. Кожна цифра шістнадцткового числа являє собою тетраду (4 біта), табл. 1.4.

Таблиця 1.4

Відповідність між двійковими, десятковими та шістнадцятковими числами у межах тетради

bin				dec	hex
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7
1	0	0	0	8	8
1	0	0	1	9	9
1	0	1	0	10	A
1	0	1	1	11	B
1	1	0	0	12	C
1	1	0	1	13	D
1	1	1	0	14	E
1	1	1	1	15	F

Дві шістнадцяткові цифри являють 1 байт:

0101|1101 b = 5D h
5 | D
старша тетрада | молодша тетрада

Розряди шістнадцяткового числа нумеруються цифрами від нуля справа наліво:

1 | 0
5 | D h

1.1.5. Логічні операції. Типові логічні елементи

Сучасні пристрої оброблення інформації побудовані на основі цифрових (логічних) інтегральних мікросхем. Для аналізу функціонування цифрових мікросхем використовується апарат математичної логіки – алгебра Джорджа Буля. Подію може вважатися істинною, що позначається логічною одиницею «1» (на схемотехнічному рівні – високим рівнем напруги) або хибною, що позначається логічним нулем «0» (низький рівень напруги). Для оброблення інформації використовується двійкова система числення. Змінна, що може приймати тільки два значення («0», «1»), називається двійковою (логічною) змінною. Алгебру Буля на практиці вперше у 1938 р. застосував родоначальник кібернетики Клод Шеннон (США) під час дослідження електричних кіл з контактними перемикачами.

Зазвичай в цифрових пристроях рівні лог. «0» та лог. «1» представляються рівнями напруги або величинами струму.

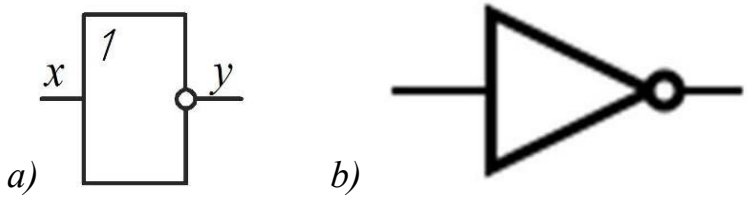
Розглянемо базові логічні функції.

1. **Функція НІ** (NOT, функція заперечення, операція інверсії):

$$y = \bar{x}, \tag{1.1}$$

де x – вхідна двійкова змінна; y – вихідна двійкова змінна.

Залежність (1.1) читається «y дорівнює не x». Графічне позначення елемента, що реалізує таку функцію, наведено на рис. 1.20.



x	$y = \bar{x}$
0	1
1	0

Рис. 1.20. Логічний елемент **НІ (NOT):** *a* – європейське позначення (IEC); *b* – американське позначення (ANSI); *c* – таблиця істинності

Функцію, яку виконує комбінаційний пристрій, для полегшення сприйняття часто представляють у вигляді таблиці, яку називають таблицею істинності. Кількість стовбців цієї таблиці дорівнює числу вхідних змінних. Є ще один стовбець, в якому вказують значення функції для кожної з можливих комбінацій вхідних змінних, числу яких відповідає кількість рядків таблиці.

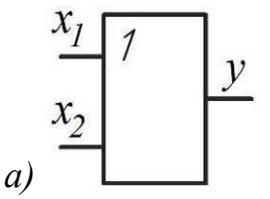
2. **Функція АБО** (OR, логічне додавання, диз'юнкція):

$$y = x_1 + x_2 = x_1 \vee x_2, \tag{1.2}$$

де x_1, x_2 – вхідні двійкові змінні (операнди); y – вихідна двійкова змінна; знак

«+» – оператор ЛОГІЧНОГО додавання (не треба плутати з додаванням арифметичним).

Залежність (1.2) читається «y дорівнює x_1 або x_2». Графічне позначення елемента, що реалізує функцію АБО, наведено на рис. 1.21.



a)

b)

c)

x_1	x_2	$y = x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1

Рис. 1.21. Логічний елемент **АБО (OR)**: *a* – європейське позначення (IEC); *b* – американське позначення (ANSI); *c* – таблиця істинності

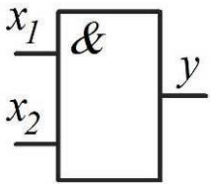
Сигнал на виході елемента АБО дорівнює логічній «1», якщо хоча б один вхідний сигнал дорівнює лог. «1».

3. **Функція І** (AND, логічне множення, кон'юнкція):

$$y = x_1 \cdot x_2 = x_1 \wedge x_2, \tag{1.3}$$

де x_1, x_2 – вхідні двійкові змінні (операнди); y – вихідна двійкова змінна; знак «·» – оператор ЛОГІЧНОГО множення (не треба плутати з множенням арифметичним).

Залежність (1.3) читається «y дорівнює x_1 і x_2». Графічне позначення елемента, що реалізує функцію І, наведено на рис. 1.22. Символ & називається амперсанд.



a)

b)

c)

x_1	x_2	$y = x_1 \cdot x_2$
0	0	0
0	1	0
1	0	0
1	1	1

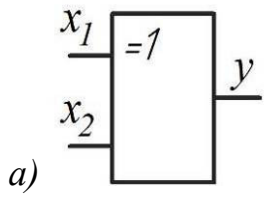
Рис. 1.22. Логічний елемент **І (AND)**: *a* – європейське позначення (IEC); *b* – американське позначення (ANSI); *c* – таблиця істинності

Сигнал на виході елемента І дорівнює логічній «1», якщо обидва вхідні сигнали дорівнюють лог. «1».

4. **Функція ВИНЯТКОВЕ_АБО** (XOR, виняткова диз'юнкція):

$$y = x_1 \oplus x_2. \tag{1.4}$$

Графічне позначення елемента, що реалізує функцію ВИНЯТКОВЕ _АБО, наведено на рис. 1.23.



a)


b)

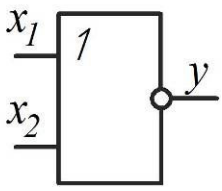
x_1	x_2	$y = x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

c)

Рис. 1.23. Логічний елемент **ВИНЯТКОВЕ_АБО (XOR):** *a* – європейське позначення (IEC); *b* – американське позначення (ANSI); *c* – таблиця істинності

Стислий опис: «істина на виході – тільки при істині на вході 1, або тільки при істині на вході 2».

Дуже часто на виході елементів АБО, І, ВИНЯТКОВЕ_АБО включають елемент НІ. Для спрощення позначення, вихід таких елементів позначають колом. Такі елементи називають АБО-НІ (рис. 1.24), І-НІ, ВИНЯТКОВЕ_АБО-НІ.



a)


b)

Рис. 1.24. Позначення елемента **АБО-НІ (NOR)**:
a – європейське позначення (IEC);
b – американське позначення (ANSI)

Порядок виконання логічних операцій задається круглими дужками. При відсутності дужок операції виконуються у наступній послідовності: NO, AND, OR.

1.1.6. Представлення символьних даних. Таблиці ASCII, UNICODE

Оскільки комп'ютер оперує тільки з числами, тому графічним образам символів (літери, цифри, знаки), що відображаються на екрані, необхідно присвоїти унікальні номери, тобто закодувати. Саме такі коди символів зберігаються в пам'яті. При необхідності відобразити певний символ його код подається на знакогенератор (або більш складну схему), який визначає пікселі, що мають засвітитися на екрані у формі необхідного символу.

На початку розвитку обчислювальної техніки для зберігання кодів символів використовувався один байт, що дозволяло кодувати 256 символів. У 1963 р. в США було розроблено систему кодування ASCII (American Standard Code for Information Interchange, Американський стандартний код для інформаційного обміну). Всі 256 символів були розділені на 2 частини.

Символи першої частини (з кодами від 0 до 127) включають службові символи, цифри, знаки пунктуації та англійський алфавіт, рис. 1.25.

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Рис. 1.25. Символи ASCII, що мають коди від 0 до 127

Рис. 1.26. Фрагмент таблиці UNICODE, що відображає кириличні літери

Друга половина таблиці («розширення ASCII», символи з кодами від 128 до 255) мала багато виконань, оскільки призначена для відображення літер різних алфавітів. Набір і порядок символів другої половини таблиці відрізняються в різних країнах і навіть в межах однієї країни. Наприклад, для букв кирилиці існує п’ять варіантів розміщення в другій половині таблиці символів ASCII. Система ASCII на сьогодні у персональних комп’ютерах майже не використовується, проте широко застосовується в промисловій автоматиці для обміну інформацією.

На сьогоднішній день переважаючим стандартом для кодування інформації є UNICODE, який забезпечує кодування письмових знаків майже всіх мов світу, рис. 1.26. Перший такий стандарт був випущений 1991 р. З того часу було опубліковано багато оновлених версій. Стандарт використовується у багатьох новітніх технологіях: XML, мові програмування JavaScript, операційних системах. Юнікод знімає обмеження на кодування символів лише одним байтом. Натомість використовується 17 просторів, кожен з яких визначає 65536 кодів і дає можливість описати максимум 1 114 112 ($17 \cdot 2^{16}$) різних символів.

1.1.7. Питання для самоперевірки

1. Який принцип дії обчислювальної машини «Паскаліна»?
2. Як була утворена корпорація IBM?
3. На який елементах працював комп'ютер Z3?
4. Яким чином програмували ENIAC?
5. Дайте визначення мікропроцесору.
6. Які типи архітектури мікропроцесорів Вам відомі? В чому ключова різниця між ними?
7. Охарактеризуйте RISC процесори. В чому їх відмінність від CISC.
8. Назвіть особливості мови асемблера.
9. Які цифри використовуються у шістнадцятковій системі числення?
10. Що таке біт? Байт?
11. Яким чином здійснити переведення числа між десятковою та двійковою формами та навпаки?
12. Як перевести число між десятковою та шістнадцятковою формами та навпаки?
13. Опишіть переведення числа між двійковою та шістнадцятковою формами.
14. Охарактеризуйте типові логічні функції.
15. Які відмінності між таблицями ASCII та UNICODE?

1.2. Мікросхеми

Загальне поняття про інтегральну мікросхему. Класифікація мікросхем. Типи логіки. Параметри мікросхем. Шифратор. Дешифратор. Мультиплексор. Демультиплексор. Принцип дії тригера. Різновиди тригерів. Регістри.

1.2.1. Загальне поняття про інтегральну мікросхему

Мікросхема (інтегральна мікросхема, integrated circuit) – це електронна схема, яка виконана на напівпровідниковому кристалі (чипі) та призначена для виконання певної функції. Перша мікросхема винайдена у 1958 р. американськими винахідниками Джеком Кілбі та Робертом Нойсом, рис. 1.27.

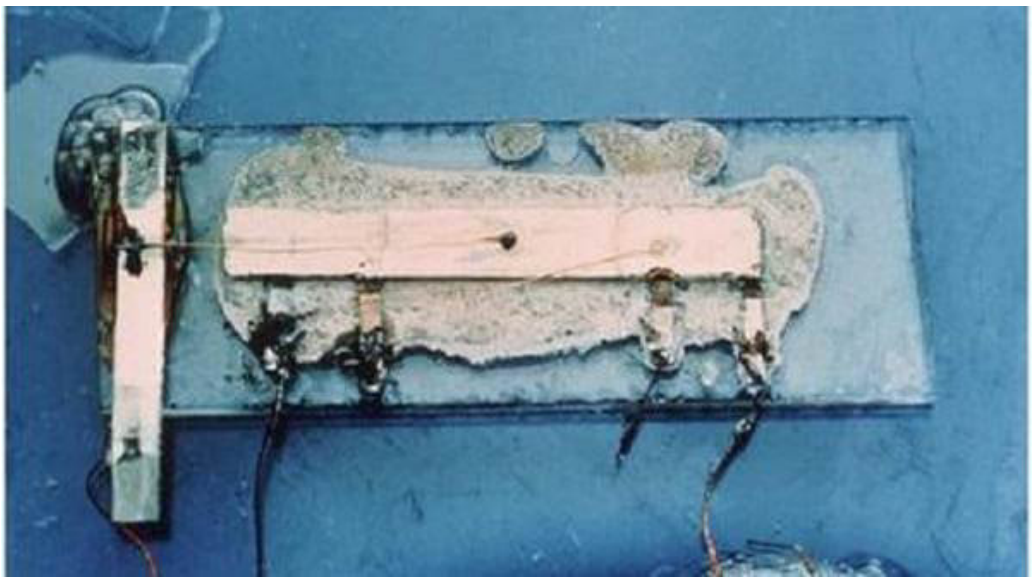


Рис. 1.27. Перша в світі інтегральна мікросхема виглядала не надто презентабельно, проте це відкриття назавжди змінило життя людства

Чип – напівпровідникова структура, на поверхні якої сформовані контактні площинки.

Часто інтегральною схемою (IC) називають власне кристал або плівку з електронною схемою, а мікросхемою (МС) – інтегральну схему в корпусі, рис. 1.28. Часто термін інтегральна мікросхема (IMC) заміняється терміном інтегральна схема (IC), або просто мікросхема. Інтегральна мікросхема забезпечує перетворення та оброблення сигналів. Вона має високу щільність пакування елементів і компонентів, що електрично з'єднані і розглядаються як єдине ціле. Одна мікросхема може містити більше 10 міліардів елементів на одному кристалі. До складу інтегральної мікросхеми входять елементи та компоненти.

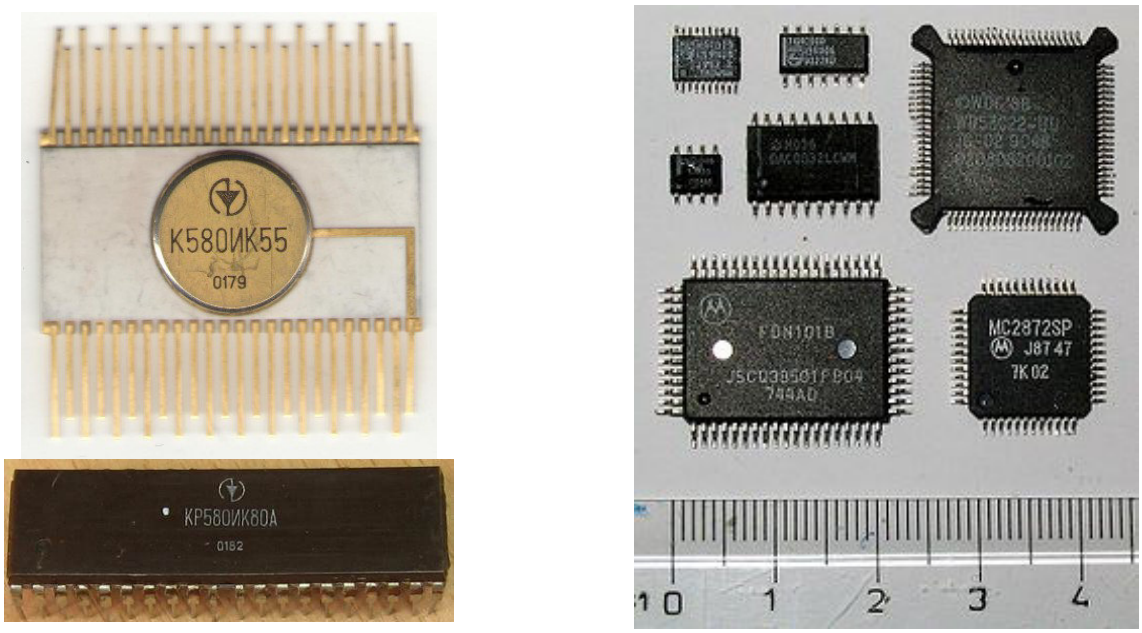


Рис. 1.28. Мікросхеми різних років випуску

Елемент (element) мікросхеми – частина ІМС, яка реалізовує функцію елементарного радіоелементу (наприклад, транзистора, діода, резистора, конденсатора). Елемент виконується нероздільно від кристала ІМС (або її підкладки). Його не можна відокремити від ІМС як самостійний виріб, тому елемент окремо не можна випробувати або експлуатувати, рис. 1.29. Компонент (component) мікросхеми – частина ІМС, що до монтажу була самостійним виробом у власному корпусі. Теоретично компонент можна відокремити від мікросхеми.

Інтегральні мікросхеми дозволяють надзвичайно зменшити розміри обчислювальних машин. Обчислювальна потужність сучасних смартфонів, завдяки використанню надзвичайно компактних мікросхем, в тисячі разів перевищує потужність перших комп'ютерів на електронних лампах, що займали площу спортивного залу. Завдяки високій якості виготовлення мікросхеми характеризуються надзвичайно високою надійністю, що визначає безвідмовність мікроелектронної апаратури.

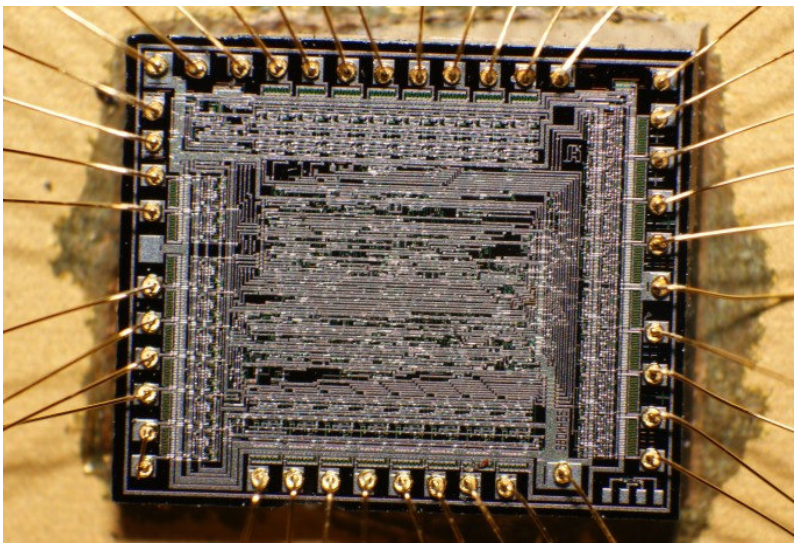


Рис. 1.29. Всередині мікросхеми

Інтегральні мікросхеми розробляються і виготовляються у вигляді серій.

Серія – сукупність типів інтегральної схеми, які призначені для виконання різних функцій, однак мають єдину технологічну будову і призначені для спільного застосування. Тобто вхідні та вихідні сигнали мікросхем однієї серії є узгодженими, що дозволяє безпосередньо з'єднувати декілька мікросхем однієї серії.

Для захисту від негативного впливу зовнішнього середовища чип розміщують у корпусі. Також корпус призначено для розташування виводів, за допомогою яких чип з'єднується з зовнішніми колами.

Корпусування мікросхем. Розташування кристала в корпусі є завершальним етапом виготовлення мікросхеми. Для з'єднання контактних майданчиків, що розміщені на периферії кристала мікросхеми, з виводами корпусу можуть застосовуватися декілька методів. Зокрема, використовується

метод розпаювання виводів (wire bonding), який передбачає використання дротяних провідників, для припаювання яких використовується лазерне або ультразвукове зварювання, рис. 1.30.

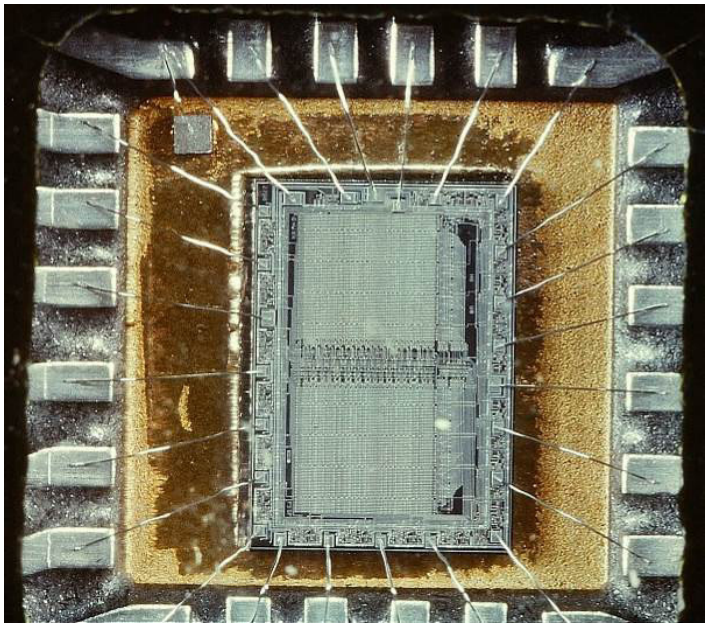


Рис. 1.30. З’єднання кристала мікросхеми та виводів корпусу за допомогою напаяних провідників (метод wire bonding)

Також може застосовуватися монтаж методом перевернутого чипа (flip chip). При такому методі корпусування кристал мікросхеми встановлюється на виводи, що виконані на його контактних майданчиках. Контактний майданчик чипа називають бамп (bump). Бампи розміщені по всій поверхні кристала мікросхеми. Бампи кристала спаюють з контактними майданчиками корпусу за допомогою спеціальних кульок припою, які оплавляються під дією гарячого повітря, рис. 1.31.

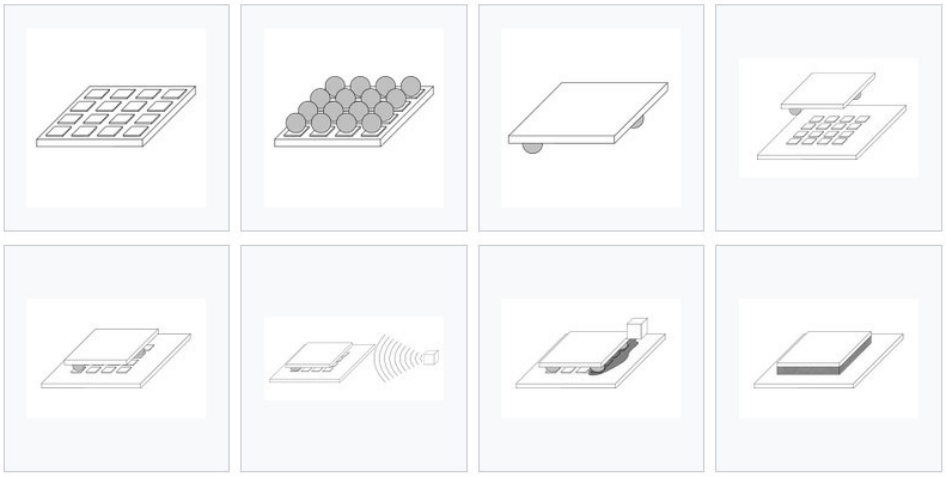
Види корпусів мікросхем

1) *DIP (Dual Inline Package)* – прямокутний корпус з двома рядами виводів по довших сторонах, що передбачає наявність монтажних отворів у друкованій платі для продівання виводів, рис. 1.32. Зазвичай виводи припаюються до контактних майданчиків з іншої сторони друкованої плати порівняно з розташуванням корпусу. Також може встановлюватися у попередньо впаяний у плату роз’єм.

Виготовляється з пластику (PDIP) або кераміки (CDIP). Загальна кількість виводів не перевищує 64. Зазвичай в позначенні корпусу вказується кількість виводів. Наприклад, корпус з 14 виводами позначається DIP-14.

Вивід № 1 визначається за ключем, в якості якого виступає виїмка на краю корпусу або нанесена точка (часто – білою фарбою). Якщо дивитися на мікросхему зверху (зі сторони позначень на корпусі), то виводи від першого нумеруються проти годинникової стрілки. Відповідно, якщо дивитися на мікросхему знизу (зі сторони виводів), то нумерація виводів здійснюється за годинниковою стрілкою.

а)



б)

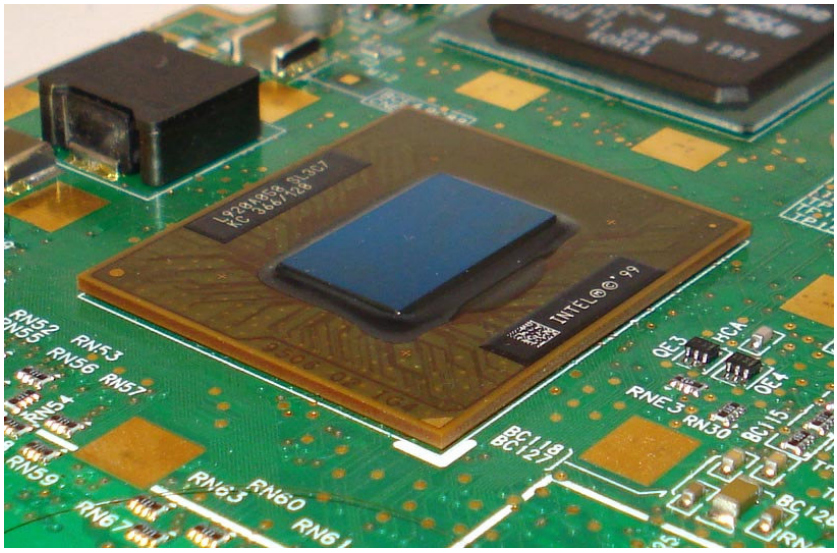


Рис. 1.31. Монтаж методом перевернутого чіпа: *а* – процес спаювання бампів кристала з контактними майданчиками корпусу мікросхеми; *б* – Intel Mobile Celeron з фліп-чіпом, кремнієва матриця виглядає як темно-синій прямокутник

Відстань між центрами сусідніх виводів становить 0,1 дюйма (2,54 мм), відстань між рядами виводів 0,3 або 0,6 дюйма (7,62 мм або 15,24 мм). Креслення DIP корпусу наведено на рис. 1.32, *е*, де позначені наступні габаритні розміри:

H=7,7 мм – загальна висота мікросхеми (Height);

L – Загальна довжина (Len), залежить від кількості виводів;

L_W=1,14–1,73 мм – ширина верхньої частини виводу;

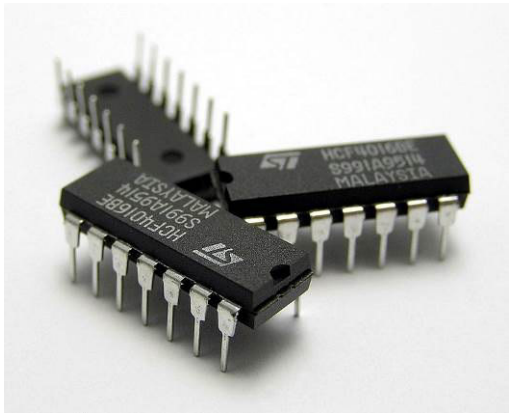
L_L=3,05–3,6 мм – довжина виводу;

P=2,54 мм – відстань між центрами виводів (Pitch);

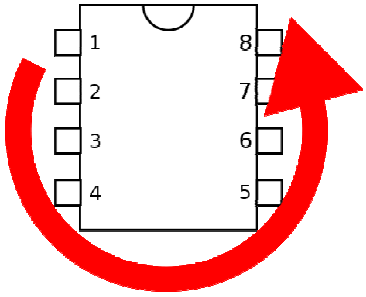
W_B – ширина корпусу (без виводів);

W_L=7,62 мм або 15,24 мм – загальна ширина корпусу.

У корпусі DIP випускаються різні напівпровідникові або пасивні компоненти – діодні збірки, мікросхеми тощо.

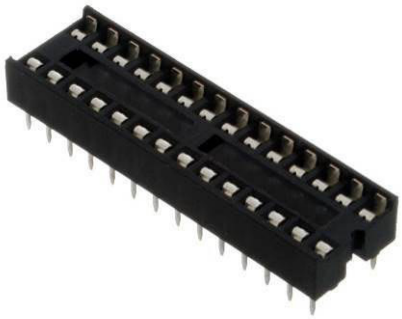


а)

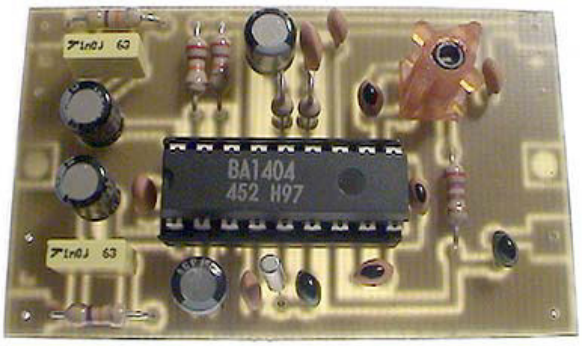


б)

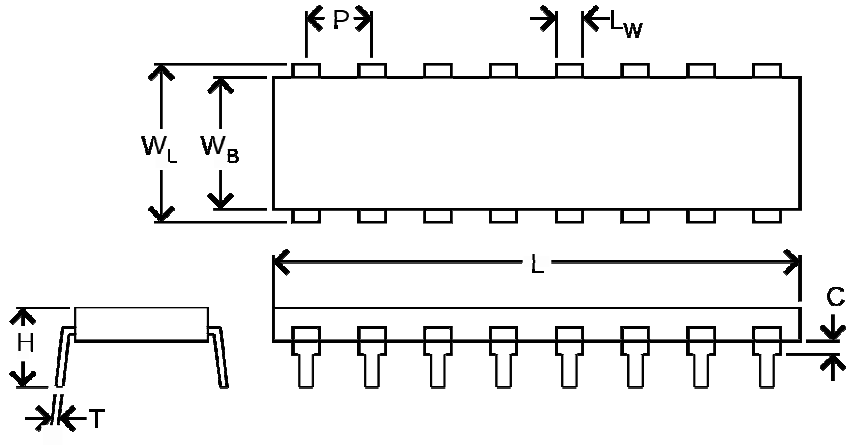
в)



г)



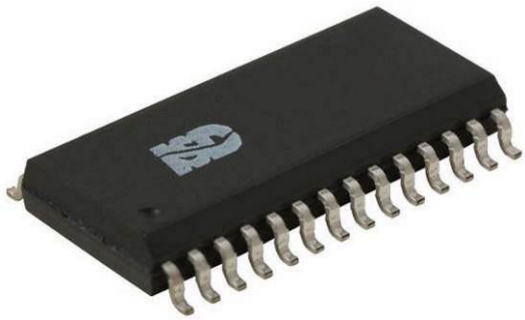
д)



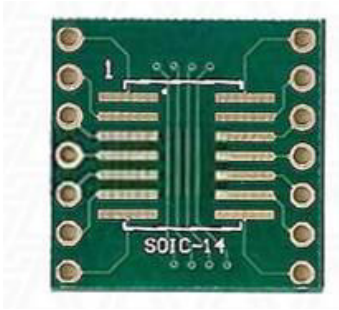
е)

Рис. 1.32. Мікросхема у DIP корпусі:
а – загальний вигляд; *б* – нумерація виводів (вид зверху); *в* – мікросхема в корпусі DIP-40, впаяна в плату; *г* – роз'єм для встановлення DIP-28 (підставка) мікросхеми; *д* – мікросхема в корпусі DIP-18, вставлена до роз'єму на платі; *е* – габаритні розміри DIP корпусу

2) *SOIC (Small-Outline Integrated Circuit)* – прямокутний корпус, виводи розташовані по двом більшим сторонам, рис. 1.33. На відміну від DIP, корпус SOIC не потребує наявності отворів у друкованій платі. Виводи мікросхеми припаюють до контактних майданчиків з тієї ж сторони друкованої плати, що і корпус мікросхеми. Відстань між центрами виводів становить 1,27 мм. Габарити мікросхеми в корпусі SOIC приблизно на половину менші, ніж в DIP корпусі.



а)



б)

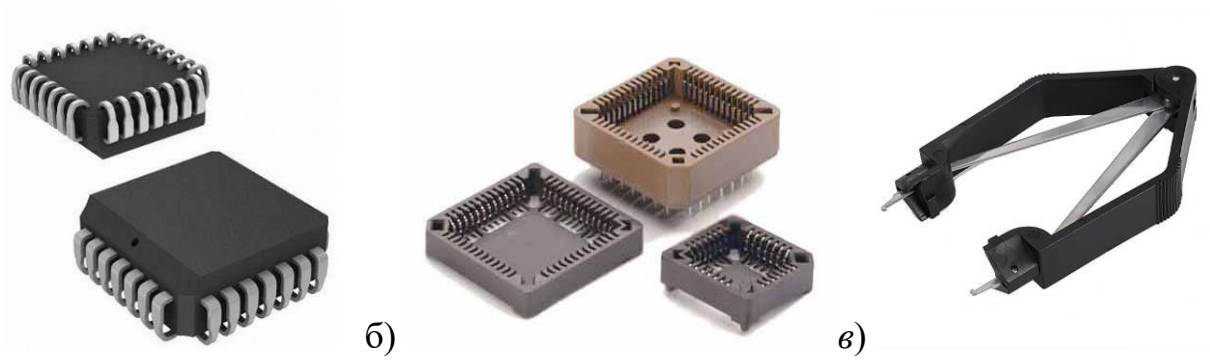
в)

Рис. 1.33. Мікросхема у корпусі SOIC: *а* – загальний вигляд; *б* – контактні майданчики для припаювання мікросхеми у корпусі SOIC-14; *в* – мікросхема у корпусі SOIC-16 на друкованій платі

3) *TQFP (Thin Quad Flat Pack)* – корпус передбачає розташування виводів по чотирьох сторонах мікросхеми, рис. 1.34. Розмір виводів 2 мм. Ширина однієї сторони корпусу може становити від 5 мм до 20 мм. Кількість виводів – від 32 до 176. Крок виводів 0,4; 0,5; 0,65; 0,8 і 1 мм.

Рис. 1.34. TQFP-мікросхема

4) *PLCC (Plastic Leaded Chip Carrier)* та CLCC (*Ceramic Leaded Chip Carrier*) – корпус з гнучкими контактами по чотирьом сторонам квадратного корпусу, має встановлюватися у спеціальну панель, що впаяна у друковану плату, рис. 1.35.



а) б) *в*)

Рис. 1.35. Мікросхеми PLCC виконання (*а*), панелі для них (*б*) та пінцет для виймання мікросхеми з панелі (*в*)

5) *LGA (Land Grid Array)* – корпус для поверхневого монтажу мікросхеми, особливістю якого є наявність виводів на сокеті, а на мікросхемі виконані контактні майданчики, рис. 1.36. Сокет припаюється до друкованої плати, мікросхема вкладається в сокет і притискається. Перевагою є відсутність жорсткого з'єднання корпусу мікросхеми з друкованою платою, що підвищує надійність роботи за наявності динамічних перевантажень. В разі сильних ударів або неправильного закріплення плати мікросхема буде «плавати» в сокеті, зберігаючи електричне з'єднання.

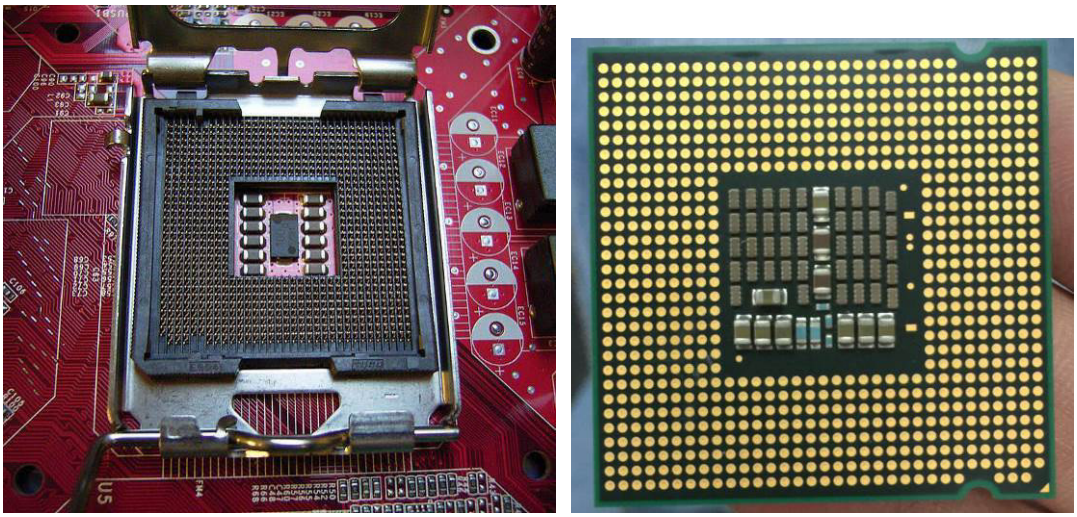


Рис. 1.36. Сокет (*а*) та LGA-мікросхема (б)

6) *BGA* (*Ball grid array – масив кульок*) – корпус інтегральної мікросхеми, що містить на одній зі сторін контактні майданчики, призначені для припаювання, у вигляді матриці. На контактні майданчики корпусу мікросхеми заводським способом наносять кульки припою. На друкованій платі також виконують відповідні контактні майданчики. Під час монтажу мікросхему

нагрівають і припій починає плавитися. Сили поверхневого натягу фіксують корпус над призначеним місцем, рис. 1.37, 1.38.

До переваг BGA-мікросхем відноситься компактність, наявність великої кількості виводів, добрий тепловий контакт мікросхеми і плати, зменшені наведені завади та збільшений діапазон робочих частот за рахунок меншої індуктивності контактних майданчиків порівняно з довгими виводами. До недоліків відноситься негнучкість виводів: деякі виводи можуть зламатися при тепловому розширенні або вібрації. Для вирішення цієї проблеми мікросхему заливають компаундом. Також до недоліків відноситься складність виявлення дефектів припаювання виводів після монтажу.

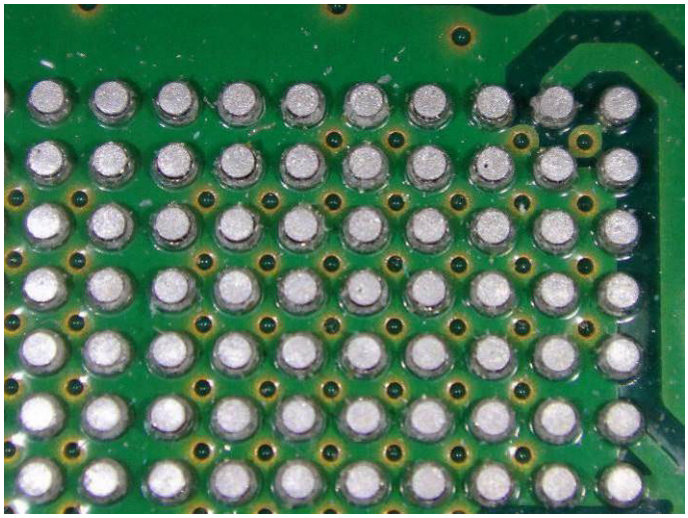


Рис. 1.37. Кульки припою на друкованій платі у вигляді матриці після видалення BGA-мікросхеми

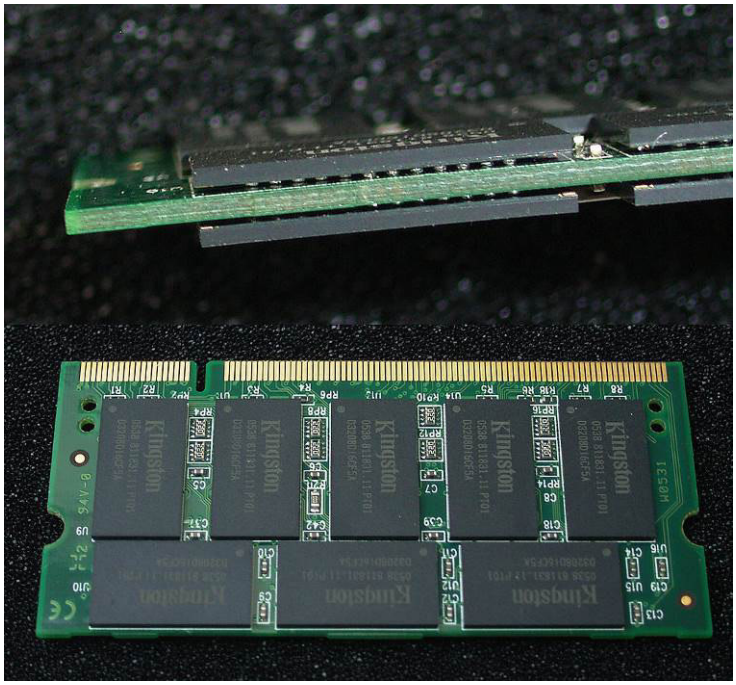


Рис. 1.38. BGA-мікросхеми на друкованій платі

1.2.2. Класифікація мікросхем

За принципами будови та технологією виготовлення інтегральні мікросхеми поділяють на наступні типи.

Напівпровідникова інтегральна мікросхема – мікросхема, що включає єдиний кристал напівпровідника, в об'ємі і на поверхні якого виконані всі елементи. Сукупність технологічних операцій, що забезпечують виготовлення пласких напівпровідникових мікросхем, називається планарною технологією, рис. 1.39.

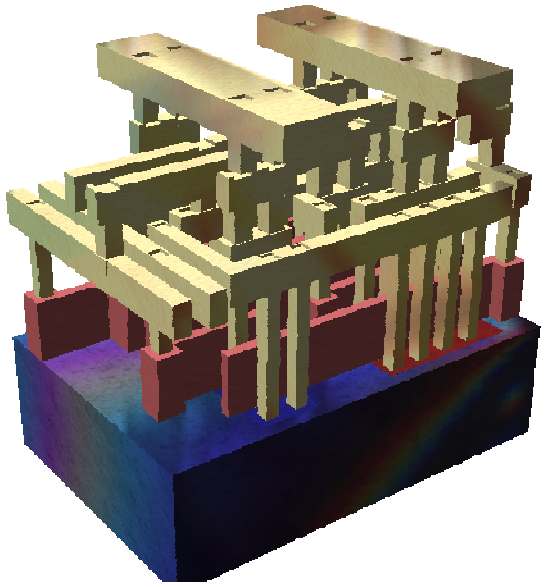


Рис. 1.39. Креслення фрагмента інтегральної схеми, виконаної за планарною технологією, що показує три металеві шари (діелектрик видалено): пісочний колір – металеві з'єднувальні елементами; вертикальні стовпчики – контакти з вольфраму; червонуваті структури – це полікремнієві вентилі, а темно-синя речовина внизу – об'єм кристалічного кремнію

Плівкова інтегральна мікросхема – мікросхема, в якій для виконання елементів використані плівки, які нанесені на поверхню діелектричної підкладки. Плівкові мікросхеми містять лише пасивні елементи, зокрема резистори та конденсатори, оскільки технологія не дозволяє формувати активні елементи належної якості. Товщина плівок може становити 1–2 мкм (тонкоплівкові мікросхеми) або 10–20 мкм (товстоплівкові).

Гібридна інтегральна мікросхема – мікросхема, при виготовленні якої використовувалися різні технології для виготовлення окремих елементів, рис. 1.40. Наприклад, пасивні елементи (резистори, конденсатори, індуктивності) виконані за плівковою технологією на поверхні діелектика, а деякі активні елементи виконані як окремі навісні елементи. Також до гібридних відносять багатокристальні інтегральні схеми.

Поєднана інтегральна мікросхема – мікросхема, що містить напівпровідниковий кристал, в поверхневому шарі якого виконані активні елементи (як у напівпровідниковій мікросхемі), а пасивні елементи виконані на ізольованій поверхні того ж кристалі за допомогою плівок.

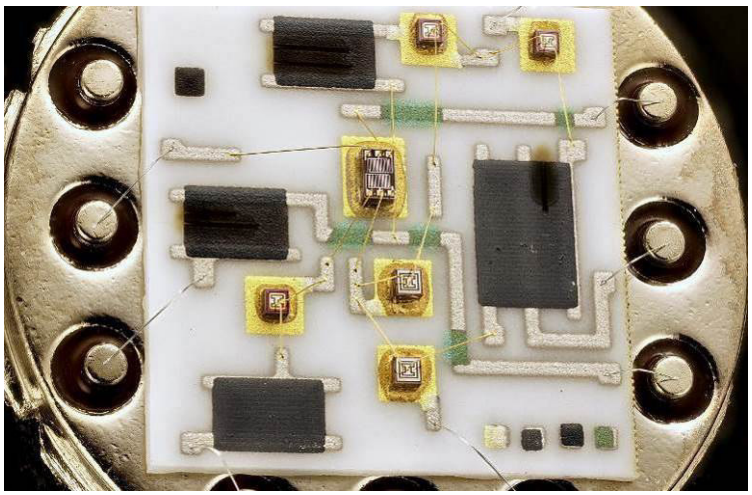
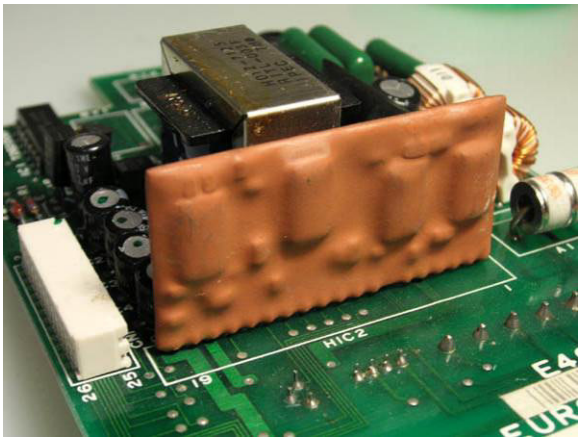


Рис. 1.40. Гібридний операційний підсилювач на керамічній підкладці з товстоплівковими резисторами

Мікрозбірка – мікроелектронний виріб, що включає елементи, компоненти, окремі інтегральні схеми тощо, які з'єднані між собою, рис. 1.41.

а)



б)

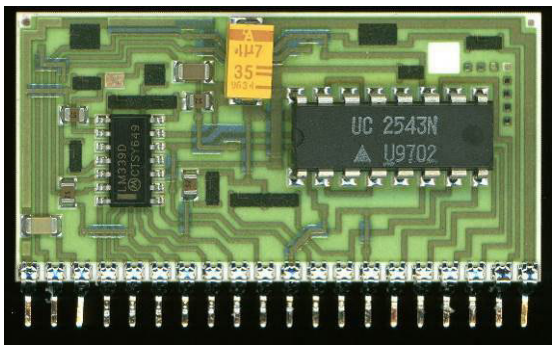


Рис. 1.41. Одноплатна мікрозбірка (*а*),
мікрозбірка з корпусними мікросхемами (*б*)

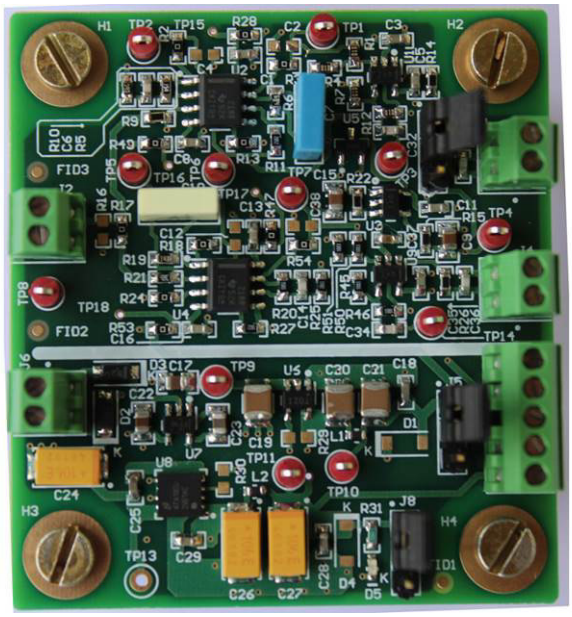
2. За характером виконуваних функцій інтегральні мікросхеми поділяються на два класи: аналогові та цифрові.

Аналогові мікросхеми виконують функції перетворення і оброблення аналогових електричних сигналів, тобто сигналів, що змінюються за законом неперервної функції в діапазоні від додатного до від'ємного рівня напруги живлення.

До аналогових мікросхем відносяться: операційні підсилювачі;
– компаратори;
– генератори сигналів;
– фільтри (у тому числі на п'єзоефекті);
– аналогові помножувачі, атенюатори і регульовані підсилювачі;
– стабілізатори джерел живлення: стабілізатори напруги і струму;
– мікросхеми управління імпульсних блоків живлення;
– перетворювачі сигналів;
– схеми синхронізації;

– різні давачі.

Наприклад, на рис. 1.42 наведено плату TIDA-00777 активного інтегратора підвищеної точності для котушки Роговського, що виготовляється фірмою Texas Instruments. Така плата зібрана з використанням аналогових мікросхем: операційних підсилювачів OPA188, OPA2188, OPA2189; перетворювачів напруги TPS60403, LP2951-N. Така схема, у сукупності з котушкою Роговського, забезпечують вимірювання струму в системах релейного захисту, лічильниках електроенергії, аналізаторах якості тощо.



а)

б)

в)

Рис. 1.42. Плата TIDA-00777 активного інтегратора (*а*)
для котушки Роговського (*б*), аналізатор якості електроенергії Mi550 з котушками Роговського (*в*)

Цифрові мікросхеми призначені для оброблення сигналів, зміна яких відповідає дискретній функції. Сигнали, які обробляються цифровими

мікросхемами, можуть мати два рівня за напругою або струмом. Такі рівні відповідають логічному «0» або логічній «1». Зокрема, для мікросхем з сигналами типу ТТЛ (транзисторно-транзисторна логіка), до яких відносять і мікроконтролери AVR на платах Arduino, при напрузі живлення +5 В діапазон напруг 0–0,4 В відповідає лог. «0», а від 2,4 до 5В – лог. «1».

До цифрових мікросхем відносяться:

– логічні елементи;
– тригери;
– лічильники;
– регістри;
– буферні перетворювачі;
– шифратори;
– дешифратори;
– цифровий компаратор;
– мультиплексори;
– демультиплексори;
– суматори;
– напівсуматори;
– ключі;
– арифметико-логічні пристрої;
– мікроконтролери;
– (мікро) процесори (у тому числі процесори для комп'ютерів);
– однокристальні мікрокомп'ютери;
– мікросхеми і модулі пам'яті;
– ПЛІС (програмовані логічні інтегральні схеми).

До переваг цифрових інтегральних мікросхем порівняно з аналоговими можна віднести:

– *зменшене енергоспоживання*, оскільки польові транзистори у складі цифрових схем працюють у ключовому режимі, споживаючи енергію тільки під час комутацій. Натомість ключові елементи аналогових схем працюють в режимі підсилення, що визначає суттєво більше споживання електроенергії;

– *висока завадостійкість* цифрових мікросхем обумовлена великою розбіжністю в діапазонах сигналів, що відповідають двом логічним рівням. Імовірність розпізнавання високого рівня сигналу як низького та навпаки є надзвичайно низькою. Також кількість помилок передачі даних знижується завдяки застосуванню спеціальних алгоритмів коригування помилок;

– *нечутливість до розкиду значень параметрів елементів*, що пояснюється суттєвою відмінністю між рівнями логічного «0» та логічної «1», позбавляє від необхідності підбору та налаштування пристроїв на основі цифрових мікросхем.

Аналого-цифрові мікросхеми поєднують функції цифрового та аналогового оброблення сигналів. До таких мікросхем відносяться:

– цифро-аналогові (ЦАП) і аналого-цифрові перетворювачі (АЦП);
– цифрові обчислювальні синтезатори (ЦОС);
– трансивери (наприклад, перетворювач інтерфейсу Ethernet);

– модулятори і демодулятори;
– радіомодеми;
– трансивери Fast Ethernet і оптичних ліній;
– приймачі цифрового телебачення;
– перетворювачі напруги живлення;
– цифрові атенюатори;
– схеми фазового автопідстроювання частоти з послідовним інтерфейсом.

В якості прикладу використання таких мікросхем можна навести використання Wi-Fi модуля ESP-12F (ESP8266) у складі релейної плати, що використовується для «розумних» будинків, рис. 1.43. Іншим прикладом є використання мікросхем аналого-цифрових перетворювачів на модулі TIDA-00310 введення-виведення аналогових сигналів для систем релейного захисту розумних електромереж, рис. 1.44.

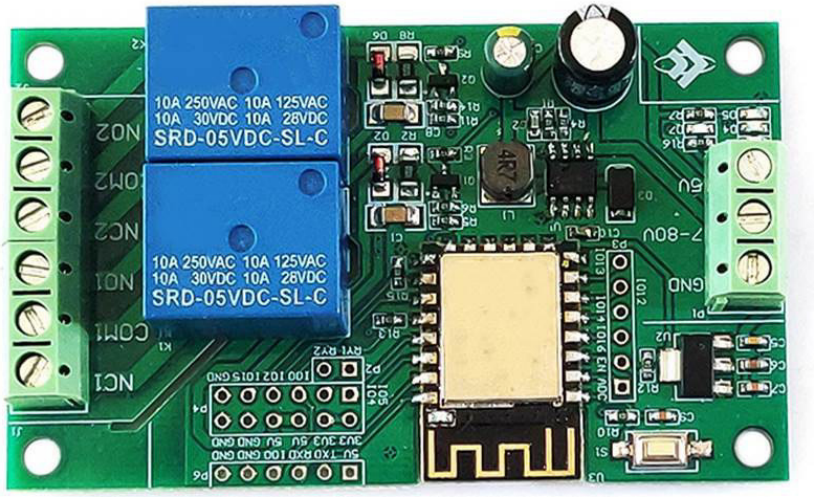


Рис. 1.43. Релейна плата з Wi-Fi модулем ESP-12F (ESP8266)

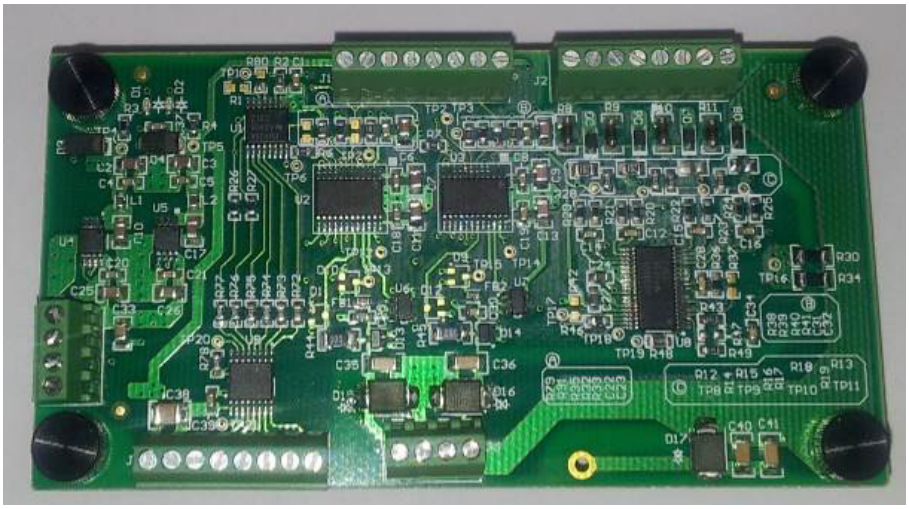


Рис. 1.44. Модуль TIDA-00310 введення-виведення аналогових сигналів

1.2.3. Типи логіки

Залежно від типу електронних компонентів, що використовуються при виготовленні, цифрові мікросхеми поділяються на наступні класи:

– РТЛ (резисторно-транзисторна логіка);
– ДТЛ (діодно-транзисторна логіка);
– ТТЛ (транзисторно-транзисторна логіка);
– ТТЛШ (транзисторно-транзисторна логіка з діодами Шотткі);
– КМОН (комплементарна структура метал-оксид-напівпровідник);
– ЕЗЛ (емітерно-зв'язана логіка).

Резисторно-транзисторна логіка (РТЛ) – технологія побудови мікросхем на основі резисторів та біполярних транзисторів. Широко використовувалася в перших комп'ютерах у 1955–1965 рр. На сьогодні практично не використовується. Розглянемо реалізацію логічного елемента 2АБО-НІ з використанням технології РТЛ, рис. 1.45. Напруги U_1, U_2 відповідають вхідним сигналам, напруга U_3 – вихідному. При високому рівні сигналу на будь-якому з входів, або на обох входах одночасно, відкриваються відповідні транзистори, що забезпечує сигнал низького рівня на виході. В разі сигналу низького рівня на обох входах транзистори знаходяться у закритому стані і сигнал на виході відповідає напрузі живлення (високий рівень).

До переваг логічних елементів РТЛ відноситься низька вартість та конструктивна простота. Недоліки: надзвичайно низька завадостійкість до шумів у вхідних сигналах, висока розсіювана потужність через застосування біполярних транзисторів, низька швидкодія, низька навантажувальна здатність виходів.

Діодно-транзисторна логіка (ДТЛ) – технологія побудови мікросхем з використанням біполярних транзисторів, резисторів та діодів, рис. 1.46. Порівняно з РТЛ, діодно-транзистора логіка дозволяє організувати велику кількість входів логічного елемента. До недоліків відноситься суттєва затримка при зміні логічного стану елемента. Технологія ДТЛ є застарілою і на сьогодні практично не використовується.

Транзисторно-транзисторна логіка (ТТЛ) – технологія виготовлення мікросхем, що передбачає використання біполярних транзисторів для реалізації логічних функцій, інвертування і посилення сигналу на виході. Ця технологія була надзвичайно поширеною, починаючи з 1960-х років і до 1990-х. За допомогою ТТЛ мікросхем виготовляли материнські плати комп'ютерів, контрольно-вимірювальну апаратуру тощо, рис. 1.47. ТТЛ була широко поширеною, тому більшість сучасних цифрових мікросхем виготовляють сумісними за рівнем сигналів з ТТЛ. Базовий елемент транзисторно-транзисторної логіки забезпечує виконання логічної операції І-НІ завдяки використанню багатоемітерного біполярного транзистора, рис. 1.48. Це, порівняно з ДТЛ, підвищує швидкодію та знижує енергоспоживання.

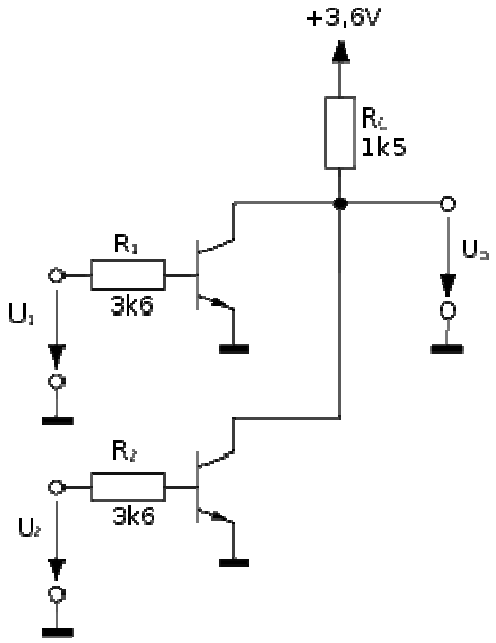


Рис. 1.45. Елемент 2АБО-НІ, виконаний на основі резисторно-транзисторної логіки (мікросхема МС717)

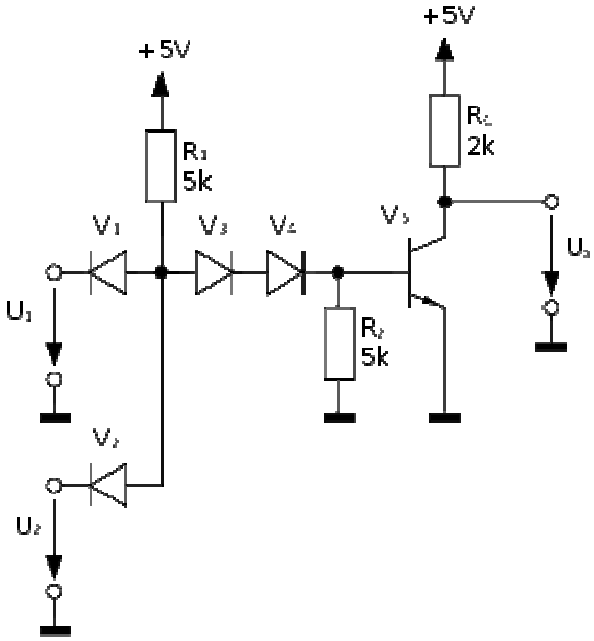


Рис. 1.46. Реалізація за допомогою діодно-транзисторної логіки елемента 2І-НІ

КМОН (КМОН; комплементарна структура метал-оксид-напівпровідник; англ. CMOS, Complementary-symmetry/metal-oxide semiconductor) – найбільш поширена на сьогодні технологія виготовлення цифрових мікросхем. Технологія передбачає використання МДН-транзистори з ізольованим затвором, що мають канали різної провідності (рис. 1.49). CMOS застосовується для побудови більшості цифрових мікросхем, в тому числі – мікропроцесорів, мікроконтролерів. Широке поширення технології обумовлене низьким енергоспоживанням польових транзисторів в статичному режимі. Енергія споживається тільки під час перемикань. Також КМОН-мікросхеми, порівняно з ТТЛ, характеризуються вищою швидкодією.

Рис. 1.47. Цифровий годинник реального часу, виготовлений на основі мікросхем ТТЛ

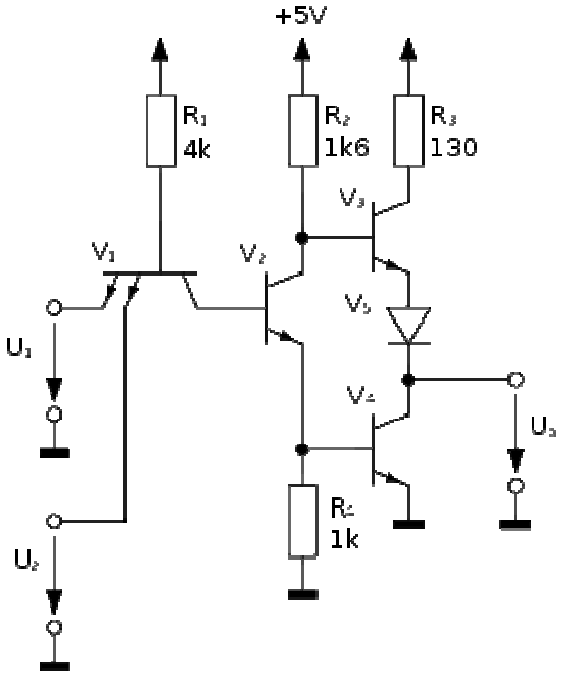


Рис. 1.48. Виконання на основі транзисторно-транзисторної логіки елемента 2І-НІ

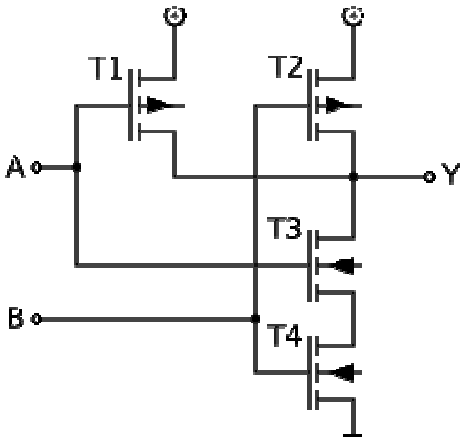


Рис. 1.49. Реалізація за допомогою технології КМОН логічного елемента 2І-НІ

1.2.4. Параметри мікросхем

Цифрові мікросхеми характеризуються рядом параметрів, які описують функціонування в статичному та динамічному режимах, рис. 1.50.

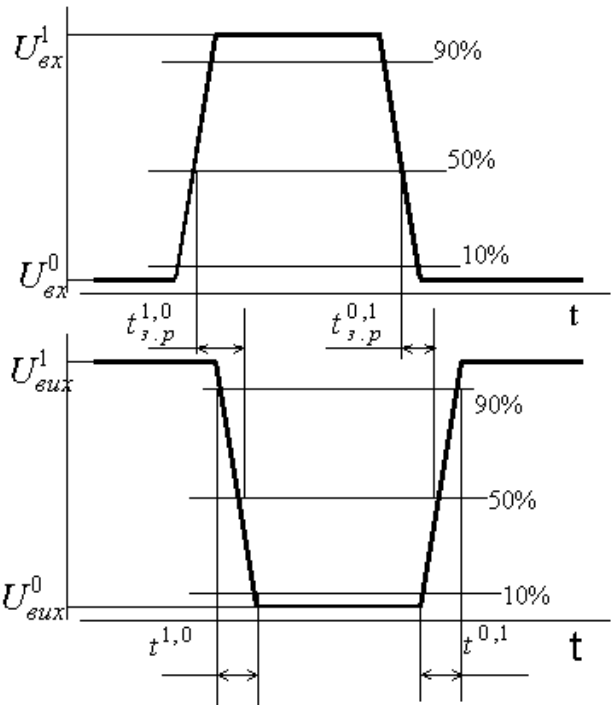


Рис. 1.50. Параметри імпульсів цифрових мікросхем

Статичні параметри характеризують мікросхему за відсутності змін логічного стану. До статичних параметрів відносяться:

– вхідна напруга низького (лог. 0) та високого (лог. 1) рівнів – $U_{вх}^{0}$, $U_{вх}^{1}$;

– вихідна напруга низького (лог. 0) та високого (лог. 1) рівнів – $U_{вих}^{0}$, $U_{вих}^{1}$;

– вхідний $I_{вх}^{0}$, $I_{вх}^{1}$ та вихідний $I_{вих}^{0}$, $I_{вих}^{1}$ струми при напругах відповідно низького та високого рівнів;

– напруга джерела живлення $U_{ж}$;

– коефіцієнт розгалуження по виходу $K_{р}$ (навантажувальна спроможність), який визначає кількість мікросхем-навантажень даної серії, що можна під'єднати до виходу даної мікросхеми;

– статична завадостійкість мікросхеми, що характеризується напругою статичної завади $U_{з}$;

– середня потужність, що споживається мікросхемою, визначається співвідношенням:

$$P_{спож.сер} = U_{ж} \frac{I_{спож}^{0} + I_{спож}^{1}}{2}.$$

Динамічні параметри описують властивості мікросхеми під час змін логічного стану, в режимі перемикання:

– динамічна потужність, що споживається мікросхемою;

– часова затримка проходження сигналу при перемиканні з «0» в «1» $t_{з.р}^{0,1}$ та з «1» в «0» $t_{з.р}^{1,0}$;

– середній час проходження сигналу через мікросхему від входів до виходів $t_{з.сер}$ (дорівнює півсумі затримок $t_{з.р}^{0,1}$ та $t_{з.р}^{1,0}$);

– динамічна завадостійкість, що оцінюється амплітудою імпульсу завади на вході мікросхеми, що не призводить до виходу за встановлені межі вихідного сигналу.

На основі логічних елементів конструюють логічні пристрої. Такі пристрої можуть бути частиною більш складних схем – мікропроцесорних. Також логічні пристрої випускаються у вигляді готових мікросхем. Розглянемо найбільш поширені логічні пристрої.

1.2.5. Шифратор

Шифратор (кодер, encoder) – логічний пристрій, який забезпечує перетворення позиційного *n*-розрядного коду в *m*-розрядний двійковий код.

Коли лог. «1» подається на один з входів шифратора (за умови лог. «0» на інших входах) на виході з'являється двійковий код, який відповідає номеру активного входу.

Повний шифратор забезпечує відповідність входів кожній з можливих комбінацій вихідних сигналів. Якщо кількість входів повного шифратора позначити *n*, а кількість виходів – *m*, то виконується співвідношення $n = 2^{m}$.

Неповний шифратор має входів менше, ніж вихідних двійкових комбінацій. На рис. 1.51 наведено умовне позначення неповного шифратора з 10 входами та 4 виходами. У табл. 1.5 наведена таблиця істинності такого шифратора. Функціональна схема такого шифратора, яка показує його внутрішнє улаштування, наведена на рис. 1.52. Зі схеми видно, що шифратор побудовано на основі логічних елементів НІ, І, АБО.

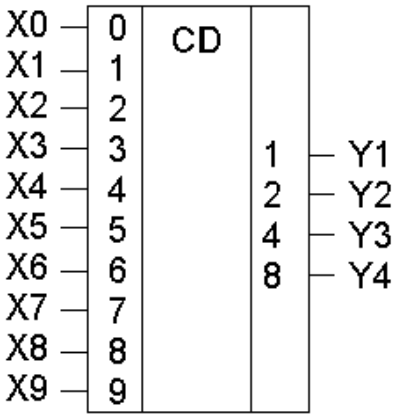


Рис. 1.51. Умовне графічне позначення шифратора

Таблиця 1.5

Таблиця істинності шифратора з 10 входами

Стан входів										Стан виходів			
X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	Y4	Y3	Y2	Y1
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	**1**	0	0	0	0	0	0	0	0	0	0	0	1
0	0	**1**	0	0	0	0	0	0	0	0	0	1	0
0	0	0	**1**	0	0	0	0	0	0	0	0	1	1
0	0	0	0	**1**	0	0	0	0	0	0	1	0	0
0	0	0	0	0	**1**	0	0	0	0	0	1	0	1
0	0	0	0	0	0	**1**	0	0	0	0	1	1	0
0	0	0	0	0	0	0	**1**	0	0	0	1	1	1
0	0	0	0	0	0	0	0	**1**	0	1	0	0	0
0	0	0	0	0	0	0	0	0	**1**	1	0	0	1

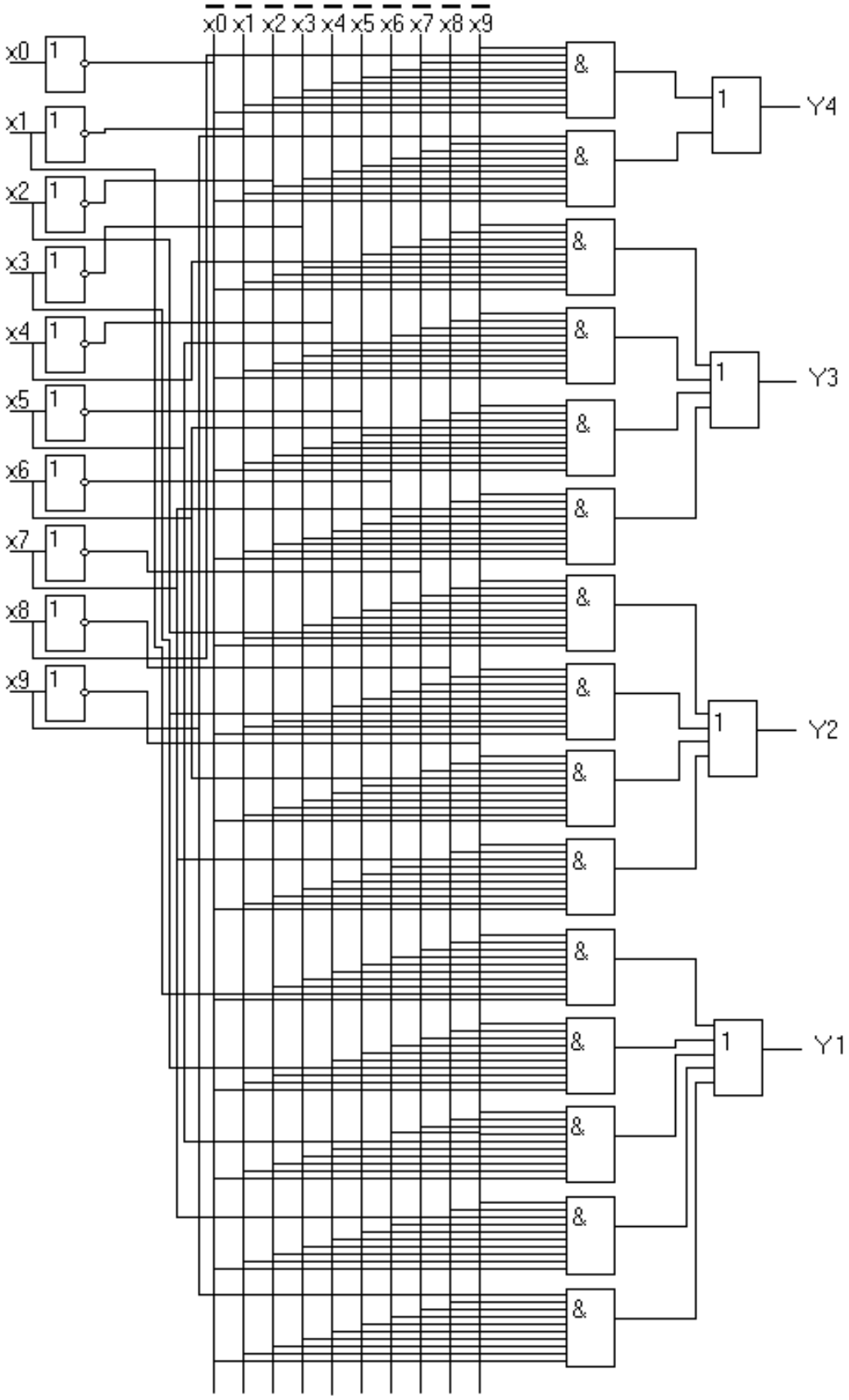


Рис. 1.52. Функціональна схема шифратора з 10 входами

1.2.6. Дешифратор

Дешифратор (декодер, decoder) – логічний пристрій, що призначений для перетворення двійкового вхідного коду у однопозиційний вихідний код.

Логічна «1» присутня тільки на тому виводі дешифратора, номер якого визначається вхідним двійковим кодом. Інші виводи переводяться у третій (високоімпедансний) стан, або на них присутня лог. «0». Для повного дешифратора кількість входів (n) та кількість виводів (m) пов'язано залежністю $m = 2^n$. Зокрема, дешифратор з 3 входами та 8 виходами є повним (рис. 1.53, табл. 1.6).

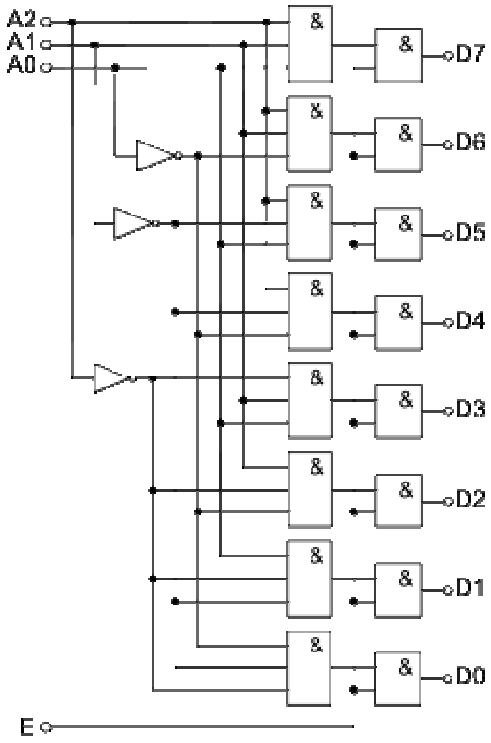


Рис. 1.53. Повний дешифратор з трьома входами адреси і входом дозволу на 8 виводів (2^3)

Таблиця 1.6

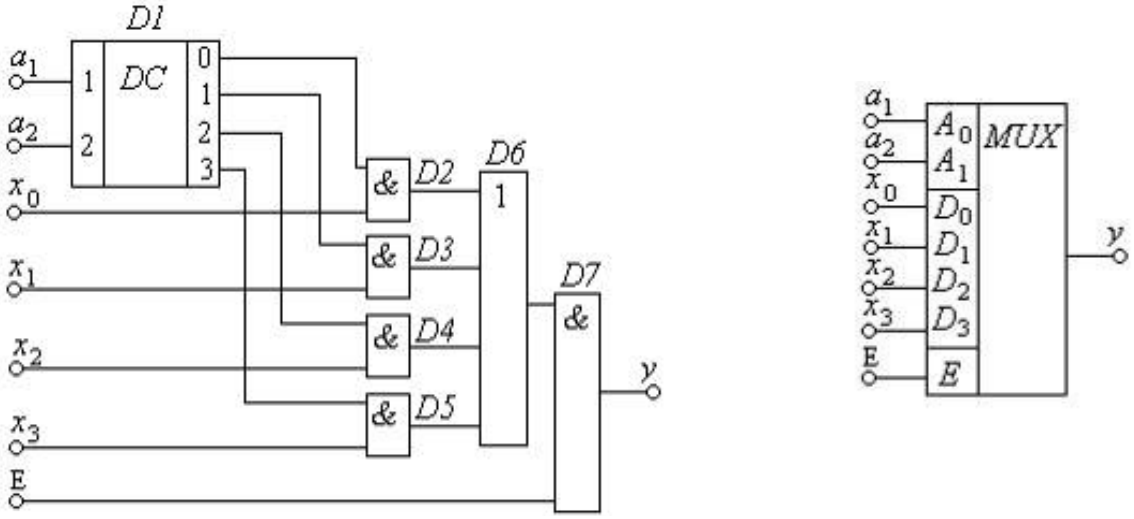
Таблиця істинності повного дешифратора 2^3

Адреса			Номер активного виводу	Стан виходів при E=1							
A_2	A_1	A_0		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	0	**1**
0	0	1	1	0	0	0	0	0	0	**1**	0
0	1	0	2	0	0	0	0	0	**1**	0	0
0	1	1	3	0	0	0	0	**1**	0	0	0
1	0	0	4	0	0	0	**1**	0	0	0	0
1	0	1	5	0	0	**1**	0	0	0	0	0
1	1	0	6	0	**1**	0	0	0	0	0	0
1	1	1	7	**1**	0	0	0	0	0	0	0

1.2.7. Мультиплексор

Мультиплексор являє собою цифровий комутатор. Він має декілька інформаційних входів, адресні входи та тільки один вихід. Кожен з інформаційних входів має свій номер, який називається адресою. На адресні входи подається у двійковому форматі номер (адреса) інформаційного входу, який цікавить користувача. Мільтиплексор підключає цей інформаційних вхід до єдиного виходу пристрою. При зміні коду на адресних входах, змінюється і інформаційний вхід, інформація з якого подається на вихід мільтиплексора.

Розглянемо мультиплексор 4–1, схема та умовне позначення якого наведені на рис. 1.54. Входи a_1, a_2 – адресні. Входи x_0–x_3 – інформаційні. E – вхід дозволу роботи мікросхеми. Вихід мікросхеми позначено y. Якщо на вхід дозволу роботи подано сигнал низького рівня (E=0), то на виході буде y=0 незалежно від сигналів на входах. Такий режим в таблиці істинності (табл. 1.7) відповідає першому рядку. Для того, щоб такий пристрій функціонував, необхідно подати сигнал високого рівня на вхід дозволу роботи: E=1. Мікросхема має 4 входи. Відповідно, адреса входу x_0 становить 00b та відповідає сигналам на адресних входах: a_1=0; a_2=0. В такому разі логічний рівень сигналу на виході буде дорівнювати сигналу на вході x_0, а саме: y=x_0. При зміні адреси, сигнал на виході повторюватиме сигнал на відповідному вході, табл. 1.7.



а) *б)*

Рис. 1.54. Мультиплексор 4–1; *а* – схема; *б* – умовне позначення

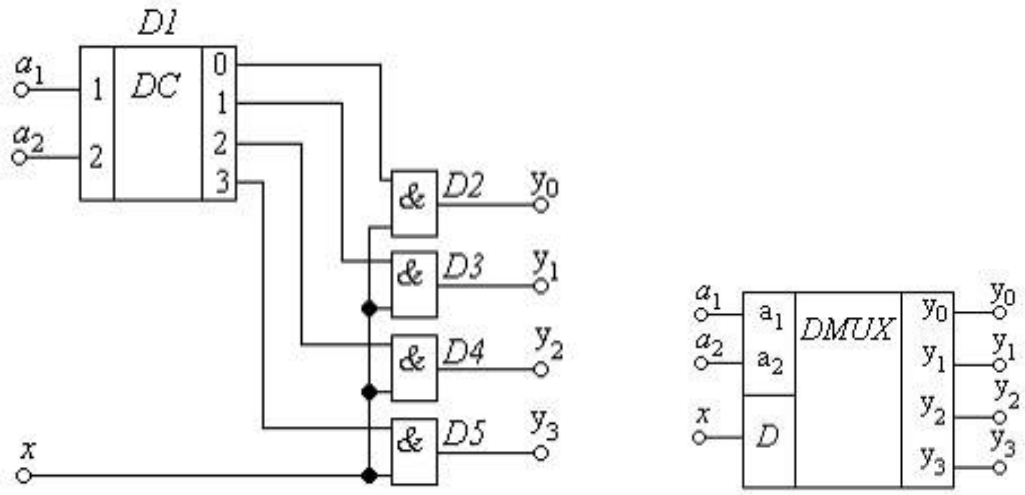
Таблиця 1.7

Таблиця істинності мультиплексора 4–1

Адресні входи		Керуючий вхід E	Вихід y дорівнює
a_1	a_2		
–	–	0	0
0	0	1	x_0
1	0	1	x_1
0	1	1	x_2
1	1	1	x_3

1.2.8. Демультиплексор

Демультиплексор забезпечує передачу логічного сигналу з єдиного входу до обраного виходу. Вихід обирається за адресою, яка задається двійковим кодом на адресних входах. Дешифратор, зображений на рис. 1.55, має 2 адресні входи, 1 інформаційний вхід та 4 інформаційні виходи. Функціонування такого пристрою пояснюється таблицею істинності, табл. 1.8.



а) *б*)

Рис. 1.55. Схема (*а*) та умовне позначення (*б*) демультиплексора

Таблиця 1.8

Таблиця істинності демультиплексора

Адресні входи		Виходи			
a_1	a_2	y_0	y_1	y_2	y_3
0	0	x	0	0	0
1	0	0	x	0	0
0	1	0	0	x	0
1	1	0	0	0	x

1.2.9. Принцип дії тригера. Різновиди тригерів

Тригер – елементарна комірка пам’яті, що призначена для зберігання одного біта інформації. Відповідно, тригер може знаходитися в одному з двох станів. Один стан позначають станом лог. «1», інший стан – лог. «0». Тригер змінює логічний стан під випливом зовнішніх сигналів. Тригер є складовою частиною більш складних пристроїв запам’ятовування інформації.

Розрізняють інформаційні та керуючі входи тригера. Перші використовуються для переведення тригера в потрібний стан. За допомогою керуючих входів здійснюється попередньої установки тригера в необхідний логічний стан, також може забезпечуватися синхронізація. Тригера мають два виходи. Сигнал на прямому виході Q відповідає логічному стану тригера. Сигнал на інверсному виході $\overline{Q}$ відповідає інверсному стану тригера.

Тригери класифікуються наступним чином.

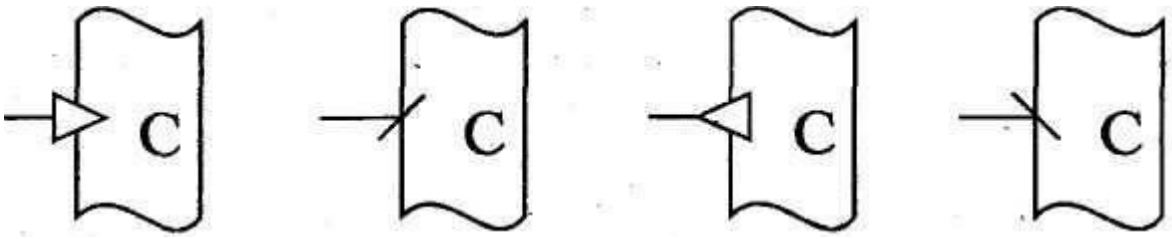
1. За моментом зміни логічного стану:

1.1. Асинхронні тригери, що змінюють стан безпосередньо при зміні величини інформаційного сигналу.

1.2. Синхронні тригери, що змінюють свій стан в разі зміни інформаційного сигналу за наявності сигналу синхронізації *C* (строб-сигналу, тактового сигналу). В разі відсутності сигналу синхронізації стан тригера не змінюється. Залежно від характеру реакції на сигнал синхронізації розрізняють статичні та динамічні синхронні тригери.

1.2.1. Статичні синхронні тригери функціонують за наявності лог. «1» на вході синхронізації *C* (прямий вхід), або за наявності лог. «0» (інверсний вхід). Тобто такі тригери реагують на статичний рівень сигналу синхронізації. Це означає, що тригер реагує на зміну стану інформаційних сигналів до тих пір, поки на вході синхронізації присутній сигнал активного рівня. За кількістю тактів статичні тригери діляться на однотактові та двотактові. Однотактові тригери мають один ступінь запам'ятовування інформації. Стан, що визначається вхідним інформаційним сигналом, зразу з'являється на виході тригера. В двотактових тригерах (позначаються «ТТ») інформація на спочатку записується до тригера і тільки на другому такті з'являється на виході.

1.2.2. Динамічні синхронні тригери забезпечують сприйняття сигналів на інформаційних входах за переднім або заднім фронтом сигналу синхронізації. В першому випадку вхідна інформація запам'ятовується при зміні сигналу на вході *C* від лог. «0» до лог. «1» (прямий динамічний вхід синхронізації). Тригер реагує на задній фронт імпульсу синхронізації, якщо спрацьовує при зміні сигналу на вході *C* від лог. «1» до лог. «0» (інверсний динамічний вхід синхронізації), рис. 1.56.



а) *б*)

Рис. 1.56. Позначення динамічних входів синхронізації:
а – прямий динамічний вхід (передній фронт),
б – інверсний динамічний вхід (задній фронт)

2. За функціональними можливостями тригери поділяють на:
– *RS*-тригери (з роздільним встановленням станів «0» і «1»);
– *JK*-тригери (універсальні);
– *D*-тригери (тригери затримки);
– *T*-тригери (з рахунковим входом).

Для входів тригерів використовуються наступні позначення:

S (set) – встановлення в стан лог. «1»;
R (reset) – встановлення в стан лог. «0»;
J (jump) – встановлення в стан лог. «1» в універсальному тригері;
K (keep) – встановлення в стан лог. «0» в універсальному тригері;
D (delay) – встановлення в стан лог. «1» або в стан лог. «0»;
T (toggle) – рахунковий (загальний) вхід;
E або *V* – додатковий керуючий вхід для дозволу прийому інформації.

Розглянемо принцип дії тригерів деяких типів.

Асинхронний RS-тригер має два входи: *S* – встановлення в лог. «1»; *R* – скидання в лог. «0», рис. 1.57. Для встановлення тригера в стан лог. «1» необхідно подати короткочасний імпульс на вхід *S*. Для скидання тригера в «0» необхідно подати короткочасний імпульс на вхід *R*. Одночасно подавати імпульси на обидва входи *S* та *R* заборонено, оскільки неможливо передбачити, в якому стані перебуватиме тригер. В таблиці істинності асинхронного *RS*-тригера (табл. 1.9) змінні S^t, R^t, Q^t позначають значення відповідних логічних сигналів в момент часу *t*, Q^{t+1} – значення вихідного моменту сигналу в наступний момент часу *t*+1.

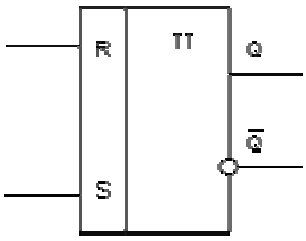


Рис. 1.57. Умовне позначення асинхронного *RS*-тригера

Таблиця 1.9

Таблиця істинності асинхронного RS-тригера

S^t	R^t	Q^{t+1}	Режим
0	0	Q^t	Зберігання
1	0	1	Встановлення лог. «1»
0	1	0	Встановлення лог. «0»
1	1	-	Невизначеність

Для реалізації асинхронного *RS*-тригера можуть бути використані два елементи АБО-НІ, рис. 1.58.

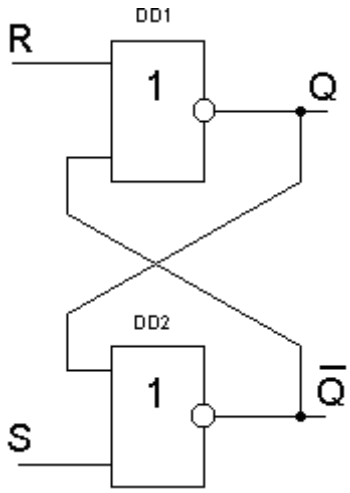


Рис. 1.58. Реалізація асинхронного *RS*-тригера за допомогою двох елементів АБО-НІ

Синхронний RS-тригер. Відрізняється від асинхронного *RS*-тригера наявністю входу синхронізації *C*, рис. 1.59. При сигналі *C*=0 тригер знаходиться в режимі зберігання інформації, зміна рівнів сигналів на входах *S* та *R* не призводить до зміни логічного стану пристрою. При *C*=1 схема функціонує відповідно до таблиці істинності, табл. 1.10.

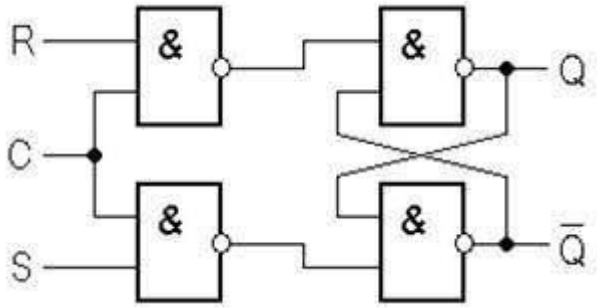


Рис. 1.59. Реалізація синхронного RS-тригера

Таблиця 1.10

Таблиця істинності синхронного RS-тригера

R^t	S^t	Q^{t+1}	C
1	0	0	1
0	1	1	1
0	0	Q^t	1
1	1	–	1

JK-тригер функціонує аналогічно до синхронного *RS*-тригера, причому для встановлення тригера в стан лог. «1» використовується вхід *J*, а для скидання в лог. «0» – вхід *K*, рис. 1.60. До переваг *JK*-тригера відноситься відсутність заборонений комбінацій. В разі подачі на обидва інформаційні входи лог. «1» тригер змінює стій стан на інверсний.

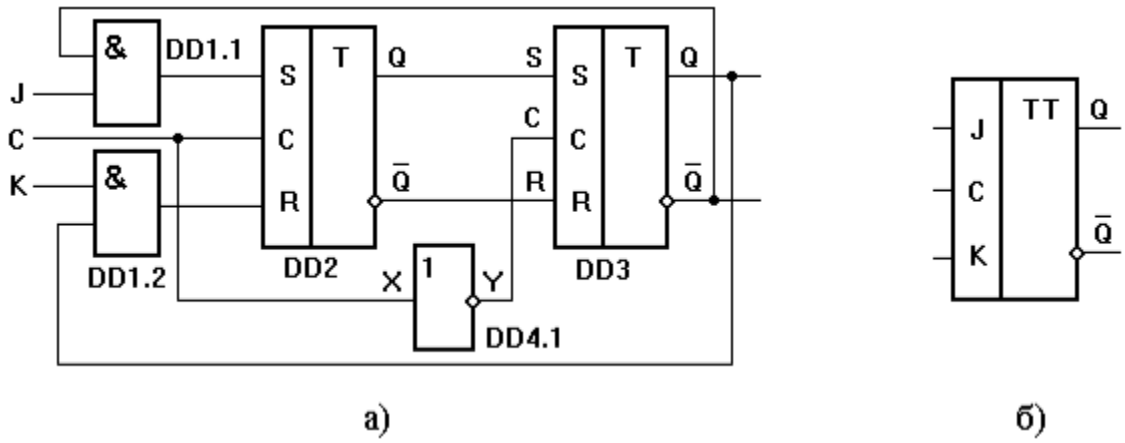


Рис. 1.60. Схема (а) та умовне позначення (б) двоступінчатого JK-тригера

D-тригер забезпечує повторення на виході стан інформаційного входу *D* за наявності сигналу синхронізації С, рис. 1.61. Стан тригера лишається незмінним до подачі нового імпульсу синхронізації, тому D-тригер називають запам’ятовуючим, табл. 1.11.

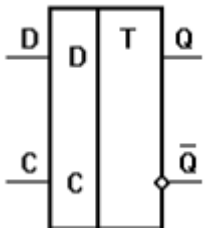


Рис. 1.61. D-тригер

Таблиця 1.11

Таблиця істинності D-тригера

D	Q^t	Q^{t+1}
0	0	0
0	1	0
1	0	1
1	1	1

1.2.10. Регістри

Регістр – запам’ятовуючий пристрій, що забезпечує приймання, запам'ятовування та видачу інформації. Регістри забезпечують збереження значень декількох бітів (слова). Будуються на основі тригерів, найчастіше використовуються *RS*-, *D*- або *JK*-тригери. Спосіб з’єднання тригерів визначає спосіб запису та зчитування інформації. Одночасний запис (зчитування) декількох бітів визначає паралельний спосіб передачі інформації. Послідовний спосіб передбачає обмін бітами послідовно у часі.

Окрім збереження інформації, регістри можуть виконувати додаткові операції. Поширеним є виконання регістрами наступних функцій:

– приймання інформації до регістра;

– передавання збереженої у регістрі інформації;

– виконання порозрядних логічних операцій;

– зсув слова вліво або вправо на визначену кількість розрядів;

– перетворення послідовного коду слова в паралельний і навпаки;

– скидання регістра (встановлення в початковий стан).

Розрізняють накопичувальні регістри (регістри пам’яті) та регістри зсуву. Регістри зсуву відповідно до способу введення-виведення даних поділяють на паралельні, послідовні та комбіновані. За напрямом передачі даних вони можуть бути однонаправлені або реверсивні.

В якості прикладу розглянемо накопичувальний регістр з паралельним введенням і виведенням інформації, що побудований на основі *D*-тригерів, рис. 1.62. Логічний сигнал Y_1, що подається до входів синхронізації тригерів, керує записом інформації до тригера (Y_1=1 – запис дозволено). За допомогою логічного сигналу Y_2, що подається до других входів логічних елементів AND на виході пристрою, забезпечується читання збереженого слова (Y_2=1 – читання дозволено). Нульовий сигнал на обох входах керування (Y_1=Y_2=0 відповідає режиму зберігання інформації).

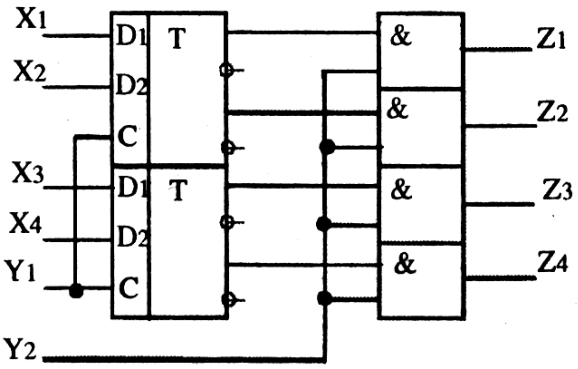


Рис. 1.62. Накопичувальний регістр з паралельним введенням і виведенням інформації: Y_1=1 – паралельний запис; Y_1=Y_2=0 – зберігання; Y_2=1 – паралельне читання

На рис. 1.63 наведена схема регістра зсуву, побудованого на основі *D*-тригерів з прямими динамічними входами синхронізації. Введення даних до такого регістра здійснюється послідовно через єдиний інформаційний вхід *X*. Зчитування чергового біта здійснюється за переднім фронтом сигналу синхронізації *C*. Виведення збереженого слова здійснюється в паралельному форматі за допомогою виводів *Q*1–*Q*4. За кожним тактовим сигналом записаний біт зсувається (перезаписується) до наступного тригера, табл. 1.12.

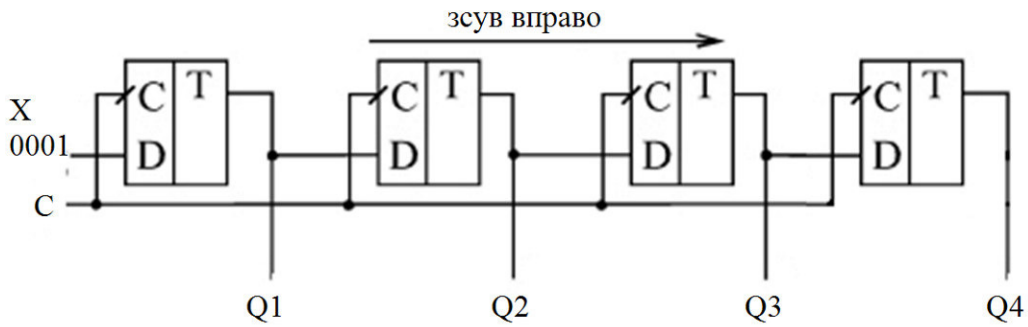


Рис. 1.63. Регістр зсуву

Таблиця 1.12

Таблиця істинності регістра зсуву

№ такту	Дані	Q1	Q2	Q3	Q4
0	0	0	0	0	0
1	1	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	0
4	0	0	0	0	1

1.2.11. Питання для самоперевірки

1. Що таке чип?
2. Дайте визначення мікросхемі, назвіть її функції.
3. Чим відрізняються елемент та компонент мікросхеми?
4. Які методи корпусування мікросхем Вам відомі?
5. Опишіть особливості корпусу типу DIP мікросхем.
6. Як нумеруються виводи мікросхеми?
7. Опишіть особливості корпусів типів SOIC, TQFP, PLCC мікросхем.
8. Які переваги корпусів LGA, BGA мікросхем?
9. Чим відрізняються напівпровідникова, плівкова та гібридна інтегральні мікросхеми?
10. Для роботи з якими сигналами призначені аналогові мікросхеми? Наведіть приклади.
11. Опишіть область застосування цифрових мікросхем. Які переваги цифрових мікросхем порівняно з аналоговими Вам відомі? Наведіть приклади.
12. Які типи логіки цифрових мікросхем Вам відомі?
13. Опишіть мікросхеми на основі резисторно-транзисторної логіки.
14. Які особливості ДТЛ-мікросхем?
15. Охарактеризуйте ТТЛ-мікросхеми.
16. В чому переваги мікросхем, виготовлених за технологією КМОН?
17. Назвіть статичні параметри, що характеризують функціонування мікросхеми?
18. Які параметри характеризують функціонування мікросхеми в динамічному режимі?
19. Поясність принцип дії шифратора та дешифратора.
20. Яким чином функціонують мультиплексор та демультиплексор?
21. В чому полягає принцип дії тригера? Які різновиди тригерів Вам відомі?
22. Розкрийте принцип дії асинхронного *RS*-тригера.
23. Які особливості роботи синхронного *RS*-тригера?
24. Опишіть роботу *JK*-тригера.
25. Як працює *D*-тригер?
26. Розкрийте призначення регістра. Які основні операції регістри можуть виконувати?

1.3. Запам’ятовуючі пристрої. Арифметико-логічний пристрій. Аналоговий компаратор. Аналого-цифрові та цифро-аналогові перетворювачі

Класифікація та області застосування запам’ятовуючих пристроїв. Оперативні запам’ятовуючі пристрої. Постійні запам’ятовуючі пристрої. Організація та принцип дії арифметико-логічного пристрою. Набір операцій арифметико-логічного пристрою. Принцип дії аналогового компаратора. Аналого-цифрові перетворювачі. Основні параметри АЦП. Принцип дії паралельних АЦП. Багатоступінчаті АЦП. Цифро-аналогові перетворювачі. ЦАП на джерелах струму.

1.3.1. Класифікація та області застосування запам’ятовуючих пристроїв

Матеріальний об’єкт (середовище), що забезпечує зберігання даних, називається носій інформації. Носії інформації класифікуються наступним чином.

1. За формою сигналу, який застосовується для запису даних, носії інформації бувають аналоговими та цифровими.

1.1. Аналогові носії інформації – забезпечують збереження інформації у формі аналогового (неперервного) сигналу. Найбільш відомими аналоговим носієм є магнітна стрічка, що розміщувалася у бобіннах або компакт-касетах, рис. 1.64. Такі носій широко використовувалися у 1950-х–1960-х рр.

а) *б*)

Рис. 1.64. Аналогові носії інформації – бобінна та магнітофонна касети (*а*) накопичувач IBM 729 Magnetic Tape Unit, 1950–1960 рр. (*б*)

Останніми роками метод збереженні інформації на магнітну стрічку одержав друге життя із розробленням у 1998 р. першої версії стандарту LTO (Linear Tape-Open). На сьогодні стрічкові накопичувачі дозволяють зберігати більше 10 ТБ даних у цифровому форматі. Наприклад, покоління накопичувачів LTO-9, випущене 2021 р., дозволяє зберігати 18 ТБ нестиснених даних, а у стисненому вигляді – до 45 ТБ (1 ТБ = 1024 ГБ), рис. 1.65. Товщина стрічки становить 5,2 мкм, довжина – 1035 м.

Рис. 1.65. Стрічковий накопичувач HPE LTO-9 Ultrium Tape (Q2079A) фірми Hewlett Packard Enterprise ємністю 18/45 ТБ

1.2. Цифрові носії інформації – забезпечують запис, збереження та видачу інформації в цифровому коді. Матеріальне виконання цифрових накопичувачів змінювалося із розвитком цифрових технологій.

1.2.1. Перфокарта (punched card) – носій інформації, виготовлений з прямокутного картону. Збереженні інформації здійснювалося шляхом пробивання отворів в певних позиціях карти. Перші перфокарти мали 12 рядків та 45 стовпчиків, пізніші – 12x80 позицій, рис. 1.66. Використовувалися до 1980х років для збереження кодів програм, в системах автоматизованого оброблення даних.

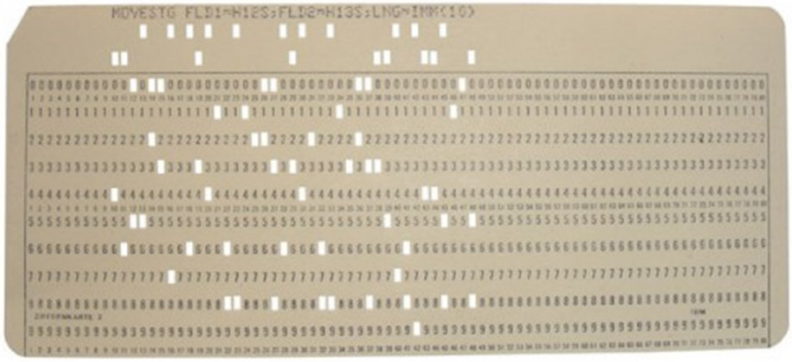


Рис. 1.66. Перфокарта з пробитими отворами

1.2.2. Перфострічка – стрічка паперу або іншого гнучкого матеріалу, в якій пробито (перфоровано) отвори, рис. 1.67. Принцип дії аналогічний перфокарті. Зчитування інформації відбувалося за допомогою фотоелементів. В середині перфострічки розміщувалася транспортна доріжка – отвори через рівномірний інтервал, що використовувалися для просування стрічки.

Рис. 1.67. Перфострічка

1.2.3. Дискета (гнучкий магнітний диск) – носій інформації, що представляє собою диск, виготовлений з магнітного композиційного матеріалу та розташований у захисному корпусі, рис. 1.68. На диску виконана неферомагнітна матриця, в якій розташовані видовжені феромагнітні частинки. Для зчитування та запису інформації використовуються магнітні головки, які притискаються до диску з обох боків. Намагнічена ділянка відповідає лог. «1», ненамагнічена – лог. «0». Найбільш поширеним розміром диску був 3,5 дюйми, ємність – 1,44 МБ. Дискети широко використовувалися з початку 1970-х р. до початку 2000-х років.

Рис. 1.68. Цифрові носії інформації: дискета, компакт-диски, карти пам’яті

1.2.4. Компакт-диск – оптичний носій інформації, являє собою полікарбонатний диск, запис та зчитування інформації на який здійснюється за допомогою лазера, рис. 1.68. Лазер записує інформацію за допомогою заглиблень (пітів, від англ. pit – заглиблення). Кожна заглиблення має наступні розміри: глибина 100 нм, ширина 500 нм, довжина – від 850 нм до 3,5 мкм, рис. 1.69. Крок доріжок спіралі становить 1,6 мкм. Заглиблення поглинають або розсіюють світло, а основа – відбиває. Компакт-диски бувають наступних видів: CD-ROM (Compact Disc Read Only Memory) – компакт-диск, записаний

на заводі, користувач може тільки зчитувати дані; CD-R (Compact Disc-Recordable) – компакт диск, що продається без запису, користувач може однократно записати інформацію (записані дані стерти неможливо, проте інформацію можна записувати частинами); CD-RW (Compact Disc-ReWritable) – компакт-диск, інформацію на якому можна стирати та записувати повторно. Ємність CD спочатку становила 650 МБ, починаючи з 2000х років ємність було збільшено до 700 МБ. Також випускалися диски DVD (Digital Versatile Disc) – оптичні носії інформації збільшеної, порівняно з CD, ємністю – 4,7 ГБ.

Компакт-диски широко використовувалися з 1980-х до 2010-х років. За даними компанії Philips, за 25 років активного використання в світі було продано більше 200 млрд штук.

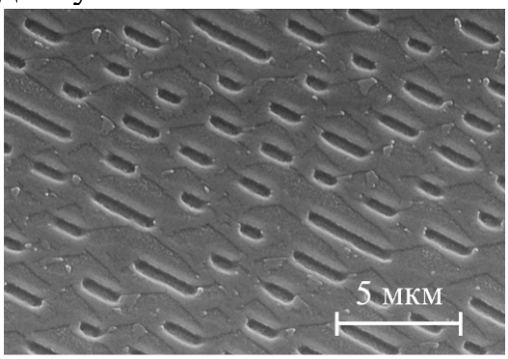


Рис. 1.69. Заглиблення (піти) в полікарбонатній основі компакт-диска під мікроскопом

1.2.5. Карта пам'яті – електронний носій інформації, виготовлений на основі флеш-пам'яті, рис. 1.68. Використовується в багатьох цифрових пристроях, в тому числі – для реєстрації результатів вимірювання електричних величин, рис. 1.70.

Рис. 1.70. Реєстратор електричних параметрів РПМ-416 (виробник – ТОВ «Новатек-Електро», м. Одеса) дозволяє записувати виміряні миттєві значення величин на карту пам’яті (SD/MMC)

2. За стійкістю запису і можливістю перезапису запам’ятовуючі пристрої поділяють на:

2.1. *Постійні запам'ятовуючі пристрої* (ПЗП, Read Only Memory – ROM), що допускають тільки зчитування інформації, вміст цих пристроїв не може бути модифікований користувачем (наприклад, диски CD-ROM, DVD-ROM, що записані під час виготовлення).

2.2. *Записувані пристрої*, зазичай це чисті носії, до яких користувач може записати дані тільки один раз (наприклад, диски CD-R, DVD-R, DVD+R, BD-R).

2.3. *Перезаписувані пристрої*, вміст яких користувач може змінювати (наприклад, CD-RW, DVD-RW, DVD+RW, BD-RE, магнітна стрічка тощо).

2.4. *Оперативні пристрої*, що забезпечують запису, зберігання та зчитування даних під час оброблення.

1.3.2. Оперативні запам’ятовуючі пристрої

Оперативний запам’ятовуючий пристрій (ОЗП, Random Access Memory – RAM) поділяються на статичні і динамічні.

Статичні ОЗП (Static Random Access Memory – SRAM) побудовані з використанням тригерів, рис. 1.71. Для виготовлення цих тригерів застосовують польові, рідше – біполярні транзистори. Використання технології КМОН (CMOS) при побудові статичних ОЗП визначає керування пам’яттю за допомогою потенційних сигналів. Записана інформація зберігається тригерами скільки завгодно довго, поки на мікросхему подається живлення. Збережена інформація може бути зчитана без ушкодження.

Рис. 1.71. Статичний ОЗП типу AS6C4008-55SIN, обсяг пам’яті 4 Мб, організація пам’яті 512 К * 8, корпус SOIC-32, тип логіки CMOS

В якості прикладу статичного ОЗП можна розглянути мікросхему ОЗП 7489, рис. 1.72. Така мікросхема є виконана на основі ТТЛ-структур, має ємність 64 біт. Структура пам’яті 16х4, тобто може зберігати 16 слів завдовжки 4 розряди кожне. Виводи мікросхеми мають наступне призначення:

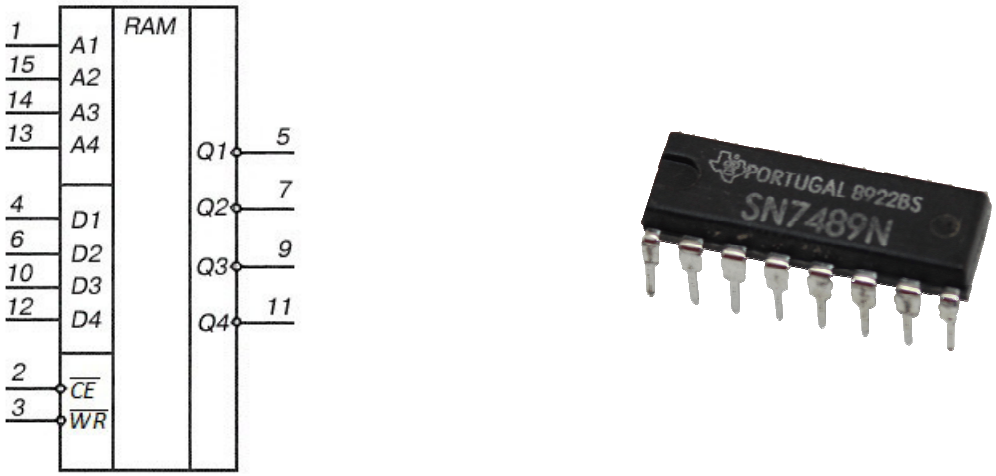
A1–A4 – адресні входи;

D1–D4 – входи даних;

$\overline{CE}$ – сигнал дозволу;

$\overline{WR}$ – дозвіл запису/читання;

Q1–Q4 – інформаційні виходи.



а) *б*)

Рис. 1.72. Мікросхема ОЗП 7489

Динамічні ОЗП (Dynamic Random Access Memory – DRAM) складаються з ємностей (зокрема, затвора польового транзистора), заряджений або розряджений стан кожної з яких відповідає біту інформації. До недоліків такого способу побудови пам’яті відноситься періодичне саморозрядження ємностей, тому динамічні ОЗП потребують періодичної регенерації, тобто підзаряджання ємностей, що почали розряджатися.

Існує багато видів динамічних ОЗП. В 1990х роках використовувалися наступні види: FPM RAM (Fast Page Mode Random Access Memory) – динамічний ОЗП, який підтримує швидкий сторінковий режим; EDO RAM (Extended Data Out Random Access Memory) – динамічний ОЗП, який має розширений вивід даних; SDR SDRAM (Single Data Rate, Synchronous Dynamic Random Access Memory) – динамічний ОЗП з одиничною швидкістю роботи. Також поширеними є динамічні ОЗП типу DDR (Double Data Rate), що забезпечують подвійну швидкість передачі даних за рахунок використання обох фронтів тактового сигналу для обміну даними, рис. 1.73.

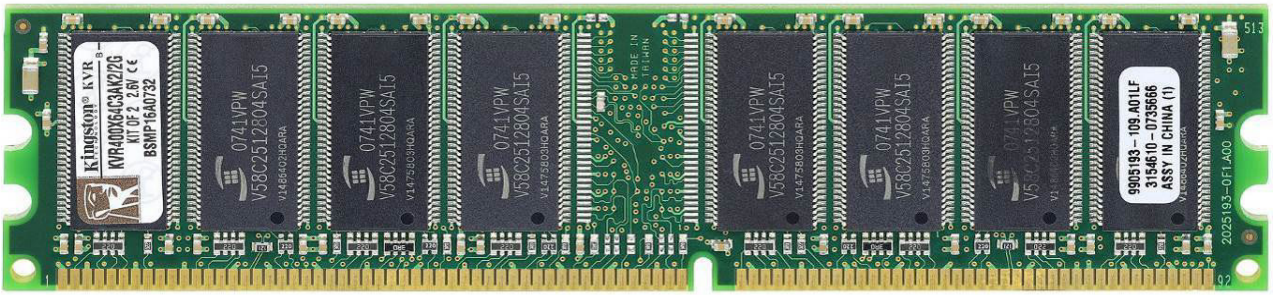


Рис. 1.73. Модуль пам'яті DDR з 184 контактами

На сьогодні відомо про розроблення п’ятого покоління оперативної пам’яті DDR5 SDRAM (double-data-rate five synchronous dynamic random access memory), рис. 1.74.

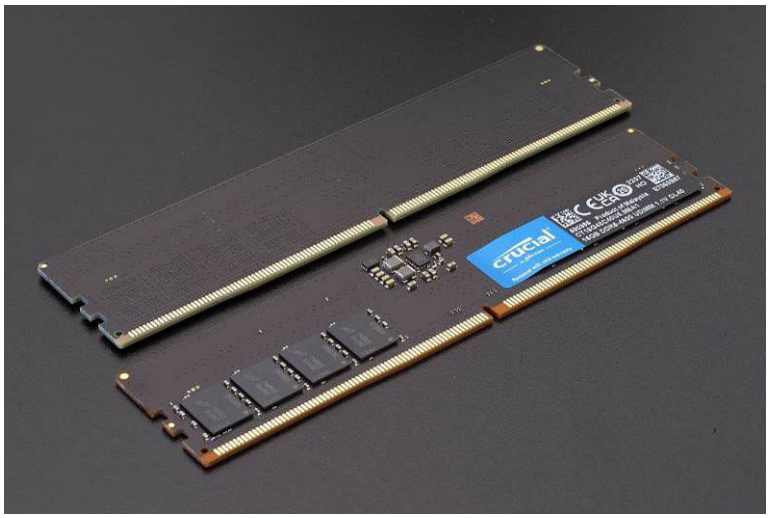


Рис. 1.74. Динамічне ОЗП 16 GiB DDR5-4800 1.1 V UDIMM

1.3.3. Постійні запам’ятовуючі пристрої

Постійні запам’ятовуючі пристрої (ПЗП, Read Only Memory – ROM) – пам’ять енергонезалежного типу, що призначена для довготривалого зберігання даних, які можуть тільки зчитуватися.

ПЗП класифікуються за наступними ознаками:

1. За типом виконання.

1.1. Масив даних виконується суміщено із зчитувальним пристроєм. Наприклад, пам’ять програм FlashROM у складі мікроконтролерів.

1.2 Масив даних фізично відокремлений від зчитувача: компакт-диск; перфокарта; перфострічка.

2. За різновидами мікросхем ПЗП.

2.1. За технологією виготовлення кристалу мікросхеми:

a) ROM (read-only memory) – ПЗП, що програмується під час виготовлення. Користувач не може змінити вміст.

b) PROM (programmable read-only memory) – програмований ПЗП, інформація до якого може бути один раз записана користувачем.

c) EPROM (erasable programmable read-only memory) – перепрограмований ПЗП, що допускає зміну записаної інформації. Наприклад, корпус мікросхеми 2708 був обладнаний віконечком з кварцовим склом, рис. 1.75. Для стирання вмісту необхідно було спрямувати на це віконечко ультрафіолетову лампу.

d) EEPROM (electrically erasable programmable read-only memory) – перепрограмований ПЗП, витирання даних в якому здійснювалося електричним способом. EEPROM використовується в твердотільних накопичувачах. Одним із різновидів EEPROM є флеш-пам'ять (flash memory). В якості прикладу можна навести пам’ять EEPROM типу 25LC256 виробництва Microchip Technology, рис. 1.76. Пам’ять має структуру 32768*8 біт, працює на тактовій частоті 10 МГц, для обміну даними обладнана послідовним інтерфейсом SPI. Кількість циклів перезапису 1000000.

е) ПЗП на магнітних доменах, в якому інформація зберігалася шляхом намагнічення різних ділянок кристалу.

Рис. 1.75. EPROM, вміст якої витирається за допомогою ультрафіолетової лампи

Рис. 1.76. Пам’ять EEPROM 25LC256

2.2. За видом доступу ПЗП поділяються на:

– ПЗП з паралельним доступом;

– ПЗП з послідовним доступом.

2.3. За способом програмування мікросхеми:

– непрограмовані ПЗП, дані до яких записуються під час виготовлення;

– ПЗП, для запису інформації до яких потрібний спеціальний програматор, причому програмування може бути одно- або багатократним;

– ПЗП, що програмуються внутрішньосхемно (in-system programming – ISP), тобто знаходячись на друкованій платі.

1.3.4. Організація та принцип дії арифметико-логічного пристрою

Арифметико-логічний пристрій (АЛП, Arithmetic Logic Unit – ALU) – один з основних блоків мікропроцесора, який виконує арифметичні та логічні операції над даними, які називаються операндами (рис. 1.77). Концепція АЛП була запропонована математиком Джоном фон Нейманом у 1945 році. На сьогодні всі мікроконтролери та мікропроцесори мають у своєму складі один або декілька АЛП.

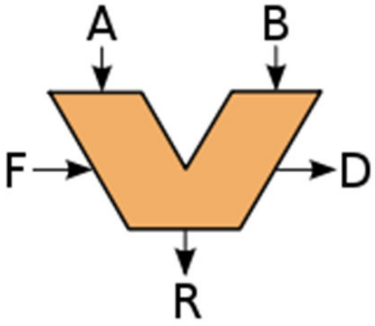


Рис. 1.77. Умовне позначення арифметико-логічного пристрою (АЛП):
A, B – операнди; R – результат; F – код операції; D – статусні біти

АЛП функціонально включає дві частини:

– пристрій управління, який визначає послідовність мікрокоманд для виконання заданої операції;

– операційний пристрій, що забезпечує реалізацію визначеної послідовності мікрокоманд.

АЛП може включати регістри для збереження значень операндів, результату виконання операції та регістр стану. Один з регістрів АЛП – акумулятор – доступний на мові асемблера.

1.3.5. Набір операцій арифметико-логічного пристрою

Набір операцій, що може виконуватися арифметико-логічним пристроєм, може включати від декількох десятків до декількох сотень операцій. Найбільш поширеними операціями є:

– встановлення в регістр константи;

– переміщення операндів між регістрами;

– інкремент (додавання одиниці) та декремент (віднімання одиниці) від вмісту регістра; інвертування вмісту регістра;

– конкатенація даних;

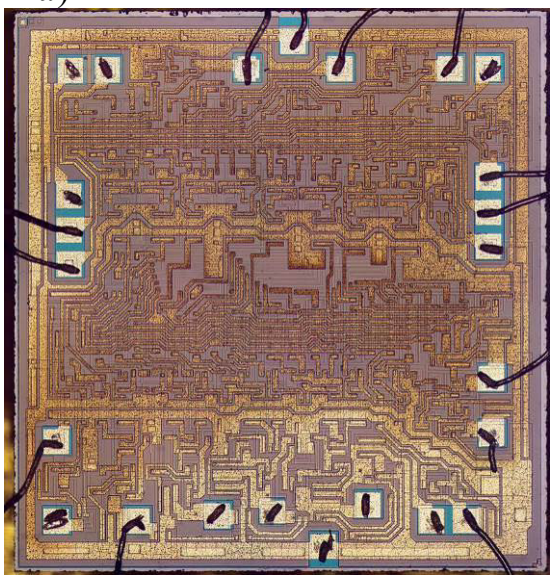
– порозрядні логічні операції;

– арифметичні операції;

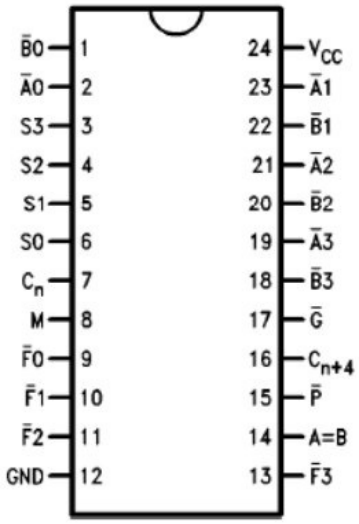
– циклічний зсув вмісту регістра.

В якості прикладу на рис. 1.78 наведено умовне позначення та логічну схему чотирирозрядного АЛП типу 74181. Система команд такого пристрою наведена у табл. 1.13. За допомогою АЛП можна виконувати 16 арифметичних та 16 логічних операцій над 4-розрядними операндами. Значення розрядів операндів A та B надходять на відповідні входи мікросхеми. При цьому активним рівнем сигналу є низький, оскільки мікросхема має інверсні входи. Режим роботи АЛП обирається логічним рівнем сигналу на вході M. Напруга низького рівня на M визначає виконання арифметичних команд, а напруга високого рівня – логічних команд. Тип операції обирається сигналами на входах S0–S3. Для видачі результату використовуються виходи $\overline{F}0-\overline{F}3$.

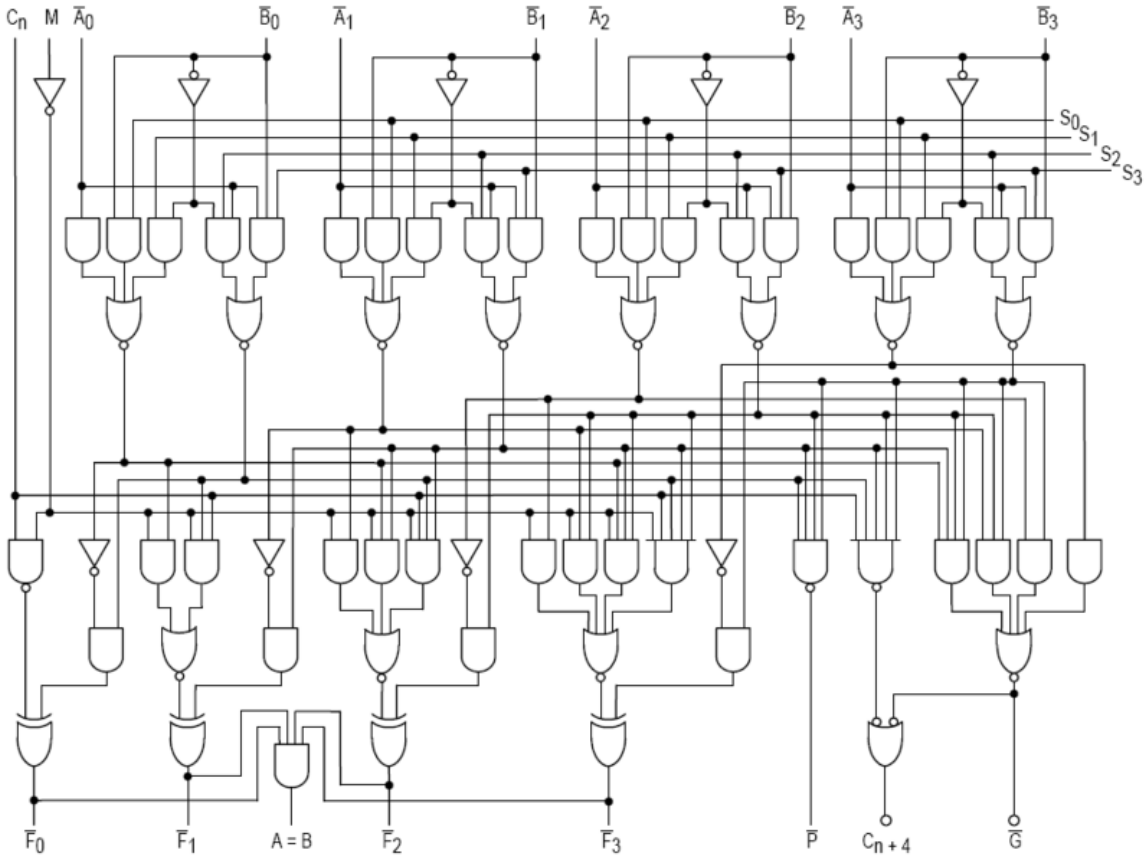
а)



б)



в)



г)

Рис. 1.78. Загальний вигляд (*а*), чип (*б*), умовне графічне позначення (*в*) та логічна схема (*г*) мікросхеми 74181, що являє собою 4-х розрядний арифметико-логічний пристрій

Таблиця 1.13

Система команд чотирирозрядного арифметико-логічного пристрою типу 74181

Входи	Функція	
$S_3S_2S_1S_0$	M = 0 (арифметична)	M = 1 (логічна)
0000	$F = A - 1 + C_0$	$F = \bar{A}$
0001	$F = A \wedge B - 1 + C_0$	$F = \bar{A} \vee \bar{B}$
0010	$F = A \wedge \bar{B} - 1 + C_0$	$F = \bar{A} \vee B$
0011	$F = 1111 + C_0$	$F = 1111$
0100	$F = A + (A \vee \bar{B}) + C_0$	$F = \bar{A} \wedge \bar{B}$
0101	$F = A \wedge B + (A \vee \bar{B}) + C_0$	$F = \bar{B}$
0110	$F = A - B - 1 + C_0$	$F = A \oplus \bar{B}$
0111	$F = A \vee \bar{B} + C_0$	$F = A \vee \bar{B}$
1000	$F = A + (A \vee B) + C_0$	$F = \bar{A} \wedge B$
1001	$F = A + B + C_0$	$F = A \oplus B$
1010	$F = A \wedge \bar{B} + (A \vee B) + C_0$	$F = B$
1011	$F = A \vee B + C_0$	$F = A \vee B$
1100	$F = A + A + C_0$	$F = 0000$
1101	$F = A \wedge B + A + C_0$	$F = A \wedge \bar{B}$
1110	$F = A \wedge \bar{B} + A + C_0$	$F = A \wedge B$
1111	$F = A + C_0$	$F = A$

1.3.6. Принцип дії аналогового компаратора

Компаратором аналогових сигналів називається електронна схема, яка забезпечує порівняння двох аналогових сигналів, рис. 1.79. Результат порівняння видається у вигляді двійкової величини. Компаратор має два входи: прямий вхід (позначається «+») та інверсний вхід (позначається «–»). Компаратор видає лог. 1, якщо сигнал на прямому вході перевищує сигнал на інверсному вході. В протилежному випадку видається лог. «0».

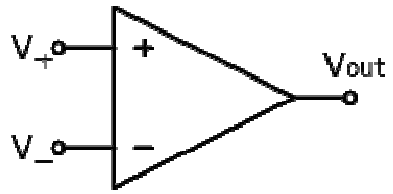


Рис. 1.79. Позначення аналогового компаратора

Найчастіше сигнал, що подається на прямий вхід компаратора, називають вхідним сигналом V_{IN}. На інверсний вхід подається опорний сигнал (рівень порівняння) V_{REF}. Позначивши вихідний сигнал V_{OUT}, алгоритм функціонування компаратора може бути подано залежністю:

$$V_{OUT} = \begin{cases} V_{OUT}^{1} & \text{якщо } V_{IN} - V_{REF} > 0 \\ V_{OUT}^{0} & \text{якщо } V_{IN} - V_{REF} < 0 \end{cases}.$$

Припустимо, що на прямий вхід компаратора подається сигнал V_{IN} трикутної форми, а опорний сигнал V_{REF} є незмінним в часі, рис. 1.80. Тоді вихідний сигнал буде нульового рівня V_{OUT}^{0} протягом тих часових інтервалів, коли $V_{IN} < V_{REF}$. В протилежному випадку вихідний сигнал має одиничних рівень V_{OUT}^{1}.

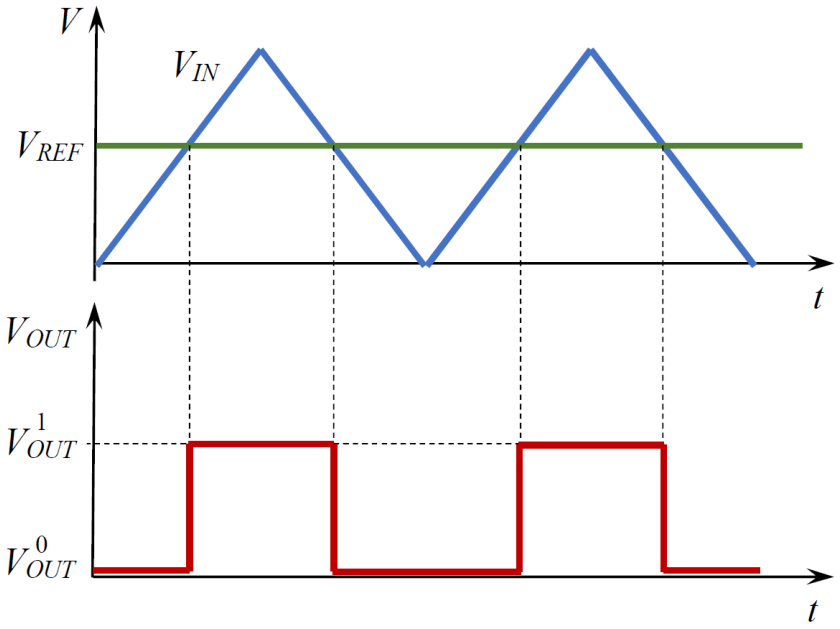


Рис. 1.80. Графіки, що ілюструють принцип функціонування аналогового компаратора

Найчастіше вихідний сигнал компаратора призначено для подачі на входи логічних пристроїв, тому має бути з ними узгоджений за рівнями напруги. Аналоговий компаратор можна вважати елементом переходу від аналогових до цифрових сигналів.

Компаратори використовуються як складові елементи аналого-цифрових перетворювачів. Як окремі мікроелектронні пристрої можуть використовуватися, зокрема, у складі систем імпульсно-фазового управління силовими керованими випрямлячами або тиристорними регулятора напруги для виявлення моменту подачі сигналу керування на тиристор.

1.3.7. Аналого-цифрові перетворювачі. Основні параметри АЦП

Аналого-цифровий перетворювач (АЦП, Analog-to-digital converter – ADC) – мікроелектронний пристрій, що перетворює аналоговий сигнал в цифровий код, рис. 1.81. АЦП надзвичайно широко використовуються для

перетворення аналогових сигналів від давачів різних фізичних величин (струм, напруга, температура, тиск тощо) в цифровий код.

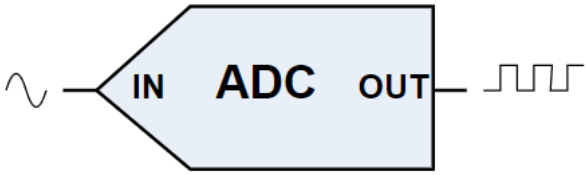


Рис. 1.81. Умовне позначення аналого-цифрового перетворювача

Аналого-цифрове перетворення сигналу передбачає одержання послідовності значень для визначених моментів часу. Такі моменти часу в більшості випадків є рівномірно розподілені на осі часу, відстань Δt між ними називається інтервал (період) дискретизації.

Одержання цифрового сигналу, що відповідає аналоговому, передбачає одночасне виконання двох операцій: дискретизація аналогового сигналу за часом та квантування за рівнем.

Припустимо, що використовується ідеальний перетворювач, здатний миттєво перетворювати аналоговий сигнал в цифровий код. При цьому припустимо, що такий код відповідає величині аналогового сигналу без похибки. Якщо такий перетворювач запускається через інтервали часу Δt, то у визначені моменти часу одержують числа, що відповідають величинам аналогового сигналу. Таке перетворення називається дискретизацію аналогового сигналу за часом, рис. 1.82, *а*.

Розглянемо інший випадок. Припустимо, що перетворювач здатний при вимірюванні аналогового сигналу видавати тільки визначені дискретні значення. Наприклад, округлює результат до цілих чисел 0, 1, 2, …. Якщо вхідна напруга перетворювача становить 2,3 В, то перетворювач видає 3. Якщо напруга становить 15,8 В, то на виході буде число 16. Дискретні значення, що з певною похибкою відповідають величинам аналогового сигналу, називають рівнями квантування сигналу. Припустимо, що перетворювач, який здійснює квантування сигналу за рівнем, діє миттєво. Тоді на його виході буде цифровий сигнал, що показаний на рис. 1.82, *б*.

Аналого-цифровий перетворювач для одержання цифрового сигналу забезпечує квантування сигналу за рівнем та запускається через визначений інтервал дискретизації, рис. 1.83, *в*. Більшість АЦП перетворюють вхідний аналоговий сигнал у двійковий код. Кожен АЦП розрахований на певний інтервал вхідного аналогового сигналу, найчастіше – від 0 до +5 В. Кількість дискретних значень, які перетворювач може видати на виході, називається розрядністю АЦП. Розрядність вимірюється в бітах.

Наприклад, якщо АЦП розрахований на 256 дискретних значення (від 0 до 255), то його розрядність дорівнює 8 біт: $2^8 = 256$. Також розрядність може бути виражена в одиницях вхідного сигналу, найчастіше у вольтах.

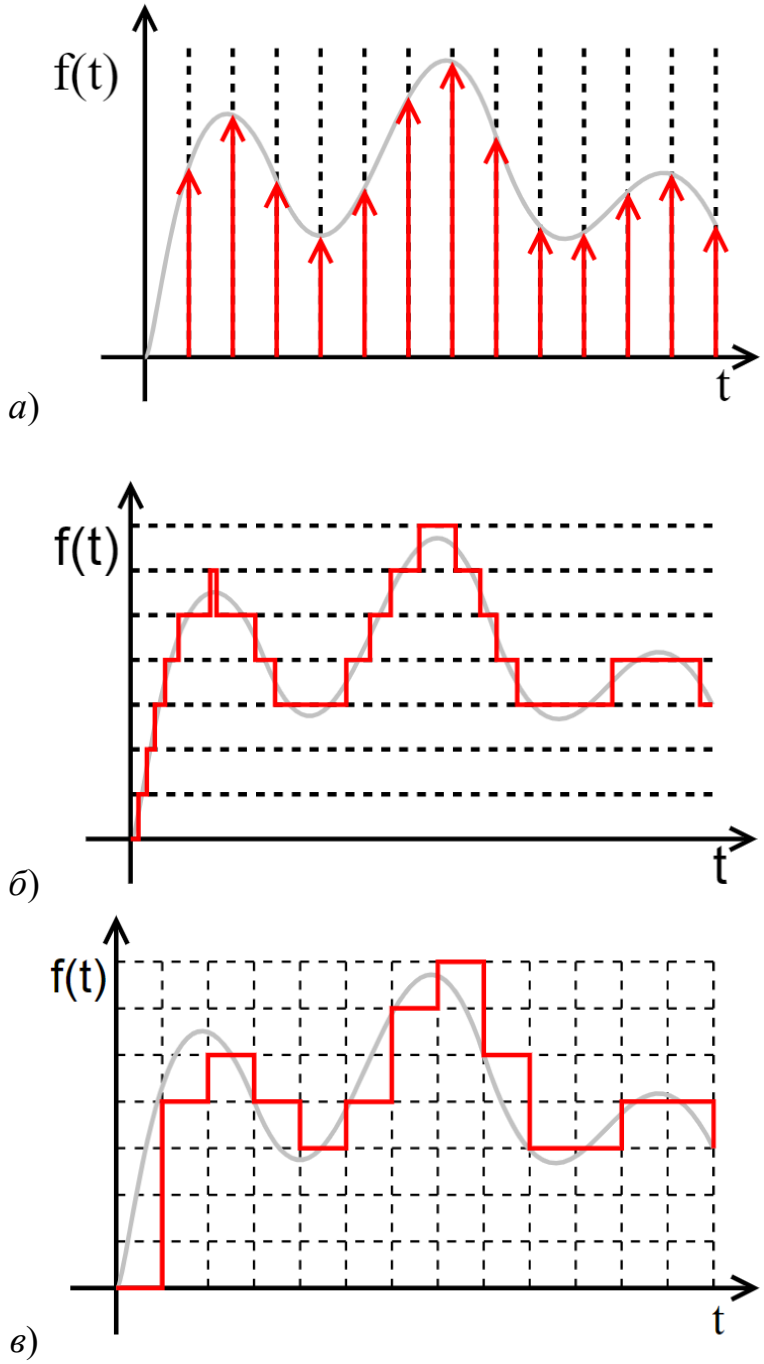


Рис. 1.82. Графіки, що ілюструють:
а – дискретизацію аналогового сигналу за часом;
б – квантування аналогового сигналу за рівнем;
в – одночасну дискретизацію аналогового сигналу за часом та квантування за рівнем для одержання цифрового сигналу

Наприклад, діапазон вхідного сигналу АЦП становить від 0 до 5 В, а розрядність дорівнює 12 біт. Тоді АЦП здійснює квантування сигналу на 2^{12} = 4096 рівнях квантування. Розрядність за напругою становить (5–0)/4096 = 0,00122 В = 1,22 мВ.

Розрядність реального АЦП обмежується співвідношенням сигнал/шум вхідного аналогового сигналу. Для опису досяжної роздільної

здатності перетворювача використовується ефективна кількість розрядів (effective number of bits – ENOB). Такий показник є меншим від роздільної здатності.

Величина, що є оберненою до періоду дискретизації Δt аналогового сигналу в часі, називається частота дискретизації (sampling rate). Згідно з теоремою Котельникова-Шеннона, частота дискретизації має бути вища, ніж подвоєна максимальна частота в спектрі сигналу.

Реальні АЦП не можуть здійснити аналого-цифрове перетворення миттєво, тому для уникнення похибки необхідно утримувати вхідне аналогове значення на постійному рівні під час процесу перетворення (протягом часу перетворення). Для цього на вході АЦП підключається пристрій вибірки-зберігання (ПВЗ). Такий пристрій містить конденсатор, що підключений до входу за допомогою аналогового ключа. Замикання ключа призводить до заряджання конденсатора, тобто відбувається вибірка сигналу. При розмиканні ключа рівень заряду конденсатора відповідає значенню сигналу, що має бути виміряний, тобто реалізується зберігання.

1.3.8. Принцип дії паралельних АЦП

Паралельні АЦП містять набір компараторів, що функціонують одночасно (паралельно), здійснюючи перетворення аналогового сигналу у цифровий код. Кожен з компараторів відповідає за формування значення одного з розрядів вихідного двійкового числа.

В якості прикладу на рис. 1.83 наведена схема паралельного 3-розрядного АЦП. Три розряди двійкового числа дозволяють подати вісім чисел, включаючи 0. Для 0 окремий компаратор не потрібне, тому схема включає 7 компараторів. Вхідний аналоговий сигнал подається на прямі входи всіх компараторів. На інверсні входи подаються еквідистантні опорні напруги (тобто напруги, що сформовані з однаковим кроком) в діапазоні від опорної напруги до 0. Такі напруги одержують за допомогою резистивного дільника. Квант вхідної напруги становить $h = \frac{V_{REF}}{7}$. Таке значення відповідає одиниці молодшого розряду АЦП. Відповідно до значення вхідної напруги спрацьовують компаратори, починаючи від K_1 до K_7. Зокрема, якщо вхідний сигнал знаходиться в діапазоні від $\frac{5}{2}h$ до $\frac{7}{2}h$, то спрацьовують компаратори K_1, K_2, K_3, видаючи лог. «1». Інші компаратори видають лог. «0». Для перетворення таких сигналів у двійковий код застосовується пріоритетний шифратор, табл. 1.14.

Як видно з табл. 1, при збільшенні вхідного сигналу компаратори встановлюються в стан лог. «1» по черзі – від низу до верху. Така черговість не гарантується при швидкому наростанні вхідного сигналу, оскільки через різницю в тривалостях затримок компаратори можуть перемикатися в іншому порядку. Пріоритетне кодування дозволяє уникнути такої помилки, оскільки одиниці в молодших розрядах не беруться до уваги шифратором.

До переваг АЦП паралельного типу відноситься найбільша швидкість роботи серед всіх типів. До недоліків – складність схеми та висока вартість.

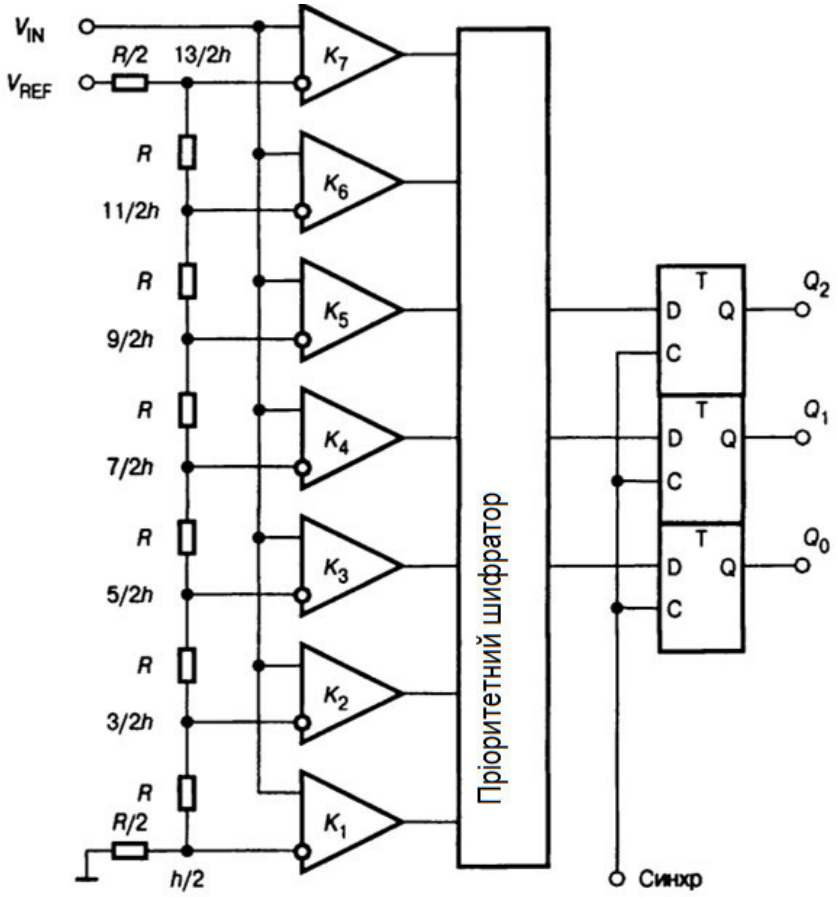


Рис. 1.83. Схема паралельного АЦП

Таблиця 1.14

Стани пріоритетного шифратора

Вхідна напруга V_{IN} / h	Стан компараторів							Виходи		
	K_7	K_6	K_5	K_4	K_3	K_2	K_1	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1	0	0	1
2	0	0	0	0	0	1	1	0	1	0
3	0	0	0	0	1	1	1	0	1	1
4	0	0	0	1	1	1	1	1	0	0
5	0	0	1	1	1	1	1	1	0	1
6	0	1	1	1	1	1	1	1	1	0
7	1	1	1	1	1	1	1	1	1	1

1.3.9. Багатоступінчаті АЦП

Багатоступінчаті АЦП передбачають одержання старших та молодших біт результаті перетворення за допомогою різних перетворювачів. Наприклад, рис. 1.8 ілюструє структурну схеми 8-розрядного АЦП з двома ступенями перетворення.

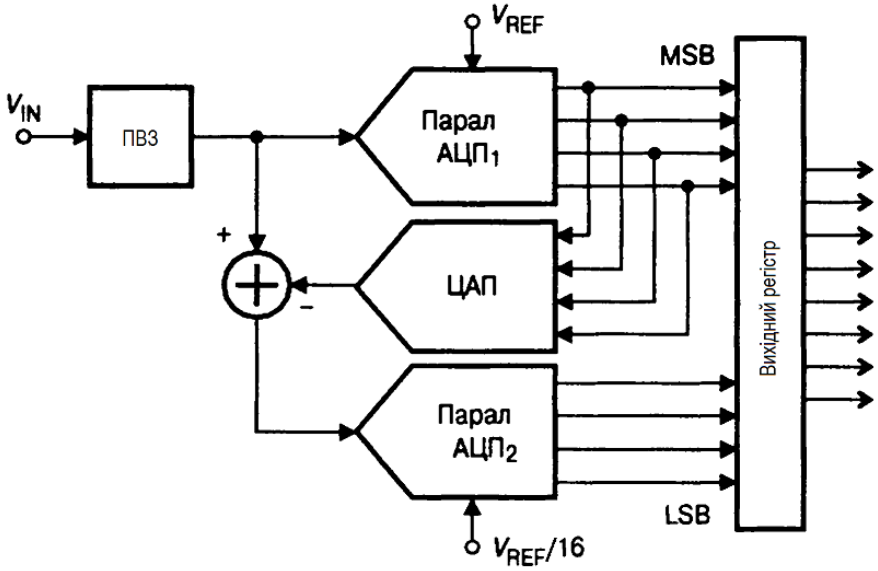


Рис. 1.84. Структурна схема двоступінчатого АЦП: LSB – Least Significant Bit (молодший біт); MSB – Most Significant Bit (старший біт); ПЗВ – пристрій запам'ятовування та вибірки

Для одержання старших 4-х розрядів результату використовується паралельний АЦП1. За допомогою цифро-аналогового перетворювача ЦАП здійснюється зворотне перетворення вказаних розрядів у аналоговий сигнал. Останній віднімається від вхідного сигналу. Для перетворення залишку від віднімання у цифровий код використовується АЦП2, що дозволяє одержати 4 молодші розряди результату. Підвищення точності перетворення досягається за рахунок зменшеної опорної напруги АЦП2 (в даному випадку – у 16 разів). Це дозволяє підвищити точність оцінення молодших бітів результату.

1.3.10. Цифро-аналогові перетворювачі

Цифро-аналоговий перетворювач (ЦАП; англ. DAC – Digital-to-Analog Convertor) – мікроелектронна схема, що призначена для перетворення цифрового сигналу в аналогову форму. На вхід ЦАП подається, як правило, двійковий код. Сигнал на виході може бути поданий як струмом, так і напругою.

В якості прикладу проаналізуємо схему ЦАП з формуванням вагових струмів резистивним колами, рис. 1.85. Принцип дії схеми заснований на формуванні струмів, величини яких пропорційні ваз розряду двійкового числа. Такі струми формуються завдяки резисторам з опорами R_0, $R_0/2$, …, $R_0/2^{N-1}$. Вихідний вузол схеми підсумовує тільки ті вагові струми, яким відповідає логічна одиниця у вхідному коді, що реалізується застосуванням

ключів S_0, S_1,…, S_{N-1}. В якості ключів найчастіше використовуються MOSFET. Схема здійснює генерацію та комутацію (відповідно до значень розрядів вхідного двійкового коду) вагових струмів. Вихідний струм визначається співвідношенням:

$$I_{OUT} = \frac{V_{REF}}{R_0} \sum_{k=0}^{N-1} d_k 2^k = \frac{V_{REF}}{R_0} D,$$

де k – розрядність вхідного коду;

d_i – значення i-го розряду вхідного коду;

D – вхідний код (сумарна вага вхідного коду).

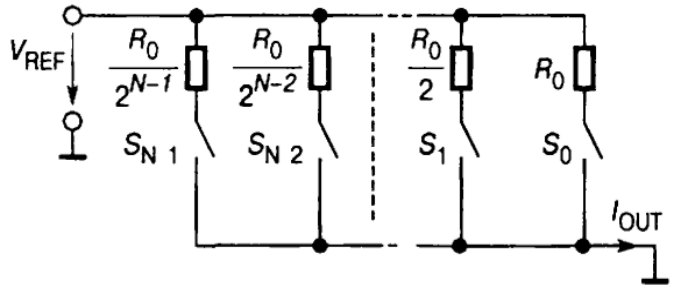


Рис. 1.85. Схема ЦАП з підсумовуванням вагових струмів

До недоліків схеми ЦАП з підсумовуванням вагових струмів слід віднести:

– жорсткі вимоги до точності опорів резисторів, що відповідають за старші розряди;

– зміна величини струму, що споживається від джерела опорної напруги, зі зміною коду, який перетворюється, що може на величину вихідної напруги джерела і, відповідно, внести похибки до результату.

1.3.11. Питання для самоперевірки

1. Дайте визначення носію інформації.
2. На які типи поділяють носії інформації за формою сигналу?
3. Охарактеризуйте матеріальні носії інформації в цифровому вигляді.
4. Дайте визначення постійним запам’ятовуючим пристроям.
5. Охарактеризуйте оперативні запам’ятовуючі пристрої.
6. Опишіть будову та особливості функціонування статичних ОЗП.
7. Надайте характеристику динамічним ОЗП.
8. Які види динамічних ОЗП Вам відомі?
9. Чим відрізняються ROM, PROM, EPROM, EEPROM?
10. Розкрийте організацію та принцип дії арифметико-логічного пристрою.
11. В чому полягає принцип дії аналогового компаратора?
12. Опишіть призначення та основні параметри аналого-цифрових перетворювачів.
13. Розкрийте принцип дискретизації аналогового сигналу за часом.
14. В чому полягає квантування аналогового сигналу за рівнем?

15. Яку властивість характеризує розрядність АЦП?

16. Розкрийте принцип дії паралельного АЦП.

17. В чому полягає особливість функціонування багатоступінчатих АЦП?

18. Розкрийте функціонування схеми ЦАП з підсумовуванням вагових струмів.

1.4. Архітектура мікропроцесорних систем. Мікроконтролери

Структура типової мікропроцесорної системи. Шинна організація. Цикл виконання команди. Концепція віртуальної машини. Історія розвитку та область застосування мікроконтролерів. Виробники мікроконтролерів. Структура типового мікроконтролера. Сімейства мікроконтролерів AVR. Загальні відомості про плати Arduino. Програмування мікроконтролерів.

1.4.1. Структура типової мікропроцесорної системи. Шинна організація

Розглянемо узагальнену структурну схему типової мікропроцесорної системи (рис. 1.86).

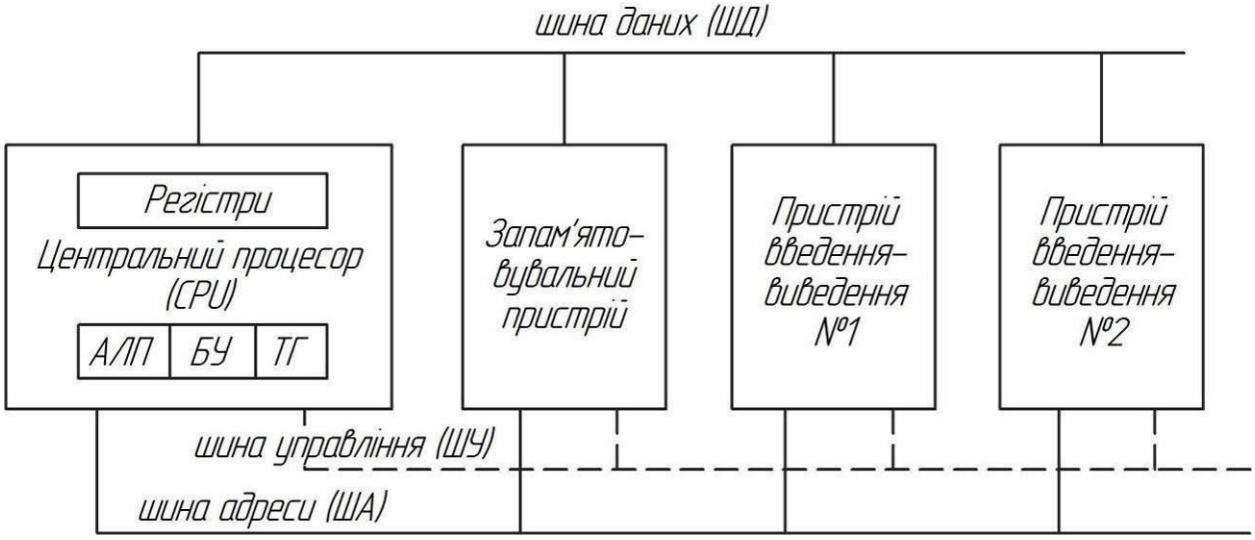


Рис. 1.86. Структурна схема типової мікропроцесорної системи

Усі обчислення і логічні операції виконуються блоком центрального процесора (ЦПУ, Central Processing Unit – CPU). У нім передбачені:

– регістри (внутрішні елементи пам'яті);

– високочастотний тактовий генератор ТГ (генератор тактових імпульсів), який є джерелом синхронізації для внутрішніх команд, що виконуються процесором, і інших компонентів системи;

– блок управління БУ – визначає послідовність мікрокоманд, що виконуються при обробці машинних команд;

– арифметико-логічний пристрій АЛП – безпосередньо виконує арифметичні (складання, віднімання) і логічні (І, АБО, НЕ) операції.

Центральний процесор електрично з'єднується з іншими пристроями комп'ютера за допомогою шин даних, управління та адреси.

При виконанні програми усі команди та дані зберігаються до оперативної пам'яті (ОЗП). Центральний процесор генерує команди звернення до блоку оперативної пам'яті, у відповідь на яких останній або видає на шину даних вміст запитаного елементу оперативної пам'яті, або записує вміст шини даних в заданий за допомогою шини адреси елемент пам'яті.

Шина (bus) є групою паралельних провідників, за допомогою яких дані передаються від одного пристрою комп'ютерної системи до іншого. Зазвичай системна шина комп'ютера складається з трьох різних шин:

– шина даних,

– шина керування,

– шина адреси.

Шина даних (data bus) використовується для обміну команд і даних між CPU і оперативною пам'яттю, а також між пристроями введення-виводу і ОЗП.

По *шині управління* (control bus) передаються спеціальні сигнали, що синхронізують роботу усіх пристроїв, підключених до системної шини.

Шина адреси (address bus) використовується для визначення адреси елемента в RAM, до якого у даний момент відбувається звернення з боку CPU або пристроїв введення-виведення.

Тактовий генератор служить джерелом прямокутних імпульсів постійної частоти, які використовуються для синхронізації внутрішніх команд, що виконуються ЦПУ, і передачі інформації по системній шині. Розрізняють два поняття: машинний такт і машинний цикл.

Машинний такт відповідає одному періоду імпульсів тактового генератора і є основною одиницею виміру часу виконання команд процесором.

Машинний цикл складається з декількох машинних тактів і відповідає часу виконання однієї команди. Наприклад, машинний цикл команди вибірки операнда з пам'яті може складатися з одного-двох машинних тактів.

Тривалість машинного такту пропорційна частоті тактового генератора, яка вимірюється в герцах (Гц). Наприклад, якщо тактовий генератор виробляє за 1 с 1 млрд імпульсів (тобто працює на частоті 1 ГГц), тривалість машинного такту відповідатиме одній мільярдній частині секунди, тобто 1 наносекунді (нс).

У мікропроцесорній системі використовуються принципи організації роботи і особливості архітектури класичних комп’ютерних засобів. Будь-який алгоритм перетворення даних має бути описаний у формі програми – кінцевої послідовності елементарних операцій. Центральний елемент такої системи – мікропроцесор, який виконує операції перетворення даних, здійснює введення і виведення всіх необхідних даних, у тому числі і кодів самої програми, і управляє взаємодією усіх елементів системи.

Набір операцій (виконуваних команд) мікропроцесора повинен мати функціональну повноту, тобто забезпечувати виконання необхідних операцій перетворення даних, управління, обміну даних. Оскільки перетворювані дані і коди програми можуть бути представлені в досить близьких форматах, як

правило, строгого розмежування між ними не роблять. Такий підхід розширює можливості програмної обробки: дані можуть використовуватися як елементи кодів програми, а елементи кодів програми можна перетворювати в процесі роботи.

Для зберігання даних і кодів програми мікропроцесорна система містить запам’ятовувальний пристрій. Стандартна організація мікропроцесорної системи допускає доступність для мікропроцесора усіх даних в пристрої, що запам'ятовує. Оскільки обсяг запам’ятовувального пристрою досить великий, доступ до вибраних даних зазвичай здійснюється через засоби адресації. Кожному елементу даних (наприклад, кожному байту), що зберігаються, відповідає певний код зберігання, який називається адресою в запам’ятовувальному пристрої. Як правило, в мікропроцесорній системі передбачається постійна готовність запам’ятовувального пристрою до обміну даними, для чого мікропроцесор повинен передати певний код адреси даних і відповідні сигнали управління.

Процедури введення-виведення даних також повинні виконуватися стандартними для мікропроцесора алгоритмами. Ці алгоритми, ефективні для управління мікропроцесорної системи, можуть не відповідати особливостям роботи різних периферійних пристроїв. Тому мікропроцесорна система зазвичай містить пристрої введення-виводу, що забезпечують узгодження алгоритмів управління обміном даними.

Типову структуру мікропроцесорної системи, що побудована відповідно до вказаних принципів, можна проілюструвати на прикладі класичного 8-бітного мікропроцесора Intel 8080 (рис. 1.87). До мікропроцесора за допомогою системою шини під’єднується запам’ятовуючий пристрій, в якому зберігається програма, а також пристрої введення-виведення для підключення зовнішніх компонентів для введення та відображення даних. Послідовне виконання кодів програми забезпечується програмним лічильником. Код чергової команди надходить до регістра команд, звідки потрапляє до пристрою управління. Останній забезпечує формування мікрокоманд керування арифметико-логічним пристроєм АЛП, які реалізують задану команду мікропроцесора. Для проміжного збереження значень операндів використовуються регістри загального призначення (A – акумулятор; B – база; C – лічильник; D – дані; H:L, D:E – регістрові пари, що часто використовуються). Результат операції, яку виконує АЛП, найчастіше записується до регістра-акумулятора. Виконану операцію характеризують прапори (розряди) регістра прапорів:

S – знак результату;

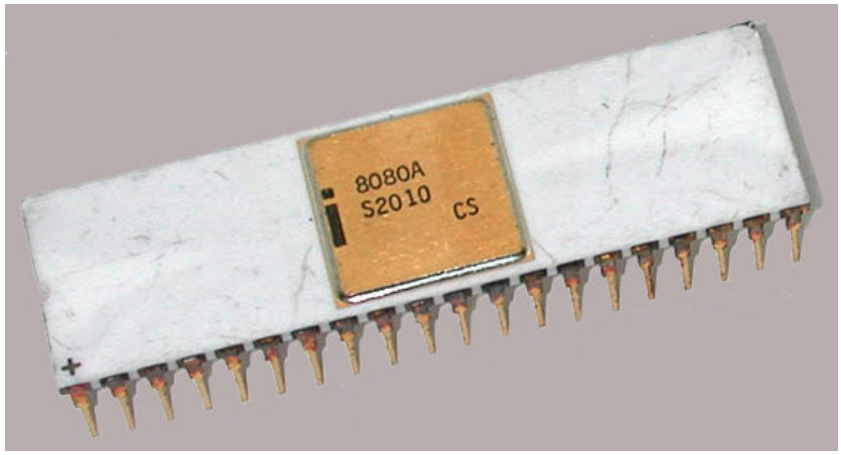
Z – нульовий результат;

AC – допоміжне перенесення;

P – парність;

C – перенесення.

а)



б)

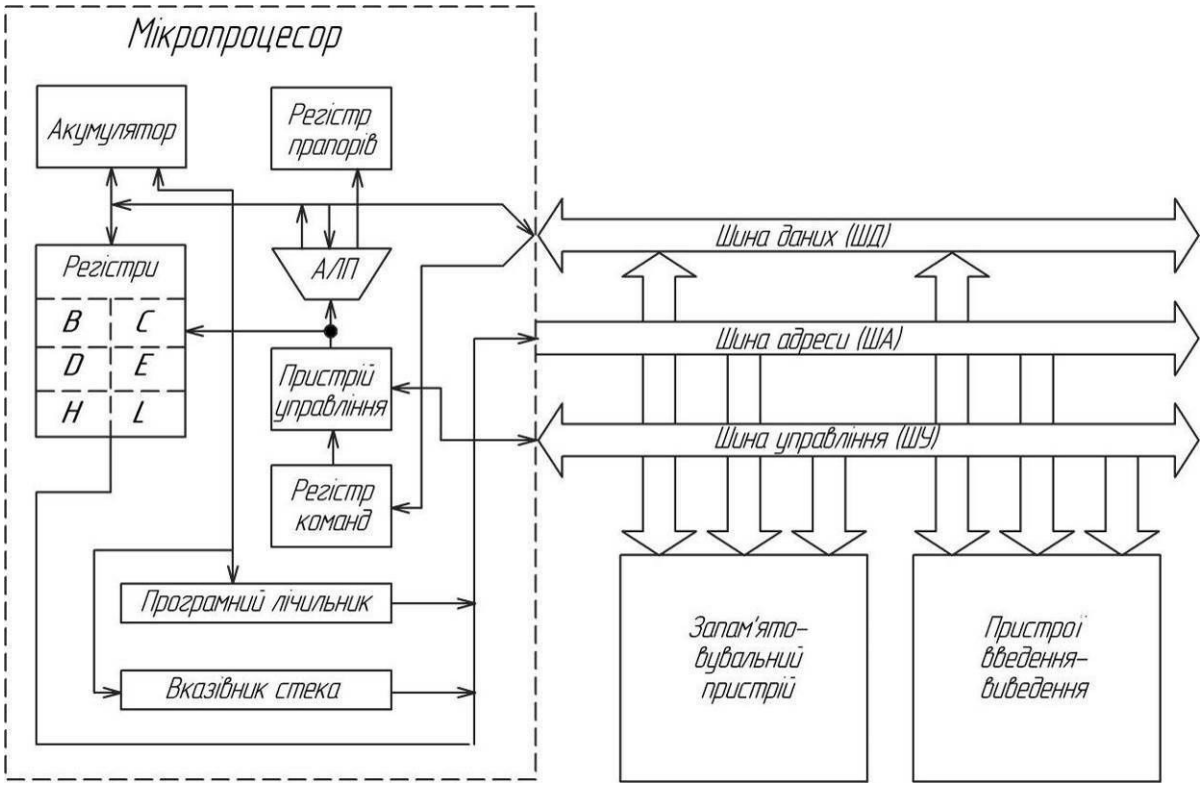
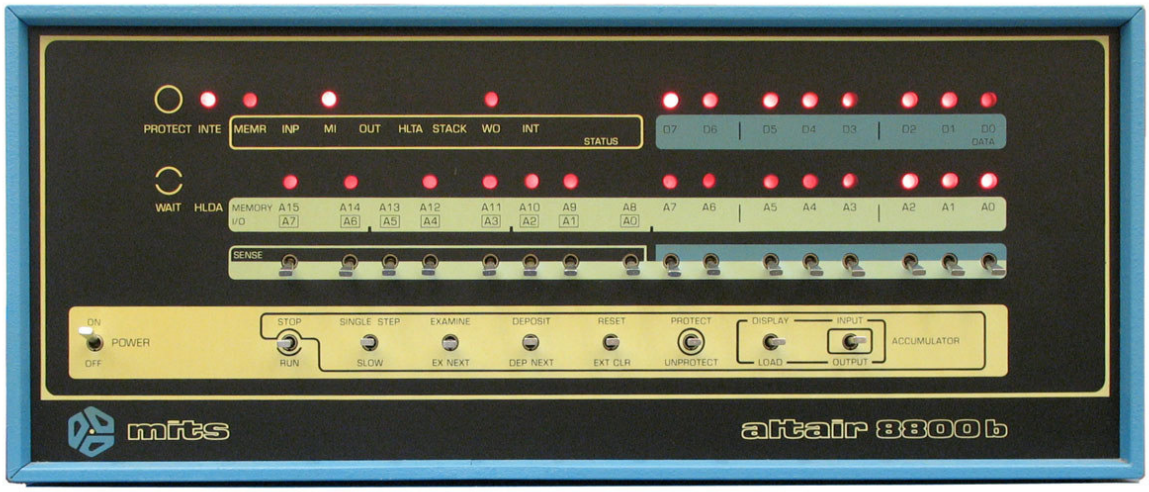


Рис. 1.87. Загальний вигляд (а) та структурна схема (б) мікропроцесора Intel 8080

На базі мікропроцесора Intel 8080 фірмою MITS в 1975 р. був випущений «перший у світі мінікомп’ютерний комплект, який може змагатися з промисловими зразками» (персональний комп'ютер) Altair-8800, який користувався неймовірно великою на ті часи популярністю (MITS не встигала навіть вчасно обробляти замовлення), рис. 1.88.

У 1975 р. Біл Гейтс і Пол Аллен вирішили написати інтерпретатор мови BASIC для комп'ютера Altair 8800 і заснували компанію Microsoft із розроблення програмного забезпечення для комп'ютерів, яка зараз відома у всьому світі. Окрім Altair-8800, мікропроцесор Intel 8080 також застосовувався в пристроях управління вуличним освітленням і світлофорами, а також в іншому устаткуванні.

а)



б)

Рис. 1.88. Altair 8800 – мікрокомп’ютер (*а*) з передньою панеллю (*б*), розроблений компанією MITS (Micro Instrumentation and Telemetry Systems, США) в 1975 році на основі мікропроцесора Intel 8080; з написання програмного забезпечення для цього комп’ютера Білом Гейтсом і Полом Алленом і почалася історія компанії Microsoft

1.4.2. Цикл виконання команди

Цикл виконання команди – послідовність операцій, що здійснюються процесором при виконанні однієї машинної команди. Схема циклу виконання команди показана на рис. 1.89.

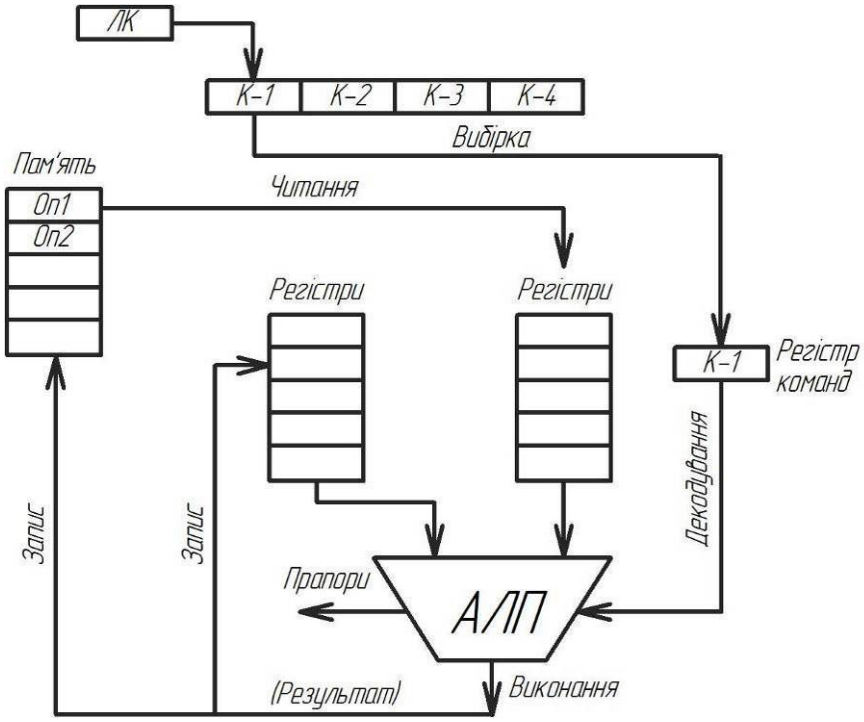


Рис. 1.89. Спрощена схема циклу виконання команди

Лічильник команд (ЛК) – регістр, в якому міститься адреса наступної команди, що виконується

Черга команд – це область надоперативної пам'яті усередині мікропроцесора, в яку поміщається одна або декілька команд безпосередньо перед їх виконанням. При виконанні кожної машинной команди процесор повинен виконати як мінімум три основні операції: вибірка, декодування і виконання. Якщо в команді використовується операнд, що розміщується в пам’яті, процесору треба виконати ще дві додаткові операції: вибірку операнда з пам'яті і запис результату в пам’ять. Іншими словами, при виконанні команди, пов’язаної із зверненням до пам'яті, процесор повинен виконати як мінімум п’ять операцій:

1) *Вибірка команди*. Блок управління зчитує команду з пам’яті, копіює її у внутрішню пам’ять мікропроцесора і збільшує значення лічильника команд на довжину цієї команди.

2) *Декодування команди*. Блок управління визначає тип виконуваної команди, пересилає вказані в ній операнди в АЛП і генерує електричні сигнали управління АЛП, що відповідають типу виконуваної операції.

3) *Вибірка операндів*. Якщо в команді використовується операнд, розташований в пам’яті, блок управління ініціює операцію по його вибірці з пам'яті.

4) *Виконання команди*. АЛП виконує вказану в команді операцію, зберігає

отриманий результат в заданому місці і оновлює стан прапорів, по значенню яких програма може зробити висновок про результат виконання команди.

5) *Запис результату в пам'ять*. Якщо результат виконання команди має бути збережений в пам'яті, блок управління ініціює операцію збереження даних в пам'яті.

Конвеєрна обробка – паралельне виконання процесором різних етапів декількох команд. Конвеєрная обробка була уперше застосована в процесорі Intel486.

Суперскалярна архітектура передбачає наявність 2 (або більше) конвеєрів для виконання команд. Це дозволяє одночасно виконувати 2 (або більше) команди. У процесорі Intel Pentium було застосоване 2 конвеєри. Таким чином, він став першим процесором сімейства IA-32, побудованим по суперскалярній архітектурі. У процесорі Pentium Pro уперше було застосовано 3 конвеєри.

Для підвищення продуктивності у 2000х роках стали розробляти багатоядерні процесори, що включають 2 або більше обчислювальних ядер, які розташовані на одному кристалі. Перші двоядерні мікропроцесори AMD Athlon 64 X2 та Intel Pentium D були випущені у 2005 р.

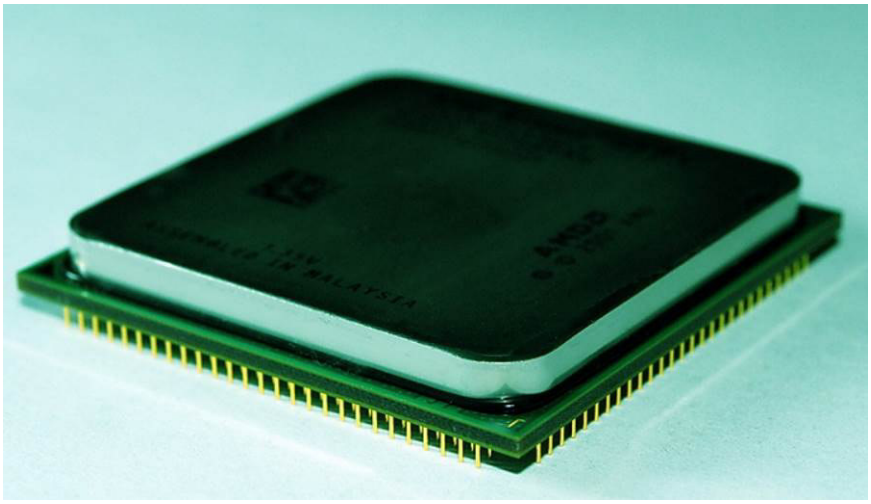
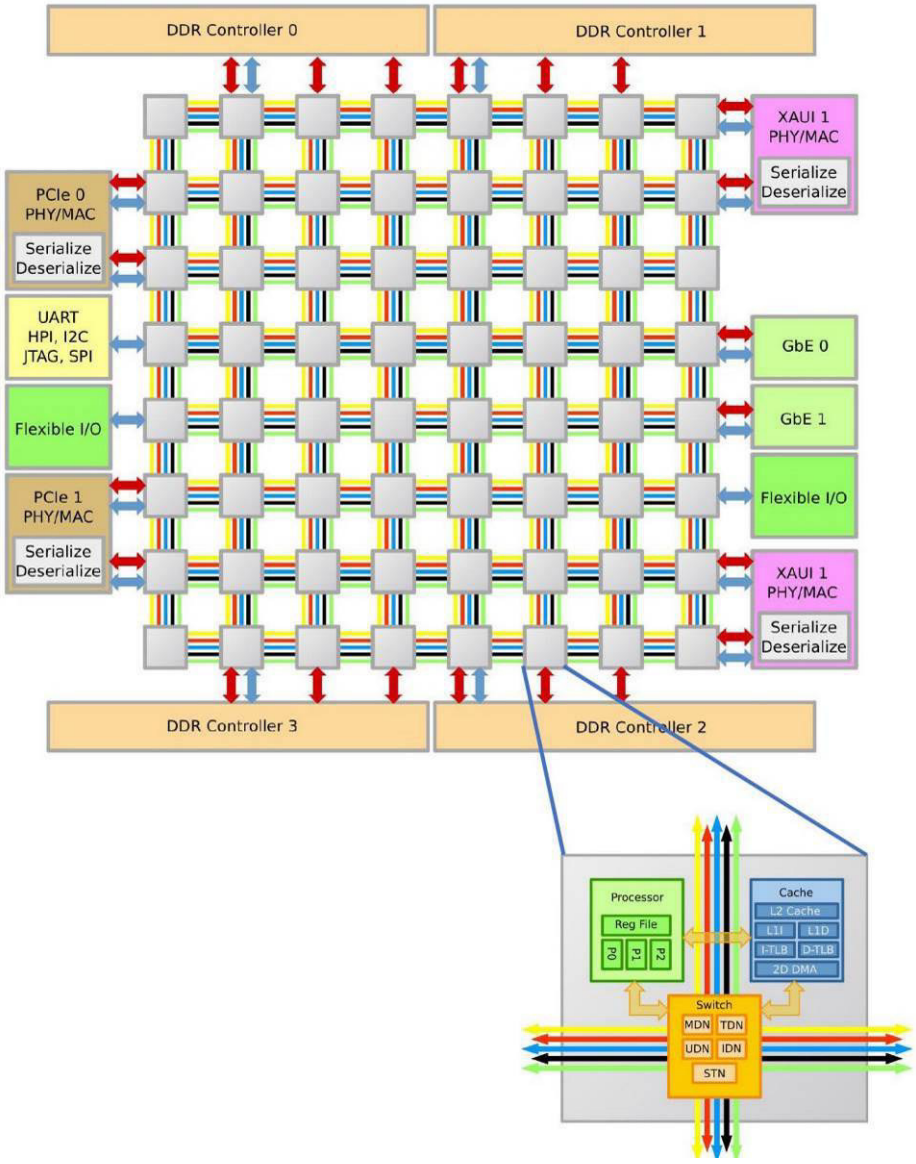


Рис. 1.90. Двоядерний мікропроцесор AMD Athlon 64 X2

Набувають широкого поширення багатоядерні мікропроцесори, що мають матричну організацію. Наприклад, багатоядерний процесор TILE64 виробництва компанії Tilera включає матричну мережу, у вузлах якої включені 64 процесори загального користування з власними регістровими файлами, цілочисленими АЛП та блоками зберігання даних, рис. 1.91. До складу кожного ядра входить маршрутизатор, що використовується для зв'язку з іншими ядрами процесора. Ядро здатне запускати повну операційну систему самостійно або кілька ядер можна використовувати для запуску симетричної багатопроцесорної операційної системи.

а)



б)

Рис. 1.91. Багатоядерний процесор TILE64 (*а*) та структурна схема матричного з’єднання ядер (*б*)

1.4.3. Концепція віртуальної машини

Концепція віртуальної машини була запропонована Ендрю Таненбаумом і використовується для пояснення взаємозв'язку апаратного і програмного забезпечення комп’ютера.

Зазвичай при проектуванні будь-якого комп'ютера в нім передбачають можливість безпосереднього запуску програм, що складаються з так званих машинних кодів (мова L0). Писати програми мовою L0 украй неефективно, оскільки ця мова складається із звичайних цифр. Для полегшення програмування має бути створена нова мова програмування (мова L1). Цю мету можна досягти двома способами:

1) *інтерпретація.* Під час виконання програми на мові L1, кожна з її команд повинна оперативно декодуватися і виконуватися програмою, написаною на мові L0. Таким чином, при запуску програма на мові L1 починає виконуватися відразу, але кожна її команда перед виконанням має бути декодована.

2) *трансляція*. Перед виконанням програма, написана на мові L1, має бути перетворена в програму на мові L0 іншою спеціально створеною для цієї мети програмою, написаною на мові L0. Після цього отримана на мові L0 програма може бути безпосередньо виконана центральним процесором комп’ютера.

Таненбаум запропонував створити модель гіпотетичного комп'ютера, або віртуальну машину, для кожної з мов L. Нехай віртуальна машина VM1 виконує команди, написані на мові L1, а віртуальна машина VM0 виконує команди мови L0 (рис. 1.92).

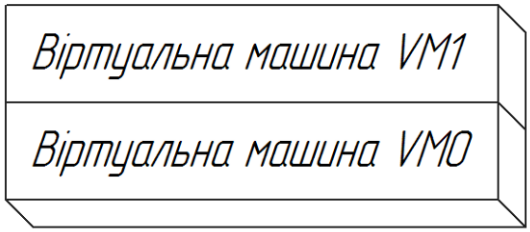


Рис. 1.92. Принцип віртуальних машин Таненбаума

Кожна віртуальна машина може бути реалізована або програмно, або апаратно. Проте незалежно від цього програміст може писати програми для віртуальної машини VM1 так само, як якби він це робив для реального комп'ютера.

Розглянемо *види віртуальних машин*, що використовуються в комп'ютерах (рис. 1.93).

Рівень 0. Віртуальна машина рівня 0 реалізована за допомогою цифрових електронних схем.

Рівень 1. Віртуальна машина рівня 1 виконана у вигляді інтерпретатора, логіка роботи якого «зашита» в спеціалізованому процесорі, керованому мікропрограмою.

Рівень 2. Машина рівня 2 є системою команд процесора. На цьому рівні користувачі можуть писати найпростіші програми, що складаються із звичайних двійкових чисел.

Рівень 3. Операційна система. Машина може вести роботу з користувачем в діалоговому режимі і виконувати його команди, такі як завантаження в пам'ять програм і їх виконання, відображення каталогу і тому подібне. Програма, або віртуальна машина, за допомогою якої реалізовані ці можливості, називається операційною системою комп'ютера.

Рівень 4. Мова асемблера. У мові асемблера використовуються короткі мнемоніки команд, такі як ADD, SUB і MOV. Програми на мові асемблера перед запуском зазвичай транслюються (чи асемблюються) в машинний код повністю.

Рівень 5. Мови високого рівня: C/C++, Java тощо. У програмах, написаних на цих мовах, зазвичай використовуються складні оператори, які транслюються відразу в декілька команд мови асемблера.

Рис. 1.93. Модель віртуальних машин з п‘ятьма рівнями

У архітектурі процесорів фірми Intel сімейства IA-32 підтримується концепція безлічі віртуальних машин. У ній передбачений спеціальний віртуальний режим, в якому повністю емулюється робота процесорів Intel 8086/8088, персональних комп'ютерів IBM PC, що використалися в перших моделях. Більше того, при роботі процесора в цьому режимі можна запустити відразу декілька екземплярів віртуальних машин 8086. Таким чином, кожна з програм, написана для процесора 8086 і запущена на окремій віртуальній машині, може виконуватися незалежно від інших програм, запущених на інших віртуальних машинах. При цьому для програми створюється «ілюзія», що вона виконується на реальному процесорі 8086 і має повний контроль над ним.

1.4.4. Історія розвитку та область застосування мікроконтролерів. Виробники мікроконтролерів

Мікроконтролер (МК) – програмований обчислювальний пристрій, що має набір периферійних пристроїв і використовується для вирішення завдань

управління в технічних системах. По суті, це однокристальний комп'ютер, здатний виконувати завдання управління і регулювання в автоматизованих системах.

З 70-х років при розроблені систем управління стали використовувати обчислювальні системи на базі мікропроцесорів. Проте самі по собі мікропроцесори не мали можливості безпосередньо вирішувати завдання управління і для досягнення цілей автоматизації вони мали забезпечуватися набором додаткових пристроїв: пам'яттю програм і даних, таймерами, лічильниками, аналого-цифровими (АЦП) і цифро-аналоговими перетворювачами (ЦАП), контролерами введення-виводу тощо.

Така структура вживалася в пристроях досить часто, у зв'язку з чим виникла ідея інтеграції елементів систем управління, що найчастіше використовуються, на одному кристалі. Перший мікроконтролер був випущений фірмою Intel в 1976 році під позначенням i8048, рис. 1.94.

Рис. 1.94. Перший мікроконтролер i8048

Наступний 8-бітовий мікроконтролер Intel 8051 для вбудованих систем, випущений в 1980 році, поклав початок цілому сімейству мікроконтролерів. Більшість фірм і сьогодні випускають пристрої, засновані на Intel 8051, рис. 1.95.

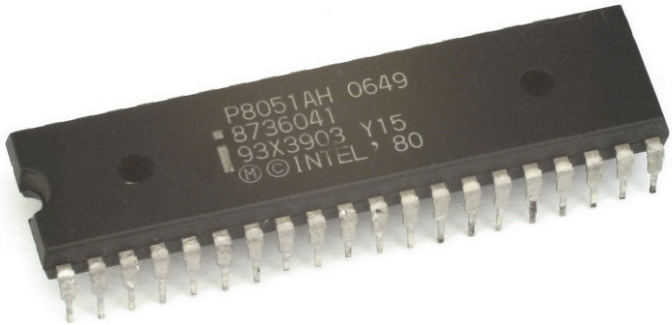


Рис. 1.95. 8-бітовий мікроконтролер Intel 8051 для вбудованих систем

Серед великих виробників мікроконтролерів слід згадати Cypress, Texas Instruments, Dallas Semiconductor, Philips, Infineon (Siemens), Motorola, Zilog, STMicroelectronics, Futjitsu, Mitsubishi Electronics, Temic, National Semoconductor, Oki Semiconductor тощо.

В промисловості та побуті мікроконтролери застосовуються надзвичайно широко, обсяг випуску мікроконтролерів продовжує збільшуватися і складає нині близько двох мільярдів штук на рік. Мікроконтролери використовуються в:

– платах управління різними пристроями і їх окремими блоками;

– в пристроях промислової автоматики (від програмованого реле і вбудованих систем до програмованих логічних контролерів);

– в релейному захисті електроенергетичних об’єктів;

– перетворювачах частоти для керованого електропривода;

– засобах телемеханічного керування та зв’язку високовольтним електроенергетичним обладнанням;

– в цифрових лічильниках електроенергії;

–в системах автоматичного управління виробничими машинами і механізмами.

1.4.5. Структура типового мікроконтролера

Мікроконтролер призначений для управління технічними пристроями, виконання оброблення вимірювальної інформації. Наприклад, мікроконтролер, що входить до складу цифрового релейного захисту, обробляє вимірювальну інформацію від давачів напруги та струму, блок-контактів силового вимикача та, за необхідності, видає команду на відключення останнього. Мікроконтролер оснащений енергонезалежною пам’яттю програм, що дозволяє зберігати коди програми. Наявна оперативна пам’ять, що використовується для виконання обчислень. До складу мікроконтролера можуть входити додаткові пристрої, рис. 1.96.

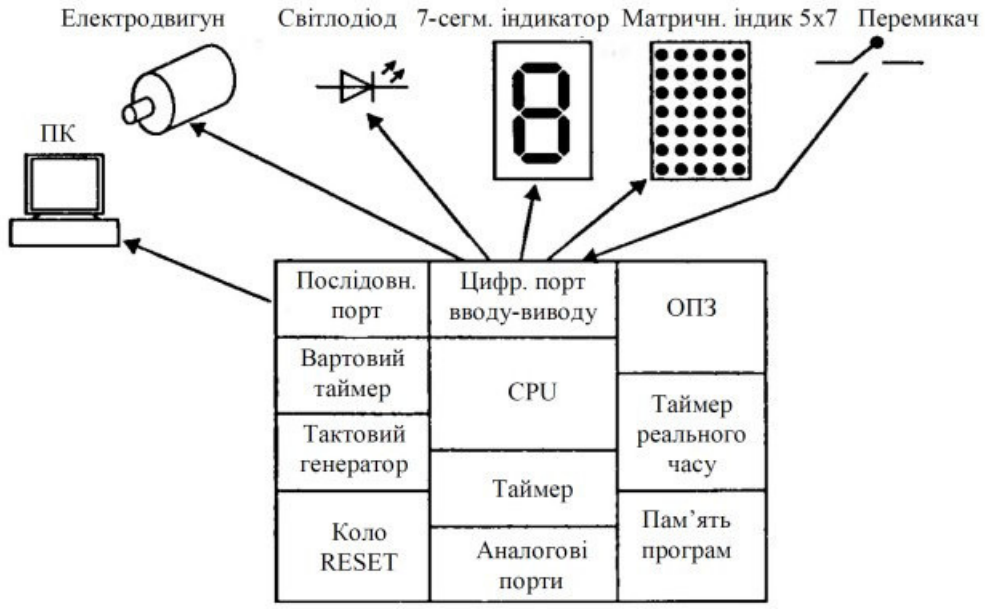


Рис. 1.96. Структурна схема типового мікроконтролера

До складу типового мікроконтролера входять наступні складові.

1. Центральний процесорний пристрій (ЦПП, Central processing unit – CPU) – мікропроцесор, що забезпечує вибірку з пам’яті програм команд, їх декодування та виконання, структура аналогічна зображеній на рис. 1.87.

2. Пам'ять програм енергонезалежного типу, призначена для зберігання кодів команд, що складають програму мікроконтролера.

3. Оперативна пам'ять даних енергозалежного типу, що використовується для тимчасового зберігання даних під час виконання програми. Найчастіше у оперативній пам’яті розміщується стек.

4. Тактовий генератор, що формує прямокутні імпульси для синхронізації функціонування вузлів мікроконтролера.

5. Коло скидання (reset), що забезпечує встановлення мікроконтролера у вихідний стійкий стан.

6. Послідовні інтерфейси для обміну даними з іншими пристроями.

7. Цифрові порти введення/виведення, за допомогою яких здійснюється управління виконавчими механізмами та опитуються дискретні давачі.

8. Таймери-лічильники, забезпечують відлік часових інтервалів, підрахунок числа подій.

9. Вартовий таймер, що запобігає зависанню програми.

1.4.6. Сімейства мікроконтролерів AVR

Американська корпорація Atmel Corporation розпочала випуск перших 8-розрядних мікроконтролерів сімейства AVR у 1996 р. Випуск 32-розрядних мікроконтролерів AVR32 розпочався 2006 р. У 2016 р. компанія Atmel була поглинена своїм конкурентом – Microchip Technology, яка продовжує випуск частини продукції Atmel та розробила нові мікроконтролери AVR (https://www.microchip.com/).

AVR мають модифіковану Гарвардську архітектура. Це один з різновидів Гарвардської архітектури, що забезпечує доступ до пам’яті команд як до даних. Це дозволяє реалізовувати функцію самопрограмування, коли програма мікроконтролера може змінювати вміст пам’яті команд.

Існують наступні сімейства мікроконтролерів AVR.

1) Сімейство tinyAVR – включає 8-розрядні мікроконтролери, виготовляються за КМОН-технологією, рис. 1.97, що мають RISC-архітектуру. Призначені в першу чергу для низьковартісних схем та є найдешевшими з усіх мікроконтролерів AVR.

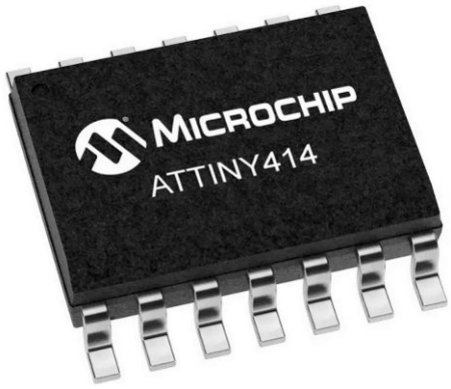


Рис. 1.97. Мікроконтролер ATTiny414, 14 МГц, 4 кБ Flash, SOIC-14

Особливості мікроконтролерів сімейства Tiny:

– можливість обчислень з швидкістю до 20 МГц;

– виготовляються в корпусах з 6–32 виводами;

– оперативна пам'ять (статичний ОЗП, SRAM) об'ємом 32–3072 байт;

– FLASH-пам’ять програм об'ємом 0,5–32 Кбайт;

– EEPROM пам'ять даних об'ємом до 64–512 байт.

2) Сімейство megaAVR – включає 8-розрядні мікроконтролери для складних схем, що вимагають великого об'єму пам'яті програм і даних, рис. 1.98.

Рис. 1.98. Мікроконтролер ATmega2560

Особливості мікроконтролерів сімейства megaAVR:

– FLASH-пам’ять програм об'ємом 4–256 Кбайт;

– статичний ОЗП об’ємом 256–16384 байт;

– EEPROM об'ємом 256–4096 байт;

– частота до 20 МГц;

– можливість програмування безпосередньо в системі через послідовні інтерфейси SPI і JTAG;

– можливість самопрограмування тощо.

3) Сімейство AVR Dx містить кілька серій мікроконтролерів, орієнтованих на HCI (Human–computer interaction – людино-комп'ютерна взаємодія), оброблення аналогового сигналу та функціональну безпеку. Сімейство було випущене 2020 року.

Особливості мікроконтролерів сімейства AVR Dx:

– FLASH-пам’ять програм об'ємом 16–128 Кбайт;

– статичний ОЗП об'ємом 4–16 Кбайт;

– EEPROM об'ємом 512 байт;

– частота 20–24 МГц.

Позначення мікроконтролерів має формат AVRffDxpp, де ff – розмір флеш-пам’яті програм, x – сімейство, pp – кількість контактів. Приклад: AVR128DA64 – 64-контактна серія DA зі пам’яттю програм 128 Кбайт.

До складу сімейства AVR Dx входять наступні серії мікроконтролерів:

Серія AVR DA – мікроконтролери мають високу щільність пам’яті, що дозволяє ефективно організовувати як дротовий, так і бездротовий зв’язок. Мають вбудовані давачі ємнісного сенсорного вимірювання (HCI).

Серія AVR DB – успадковує багато функцій сімейства DA, додаючи власні: 2 або 3 вбудованих операційних підсилювача, MultiVoltage IO (MVIO) на PORTC тощо.

Серія AVR DD – невеликі мікроконтролери, призначені для управління в режимі реального часу та роботи з різними напругами для промислового керування додатками, побутовою технікою, автомобілями та Інтернетом речей (IoT).

Серія AVR EA – мікроконтролери оснащені високошвидкісними інтегрованими аналоговими апаратними незалежними периферійними пристроями, характеризуються продуктивністю з низьким енергоспоживанням для ефективного керування в режимі реального часу, сенсорних вузлів і вторинних програм моніторингу безпеки.

4) Сімейство XMega. 8/16-битные мікроконтролери. Містять інноваційну систему обробки подій «Event System», яка забезпечує незалежну від центрального процесора швидкодіючу передачу даних між внутрішніми периферійними пристроями.

5) Сімейство AVR32. 32 бітові мікроконтролери архітектура RISC. Можуть використовуватися в мобільних високопродуктивних пристроях.

Особливості:

– більшість інструкцій виконуються за один такт;

– 186 інструкцій і 7-ступінчастий конвеєр;

– інструкції цифрового сигнального процесора;

– тактова частота до 200 МГц, продуктивність до 295 мільйон операцій в секунду;

– LCD контролер;

– аудіоконтролер;

– інтерфейси Ethernet, USB тощо;

– 32 Кбайт вбудованої статичної оперативної пам'яті.

1.4.7. Загальні відомості про плати Arduino

Arduino – апаратна обчислювальна платформа для освоєння основ програмування мікроконтролерів. Основними компонентами платформи є мікроконтролерна плата та середовище розроблення програмного забезпечення на мові програмування C for Arduino, що базується на мові C/C++ (https://www.arduino.cc/). В більшості плат встановлені мікроконтролери AVR. Платформа названа на честь короля Ардуїна I (італ. Arduino), що правив Італією у XI ст.

Плати Arduino поділяються на декілька сімейств.

Сімейство Classic – включає широко поширені плати (Arduino UNO, Leonardo, Micro тощо), рис. 1.99. Ці плати вважаються основою проекту Arduino і користуються успіхом протягом багатьох років (і далі в майбутньому).

Сімейство Nano – це набір мініатюрних плат із безліччю функцій, рис. 1.100. До складу сімейства входять плати від недорогої базової Nano Every до багатофункціональної Nano 33 BLE Sense / Nano RP2040 Connect, що оснащена

радіомодулями Bluetooth/Wi-Fi. Ці плати також мають набір вбудованих давачів температури, вологості, тиску, мікрофон тощо. Їх також можна програмувати за допомогою MicroPython, плати підтримують машинне навчання.

Рис. 1.99. Мікроконтролерні плати Arduino сімейства Classic

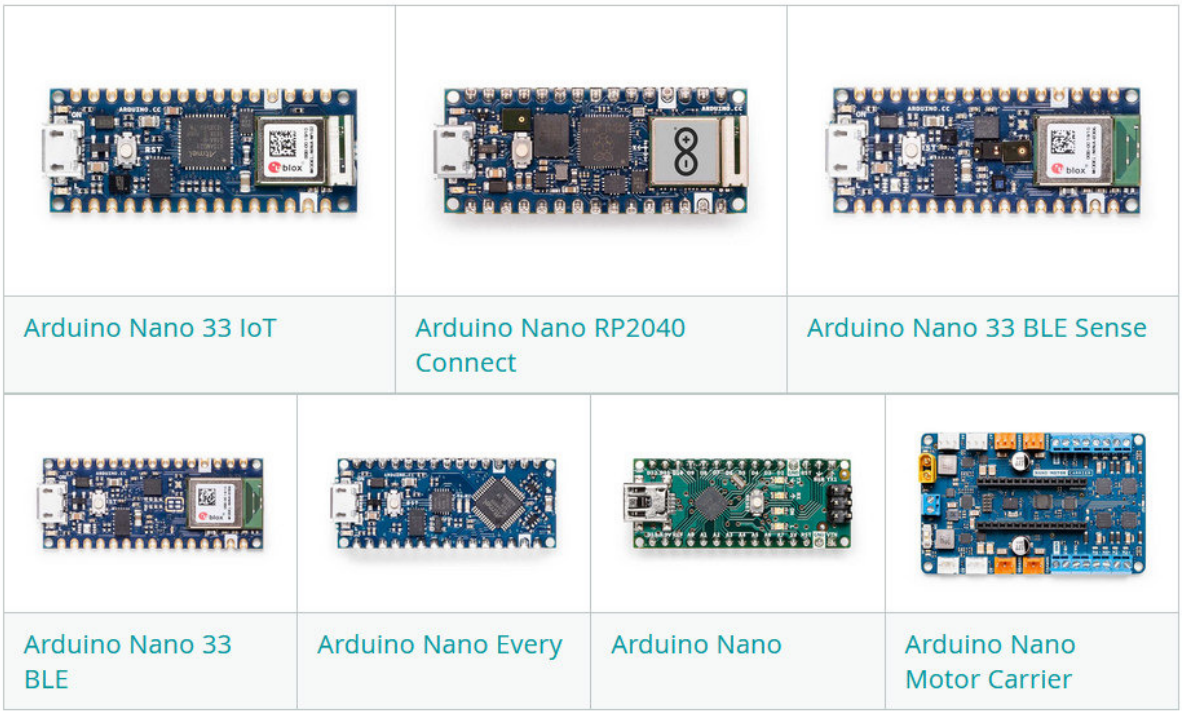


Рис. 1.100. Мікроконтролерні плати Arduino сімейства Nano

Сімейство MKR – це серія плат, екранів і носіїв, які можна комбінувати для створення проектів без будь-яких додаткових схем, рис. 1.101. Кожна плата оснащена радіомодулем (крім MKR Zero), який забезпечує зв'язок Wi-Fi,

Bluetooth, LoRa, Sigfox, NB-IoT. Усі плати сімейства засновані на 32-розрядному процесорі Cortex-M0 SAMD21 з низьким енергоспоживанням і оснащені криптографічним чіпом для безпечного зв’язку. Екрани та носії сімейства MKR розроблені для розширення функцій плати: таких як датчики навколишнього середовища, GPS, Ethernet, керування двигуном і матриця RGB.

Рис. 1.101. Мікроконтролерні плати Arduino сімейства MKR

В якості прикладу розглянемо мікроконтролерну плату Arduino Uno, рис. 1.102. Така плата включає 8-розрядний мікроконтролер ATmega328p. Плата містить 14 цифрових входів/виходів (з них 6 можуть використовуватися як ШІМ-виходи), 6 аналогових входів, кварцовий резонатор на 16 МГц, роз'єм USB, роз'єм живлення, роз'єм для внутрішньосхемного програмування (ICSP) і кнопка скидання. Для початку роботи з пристроєм досить просто подати живлення від AC/DC-адаптера або батареї, або підключити його до комп'ютера за допомогою USB-кабелю. У плат Arduino Uno в якості перетворювача інтерфейсів USB – UART використовує мікроконтролер ATmega16U2.

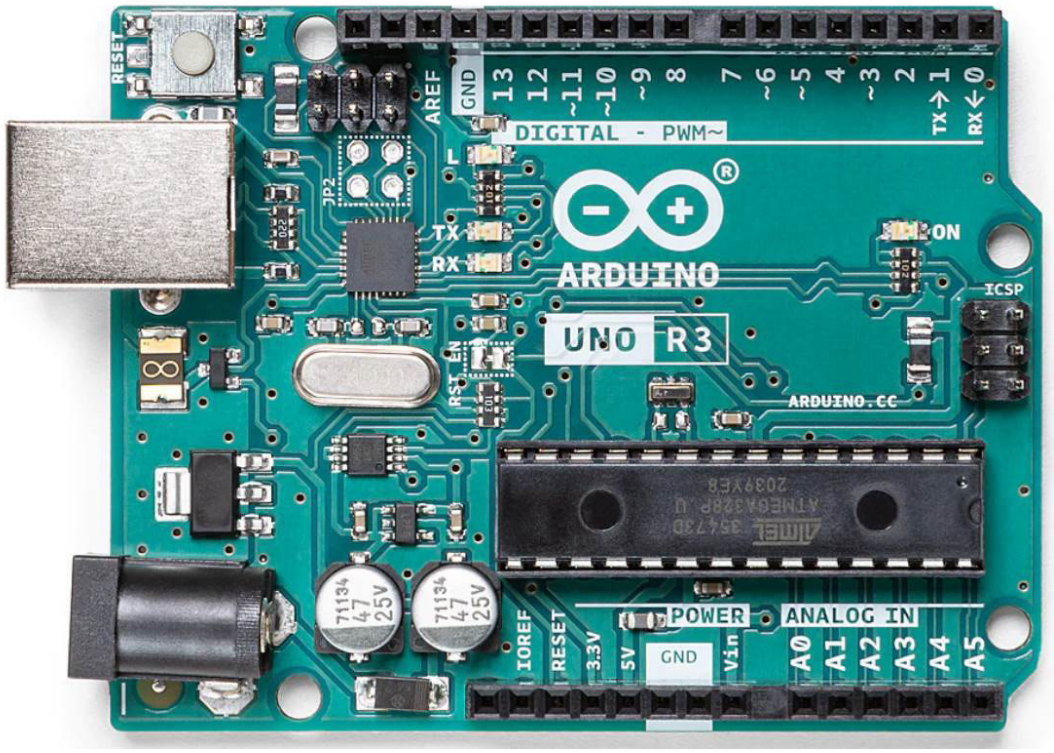


Рис. 1.102. Плата Arduino Uno на основі мікроконтролера ATmega328p

Характеристики плати Arduino Uno наступні:

Мікроконтролер……ATmega328
Робоча напруга……5В
Напруга живлення (рекомендована)……7–12В
Напруга живлення (гранична)……6–20В
Кількість цифрових входів-виходів……...14
Кількість аналогових входів……6
Максимальний струм одного вивода……..40 мА
Flash-пам’ять……32 Кбайт
SRAM……..2 Кбайт
EEPROM……1 Кбайт
Тактова частота……16 МГц

1.4.8. Програмування мікроконтролерів

Програмування мікроконтролера передбачає запис програми і даних (у шістнадцяткових кодах) в пам'ять контролера, а також його конфігурування.

В процесі програмування можуть виконуватися наступні операції:

1) стирання кристала (повинна виконуватися перед кожним перепрограмуванням мікроконтролера);

2) читання/запис FLASH–пам’яті програм;

3) читання/запис EEPROM–пам’яті даних;

4) читання/запис конфігураційних комірок (Fuse Bits – визначають режим роботи тактового генератора, тривалість затримки скидання і інші конфігураційні параметри);

5) читання/запис комірок захисту (захист пам'яті програм і пам'яті даних від запису і/або читання);

6) читання комірок ідентифікатора (три 8-бітові осередки, містять код виробника, код об'єму FLASH-пам’яті, код пристрою);

7) читання калібрувального байта (містить калібрувальну константу, призначену для підлаштування на номінальну частоту внутрішнього RC-генератора).

Мікроконтролери сімейств Tiny і Mega підтримують наступні режими програмування:

1. *Паралельне програмування при високій напрузі*. Під «високою» напругою тут розуміється напруга (12 В), що подається на виведення RESET мікроконтролера для переведення останнього в режим програмування, рис. 1.103.

Особливості:

- від програматора до мікроконтролера передаються одночасно усі розряди коду команди або байта даних, тобто задіяне велике число виводів мікроконтролера;
- вимагається використання додаткового джерела підвищеної напруги (12 В);
- здійснюється спеціалізованими програматорами, куди необхідно вставляти контролер для виконання «прошивки».

Основне застосування режиму паралельного програмування – «прошивка» мікроконтролерів перед установкою їх на плату в умовах масового виробництва.

Паралельне програмування, яке вимагає виймання AVR-мікроконтролера з системи і установки його в програматор, дуже незручне на етапі відлагодження програми.

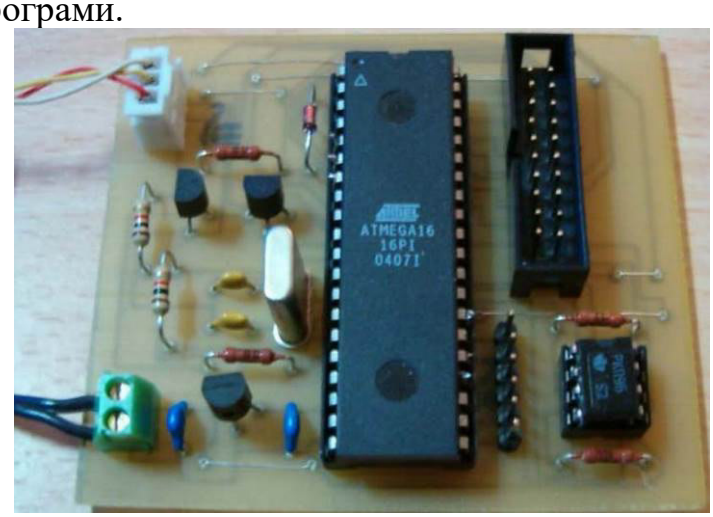


Рис. 1.103. Паралельний програматор із вставленим мікроконтролером ATMEGA16

2. *Послідовне програмування при низькій напрузі* (по інтерфейсу SPI). Не вимагає додаткового джерела живлення і може виконуватися безпосередньо в мікропроцесорній системі (In System Programming, ISP) через послідовний SPI-інтерфейс (рис. 1.104). Можливість внутрісистемного програмування є одним з

найважливіших переваг AVR, оскільки дозволяє значно спростити і здешевити процес розробки і модернізації програмного забезпечення.

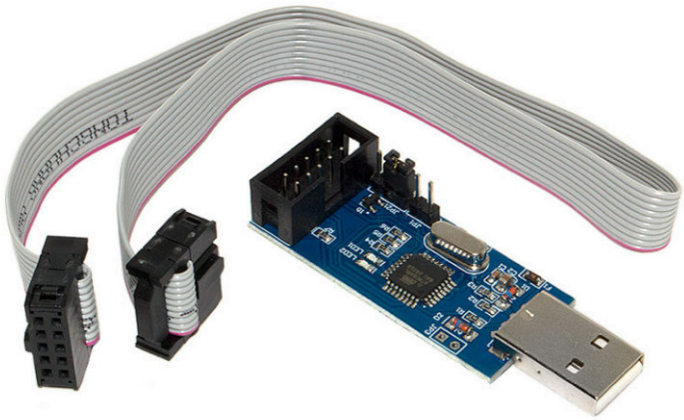


Рис. 1.104. Внутрішньосхемний програматор для мікроконтролерів AVR, підключається до комп'ютера через USB порт, до мікроконтролера – через інтерфейс SPI

3. *Послідовне програмування при високій напрузі*. Цей режим вимагає додаткового джерела підвищеної напруги (12 В) і застосовується, як правило, для програмування мікроконтролерів перед установкою їх на плату.

4. *Програмування по інтерфейсу JTAG*. Joint Test Action Group – спеціалізований апаратний інтерфейс, розроблений для тестування зібраних друкованих плат, рис. 1.105. При програмуванні з використання цього інтерфейсу можливе тестування друкованих плат, програмування і конфігурування кристала, здійснення внутрішньосхемного відлагодження.

Рис. 1.105. Програматор AVR JTAG Atmel's AVR Studio ICE

5. *Самопрограмування* – зміна вмісту пам'яті програм самим мікроконтролером.

1.4.9. Питання для самоперевірки

1. Розкрийте загальну структуру типової мікропроцесорної системи.
2. Назвіть особливості шинної організації обміну даними в мікропроцесорній системі.

3. Чим відрізняються машинний цикл та машинний такт?

4. Які складові входять до складу мікропроцесора Intel 8080?

5. Опишіть функціонування мікропроцесора Intel 8080 за структурною схемою.

6. Опишіть цикл виконання команди.

7. Які операції має виконати процесор при виконанні команди, пов'язаної із зверненням до пам'яті?

8. В чому полягає концепція віртуальної машини?

9. Що передбачає інтерпретація програми?

10. Що передбачає трансляція програми?

11. Назвіть рівні віртуальних машин, що використовуються в комп’ютері.

12. Дайте визначення мікроконтролеру.

13. Чим мікроконтролер відрізняється від мікропроцесора?

14. Розкрийте структуру типового мікроконтролера.

15. Які існують сімейства мікроконтролерів AVR?

16. Які сімейства плат Arduino Вам відомі?

17. Які способи завантаження кодів програми до flash-пам’яті мікроконтролера Вам відомі?

1.5. Мови програмування мікроконтролерів

Основи мови асемблера. Цикл трансляції, компонування та виконання програми. Структура програми на мові асемблера. Система команд мікроконтролерів AVR. Директиви асемблера. Мова програмування C for Arduino.

1.5.1. Основи мови асемблера

Мова асемблера – мова програмування низького рівня, мнемонічні команди якої відповідають інструкціям процесора обчислювальної системи. Текст програми, написаний на мові асемблера, має бути перекладений у виконуваний машинний код, тобто необхідно здійснити трансляцію програми. Для цього використовується програма-транслятор, яка називається асемблером (від англ. assembler – збирач). Ця програма і дала мові асемблера його назву. Таким чином, асемблер – комп'ютерна програма-транслятор (компілятор) початкового тексту програми, написаної на мові асемблера, в програму на машинній мові.

Компіляція – трансляція програми на машинну мову.

Отже, програму, написану на мові асемблера, не можна безпосередньо запустити на комп'ютері. Спочатку її треба відтранслювати (асемблювати) у виконуваний код.

В результаті роботи асемблера початковий текстовий файл перетворюється в бінарний файл, що називається об'єктним файлом і містить машинний код. Безпосередньо об’єктний файл не можна запустити на виконання. Його треба обробити за допомогою спеціальної програми – компонувальника (linker), інша назва – редактор зв’язків (linkage editor), яка створює виконуваний файл. Саме цей файл і можна запустити на виконання.

1.5.2. Цикл трансляції, компонування та виконання програми

Процес редагування початкового файлу на асемблері (тобто написання програми), його компіляції, компонування і виконання схематично показані на рис. 1.106.

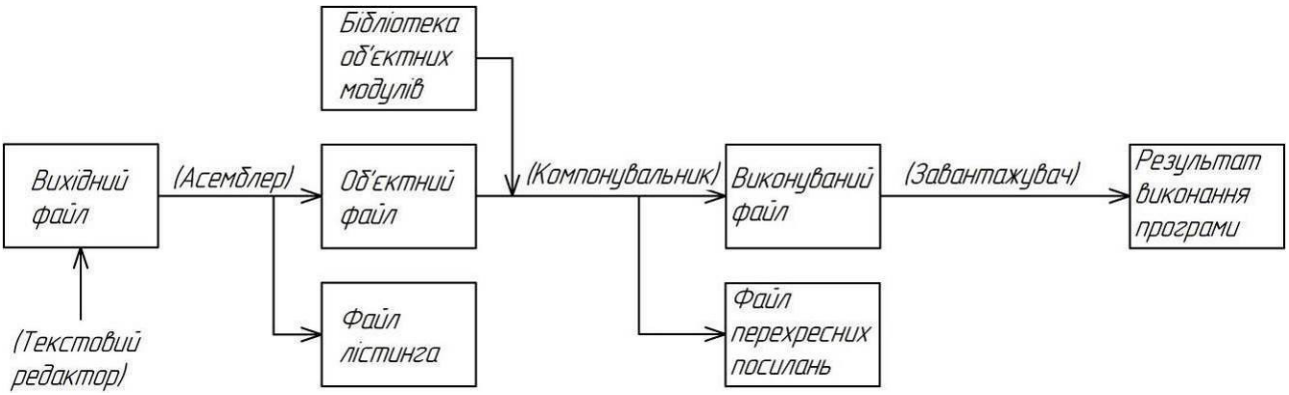


Рис. 1.106. Схематичне зображення циклу трансляції, компонування та виконання програми

Розглянемо кожен крок.

1. За допомогою текстового редактора програміст створює початковий текстовий файл (source file), що містить програму на асемблері.

2. На вхід програми асемблера подається початковий файл, а на виході присутній об'єктний файл, що містить машинний код. Асемблер також може створити файл лістингу (listing file) програми. Якщо при компіляції виникнуть помилки, необхідно повернутися до п. 1 і виправити їх.

Файл лістингу програми містить, окрім тексту програми, номери рядків, адреси команд (їх зміщень відносно сегмента коду), відтрансльований машинний код, представлений в шістнадцятковому виді, і таблицю символів.

3. Вміст об'єктного файлу аналізується компонувальником. Він визначає, чи є в програмі зовнішні посилання, тобто чи містить програма команди виклику процедур, що знаходяться в одній з бібліотек об'єктних модулів (link library). Компонувальник знаходить ці посилання в об'єктному файлі програми, копіює необхідні процедури з бібліотек, об'єднує їх разом з об’єктним файлом (цей процес називається розв’язанням зовнішніх посилань) і створює виконуваний файл (executable file). Також компонувальник може створити файл перехресних посилань (map file).

4. Компонент операційної системи, що називається завантажувачем (loader), прочитує дані з виконуваного файлу, завантажує програму в пам'ять і передає управління за адресою точки входу і забезпечує взаємодію програми з пристроями введення-виводу, файловими системами і іншими програмами. В результаті програма починає виконуватися.

Отже, щоб програма виконалася на будь-якій операційній системі, вона має бути скомпільована у виконуваний файл. Основними форматами виконуваних файлів є COM і EXE.

Файли типу COM містять тільки скомпільований код без будь-якої додаткової інформації про програму. Увесь код, дані і стек такої програми розташовуються в одному сегменті і не можуть перевищувати 64 Кб.

Файли типу EXE містять заголовок, де описується розмір файлу, необхідний об'єм пам'яті, список команд в програмі, що використовують абсолютні адреси, які залежать від розташування програми в пам'яті, і т. д. EXE-файл може мати будь-який розмір.

Програми на асемблері, написані для мікроконтролера, компілюються в шістнадцяткові файли з розширенням *.hex.

Для написання та трансляції програм на мові асемблера для мікроконтролерів AVR може бути використано середовище Microchip Studio, рис. 1.107.

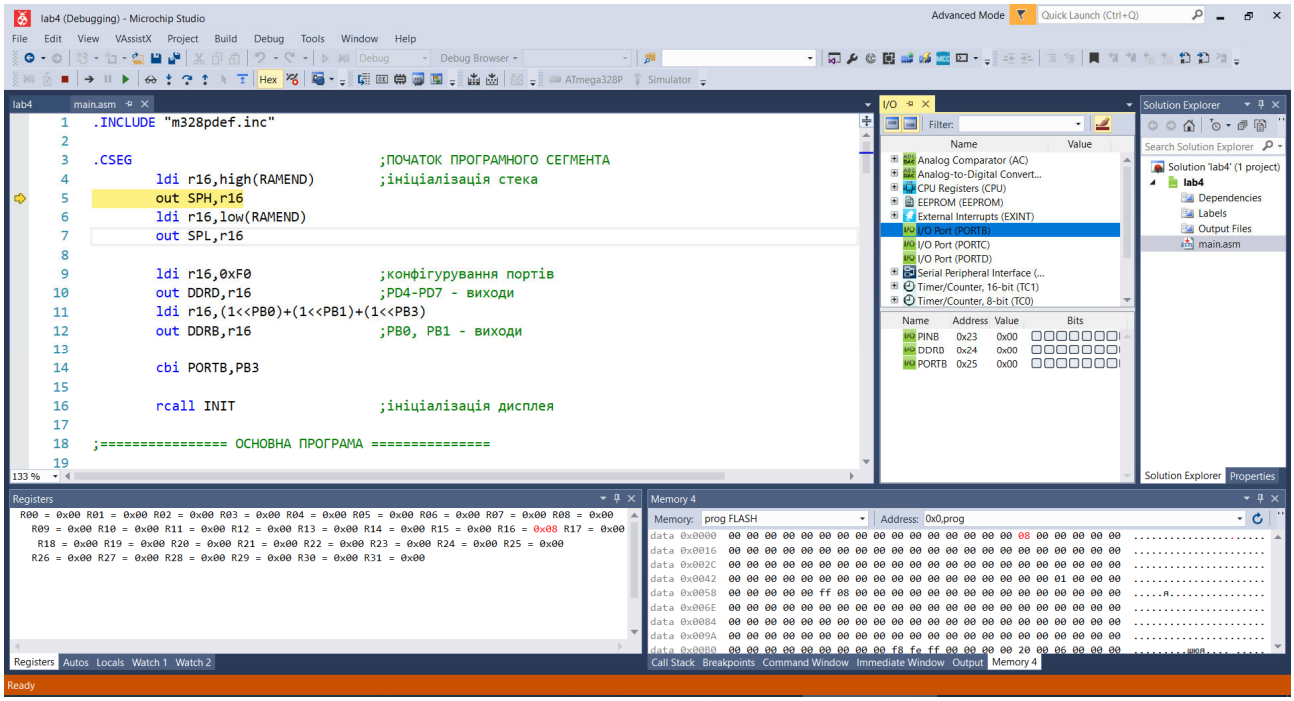


Рис. 1.107. Відлагодження програми для мікроконтролера AVR на мові асемблера у середовищі Microchip Studio

1.5.3. Структура програми на мові асемблера

Програма на мові асемблера складається з рядків, що мають наступний вигляд:

```
мітка:    мнемоніка команди    операнди        ; коментар
```

Причому усі ці поля необов'язкові.

Мітка може бути будь-якою комбінацією букв англійського алфавіту, цифр і символів _, $, @, ?, але цифра не може бути першим символом мітки, а символи $ і ? іноді мають спеціальні значення і зазвичай не рекомендуються до використання. Великі і маленькі букви за умовчанням не розпізнаються, але відмінність можна включити, задавши ту або іншу опцію в командному рядку асемблера.

У полі мнемоніки команди розташовується коротке ім'я команди процесора, яка транслюється у виконуваний код. Також в цьому полі може розташовуватися директива, яка не призводить до появи нового коду, а управляє роботою самого асемблера.

У полі операндів розташовуються потрібні команди або директиви. Як операнд в команді може використовуватися назва регістра, посилання на ділянку пам'яті, константа, адреса порту введення-виводу.

У полі коментарів, початок якого відзначається символом; (крапка з комою), можна написати пояснення – текст від символу; до кінця рядка не аналізується асемблером.

Для полегшення сприйняття асемблерних текстів прийнято, що мітка починається на першій позиції в рядку, команда – на 17-ій (дві табуляції), операнди – на 25-ій (три табуляції) і коментарі – на 41-ій або 49-ій. Якщо рядок складається тільки з коментаря, його починають з першої позиції.

Структурно програма на мові асемблера складається з трьох основних сегментів:

1) сегмент коду – визначає вміст області пам'яті, в якій знаходяться команди;

2) сегмент даних – визначає вміст області пам'яті з даними;

3) сегмент стека – визначає вміст області пам'яті, яка відведена під стек.

1.5.4. Система команд мікроконтролерів AVR

Система команд мікроконтролерів AVR сімейств Tiny і Mega налічує в різних моделях від 90 до 133 інструкцій. Повний опис системи команд конкретного мікроконтролера доступний у технічному описі (datasheet).

Доступ до регістрів введення/виводу здійснюється за їх адресами, що є операндами команди. В той же час при написанні програм на асемблері набагато зручніше звертатися до регістрів, використовуючи замість числових значень адрес їх стандартні символічні імена. Щоб задати відповідність цих імен реальним адресам, необхідно підключити на початку програми (за допомогою директиви асемблера .INCLUDE) файл визначення адрес регістрів введення/виводу для мікроконтролера конкретного типу. Назви цих файлів мають наступний формат:

```
<модель мікроконтролера>def.inc
```

Наприклад, програма для мікроконтролера ATtiny15 повинна містити наступну директиву асемблера:

```
.include "tn15def.inc"
```

для мікроконтролера ATmega128 :

```
.include "m128def.inc"
```

Всі команди мікроконтролерів AVR сімейств Tiny та Mega розбиті на декілька груп.

1) *Команди логічних операцій* – дозволяють виконувати стандартні логічні операції над байтами, а також очищення (установку) регістрів і перестановку тетрад. Операції виконуються між РОН або між регістром і константою,

результат зберігається в РОН.

Наприклад:

AND Rd, Rr

- виконання логічного AND між вмістом регістрів Rd і Rr і приміщення результату в регістр призначення Rd.

```
and r2, r3  ;Порозрядне and r2 і r3, результат помістити в r2
```

OR Rd, Rr

- команда виконує логічне OR вмісту регістрів Rd і Rr і розміщує результат в регістрі призначення Rd.

```
or r15, r16  ;Виконати порозрядне "АБО" між регістрами
```

2) *Команди арифметичних операцій і команди зсуву* – дозволяють виконувати складання, віднімання, зсув (вправо і вліво), інкремент і декремент. Усі операції виконуються тільки над регістрами загального призначення.

Наприклад:

ADD Rd, Rr

- складання двох регістрів без додавання прапора перенесення (C), розміщення результату в регістрі призначення Rd.

```
add r1, r2   ; Скласти r2 з r1 (r1=r1+r2)
```

3) *Команди операцій з бітами* – команди, що виконують установку або скидання заданого розряду РОН або РВВ.

Наприклад:

CBI A, b

- скидає розряд b регістра введення-виводу, розташованого за адресою A.

```
cbi PORTB, PB0 ;Скинути нульовий розряд порту B
```

SBI A, b

- встановлює розряд b регістра введення-виводу, розташованого за адресою A.

```
sbi PORTB, PB1 ;Встановити розряд 1 порту B
```

4) *Команди пересилки даних* – призначені для пересилання вмісту комірок, що знаходяться в адресному просторі пам'яті даних. Поділ адресного простору на три частини (регістри загального призначення РЗП, регістри введення/виведення РВВ, оперативний запам’ятовуючий пристрій ОЗП) визначило можливість пересилання в наступних напрямах:

– РЗП <=> РЗП;

– РЗП <=> РВВ;

– РЗП <=> пам'ять даних.

Також до цієї групи можна віднести стекові команди PUSH і POP (відсутні в мікроконтролерах сімейства Tiny), що дозволяють зберігати в стеку і відновлювати із стека вміст РОН.

Наприклад:

MOV Rd, Rr

- копіює вміст регістра Rr в регістр Rd, регістр-джерело Rr не змінюється

```
mov r16, r18  ;переслати вміст r18 в r16
```

LDI Rd, K

- завантажується 8-розрядна константа в регістр від r16 по r31.

```
ldi r16,25  ;завантажити константу 25 в r16.
```

5) *Команд передачі управління* – команди переходу, виклику підпрограм і повернення з них і команди типу «перевірка/пропуск». Також до цієї групи відносяться команди порівняння, що встановлюють прапори регістра SREG і призначені, як правило, для роботи спільно з командами умовного переходу.

Наприклад:

rjmp k

- команда відносного безумовного переходу за адресою, що дорівнює сумі лічильника команд і константи k. Замість числових зміщень використовуються мітки

```
rjmp L1
. . .
L1:
```

6) *Команди управління системою* – включають наступні команди: NOP – порожня команда; SLEEP – переведення мікроконтролера в режим зниженого енергоспоживання; WDR – скидання вартового таймера.

1.5.5. Директиви асемблера

Директиви асемблера – ключові слова в тексті програми на мові асемблера, що впливають на процес асемблювання або властивості вихідного файлу. Директиви не транслюються безпосередньо в код. Замість цього вони використовуються для вказання положення в програмній пам'яті, визначення макросів, ініціалізації пам'яті тощо. Перед всіма директивами ставиться крапка.

1) Директиви резервування пам'яті

1.1) .BYTE – зарезервувати байти в ОЗП. Допустима в сегменті даних .DSEG. Параметр – кількість зарезервованих байт.

1.2) .DB – зарезервувати байти в пам'яті програм або EEPROM. Може бути розміщена в сегменті програм .CSEG або в сегменті EEPROM .ESEG. Параметри, що передаються директиві, – послідовність виразів, розділені комами. Кожен вираз має бути числом в діапазоні (–128…255).

1.3) .DW – зарезервувати слова в пам'яті програм або EEPROM. Може бути розміщена в сегменті програм .CSEG або в сегменті EEPROM .ESEG. Параметри, що передаються директиві, – послідовність виразів, розділених комами. Кожне вираження має бути числом в діапазоні (–32768…65535).

Якщо необхідно посилатися на виділену область пам'яті, то перед відповідною директивою має бути мітка.

2) Директиви сегментації

2.1) .CSEG – визначає початок програмного сегменту. Програмні сегменти мають свої власні лічильники положення.

2.2) .DSEG – визначає початок сегменту даних. Сегменти даних мають свої власні побайтові лічильники положення.

2.3) .ESEG – визначає початок сегменту EEPROM. Сегменти EEPROM мають свої власні побайтові лічильники положення.

2.4) .ORG – встановлює лічильник положення рівним заданій величині, яка передається як параметр. Для сегменту даних директива .ORG встановлює

побайтовий лічильник положення в SRAM (ОЗП), для сегменту програм – це програмний послівний лічильник, а для сегменту EEPROM – це положення в EEPROM (побайтовий лічильник положення). Перед початком компіляції програмний лічильник і лічильник EEPROM дорівнюють нулю, а лічильник ОЗП дорівнює 32 (оскільки адреси 0–31 зайняті регістрами).

```
            .ESEG
eevar1:     .DW 0xffff  ;ініціалізувати 1 слово в EEPROM
consts:     .DB 0, 255, 0b01010101, - 128, 0xAA
            .DSEG       ;Початок сегменту даних
var1:       .BYTE 4     ;Резервує 4 байти в ОЗП
            .ORG 0x37   ;Встановити адресу ОЗП 0x37
variable:   .BYTE 1     ;Зарезервувати байт за адресою 0x37H
            .CSEG       ;Початок програмного сегменту
const:      .DW 2       ;Розмістити константу 0x0002 в пам'яті програм
            .ORG 0x10   ;Встановити програмний лічильник рівним 0x10
           mov r1, r0  ;Виконати дії
```

3) Директиви символічних імен

3.1) .DEF – призначити регістру символічне ім'я. Дозволяє посилатися на регістр через деяке символічне ім'я. Призначене ім'я може використовуватися в усій наступній частині програми для звернень до цього регістра. Регістр може мати декілька різних імен. Символічне ім'я може перепризначатися пізніше в програмі.

Приклад:

```
.DEF temp=R16
.CSEG
 ldi temp, 0xf0  ;Завантажити 0xf0 в регістр temp (R16)
```

3.2) .EQU – встановити постійний вираз. Директива привласнює мітці значення. Ця мітка може пізніше використовуватися у виразах.

Приклад:

```
.EQU t = 0x23
.EQU pa = t + 2
.CSEG            ; Початок сегменту даних
out pa, r2       ; Записати в порт A
```

3.3) .SET – встановити змінний символічний еквівалент виразу. Директива привласнює імені деяке значення. Це ім'я пізніше може використовуватися у виразах. Причому, на відміну від директиви .EQU, значення імені може бути змінене іншою директивою SET.

Приклад:

```
.SET t = 0x23
.SET pa = t + 2
.CSEG              ; Початок кодового сегменту
out pa, r2           ; Записати в порт A
```

4) Директиви створення макросів

4.1) .MACRO – початок макросу. В якості параметру директиві передається ім’я макросу. Якщо в тексті програми зустрічається ім’я макросу, компілятор замінює це ім’я на тіло макросу. Макрос може мати до 10 параметрів, до яких в його тілі звертаються через @0–@9. Під час виклику параметри перераховуються через коми.

4.2) .ENDM, .ENDMACRO – кінець макросу

Приклад:

```
.MACRO SUBI16               ; Початок визначення макрос
subi r16, low(@0)           ; Відняти молодший байт першого параметра
sbci r17, high(@0)          ; Відняти старший байт першого параметра
.ENDMACRO
```

5) Директиви компіляції

5.1) .DEVICE – визначити пристрій, для якого компілюється програма. Компілятор видасть попередження, якщо буде знайдена інструкція, яку не підтримує цей мікроконтролер. Приклад:

```
.DEVICE ATtiny22
```

5.2) .EXIT – вийти з файлу. Зустрівши директиву .EXIT, компілятор припиняє компіляцію цього файлу.

5.3) .INCLUDE – вкласти інший файл. Зустрівши директиву .INCLUDE, компілятор відкриває вказаний в ній файл, компілює його, доки файл не закінчиться або не зустрінеться директива .EXIT, після цього продовжує компіляцію початкового файлу з рядка, що слідує за директивою .INCLUDE. Вкладений файл може також містити директиви .INCLUDE. Синтаксис:

```
.INCLUDE "ім'я_файлу"
```

5.4) .LIST – дозволити генерацію лістинга.

5.5) .NOLIST – заборонити генерацію лістинга.

5.6) .LISTMAC – дозволити розгортання макросів в лістингу.

Компілятор AVR підтримує наступні *вбудовані функції*:

1) LOW(вираз) – повертає молодший байт виразу (слова).
2) HIGH(вираз) – повертає старший байт виразу (слова).
3) BYTE2(вираз) – те ж, що і функція HIGH.
4) BYTE3(вираз) – повертає третій байт виразу.
5) BYTE4(вираз) – повертає четвертий байт виразу.
6) LWRD(вираз) – повертає біти 0–15 виразів.
7) HWRD(вираз) – повертає біти 16–31 виразів.
8) PAGE(вираз) – повертає біти 16–21 виразів.
9) EXP2(вираз) – повертає 2 в степені, що відповідає виразу.
10) LOG2(вираз) – повертає цілу частину log2(вираз).

1.5.6. Мова програмування C for Arduino

Розроблення власних програм на базі плат Arduino здійснюється в офіційному безкоштовному середовищі програмування Arduino IDE (https://www.arduino.cc/en/software), рис. 1.108. Середовище призначене для написання, компіляції та завантаження власних програм до мікроконтролера, встановленого на платі Arduino. Основою середовища є мова Processing/Wiring – це звичайний C++, доповнений простими і зрозумілими функціями керування апаратною частиною мікроконтролера.

Інтерфейс середовища Arduino, рис. 1.108, містить такі основні елементи: текстовий редактор для написання коду, область для виведення повідомлень, текстова консоль, панель інструментів із традиційними кнопками та головне меню. Дана програма дозволяє комп'ютеру взаємодіяти з платою як обміну даними, так прошивки коду в контролер.

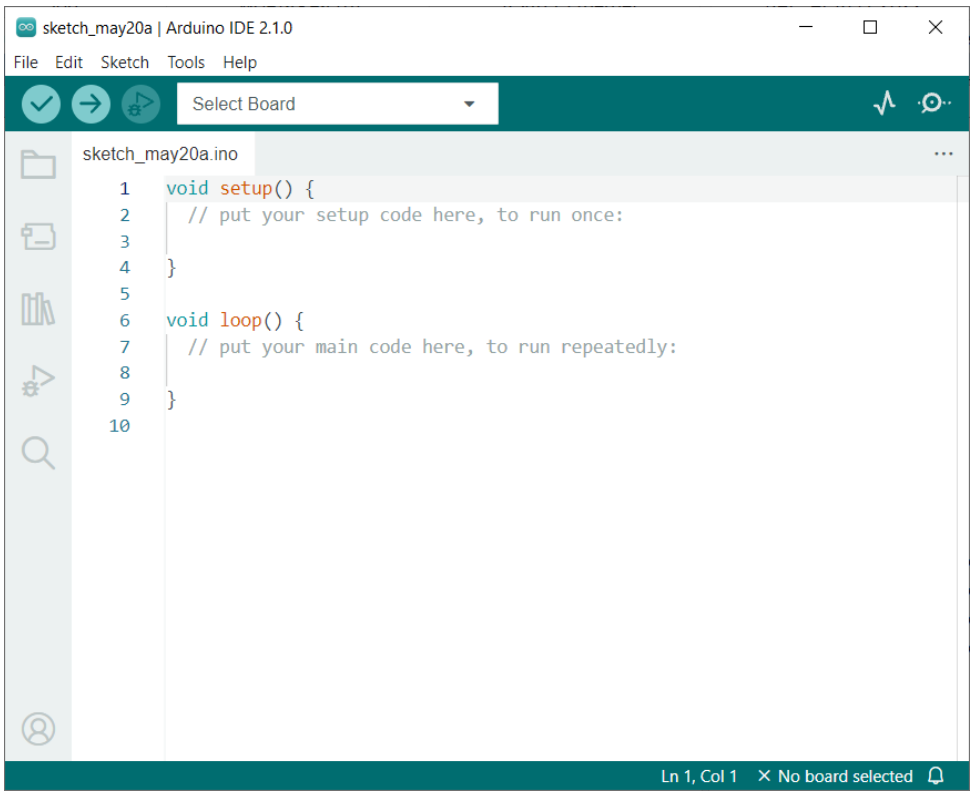


Рис. 1.108. Середовище Arduino IDE

Програми для Arduino називають скетчами (sketch). Скетчі пишуться у текстовому редакторі та зберігаються у файлах з розширенням .ino. Вбудований текстовий редактор має стандартні інструменти копіювання, вставки, пошуку та заміни тексту. Область повідомлень у вікні програми є свого роду зворотним зв'язком для користувача та інформує його про події (у тому числі і про помилки), що виникають у процесі запису або експорту написаного коду. Консоль відображає у вигляді тексту потік вихідних даних середовища Arduino, включаючи всі повідомлення про помилки та іншу інформацію. У правому нижньому куті вікна програми показується модель поточної плати і послідовний порт, до якого вона підключена. Кнопки на панелі інструментів призначені для створення, відкриття, збереження та прошивки програм у пристрій. Окрема кнопка запускає програму SerialMonitor.

Базова структура програми для Arduino складається щонайменше з двох обов’язкових частин: функцій setup() і loop(). Перед функцією setup() йде оголошення змінних, підключення бібліотек.

Функція setup() запускається один раз після кожного ввімкнення живлення або скидання плати Arduino. Вона використовується для ініціалізації змінних, встановлення режиму роботи портів та інших підготовчих для основного циклу програми дій. Вона обов'язково має бути включена до програми, навіть якщо не виконує жодних дій.

Функція loop() у нескінченному циклі послідовно виконує команди, які описані у її тілі. Ця функція виконується циклічно, вона виконує основний алгоритм функціонування мікроконтролера.

Розглянемо синтаксис та оператори мови програмування Processing/Wiring (C for Arduino).

Оператори

Оператори управління

if. Оператор if використовується в поєднанні з операторами порівняння, він перевіряє виконання умови, наприклад, чи перевищує вхідне значення задану величину. Формат оператора if:

```
if (Variable > 100) {
       // дії
}
```

if...else. Дозволяє виконувати різні дії у випадках виконання та не виконання умова:

```
if (Input < 500) {
  // дія A
} else {
  // дія B
}
```

switch case. Конструкція дозволяє реалізовувати множинний вибір. Значення var порівнюється зі значеннями label1, label2…

```
switch (var)
{
case label1:
      // код для виконання при var=label1
break;
case label2:
      // код для виконання при var=label2
break;
case label3:
      // код для виконання при var=label3
break;
default:
      // код для виконання якщо жодна умова не була виконана
break;
}
```

for. Цикл із визначеною кількістю повторень. Формат:

for (initialization; condition; increment) {оператори, що виконуються в циклі}

Ініціалізація (initialization) виконується найпершою та один раз. Зазвичай це оголошення лічильника циклу. Кожен раз у циклі перевіряється умова (condition), якщо вона вірна, виконується блок операторів і лічильник циклу збільшується (increment), потім умова перевіряється знову. Коли логічне значення умови стає хибним, цикл завершується. Наприклад:

```
for (int i=0; i <= 135; i++)
{
  analogWrite(PWMpin, i);
  delay(5);
}
```

while. Цикл із перевіркою умови на початку. Оператори в циклі будуть виконуватися доти, поки вираз у круглих дужках є істинним.

```
while(вираз)
{
   // оператори
}
```

do... while. Цикл із перевіркою умови в кінці. Оператори в циклі будуть виконуватися доти, поки вираз у круглих дужках є істинним.

```
do {
// оператори
```

```
} while(вираз)
```

break. Оператор використовується для примусового виходу з циклів do, for або while, не чекаючи завершення циклу за умовою. Він також використовується для виходу із оператора switch.

continue. Пропускає оператори, що залишилися, в поточному кроці циклу. Замість них виконується перевірка умови циклу, що відбувається на кожній наступній ітерації.

return. Оператор припиняє обчислення у функції та повертає значення з перерваної функції у основну програму, якщо це потрібно

goto. Примусовий перехід на мітку.

Синтаксис

; (крапка з комою). Використовується для позначення кінця оператора. Кожен оператор програми має закінчуватися крапкою з комою.

{} (фігурні дужки). Використовують для об'єднання операцій у блоки.

// (однорядковий коментар). Символи в рядку, що стоять після //, ігноруються компілятором.

/* */ (багаторядковий коментар). Символи декількох рядків, що знаходяться між /* та */, ігноруються компілятором.

#define. Директива оголошення константи:

```
#define ledPin 3
```

#include. Директива підключення бібліотеки до скетча:

```
#include <avr/pgmspace.h>
```

Арифметичні оператори

= (оператор присвоєння). Присвоює змінній ліворуч від оператора значення змінної або виразу, що знаходиться праворуч:

```
Val=analogRead(0); // Присвоїти змінній Val значення, зчитане з
                   //  аналогового входу 0
```

+ (додавання), **-** (віднімання), ***** (множення), **/** (ділення). Повертають результат виконання арифметичних дій над двома операндами.

% (залишок від ділення). Повертає залишок від ділення одного цілого (int) операнда на інший:

```
x = 9 % 5; // x має значення 4
```

Оператори порівняння

== (дорівнює); **!=** (не дорівнює); **<** (меньше ніж); **>** (більше ніж);
<= (менше або дорівнює); **>=** (більше або дорівнює).

Логічні оператори

&& (І).
|| (АБО).
! (НІ).

Дані

Константи

HIGH | LOW – високий/низький рівень сигналу на фізичному виводі мікроконтролера.

INPUT | OUTPUT| INPUT_PULLUP – конфігурація вивода мікросхеми: вхід/вихід/вхід з внутрішнім підтягуючим резистором.

true | false – логічні значення.

Цілочисельні константи – за замовчуванням у десятковій системі, префікс двійкових чисел «B», шістнадцяткових «0x».

Константи з плаваючою комою, наприклад: 11.0; 8.56E5; 81e-11.

Типи даних

void – ключове слово void використовується лише при оголошенні функцій. Воно вказує на те, що функція, яка оголошується, не повертає ніякого значення тій функції, з якої вона була викликана.

boolean – логічний (булевий) тип даних, змінні типу boolean можуть набувати одне з двох значень: true або false.

char – тип даних, що призначений для збереження одної літери: char C = 'B'; char C = 66.

byte – зберігає 8-бітове числове значення без десяткової точки. Має діапазон від 0 до 255.

unsigned char – беззнаковий тип даних, що у пам'яті 1 байт. Те саме, що й тип даних byte. Змінні типу unsigned char можуть зберігати значення від 0 до 255.

int – цілочислений тип, int займає 2 байти пам’яті і може зберігати числа від –32 768 до 32 767.

unsigned int – беззнакове ціле число, так само як і тип int (знакове), займає в пам'яті 2 байти. Але, на відміну від int, тип unsigned int може зберігати тільки додатні цілі числа в діапазоні від 0 до 65 535.

word – змінні типу word зберігають 16-бітове беззнакове число в діапазоні від 0 до 65 535. Те саме, що і unsigned int.

long – змінні типу long мають розмірність 32 біти (4 байти), що дозволяє їм зберігати числа в діапазоні від –2 147 483 648 до 2 147 483 647.

unsigned long – змінні мають розмірність 32 біти (4 байти). Змінні типу unsigned long, на відміну від звичайного long, зберігають лише позитивні числа в діапазоні від 0 до 4 294 967 295 (2^{32}–1).

short – змінні типу short містять 16-бітові (2-байтові) значення, що дозволяє зберігати в них числа від –32 768 ($–2^{15}$) до 32 767 (2^{15}–1).

float – тип даних для чисел із плаваючою комою (чисел із десятковим роздільником). Числа з плаваючою комою часто використовуються для подання аналогових або безперервних величин, оскільки дозволяють описати їх точніше, ніж цілі числа. Числа з плаваючою комою являють собою 32 біти (4 байти) інформації і можуть досягати значень від $–3{,}4028235 \cdot 10^{38}$ до $3{,}4028235 \cdot 10^{38}$. Точність дробових чисел типу float становить 6–7 десяткових знаків.

double – дробовий тип подвійної точності. У платах на базі мікроконтролерів ATmega змінні типу double займають 4 байти. Іншими

словами, у цих пристроях змінні типу double повністю аналогічні змінним float без будь-якого приросту точності.

string – текстовий рядок. Текстові рядки можуть бути оголошені двома способами: можна використовувати тип даних String; або оголосити рядок як масив символів char з нульовим символом вкінці.

Функції

Цифрове введення/виведення

pinMode() – визначає режим роботи зазначеного виводу: як вхід або як вихід. Синтаксис: pinMode(pin, mode).

digitalWrite() – встановлює значення цифрового виводу (HIGH або LOW). Якщо функцією pinMode() вивід налаштований як вихід (OUTPUT), то при виконанні функції digitalWrite() його напруга буде змінена на відповідне значення: 5 В при значенні HIGH, 0 В (земля) – при LOW. Синтаксис: digitalWrite(pin, value).

digitalRead() – зчитує рівень сигналу HIGH або LOW із зазначеного цифрового виводу. Синтаксис: digitalRead(pin).

Аналогове введення/виведення

analogReference() – встановлює джерело опорної напруги, що використовується при зчитуванні аналогового сигналу (іншими словами, задає максимальне значення вхідного діапазону). Для вибору джерела опорної напруги доступні такі значення:

DEFAULT: опорна напруга за умовчанням, що дорівнює 5 В (на 5В-платах) або 3,3 В (на 3,3В-платах);

INTERNAL: внутрішня опорна напруга, що дорівнює 1,1 В або 2.56 В;

INTERNAL1V1: внутрішня опорна напруга 1,1 В;

INTERNAL2V56: внутрішня опорна напруга 2,56 В;

EXTERNAL: в якості джерела опорної напруги буде використовуватися напруга, прикладена до виводу AREF (від 0 до 5В).

analogRead() – зчитує величину напруги із зазначеного аналогового виводу. Повертає ціле число int (від 0 до 1023). Синтаксис: analogRead(pin).

analogWrite() – Формує задану аналогову напругу на виводі у вигляді ШІМ-сигналу. Після виклику analogWrite(), на виводі безперервно генеруватиметься ШІМ-сигнал із заданим коефіцієнтом заповнення до наступного виклику функції analogWrite() (або до моменту виклику digitalRead() або digitalWrite() для того самого виводу). Частота ШІМ становить приблизно 490 Гц. Синтаксис: analogWrite(pin, value).

Функції для роботи з часом

millis() – повертає кількість мілісекунд, що минули з моменту старту програми Ардуїно. Лічильник переповниться (скинеться в 0) приблизно через 50 днів.

micros() – повертає кількість мікросекунд, що пройшли з початку виконання програми Arduino. Лічильник переповниться (скинеться в 0) приблизно через 70 хвилин.

delay() – зупиняє виконання програми на вказаний проміжок часу (у мілісекундах). Синтаксис: delay(ms).

delayMicroseconds() – зупиняє виконання програми на вказаний проміжок часу (у мікросекундах. На даний момент найбільше число, що дозволяє сформувати точну затримку, 16383. Синтаксис: delayMicroseconds(us).

Математичні функції

min() – повертає мінімальне з двох чисел, min(x, y).

max() – повертає максимальне з двох чисел, max(x, y).

abs() – обчислює модуль числа.

constrain() – обмежує значення змінної заданими межами, сигнатксис: constrain(x, a, b).

map() – перетворює значення змінної з одного діапазону на інший. Значення змінної value, що дорівнює fromLow, буде перетворено в число toLow, а значення fromHigh – в toHigh. Усі проміжні значення value масштабуються щодо нового діапазону [toLow; toHigh]. Синтаксис: map(value, fromLow, fromHigh, toLow, toHigh).

pow() – підносить число base до степеня exponent: pow(base, exponent).

sqrt() – обчислює квадратний корінь числа.

sq() – обчислює квадрат числа.

sin(), **cos(), tan()** – обчислюють тригонометричні функції кута (в рад.).

1.5.7. Питання для самоперевірки

1. Назвіть особливості мови асемблера.
2. Поясніть цикл трансляції, компонування та виконання програми.
3. Яка структура програми на мові асемблера?
4. З яких сегментів складається програма на мові асемблера?
5. Розкрийте систему команд мікроконтролерів AVR.
6. Які директиви асемблера Вам відомі?
7. Які особливості мови C for Arduino (Processing/Wiring)?

1.6. Архітектура та організація пам′яті мікроконтролерів AVR

Структура ядра. Пам'ять даних. Регістри загального призначення та регістри введення-виведення. Ініціалізація та робота зі стеком. FLASH-пам’ять програм. Переривання. Розміщення таблиці векторів переривань.

1.6.1. Структура ядра

Архітектуру ядра мікроконтролера розглянемо на прикладі 8-розрядного контролера ATtiny15L сімейства Tiny.

Основними характеристиками центрального процесора мікроконтролерів сімейства Tiny є:

– повністю статична архітектура;

– АЛП підключене безпосередньо до регістрів загального призначення;

– більшість команд виконуються за один машинний цикл;

– багаторівнева система переривань; підтримка черги переривань;

– 5–8 джерел переривань;

– трирівневий апаратний стек.

Основними характеристиками підсистеми введення/виведення є:

– програмне конфігурування та вибір портів введення/виведення;

– виводи можуть бути запрограмовані як вхідні або як вихідні незалежно один від одного;

– вхідні буфери з тригером Шмітта на усіх виводах;

– можливість підключення до входів внутрішніх підтягуючих резисторів (опір резисторів складає 35–120 кОм).

Набір периферійних пристроїв, наявних у складі мікроконтролера, залежить від конкретної моделі. Загалом у складі мікроконтролерів сімейства зустрічаються наступні периферійні пристрої:

– 8-розрядний таймер/лічильник з переддільником (таймер T0);

– другий 8-розрядний таймер/лічильник з переддільником (таймер T1);

– вартовий таймер WDT;

– одноканальний генератор сигналу з ШІМ розрядністю 8 біт (один з режимів роботи таймера T1);

– аналоговий компаратор;

– 10-розрядний АЦП (4 канали).

Розташування виводів мікроконтролера ATtiny15L приведене на рис. 1.109.

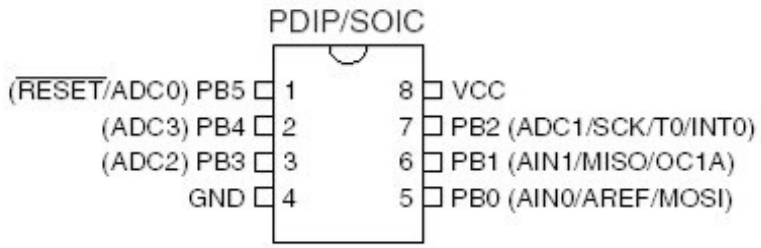


Рис. 1.109. Розташування виводів мікроконтролера ATtiny15L

Кожен вивід має основне призначення (зазвичай – дискретний ввід/вивід або живлення) і додаткове призначення (визначається периферійним пристроєм).

Ядро мікроконтролерів AVR сімейства Tiny виконане за удосконаленою RISC (enhanced RISC) архітектурою (рис. 1.101).

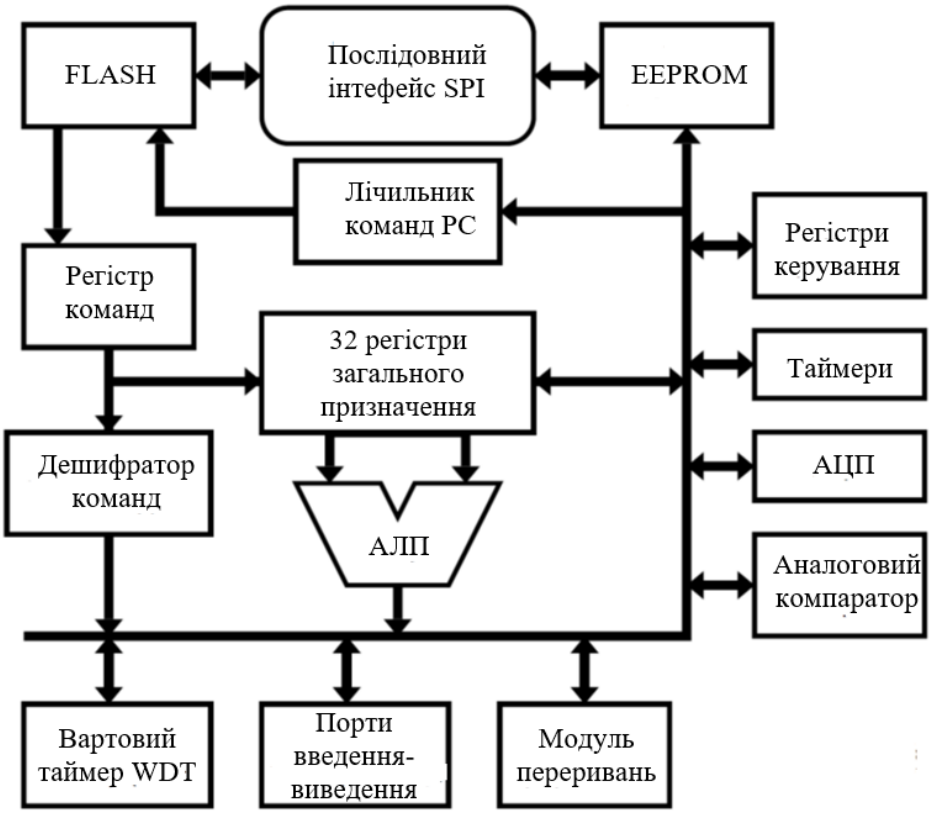


Рис. 1.110. Архітектура ядра мікроконтролерів AVR сімейства Tiny

Арифметико-логічний пристрій забезпечує виконання обчислень. АЛП безпосередньо під'єднано до 32-х регістрів загального призначення, що об'єднані в регістровий файл. Завдяки цьому АЛП виконує одну операцію (читання вмісту регістрів, виконання операції і запис результату назад до регістрового файлу) за один машинний цикл.

Мікроконтролери AVR побудовані відповідно до Гарвардської архітектури, що передбачає роздільні пам'яті програм і даних, кожна з яких має власні шини доступу до них. Це дозволяє одночасно працювати як з пам'яттю програм, так і з пам'яттю даних, а також забезпечує конвеєризацію. У мікроконтролерах AVR використовується 2-рівневий конвеєр.

Пам'ять даних складається з двох областей: регістрова пам'ять і пам'ять на основі EEPROM. Кожна область розміщується у своєму адресному просторі.

1.6.2. Пам'ять даних. Регістри загального призначення та регістри введення-виведення

Пам'ять даних мікроконтролерів сімейства Tiny розділена на дві частини: регістрова пам'ять і незалежне EEPROM. Внутрішнє статичне ОЗП в мікроконтролерах сімейства Tiny відсутнє. Регістрова пам'ять включає 32 регістри загального призначення (РЗП), об'єднаних в файл, і службові регістри введення/виведення (РВВ).

Для довготривалого зберігання різної інформації, яка може змінюватися в процесі функціонування готової системи (калібрувальні константи, серійні номери, ключі і тому подібне) в моделях може бути використана EEPROM-пам'ять. Ця пам'ять розташована в окремому адресному просторі, а доступ до

неї здійснюється за допомогою певних регістрів введення/виводу.

Регістри загального призначення. Усі регістри загального призначення об'єднані у файл. Усі 32 РЗП R0–R31 безпосередньо доступні АЛП, рис. 1.111. Завдяки цьому будь-який РЗП може використовуватися в усіх командах і як операнд-джерело, і як операнд-приймач. Два старші регістри загального призначення (R30 і R31) формують 16-розрядний індексний регістр Z, який використовується як покажчик при непрямій адресації пам'яті програм і пам'яті даних. У мікроконтролерах сімейства Mega регістри R16–R31 можуть об'єднуватися в три 16-розрядні регістри X, Y, Z:

R26 – молодший байт регістра X (символічне ім'я XL);

R27 – старший байт регістра X (XH);

R28 – молодший байт регістра Y (YL);

R29 – старший байт регістра Y (YH);

R30 – молодший байт регістра Z (ZL);

R31 – старший байт регістра Y (ZH).

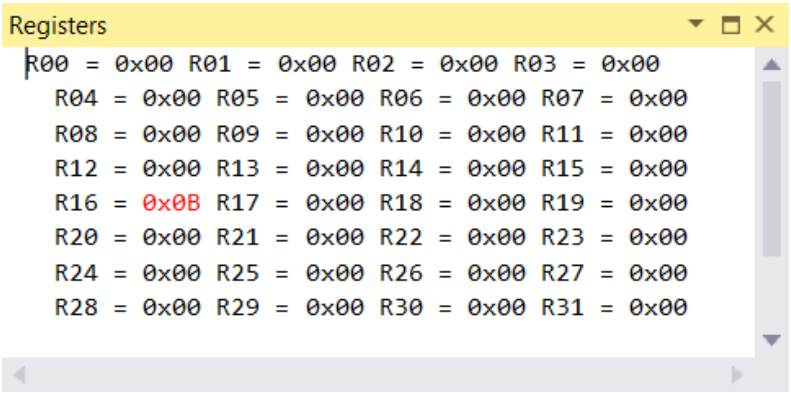


Рис. 1.111. Вікно для відображення значень регістрів загального призначення під час відлагодження програми в середовищі Microchip Studio

Регістри введення/виведення. Усі РВВ можна розділити на дві групи:

– службові регістри мікроконтролера;

– регістри, що відносяться до периферійних пристроїв (у тому числі порти введення/вивода). Розмір кожного регістра – 8 біт.

РВВ призначені для конфігурації мікроконтролера і периферійних пристроїв, а також для введення/виведення інформації через порти, рис. 1.112. Розподіл адрес простору введення/виводу залежить від конкретної моделі мікроконтролера, оскільки різні моделі мають різний склад периферійних пристроїв і відповідно різна кількість регістрів. Усі РВВ мають літерні назви.

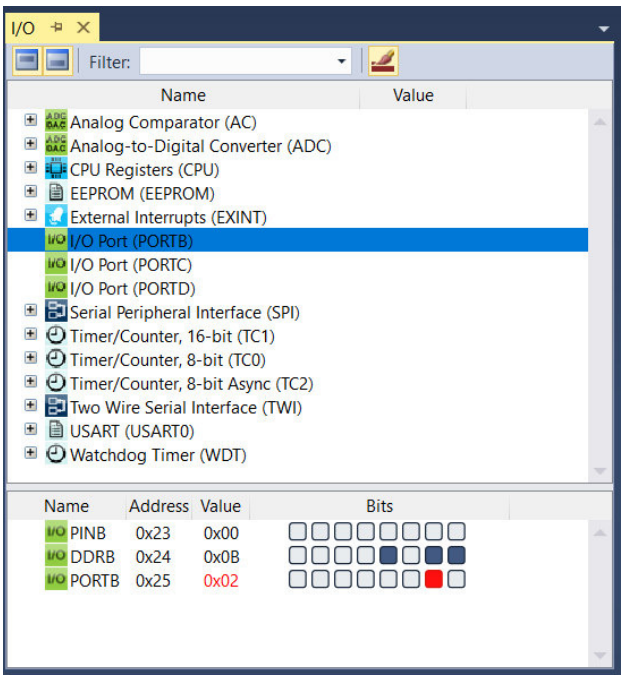


Рис. 1.112. Вікно відображення значень регістрів введення/виведення під час відлагодження програми в середовищі Microchip Studio

Виконання кожної команди оновлює вміст регістра стану SREG (Status Register). Прапори (розряди) цього регістра оновлюються АЛП та свідчать про результат виконання попередньої команди, рис. 1.113. При скиданні мікроконтролера всі прапори автоматично встановлюються в лог. «0». Відповідно до результату виконання команди прапори набувають значень лог. «0» або лог. «1». Регістр стану має наступний формат:

	7	6	5	4	3	2	1	0
SREG	I	T	H	S	V	N	Z	C

Призначення розрядів регістра SREG описане в табл. 1.15.

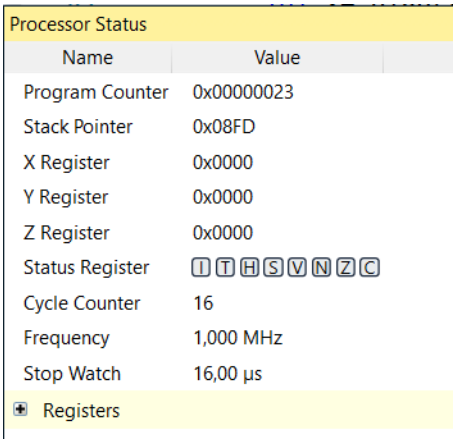


Рис. 1.113. Вікно для стану процесора під час відлагодження програми в середовищі Microchip Studio, серед іншого відображаються прапори регістра стану

Таблиця 1.15

Призначення розрядів (прапорів) регістра SREG

Розряд	Назва	Опис
7	I	Загальний дозвіл переривань. Для дозволу переривань цей прапор має бути встановлений в "1".
6	T	Зберігання біта, що копіюється. Цей розряд регістра використовується як джерело або приймач команд копіювання бітів BLD (Bit LoaD) і BST (Bit STore). Заданий розряд будь-кого РЗП може бути скопійований в цей розряд командою BST або встановлений відповідно до вмісту цього розряду командою BLD.
5	H	Прапор половинного перенесення. Цей прапор встановлюється в "1", якщо сталося перенесення з молодшої половини байта (з 3-го розряду в 4-й) або позичання зі старшої половини байта при виконанні деяких арифметичних операцій.
4	S	Прапор знаку. Цей прапор дорівнює результату операції "Виключаюче АБО" (XOR) між прапорами N (від'ємний результат) і V (переповнення числа в додатковому коді). Прапор встановлюється в "1", якщо результат виконання арифметичної операції менше нуля.
3	V	Прапор переповнення додаткового коду. Цей прапор встановлюється в "1" при переповненні розрядної сітки знакового результату. Використовується при роботі зі знаковими числами.
2	N	Прапор від'ємного значення. Цей прапор встановлюється в "1", якщо старший (7-й) розряд результату операції дорівнює "1". Інакше прапор дорівнює "0".
1	Z	Прапор нуля. Цей прапор встановлюється в "1", якщо результат виконання операції дорівнює нулю.
0	C	Прапор перенесення. Цей прапор встановлюється в "1", якщо в результаті виконання операції стався вихід за межі байта.

В мікроконтролерах Mega та інших сімейства існує статичне ОЗП, доступне для користувача. В адресному просторі ОЗП також розташовані всі регістри мікроконтролерів, під них відведено молодші адреси. Інші адреси відведені під комірки статичного ОЗП користувача, рис. 1.114.

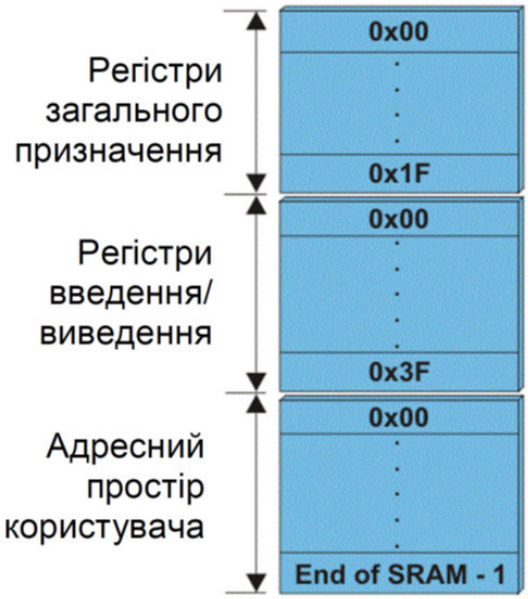


Рис. 1.114. Структура статичного ОЗП в мікроконтролері AVR

1.6.3. Ініціалізація та робота зі стеком

Стек – спеціальним чином виділена область пам’яті, доступ до якої здійснюється за принципом LIFO (Last In First Out – останнім увійшов, першим вийшов). Значення, записане до стека останнім, буде прочитано першим. Стек використовується для впорядкованого збереження адрес повернення до підпрограм та параметрів, що передаються підпрограмам.

Мікроконтролери AVR сімейства Tiny мають апаратно реалізований стек. Глибина стека дорівнює трьом рівням, а розмір дорівнює розміру лічильника команд (9 або 10 розрядів). Стек розташований у власній області пам'яті.

Для мікроконтролерів сімейства Tiny стек програмно недоступний, оскільки в наборі команд мікроконтролера відсутні команди завантаження до стеку і читання зі стека. Покажчик стека також недоступний з програми, тобто він не може бути явно прочитаний або модифікований. Тому мікроконтролер сам управляє переміщенням даних по стеку. Зокрема, до стеку записується адреса повернення в разі виклику підпрограми.

Мікроконтролери Mega та інших сімейств мають програмно реалізований стек. Для його розміщення використовуються старші адреси ОЗП. Адреса чергової вільної комірки стека зберігається у спеціальному регістрі – вказівник стека (SP – stack pointer). У 8-розрядних контролерах вказівник стека має 16-розрядну організацію, для його збереження використовується регістрова пара SPH:SPL. Оскільки після включення контролера в цих регістрах знаходиться нульове значення, то на початку програми покажчик стека потрібно ініціалізувати, записавши в нього значення верхньої адреси пам'яті даних:

```
ldi r16, high(RAMEND)     ;RAMEND - значення верхнього
out SPH, r16              ;адреси пам'яті даних
ldi r16, low(RAMEND)
out SPL, r16
```

Для роботи зі стеком використовуються команди: push – записати дані до

стека (виштовхнути в стек); pop – зчитати дані зі стека (видобути зі стека), рис. 1.115. При виконанні команди push в комірку, адреса якої знаходиться у вказівнику стека, записується вказане число. Після цього вказівник стека збільшується на один для того, щоб вказувати на чергову вільну комірку пам’яті. При зчитуванні значення з вершини стека використовується команда pop, при цьому вказівник стека зменшується на 1.

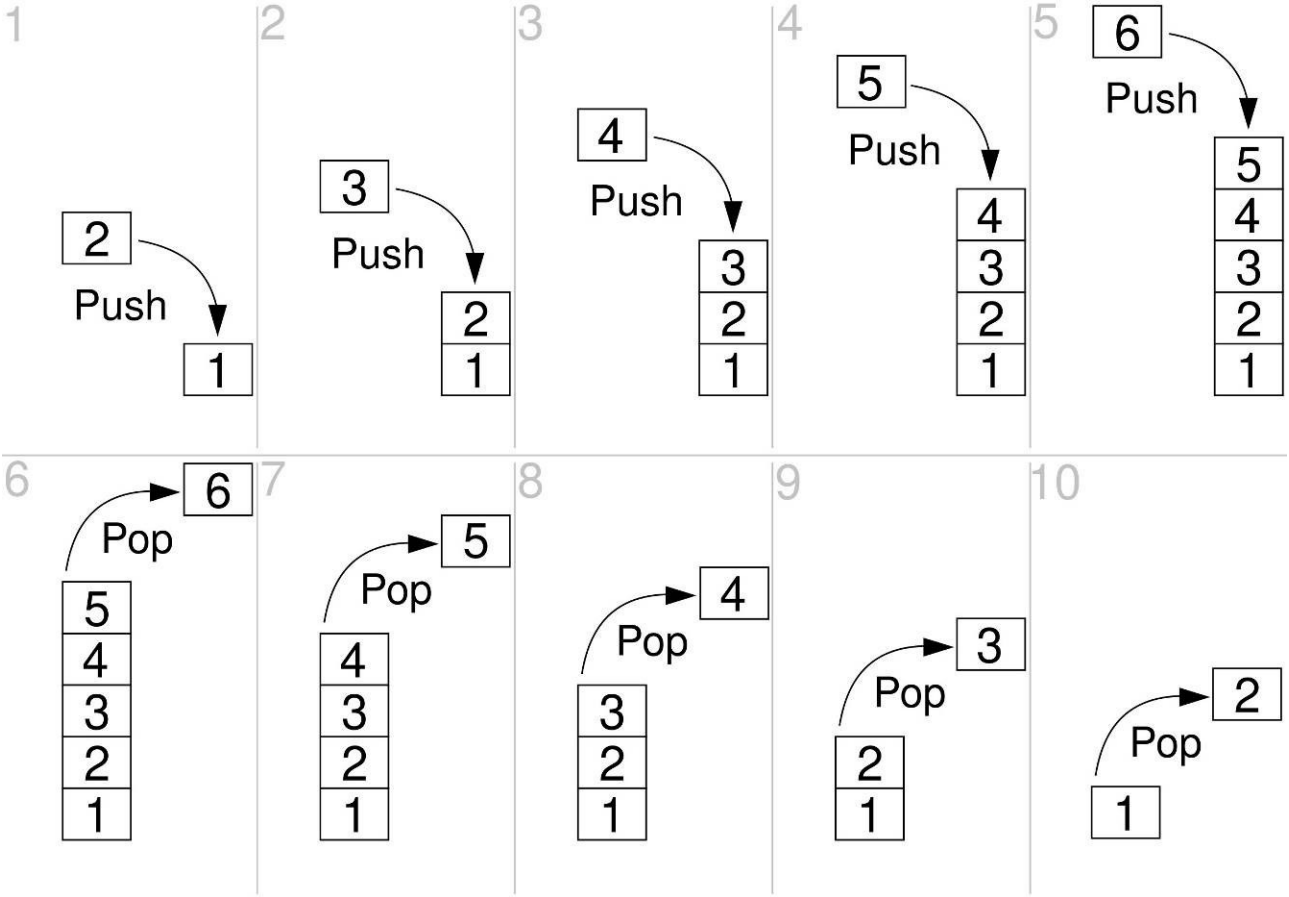


Рис. 1.115. Принцип запису та читання даних зі стека

1.6.4. FLASH-пам’ять програм

Коди команд, що складають програму для мікроконтролера, зберігаються в енергонезалежній пам’яті програм, що виконана за технологією FLASH. На відміну від EEPROM, вміст довільної комірки FLASH-пам’яті не можна змінити. Необхідно повністю очистити таку пам’ять та записати нову програму. Пам’ять програм має 16-розрядну організацію, оскільки розмір кодів команд становить 16 біт, рис. 1.116.

Адресація пам’яті програм здійснюється за допомогою лічильника команд (PC – Program Counter), рис. 1.113, розміром 9 або 10 розрядів, що визначається обсягом пам'яті.

Вектор скидання розташовано за адресою 0x000 пам'яті програм. Після ініціалізації (reset) мікроконтролера з даної адреси розпочинається виконання програми. Зазвичай за адресою 0x000 розміщується команда відносного переходу (rjmp) до частини ініціалізації програми.

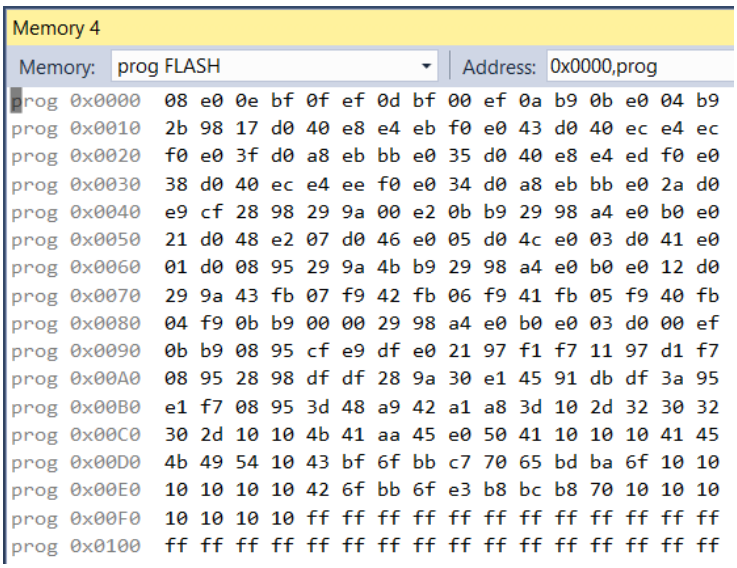


Рис. 1.116. Вікно для відображення вмісту FLASH-пам’яті програм мікроконтролера в середовищі Microchip Studio

```
.INCLUDE "m8def.inc"
       rjmp Lreset       ;вектор скидання
                         ;таблиця векторів переривань:
       rjmp key2_down    ;INT0 External Interrupt Request 0
       rjmp key1_up      ;INT1 External Interrupt Request 1
       reti              ;TIMER2 COMP Timer/Counter2 Compare Match
       reti              ;TIMER2 OVF Timer/Counter2 Overflow
       reti              ;TIMER1 CAPT Timer/Counter1 Capture Event
       reti              ;TIMER1 COMPA Timer/Counter1 Compare Match A
       reti              ;TIMER1 COMPB Timer/Counter1 Compare Match B
       reti              ;TIMER1 OVF Timer/Counter1 Overflow
       rjmp T0_OVF       ;TIMER0 OVF Timer/Counter0 Overflow
       reti              ;SPI, STC Serial Transfer Complete
       reti              ;USART, RXC USART, Rx Complete
       reti              ;USART, UDRE USART Data Register Empty
       reti              ;USART, TXC USART, Tx Complete
       reti              ;ADC ADC Conversion Complete
       reti              ;EE_RDY EEPROM Ready
       reti              ;ANA_COMP Analog Comparator
       reti              ;TWI Two-wire Serial Interface
       reti              ;SPM_RDY Store Program Memory Ready
     ;Основна програма:
Lreset:
                          ;ініціалізація стека
       ldi r16,LOW(RAMEND)
       out SPL,r16
       ldi r16,HIGH(RAMEND)
       out SPH,r16

       sei         ;дозвід переривань
      ...................
```

Рис. 1.117. Фрагмент програми для мікроконтролера на мові асемблера, що ілюструє вектор скидання та таблицю векторів переривань

Починаючи з адреси 0x001 розміщується таблиця векторів переривань. Її

розмір, тобто кількість векторів переривань, визначається моделлю мікроконтролера. В разі виникнення переривання до стеку зберігається значення лічильника команд, після чого виконується команда, що розміщується за адресою відповідного вектора. Це команда rjmp відносного переходу до підпрограми оброблення відповідного переривання.

Наприклад, у фрагменті програми, що наведений на рис. 1.117, першим рядком є директива .INCLUDE "m8def.inc" включення файлу з визначенням символічних імен регістрів введення-виведення для мікроконтролера ATMega8. Однак, директива не є командою, вона є тільки інструкцією компілятору. Тому за адресою 0x000 пам'яті програм виявиться вектор скидання rjmp Lreset. Після цього вектору іде таблиця векторів переривань. Вона складається з команд переходу до підпрограм оброблення деяких переривань. Звичайно, що не всі переривання передбачається обробляти. Тому на місці такого вектора строїть команда reti, яка, в разі виникнення переривання, що не обробляється, повертає управління до основної програми. Якщо оброблення переривань в програмі заборонено, що основна програма може починатися з адреси 0x000.

1.6.5. Переривання. Розміщення таблиці векторів переривань

Переривання припиняє нормальний хід програми для виконання пріоритетного завдання, визначеного внутрішньою або зовнішньою подією мікроконтролера, рис. 1.118. Джерелами переривань є різні внутрішні або зовнішні події. Зокрема: зміна рівня напруги на виводі, переповнення таймера/лічильника, завершення перетворення АЦП тощо.

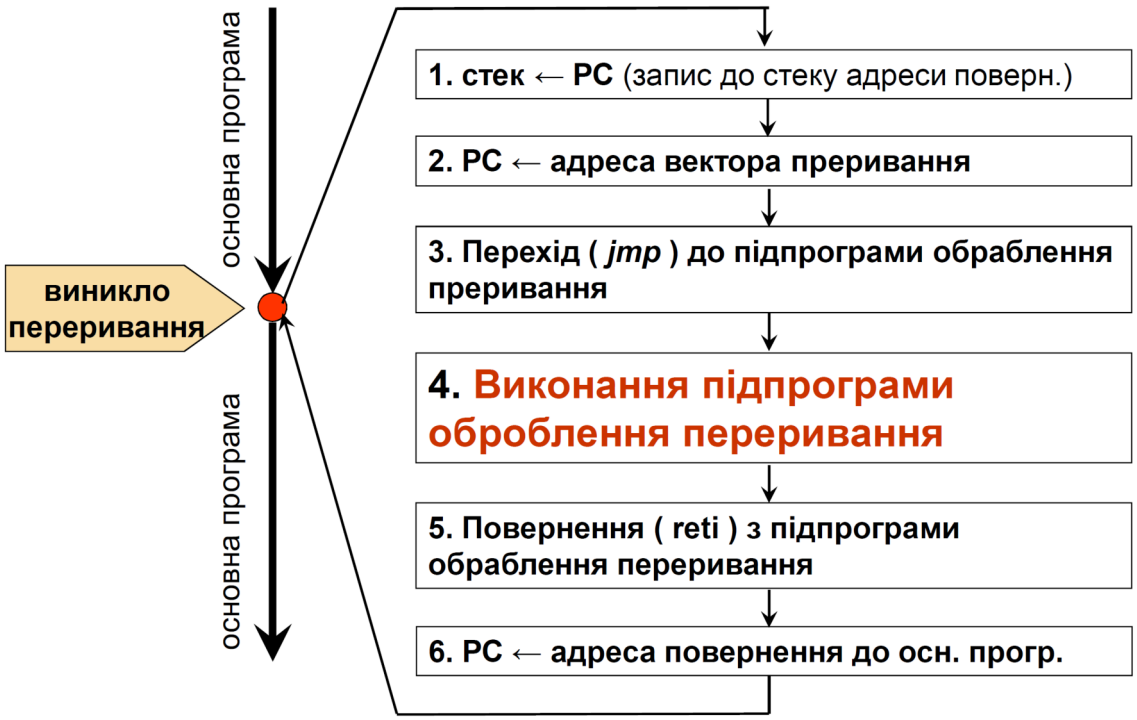


Рис. 1.118. Механізм оброблення переривання

Припустимо, що контролер виконує основну програму і в цей час

трапилося переривання. Тоді значення лічильника команд зберігається до стеку. До лічильника команд завантажується адреса вектора відповідного переривання. За цією адресою має знаходитися команда rjmp відносного переходу до підпрограми оброблення переривання. Підпрограма оброблення переривання має завершуватися командою RETI, що забезпечує повернення в основну програму і відновлення адреси повернення, тобто завантаження адреси повернення зі стеку до лічильника команд, рис. 1.119.

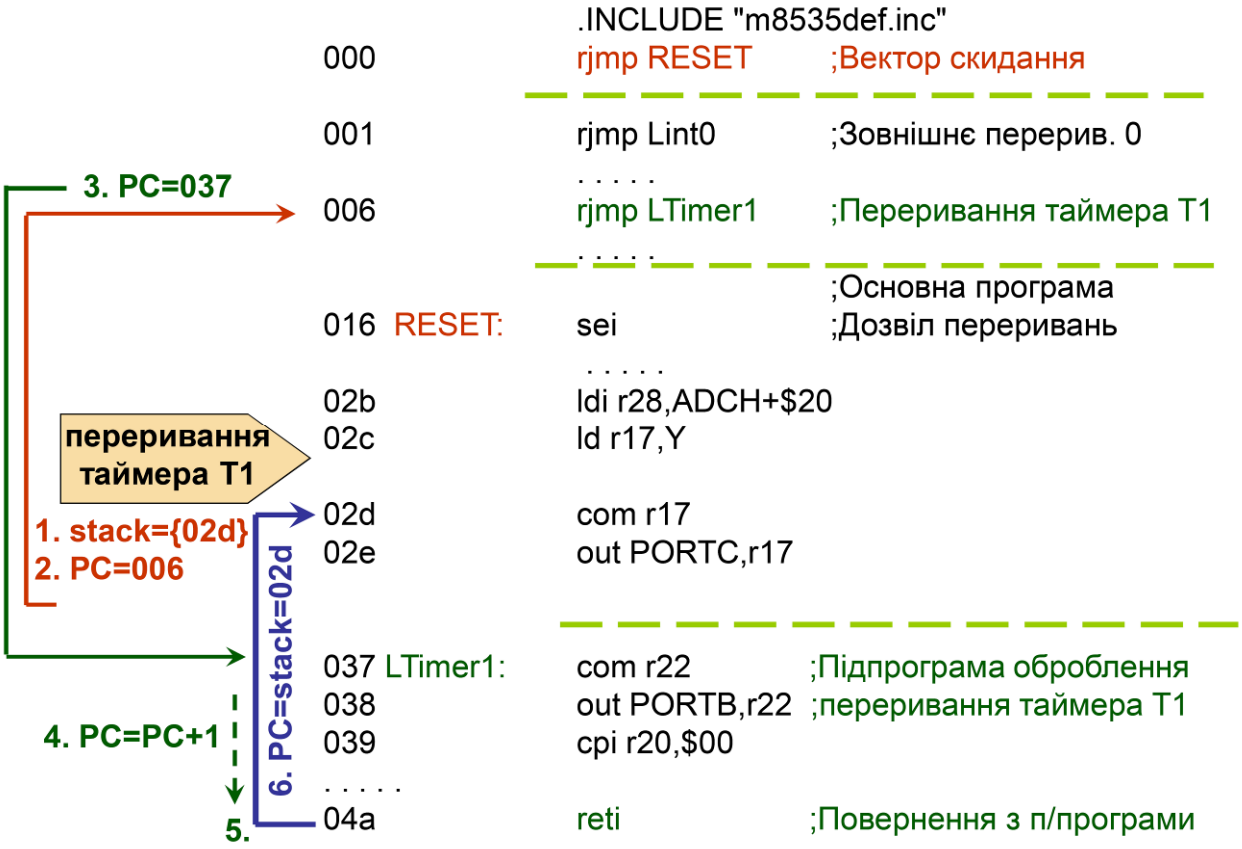


Рис. 1.119. Програмне оброблення переривання

1.6.6. Питання для самоперевірки

1. Дайте характеристику архітектурі ядра 8-розрядного мікроконтролера AVR.
2. Скільки регістрів загального призначення доступні для користувача? Які їх назви? Для чого вони використовуються?
3. Для чого використовуються регістри введення-виведення?
4. Які типові периферійні пристрої входять до складу 8-розрядних мікроконтролерів AVR?
5. Для чого використовується EEPROM?
6. Для чого призначено регістр SREG?
7. Розкрийте структуру статичного ОЗП у складі мікроконтролера.
8. Для чого використовується стек?
9. Який принцип покладено в основу роботи стека?
10. Де розташовано стек в разі програмної реалізації?
11. Для чого використовується регістр-вказівник стека?

12. Які команди використовуються для роботи зі стеком?

13. За допомогою чого здійснюється адресація пам’яті програм?

14. Опишіть призначення та розташування вектора скидання та таблиці векторів переривань.

15. Для чого використовується механізм переривань?

16. Опишіть порядок оброблення переривання.

1.7. Дискретні порти введення-виведення

Характеристика портів. Програмна модель портів введення-виведення. Команди для роботи з портами. Приклад налаштування портів. Схеми узгодження сигналів для підключення типових пристроїв до мікропроцесора: підсилювачі на транзисторах, підключення енкодерів, герконів, кнопок та перемикачів, механічних контактів, фотоприймачів на оптопарах.

1.7.1. Характеристика портів. Програмна модель портів введення-виведення

Підсистема введення/виведення мікроконтролера включає цифрові порти введення/виведення. Майже всі фізичні виводи мікросхеми є виводами портів, окрім виводів живлення та (в деяких моделях) тактування роботи. Оскільки мікроконтролери AVR мають 8-розрядну архітектуру, то порти введення/виведення (input/output ports, IO ports) мають 8 виводів. Порти позначаються великими латинськими літерами: A, B, C, E, …. В різних моделях мікроконтролерів не обов’язково є порт A, позначення портів може починатися з іншої літери. Наприклад, розглянемо цифрові IO порти мікроконтролера ATMega328p, рис. 1.120. Цей мікроконтролер є основною плати Arduino UNO.

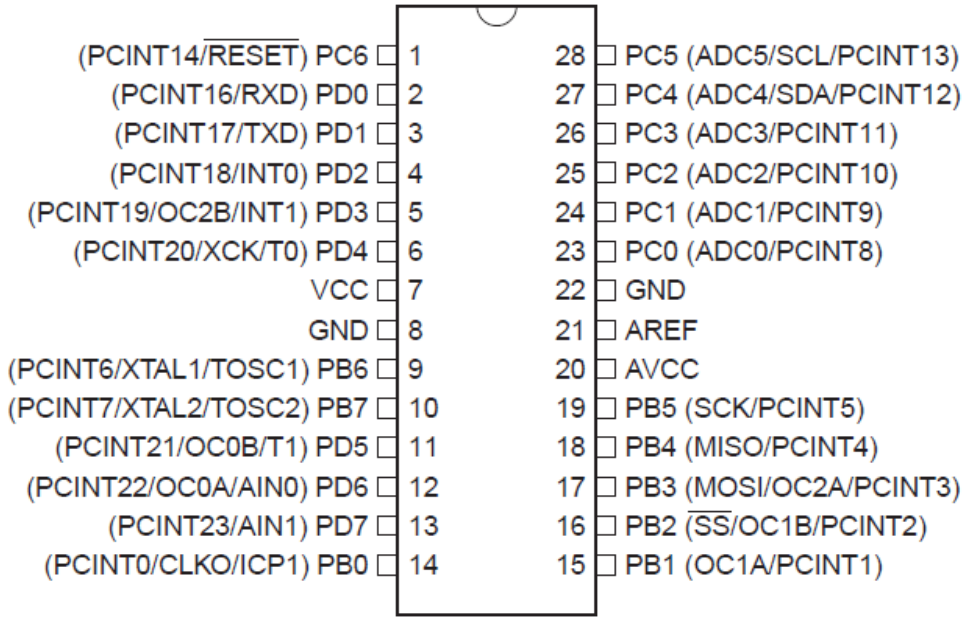


Рис. 1.120. Розташування фізичних виводів мікроконтролера ATMega328p

Мікроконтролер ATMega328p має 3 порти, що позначені літерами B, C, D. Виводи портів включають P (від англ. Port), літеру, яка присвоєна порту, та номер вивода порта (від 0 до 7). Наприклад, позначення PB1 означає, що це вивід цифрового порта B під номером 1. Кожен з виводів порту може бути сконфігурований незалежно від стану та режиму роботи інших виводів порта. Кожен вивід може бути запрограмований для функціонування як вхід (input), або як вихід (output). Якщо вивід запрограмований як вхід, то існує можливість використовувати внутрішні підтягуючі резистори опором від 35 до 120 кОм. В портах реалізовано алгоритм «читання – модифікація – запис». Завдяки цьому досягається незалежність роботи різних виводів портів з використанням команд SBI та CBI.

Звернення до портів здійснюється через три регістри введення/виведення, табл. 1.16. В таблиці позначено: x – літерне значення порта (A, B, C, D, …); n – номер розряду порта (0–7).

Таблиця 1.16

Регістри цифрових портів введення/виведення
8-розрядних мікроконтролерів AVR

№	Назва регістра	Позначення регістра		Позначення розрядів	
		У загальному вигляді	Приклад	У загальному вигляді	Приклад
1	Регістр напряму передачі даних	DDRx	DDRB	DDxn	DDB1
2	Регістр даних	PORTx	PORTB	PORTxn або Pxn	PORTB1 або PB1
3	Регістр стану виводів порту	PINx	PINB	PINxn	PINB1

Розглянемо програмну модель порта введення/виведення, тобто призначення регістрів.

1. *Регістр напряму передачі даних DDRx* визначає напрям передачі даних через контакт введення/виводу. Розряд регістру може приймати наступні значення:

якщо DDxn=1, то n-й контакт порта x є виходом (output);

якщо DDxn=0, то n-й контакт порту x є входом (input).

2. Призначення *регістра даних PORTx* залежить від режиму функціонування виводу мікроконтролера:

а) якщо вивід конфігурований як вихід (DDxn =1), то за допомогою розряду Pxn регістра PORTx здійснюється *управління виводом:*

Pxn=1 – на відповідному виводі встановлюється напруга високого рівня;

Pxn=0 – на виводі встановлюється напруга низького рівня;

б) якщо вивід функціонує як вхід (DDxn =0), розряд Pxn регістра PORTx визначає *стан внутрішнього підтягуючого резистора* для цього виводу:

Pxn=1 – підтягуючий резистор підключається між виводом

мікроконтролера і шиною живлення Vcc;

Pxn=0 – підтягуючий резистор відключений.

3. *Регістр стану виводів порту PINx* контролю стану вивода мікроконтролера, причому незалежно від режиму роботи (output, input). Цей регістр доступний тільки для читання.

Графічно зв'язок регістрів введення/виведення цифрового порта B та фізичних виводів мікроконтролера подає схема на рис. 1.121.

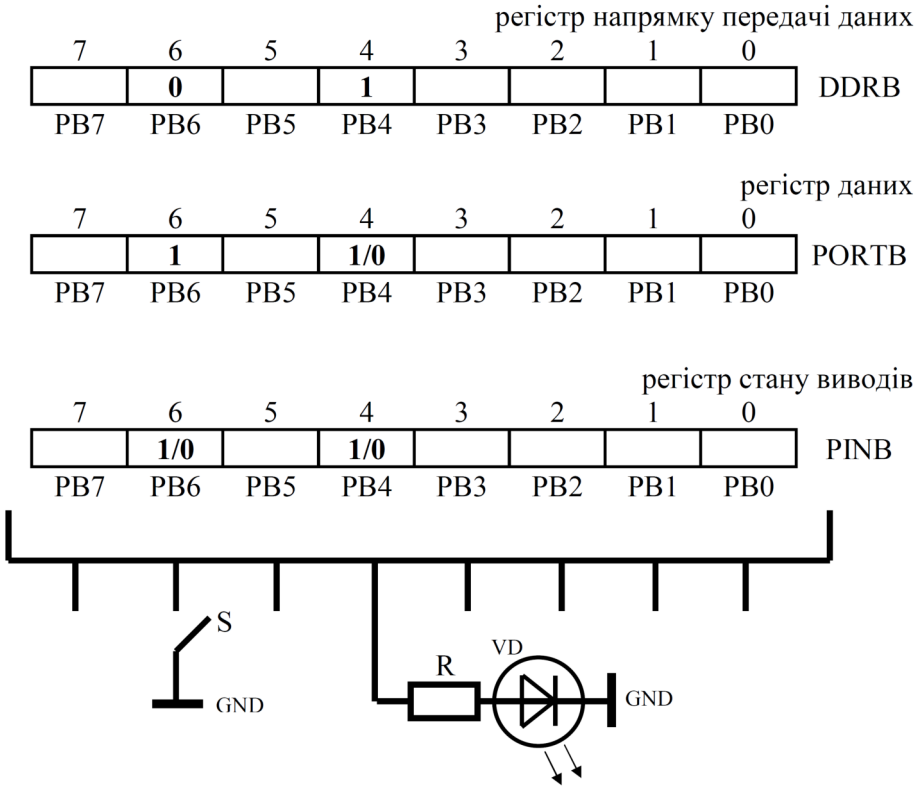


Рис. 1.121. Графічне представлення зв'язку програмної моделі цифрового порту B з фізичними виводами мікроконтролера

До вивода PB4 підключено світлодіод VD через струмообмежувальний резистор R. Тому PB4 має бути сконфігурований як вихід (output). Для цього у відповідний розряд регістра напрямку передачі даних DDRB заноситься лог. «1». Керування світлодіодом здійснюється за допомогою розряда PORTB.PB4: світлодіод включиться, якщо у цей розряд записати лог. «1», та відключається при запису лог. «0». Відповідний розряд регістра стану PINB повторює стан вивода.

Між фізичним виводом мікроконтролера PB6 та землею (GND, загальний вивід джерела живлення) приєднано кнопку S. Цей вивід має бути сконфігурований як вхід (input), тому до розряду DDRB.PB6 заноситься лог. «0». Оскільки зовнішній підтягуючий резистор відсутній, то виникає необхідність застосовувати вбудований. Для цього PORTB.PB6=1. Зчитувати стан кнопки можливо шляхом опитування розряда PINB.PB6.

Загальні рекомендації щодо конфігурування виводів цифрових портів наведені у табл. 1.16.

Таблиця 1.16

Рекомендації щодо конфігурування цифрових портів мікроконтролера

До виводу n порта x підключено	Режим функціонування виводу n порта x	Дії			
		Конфігурування порта (задавання режиму роботи)	Управління виводом (на виводі встановлюється напруга)	Стан внутрішнього підтягуючого резистора	Зчитувати стан виводу
		Регістр DDRx	Регістра PORTx		Регістр PINx
Засіб введення інформації: кнопка, дискретний давач тощо	ВХІД (input)	DDxn=0	неможливо	Pxn=1 – підключений Pxn=0 – відключений	зчитувати PINxn
Виконавчий механізм або індикатор: реле, світлодіод тощо	ВИХІД (output)	DDxn=1	Pxn=1 – високого рівня Pxn=0 – низького рівня	не вимагається	

Для плат Adruino використовуються спрощені назви цифрових портів: виводи нумеруються цифрами 0, 1, 2, …. Наприклад, для плати Arduino UNO, рис. 1.122, цифрові порти пронумеровані цифрами від 0 до 13, що відповідає виводам PD0,…,PD7, PB0,…,PB5 портів мікроконтролера. Виводи, що використовуються АЦП у складі мікроконтролера, позначені на платі A0,…,A5. Ці виводи також можуть використовуватися як виводи PC0,…,PC5 порта C мікроконтролера. Вивід PC6 також використовується для подавання сигналу скидання.

1.7.2. Команди для роботи з портами

Керування окремими розрядами регістрів введення/виведення можливо здійснювати за допомогою наступних команд мови асемблера:

`sbi A, b` – встановити (в лог. «1») розряд b регістра введення/виведення A, тобто виконати операцію A.b=1 (мнемокод sbi розшифровується як set bit input/output);

`cbi A, b` – скинути (в лог. «0») розряд b регістра введення/виведення A, тобто виконати операцію A.b=0 (мнемокод cbi розшифровується як clear bit input/output).

Наприклад:

```
sbi DDRB, PB4    ;встановити розряд 4 регістра DDRB
cbi PORTB, PB4   ;скинути розряд 4 регістра PORTB
```

Також для встановлення значення розряда регістра введення/виведення можна використати команди ldi та out. Наприклад:

```
ldi r16, (1<<PC0);встановити розряд PC0 регіста r16
```

```
out PORTC, r16   ;переслати вміст r16 до PORTC
```

Символ << повертає ліве вираження зсунуте вліво на число біт, вказане справа.

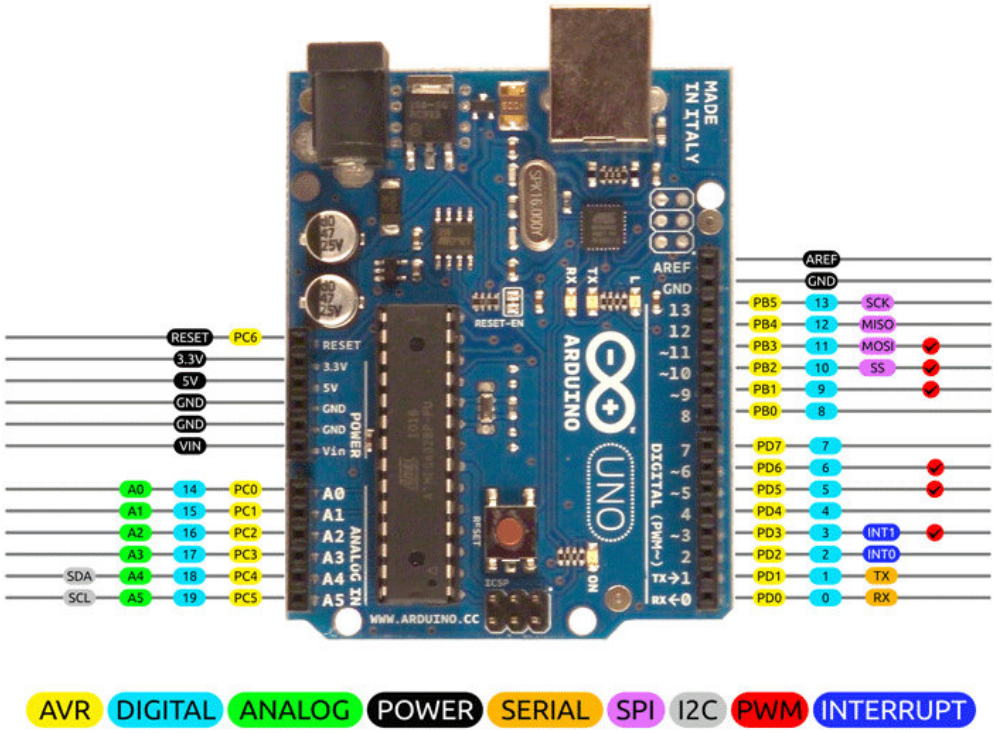


Рис. 1.122. Відповідність між виводами плати Arduino UNO та виводами мікроконтролера ATMega328p

Також можна одночасно встановлювати декілька розрядів. Для цього використовується операція АБО (символ " | " або " + "):

```
ldi r16, (1<<PC0)|(1<<PC5);встановити розряди PC0 та PC5 регіста r16
out PORTC, r16   ;переслати вміст r16 до PORTC
```

Крім того, існує можливість безпосереднього встановлення значення регістра введення/виведення за допомогою команди out. Наприклад:

```
out DDRB, 0b01100101
```

Перевірку стану окремих розрядів регістрів введення/виведення можливо здійснювати за допомогою команд:

`sbic A, b` – перевіряє стан розряду b регістра введення/виведення A. Якщо розряд скинутий, команда, що йде за "sbic A, b" пропускається.

`sbis A, b` – перевіряє стан розряду b регістра введення/виведення A. Якщо розряд встановлений, команда, що йде за "sbis A, b" пропускається.

У разі натиснутого стану кнопки на схемі (рис. 1.121) вивід мікроконтролера буде притягнуто до GND, тобто мати нульовий потенціал відносно схемої землі. На виводі мікроконтролера буде сигнал низького рівня, у відповідному розряді регістра PIN буде лог. «0». Якщо кнопка буде відпущена, то завдяки внутрішньому підтягуючому резистору фізичний вивід буде

притягнуто до шини Vcc, що відповідає сигналу високого рівня, і у відповідному розряді регістра PIN буде лог. «1».

Наприклад, для зчитування стану кнопки S, рис. 1.121, можуть використовуватися наступні команди:

```
    sbic PINB, PB6
    rjmp B_UP   ;якщо розряд встановлено (кнопка не
                ;натиснута), йти на мітку B_UP
    . . .
    <код, якщо кнопка натиснута>
    . . .
    rjmp L1  ;кінець фрагмента кода для натисненої кнопки
B_UP:
    . . .
    <код, якщо кнопка не натиснута>
    . . .
L1:
```

Також для зчитування стану кнопки S можна використовувати наступний код:

```
    sbis PINB, PB6
    rjmp B_DN     ;розряд не встановлений (кнопка
                  ;натиснута)йти на B_DN
    . . .
    <код, якщо кнопка не натиснута>
    . . .
    rjmp L2
B_DN:
    . . .
    <код, якщо кнопка натиснута>
    . . .
L2:
```

На мові C for Arduino для налаштування режиму роботи вивода порта використовується команда pinMode(pin, mode), де pin – номер вивода плати; mode – змінна, що визначає режим вивода, може приймати значення: INPUT – вхід без застосування внутрішнього підтягуючого резистора; INPUT_PULLUP – вхід із застосуванням такого резистора; OUTPUT – вихід. Налаштування режиму роботи порта здійснюється у підпрограмі setup.

1.7.3. Приклад налаштування портів

Розглянемо принципову схему мікропроцесорного пристрою управління електродвигуном M (рис. 1.123), що включає:

– мікроконтролер ATmega8;

– однофазний електродвигун, підключений до мережі 220 В через нормально розімкнений контакт K1.1 реле K1;

– підсилювач на транзисторі VT1, що керує котушкою реле K1, підключений до виводу PB6 мікроконтролера;

– світлодіод VD1 «робота», підключений до виводу PC5;

– три кнопки управління режимом роботи пристрою S1–S3, причому S1 підключена до PD2, S2 – до PD3, S3 – до PC0.

Параметри портів мікроконтролера зведені в табл. 1.17.

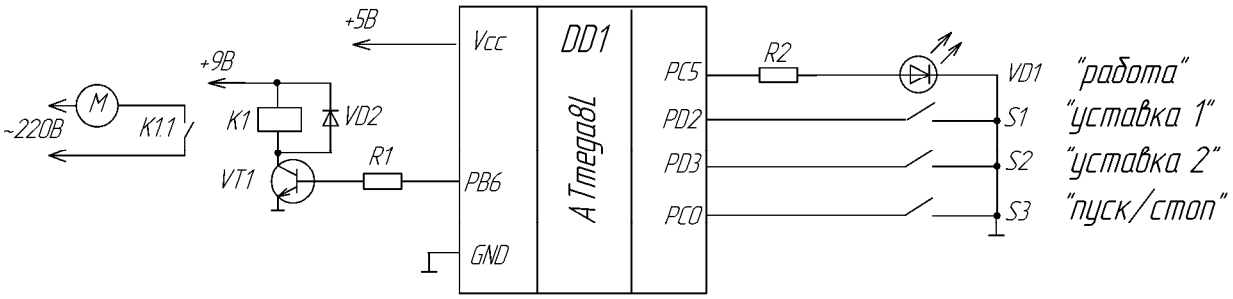


Рис. 1.123. Принципова схема мікропроцесорного пристрою управління електродвигуном

Таблиця 1.17

Конфігурування портів мікроконтролера для пристрою управління електродвигуном

Позначення виводу	Порт	№ виводу порту	До виводу порта підключено	Режим функ-ціону-вання виводу	Конфігу-рація порта	Стан внутрішньо го підтягуючог о резистора
PB6	B	6	реле K1 (вкл/відкл двигуна)	вихід	**DDB6=1**	-
PC0	C	0	кнопка S3 ("пуск/стоп")	вхід	DDC0=0	PC0=1
PC5		5	світлодіод VD1 ("робота")	вихід	**DDC5=1**	-
PD2	D	2	кнопка S1 ("уставка 1")	вхід	DDD2=0	PD2=1
PD3		3	кнопка S2 ("уставка 2")	вхід	DDD3=0	PD3=1

Фрагмент програми конфігурації портів:

```
    .INCLUDE "m8def.inc";включити файл з символічними іменами
    ldi r16, high(RAMEND) ;ініціалізація показника стека
    out SPH, r16
    ldi r16, low(RAMEND)
    out SPL, r16

    ldi r16,(1<<DDB6)     ;конфігурація порту B
```

```
out DDRB, r16

ldi r16,(1<<DDC5)      ;конфігурація порта C
out DDRC, r16

ldi r16,(1<<PC0)      ;вхід з підтягуючими резисторами
out PORTC, r16

ldi r16(1<<PD3)       ;вхід з підтягуючими резисторами
out PORTD, r16
```

Для управління світлодіодом використовуються наступні команди:
– ввімкнути світлодіод VD1:

```
sbi PORTC, PC5
```

– вимкнути світлодіод VD1:

```
cbi PORTC, PC5
```

Аналогічні команди використовуються для управління двигуном:
– ввімкнути двигун М за допомогою включення реле:

```
sbi PORTB, PB6
```

– вимикнути двигун:

```
cbi PORTB, PB6
```

1.7.4. Схеми узгодження сигналів для підключення типових пристроїв до мікропроцесора

1.7.4.1. Підсилювачі на транзисторах

Як правило, сигнали, що надходять до мікроконтролера від давачів, мають малу амплітуду. Виникає необхідність їх підсилення. Цю функцію виконують вхідні підсилювачі напруги. Розглянемо деякі типові схеми вхідних транзисторних підсилювачів, що використовуються у мікропроцесорній техніці.

На рис. 1.124, *а* наведена схема, розрахована на максимальну частоту вхідного сигналу до 30 МГц. Схема побудована на транзисторів VT1 типу КТ368 і забезпечує чутливість 200 мК. Діоди VD1 та VD2, включені зустрічно-паралельно, забезпечують обмеження амплітуди вхідного сигналу. Кола R1, C1 та R3, L1 використовуються для корекції фронтів сигналу. Застосування змінного резистора R1 в схемі рис. 1.124, *б*, забезпечує плавне регулювання чутливості. На рис. 1.124, *в*, наведено схему широкополосного підсилювача, що розрахований на діапазон вхідного сигналу від 30 Гц до 100 МГц. Вхідний опір – не менше 1 МОм на частоті 1 кГц. Чутливість схеми становить 75 мВ. В схемі на рис. 1.124, *г*, транзистори VT2 та VT3 ввімкнені за схемою струмового дзеркала. Діоди VD1 та VD2 застосовані для обмеження вхідного сигналу за амплітудою зверху та знизу. Функція діода VD3 – відсікання шумів та завад.

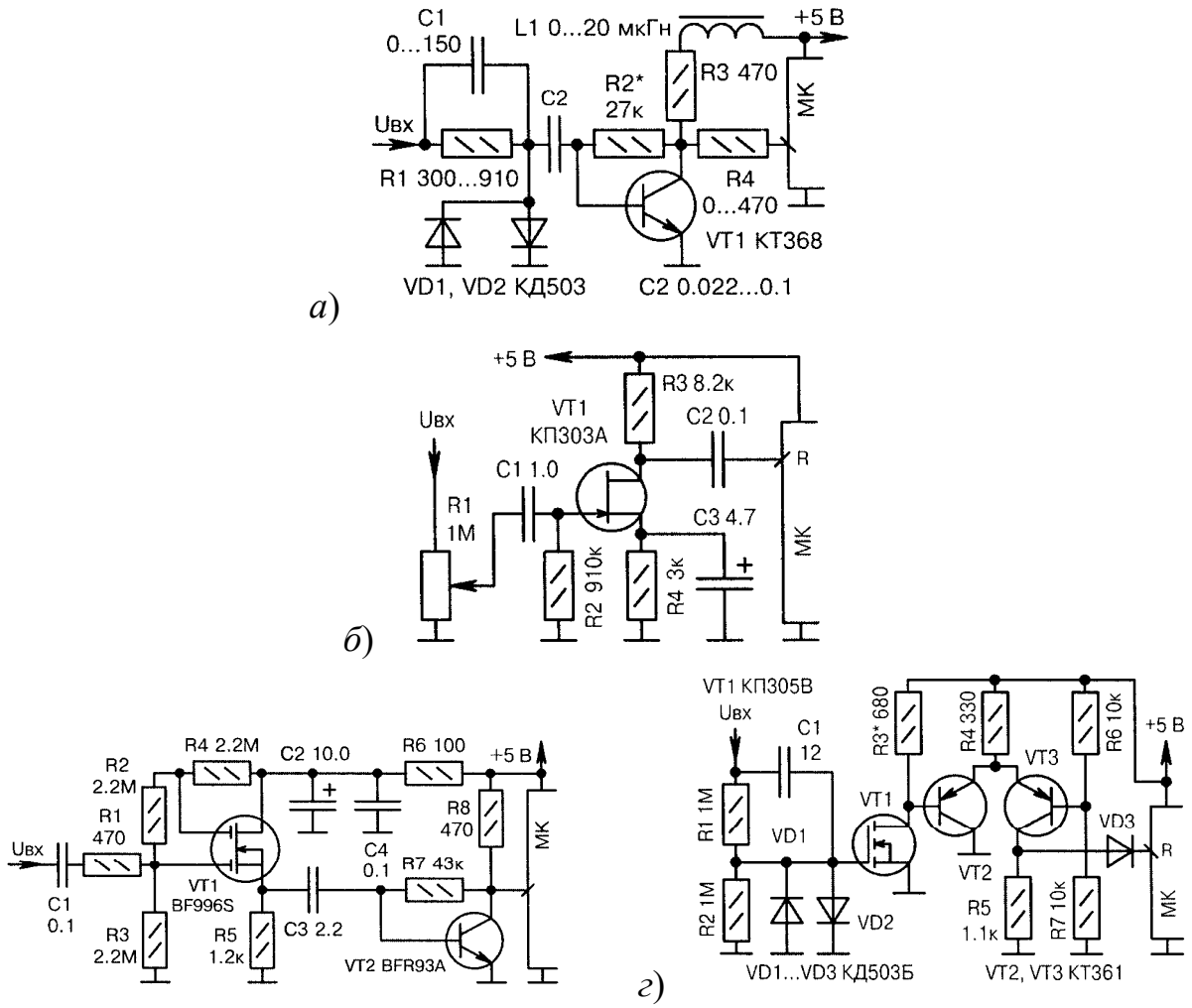


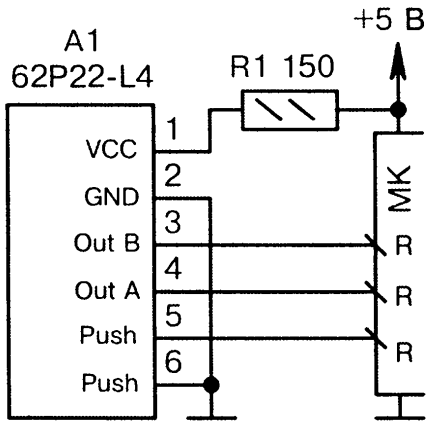
Рис. 1.124. Принципові схеми підсилювачів вхідних сигналів на транзисторах

1.7.4.2. Підключення енкодерів

Енкодер являє собою механічний або оптико-механічний давач, який використовується для визначення поточного положення або частоти обертання вала. Конструктивно енкодер являє собою диск, що закріплений на валу, з отворами (прорізями, виступами). Такі елементи або перекривають світловий потік при обертанні вала, або замикають механічні контакти. На виході енкодера формуються прямокутні сигнали. Під час перемикання вихідного елемента можуть спостерігатися високочастотні завади. Мікроконтролер визначає напрямок обертання енкодера та швидкість обертання програмно.

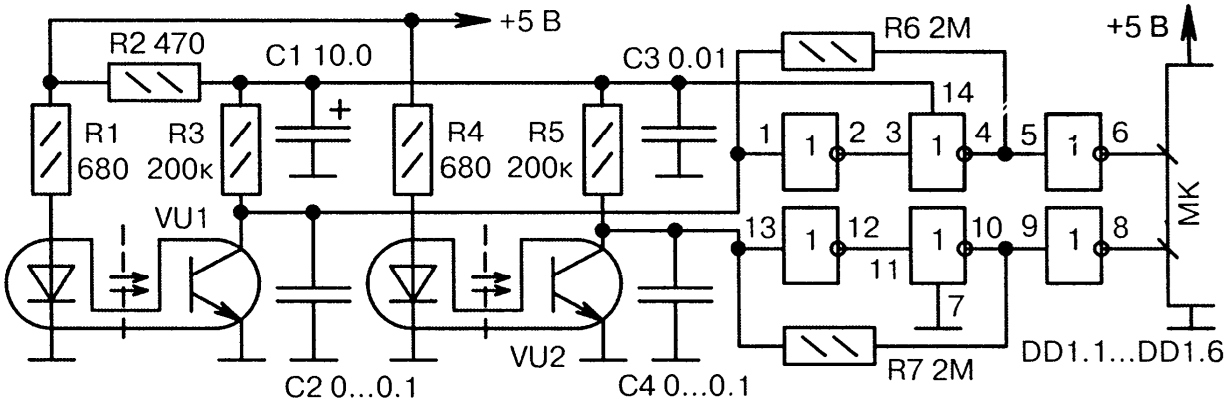
На рис. 1.125, *а*, наведено загальний вигляд оптичного енкодера 62P22–L4. Типову схему його підключення до мікроконтролера ілюструє рис. 1.125, *б*. Контролер аналізує імпульсні сигнали енкодера, завдяки чому визначає напрямок обертання та частоту. Оптичний енкодер B38S-6G5-26C360B-2M зображено на рис. 1.125, *в*. Оптична система розташована всередині корпусу, кількість імпульсів за оберт становить 1440, максимальна механічна швидкість вала 5000 об/хв.

а)



б)

в)



г)

Рис. 1.125. Схеми підключення енкодерів до мікроконтролера

1.7.4.3. Підключення герконів

Геркон (скорочення від «герметичний контакт», англійською називається reed switch) – механічний контакт, розміщений у герметичній колбі, рис. 1.126, *а*. Замикається контакт під впливом зовнішнього магнітного поля. Контакт не має фіксації, тобто розмикається при зникненні магнітного поля. Геркони випускають замикальні та перемикальні. До переваг герконів відносять: невелика потужність управління, низький перехідний опір між замкненими контактами, високий опір ізоляції між розімкненими контактами, гальванічна розв'язка за рахунок скляного корпусу, механічна стійкість. Розрізняють низьковольтні (до 1 кВ) та високовольтні геркони. Геркони використовуються у складі кінцевих вимикачів, кнопок керування, для передавання дискретних сигналів управління у високовольтних електроустановках.

Підключення геркона до мікроконтролера може здійснюватися відповідно до схеми, що наведена на рис. 1.126, *б*. Коло C1, R1 усуває помилкові спрацювання геркона SF1 та брязкіт його контактів. Послідовне ввімкнення декількох герконів (схема реалізація логічної функції AND) здійснюється відповідно до рис. 1.127, *в*. В разі необхідності з'єднати декілька герконів паралельно (реалізувати функцію OR), застосовується схема, наведена на рис. 1.126, *г*.

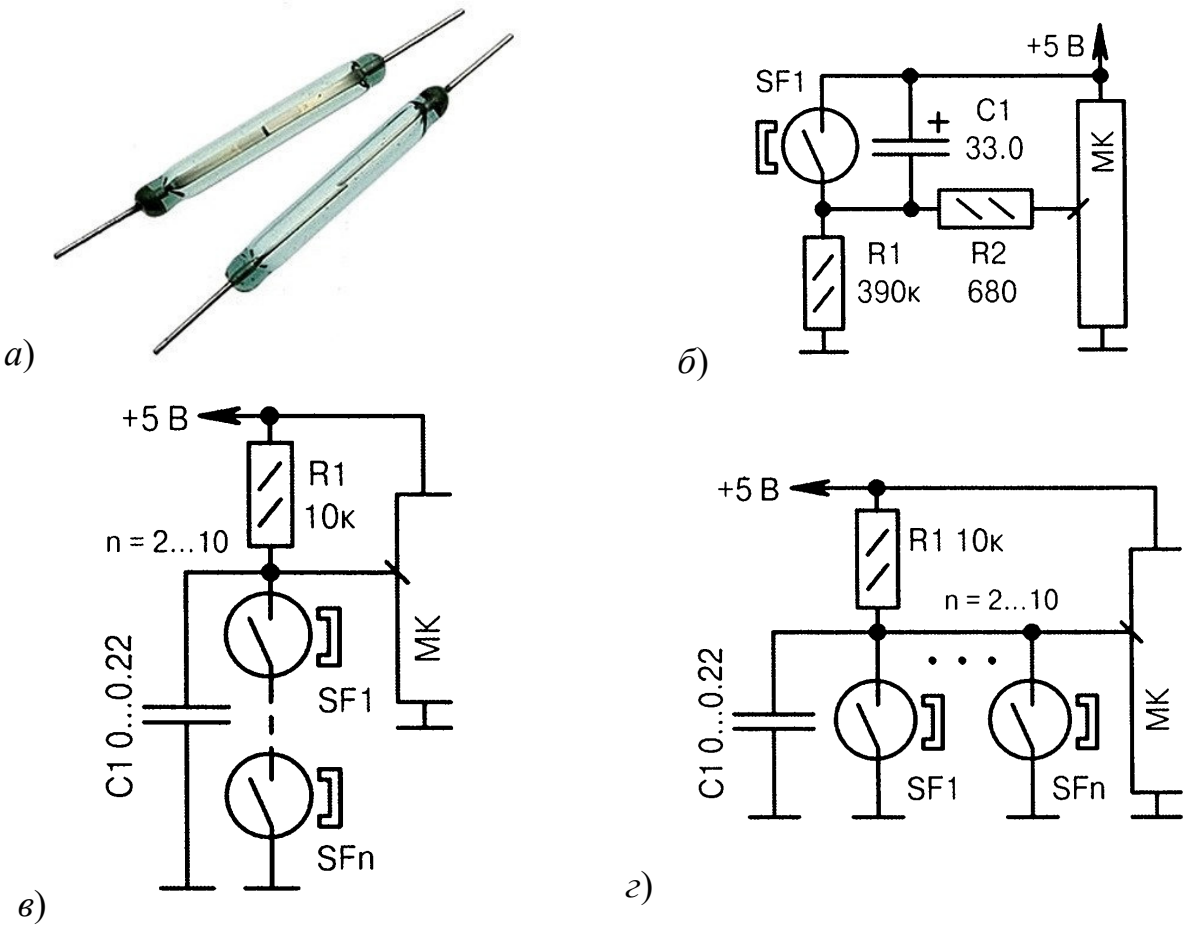


Рис. 1.126. Підключення герконів до мікроконтролера

1.7.4.4. Підключення механічних кнопок та перемикачів

Механічні кнопки та перемикачі надзвичайно широко використовуються у складі мікропроцесорних пристроїв для організації інтерфейса користувача. Надійне введенні інформації визначається, великою мірою, найбільшою робочою напругою та робочим струмом. Якщо робоча напруга більшості кнопок є більшою від напруги живлення мікроконтролера, то найменший робочий струм кнопки, при якому гарантується надійне спрацювання, може перевищувати вхідний струм вивода мікроконтролера. Причина полягає у суттєвій величині внутрішнього підтягуючого резистора, опір якого складає десятки кОм. Якщо для надійного спрацювання кнопки необхідно, щоб струм через її контакт становив не менше 1 мА, то виникає необхідність використовувати зовнішній підтягуючий резистор опором не більше 4,7 кОм при напрузі живлення 5 В.

Крім того, проблема полягає у наявності брязкоту контактів при замикання та розмиканні кнопки, рис. 1.127. Брязкіт супроводжує комутацію всіх типів механічних контактів, що використовуються у кнопках та давачів на їх основі. Тривалість випадкових переривань електричного контакту може тривати до 40 мс. Брязкіт може призвести до неправильного розпізнавання стану кнопки. Для усунення брязкоту використовуються програмні та апаратні засоби. Програмний підхід передбачає організацію програмних затримок (тривалістю до 50 мс) при виявленні комутації з подальшим багатократним опитування стану кнопки.

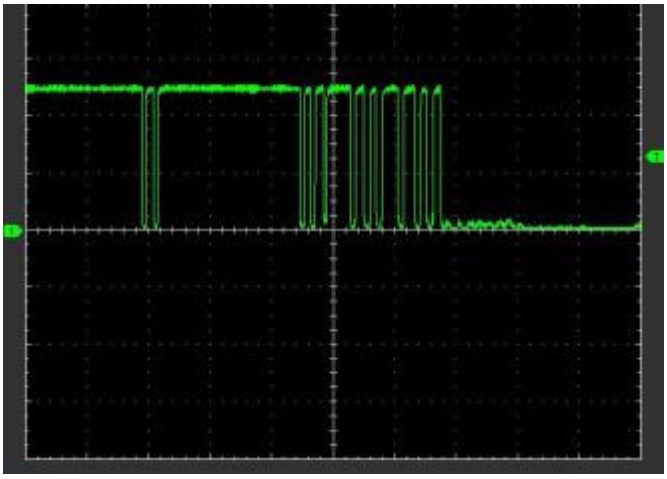


Рис. 1.127. Осцилограма напруги на механічній кнопці при натисканні, що ілюструє брязкіт контактів

Апаратні заходи щодо зниження брязкоту контактів передбачають використання аналогових фільтрів, найчастіше – на основі конденсаторів. Зокрема, для підключення кнопки до цифрового виводу мікроконтролера може бути використана схема, наведена на рис. 1.128, *а*, яка передбачає використання внутрішнього підтягуючого резистора мікроконтролера. При натисканні кнопки SB1 струм від джерела +5 В протікає через вказаних підтягуючий резистор, зовнішній резистор R1. Конденсатор C1 знижує брязкіт контактів.

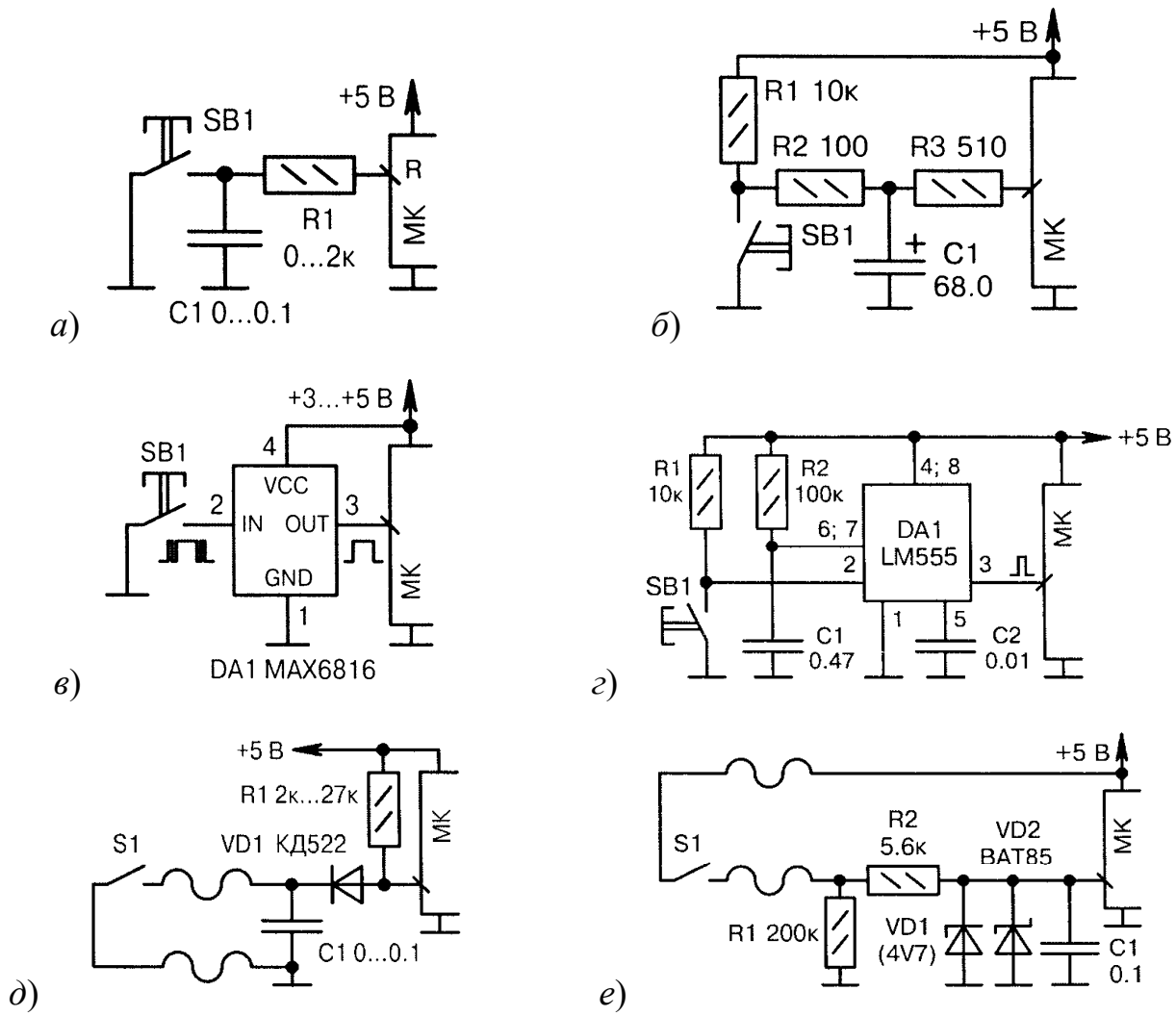


Рис. 1.128. Схеми підключення механічних кнопок та перемикачів до мікроконтролера

В схемі на рис. 1.128, *б*, суттєва ємність конденсатора С1 (68 мкФ) дозволяє зуттєво знизити брязкін контакта та робить схему нечутливою до короткочасних помилкових натискань кнопки. Резистор R2 призначено для обмеження струму розрядку конденсатора через SB1, а резистор R3 – через внутрішній діод мікроконтролера.

Надійне апаратне придушення брязкону контактів кнопки SB1 забезпечує схема на рис. 1.128, *в*, що передбачає використання одноканального апаратного подавлювача брязкоту MAX6816. Така мікросхема приймає викривлений брязкотом сигнал та з деякою затримкою видає очищений цифровий сигнал, рис. 1.129. Також випускають дво- та воськи канальні подавлювачі: MAX6817, MAX6818.

Також брязкіт механічного контакту може бути усунутий шляхом використання одновібратора на аналоговому таймері LM555, рис. 1.128, *г*. При цьому тривалість імпульсу на вході мікроконтролера визначається величинами R2, C1.

Для підключення віддалених механічних давачів можуть використовуватися схеми на рис. 1.128, *д*, *е*.

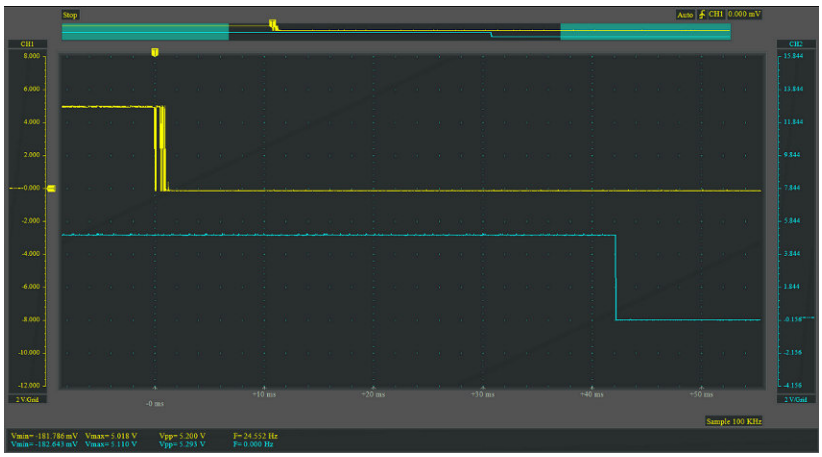
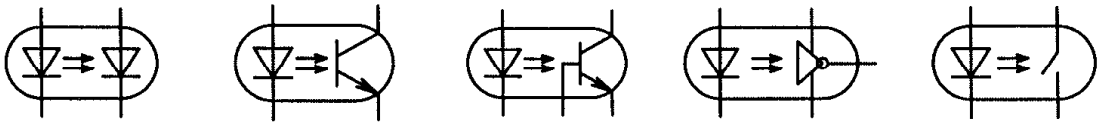


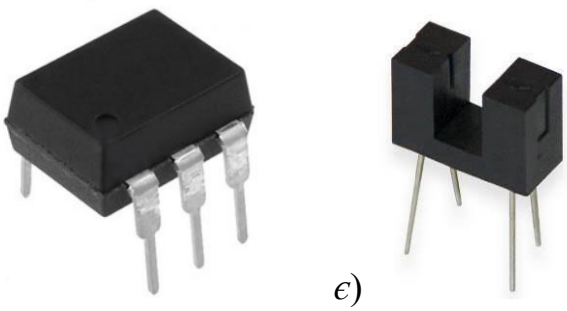
Рис. 1.129. Осцилограма сигналу з механічної кнопки (з брязкотом) та очищеного сигналу на виході мікросхеми MAX6816

1.7.4.5. Підключення фотоприймачів на оптопарах

Оптопари (оптрони) використовуються для гальванічної розв'язки кіл керування від високопотенційних силових кіл. Сигнал передається від світлодіода до фотоприймача світловим потоком. Оптопари широко викостовуються для введення до мікроконтролера сигналів від блок-контактів високовольтних вимикачів або інших механічних давачів, що функціонують у складі високовольтного електрообладнання. В якості приймача світлового потоку можуть використовуватися фотодіод, фототранзистор тощо, рис. 1.130. Оптопари також можуть виконуватися з відкритим оптичним каналом, в такому разі вони використовуються в якості давачів положення, частоти обертання тощо, оскільки світловий потік може перекриватися рухомою частиною механізма.



а) *б*) *в*) *г*) *д*)



е) *є*)

Рис. 1.130. Види оптопар: *а* – з діодним виходом; *б*, *в* – транзисторним виходом; *г* – з цифровим виходом; *д* – з електронним ключем (оптореле); *е* – транзистора оптопара 4N35; *є* – щілинна оптопара (з віткритим каналом) з вихідним фототранзистором GK105A

Фотоприймач транзисторної оптопари може бути приєднаний до мікроконтролера відповідно до типової схеми, що зображена на рис. 1.131, *а*. Резистор R1 обмежує колекторний струм. Конденсатор C1 призначено для фільтрації коротких імпульсів, що можуть призвести до помилкових спрацювань, а також формує більш чіткі фронти сигналів. Використовуючи тригер Шмітта DD1, рис. 1.131, *б*, можна підвищити завадозахищеність та збільшити крутизну фронтів сигналів. Для оптопари з відкритим каналом використовуватися схеми, що наведені на рис. 1.131, *в*, *г*.

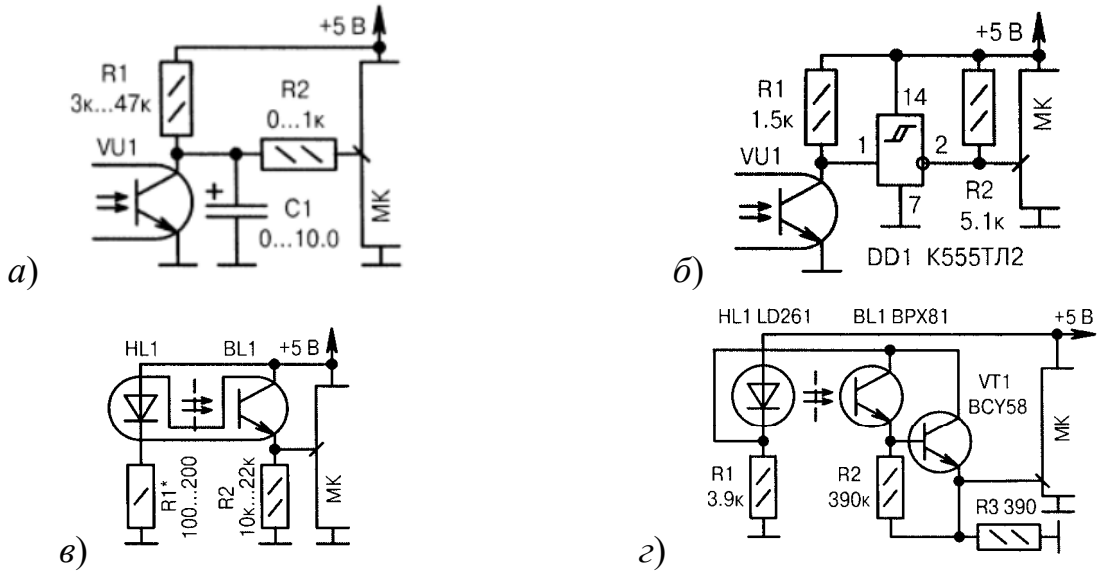


Рис. 1.131. Схеми підключення фотоприймачів транзисторних оптопар (*а*, *б*) та оптопар з відкритим каналом (*в*, *г*)

1.7.5. Питання для самоперевірки

1. Яким чином позначаються цифрові потри введення/виведення мікроконтролера? Виводи портів?

2. Які регістри утворюють програмну модель цифрового порта?

3. Що визначають стани лог. «0» та лог. «1» розряду регістру напрямку передачі даних?

4. Опишіть призначення розрядів регістра даних порта.

5. Яким чином використовується регістр стану виводів порта?

6. Які команди мови асемблера використовуються для роботи з портами?

7. Які команди C for Arduino використовуються для конфігурування та опитування портів?

8. Яке призначення та принцип дії транзисторних підсилювачів вхідних цифрових сигналів?

9. Яким чином можна підключити енкодер до мікроконтролера?

10. Яким чином виконується підключення герконів до мікроконтролера?

11. В чому полягає ефект брязкоту механічних контактів?

12. Які Вам відомі схемні рішення для мінімізації брязкоту механічних контактів?

13. Яким чином підключають фотоприймачі на оптопарах до мікроконтролера?

1.8. Периферія мікроконтролерів

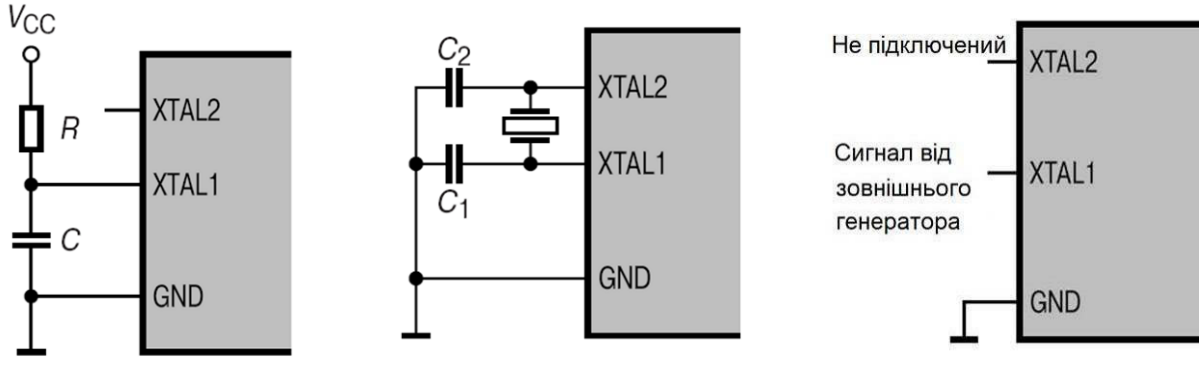
Тактування роботи мікроконтролера. Скидання мікроконтролера. Таймери / лічильники у складі мікроконтролерів. Реалізація цифро-аналогового перетворення. Вартовий таймер. Вбудований аналоговий компаратор. Аналого-цифровий перетворювач у складі мікроконтролера. Програмні засоби для роботи з АЦП.

1.8.1. Тактування роботи мікроконтролера

Тактування передбачає подачу до мікроконтролера прямокутних імпульсів визначеної частоти. Ці імпульси використовуються центральним ядром та периферійними пристроями як спільний сигнал синхронізації, що є обов'язковою умовою для здійснення обчислень та обміну даними. Частота таких імпульсів визначає швидкодію мікроконтролера. Для кожного мікроконтролера в технічній документації вказана найбільша допустима тактова частота. Наприклад, для мікроконтролера ATMega328p при напрузі живлення від 2,7 В до 5,5 В тактова частота не має перевищувати 8 МГц, а при напрузі 4,5–5,5 В – 16 МГц.

Мікроконтролери AVR можуть трактуватися декількома способами. Вибір конкретного джерела тактового сигналу здійснюється за допомогою конфігураційних комірок (FUSE Bits, FUnction SElect Bits) CKSEL3..0.

До складу AVR входить вбудований генератор тактового сигналу. За замовчуванням для нової мікросхеми тактування здійснюється саме від цього пристрою. Для налаштування частоти вбудованого генератора може застосовуватися внутрішнє або зовшішнє RC-коло, рис. 1.132, *а*. Переваги такого підходу: відсутність або мінімальна кількість зовнішніх компонентів. Основни недолік: низька точність вбудованого генератора. Такий підхід може бути застосований для схем, які не вимагають високої точності в часовій області.



а) *б*) *в*)

Рис. 1.132. Способи тактування роботи мікроконтролера: *а* – від вбудованого генератора із зовнішнім RC-колом; *б* – за допомогою зовнішнього кварцевого резонатора; *в* – зовнішнім сигналом синхронізації

До виводів XTAL1 і XTAL2 мікроконтролера може підключатися зовнішній кварцевий резонатор, рис. 1.132, *б*. Кварцевий резонатор, в основі

роботи якого лежить використання п’єзоефекту, дозволяє задавати частоту функціонування мікроконтролера з високою точністю, рис. 1.133. Також тактування роботи мікроконтролера може здійснюватися від зовнішнього джерела синхронізації, рис. 1.132, *в*.

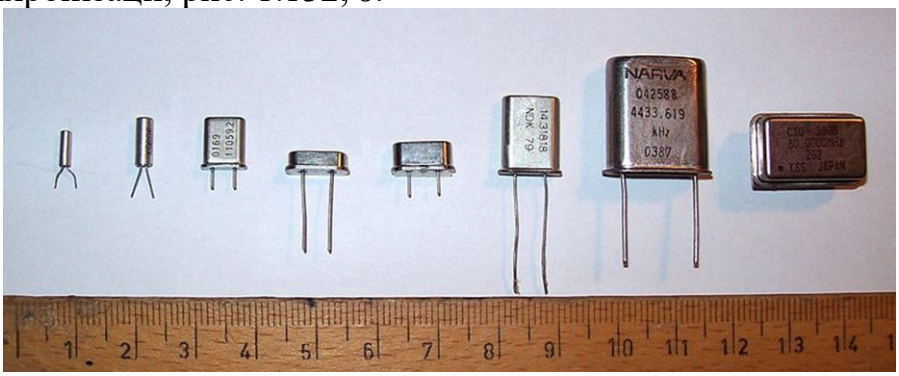


Рис. 1.133. Кварцеві резонатори

1.8.2. Скидання мікроконтролера

Скидання (реініціалізація, reset) – операція по переведенню мікроконтролера до вихідного стійкого стану. Вихідний стан характеризується нульовими значеннями у комірках оперативної пам’яті, регістрах загального призначення, регістрах введення/виведення, програмному лічильнику, регістрі-вказівнику стека тощо. Програміст, коли складає програму для мікроконтролера, орієнтується на нульові значення вказаних комірок на початку функціонування програми. Проте, при подачі на мікроконтролер живлення (або тимчасовому знижені напруги) тригери, на основі яких побудовані комірки пам’яті, можуть встановитися в довільний стан. Тому перед виконанням програми необхідно здійснити скидання мікроконтролера для підготовки його до роботи.

Скидання виконується в наступних випадках.

1. Включення напруги живлення мікроконтролера (скидання при включенні живлення). При подачі живлення на мікроконтролер, скидання забезпечується схемою POR (Power-on Reset), яка формує сигнал скидання протягом визначеного часового інтервалу, рис. 1.134. Початковий момент скидання визначається досягненням фактичної напруги живлення порогової величини, а тривалість – внутрішнім таймером скидання. Після закінчення тайм-аута такого таймера внутрішній сигнал reset знімається і контролер починає виконувати коди, записані до пам’яті програм, з нульової адреси.

2. Зниження напруги живлення нижче заданої величини (скидання при зниженні напруги живлення). Падіння напруги живлення може призвести до випадкової зміни вмісту регістрів та комірок ОЗП, що спричинить помилкове виконання програми. Контроль величини напруги здійснює схема BOD (Brown – Out Detection). При нестабільній напрузі мікроконтролер переводиться до стану reset, рис. 1.135. Після відновлення стабільного живлення запускається таймер скидання і, після закінчення його тайм-ауту, сигнал скидання знімається. Робота програми починається з початку, що має бути враховано в програмі.

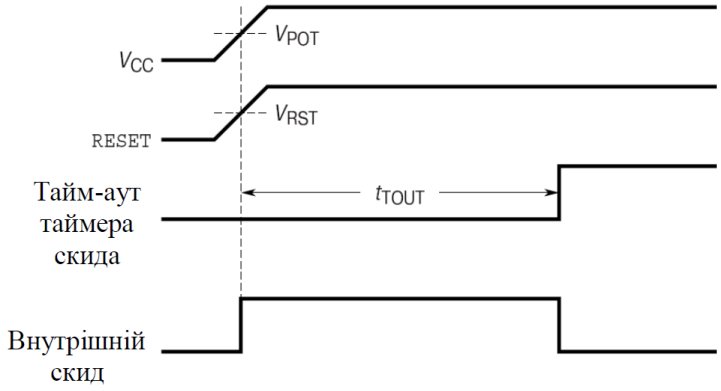


Рис. 1.134. Часові діаграми сигналів, що ілюструють скидання мікроконтролера під час ввімкнення напруги живлення

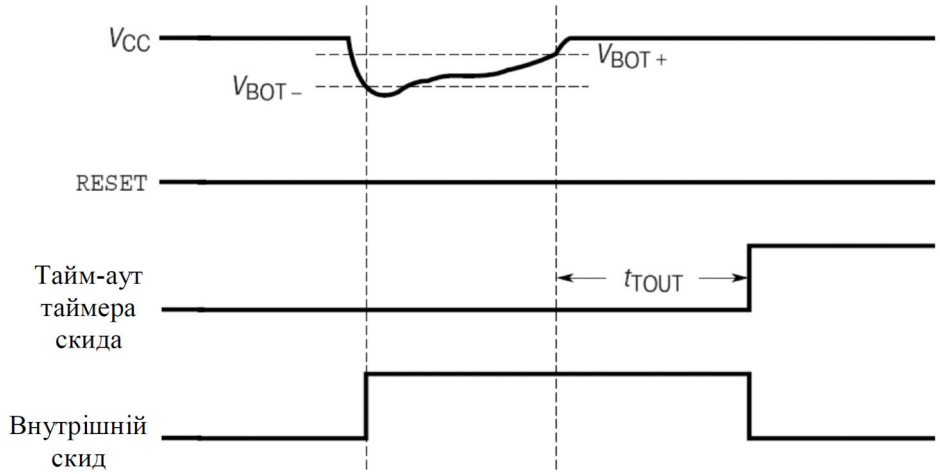


Рис. 1.135. Часові діаграми сигналів при скиданні мікроконтролера через зниження напруги живлення

3. Тайм-аут вартового таймера (скидання від вартового таймера). Сигнал скидання також може формуватися в разі зависання програми мікроконтролера, що було виявлено вартовим таймером. За тайм-аутом вартового таймера (при його використанні) генерується короткий імпульс скидання, що перезапускає виконання програми контролера, рис. 1.136. Найчастіше такі випадки трапляються через алгоритмічні помилки у програмному забезпеченні.

4. Подача сигналу низького рівня на вивід $\overline{\text{RESET}}$ (апаратне скидання). Можливо виконувати примусове скидання мікроконтролер шляхом використання інверсного входу $\overline{\text{RESET}}$. Скидання триває весь час, поки на вказаному вході наявний сигнал низького рівня, тобто поки він приєднаний до GND, рис. 1.137.

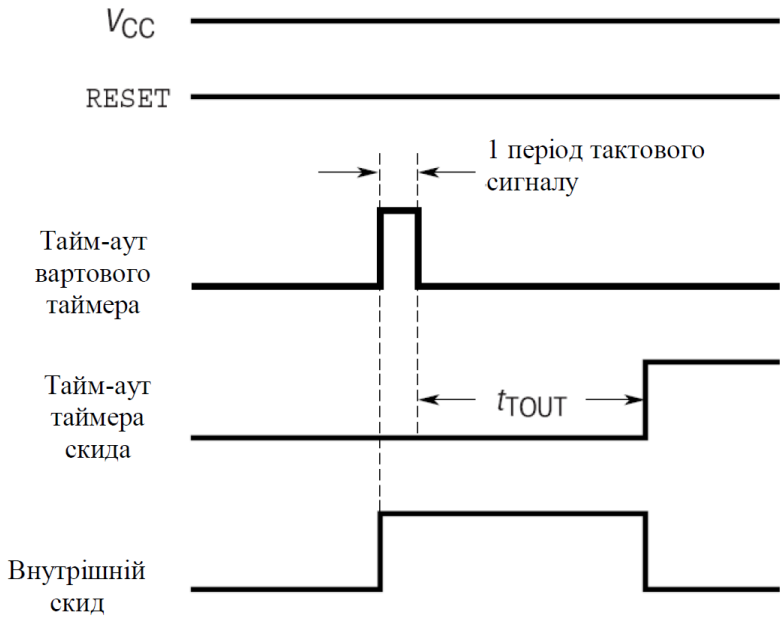


Рис. 1.136. Скидання мікроконтролера за тайм-аутом вартового таймера

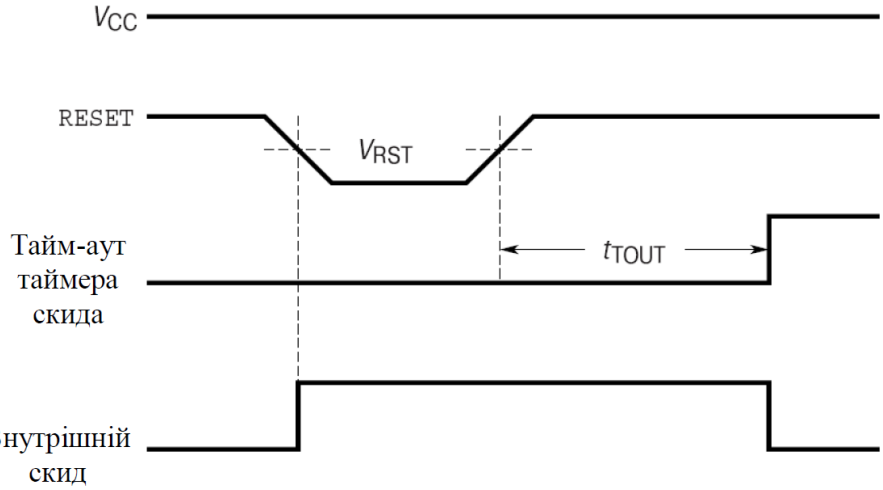


Рис. 1.137. Скидання мікроконтролера за подачею зовнішнього сигналу на вхід $\overline{\text{RESET}}$

1.8.3. Таймери / лічильники у складі мікроконтролерів

До складу мікроконтролерів AVR входять декілька таймерів/лічильників, які дозволяють вимірювати часові інтервали, підраховувати кількість внутрішніх та зовнішніх подій. Розглянемо таймери/лічильники загального призначення T0 та T1, якими обладнані практично всі моделі мікроконтролерів.

Функції таймера T0: відлік та вимірювання часових інтервалів, підрахунок зовнішніх імпульсів. Такі дії T0 виконує з використанням рахункового регістра, при переповненні якого здійснюється виклик переривання. Таймер T1, повторюючи можливості T0, додатково може генерувати запит на переривання при досягненні рахунковим регістром визначеної уставки. Також T1 можливо використовувати для формування широтно-імпульсно модульованого (ШІМ) сигналу. Таймери можуть

використовувати фізичні виводи мікроконтролера: T0 – вхід зовнішнього сигналу таймера T0; OC1A – вихід схеми порівняння таймера T1; IR – вихід переповнення таймера T0.

Керування T0 здійснюється з використанням двох регістрів: рахунковий регістр TCNT0 та регістр управління TCCR0. Для мікроконтролерів ATtiny 11,12 та 15 регістр управління має наступний формат:

	7	6	5	4	3	2	1	0
TCCR0	-	-	-	-	-	CS02	CS01	CS00

де розряди CS02, CS01, CS00 визначають джерело тактового сигналу, запускають і зупиняють таймер, а також визначають режим роботи T0: таймер або лічильник зовнішніх подій.

Тактування таймера здійснюється тактовим сигналом мікроконтролера через переддільник частоти. Змінюючи коефіцієнт поділу переддільника, можна змінювати період спрацювання таймера. Якщо T0 працює як лічильник, то інкремент (збільшення на 1) рахункового регістра виконується за кожним переднім фронтом сигналу на фізичному вході T0 мікроконтролера. При переповненні рахункового регістра TCNT0 (перехід зі значення 0xFF до наступного 0x00) генерується запит на переривання.

Для T1, на відміну від T0, не передбачена можливість підрахунку зовнішніх подій. T1 має два режими роботи: режим таймера та режим широтно-імпульсної модуляції. Програмне модель T1 включає наступні регістри:

– регістр управління TCCR1;

– регістр спеціальних функцій SFIOR;

– рахунковий регістр TCNT1 (реалізований як інкрементний лічильник);

– два регістри порівняння OCR1A і OCR1B.

Регістр управління TCCR1 використовується вибору джерела тактового сигналу, налаштування особливостей скидання таймера, дозволу режиму ШІМ, задавання режиму функціонування фізичного виводу мікроконтролера OC1A, запуска та зупинки T1. В режимі таймера за кожним імпульсом, що надходить на тактовий вхід таймера, виконується інкремент рахункового регістра TCNT1. В момент переходу таймера зі стану 0xFF до стану 0x00 генерується запит на переривання. При кожній зміні вмісту рахункового регістра його вміст порівнюється з регістром порівняння OCR1A. Якщо їх вміст тотожний, генерується запит на відповідне переривання. Окрім генерації переривання при тотожності рахункового регістра і регістра порівняння можуть виконуватися також скидання таймера/лічильника або зміна стану виводу OC1A мікроконтролера.

1.8.4. Реалізація цифро-аналогового перетворення

Таймер T1 може бути налаштований для роботи в режимі широтно-імпульсного модулятора. Це використовується для керування силовими напівпровідниковими ключами у складі DC/DC перетворювачів, формування мікроконтролерною схемою аналогового сигналу заданої величини тощо. Частота та шпаруватість ШІМ сигналу налаштовується програмно.

В режимі ШІМ вміст рахункового регістра TCNT1 змінюється від $00 до

значення, що знаходиться в регістрі OCR1B, після чого рахунковий регістр скидається і цикл повторюється. При тотожності рахункового регістра і регістра порівняння OCR1A стан фізичного виводу OC1A змінюється, рис. 1.138. Частота вихідного сигналу визначається вмістом регістра OCR1B, а шпаруватість – OCR1A.

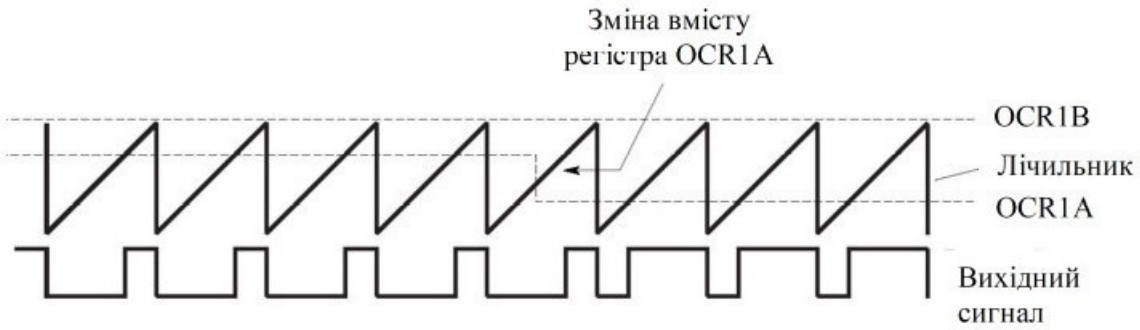


Рис. 1.138. Формування ШІМ-сигналу за допомогою таймера T1

1.8.5. Вартовий таймер

Вартового таймера – пристрій, який запобігає зависанню мікроконтролера. Помилки при написанні програмного забезпечення, зовнішні збурення можуть призвести до входу мікроконтролера у нескінеченний цикл, вихід з якого програмно не був передбачений. Якщо мікроконтролер зациклюється і нездатний виконувати свої функції, вартовий таймер забезпечує скидання. Програма розпочинається з початку, відновлюється нормальне функціонування пристрою.

Вартовий таймер одержує тактовий сигнал від незалежного генератора. Якщо вартовий таймер включений, то через визначені проміжки часу (при настанні тайм-ауту) виконується скидання мікроконтролера. В програмі має бути передбачене періодичне обнулення лічильного лічильного регістра таймера за допомогою команди WDR. Якщо програма не виконує скидання вартового таймера, то робиться висновок про збій і вартовий таймер виконує reset мікроконтролера.

1.8.6. Вбудований аналоговий компаратор

Всі моделі мікроконтролерів AVR оснащені вбудованим аналоговим компаратором. Він призначений для порівняння значень напруги на двох виводах мікроконтролера. Результатом порівняння є логічне значення, яке може бути прочитане з програми. За результатом порівняння також може бути згенеровано переривання.

Компаратор підключено до двох виводів мікроконтролера. Вивід AIN0 відповідає прямому входу компаратора, AIN1 – інверсному. Перевищення напруги на AIN0 напруги на AIN1 призводить до зчитування з виходу компаратора лог. «1». В протилежному випадку компаратор видає лог. «0». За необхідності користуватися компаратором виводи AIN0 та AIN1 мають бути сконфігуровані як входи.

Для керування компаратором та зчитування результату порівняння використовується регістр ACSR (Analog Comparator Status Register):

	7	6	5	4	3	2	1	0
ACSR	ACD	ACBG	ACO	ACI	ACIE	–	ACIS1	ACIS0

Розряди регістра ACSR мають наступне призначення:

ACD – ввімкнення компаратора («0» – ввімкнено; «1» – відключено, тобто після скидання мікроконтролера компаратора автоматично запускається);

ACBG – підключення до прямого входу компаратора внутрішнього джерела опорної напруги («0» – не підключено; «1» – підключено);

ACO – результат порівняння;

ACI – прапор переривання компаратора;

ACIE – дозвіл переривань компаратора;

ACIS1:ACIS0 – умова виникнення переривання компаратора.

Результат порівняння зберігається в розряді ACO регістра ACSR. Відповідно до результату порівняння схема компаратора може генерувати запит на переривання. Якщо стан виходу компаратора (розряд ACO) змінився заданим чином, встановлюється прапор переривання ACI регістра ACSR і генерується запит на переривання. Умова генерації запиту на переривання від компаратора визначається станом розрядів ACIS1:ACIS0 регістра ACSR.

1.8.7. Аналого-цифровий перетворювач у складі мікроконтролера

Мікроконтролери AVR обладнані вбудованим аналого-цифровим перетворювачем з вхідним аналоговим мультиплексором. Розглянемо модель АЦП на прикладі мікроконтролера ATtiny15L. Вбудований АЦП здійснює перетворення з 10-розрядною точністю, тобто може видавати значення від 0 до 1023, всього 1024 рівні квантування сигналу. Швидкодія досягає 15 тис. вибірок за секунду.

Для спрощення опитування декількох аналогових давачів на вході АЦП встановлено 4-канальний аналоговий мультиплексор. Це дозволяє користуватися 4 каналами з несиметричними входами або 1 каналом з диференціальним входом з можливістю 20-кратного попереднього посилення. Існує можливість використовувати внутрішнє або зовнішнє джерело опорної напруги.

Модуль АЦП може функціонувати в одиночному режимі, коли після одержання результату модуль зупиняється, або неперервному, коли наступний цикл перетворення запускається одразу після закінчення поточного.

Для керування АЦП використовується програмна модель у складі, рис. 1.139:

– регістр стану ADCSR (Analog Digital Converter Status Register);

– регістр управління вхідним мультиплексором і додаткових функцій ADMUX;

– регістрова пара ADCH:ADCL для збереження результату перетворення.

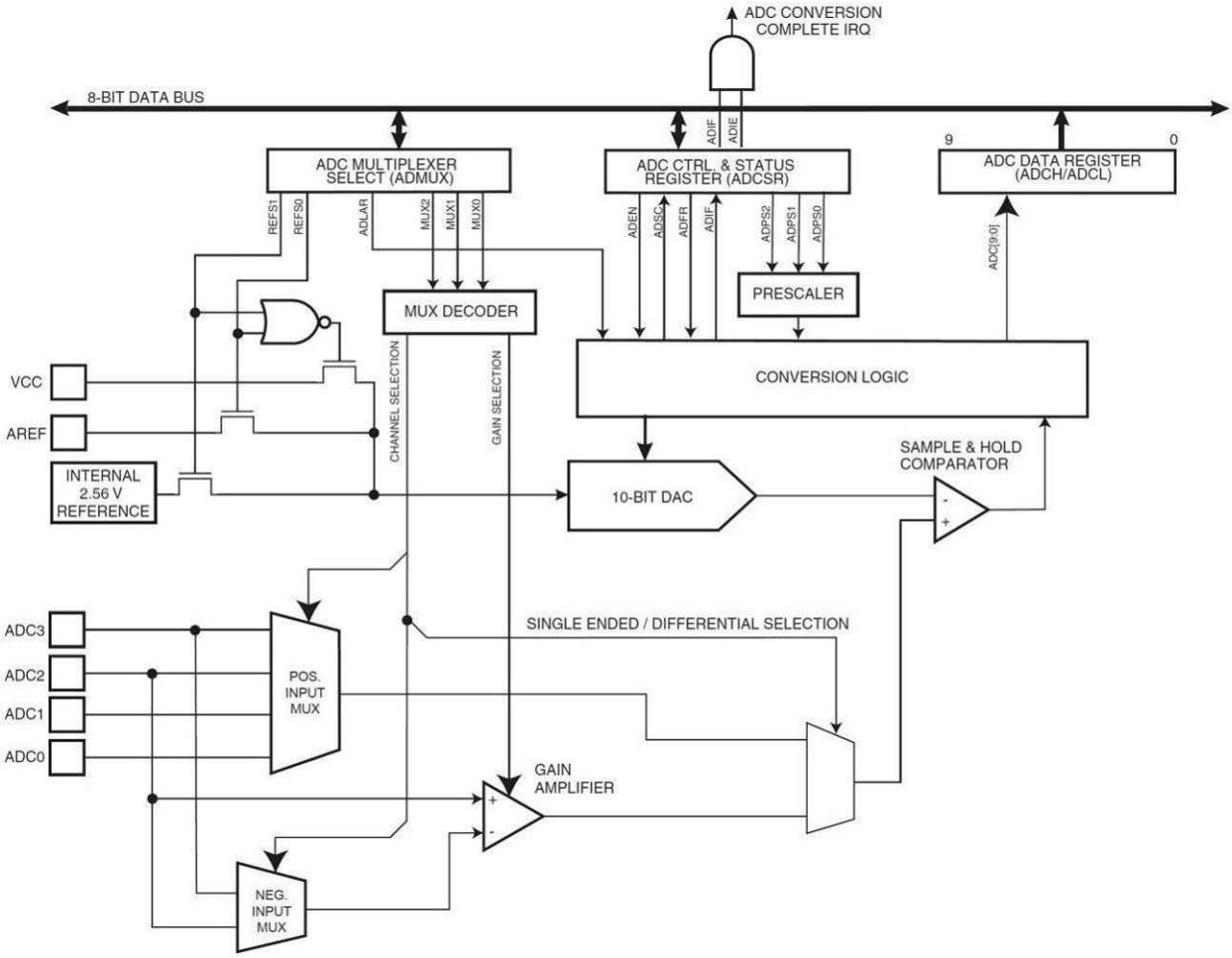


Рис. 1.139. Структурна схема модуля АЦП у складі мікроконтролера ATtiny15L

Регістр стану ADCSR має наступний формат:

	7	6	5	4	3	2	1	0
ADCSR	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0

Призначення розрядів регістра наступне:

ADEN – дозвіл роботи АЦП («1» – дозволено; «0» – заборонено);

ADSC – запуск перетворення («1» – запустити);

ADFR – вибір режиму роботи АЦП («0» – одиночне перетворення);

ADIF – прапор переривання АЦП;

ADIE – дозвіл переривання АЦП;

ADPS2:ADPS0 – вибір частоти перетворення.

Робота з модулем АЦП починається з дозволу роботи, для чого необхідно встановити ADCSR.ADEN=1. Необхідно обрати режим роботи: одиночні перетворення (ADCSR.ADFR=0) або неперервні (ADCSR.ADFR=1). Також необхідно обрати частоту перетворення (коефіцієнт переддільника частоти) за допомогою розрядів ADPS2:ADPS0. Найбільша точність перетворення досягається на частотах до 200 кГц. Якщо результат перетворення передбачається зчитувати в підпрограмі оброблення переривань, необхідно дозволити переривання АЦП: ADCSR.ADIE=1. Перетворення починається після встановлення прапору ADSC. Тривалість циклу перетворення становить

13 тактів. Результат перетворення зберігається до 16-розрядної пари регістрів ADCH:ADCL. В цей момент встановлюється прапор ADIF регістра ADCSR і генерується запит на переривання. В режимі неперервних перетворень після цього запускається наступний цикл. Вибір каналу вхідного мультиплексора, джерела опорної напруги та способу вирівнювання 10-розрядного результату в 16-розрядній регістровій парі ADCH:ADCL визначається розрядами регістра ADMUX.

1.8.8. Програмні засоби для роботи з АЦП

Регістри модуля АЦП у більшості моделей мікроконтролерів розташовані в додатковій області оперативної пам'яті. Для звернення до таких регістрів команди in та out не можуть бути застосовані, оскільки їх область досяжності обмежена основним адресним простором регістрів введення-виведення. Тому на мові асемблера для роботи з регістрами з додаткової області ОЗП використовуються команди непрямого звертання до пам'яті:

ST X, Rr – команда непрямого запису вмісту регістра загального призначення Rr до пам'яті даних. Адреса комірки, до якої здійснюється запис, має міститися в індексному регістрі X (регістрова пара R27:R26);

LD Rd, X – непряме зчитування вмісту комірки ОЗП до регістру загального призначення Rd. При цьому індексний регістр X має містити адрес комірки ОЗП.

Виводи аналогового мільтиплексора на платі Arduino позначаються A0, A1, …, що відповідає входам мікросхеми ADC0, ADC1, …. Для автоматичного запуску АЦП та зчитування з входу pin величини вхідного сигналу використовується функція analogRead(pin), яка повертає цілочислене значення в діапазоні від 0 до 1023.

1.8.9. Питання для самоперевірки

1. Для чого необхідно тактувати мікроконтролер?
2. Які способи тактування роботи мікроконтролера Вам відомі?
3. Які особливості тактування від вбудованого генератора?
4. Яким чином здійснюється тактування за допомогою кварцевого резонатора?
5. Для чого виконується скидання мікроконтролера?
6. Які події призводять до скидання?
7. Поясніть виконання скидання при включенні напруги живлення.
8. Яким чином здійснюється скидання при зниженні напруги живлення?
9. Для чого виконується скидання при спрацюванні вартового таймера?
10. Яким чином здійснюється апаратне скидання мікроконтролера?
11. Опишіть функціонування таймера T0.
12. Які особливості робота таймера T1?
13. Яким чином таймер T1 може генерувати ШІМ-сигнал?
14. Поясність призначення та особливості роботи вартового таймера.
15. Яким чином працює вбудований аналоговий компаратор?
16. В яких режимах може працювати модуль АЦП мікроконтролера?

17. Які регістри використовуються модулем АЦП для роботи?
18. Поясніть улаштування модуля АЦП за структурною схемою.

1.9. Обмін даними в мікропроцесорній системі

Паралельні та послідовні інтерфейси. Інтерфейс RS-232. Інтерфейс RS-485. Інтерфейс «струмова петля». Універсальний асинхронний (синхронний / асинхронний) приймач-передавач UART (USART). Послідовний периферійний інтерфейс SPI. Послідовний двопровідний інтерфейс TWI (I2C). Польова шина Profibus. Мережа Profinet.

1.9.1. Паралельні та послідовні інтерфейси

Інтерфейс – сукупність засобів, методів і правил взаємодії між елементами системи. Для обміну даними між двома мікропроцесорними пристроями використовуються інтерфейси двох основних типів: паралельні та послідовні

Паралельні інтерфейси передбачають передачу декількох біт (тетрада, байт, слово тощо) в один момент часу. Для передачі кожного розряда використовується окремий провідник, рис. 1.140.

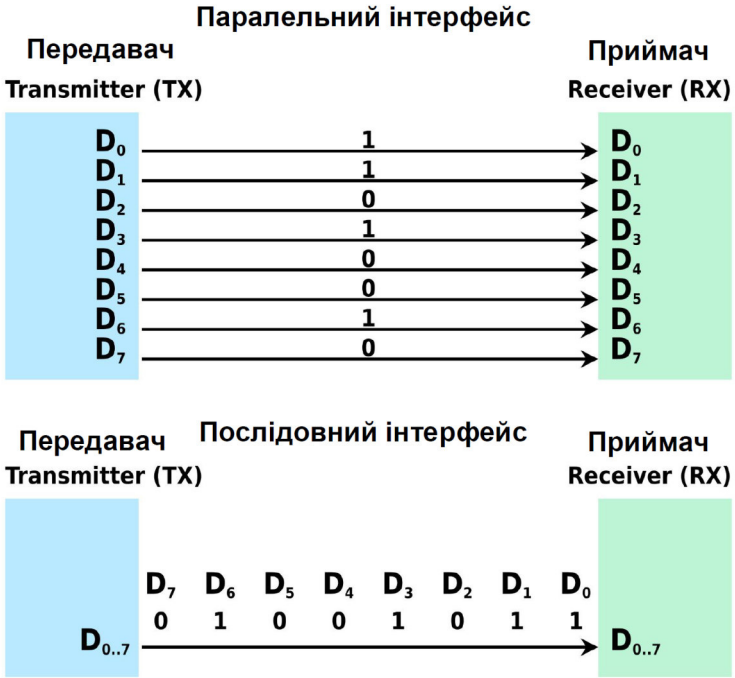
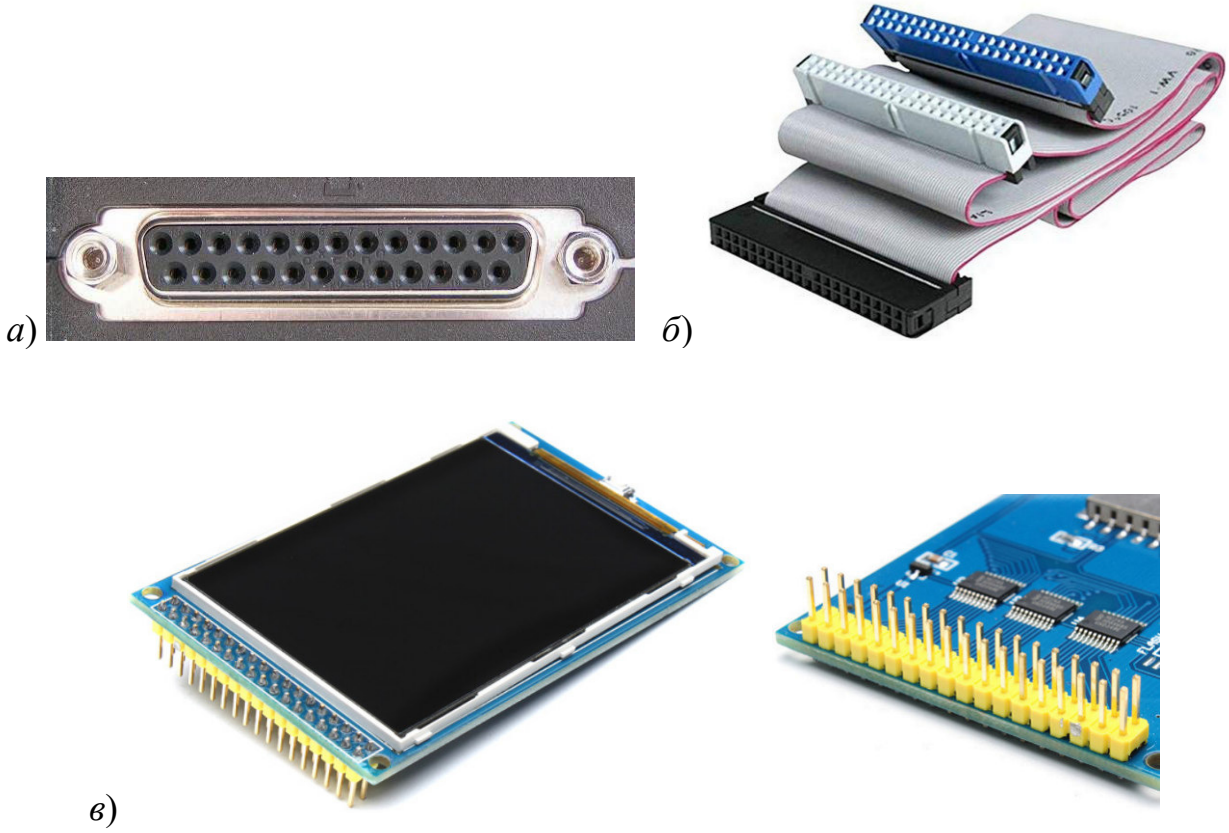


Рис. 1.140. Порівняння передачі даних за інтерфейсами паралельного та послідовного типів

При розробленні паралельних інтерфейсів вважалося, що збільшуючи кількість біт, які передаються одночасно, та частоту обміну можна суттєво збільшити швидкість передачі даних. Проте, на практиці з'ясувалося, що збільшення кількості паралельних провідників нераціонально, оскільки інтерфейсні кабелі стають громіздкими. Підвищення частоти обміну призводить до того, що в провідниках паралельної лінії зв'язку, вихідних та

вхідних каскадах передавача та приймача виникають затримки різної величини, що спричиняє похибки обміну даними. Паралельний інтерфейс використовувався в кінці 1990-х та на початку 2000-х років у LPT-портах комп'ютера для приєднання принтерів, сканерів. Також жорсткі диски приєднувалися до материнської плати за паралельним IDE (Integrated Drive Electronics) інтерфейсом. На сьогодні паралельний інтерфейс деколи використовується для приєднання рідкокристалічного дисплея до мікроконтролера, рис. 1.141.

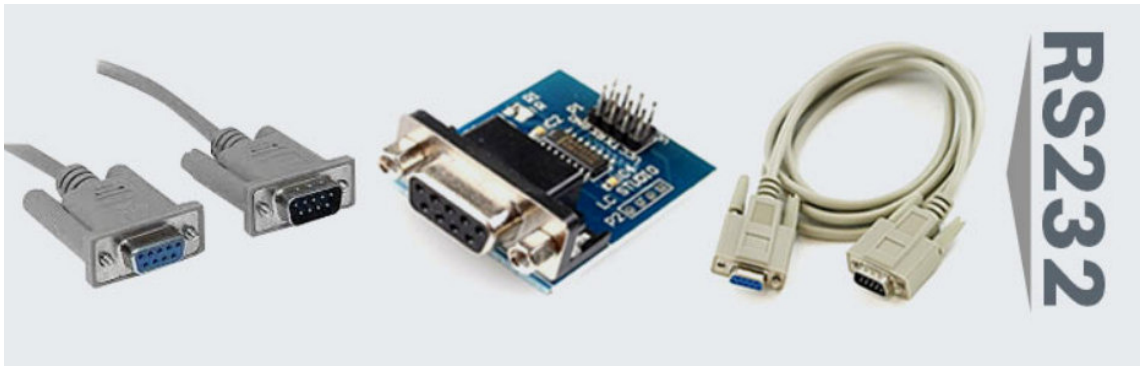


а) *б*) *в*)

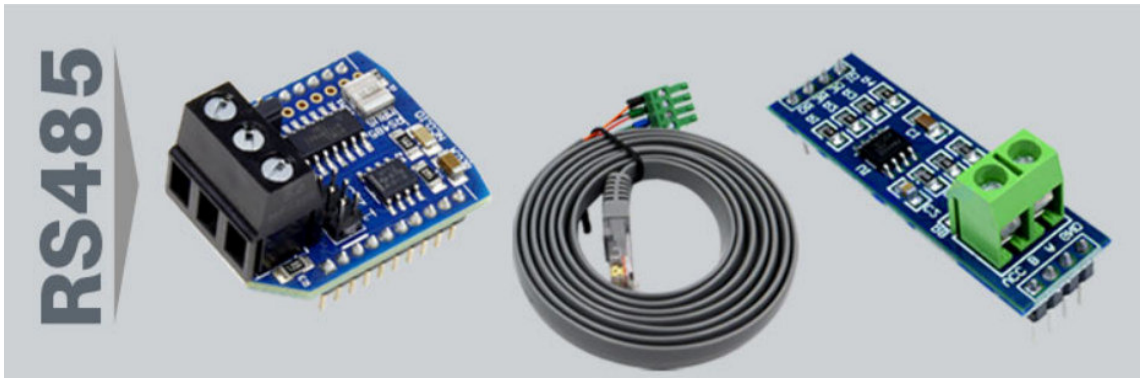
Рис. 1.141. Приклади реалізації паралельних інтерфейсів:
а – LPT-порт комп’ютера; *б* – шлейф (40 паралельних провідників) для підключення жорсткого диску до материнської плати за IDE інтерфейсом;
в – кольоровий TFT-дисплей, що призначений для підключення до Arduino Mega 2560 за 16-розрядним паралельним інтерфейсом

Послідовні інтерфейси передбачають передачу в кожен момент часу тільки одного інформаційного біти. Це дає змогу суттєво знизити кількість проводів, що з’єднують два пристрої. Крім того, порівняно з паралельним інтерфейсом, зникає проблема нерівномірних затримок для окремих бітів. Послідовні інтерфейси на сьогодні є найбільш розповсюдженими. Приклади: COM-порт комп’ютера, що використовує інтерфейс RS-232C, такий фізичний порт на сьогодні на сьогодні витіснений USB, однак на віртуальному рівні послідовний порт продовжує використовуватися; інтерфейс RS-485, що застосовується в промисловості; інтерфейс універсальної послідовної шини USB, рис. 1.142.

а)



б)



в)

Рис. 1.142. Приклади послідовних інтерфейсів для обміну даними:
а – COM-порт комп'ютера, що реалізує RS-232; *б* – інтерфейс RS-485;
в – різновиди USB-роз'ємів

Головна перевага послідовних портів перед паралельними – зниження числа провідників. Також при високих швидкостях передачі даних, коли на електричні сигнали суттєво впливають затримки в лініях, послідовні інтерфейси прийнятніші від паралельних. Крім того, при реалізації послідовного інтерфейсу можна збільшити дальність зв'язку в порівнянні з паралельним інтерфейсом. У мікроконтролерних пристроях з малими об'ємами даних швидкість передачі не є критичним чинником при виборі типу інтерфейсу. В першу чергу має значення кількість проводів, тобто число виводів мікроконтролера, що використовуються для передачі даних. Тому при

проектуванні мікропроцесорних пристроїв автоматизації перевагу віддають послідовним інтерфейсам.

Послідовна передача даних може здійснюватися в двох режимах: асинхронному або синхронному.

При асинхронній передачі кожному байту передує старт-біт, що сигналізує приймачу про початок посилки, за яким йдуть біти даних і, можливо, біт паритету (парності). Завершується посилка стоп-бітом, що гарантує паузу між посилками, рис. 1.143. Після стоп-біта може тривати нерегламентована пауза, після чого в будь-який момент може бути посланий старт-біт наступної посилки. Призначення старт-біта – забезпечити синхронізацію приймача за сигналом передавача. Для асинхронного режиму прийнятий ряд стандартних швидкостей обміну: 50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19 200, 38 400, 57600 і 115200 біт/с.

а)

б)

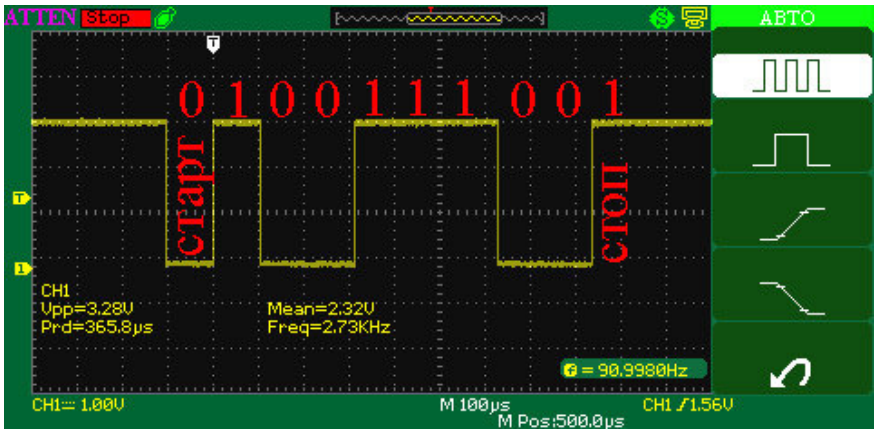


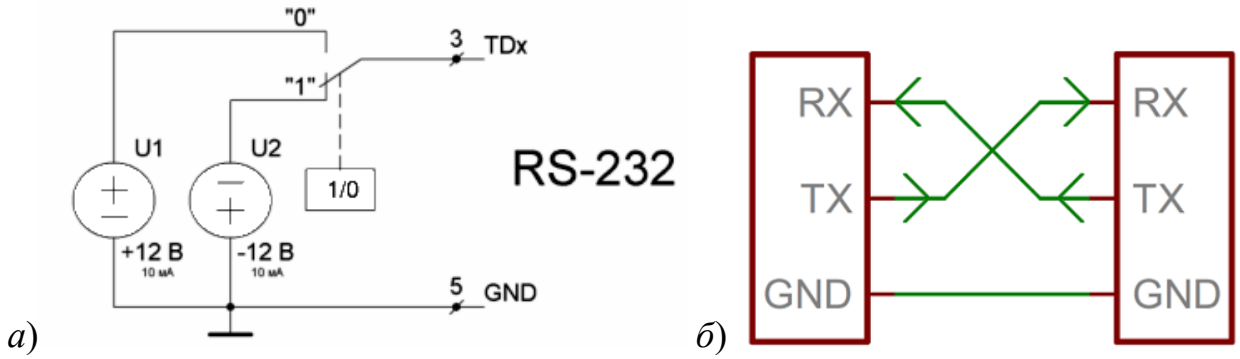
Рис. 1.143. Формат асинхронної передачі даних за послідовним інтерфейсом:
а – загальна схема; *б* – сигнали на екрані осцилографа

При синхронному режимі за каналом зв’язку дані передають неперервно. Такий режим застосовується в разі необхідності постійної передачі великих об’ємів даних. Для уникнення спотворення даних використовується зовнішня синхронізація приймача з передавачем.

1.9.2. Інтерфейс RS-232

Стандарт RS-232 забезпечує обмін даними між двома пристроями за послідовним каналом. Сигнал передається напругою відносно загального проводу – схемної землі. Гальванічна розв'язка передавача та приймача не реалізується. Логічна одиниця передається напругою з діапазону від –12В до –3В. Логічному нулю відповідає діапазон напруг від +3В до +12В, рис. 1.144, *а*. Діапазон від –3 В до +3 В називається зоною нечутливості. За інтерфейсом RS-232 можливо передавати дані на відстань до 15 м при максимальній швидкості 115200 біт/с.

Керування потоком даних може здійснюватися апаратно або програмно. Апаратний протокол забезпечує найбільш швидку реакцію передавача на стан приймача. Апаратний протокол передбачає наявність додаткових проводів для керування процесом передачі даних. Програмний протокол управління потоком XON/XOFF потребує двонаправленого каналу обміну даними, проте для організації зв'язку достатньо 3х проводів (TX – передача, RX – прийом, GND – схемна земля), рис. 1.144, *б*.



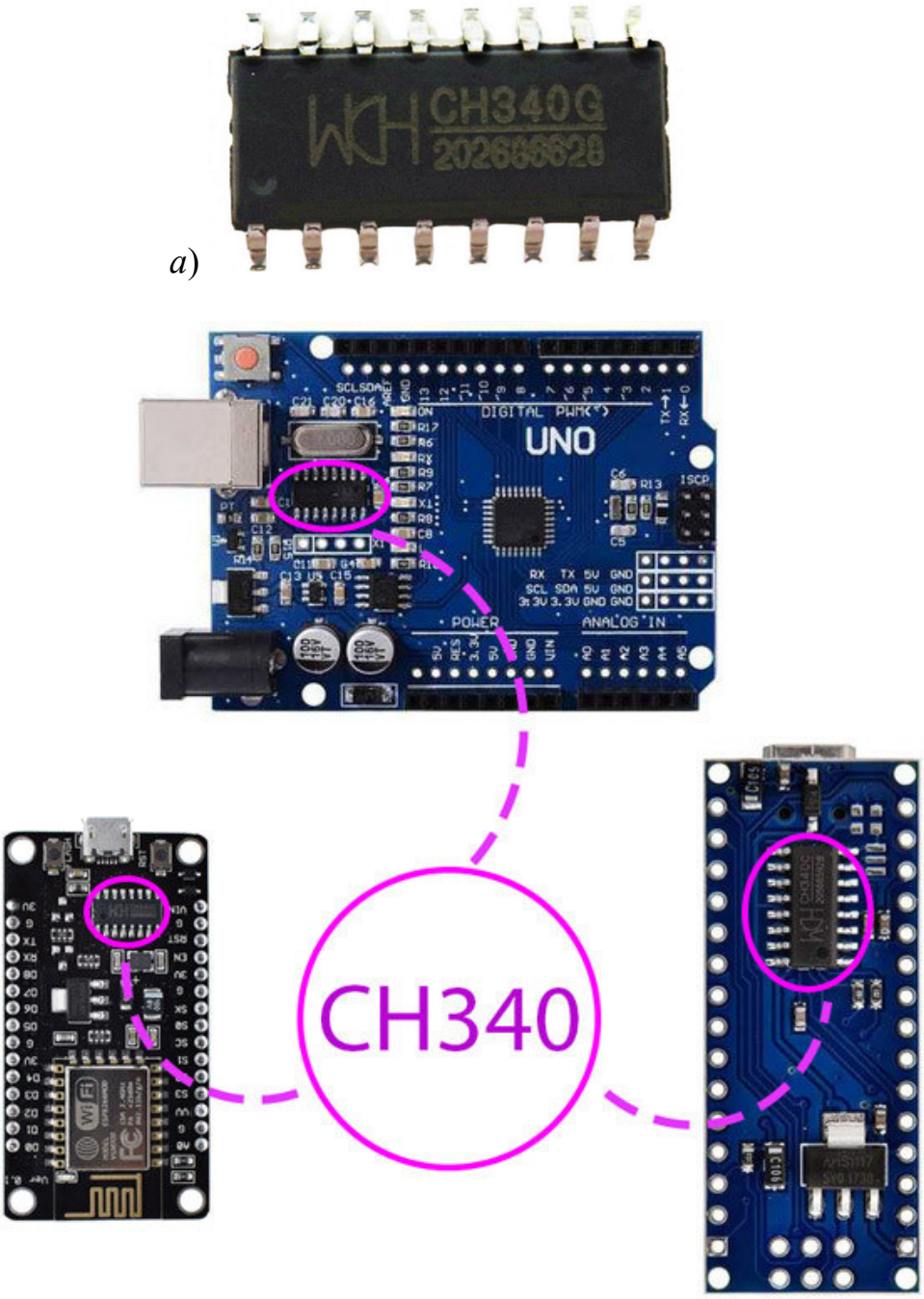
а) *б*)

Рис. 1.144. Заступна схема передавача RS-232 (*а*) та двонаправлене з'єднання двох пристроїв (*б*)

На сьогодні RS-232 застосовується в мікропроцесорних пристроях для обміну даними. Також широко використовується програмне емуляція COM порта, оскільки робота з ним є надзвичайно простою. Наприклад, при підключенні багатьох цифрових пристроїв до комп'ютера (в тому числі – мікропроцесорних плат Arduino, терміналів релейного захисту, мобільних телефонів, програматорів тощо) фізично використовується USB порт, сигнали якого за допомогою драйвера перетворюються у програмні сигнали RS-232 (COM-порт) і прикладне програмне забезпечення працює не з USB портом, а з COM-портом.

Розглянемо приклад використання віртуального COM порта для програмування плат Arduino. Мікроконтролер, що використовується у складі плати Arduino, використовує властивість самопрограмування для завантаження кодів програми до flash-пам'яті. Попередньо до пам'яті програм завантажується спеціальний програмний код, що називається завантажувачем. Коди основної програми приймаються за інтерфейсом RS-232 та записуються до flash-пам'яті вказаним завантажувачем. Однак, плата підключається до комп'ютера за допомогою USB і програмні коди надходять до плати через USB. Перетворення

електричних сигналів інтерфейсу USB у RS-232 здійснюється мікросхемою CH340, рис. 1.145.



а)

б)

Рис. 1.145. Перетворювач інтерфейсу CH340 (*а*), що використовується у складі плат Arduino (*б*)

Програмне середовище ArduinoIDE орієнтоване на завантаження програми до плати через COM-порт. Оскільки фізично такий порт у сучасних комп'ютерах відсутній, а плата підключається через USB, то використовується драйвер, який перетворює інформацію зі стандарта RS-232 у стандарт USB. В результаті застосування такого драйвера плата Arduino, що підключається до комп'ютера, розпізнається в диспетчері задач як пристрій, що підключений до віртуального COM-порта, рис. 1.146.

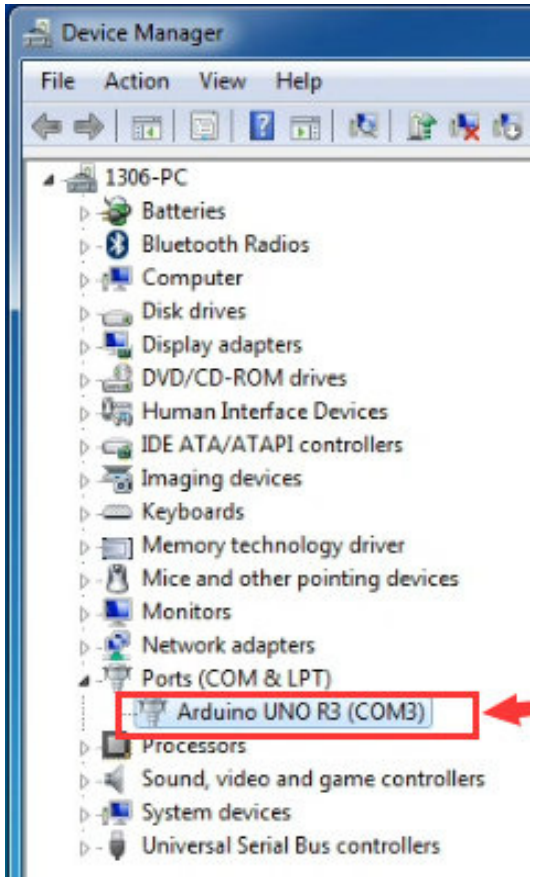


Рис. 1.146. Розпізнавання плати Arduino, що підключена до фізичного USB порта, як підключеної до віртуального COM порта завдяки використанню драйвера

1.9.3. Інтерфейс RS-485

RS-485 (Recommended Standard 485) – рекомендований стандарт передачі даних по двопровідному напівдуплексному багатоточковому послідовному симетричному каналу зв'язку. RS-485 створювався для розширення фізичних можливостей інтерфейсу RS-232 по передачі двійкових даних.

Порівняно з RS-232, RS-485 має наступні переваги:

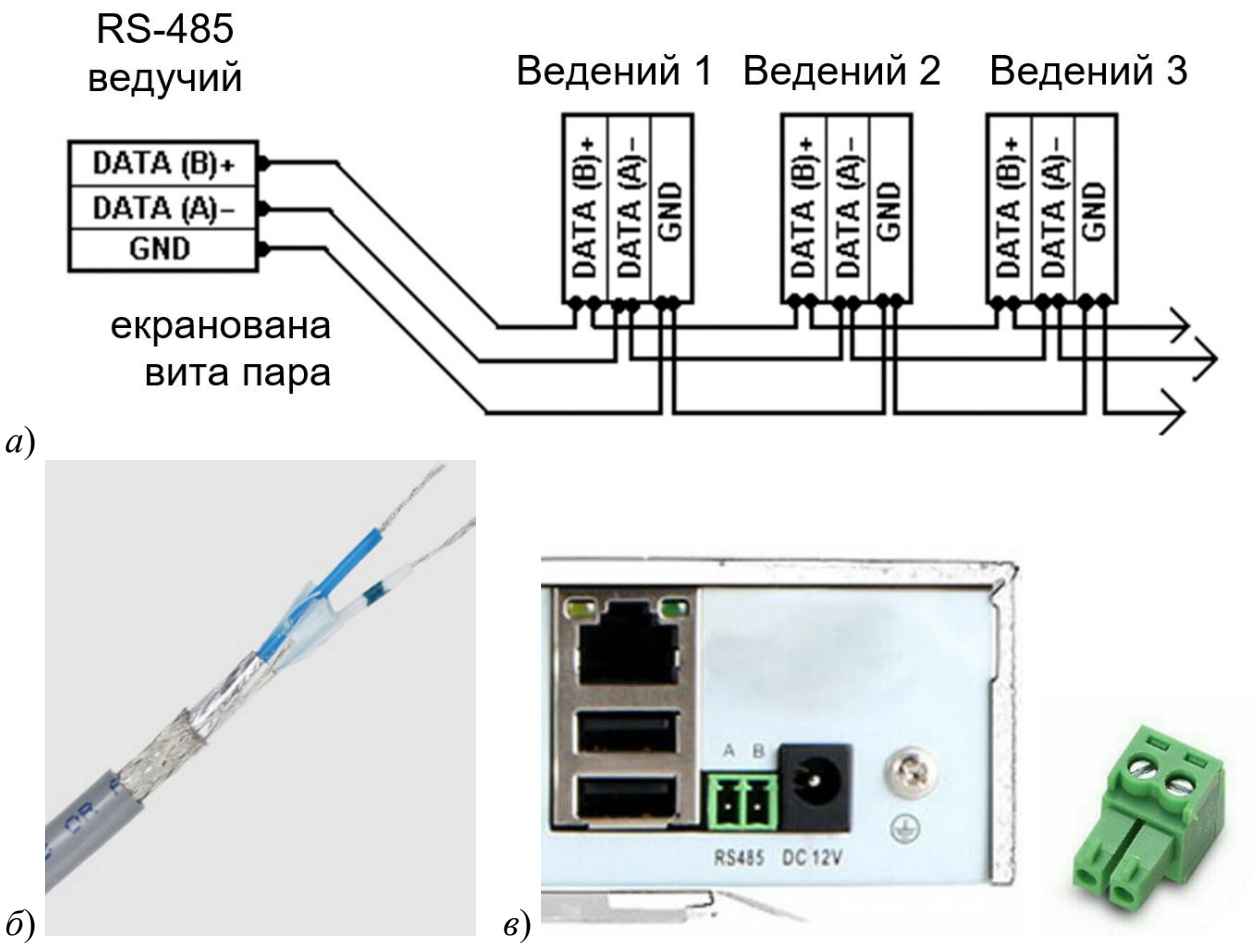
– забезпечує з’єднання до 32х пристроїв замість тільки двох у RS-232, тому RS-485 називається багатоточковим;

– передає дані на відстань до 1200 м замість 15 м у RS-232;

– збільшена максимальна швидкість обміну даними до 10 Мбіт/с;

– лінія зв’язку представляє собою 2 провідника (вита пара), рис. 1.147.

Форма сигналу RS-485 повністю повторює форму сигналу RS-232. Відмінність полягає у величинах напруги, що відповідають логічним рівням: у RS-232 лог. «1» кодується –12 В, а в RS-485 +5 В). Якщо у RS-232 напруга формується відносно схемної землі, то у RS-485 розглядається напруга між двома інформаційними проводами A і B, рис. 1.148.



а) *б*) *в*)

Рис. 1.147. Схема з’єднання ведучого та декількох ведених пристроїв за RS-485 (*а*), екранована вита пара (*б*) та один з варіантів роз’єму для підключення лінії RS-485 (*б*)

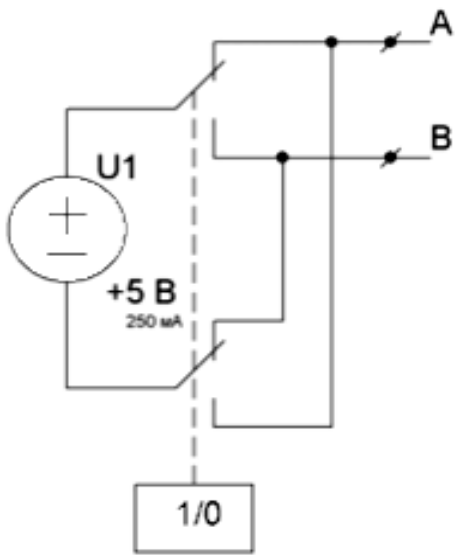


Рис. 1.148. Заступна схема, що пояснює формування сигналів RS-485

Властивості інтерфейсу стандарту RS-485:

1. Двонаправлена напівдуплексна передача даних. Потік послідовних даних передається одночасно тільки в один бік, передача даних в інший бік вимагає перемикання приймача.

2. Симетричний канал зв'язку. Для приймання/передачі даних використовуються два рівнозначні сигнальні проводи, які позначаються A і B. По цих проводах здійснюється послідовний обмін даними в обох напрямах (по черзі). При використанні витої пари симетричний канал істотно підвищує стійкість сигналу до синфазної перешкоди і добре пригнічує електромагнітні завади, що створюються корисним сигналом.

3. Використовується диференціальний (балансний спосіб передачі даних). На виході приймача змінюється різниця потенціалів, при передачі лог. «1» різниця потенціалів між A і B позитивна, при передачі лог. «0» вказана різниця потенціалів негативна. Струм між A і B, при передачі «0» і «1», протікає (балансує) в протилежних напрямах.

4. Багатоточковість. Допускає множинне підключення приймачів і приймачів до однієї лінії зв'язку. При цьому допускається підключення до лінії тільки одного передавача в даний момент часу і декількох приймачів, інші передавачі повинні чекати звільнення лінії зв'язку для передачі даних.

5. Низкоімпедансний вихід передавача. Буферний підсилювач передавача характеризується наявністю низькоомного виходу. Це реалізує передачу сигналів одночасно до багатьох приймачів. Навантажувальна здатність передавача становить 32 приймачі.

6. Зона нечутливості ±200мВ, що завадостійкість передачі даних.

1.9.4. Інтерфейс «струмова петля»

Поширеним варіантом послідовного інтерфейсу є струмова петля. Використовується логіка роботи RS-232, тільки логічні сигнали передаються не рівнями напруги, а величиною струму по замкненому колу між передавачем та приймачем. Лог. «1» кодується струмом 20 мА, лог. «0» – відсутністю струму. Застосування сигналу за струмом замість сигналу за напругою дозволяє підвищити надійність обміну даними. В разі обриву лінії до приймача не буде надходити стоп-біт, оскільки обрив лінії відповідає постійному логічному нулю.

В струмовій петлі зазвичай використовується гальванічна розв’язку вхідних кіл приймача від схеми пристрою. При цьому джерелом струму в петлі є передавач (цей варіант називають активним передавачем). Можливо і живлення від приймача (активний приймач), при цьому вихідний ключ передавача може бути також гальванічно розв'язаний з іншою схемою передавача, рис. 1.149. Струмова петля з гальванічною розв'язкою дозволяє передавати сигнали на відстані до декількох кілометрів. Відстань визначається опором пари проводів та рівнем завад.

Рис. 1.149. Перетворювач інтерфейсу RS-232–«струмова петля», що дозволяє під’єднувати до 8 пристроїв зі струмовим колом

1.9.5. Універсальний асинхронний (синхронний / асинхронний) приймач-передавач UART (USART)

Для забезпечення зв'язку по каналах RS-232 або RS-485 мікроконтролерів Atmel сімейства Mega і вище мають у своєму складі апаратні модулі універсального асинхронного приймача-передавача UART (Universal Asynchronous Receiver and Transmitter) або універсального синхронно/асинхронного приймача-передавача USART.

Відмінність UART(USART) від RS-232 полягає у рівнях напруги, що кодують логічні сигнали. Рівні сигналів UART(USART) відповідають стандарту транзисторно-транзисторної логіки:

	RS-232	UART
Лог. «1»	від –12 В до –3 В	від +2,4 В до +5 В.
Лог. «0»	від +3В до +12 В	від 0 В до +0,4 В

В мікроконтролерах AVR з UART апаратно-програмними засобами підтримуються тільки дві лінії:

RxD – прийом даних;

TxD – передача даних.

Оскільки рівні напруг RS-232 та UART різні, для підключення мікроконтролера до пристрою RS-232 необхідно використовувати перетворювач рівнів. В якості останнього найчастіше використовується інтегральна мікросхема MAX232 (рис. 1.150), що перетворює сигнали послідовного порту RS-232 в сигнали, придатні для використання в цифрових схемах на базі ТТЛ-технологій.

У більшості моделей сімейства Mega UART замінений більш функціональним USART (синхронно-асинхронний приймач-передавач). USART повністю сумісний з UART (окрім найменувань деяких регістрів), і відрізняється від UART підтримкою синхронного режиму обміну даними.

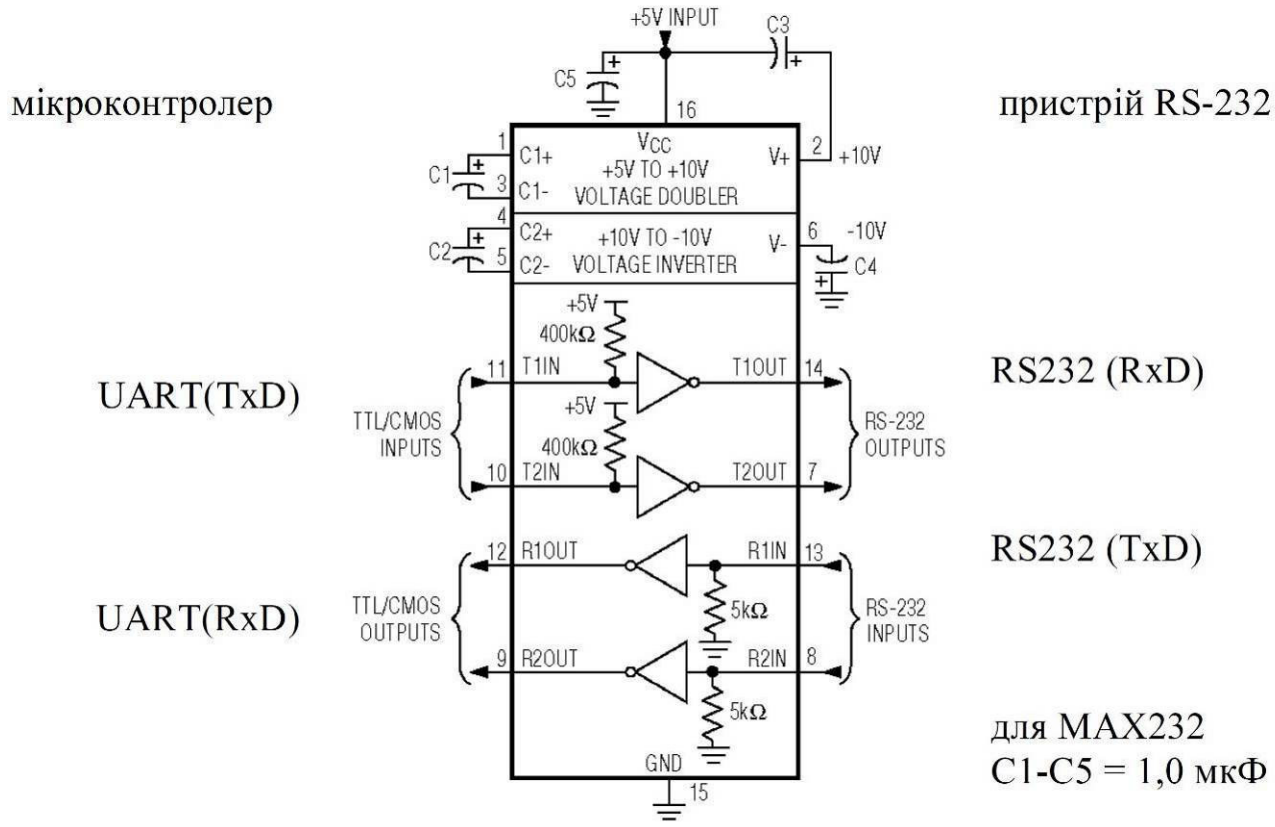


Рис. 1.150. Перетворювач рівнів сигналів MAX232 між UART і RS-232

UART(USART) використовується для віддаленої передачі даних. В разі використання перетворювача інтерфейсів UART – RS-485, можливо організувати віддалений обмін даними. Таке технічне рішення використовується при побудові розподілених мереж обліку електроенергії або керування електротехнічними комплексами.

1.9.6. Послідовний периферійний інтерфейс SPI

Інтерфейс SPI (Serial Peripheral Interface), реалізований в усіх мікроконтролерах сімейства Mega і вище, забезпечує виконання наступних функцій:

– обмін даними між мікроконтролером і різними периферійними пристроями, які розташовуються в межах одного мікропроцесорного блоку (зокрема, мікросхеми АЦП, ЦАП, мікросхеми пам’яті тощо);

– обмін даними між мікроконтролерами (якщо функції розділені між мікроконтролерами і необхідно узгоджувати їх функціонування);

– внутрішньосхемне програмування мікроконтролера (режим послідовного програмування, що дозволяє записувати коди команд до пам’яті до контролера, якій знаходиться на платі).

SPI не розрахований для передачі інформації на великі відстані. При з’єднанні по цьому інтерфейсу один з пристроїв є головним (ведучим, master), а інших – підлеглим (веденим, slave). Передача даних здійснюється синхронно,

тобто вона узгоджується із тактовим сигналом мікропроцесорної системи. Зазвичай такий тактовий сигнал задається ведучим пристроєм.

Модуль SPI використовує наступні цифрові сигнали, для передачі яких використовуються відповідні виводи мікроконтролера:

1) MOSI або SI – вихід даних ведучого, вхід даних веденого (Master Out Slave In), забезпечує передачу даних від master до slave;

2) MISO або SO – вхід даних ведучого, вихід даних веденого (Master In Slave Out), передає дані від slave до master;

3) SCLK або SCK – послідовний тактовий сигнал (Serial Clock), забезпечує передачу тактового сигналу до ведених пристроїв;

4) CS або SS – вибір мікросхеми, вибір веденого (Chip Select, Slave Select).

Ведучий та ведений пристрої обладнані регістрами зсуву. Обмін даними полягає у послідовному зсуві одного біта від ведучого до веденого і навпаки.

Існує три типи підключення за шиною SPI, в кожному з яких беруть участь чотири сигнали.

1) Підключення тільки двох мікросхем: веденої та ведучої (рис. 1.151). Ведучий пристрій передає дані до веденого за лінією MOSI. Ведений одночасно передає вміст власного регістра зсуву до ведучого за лінією MISO. Обмін даними синхронізується сигналом SCLK. Ведений бере участь в обміні тільки за наявності сигналу низького рівня на вході $\overline{SS}$.

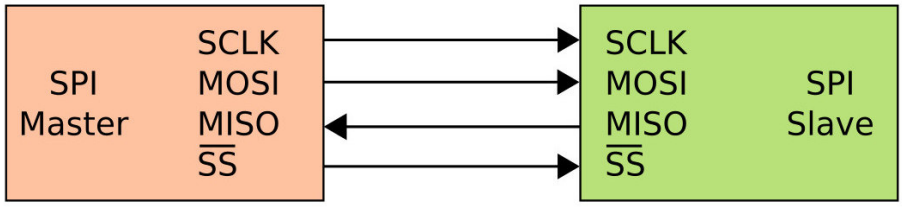


Рис. 1.151. З’єднання двох пристроїв шиною SPI

2) Незалежне (паралельне) підключення декількох ведених мікросхем до ведучої (рис. 1.152). При такому способі підключення всі інформаційні лінії з’єднуються паралельно, проте лінії вибору кристалу $\overline{SS}$ прокладаються від ведучого до кожного з ведених. Ведучий, переводячи відповідний $\overline{SS}$ у низький стан активує обраний ведений та обмінюється з ним даними.

3) Каскадне (послідовне) підключення декількох ведених мікросхем до ведучої (рис. 1.153). Такий спосіб підключення позбавлений недоліку попередньої схеми, оскільки не потрібно прокладати лінію $\overline{SS}$ до кожного веденого пристрою. При каскадному підключенні декілька мікросхем утворюють один великий регістр зсуву. Для цього вихід передачі даних однієї мікросхеми з'єднується з входом прийому даних іншої. За рахунок цього загальне число ліній зв'язку збережене рівним 4.

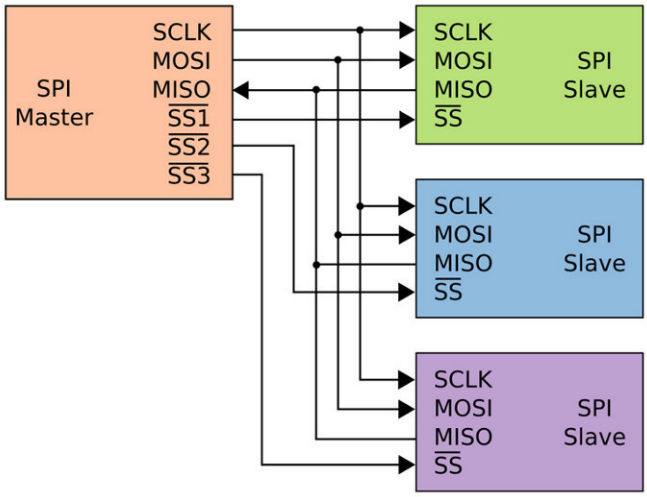


Рис. 1.152. Незалежне підключення декількох ведених пристроїв (slave) до ведучого (master) шиною SPI

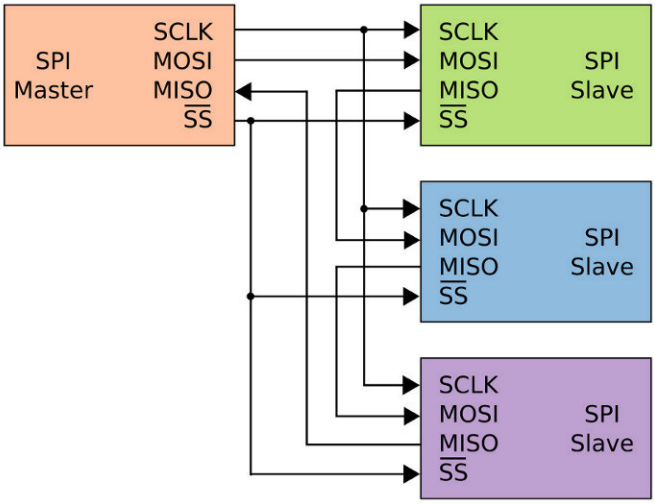


Рис. 1.153. Каскадне підключення декількох ведених пристроїв (slave) до ведучого (master) шиною SPI

1.9.7. Послідовний двопровідний інтерфейс TWI (I^2C)

Модуль двопровідного послідовного інтерфейсу (Two-wire Serial Iterface, TWI) є повним аналогом базової версії інтерфейсу I^2C фірми Philips. Він призначений для обміну даними між мікросхемами, розташованими на одній платі або в одному пристрої автоматизації. TWI дає змогу організувати обмін даними між 128 різних пристроїв за допомогою двонаправленої шини, що складається з двох ліній: лінії тактового сигналу (SCL) та лінії даних (SDA), рис. 1.154.

Шинні формувачі усіх TWI пристроїв виконуються за схемою з відкритим колектором (стоком), що дозволяє активізувати функцію «монтажне AND». Відповідно, низький рівень на лінії встановлюється тоді, коли один або більш пристрої виставляють на лінію сигналу «0», а високий рівень на лінії

встановлюється тоді, коли усі пристрої, підключені до неї, встановлюють свої виходи в третій стан (Z-стан).

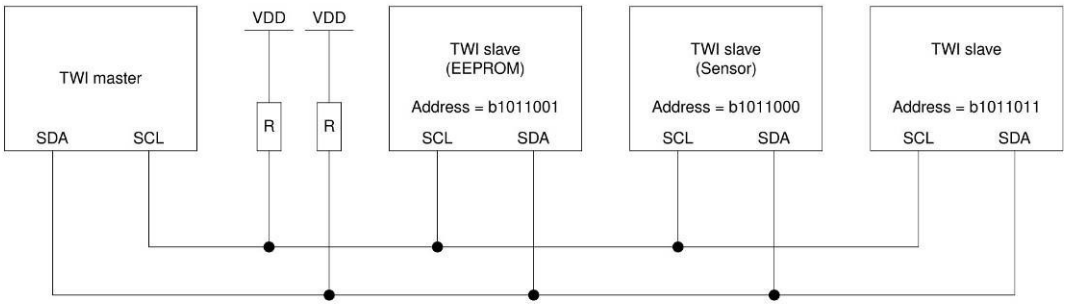


Рис. 1.154. Під’єднання пристроїв до шини TWI

Стан високого імпедансу або Z-стан – такий стан контакту логічної схеми, при якому опір між цим контактом і іншою схемою дуже великий. Фізично реалізується закритим транзистором, що працює в ключовому режимі. Вивід, переведений в Z-стан, поводиться як не підключений до схеми. Зовнішні пристрої, підключені до цього виводу, можуть змінювати напругу на цьому виводу на власний розсуд (у деяких рамках), не впливаючи на роботу схеми. І навпаки – схема не заважає зовнішнім пристроям міняти напругу на контакті. У цифровій електроніці існують поняття «логічна одиниця» (контакт приєднаний до джерела живлення і може видавати в навантаження великий струм, близько сотень міліампер) і «логічний нуль» (контакт приєднаний до «землі», і також витримує високі струми). Але такі виходи не можна об'єднувати: якщо на одному буде 1, а на іншому 0, виникає коротке замикання, що небезпечне вигоранням вихідних транзисторів. Тому, для організації з'єднання типу «шина», було введено третій «стан високого імпедансу», коли додатковий ключ просто відключає вихід і він «повисає в повітрі» – з'єднується з іншою схемою через високий опір (імпеданс) закритого транзистора. Такий вихід не впливає на підключений до нього провід, отже до одного проводу можна підключати декілька виходів. Однак, необхідно стежити, щоб в кожен момент часу тільки один пристрій був активним, а інші у високоімпедансном стані. Маємо з'єднання типу «шина». Z-стан застосовується, коли пристрою доводиться тимчасово відключатися від шини – наприклад, в програматорах, мультиплексорах, багатоточкових інтерфейсах передачі даних JTAG, I2C або USB тощо.

Оскільки шина TWI є послідовною, усі дані передаються по лінії SDA порозрядний. Кожен розряд супроводжується імпульсом на лінії тактового сигналу SCL. Причому сигнал на лінії SDA має бути стабільним весь той час, поки на шині SCL присутній сигнал «1». Єдиним виключенням з цього правила є два особливі стани шини TWI – СТАРТ і СТОП. Стани СТАРТ і СТОП формуються ведучим на початку і в кінці передачі даних відповідно. Між цими станами шина вважається зайнятою і інші ведучі не повинні намагатися

управляти нею.

Якщо ведучий хоче почати передачу нового блоку даних без втрати/відновлення контролю над шиною, він може сформувати стан СТАРТ до формування стану СТОП. Формований таким чином стан називається «Повторний СТАРТ» (ПОВСТАРТ), а шина вважається зайнятою до наступного формування стану СТОП.

1.9.8. Польова шина Profibus

Profibus – шина польового рівня відкритої промислової мережі, що була запропонована компанією Siemens AG для контролерів Simatic. На основі цього були розроблені міжнародні стандарти шини Profibus: IEC 61158 та EN 50170. Ця мережа надзвичайно широко поширена в європейських країнах, особливо у керуванні промисловим обладнанням. Мережа Profibus забезпечує об'єднання різних пристроїв (давачів, виконавчих механізмів, найпростіших релейних захистів, лічильників електроенергії) на польовому рівні. Наприклад, на рис. 1.155 наведено смарт-реле для захисту реле SRW 01 для захисту низьковольтного двигуна, що забезпечує передачу даних за інтерфейсом PROFIBUS DP.

Рис. 1.155. Смарт-реле SRW 01 для захисту низьковольтного двигуна, що обладнане інтерфейсом PROFIBUS DP

Profibus реалізує обмін даними між ведучим та декількома веденими пристроями (протоколи DP та PA), або між декількома ведучими пристроями (протоколи FDL та FMS).

Profibus визначає наступні рівні багаторівневої мережевої моделі (відповідно до ISO 7498).

Рівень 1 – фізичний, визначає особливості передачі інформації на фізичному рівні. Для організації мережі може використовуватися вита пара (рис. 1.156), за допомогою якої реалізується шинна топологія, що відповідає стандарту RS-485. Також може використовуватися оптична мережа, що

будується на основі оптико-волоконного кабелю. Може використовуватися інфрачервона мережа.

а)

б)

Рис. 1.156. Пристрої, з’єднані за PROFIBUS DP (фіолетові проводи), *а*, та провід для виконання польової шини PROFIBUS DP, *б*

Рівень 2 – канальний, що визначає протокол доступу до шини. Для всіх версій Profibus визначений єдиний протокол FDL (Field bus Data Link), що реалізується на 2 рівні моделі OSI. Він реалізує процедуру доступу з використанням маркера (token). Мережа Profibus включає ведучі (master) та ведені (slave) станції. Право ведучої станції контролювати шину (тобто передавати повідомлення без віддалених запитів) визначається наявністю маркера. Ведена станція тільки розпізнає одержані повідомлення та передавати дані у відповідь на запити. Маркер передається між ведучими пристроями. За наявності лише одного ведучого, він постійно володіє маркером.

Рівень 3 – відповідає за прикладні функції.

На одних і тих самих каналах зв’язку мережі Profibus можливо одночасно використовувати декілька протоколів передачі даних: Profibus DP, Profibus PA, Profibus FMS.

Profibus DP (Decentralized Peripheral – розподілена периферія) – протокол, що забезпечує швидкісний обмін даними між ведучими DP-пристроями та пристроями розподіленого введення-виведення (веденими DP-пристроями), рис. 1.157. Для протокола характерний найменший час реакцій та висока стійкість до електромагнітних завад. Ця версія призначена для зв'язку між автоматизованими системами управління та розподіленою периферією.

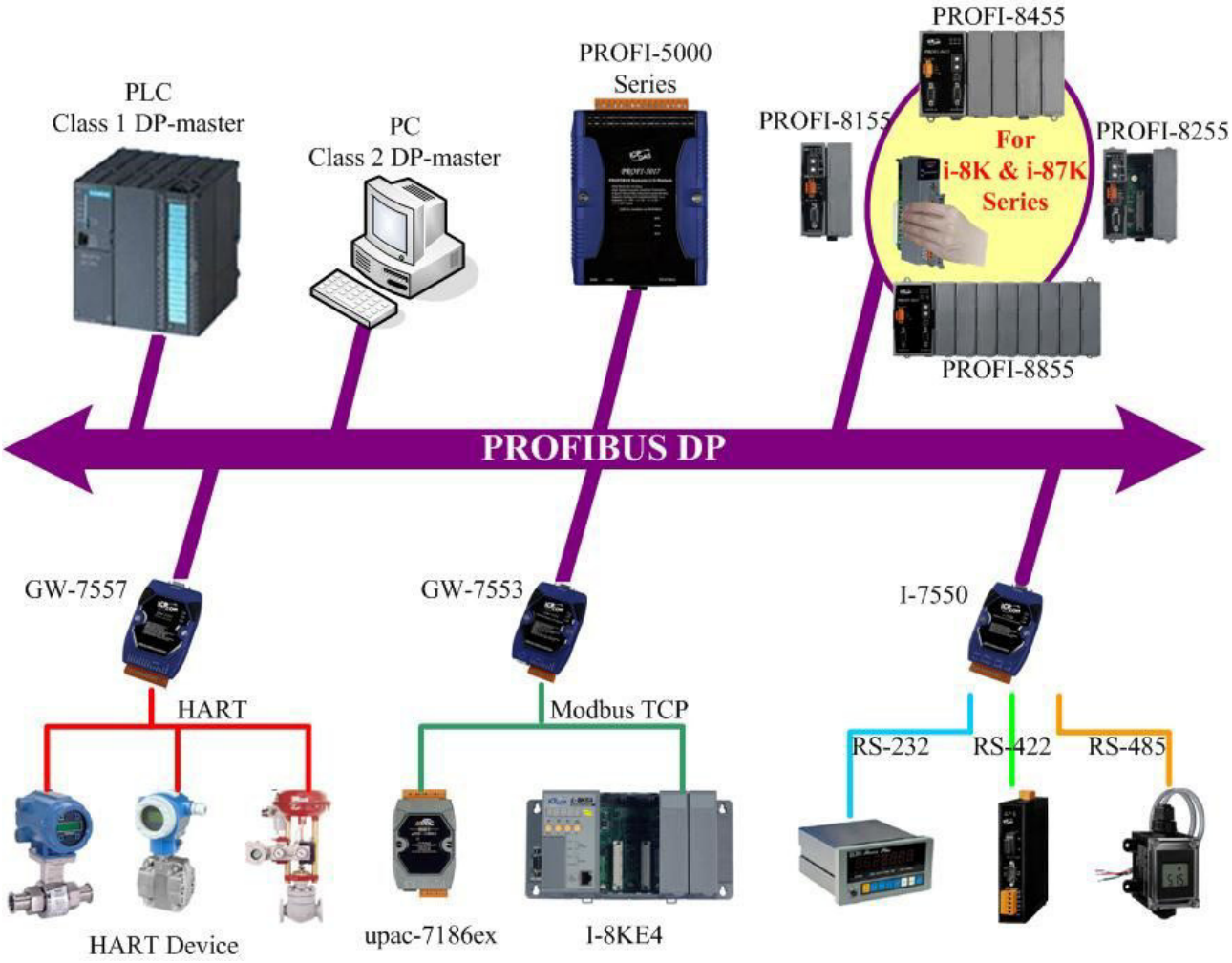


Рис. 1.155. Підключення пристроїв до шини PROFIBUS DP

Profibus PA (Process Automation – автоматизація процесу) – протокол обміну даними з обладнанням, що розташований у звичайних або вибухонебезпечних зонах.

Profibus FMS (Fieldbus Message Specification – специфікація повідомлень польового рівня) – універсальний протокол для обміну даними між інтелектуальними мережевими пристроями на польовому рівні. Швидкість до 12 Мбіт/с.

1.9.9. Мережа Profinet

Profinet (Process Field Network – мережа польового рівня) – відкритий промисловий стандарт обміну даними, що використовує TCP/IP та IT-стандарти та забезпечує режим реального часу Ethernet. Мережа Profinet має модульну

структуру. Існує два види додатків: PROFINET CBA і PROFINET IO. PROFINET CBA використовується для компонентів на основі зв'язку через TCP/IP, а PROFINET IO – для обміну даними в режимі реального часу. PROFINET IO була розроблена для зв'язку реального часу (RT) або ізохронного реального часу (IRT) с децентралізованою периферією. Позначення RT та IRT лише характеризують властивості часу для передачі даних в PROFINET IO, рис. 1.156.

Для реалізації функцій Profinet визначені три види протоколу різного рівня:

TCP/IP для PROFINET CBA, що забезпечує введення в експлуатацію обладнання з часом реакції від 100 мс.

RT (Real-Time) протокол реального часу для PROFINET CBA та PROFINET IO з часом циклу до 1 мс

IRT (Isochronous Real-Time) для додатків PROFINET IO в привідних системах з часом циклу до 1 мс

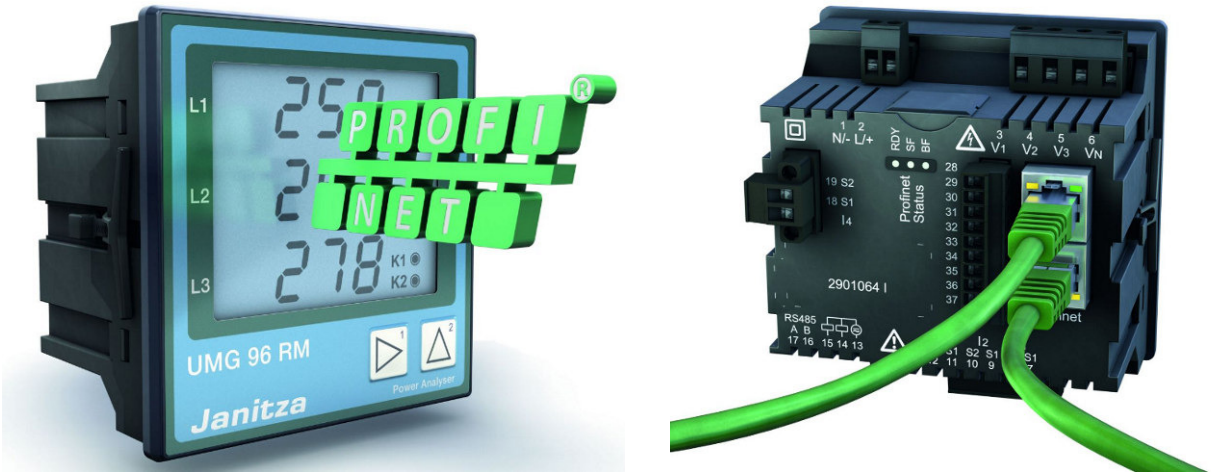


Рис. 1.156. Вимірювач параметрів електромережі UMG 96RM-PN з інтерфейсом Profinet

1.9.10. Питання для самоперевірки

1. Проаналізуйте відмінності між паралельним та послідовним інтерфейсом.
2. Яким чином здійснюється послідовна передача даних?
3. Проаналізуйте формат асинхронної передачі даних.
4. Охарактеризуйте інтерфейс RS-232.
5. Охарактеризуйте інтерфейс RS-485.
6. Порівняйте RS-232 та RS-485.
7. Проаналізуйте призначення та особливості функціонування UART (USART).
8. Яким чином здійснюється передача даних за SPI?
9. Охарактеризуйте послідовний двопровідний інтерфейс TWI (I2C).
10. Розкрийте призначення польової шини Profibus.
11. Охарактеризуйте мережу Profinet.

РОЗДІЛ 2. ВИМІРЮВАЛЬНІ ОРГАНИ ЦИФРОВОГО РЕЛЕЙНОГО ЗАХИСТУ

2.1. Функціонування вимірювальних органів цифрового релейного захисту

Структура цифрових вимірювальних органів. Попереднє оброблення аналогових сигналів. Векторне відображення дискретизованих синусоїдних сигналів.

2.1.1. Структура цифрових вимірювальних органів

Цифровий релейний захист – мікропроцесорні пристрої, що забезпечують захист електромереж та електротехнічних об’єктів від аварійних режимів. На відміну від захистів на базі електромеханічних реле, де принцип захисту реалізується типами застосованих реле та способом їх з’єднання, алгоритм роботи цифрового захисту визначається програмно, рис. 2.1. До переваг цифрового релейного захисту відносяться: малі габарити та енергоспоживання; надзвичайно розвинені функції; можливість реалізовувати захисні алгоритми будь-якої складності; можливість працювати у складі телекомунікаційних промислових систем. Недоліками є необхідність резервування, оскільки вихід з ладу одного терміналу може призвести до серйозних наслідків. На сьогодні цифрові релейні захисти використовуються надзвичайно широко.

Рис. 2.1. Термінал цифрового релейного захисту SIPROTEC 5

Одним з основних компонентів цифрового релейного захисту є цифрові вимірювальні органи – це органи, які здійснюють цифрове оброблення сигналів за заданим алгоритмом. Такі органи є мікропроцесорними пристроями, які реалізують алгоритми аналізу стану електроенергетичного об'єкта, що захищається. Виявлення аварійних та ненормальних станів здійснюється за дискретними значеннями виміряних параметрів стану (струмів та напруг). Тому в програмному забезпеченні мікропроцесорних засобів релейного захисту широко використовуються методи цифрового оброблення сигналів. Алгоритми роботи різних вимірювальних органів у складі релейного захисту реалізуються підпрограмами, тому такі цифрові органи в деякому сенсі є віртуальними.

Цифровий вимірювальний орган, рис. 2.2, визначає стан об'єкта, який захищається, за миттєвими значеннями напруг u та струмів i. Над аналоговими сигналами, що відповідають цим величинам, здійснюється лінійне перетворення формувачем аналогових сигналів ФАС, після чого виконується операція аналого-цифрового перетворення за допомогою АЦП. Числові значення, одержані через рівні проміжки часу (дискретизовані за часом), надходять до мікропроцесора МП, де обробляються відповідно до алгоритму захисту. Алгоритм роботи вимірювального органу зберігається у вигляді програмних кодів у пам'яті П. Крім того, контролюється стан дискретних давачів (блок-контакти високовольтного вимикача, кінцеві вимикачі тощо). Сигнали $x_1,\ldots,x_k$ від останніх, після нормування величини перетворювачем Д1, також надходить до МК. Вимірювальний орган видає дискретні сигнали $y_1,\ldots,y_n$, що підсилюються блоком Д2.

Цифрові вимірювальні органи є невід'ємною частиною цифрового релейного захисту, рис. 2.3.

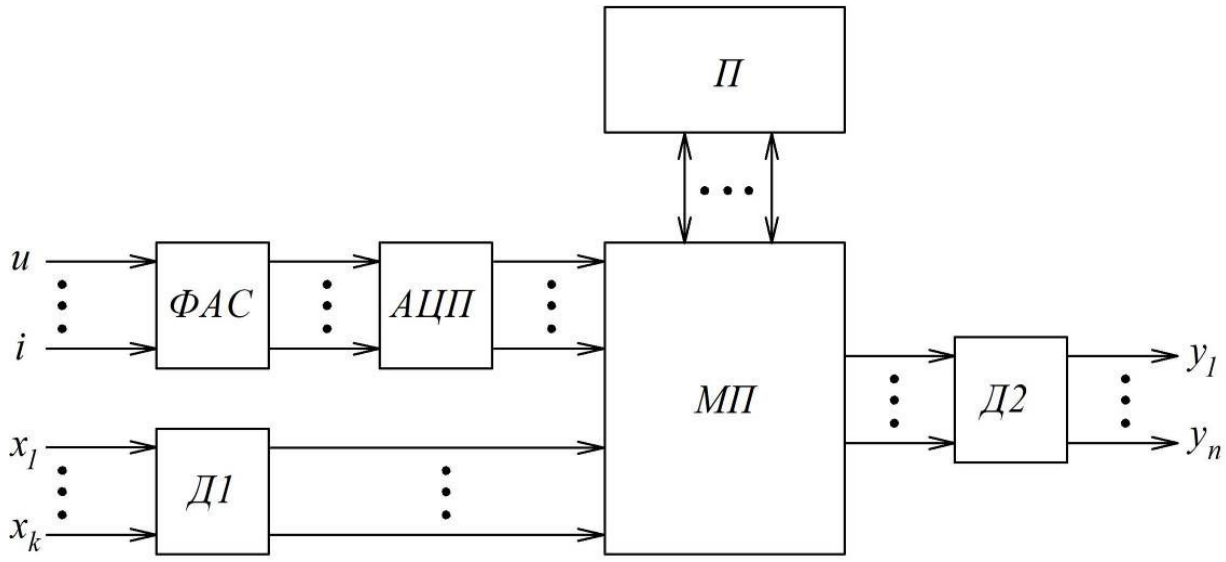


Рис. 2.2. Структурна схема цифрового вимірювального органу:
ФАС – формувач аналогових сигналів;
АЦП – аналого-цифровий перетворювач;
МП – мікропроцесор; П – пам'ять;
Д1 – перетворювач вхідних дискретних сигналів;
Д2 – підсилювач вихідних сигналів

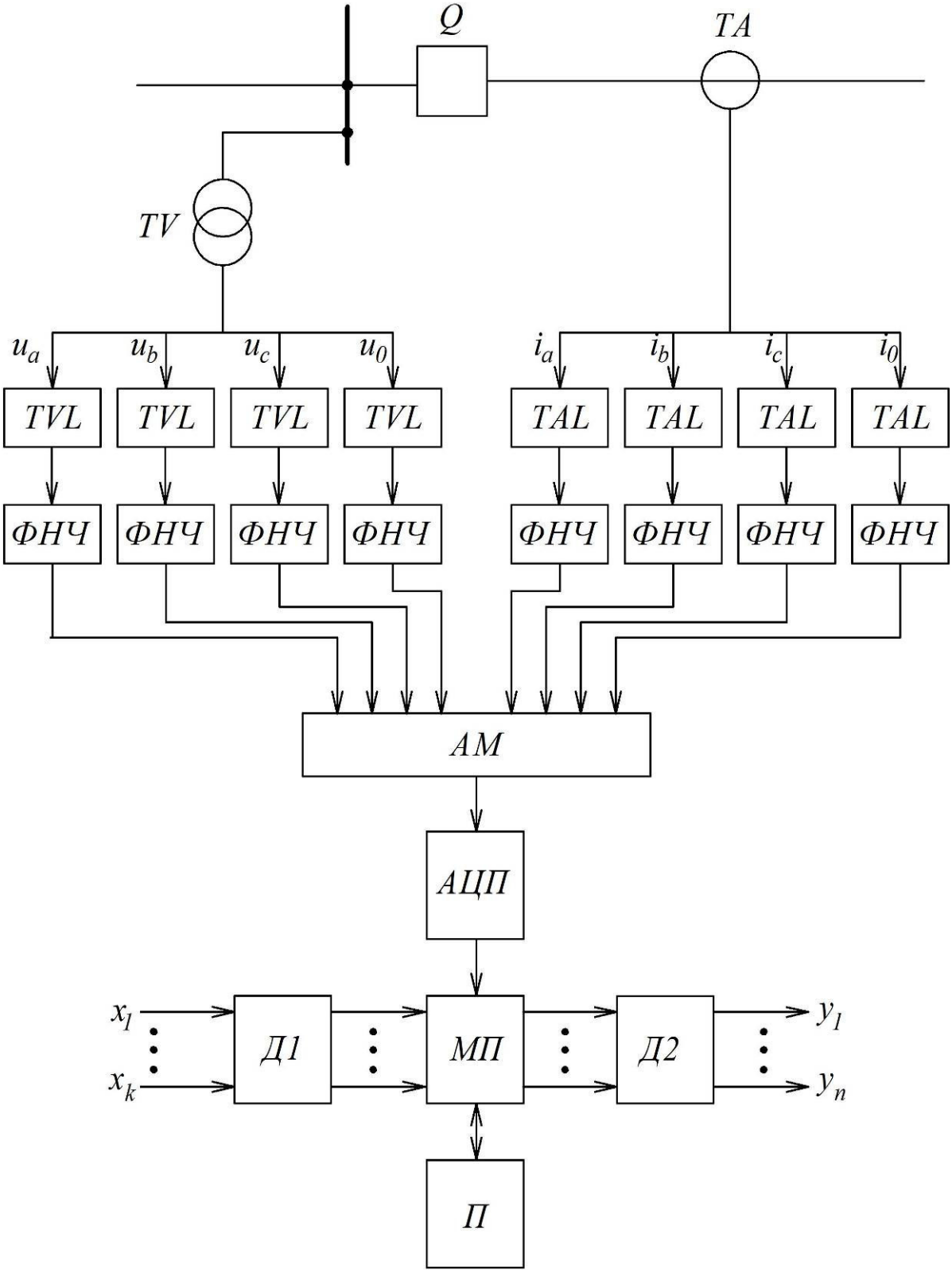


Рис. 2.3. Загальна структура цифрового релейного захисту:
Q – високовольтний вимикач;
TV, TA – вимірювальні трансформатори напруги та струму;
TVL, TAL – проміжні трансформатори напруги та струму;
ФНЧ – фільтр низьких частот;
АМ – аналоговий мультиплексор

Для контролю напруг електромережі цифровий релейний захист, рис. 2.2, використовує вимірювальний трансформатор напруги TV (може використовуватися декілька трансформаторів). Миттєві значення фазних напруг (u_a, u_b, u_c) та напруги нульової послідовності u_0 додатково перетворюються проміжними трансформаторами напруги TVL, що встановлені всередині блока релейного захисту. Вихідні сигнали TVL пропускають через фільт низьких частот ФНЧ, що дозволяє послабити високочастотні завади у корисному сигналі. Для вимірювання струмів в силовій мережі використовуються вимірювальні трансформатори струму TA (на схемі умовно показано тільки один трансформатор). Вторинні струми надходять до проміжних трансформаторів TAL, вихідні сигнали яких також пропускають через ФНЧ. Перетворення аналогових сигналів в цифрову форму здійснюється модулем АЦП. В схемі на рис. 2.3 використовується один АЦП, який послідовно вимірює кожен з аналогових сигналів, що забезпечується аналоговим мультиплексором AM.

2.1.2. Попереднє оброблення аналогових сигналів

До цифрових вимірювальних органів, в першу чергу, висувається вимога високої точності. Одним з основних джерел похибок вимірювань є процедура аналого-цифрового перетворення. Коректний вибір періоду T дискретизації аналогового сигналу за часом дозволяє одержати максимум інформації про стан електромережі. При виборі значення періоду необхідно брати до уваги: частотний спектр аналогового сигналу, що вимірюється; часові параметри АЦП; можливості мікропроцесора з оброблення вимірювальної інформації.

Вибір занадто великого періоду дискретизації може призвести до втрати інформації про вищі гармонійні або перехідні складові у сигналі. Наприклад, інформація, що несуть дискретні відліки функції $u(nT)$, є більше спотвореними по відношенню до функції $u(t)$ при періоді дискретизації T_1 (рис. 2.4, а) порівняно з періодом дискретизації $T_2 < T_1$ (рис. 2.4, б).

Кожен аналоговий сигнал характеризується частотним спектром, що визначає відносний рівень різних частотних складових у його складі. Зазвичай в сигналах з давачів струму та напруги можна виділити частоту зрізу $\omega_з = 2\pi f_з$. Складові з частотами, що перевищують частоту зрізу, мають практично нульову амплітуду.

Відповідно до теореми відліків Котельникова–Шеннона, якщо $f_з = \omega_з / 2\pi$ – верхня границя суттєвого діапазону частот аналогового сигналу, що обробляється (складові з частотами $f > f_з$ можна знехтувати), $f_d = 1/T$ – частота дискретизації, то для уникнення суттєвих спотворень при дискретизації сигналу частота дискретизації повинена бути принаймні в 2 рази більша, ніж частота найвищої гармонічної складової в аналоговому сигналі:

$$f_d \geq 2 f_з .$$

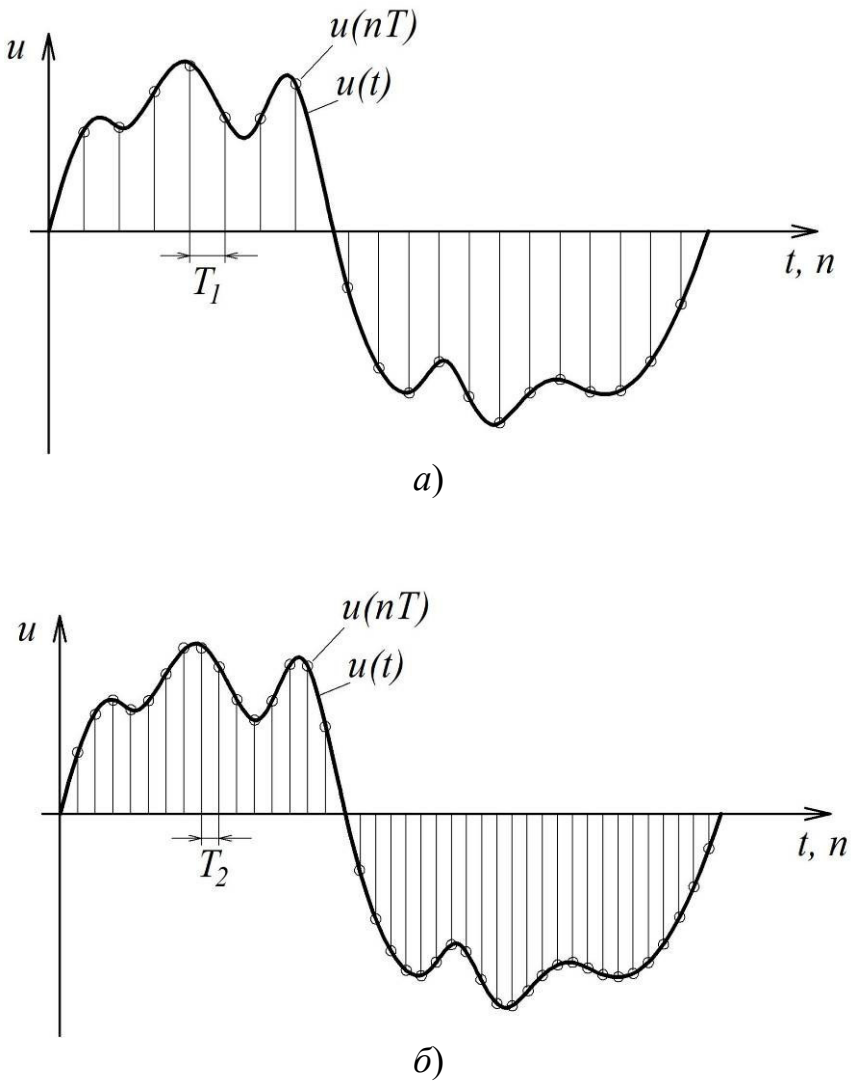


Рис. 2.4. Дискретизація аналогового сигналу за часом при періодах дискретизації T_1 (*а*) та T_2 (*б*), причому $T_1 > T_2$

2.1.3. Векторне відображення дискретизованих синусоїдних сигналів

Релейний захист призначено для використання у складі об'єктів електроенергетики, які генерують, трансформують, передають та забезпечують утилізацію електричної енергії у вигляді синусоїдного змінного струму.

Для опису синусоїдних величин, проведення розрахунків коротких замикань, вибору уставок і характеристик релейного захисту в енергетиці широко використовується векторний метод. Він передбачає представлення синусоїдних сигналів, які відповідають електричним величинам, векторами на комплексній площині. Зміна синусоїдної величини в часі відповідає обертанню вектора. Наприклад, синусоїдна величина $x(t) = A\sin(\omega_0 t + \varphi)$ може бути подана вектором $X(t) = Ae^{j(\omega_0 t + \varphi)}$, що рівномірно обертається з кутовою частотою ω_0 проти годинникової стрілки. Проекція цього вектора в будь–який момент часу на вісь ординат відповідає миттєвому значенню сигналу $x(t)$. Для зручності розгляду і обчислень вектор, що обертається, гальмують, домножуючи на

$e^{-j\omega_0 t}$. При цьому будь–якому із синусоїдних сигналів $A\sin(\omega_0 t+\varphi)$ відповідає незмінний у часі вектор:

$$A = Ae^{j\varphi} = A\cos\varphi + jA\sin\varphi = A_x + jA_y. \qquad (2.1)$$

Проекція $A_y = A\sin\varphi$ цього вектора на уявну ось дорівнює миттєвому значенню $x(t)$ при $t=0$. За допомогою векторного відображення зручно розглядати амплітудні та фазові співвідношення (параметри A, φ) між синусоїдними сигналами змінного струму, зважаючи на те, що основні операції з цими сигналами (додавання, множення тощо) досить просто описуються такими ж операціями з відповідними векторами.

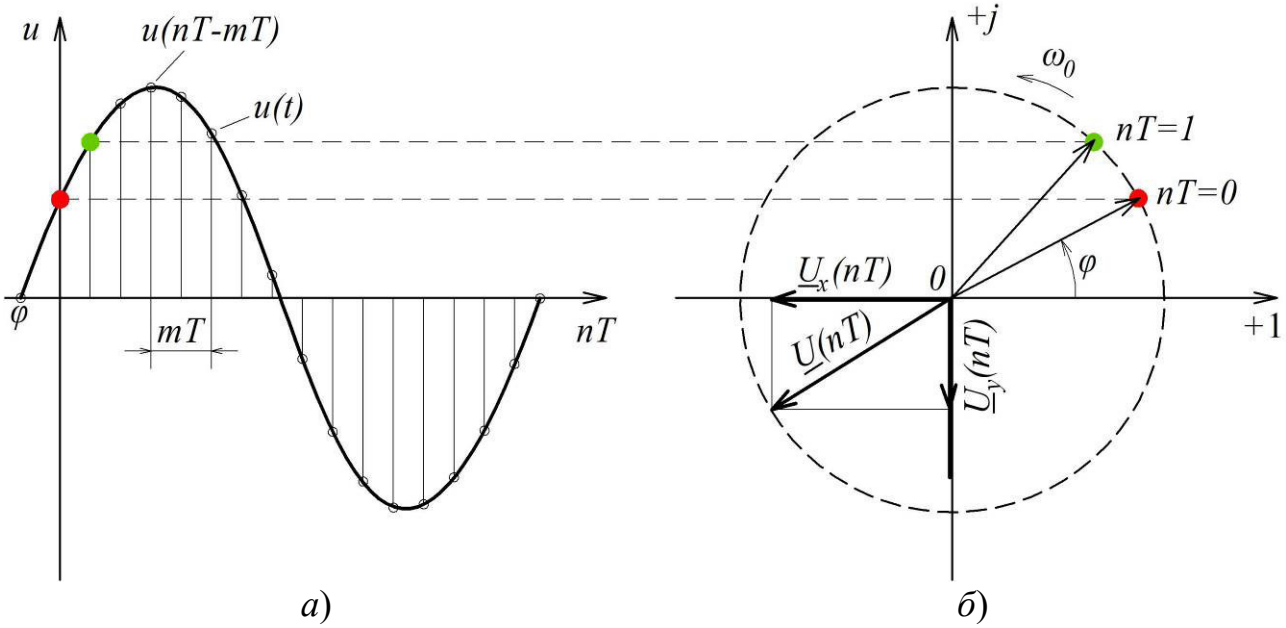
Зазначений метод застосовний і до дискретизованих цифрових сигналів (рис. 2.5), що є цифровими вибірками $u(nT)$, $i(nT)$ аналогових синусоїдних сигналів змінного струму $u(t), i(t)$, відповідно:

$$u(nT) = U\sin(\omega_0 nT+\varphi_u); \quad i(nT) = I\sin(\omega_0 nT+\varphi_i). \qquad (2.2)$$

Таким чином, дискретизовані синусоїдні сигнали можуть бути представлені наступними обертовими векторами:

$$\begin{cases} U(nT) = Ue^{j(\omega_0 nT+\varphi_u)} = U_x(nT) + jU_y(nT); \\ I(nT) = Ie^{j(\omega_0 nT+\varphi_i)} = I_x(nT) + jI_y(nT), \end{cases} \qquad (2.3)$$

проекції яких на уявну ось у моменти $t=0,1,2,...,n$ відповідають цифровим вибіркам величин $u(t)$ та $i(t)$.



а) *б*)

Рис. 2.5. Дискретизовані значення синусоїдної величини (*а*) та відповідні положення обертового вектора на комплексній площині (*б*)

При гальмуванні обертових векторів шляхом множення на $e^{-j\omega_0 nT}$ одержуємо вектори, що не залежать від часу: $\underline{U} = U_x + jU_y$, $I = I_x + jI_y$ для довільного дискретного моменту часу $t = 0, T, 2T, ..., nT$.

В реальних умовах через специфічні властивості цифрового перетворення сигналів вимірювані цифрові значення загальмованих векторів можуть також певною мірою залежати від часу. Тому в загальному випадку сигнали змінного струму надалі відображаються векторами:

$$\begin{cases} \underline{U}(nT) = U_x(nT) + jU_y(nT) = U(nT)e^{j\varphi_u(nT)}; \\ \underline{I}(nT) = I_x(nT) + jU_y(nT) = I(nT)e^{j\varphi_i(nT)}. \end{cases} \quad (2.4)$$

Обчислення поточного вектора $E(nT)$, що відповідає дискретизованій величині $e(nT) = E\sin(\omega_0 nT + \varphi)$, пояснює рис. 2.6. При цьому в загальному випадку:

$$\underline{E}(nT) = E(nT)e^{j\varphi(nT)} = E_x(nT) + jE_y(nT),$$

де $E_x(nT)$ і $E_y(nT)$ – поточні ортогональні складові вектора $\underline{E}(nT)$.

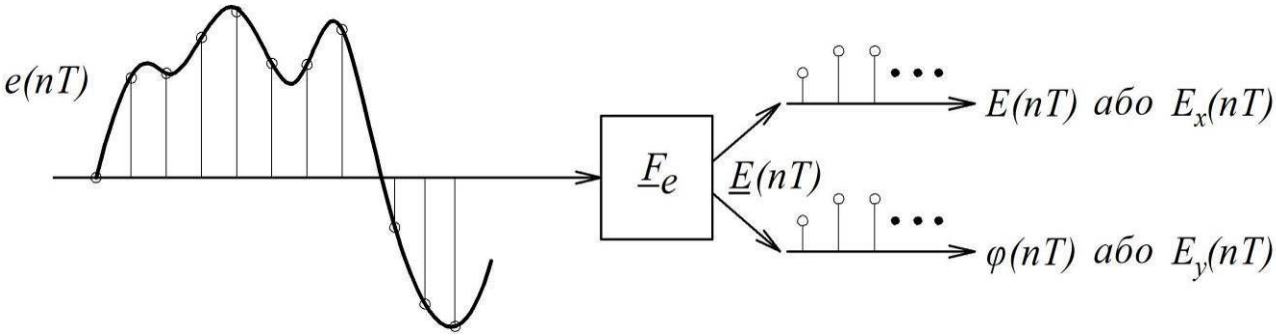


Рис. 2.6. Функціонування алгоритму обчислення вектора $E(nT)$, що відповідає величині $e(nT)$

Векторне перетворення дискретизованих сигналів є основним, але не єдиним можливим способом цифрового вимірювання величин, що використовуються при програмуванні релейний захист. Існують алгоритми перетворення, що безпосередньо використовують миттєві значення струму та напруги.

2.1.4. Питання для самоперевірки

1. Вкажіть переваги цифрового релейного захисту порівняно із захистом на електромеханічній елементній базі.
2. Проаналізуйте обмеження та недоліки цифрового релейного захисту.
3. Поясність призначення цифрового вимірювального органу.
4. Розкрийте структуру цифрового вимірювального органу.
5. Поясність призначення складових цифрового релейного захисту.
6. Яким чином обирається частота дискретизації аналогового сигналу?

7. Які фактори необхідно враховувати при виборі частоти дискретизації аналогового сигналу?

8. Розкрийте принцип представлення синусоїдної величини обертовим вектором.

9. Яким чином обертовий вектор можна загальмувати?

10. Для чого гальмують обертовий вектор?

11. Поясність формування дискретизованого обертового вектора.

2.2. Алгоритми цифрового перетворення сигналів релейного захисту

Визначення алгоритму цифрового перетворення сигналів. Обчислення середніх та діючих значень сигналів. Обчислення векторів на основі миттєвих значень величин та їх похідних. Алгоритм двох вибірок. Алгоритми на основі диференціального рівняння лінії. Алгоритми цифрового вимірювального органу на основі виділення складових ортогональних функцій.

2.2.1. Визначення алгоритму цифрового перетворення сигналів

В аналогових пристроях захисту різних принципів виконання вимірювання параметрів контрольованих синусоїдних величин енергосистеми (амплітуд струмів і напруг, зсуву фаз) здійснюється шляхом впливу аналогових сигналів, що залежать від вхідних струмів і напруг, на фізичну систему (індукційну, електромагнітну, напівпровідникову тощо). За результатом цього впливу оцінюється знаходження контрольованих параметрів сигналів у заданій області. Наприклад, перевищення фактичною величиною струму уставки викликає замикання контактів електромагнітного реле струму, зменшення імпедансу електромережі, що виявляється за входженням робочої точки до характеристики спрацювання, викликає спрацювання реле мінімального опору.

Суттєва особливість цифрових органів захисту – здійснення операцій не над неперервними величинами, а над дискретними значеннями, які відповідають вимірюваній величині у визначені моменти часу nT.

Алгоритм цифрового перетворення сигналів – послідовність операцій над дискретними значеннями виміряних величин, що забезпечує оцінку стану електромережі та виявлення аварійних режимів.

В подальшому при аналізі операцій з цифровими величинами, будуть використовуватися наступні позначення:

$u(t)$ – узагальнений аналоговий сигнал;

$u(nT)$ – дискретні значення сигналу в моменти часу nT, де n=0,1,2,… – номер відліку;

T – період дискретизації (час між двома сусідніми вибірковими значеннями);

f_0 – основна (промислова) частота синусоїдного сигналу в енергосистемі;

$\omega_0 = 2\pi f_0$ – кутова основна частота;

$T_0 = 2\pi / \omega_0$ – період основної частоти;

$N = T_0 / T$ – кількість вибірок за період основний частоти.

Алгоритми цифрового оброблення даних ефективно представляти у вигляді структурних схем, для побудови яких використовуються блоки, що наведені на рис. 2.7.

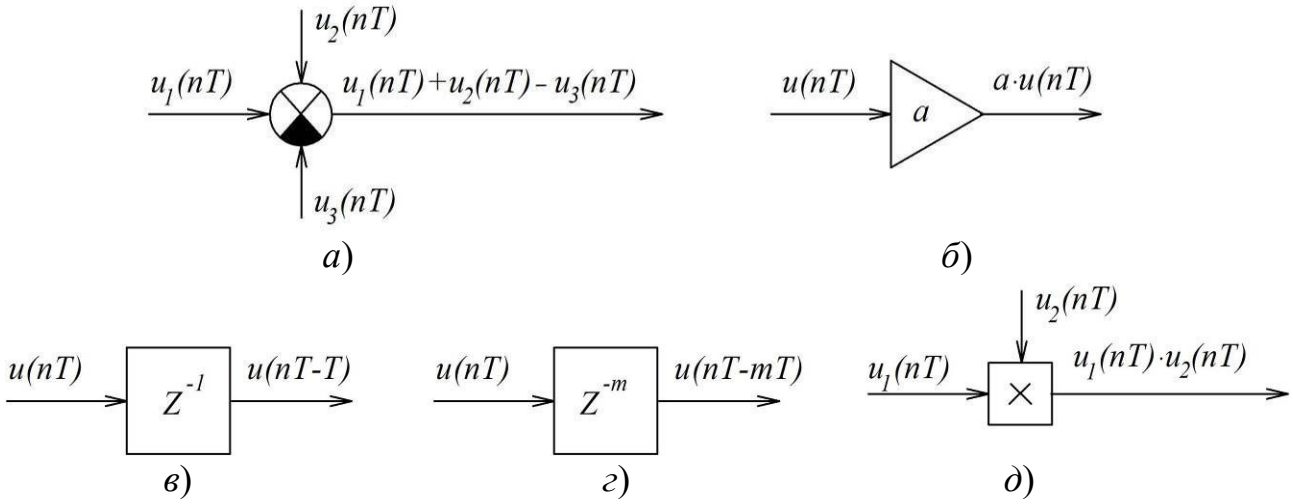


Рис. 2.7. Позначення основних операцій у цифрових колах:
а – алгебраїчне додавання дискретних сигналів; *б* – підсилення дискретного сигналу (множення на константу); *в* – затримка на один період дискретизації;
г – затримка на *m*–періодів дискретизації; *д* – множення дискретних сигналів

2.2.2. Обчислення середніх та діючих значень сигналів

Поточне середнє значення модуля $U_{cp}(t)$ функції $u(t)$ і дійсне значення $U(t)$ за період часу T_0 визначаються відповідно за формулами:

$$U_{cp}(t)=\frac{1}{T_0}\int\limits_{t-T_0}^{t}\left|u(t)\right|dt; \tag{2.5}$$

$$U(t)=\sqrt{\frac{1}{T_0}\int\limits_{t-T_0}^{t}u^2(t)dt}. \tag{2.6}$$

З урахуванням (2.5) та (2.6) та прийнятих вище позначень для дискретизованих цифрових сигналів $u(nT)$ маємо:

$$U_{cp}(nT)=\frac{1}{T_0}\sum_{n-T_0+1}^{n}\left|u(nT)\right|T=\frac{1}{N}\sum_{n-T_0+1}^{n}\left|u(nT)\right|; \tag{2.7}$$

$$U(nT)=\sqrt{\frac{1}{T_0}\sum_{n-T_0+1}^{n}u^2(nT)T}=\sqrt{\frac{1}{N}\sum_{n-T_0+1}^{n}u^2(nT)}. \tag{2.8}$$

2.2.3. Обчислення векторів на основі миттєвих значень величин та їх похідних

Основною багатьох цифрових алгоритмів релейного захисту є визначення амплітуди та фази дискретного сигналу, оскільки ці дані дозволяють одержати дискретні значення вектора. Для синусоїдного сигналу $u(t) = U\sin(\omega_0 t + \varphi)$ амплітуда U і фаза φ при відомій частоті ω_0 можуть бути визначені на основі миттєвого значення $u(t)$ та його похідної $u'(t)$. Похідна від синусоїдного сигналу становить: $u'(t) = \omega_0 U\cos(\omega_0 t + \varphi)$. З рівнянь для $u(t)$ та $u'(t)$, можна визначити шукані невідомі U та φ в довільний момент часу:

$$\begin{cases} U = \sqrt{u^2(t) + \left[\dfrac{u'(t)}{\omega_0}\right]^2}; \\ \omega_0 t + \varphi = \operatorname{arctg}\dfrac{\omega_0 u(t)}{u'(t)}. \end{cases} \tag{2.9}$$

Для ідеального сигналу $u(t) = U\sin(\omega_0 t + \varphi)$ при відомі й похідній $u'(t)$ залежності (2.9) дають можливість визначити у будь-який момент часу t амплітуду та фазу вказаного сигналу. Однак, на практиці, сигнали включають перешкоди, похідна може бути оцінена з певною похибкою, що знижує ефективність безпосереднього застосування залежностей (2.9). Для підвищення точності вимірювання можливо використовувати усереднені значення амплітуд та фаз, визначені для декількох моментів часу.

Припустимо, що розв'язуючи систему рівнянь (2.9), одержані значення U і φ, які визначають параметри вектора на комплексній площині, що обертається з кутовою частотою ω_0, має амплітуду U і початковий кут φ. Дійсна частина вектора становить $u'(t)/\omega_0$, уявна частина дорівнює $u(t)$, а саме:

$$\underline{U}(t) = Ue^{j\varphi}e^{j\omega_0 t} = \underline{U}e^{j\omega_0 t} = \frac{u'(t)}{\omega_0} + ju(t). \tag{2.10}$$

Вираз (2.10) для неперервних величин є основою для алгоритмів оцінювання параметрів дискретного вектор $\underline{U}(nT) = U(nT)e^{j\varphi(\mathrm{nT})}$, який описує дискретизований сигнал $u(nT) = U\sin(\omega_0 nT + \varphi)$.

Припустимо, що в момент t виміряне значення неперервного сигналу становило $u(t) = u(nT)$. Похідна першого порядку може бути визначена за методом Ейлера з урахуванням попереднього виміряного значення $u(nT - T)$, а саме:

$$u'(t) = \frac{1}{T}[u(nT) - u(nT - T)]. \tag{2.11}$$

У такому випадку, враховуючи, що $\omega_0 T = \omega_0 T_0 / N = 2\pi / N$, підставляючи значення (2.11) до (2.10), отримаємо:

$$\underline{U}(nT) = \left(\frac{N}{2\pi} + j\right)u(nT) - \frac{N}{2\pi}u(nT - T). \tag{2.12}$$

Вираз (2.12) визначає алгоритм обчислення вектора $\underline{U}(nT) = U_x(nT) + U_y(nT)$ за поточним $u(nT)$ та попереднім $u(nT-T)$ дискретними значеннями, що були виміряні АЦП. Цьому методу притаманні недоліки методу Ейлера. Основне джерело похибки – обчислення похідної тільки на основі двох точок.

Позначимо у виразі (2.12) коефіцієнти:

$$\underline{a}_0 = \frac{N}{2\pi} + j\,;\ \underline{a}_1 = -\frac{N}{2\pi},$$

тоді залежність (2.12) може бути подана у вигляді структурної схеми, рис. 2.8.

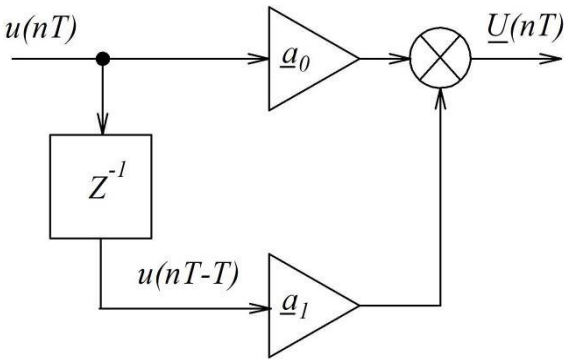


Рис. 2.8. Структурна схема алгоритму визначення дискретних значень комплексного вектора синусоїдної величини за поточним та попередніми виміряними дискретними значеннями (алгоритм першого порядку)

Підвищення точності алгоритму можливе шляхом використання поточної та двох попередніх точок. При цьому приймається:

$$u(t) = u(nT-T)\,;\quad u'(t) = \frac{1}{2T}[u(nT) - u(nT-2T)]. \qquad (2.13)$$

Враховуючи (2.13) у (2.10), маємо залежність для обчислення вектора відповідно до алгоритму другого порядку:

$$\underline{U}(nT) = \frac{N}{4\pi}u(nT) + ju(nT-T) - \frac{N}{4\pi}(nT-2T). \qquad (2.14)$$

Структурна схема алгоритму, що відповідає (2.14), наведена на рис. 2.9, причому позначені наступні коефіцієнти:

$$\underline{a}_0 = \frac{N}{4\pi};\quad \underline{a}_1 = j\,;\quad \underline{a}_2 = -\frac{N}{4\pi}.$$

Безпосередньо реалізувати залежність (2.14), що має комплексні коефіцієнти, у програмі для мікроконтролера складно, оскільки переважна більшість контролерів не підтримують безпосередні операції з комплексними числами. Тому для коефіцієнтів залежності можна виділити дійсні та уявні частини коефіцієнтів:

$$a_{0x} = \frac{N}{4\pi};\qquad a_{0y} = 0\,;$$

$$a_{1x} = 0\,;\qquad a_{1y} = j\,;$$

$$a_{2x} = -\frac{N}{4\pi};\qquad a_{2y} = 0\,;$$

Структурна схема алгоритму другого порядку, що передбачає виконання операцій тільки з дійсними числами для обчислення дійсної та уявної частин вектора, наведена на рис. 2.10.

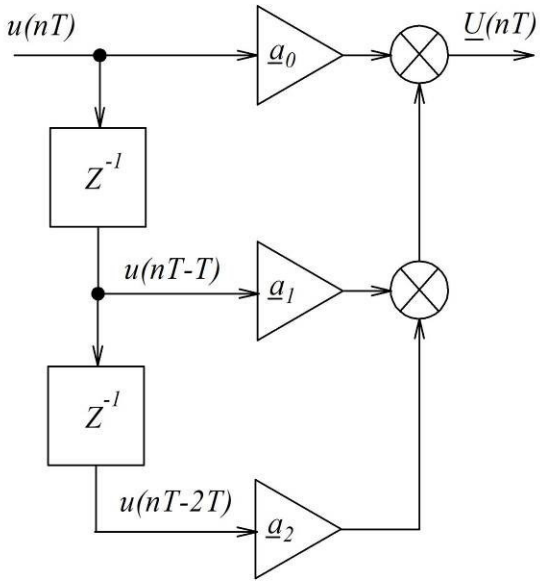


Рис. 2.9. Структурна схема алгоритму другого порядку визначення дискретних значень комплексного вектора синусоїдної величини за умови використання комплексних коефіцієнтів

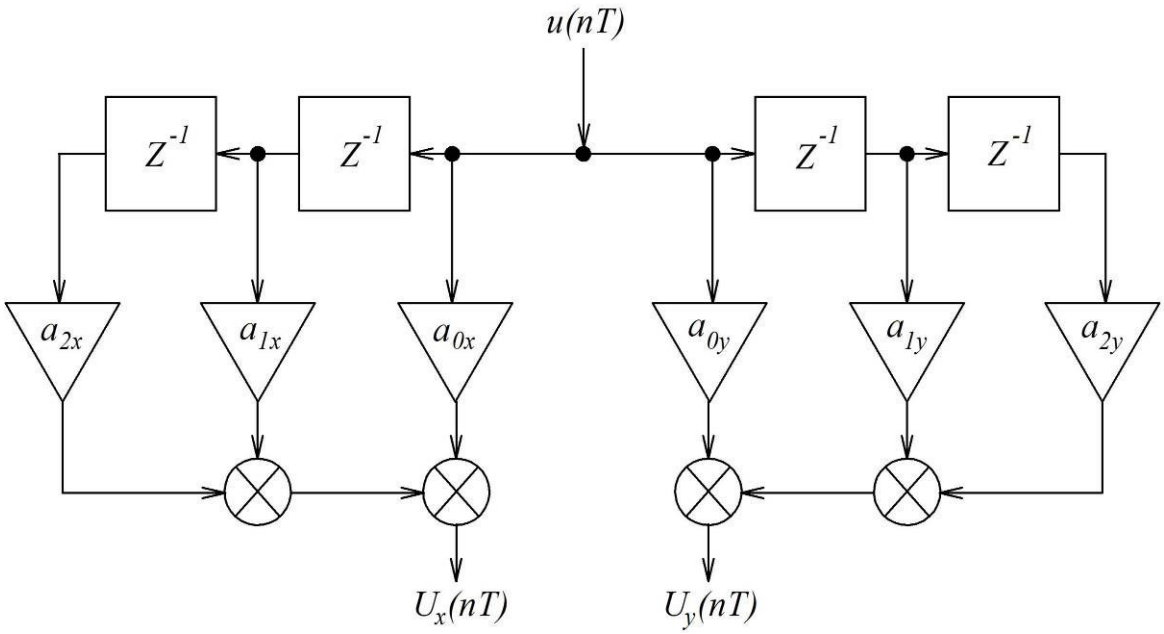


Рис. 2.10. Структурна схема алгоритму другого порядку визначення дискретних значень комплексного вектора синусоїдної величини при використанні дійсних коефіцієнтів (роздільне обчислення дійсної та уявної проекцій вектора)

2.2.4. Алгоритм двох вибірок

Основний недолік алгоритмів, що базуються на визначенні похідних, полягає у наявності суттєвих похибок навість на промисловій частоті. Підвищити точність вимірювання обертового вектора $\underline{U}(t) = Ue^{j(\varphi_0+\varphi)}$ можливо з використанням алгоритму, що базується на двох вибірках $u(nT - mT)$, $u(nT)$ синусоїдного сигналу відомої частоти ω_0, взятих через час mT, рис. 2.5. Для таких вибірок шукані амплітуда та фаза визначаються спільним розв'язанням двох рівнянь, складених для цих вибірок:

$$\begin{cases} u(nT - mT) = U\sin(\omega_0 nT + \varphi - \omega_0 mT); \\ u(nT) = U\sin(\omega_0 nT + \varphi). \end{cases} \tag{2.15}$$

З (2.15) отримаємо значення амплітуди U вимірюваного сигналу може бути одержано за відомими двома вибірками:

$$U = \frac{1}{\left|\sin\omega_0 nT\right|}\sqrt{\left[u(nT - mT)\right]^2 - 2\left[u(nT - mT)\right]\left[u(nT)\right]\cos\omega_0 mT + \left[u(nT)\right]^2}.$$

Цифрове перетворення, що забезпечує отримання вектора за двома вибірками синусоїдної величини $U\sin(\omega_0 t + \varphi)$, взятим через m циклів, має вигляд:

$$\underline{U}(nT) = \frac{1}{\left|\sin\omega_0 mT\right|}\left[u(nT)e^{j\omega_0 mT} - u(nT - mT)\right] = U_x(nT) + jU_y(nT). \tag{2.16}$$

Для обчислення амплітуди вектора $\underline{U}(nT)$ та фазового зсуву між векторами $\underline{U}_1(nT)$ та $\underline{U}_2(nT)$, що відповідають синусоїдним сигналам частоти ω_0, необхідно виконати перетворення:

$$U(nT) = \sqrt{\left[U_x(nT)\right]^2 + \left[U_y(nT)\right]^2};$$

$$\Delta\varphi = \operatorname{arctg}\frac{U_{2y}}{U_{2x}} - \operatorname{arctg}\frac{U_{1y}}{U_{1x}}.$$

У найбільш поширеному випадку $\omega_0 mT = \pi/2$, тобто $m = N/4$, з (2.16) маємо $\underline{U}(nT) = ju(nT) - u(nT - NT/4)$, звідки:

$$U^2(nT) = u^2(nT) + u^2(nT - NT/4), \tag{2.17}$$

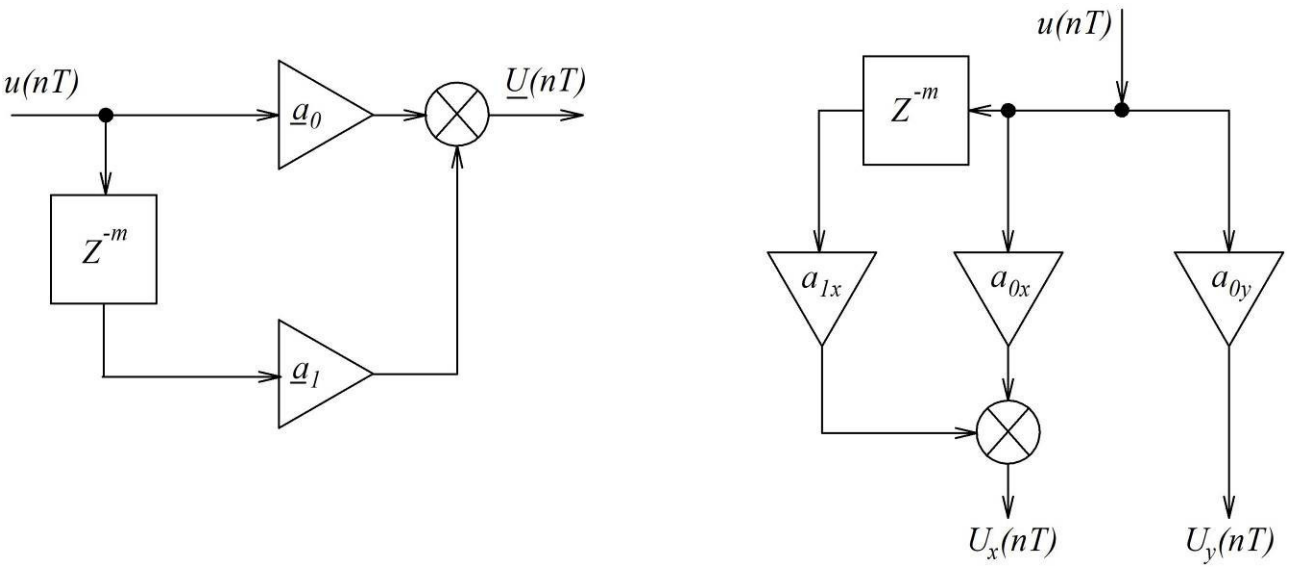
тобто для визначення квадрата амплітуди синусоїдного сигналу достатньо додавання квадратів двох дискретних значень, взятих через $N/4$ циклів.

Структуру алгоритму (2.16) для послідовності комплексних чисел наведено на рис. 2.11, а, для двох послідовностей дійсних чисел – на рис. 2.11, б. При цьому на схемах позначені наступні коефіцієнти:

$$\underline{a}_0 = a_{0x} + ja_{0y} = e^{j\beta}/\sin\beta = \operatorname{ctg}\beta + j;$$

$$\underline{a}_1 = \underline{a}_{1x} = -1/\sin\beta,$$

причому $\beta = \omega_0 mT$.



а) *б*)

Рис. 2.11. Структура алгоритму двох вибірок для послідовностей комплексних (а) та дійсних (б) чисел

Алгоритм двох вибірок також дозволяє розраховувати імпеданс $\underline{Z}$. Позначимо:

$$u(nT-mT)=u_1;$$
$$u(nT)=u_2;$$
$$i(nT-mT)=i_1;$$
$$i(nT)=i_2.$$

Тоді, відповідно до (2.16), можливо одержати:

$$\begin{cases} \underline{Z}=\dfrac{\underline{U}(nT)}{\underline{I}(nT)}=\dfrac{u_2e^{i\beta}-u_1}{i_2e^{i\beta}-i_1}=R+jX; \\ \beta=\omega_0 mT; \\ R=\dfrac{u_1i_1-(u_1i_2+u_2i_1)\cos\beta+u_2i_2}{i_1^2-2i_1i_2\cos\beta+i_2^2}; \\ X=\dfrac{(u_1i_2-u_2i_1)\sin\beta}{i_1^2-2i_1i_2\cos\beta+i_2^2}. \end{cases} \qquad (2.18)$$

При $m=N/4$ $(\beta=\pi/2)$ з останньої системи рівнянь маємо:

$$R=\frac{u_1i_1+u_2i_2}{i_1^2+i_2^2};\ X=\frac{u_1i_2-u_2i_1}{i_1^2+i_2^2}.$$

Алгоритми за виразами (2.12), (2.14), (2.16) є найбільш поширеними прикладами класу алгоритмів, які дають змогу на основі вибірок сигналів $u(t)$, $i(t)$ визначити параметри відповідних векторів напруги $\underline{U}(nT)$ та струму $\underline{I}(nT)$. Розглянуті алгоритми є швидкодіючими (теоретично результат може бути отриманий протягом періоду дискретизації T між двома вибірками). Однак, якщо сигнали відмінні від синусоїдних, ці алгоритми мають низьку

точність. Тому перед їх практичним застосуванням необхідно здійснити цифрова фільтрація сигналів.

2.2.5. Алгоритми на основі диференціального рівняння лінії

Для будь-якого RL-кола, в тому числі і для повітряної лінії, у загальному випадку для будь-якого моменту часу справедливе диференціальне рівняння:

$$u = Ri + \frac{X}{\omega_0} i', \tag{2.19}$$

де u, i – миттєві значення струму та напруги; X / ω_0 – індуктивність лінії; i' – похідна струму.

Якщо u_1, i_1, i_1' та u_2, i_2, i_2' – значення величин на вході цифрового вимірювального органу в будь-які два моменти часу, то, розв'язавши на основі (2.19) систему двох рівнянь, маємо складові імпедансу $\underline{Z} = R + jX$ на вході цифрового вимірювального органу:

$$R = \frac{u_1 i_2' + u_2 i_1'}{i_1 i_2' - i_2 i_1'}; \; \frac{X}{\omega_0} = \frac{i_1 u_2 - i_2 u_1}{i_1 i_2' - i_2 i_1'}. \tag{2.20}$$

Перевагою цього алгоритму є незалежність результатів від форми сигналу, тобто від аперіодичних складових у струмі та напрузі, що виникають при коротких замиканнях. Недоліки – недостатня точність через використання лише двох вибірок сигналу, суттєві похибки за наявності дуги в місці ушкодження, низька завадостійкість.

Для практичного застосування даного алгоритму необхідна попередня цифрова фільтрація сигналів u та i. При цьому операції цифрової фільтрації можуть проводитися безпосередньо зі складовими рівнянь. Зокрема, почленне інтегрування рівняння (2.19) в межах від t_1 до $t_1 + \Delta t$ та від t_2 до $t_2 + \Delta t$ відповідає наступній системі з двома невідомими:

$$\begin{cases} \int\limits_{t_1}^{t_1+\Delta t} u dt = R \int\limits_{t_1}^{t_1+\Delta t} i dt + \dfrac{X}{\omega_0} \int\limits_{t_1}^{t_1+\Delta t} di; \\ \int\limits_{t_2}^{t_2+\Delta t} u dt = R \int\limits_{t_2}^{t_2+\Delta t} i dt + \dfrac{X}{\omega_0} \int\limits_{t_2}^{t_2+\Delta t} di, \end{cases}$$

яка може бути представлена у вигляді:

$$\begin{cases} S_{1u} = RS_{1i} + \dfrac{X}{\omega_0} S_{1i}'; \\ S_{2u} = RS_{2i} + \dfrac{X}{\omega_0} S_{2i}'. \end{cases}$$

Розв'язок цієї системи має вигляд:

$$\begin{cases} R = \dfrac{S_{1u}S'_{2i} - S_{2u}S'_{1i}}{S_{1i}S'_{2i} - S_{2i}S'_{1i}}; \\ \dfrac{X}{\omega_0} = \dfrac{S_{1i}S_{2u} - S_{1u}S_{2i}}{S_{1i}S'_{2i} - S_{2i}S'_{1i}}. \end{cases} \tag{2.21}$$

2.2.6. Алгоритми цифрового вимірювального органу на основі виділення складових ортогональних функцій

Дану групу алгоритмів роботи цифрових вимірювальних органів характеризує наявність фільтруючих властивостей у самих перетвореннях, що полягають у виділенні із сигналів складових взаємно ортогональних функцій (тригонометричних з розкладанням у ряд Фур'є, прямокутних, трапецеїдальних тощо).

Зокрема, для виділення коефіцієнтів a і b першої гармоніки $u_0(t) = a\cos\omega_0 t + b\sin\omega_0 t$ ряду Фур'є для функції $u(t)$, цифровий перетворювач повинен реалізувати до моменту часу t алгоритм:

$$a = \frac{2}{T}\int\limits_{t-T_0}^{t} f(t)\cos\omega_0 t dt;$$

$$b = \frac{2}{T}\int\limits_{t-T}^{t} f(t)\sin\omega_0 t dt,$$

де a і b – ортогональні складові вектора $\underline{F}(t) = F_x(t) + jF_y(t)$, який характеризує синусоїдну складову $u_0(t)$ з частотою ω_0:

$$\underline{F}(t) = j(a - jb) = \frac{2j}{T_0}\int\limits_{t-T_0}^{t} u(t)e^{-j\omega_0 t}dt. \tag{2.22}$$

За наявності на вході цифрового перетворювача синусоїдного сигналу $u_0(t) = U_0\sin(\omega_0 t + \varphi)$ частоти ω_0 із (2.22) після перетворень маємо:

$$\underline{F}(t) = \frac{2j}{T_0}\int\limits_{t-T_0}^{t} U_0\sin(\omega_0 t + \psi_0)e^{-j\omega_0 t}dt = U_0 e^{j\psi_0},$$

тобто синусоїдний сигнал частоти ω_0 алгоритм Фур'є зіставляє із незмінним у часі вектором. Для дискретизованих сигналів з числом вибірок N за період у найпростішому випадку ступінчастої інтерполяції значення функції $u(nT)$ приймаються постійними протягом періоду дискретизації T. У цьому випадку з урахуванням (2.22) на момент часу $t = nT$ при $T_0 / T = N$ маємо:

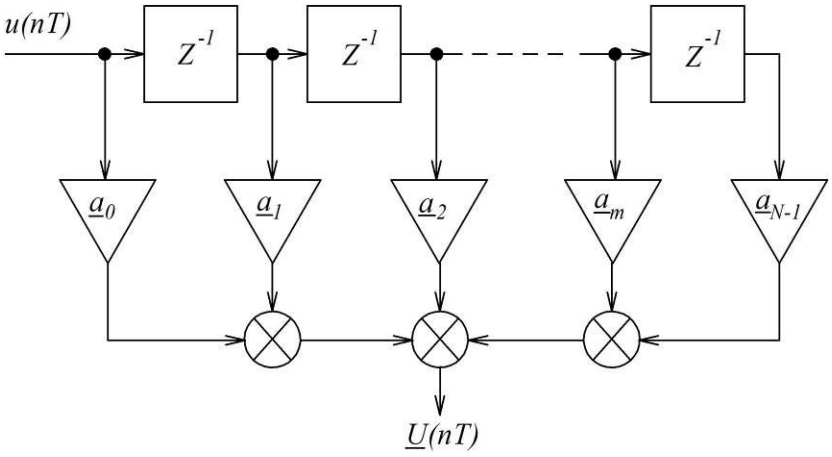
$$\underline{F}(nT) = \frac{2j}{NT}\sum_{n-N+1}^{n} u(nT)e^{-j\omega_0 nT}T = \frac{2j}{N}\sum_{n-N+1}^{n} u(nT)e^{-j\omega_0 nT}. \tag{2.23}$$

Так само, як і при використанні алгоритму Фур'є, при інших можливих розкладаннях на ортогональні функції відбувається фільтрація вхідного сигналу, що забезпечує виділення із вхідних сигналів ортогональних складових, але вже не синусоїдних, а прямокутних або трапецеїдальних.

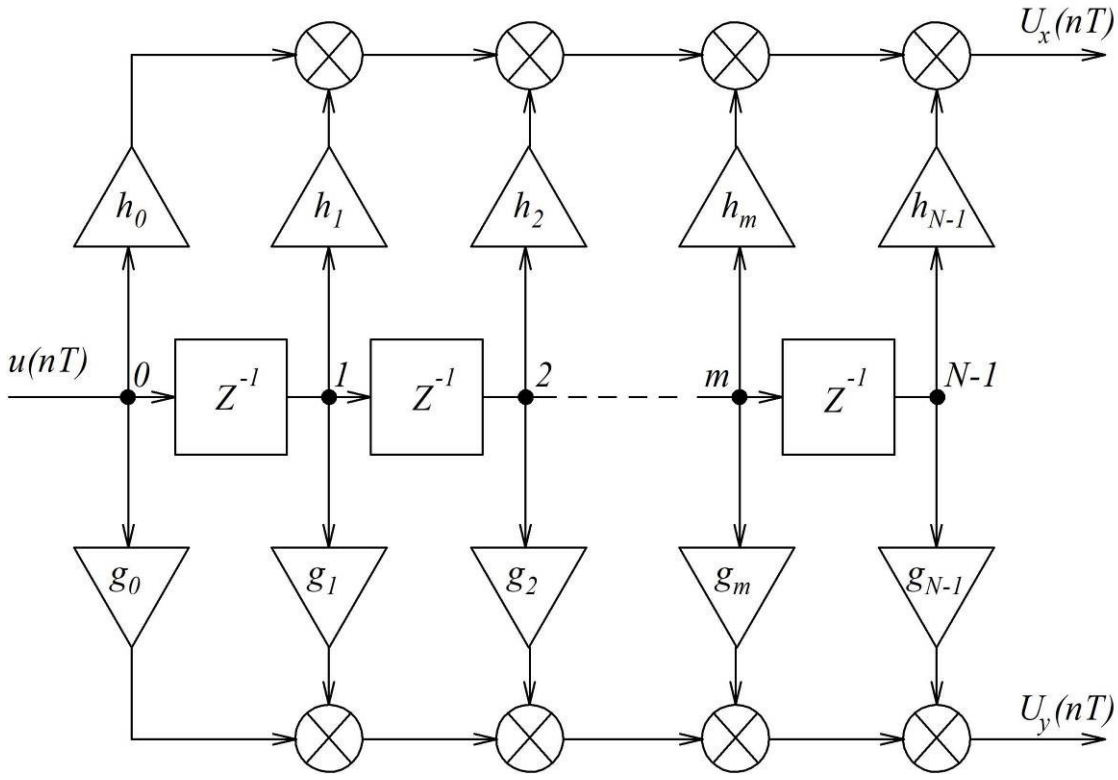
Алгоритм фільтрації забезпечується методом згортки на інтервалі $T_0 = NT$ вхідного сигналу $u(nT)$ з парними та непарними ортогональними функціями.

Структуру алгоритму Фур'є з комплексними коефіцієнтами ілюструє схема на рис. 2.12, а. При цьому комплексні коефіцієнти дорівнюють:

$$\underline{a}_0 = e^{-j\omega_0(N-1)T};\ \underline{a}_1 = e^{-j\omega_0(N-2)T};\ ...;\ \underline{a}_{N-2} = e^{-j\omega_0 T};\ \underline{a}_{N-1} = e^{0} = 1.$$



а)



б)

Рис. 2.12. Структура алгоритму Фур'є з комплексними (*а*) та дійсними (*б*) коефіцієнтами

Зручніше використовувати окремі фільтри для виявлення ортогональних складових вимірюваного вектора $\underline{F}(nT) = F_x(nT) + jF_y(nT)$. Для такого випадку виконується:

$$\underline{F}(nT)=\frac{2}{N}\sum_{n-N+1}^{n}u(nT)\sin\omega_0 nT+j\frac{2}{N}\sum_{n-N+1}^{n}u(nT)\cos\omega_0 nT. \qquad (2.24)$$

Для визначення ортогональних складових вектора можуть використовуватися два окремі алгоритми з дійсними коефіцієнтами h_n, g_n (рис. 2.12, *б*):

$$h_0=\sin\omega_0(n-1)T;\ h_1=\sin\omega_0(n-2)T;\ ...;\ h_{N-1}=0;$$
$$g_0=\cos\omega_0(n-1)T;\ g_1=\cos\omega_0(n-2)T;\ ...;\ g_{N-1}=1.$$

2.2.7. Питання для самоперевірки

1. Дайте визначення періоду дискретизації?
2. Яким чином визначається кількість вибіркових значень за період основної частоти?
3. Які типові блоки використовуються для побудови схем алгоритмів роботи цифрових вимірювальних органів?
4. Яким чином за дискретними відліками миттєвих значень напруги можна обрахувати діюче значення?
5. В чому полягають переваги та недоліки методу обчислення векторів на основі миттєвих значень величин та їх похідних?
6. Розкрийте відомі Вам алгоритми обчислення вектора $\underline{U}(nT)=U_x(nT)+U_y(nT)$ за дискретними значеннями вимірюваної величини.
7. Поясніть схема алгоритму визначення дискретних значень комплексного вектора синусоїдної величини за поточним та попередніми виміряними дискретними значеннями (алгоритм першого порядку).
8. Яким чином можна визначати дискретні значення комплексного вектора з урахуванням двох попередніх точок?
9. В чому полягає алгоритм двох вибірок?
10. Розкрийте принцип, що лежить в основі алгоритмів цифрового вимірювального органу на основі виділення складових ортогональних функцій

2.3. Вимірювальні органи однієї електричної величини

Цифрові вимірювальні органи струму та напруги. Цифрові вимірювальні органи напрямку потужності: характеристики спрацювання; безпосереднє використання вибірок миттєвих значень; цифрові вимірювальні органи симетричних складових.

2.3.1. Цифрові вимірювальні органи струму та напруги

Цифровий вимірювальний орган струму забезпечує виконання двох основних функцій:

– за дискретними значеннями струму обчислення діючого (або амплітудного) значення струму;

– порівняння визначеного діючого значення струму з уставкою.

Для обчислення діючого (або амплітудного) значення струму використовується цифровий вимірювач струму F_i, а для порівняння з уставкою – компаратор K із внутрішньою уставкою спрацювання, рис. 2.13, *а*. Аналогічним чином будується цифровий вимірювальний орган напруги. Аналогами таких органів є електромеханічні реле струму та напруги.

Аналогом реле струму із залежною витримкою часу спрацювання є цифровий орган, що включає блок формування часострумової характеристики БФХ, рис. 2.13, *б*.

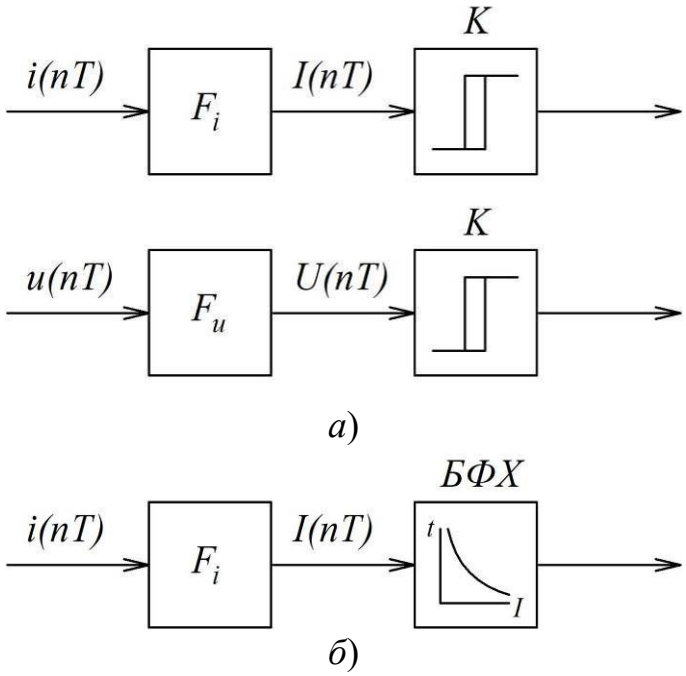


Рис. 2.13. Структура цифрових вимірювальних органів, які реагують на значення струму та напруги

Для захисту генераторів та трансформаторів від несиметричних коротких замикань та перевантаження струмами зворотної послідовності відомо про використання електромеханічного реле струму зворотної послідовності серії РТФ. Також використовуються інші захисти для виявлення струмів (напруг) зворотної або нульової послідовності. Зокрема, за струмами нульової послідовності виявляють однофазні замикання на землю. Пристрої, що виконують аналогічні до електромеханічних функції, можуть бути реалізовані на мікропроцесорній елементній базі програмно. Для обґрунтування алгоритму функціонування цифрового органу на основі вимірювання симетричних складових, проаналізуємо аналітичні залежності, що лежать в основі такого метода.

Будь-яку несиметричну систему, що складається з трьох векторів, наприклад, струмів $\underline{I}_A$, $\underline{I}_B$, $\underline{I}_C$, можна представити як сукупність трьох симетричних систем прямої (індекс 1), зворотної (індекс 2) та нульової (індекс 0) послідовностей. Аналітично це можна представити як:

$$\begin{cases} \underline{I}_A = \underline{I}_{A1} + \underline{I}_{A2} + \underline{I}_{A0}; \\ \underline{I}_B = \underline{I}_{B1} + \underline{I}_{B2} + \underline{I}_{B0} = a^2 \underline{I}_{A1} + a\underline{I}_{A2} + \underline{I}_{A0}; \\ \underline{I}_C = \underline{I}_{C1} + \underline{I}_{C2} + \underline{I}_{C0} = a\underline{I}_{A1} + a^2 \underline{I}_{A2} + \underline{I}_{A0}; \\ a = e^{j120^\circ} = -\dfrac{1}{2} + j\dfrac{\sqrt{3}}{2}. \end{cases} \qquad (2.25)$$

Системи прямої та зворотної послідовностей утворюються трьома векторами з однаковими амплітудами, зрушеними на 120° відповідно. Вектори прямої послідовності мають пряме чергування фаз (вектор $\underline{I}_{B1}$ відстає від вектора $\underline{I}_{A1}$ на 120°). Вектори зворотної послідовності характеризуються зворотним чергуванням фаз (вектор $\underline{I}_{B2}$ випереджає вектор $\underline{I}_{A2}$ на 120°). Між векторами нульової послідовності фазові зсуви відсутні.

Складові окремих послідовностей можна обчислити на основі співвідношень:

$$\begin{cases} \underline{I}_{A1} = \dfrac{1}{3}(\underline{I}_A + a\underline{I}_B + a^2 \underline{I}_C); \\ \underline{I}_{A2} = \dfrac{1}{3}(\underline{I}_A + a^2 \underline{I}_B + a\underline{I}_C); \\ \underline{I}_{A0} = \dfrac{1}{3}(\underline{I}_A + \underline{I}_B + \underline{I}_C). \end{cases}$$

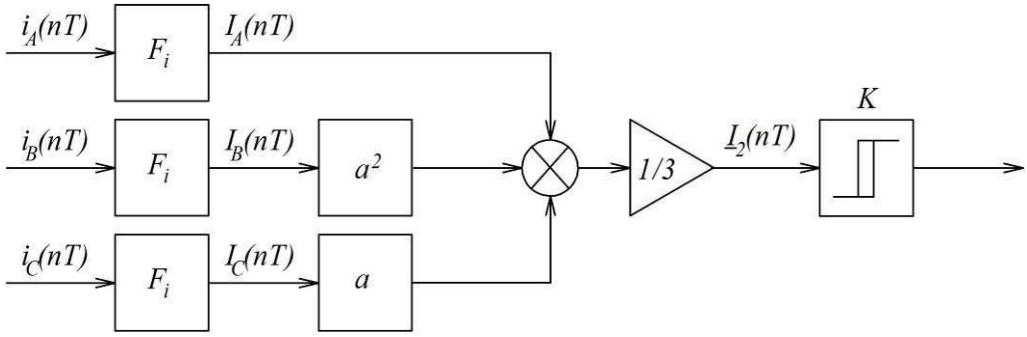
Симетричні складові струму $\underline{I}_1$, $\underline{I}_2$, $\underline{I}_3$ і напруги $\underline{U}_1$, $\underline{U}_2$, $\underline{U}_3$ можуть бути обраховані за дискретними відліками фазних струмів $I_A(nT)$, $I_B(nT)$, $I_C(nT)$ або фазних напруг $U_A(nT)$, $U_B(nT)$, $U_C(nT)$. Зокрема, для фази А (індекс А не вказаний) в основі функціонування цифрового вимірювального органу можна покласти наступні рівняння:

$$\begin{cases} \underline{I}_1(nT) = \dfrac{1}{3}\Big[I_A(nT) + aI_B(nT) + a^2 I_C(nT) \Big] = I_{1x}(nT) + jI_{1y}(nT); \\ \underline{I}_2(nT) = \dfrac{1}{3}\Big[I_A(nT) + a^2 I_B(nT) + aI_C(nT) \Big] = I_{2x}(nT) + jI_{2y}(nT); \\ \underline{I}_0(nT) = \dfrac{1}{3}\big[I_A(nT) + I_B(nT) + aI_C(nT) \big] = I_{0x}(nT) + jI_{0y}(nT). \end{cases} \qquad (2.26)$$

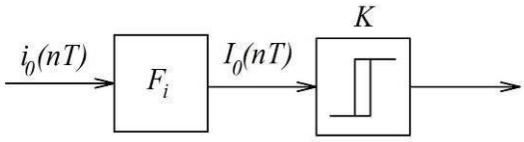
На рис. 2.14, *а*, наведено структуру цифрового вимірювального органу, що реалізує друге рівняння системи (2.26). Орган контролює величину струму зворотної послідовності та спрацьовує при умові $\left|\underline{I}_2(nT)\right| \geq C$, де C – уставка спрацювання компаратора K.

Якщо на об'єкті є вимірювальний трансформатор струму i_0 нульової послідовності, то для одержання цифрових відліків діючого значення струму нульової послідовності може використовуватися цифровий вимірювач струму F_i, рис. 2.14, *б*. Дискретні значення вектора струму нульової послідовності $\underline{I}_0(nT)$ можуть бути обраховані непрямим шляхом – як сума дискретних

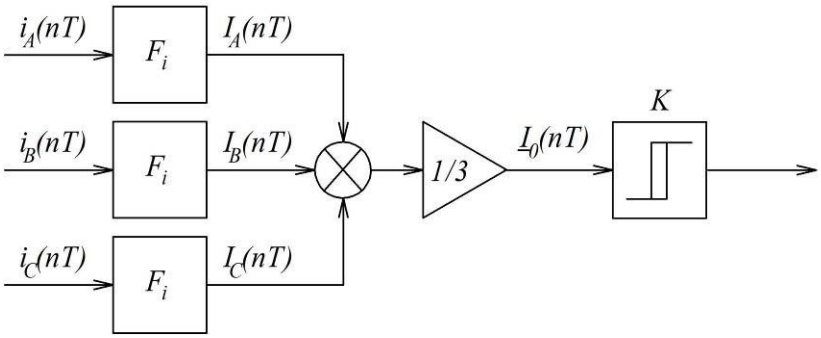
значень векторів фазних струмів, відповідно до третього рівняння системи (2.26), рис. 2.14, *в*.



а)



б)



в)

Рис. 2.14. Алгоритмічні структури цифрових вимірювальних органів на основі симетричних складових струму для контролю:
а – струму зворотної послідовності; *б* – струму нульової послідовності за даними трансформатора струму нульової послідовності;
в – струму нульової послідовності за фазними струмами

2.3.2. Цифрові вимірювальні органи напрямку потужності

2.3.2.1. Характеристики спрацювання

Напрямок потужності в лінії електропередачі визначається кутом φ між векторами напруги $\underline{U}$ та струму $\underline{I}$. Такий кут характеризує напрямок потужності як в нормальному режимі, так і при короткому замиканні. Величина кута, який вимірюється мікропроцесорним захистом, залежить від взаємного розташування вимірювального органу та точки короткого замикання, а також від способу приєднання вимірювальних трансформаторів струму та напруги.

Розглянемо фрагмент електричної схеми, рис. 2.15, *а*, що включає шини *A* підстанції, до яких приєднано дві лінії електропередачі $W1$, $W2$. Захист вказаних ліній здійснюється захистами $S1$ і $S2$.

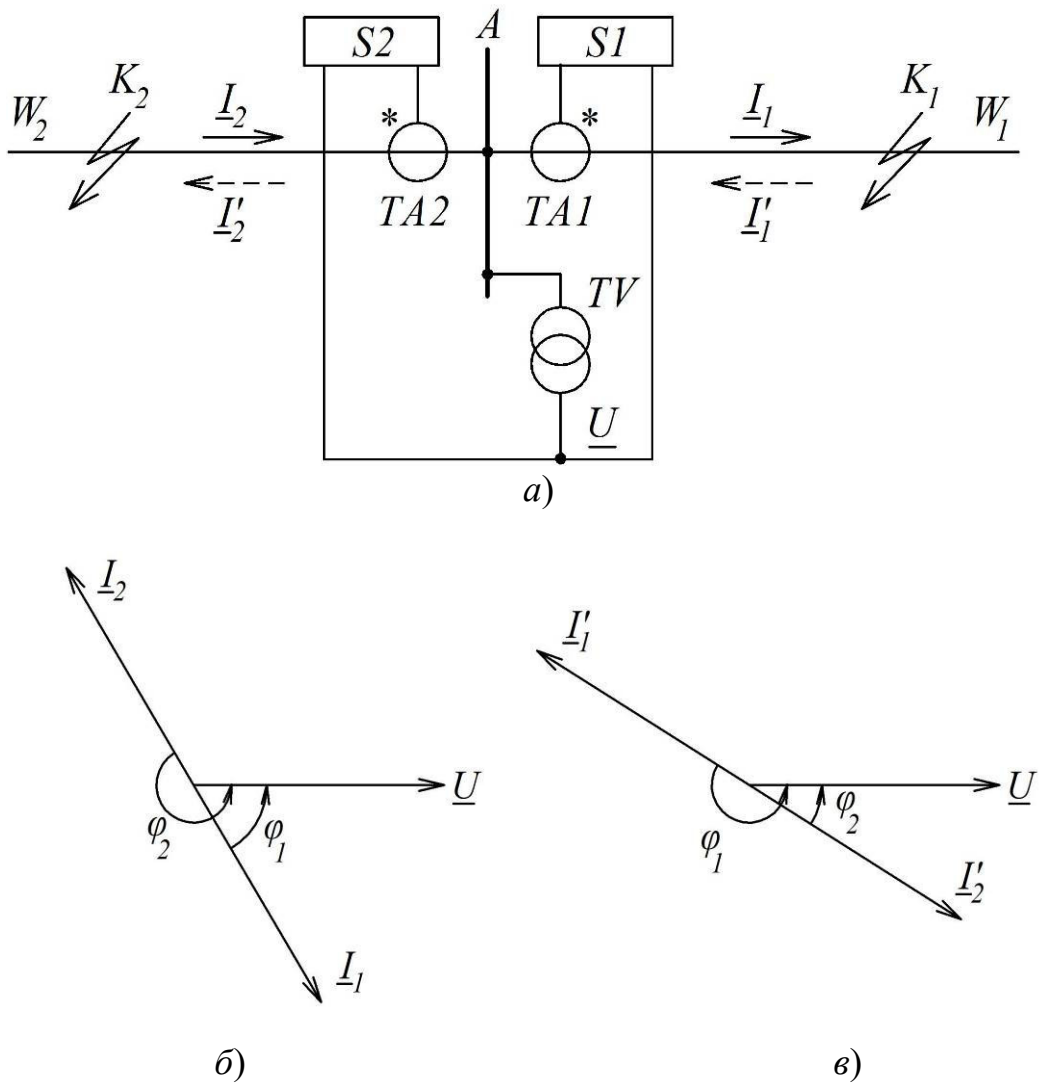
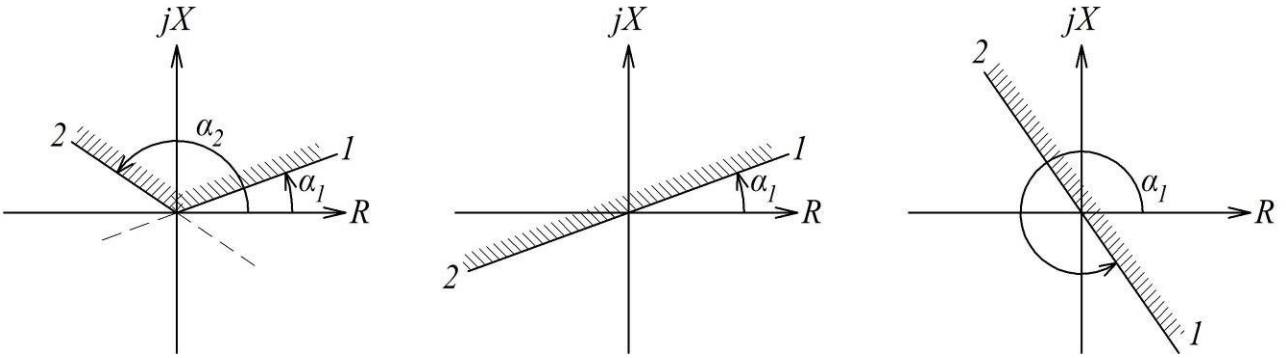


Рис. 2.15. Фазові співвідношення між струмами та напругами при короткому замиканні в електромережі

До захисті $S1$, $S2$ підводяться струми I_1, I_2, відповідно, від трансформаторів струму $TA1$, $TA2$, підключених початками обмотки (позначені *) у напрямку лінії. До обох захистів підводиться однакова напруга $\underline{U}$ від вимірювального трансформатора TV, підключеного до шин. Надалі відраховуватимемо кут φ_1 між струмом і напругою від вектора струму проти годинникової стрілки до вектора напруги. При короткому замиканні в точці K_1 на лінії $W1$ кут φ_1 між струмом $\underline{I}_1$ і напругою $\underline{U}$, що підводяться до захисту $S1$, дорівнює куту лінії $W1$. Кут φ_2 між струмом $\underline{I}_2$ і напругою U, що підводяться до захисту $S2$, більше кута φ_1 на π рад., оскільки струм $\underline{I}_2$ має протилежний напрямок щодо шин порівняно зі струмом $\underline{I}_1$ (рис. 2.15, *б*).

В разі виникнення короткого замикання в точці K_2 через захисти $S1$ і $S2$ протікають струми $\underline{I}'_1$ і $\underline{I}'_2$ (рис. 2.15, *а*). У цьому випадку кут між струмом $\underline{I}'_2$ і напругою $\underline{U}$, що підводяться до захисту $S2$ дорівнює куту лінії (рис. 2.16, *в*). Кут між струмом $\underline{I}'_1$ і напругою $\underline{U}$ більше кута φ_2 на π рад. Таким чином, за величиною кута між струмом і напругою, що підводяться до захисту, можна встановити розташування місця короткого замикання щодо шин A. Визначення аварійної точки дозволяє здійснювати, з урахуванням вимог селективності, захисні відключення. Вимірювальні органи, що виконують дані функції, називаються органами спрямування потужності. Кут між струмом та напругою відповідає аргументу комплексного опору $\varphi = \arg(\underline{U} / \underline{I}) = \arg \underline{Z}$.

Характеристика спрацювання цифрового вимірювального органу напрямку потужності визначається двома параметрами: дійсною та уявною частинами імпедансу, тобто активним та реактивним опором. Тому, на відміну від струмових органів захисту, спрацювання визначається не уставкою, а частиною комплексної площини. Наприклад, на рис. 2.16, *а*, наведена комплексна площина та промені 1, 2, що розташовані під кутами α_1, α_2, відповідно, до дійсної осі. Промінь 1 є початковим, під час руху від нього проти годинникової стрілки в напрямку променя 2 кут $\alpha_1 \le \varphi \le \alpha_2$. знаходиться в області спрацьовування цифрового вимірювального органу.



а) *б*) *в*)

Рис. 2.16. Характеристики спрацювання цифрового вимірювального органу напрямку потужності

2.3.2.2. Органи з використанням ортогональних складових векторів

Припустимо, що в результаті дискретизації виміряних значень визначені проекції дискретних векторів струму $\underline{I}(nT) = I_x(nT) + jI_y(nT)$ та напруги $U(nT) = U_x(nT) + jU_y(nT)$. В програмі для мікроконтролера це відповідає одержанню чисел для кожної з проекцій через інтервали дискретизації. Сформулюємо умову спрацювання цифрового вимірювального органу через дискретні значення проекцій векторів. Розглянемо найпростіший випадок прямолінійної характеристики спрацювання цифрового вимірювального органу (рис. 2.16, *б*, *в*). Така характеристика може бути описана співвідношенням:

$$\alpha_1 \le \arg\frac{\underline{U}(nT)}{\underline{I}(nT)} \le \alpha_1 + \pi, \tag{2.27}$$

що відповідає:

$$0 \le \arg\frac{\underline{U}(nT)}{\underline{I}(nT)} - \alpha_1 \le \pi.$$

Враховуючи, що $\underline{U}(nT)/\underline{I}(nT) = \underline{Z}(nT)$, отриману умову запишемо у вигляді:

$$0 \le \arg\left[\underline{Z}(nT)e^{-j\alpha_1}\right] \le \pi. \tag{2.28}$$

Умова (2.28) означає, що вектор $\underline{Z}(nT)e^{-j\alpha_1}$ повинен бути у верхній напівплощині $\underline{Z}$ (рис. 2.17), тобто, що його уявна складова має бути завжди додатною. Таким чином, умова спрацьовування цифрового вимірювального органу напряму потужності з характеристиками спрацьовування відповідно до рис. 2.16, *б*, *в*, має вигляд:

$$\mathrm{Im}\left[\underline{Z}(nT)e^{-j\alpha_1}\right] \ge 0. \tag{2.29}$$

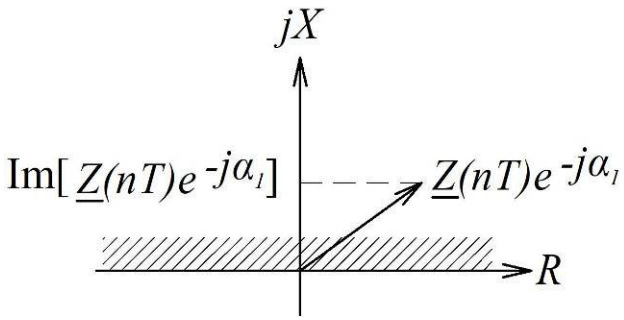


Рис. 2.17. Область спрацьовування цифрового вимірювального органу напряму потужності з прямолінійною характеристикою

Імпеданс $\underline{Z}(nT)$ може бути виражений за допомогою ортогональних складових векторів $\underline{I}(nT)$ та $\underline{U}(nT)$:

$$\underline{Z}(nT) = \frac{\underline{U}(nT)}{\underline{I}(nT)} = \frac{U_x(nT) + jU_y(nT)}{I_x(nT) + jI_x(nT)} = R(nT) + jX(nT), \tag{2.30}$$

де

$$R(nT) = \frac{U_x(nT)I_x(nT) + U_x(nT)I_y(nT)}{I_x^2(nT) + I_y^2(nT)};$$

$$X(nT) = \frac{I_x(nT)U_y(nT) - I_y(nT)U_x(nT)}{I_x^2(nT) + I_y^2(nT)}.$$

Умова спрацювання (2.29) з урахуванням (2.30) набуде вигляду:

$$\mathrm{Im}\left[\frac{U_x(nT) + jU_y(nT)}{I_x(nT) + jI_y(nT)}(\cos\alpha_1 - j\sin\alpha_1)\right] \ge 0. \tag{2.31}$$

Перетворивши (2.31) і виділивши уявну частину, отримаємо остаточно алгоритм реле напрямку потужності з діапазоном кутів спрацьовування π, що використовує ортогональні складові векторів струму та напруги:

$$\begin{aligned}&\cos\alpha_1\left[U_y(nT)I_x(nT)-U_x(nT)I_y(nT)\right]-\\&-\sin\alpha_1\left[U_x(nT)I_x(nT)+U_y(nT)I_y(nT)\right]\geq 0.\end{aligned} \tag{2.32}$$

Більш складна характеристика цифрового вимірювального органу напряму потужності при $\alpha_1 \leq \varphi \leq \alpha_2$ визначиться як результат одночасної дії (логічна операція AND) двох реле з початковими променями 1 і 2 (рис. 2.16, *а*). З урахуванням нерівності (2.32), показана на рис. 2.16, *а*, область спрацювання визначається як одночасне виконання наступних умов:

$$\begin{cases}\cos\alpha_1\left[U_y(nT)I_x(nT)-U_x(nT)I_y(nT)\right]-\\\quad-\sin\alpha_1\left[U_x(nT)I_x(nT)+U_y(nT)I_y(nT)\right]\geq 0;\\\cos\alpha_2\left[U_y(nT)I_x(nT)-U_x(nT)I_y(nT)\right]+\\\quad+\sin\alpha_2\left[U_x(nT)I_x(nT)+U_y(nT)I_y(nT)\right]\geq 0.\end{cases} \tag{2.33}$$

2.3.2.3. Безпосереднє використання вибірок миттєвих значень

Існує можливість реалізувати цифровий вимірювальний орган напрямку потужності, що визначає стан електромережі за миттєвими дискретними значеннями струмів на напруг, без використання ортогональних складових векторів. Припустимо, що дискретні значення $u(nT)$ та $i(nT)$ попередньо відфільтровані. Зсув дискретизованої синусоїдної величини $u=\sin(\omega_0 nT+\varphi)$ на кут α_1 відповідає заміні дискретного моменту часу nT на $n\cdot T-d\cdot T$, де d – число періодів дискретизації, що відповідають зсуву:

$$d=\alpha_1/\omega_0 T=\alpha_1 N/2\pi. \tag{2.34}$$

Тоді умова спрацьовування цифрового вимірювального органу напряму потужності (2.29) набуде вигляду:

$$\mathrm{Im}\left[\frac{\underline{U}(nT)e^{-j\alpha_1}}{\underline{I}(nT)}\right]\geq 0,$$

що еквівалентно

$$\mathrm{Im}\left[\frac{\underline{U}(nT-dT)}{\underline{I}(nT)}\right]\geq 0, \tag{2.35}$$

де додаткове число d зсувних вибірок напруги $u(nT)$ визначається виразом (2.34). Використовуючи (2.16) алгоритму «двох вибірок», розділених часовим проміжком mT, при $\beta=\omega_0 mT$ отримаємо:

$$\underline{U}(nT-dT)=\frac{1}{\sin\beta}\left[u(nT-dT)(\cos\beta+j\sin\beta)-u(nT-dT-mT)\right];$$

$$\underline{I}(nT)=\frac{1}{\sin\beta}\left[i(nT)(\cos\beta+j\sin\beta)-i(nT-mT)\right].$$

Підставляючи отримані значення $\underline{U}(nT-dT)$ та $\underline{I}(nT)$ до нерівності (2.35) і враховуючи, що відстань mT між двома вибірками менша за напівперіод $T_0/2$ ($\beta<\pi$, $\sin\beta\geq 0$), отримаємо перетворений алгоритм цифрового вимірювального органу напрямку потужності з характеристикою рис. 2.16, *б*, *в*, а саме:

$$u(nT-dT-mT)i(nT)-u(nT-dT)i(nT-mT)\geq 0.$$

Даний алгоритм використовує по дві поточні вибірки миттєвих значень напруги $u(nT-dT)$, $u(nT-dT-mT)$ і струму $i(nT)$, $i(nT-mT)$, віддалені одна від одної на обране число m періодів дискретизації. При цьому напруга $u(nT)$, зрушена на кількість періодів дискретизації d, пропорційна куту нахилу характеристики спрацьовування α_1.

2.3.2.4. Цифрові вимірювальні органи симетричних складових

Використання симетричних складових зворотної та нульової послідовностей в цифрових вимірювальних органах напряму потужності дозволяє підвищити чутливість і селективність порівняно з органами, що реагують на повні струми і напруги. Зазначене викликано насамперед значно більшими можливостями відведення від різних навантажувальних режимів через те, що у цих режимах симетричні складові досить малі. Можливості використання складових зворотної та нульової послідовностей для визначення напрямку до місця короткого замикання пояснює рис. 2.18.

При вимірюванні симетричних складових $\underline{U}_{2к}$ ($\underline{U}_{0к}$) зворотної та нульової послідовностей у місці встановлення релейного захисту джерело цих складових розташовується в точці короткого замикання. А саме: в точці K_1 – при замиканні у зоні дії захисту $S1$; K_2 – при замиканні поза зоною дії захисту $S2$ (рис. 2.18, *а*, *б*). Векторна діаграма рис. 2.18, *в*, показує, що при замиканні в зоні дії захисту $S1$ складові $\underline{I}_2^{'}$, $\underline{I}_0^{'}$ спрямовані до шин. При цьому кут φ_1 між $\underline{U}_2$ та $\underline{I}_2^{'}$ для зворотної послідовності між $\underline{U}_0$ та $\underline{I}_0^{'}$ для нульової послідовності знаходиться в третьому квадранті. При короткому замиканні K_2 на сусідній лінії струми $\underline{I}_2^{''}$ ($\underline{I}_0^{''}$) в захисті протікають від шин до лінії (кут φ_2 знаходиться в першому квадранті). Таким чином, для коректного визначення напрямку короткого замикання на основі вимірювання кута між симетричними складовими струму і напруги зворотної (нульової) послідовностей характеристика цифрового органу напряму потужності, на відміну від органу, що реагує на фазні величини, повинна мати область спрацьовування, яка охоплює третій квадрант площини опорів $\underline{Z}_2=\underline{U}_2/\underline{I}_2$ або $\underline{Z}_0=\underline{U}_0/\underline{I}_0$ (рис. 2.18, *г*).

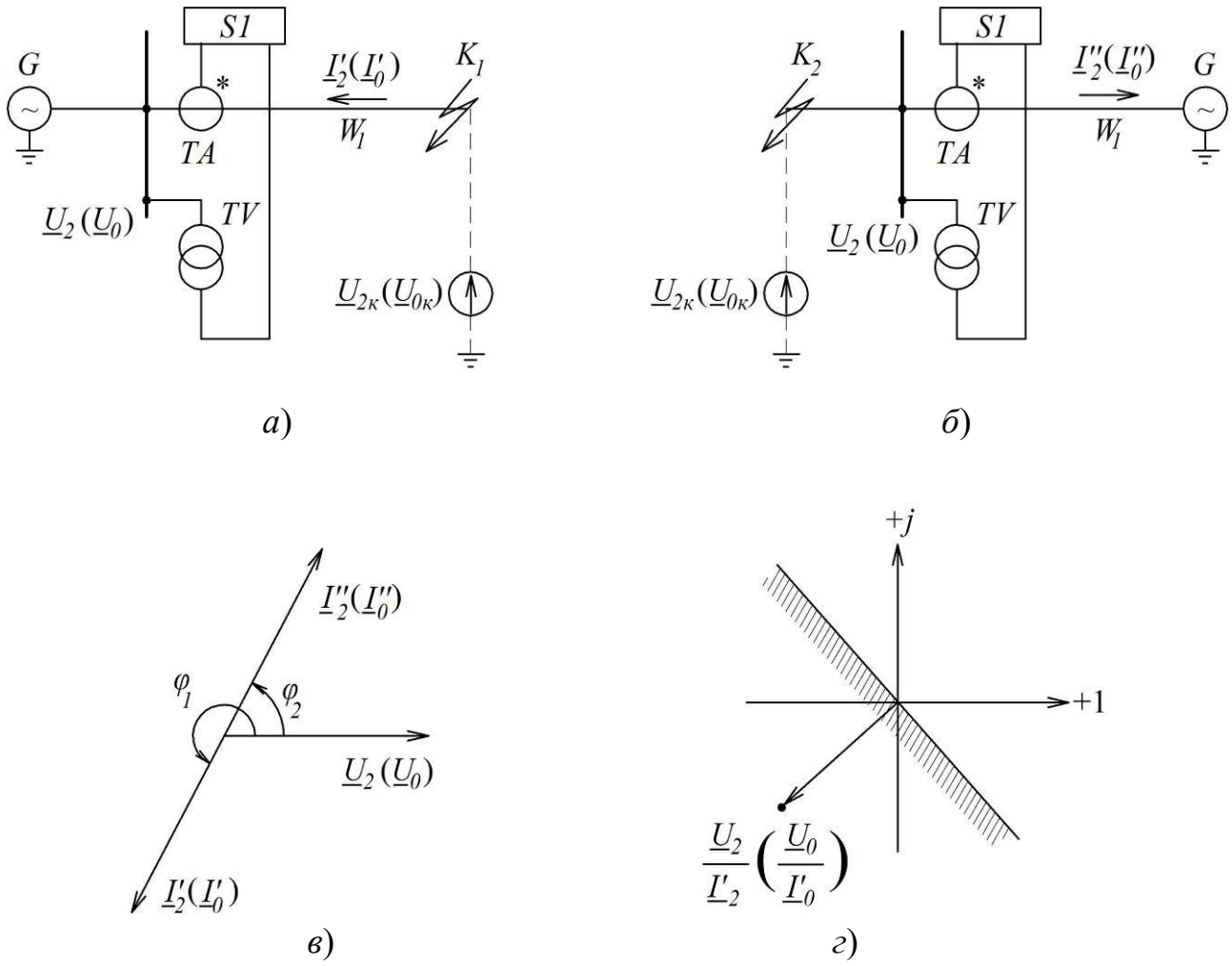


Рис. 2.18. Визначення складових зворотної та нульової послідовностей при несиметричних коротких замиканнях

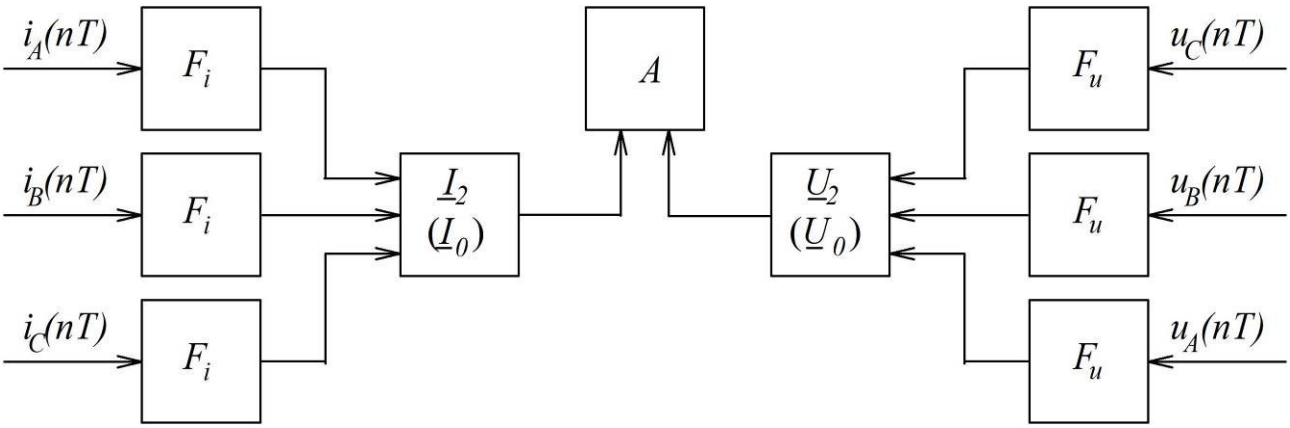


Рис. 2.19. Алгоритмічна структура цифрового вимірювального органу напрямку потужності зворотної (нульової) послідовності

Структура цифрового органу напряму потужності зворотної (нульової) послідовності наведено на рис. 2.19, де позначено: F_i, F_u – цифрові вимірювачі векторів струмів та напруг; $\underline{I}_2(I_0)$, $\underline{U}_2(\underline{U}_0)$ – блоки обчислення векторів струмів і напруг зворотної (нульової) послідовностей; А – алгоритм оцінки співвідношень потужності зворотної (нульової) послідовності. При цьому у

зазначених виразах замість фазних величин використовуватись відповідні складові зворотної (нульової) послідовності.

2.3.3. Питання для самоперевірки

1. Які основні функції має виконувати цифрові вимірювальні органи струму та напруги?
2. Які переваги виявлення аварійних режимів в електромережі за струмом зворотної послідовності?
3. В чому полягає метод симетричних складових?
4. Поясніть алгоритмічну структуру цифрового вимірювального органу для контролю струму зворотної послідовності?
5. Розкрийте принцип побудови характеристики спрацювання цифрового вимірювального органу напрямку потужності.
6. Яким чином цифровий орган з використанням ортогональних складових векторів виявляє коротке замикання за напрямком потужності?
7. Як можна виразити умову спрацювання цифрового вимірювального органу напрямку потужності через дискретні значення проекцій векторів?
8. Як можна реалізувати цифровий вимірювальний орган напрямку потужності, що визначає стан електромережі за миттєвими дискретними значеннями струмів на напруг?
9. Поясніть принцип функціонування цифрових вимірювальних органів симетричних складових.

2.4. Цифрові дистанційні органи релейного захисту

Вхідні величини та характеристики спрацювання дистанційних органів. Пофазні та трифазні дистанційні органи. Цифрові дистанційні органи на основі порівняння абсолютних значень електричних величин. Дистанційні органи на основі порівняння фаз двох електричних величин.

2.4.1. Вхідні величини та характеристики спрацювання дистанційних органів

Дистанційні релейні захисти забезпечують визначення короткого замикання за величиною імпедансу $\underline{Z} = \underline{U} / \underline{I}$ контрольованої електромережі Суттєве зниження імпедансу при короткому замиканні виступає ознакою аварійного стану. Крім того, захисти дозволяють визначити відстань (дистанцію) до місця ушкодження за рахунок того, що імпеданс в аварійному режимі є пропорційним відомому питомому імпедансу лінії.

Основою дистанційних релейних захистів є дистанційні вимірювальні органи. Для дистанційних органів вхідними величинами є сигнали $\underline{U}_A$, $\underline{U}_B$, $\underline{U}_C$ від вимірювальних трансформаторів напруги та $\underline{I}_A$, $\underline{I}_B$, $\underline{I}_C$ від вимірювальних трансформаторів струму. Припустимо, що $\underline{U}_p$ – величина, що є лінійною функцією фазних напруг, а $\underline{I}_p$ – фазних струмів. За умови $\underline{U}_p = \underline{U}_A - \underline{U}_B$ та $\underline{I}_p = \underline{I}_A - \underline{I}_B$, імпеданс (комплексний опір) дорівнює:

$$\underline{Z}_p = \frac{\underline{U}_p}{\underline{I}_p} = \frac{\underline{U}_A - \underline{U}_B}{\underline{I}_A - \underline{I}_B}. \qquad (2.36)$$

пропорційно опору прямої послідовності $\underline{Z}_{1k}$ ділянки лінії між місцем КЗ АВ та місцем встановлення захисту, тобто:

$$\underline{Z}_p = \underline{Z}_{AB} = \underline{Z}_{1k} = \underline{Z}_{1r} l_к,$$

де $\underline{Z}_{1r}$ – питомий імпеданс прямої послідовності лінії; $l_к$ – відстань між місцем встановлення захисту та місцем короткого замикання.

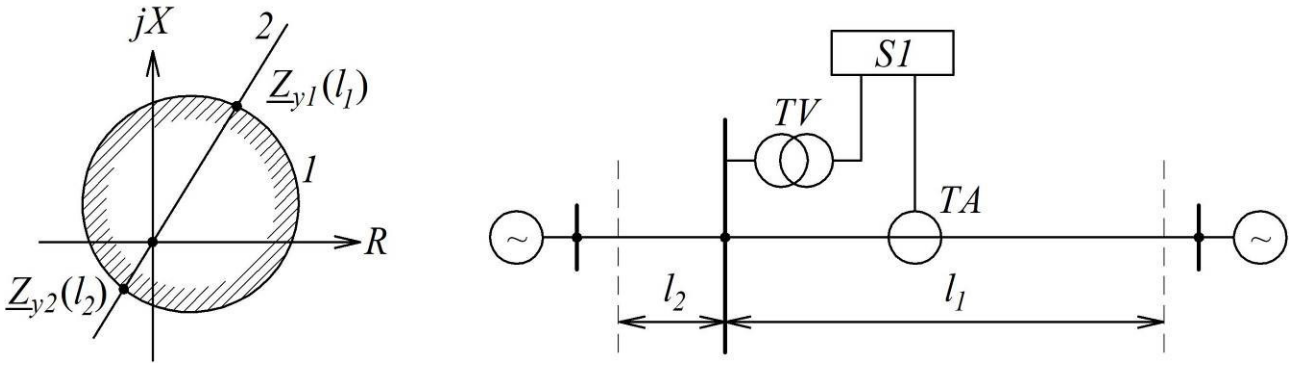
При $\underline{U}_p = \underline{U}_A$, $\underline{I}_p = \underline{I}_A + \underline{k}_0^{'} \underline{I}_0$, де $\underline{k}_0^{'} = (\underline{Z}_{0r} - \underline{Z}_{1r}) / \underline{Z}_{1r}$, вимірюваний опір

$$\underline{Z}_{A0} = \underline{Z}_p = \underline{U}_p / \underline{I}_p = \underline{U}_A / (\underline{I}_A + \underline{k}_0^{'} \underline{I}_0) = \underline{Z}_{1r} l_к$$

є пропорційним відстані між місцем встановлення захисту та точкою короткого замикання для випадку однофазного замикання фази А на землю.

Таким чином, якщо побудувати дистанційний орган, що фіксує знаходження вектора $\underline{Z}_p$ у визначеній області комплексної площини $\underline{Z}$ (всередині характеристики спрацьовування), то вказане буде відповідати встановленню віддаленості точки короткого замикання від місця встановлення захисту.

Припустимо, що характеристика спрацювання дистанційного захисту відповідає колу (рис. 2.20, *а*, поз. 1). Коефіцієнт потужності лінії (без урахування навантаження), яка захищається, є сталим, тому характеристика лінії описується прямою (поз. 2). Точки $\underline{Z}_{y1}$, $\underline{Z}_{y2}$, в яких характеристика спрацювання (1) перетинається з характеристикою лінії (1), визначають довжини l_1 і l_2 лінії в обох напрямках від місця встановлення дистанційного захисту S1, що захищені, рис. 2.20, *б*.



а) *б*)

Рис. 2.20. Колова характеристика спрацювання дистанційного захисту (*а*) та ділянки лінії, які захищаються (*б*)

2.4.2. Пофазні та трифазні дистанційні органи

При міжфазних металевих пошкодженнях на лінії, що захищається, один з опорів $\underline{Z}_{AB}$, $\underline{Z}_{BC}$, $\underline{Z}_{CA}$ (при двофазних замиканнях) або всі три опори (при трифазному замиканні) пропорційні відстані між місцем пошкодження та місцем встановлення дистанційного захисту (місцем підключення вимірювальних трансформаторів струму та напруги). При металевих однофазних і двофазних замиканнях на землю опори $\underline{Z}_{A0}$, $\underline{Z}_{B0}$, $\underline{Z}_{C0}$ (залежно від пошкодженої фази) також пропорційні відстані до місця пошкодження. Надалі напруги та комбінації напруг, що беруть участь у формуванні опору на вході дистанційного захисту та входять до чисельників виразів типу (2.36) для обчислення $\underline{Z}_p$, будемо позначати як $\underline{U}_p$ (напруги, що підводяться до реле). Струми або комбінації струмів, що входять до знаменників вирази (2.36) – як $\underline{I}_p$ (струми, що підводяться до реле). У табл. 2.1 наведено значення $\underline{U}_p$ та $\underline{I}_p$, відповідні їм вхідні опори та види пошкодження, при яких опори пропорційні віддаленості ушкодженя.

Таблиця 2.1

Величини, що використовуються при побудові пофазних дистанційних органів

Вхідні величини			Вхідні опори	Вид короткого замикання
Тип дистанційного органу	$\underline{U}_p$	$\underline{I}_p$		
ДО$_{AB}$	$\underline{U}_{AB}$	$\underline{I}_A - \underline{I}_B$	$\underline{Z}_{AB}$	ABC, AB, AB0
ДО$_{BC}$	$\underline{U}_{BC}$	$\underline{I}_B - \underline{I}_C$	$\underline{Z}_{BC}$	ABC, BC, BC0
ДО$_{CA}$	$\underline{U}_{CA}$	$\underline{I}_C - \underline{I}_A$	$\underline{Z}_{CA}$	ABC, AC, AC0
ДО$_{A0}$	$\underline{U}_A$	$\underline{I}_A + \underline{k}_0^{'}\underline{I}_0$	$\underline{Z}_{A0}$	A0, AB0, AC0
ДО$_{B0}$	$\underline{U}_B$	$\underline{I}_B + \underline{k}_0^{'}\underline{I}_0$	$\underline{Z}_{B0}$	B0, AB0, BC0
ДО$_{C0}$	$\underline{U}_C$	$\underline{I}_C + \underline{k}_0^{'}\underline{I}_0$	$\underline{Z}_{C0}$	C0, AC0, BC0

З урахуванням даних табл. 2.1, обрахування опору $\underline{Z}_{AB}$ дозволяє з мінімальними похибками визначити віддаленість пошкодження при міжфазних замиканнях ABC, AB, AB0 тощо. Повноцінний захист, що реагує на всі види пошкоджень, можна виконати, фіксуючи всі шість вхідних опорів відповідно до табл. 2.1 за допомогою дистанційного органу. При цьому можливе виконання дистанційного органу, що фіксує правильно лише якийсь один вхідний опір, наприклад $\underline{Z}_{AB}$ при замиканні між фазами А і В. У цьому випадку до дистанційного органу підводяться тільки $\underline{U}_p = \underline{U}_{AB}$ та $\underline{I}_p = \underline{I}_{AB}$. І при інших видах замикань, наприклад ВС, на вході ДО$_{AB}$ будуть певні значення $\underline{U}_p$ і $\underline{I}_p$, але вхідний опір вже не буде відповідати відстані до місця пошкодження.

Органи, до яких підводиться одне поєднання $\underline{U}_p$, $\underline{I}_p$ із табл. 2.1, називаються пофазними. Пофазні органи можуть бути призначені для дії при однофазних і міжфазних замиканнях, що визначається величинами, які підводяться до них. Існують і трифазні органи, що діють правильно при кількох видах ушкоджень. До таких органів підводяться кілька величин $\underline{U}_p$, $\underline{I}_p$, що використовуються для фіксації вхідних опорів при різних видах замикання.

2.4.3. Цифрові дистанційні органи на основі порівняння абсолютних значень електричних величин

Для порівняння двох синусоїдних електричних величин за абсолютним значенням необхідно з'ясувати виконання умови:

$$E_1 \geq E_{2,} \tag{2.37}$$

де E_1 та E_2 – амплітудні або пропорційні їм значення синусоїдних величин.

Вираз (2.37) визначає алгоритм порівняння абсолютних значень двох величин. У комплексній площині $\underline{w} = \underline{E}_2 / \underline{E}_1$, що характеризує співвідношення між векторними величинами $\underline{E}_1$, $\underline{E}_2$, умова спрацьовування (2.37) має вигляд $\left|\underline{w}\right| \leq 1$, що відповідає області, обмеженою колом одиничного радіусу з центром на початку координат. Знаходження точки $\underline{w}$ всередині цього кола означає, що незалежно від кута $\varphi = \arg(\underline{E}_2 / \underline{E}_1)$ між векторами $\underline{E}_1$ та $\underline{E}_2$ умова (2.37) виконується. Прийнявши порівнювані величини

$$\begin{cases} \underline{E}_1 = \underline{k}_1 \underline{U}_p + \underline{k}_2 \underline{I}_p; \\ \underline{E}_2 = \underline{k}_3 \underline{U}_p + \underline{k}_5 \underline{I}_p, \end{cases} \tag{2.38}$$

де $\underline{U}_p$ та $\underline{I}_p$ вибираються відповідно до табл. 2.1, отримаємо, що умові $\left|\underline{w}\right| \leq 1$, тобто умові спрацьовування дистанційного органу, відповідає співвідношення

$$\frac{\left|\underline{k}_3 \underline{U}_p + \underline{k}_4 \underline{I}_p\right|}{\left|\underline{k}_1 \underline{U}_p + \underline{k}_2 \underline{I}_p\right|} = \frac{\left|\underline{k}_3 \underline{Z} + \underline{k}_4\right|}{\left|\underline{k}_1 \underline{Z} + \underline{k}_2\right|} \leq 1. \tag{2.39}$$

Порівнянню величин $\underline{E}_1$ і $\underline{E}_2$ (2.38) відповідає структура дистанційного органу на рис. 2.21.

З виразу (2.39) можна встановити, що область спрацювання дистанційного органу в площині $\underline{Z}$ обмежується колом (при $k = \left|\underline{k}_3 / \underline{k}_1\right| \neq 1$) і прямою (при $k = 1$). Координати центра кола $\underline{Z}_0$ та радіус R_0 (рис. 2.22, характеристика 1) визначається співвідношеннями:

$$\underline{Z}_0 = \frac{k^2 \underline{a} - \underline{b}}{k^2 - 1}; \; R_0 = \frac{k\left|\underline{a} - \underline{b}\right|}{\left|k^2 - 1\right|}; \; k = \left|\frac{\underline{k}_3}{\underline{k}_1}\right|; \; \underline{a} = -\frac{\underline{k}_2}{\underline{k}_1}. \tag{2.40}$$

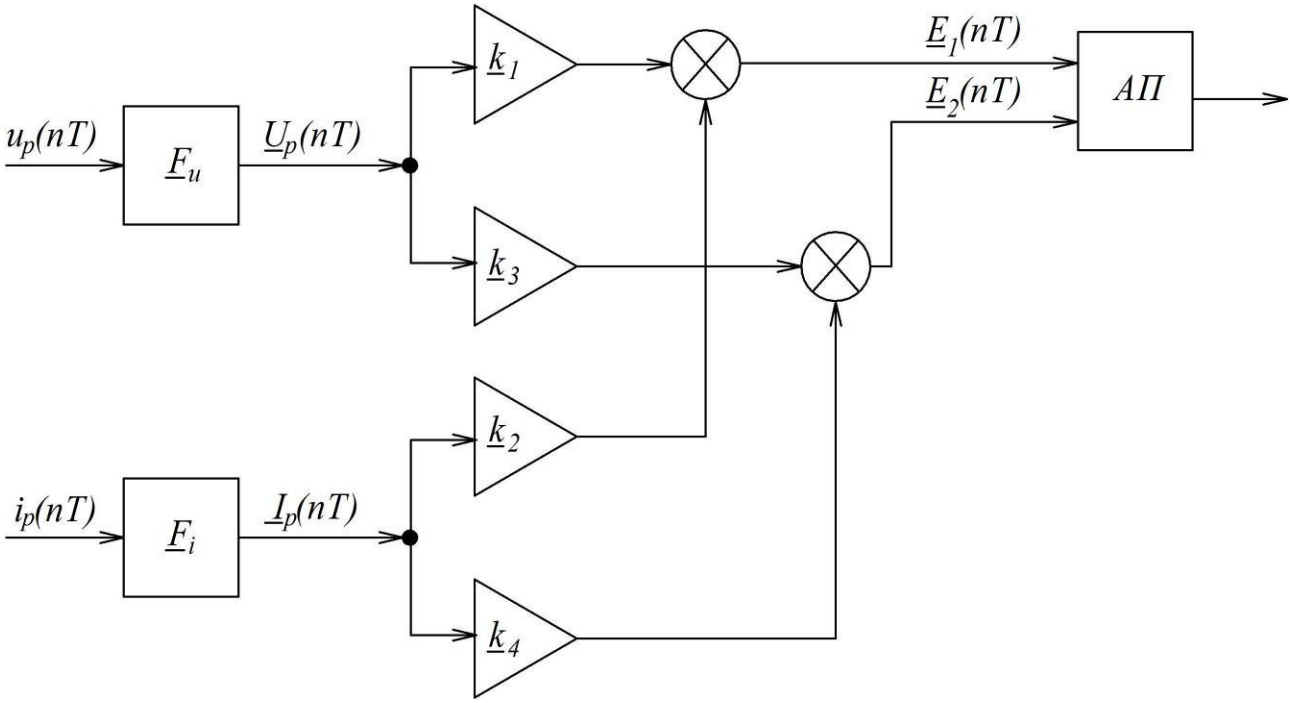


Рис. 2.21. Структура дистанційного органу на основі порівняння абсолютних значень двох електричних величин, АП – алгоритм порівняння

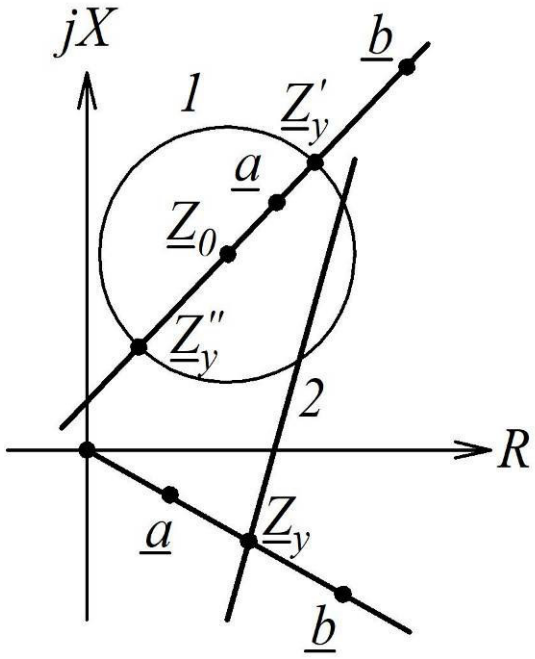


Рис. 2.22. Характеристика спрацювання дистанційного органу

При $k=1$ виразу (2.40) відповідає пряма, що розташовується в площині $\underline{Z}$ таким чином, що особливі точки $\underline{a}$ і $\underline{b}$ є симетричними щодо неї (рис 2.22, характеристика 2). При характеристиці спрацьовування у вигляді кола точки $\underline{a}$ і $\underline{b}$ знаходяться на одній прямій з центром $\underline{Z}_0$ кола 1, завжди по один бік від нього. При цьому точка $\underline{a}$ завжди розташована в області спрацювання, а точка $\underline{b}$ – поза її межами. Це слідує з того, що при $\underline{Z}=\underline{a}$ з (2.38) маємо $\underline{E}_2=0$, тобто

завжди $E_1 > E_2$, а при $\underline{Z} = \underline{b}$ маємо $\underline{E}_1 = 0$, тобто $E_1 < E_2$. Значення $\underline{Z}_y^{'}$ і $\underline{Z}_y^{''}$, що належать характеристиці спрацювання та прямій, на якій розташовані точки $\underline{a}$ *і* $\underline{b}$, є уставками дистанційного органу щодо опору спрацювання. При цьому $\underline{Z}_y^{'}$ завжди розташовується з тієї ж сторони від центру $\underline{Z}_0$, що і точки $\underline{a}$ *і* $\underline{b}$.

Враховуючи, що

$$\underline{Z}_0 = \frac{\underline{Z}_y^{'} + \underline{Z}_y^{''}}{2},\ R_0 = \frac{\left|\underline{Z}_y^{'} - \underline{Z}_y^{''}\right|}{2}, \tag{2.41}$$

і розв'язуючи спільно (2.40) та (2.41), отримаємо:

$$\underline{Z}_y^{'} = \frac{k\underline{a} + \underline{b}}{k+1};\ \underline{Z}_y^{''} = \frac{k\underline{a} - \underline{b}}{k-1}. \tag{2.42}$$

Для прямої при $k = 1$ із (2.42) маємо (рис. 2.22, характеристика 2):

$$\underline{Z}_y^{''} = \infty;\ \underline{Z}_y^{'} = \underline{Z}_y = \frac{\underline{a} + \underline{b}}{2}. \tag{2.43}$$

Деякі коефіцієнти $\underline{k}_1 - \underline{k}_4$ у виразах (2.38) можна прийняти нульовими. При потрібній характеристиці спрацювання у вигляді кола, враховуючи, що точка $\underline{Z} = \underline{a}$ $(\underline{E}_2 = 0)$ завжди знаходиться всередині характеристики спрацювання, а точка $\underline{Z} = \underline{b}$ $(\underline{E}_1 = 0)$ завжди поза нею, можна прийняти такі спрощення:

1) допускається вважати $\underline{k}_1 = 0$, тобто

$$\underline{E}_1 = \underline{k}_2\underline{I}_p;\ \underline{E}_2 = \underline{k}_3\underline{U}_p + \underline{k}_4\underline{I}_p. \tag{2.44}$$

При цьому із (2.40) маємо $\underline{Z}_0 = \underline{a};\ R_0 = k_2 / k_3$. Для спрямованого дистанційного органу з характеристикою 1, що наведена на рис. 2.22, *б*, маємо при цьому $k_2 = k_4$. В іншому окремому випадку дистанційного органу повного опору (характеристика спрацювання у вигляді кола з центром на початку координат) маємо $\underline{Z}_0 = \underline{a} = 0$ $(\underline{k}_4 = 0)$, і порівнювані величини набувають вигляду:

$$\underline{E}_1 = \underline{k}_2\underline{I}_p;\ \underline{E}_2 = \underline{k}_3\underline{U}_p.$$

2) якщо характеристика спрацювання має вигляд кола, що охоплює початок координат, можливе також спрощення $\underline{a} = 0$ $(\underline{k}_4 = 0;\ \underline{E}_1 = \underline{k}_1\underline{U}_p + \underline{k}_2\underline{I}_p;\ \underline{E}_2 = \underline{k}_3\underline{U}_p)$, тоді з урахуванням (2.40) маємо:

$$\underline{Z}_0 = \frac{\underline{b}}{k^2 - 1};\ \underline{R}_0 = \frac{k\left|\underline{b}\right|}{\left|k^2 - 1\right|}. \tag{2.45}$$

3) якщо характеристика спрацювання має вигляд кола, що не охоплює початок координат, можливе спрощення $\underline{b} = 0$ $(\underline{k}_2 = 0;\ \underline{E}_1 = \underline{k}_1\underline{U}_p;\ \underline{E}_2 = \underline{k}_3\underline{U}_p + \underline{k}_4\underline{I}_p)$, звідки з (2.40) маємо

$$\underline{Z}_0 = \frac{k^2\underline{a}}{k^2 - 1};\ \underline{R}_0 = \frac{k\left|\underline{a}\right|}{\left|k^2 - 1\right|}. \tag{2.46}$$

При прямолінійних характеристиках спрацювання, не проходять через початок координат, в залежності від необхідних зон дії одну з точок $\underline{a}$ і $\underline{b}$ можна розташовувати на початку координат. При $\underline{a}=0$ з (2.43) маємо $\underline{b}=2\underline{Z}_y$, де $\underline{Z}_y$ – вектор, що відповідає відстані від початку координат до прямої (рис. 2.22).

Реалізація дистанційного органу здійснюється на основі структури, що наведена на рис. 2.21 де, зокрема, при використанні ортогональних складових на виходах формувачів порівнюваних величин маємо:

$$\underline{E}_1(nT)=\underline{E}_{1x}(nT)+j\underline{E}_{1y}(nT);\ \underline{E}_2(nT)=\underline{E}_{2x}(nT)+j\underline{E}_{2y}(nT).$$

Алгоритм спрацювання $(E_2 \le E_1)$ має вигляд:

$$\underline{E}_{2x}^2(nT)+\underline{E}_{2y}^2(nT)\le\underline{E}_{1x}^2(nT)+E_{1y}^2(nT). \qquad (2.47)$$

2.4.4. Дистанційні органи на основі порівняння фаз двох електричних величин

Порівняння двох електричних величин $e_1=E_1\sin\omega t$ та $e_2=E_2\sin(\omega t+\varphi)$ за фазою відповідає фіксації факту знаходження кута $\varphi=\arg\underline{E}_2/\underline{E}_1=\arg\underline{\omega}$ всередині заданої області:

$$\alpha_1\le\varphi\le\alpha_2. \qquad (2.48)$$

Якщо сформувати величини $\underline{E}_1$ і $\underline{E}_2$ відповідно до (2.38), то умова (2.48) спрацьовування дистанційного органу набуде вигляду:

$$\alpha_1\le\arg\frac{\underline{k}_3\underline{U}_p+\underline{k}_4\underline{I}_p}{\underline{k}_1\underline{U}_p+\underline{k}_2\underline{I}_p}=\arg\frac{\underline{k}_3\underline{Z}+\underline{k}_4}{\underline{k}_1\underline{Z}+\underline{k}_2}\le\alpha_2. \qquad (2.49)$$

Таким чином, порівняння по фазі двох величин $\underline{E}_1=\underline{k}_1\underline{U}_p+\underline{k}_2\underline{I}_p$ та $\underline{E}_2=\underline{k}_3\underline{U}_p+\underline{k}_4\underline{I}_p$ еквівалентно відображенню області спрацьовування схеми порівняння в площині $\underline{w}$ а на площину $\underline{Z}$.

Області спрацювання в площині $\underline{Z}$, які можна отримати порівнянням по фазі двох величин $\underline{E}_1$ і $\underline{E}_2$ за виразом (2.38), є колами, прямими чи комбінаціями дуг кіл і відрізків прямих. Ці області визначаються коефіцієнтами $\underline{k}_1-\underline{k}_4$ (особливими точками $\underline{a}=-\underline{k}_4/\underline{k}_3$, $\underline{b}=-\underline{k}_2/\underline{k}_1$, параметром $\beta=\arg\underline{k}_3/\underline{k}_1$) і характеристикою схеми порівняння в площині $\underline{w}$ (кутами α_1 і α_2). Наступні положення визначають характеристики дистанційного органу в площині $\underline{Z}$:

1) при найбільш поширеному алгоритмі спрацювання $\alpha_1\le\varphi\le\alpha_1+\pi$, коли характеристика схеми порівняння в площині $\underline{w}$ – пряма, що проходить через початок координат, характеристика спрацювання в площині $\underline{Z}$ є загальному випадку є довільно розташованим колом при $\alpha_1-\beta\ne0;\ \alpha_1-\beta\ne\pi$ або прямою при $\alpha_1-\beta=0$ або $\alpha_1-\beta=\pi$;

2) особливі точки $\underline{a}$ і $\underline{b}$ розташовані на характеристиці дистанційного органу в площині $\underline{Z}$. При побудові характеристики спрацювання необхідно провести коло через точки $\underline{a}$ і $\underline{b}$ таким чином, щоб вписані кути, що спираються на точки $\underline{a}$ і $\underline{b}$, дорівнювали $\alpha_1-\beta$ або $\alpha_1-\beta+\pi$. Область дії знаходиться всередині кола при $\alpha_1-\beta<\varphi<\alpha_1-\beta+\pi$. Для прямолінійної характеристики спрацювання, що проходить через точки $\underline{a}$ і $\underline{b}$, область дії знаходиться зліва від прямої при русі від $\underline{a}$ до $\underline{b}$, якщо $-\pi<\arg\dfrac{\underline{Z}-\underline{a}}{\underline{Z}-\underline{b}}<0$, і справа від прямої при $0<\arg\dfrac{\underline{Z}-\underline{a}}{Z-b}<\pi$;

3) при $\alpha_1-\beta\neq\pi$ характеристика спрацювання визначається в залежності від значень $\underline{k}_1-\underline{k}_4$ двома дугами кіл, що перетинаються, або комбінаціями дуг і відрізків прямих;

4) спрощення при побудові дистанційного органу можливі за прямолінійної характеристики спрацювання ($\underline{a}=\infty$ або $\underline{b}=\infty$, тобто $\underline{k}_3=0$ або $\underline{k}_1=0$) або при характеристиці, що проходить через початок координат ($\underline{a}=0$ або $\underline{b}=0$, тобто $\underline{k}_3=0$ або $\underline{k}_1=0$) Найбільш часто при виконанні дистанційного органу застосовуються «косинусний» або «синусний» алгоритми порівняння, наприклад:

$$-\frac{\pi}{2}\le\arg\underline{w}\approx\varphi\le\frac{\pi}{2};\tag{2.50}$$

$$0\le\arg\underline{w}\approx\varphi\le\pi.\tag{2.51}$$

При виконанні в цьому випадку спрямованого дистанційного органу характеристикою спрацювання, що проходить через початок координат, одну особливу точку, наприклад $\underline{a}$, найпростіше розташувати на початку координат, а іншу – на кінці діаметра кола, що проходить через початок координат (рис. 2.23, *б*).

При цьому $\underline{k}_4=0;-\underline{k}_2/\underline{k}_1=\underline{b}=\underline{Z}_y$ і до схеми порівняння підводяться величини:

$$\underline{E}_1=\underline{k}_1\underline{U}_p+\underline{k}_2\underline{I}_p;\ \underline{E}_2=\underline{k}_3\underline{U}_p.\tag{2.52}$$

За виконання дистанційного органу з прямолінійної характеристикою спрацювання (рис. 2.22, характеристика 2), прийнявши $\underline{a}=\infty$, маємо $\underline{k}_3=0$, тобто $\underline{E}_1=\underline{k}_1\underline{U}_p+\underline{k}_2\underline{I}_p;\ \underline{E}_2=\underline{k}_4\underline{U}_p$. Точку $\underline{b}$ можна розташувати в будь-якому місці прямий.

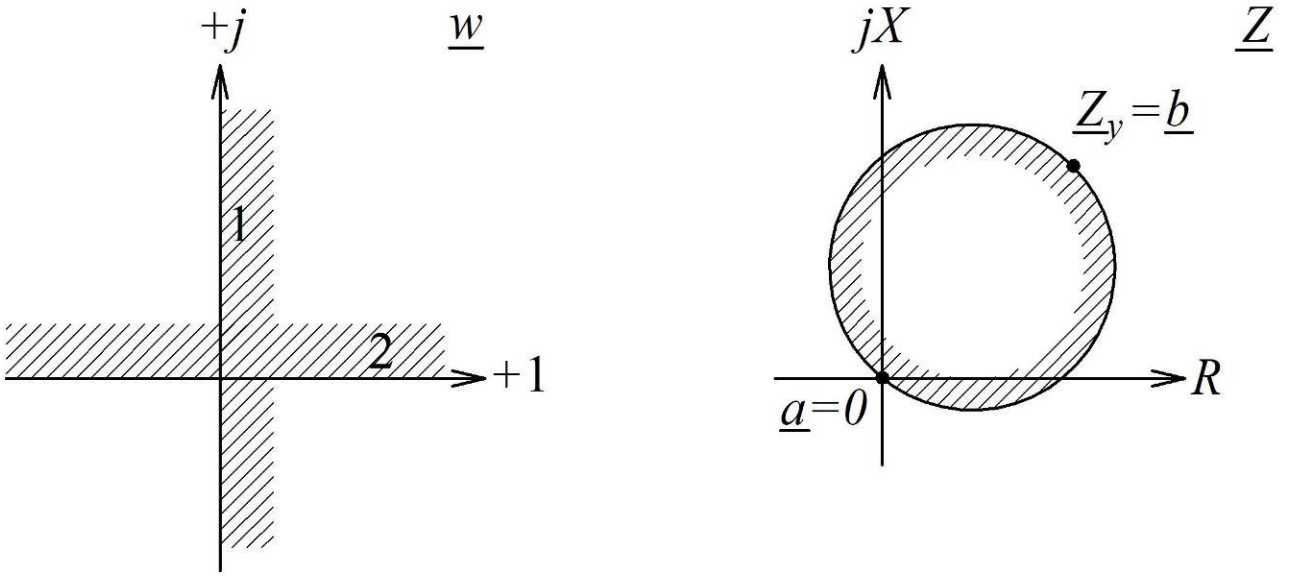
Цифрова реалізація дистанційного органу здійснюється на основі структурної схеми рис. 2.21. При цьому «косинусний» та «синусний» алгоритми порівняння може бути одержаний з аналогічного за структурою алгоритму реле напрямку потужності (2.27) відповідно при $\alpha_1=\pi/2$, $\alpha_1=0$ та заміні $\underline{U}(nT)$ на $\underline{E}_2(nT)$, $\underline{I}(nT)$, на $\underline{E}_1(nT)$. В результаті з виразу (2.32) отримаємо:

для «косинусного» алгоритму порівняння ($\alpha_1 = -\pi/2$):

$$E_{1x}(nT)E_{2x}(nT) + E_{1y}(nT)E_{2y}(nT) \geq 0; \qquad (2.53)$$

для «синусного» алгоритму порівняння ($\alpha_1 = 0$):

$$E_{2y}(nT)E_{1x}(nT) - E_{2x}(nT)E_{1y}(nT) \geq 0. \qquad (2.54)$$



а) *б*)

Рис. 2.23. Характеристики спрацьовування дистанційного органу:
а – у площині $\underline{w}$; б – у площині $\underline{Z}$; 1 – «косинусна» характеристика спрацювання;
2 – «синусна» характеристика спрацювання

2.4.5. Питання для самоперевірки

1. Які величини використовуються при побудові пофазних дистанційних органів?
2. В чому різниця між пофазними і трифазними дистанційними органами?
3. Поясніть принцип дії цифрових дистанційних органів на основі порівняння абсолютних значень електричних величин.
4. Проаналізуйте структуру дистанційного органу на основі порівняння абсолютних значень двох електричних величин.
5. Поясніть принцип побудови дистанційних органів на основі порівняння фаз двох електричних величин.

ГЛОСАРІЙ

А

Аналогова мікросхема – мікросхема, що виконує функції перетворення і оброблення аналогових електричних сигналів, тобто сигналів, що змінюються за законом неперервної функції в діапазоні від додатного до від'ємного рівня напруги живлення.

Аналого-цифровий перетворювач – мікроелектронний пристрій, що перетворює аналоговий сигнал в цифровий код.

Аналого-цифрова мікросхема – мікросхема, що поєднують функції цифрового та аналогового оброблення сигналів.

Арифметико-логічний пристрій – один з основних блоків мікропроцесора, який виконує арифметичні та логічні операції над даними (операндами).

Архітектура – організаційна структура мікропроцесорної системи, що визначає організацію та взаємозв'язки функціональних компонентів.

Архітектура фон Неймана – тип архітектури мікропроцесорної системи, що передбачає спільне зберігання даних та машинних команд в комірках однієї пам'яті.

Асемблер – програма, що виконує трансляцію програми, яка написана з використанням мнемокодів, у виконуваний машинний код.

Б

Базове значення – максимальне значення числа в конкретній системі числення, яке можна подати одним розрядом.

Байт – одиниця інформації розміром 8 біт.

Біт – мінімальна одиниця інформації, що приймає значення «0» або «1».

Брязкіт контактів – явище випадкових переривань електричного контакту, що супроводжує комутацію всіх типів механічних контактів. Тривалість – до 40 мс. Негативне явище, яке може призвести до неправильного розпізнавання стану кнопки (механічного давача).

В

Вартовий таймер – пристрій, який запобігає зависанню мікроконтролера.

Вектор скидання – команда безумовного переходу до частини ініціалізації програм, яка розташована за нульовою адресою пам'яті програм.

Виняткова диз'юнкція (операція XOR, додавання за модулем два) – логічна операція, в результаті виконання якої одержують значення логічної «1» тоді й лише тоді, коли логічній «1» відповідає суто один з її операндів.

Г

Гарвардська архітектура – тип архітектури мікропроцесорної системи, що передбачає наявність окремих запам'ятовуючих пристроїв для збереження даних та кодів програми.

Геркон – механічний контакт, розміщений у герметичній колбі, що замикається під впливом зовнішнього магнітного поля.

Гібридна інтегральна мікросхема – мікросхема, при виготовленні якої використовувалися різні технології для виготовлення окремих елементів.

Д

Двійкова система числення – позиційна система числення, база якої дорівнює двом та використовує для запису чисел тільки два символи: «0» та «1».

Демультиплексор – цифровий комутатор, що забезпечує передачу логічного сигналу з єдиного входу до обраного виходу. Вихід обирається за адресою, яка задається двійковим кодом на адресних входах.

Дешифратор (декодер, decoder) – логічний пристрій, що призначений для перетворення двійкового вхідного коду у однопозиційний вихідний код.

Диз'юнкція (операція АБО, OR) – логічна операція, в результаті виконання якої одержують значення логічної «1», якщо хоча б один з операндів має значення логічної «1».

Директива асемблера – ключове слово в тексті програми на мові асемблера, що впливають на процес асемблювання або властивості вихідного файлу, безпосередньо в код не транслюються.

Дискретизація – процес оброблення аналогового сигналу, що передбачає одержання через визначені проміжки часу, які визначаються періодом дискретизації, чисел, що відповідають величинам аналогового сигналу.

Діодно-транзисторна логіка – технологія побудови мікросхем з використанням біполярних транзисторів, резисторів та діодів.

Е

Елемент мікросхеми – частина інтегральної мікросхеми, яка реалізовує функцію елементарного радіоелементу (наприклад, транзистора, діода, резистора, конденсатора), виконується нероздільно від кристала мікросхеми або її підкладки, елемент не можна відокремити від ІМС як самостійний виріб.

Енкодер – механічний або оптико-механічний давач, який використовується для визначення поточного положення або частоти обертання вала.

І

Інверсія – логічна операція над двійковим числом, яке передбачає взаємозаміну логічного «0» та логічної «1».

Інтерфейс – сукупність засобів, методів і правил взаємодії між елементами системи.

К

Квантування – процес оброблення аналогового сигналу, що передбачає визначення дискретних значень, які з певною похибкою відповідають величинам аналогового сигналу, і належать до множини рівнів квантування.

Кварцевий резонатор – п’єзоелектричний резонатор на основі кристалічного елементу з кварцу, використовується для генерації тактових імпульсів.

Компаратор аналогових сигналів – електронна схема, яка забезпечує порівняння двох аналогових сигналів, результат порівняння видається у вигляді двійкової величини.

Компіляція – трансляція програми на машинну мову.

Компонент мікросхеми – частина інтегральної мікросхеми, що до монтажу була самостійним виробом у власному корпусі, теоретично компонент можна відокремити від мікросхеми.

Кон’юнкція (операція І, AND) – логічна операція, в результаті виконання якої одержують значення логічної «1», якщо всі операнди мають значення логічної «1».

Конвеєрна обробка – паралельне виконання процесором різних етапів декількох команд.

Л

Лічильник команд – регістр введення/виведення, що використовується для адресації пам’яті програм.

Логічна функція – функція, один або кілька аргументів якої є логічними виразами.

Логічний елемент — пристрій, призначений для оброблення інформації в цифровій формі.

М

Машинний такт – час, що відповідає одному періоду імпульсів тактового генератора і є основною одиницею виміру часу виконання команд процесором.

Машинний цикл – час, що складається з декількох машинних тактів і відповідає часу виконання однієї команди.

Мікрозбірка – мікроелектронний виріб, що включає елементи, компоненти, окремі інтегральні схеми тощо, які з'єднані між собою.

Мікроконтролер – програмований обчислювальний пристрій, що має набір периферійних пристроїв і використовується для вирішення завдань управління в технічних системах.

Мікропроцесор – мікроелектронний пристрій, що забезпечує приймання, оброблення (відповідно до програми) та видачу інформації.

Мікропроцесорна система – обчислювальна, вимірювальна або управляюча система, оброблення інформації в якій забезпечується мікропроцесором.

Мікросхема (інтегральна мікросхема, integrated circuit) – електронна схема, яка виконана на напівпровідниковому кристалі (чипі) та призначена для виконання певної функції.

Мнемоніка команди – коротке символьне ім'я команди процесора, яка транслюється у виконуваний код.

Мова асемблера – мова програмування низького рівня, мнемонічні команди якої відповідають інструкціям процесора обчислювальної системи.

Мультиплексор – цифровий комутатор, що забезпечує передачу цифрового сигналу з обраного інформаційного входу до єдиного виходу. Вхід обирається за адресою, яка задається двійковим кодом на адресних входах.

Н

Напівпровідникова інтегральна мікросхема – мікросхема, що включає єдиний кристал напівпровідника, в об'ємі та на поверхні якого виконані всі елементи.

О

Оперативний запам'ятовуючий пристрій – енергозалежна швидкодіюча пам'ять, призначена для запису, зберігання та читання інформації під час виконання програми мікропроцесорною системою або мікроконтролером.

П

Паралельний інтерфейс – спосіб обміну даними у мікропроцесорних системах, що передбачає передачу декількох біт (тетрада, байт, слово тощо) в один момент часу. Для передачі кожного розряда використовується окремий провідник.

Переривання – механізм реагування мікроконтролера (мікропроцесора) на внутрішню або зовнішню подію шляхом припинення нормального ходу програми для виконання пріоритетного завдання.

Період дискретизації – час між двома сусідніми вибірковими значеннями вимірюваної величини.

Плівкова інтегральна мікросхема – мікросхема, в якій для виконання елементів використані плівки, які нанесені на поверхню діелектричної підкладки.

Поєднана інтегральна мікросхема – мікросхема, що містить напівпровідниковий кристал, в поверхневому шарі якого виконані активні елементи (як у напівпровідниковій мікросхемі), а пасивні елементи виконані на ізольованій поверхні того ж кристалі за допомогою плівок.

Послідовний інтерфейс – спосіб обміну даними у мікропроцесорних системах, що передбачає передачу в кожен момент часу тільки одного інформаційного біти.

Постійний запам'ятовуючий пристрій – пам'ять енергонезалежного типу, що призначена для довготривалого зберігання даних.

Прапор – двійковий розряд регістра, значення якого визначає певну властивість периферійного пристрою (або центрального процесора) або подію.

Р

Регістр – запам'ятовуючий пристрій на основі тригерів, що забезпечує приймання, запам'ятовування та видачу декількох бітів інформації.

Регістри введення/виведення – комірки оперативної пам'яті, які призначені для конфігурації мікроконтролера (мікропроцесора) і периферійних пристроїв, а також для введення/виведення інформації через порти.

Регістр загального призначення – комірка оперативної пам'яті, яка доступна безпосередньо арифметико-логічному пристрою

Резисторно-транзисторна логіка – технологія побудови мікросхем на основі резисторів та біполярних транзисторів.

С

Сегмент даних – частина програми на мові асемблера, що визначає вміст області пам'яті з даними.

Сегмент коду – частина програми на мові асемблера, що визначає вміст області пам'яті для розміщення команд.

Сегмент стека – частина програми на мові асемблера, що визначає вміст області пам'яті, яка відведена під стек.

Серія мікросхем – сукупність типів інтегральної схеми, які призначені для виконання різних функцій, однак мають єдину технологічну будову і призначені для спільного застосування.

Сигнальний процесор – спеціалізований мікропроцесор, що забезпечує цифрове оброблення сигналів в реальному масштабі часу

Система команд – набір низькорівневих команд, які може виконувати мікропроцесор або центральний процесор мікроконтролера.

Система числення – сукупність правил і знаків, за допомогою яких можна представити будь-яке число.

Скидання (реініціалізація, reset) – операція по переведенню мікроконтролера (мікропроцесора) до вихідного стійкого стану.

Слово – одиниця інформації розміром 16 біт (2 байти).

Стек – спеціальним чином виділена область пам'яті, доступ до якої здійснюється за принципом LIFO (Last In First Out – останнім увійшов, першим вийшов). Значення, записане до стека останнім, буде прочитано першим. Стек використовується для впорядкованого збеження адрес повернення до підпрограм та параметрів, що передаються підпрограмам.

Струмова петля – поширений варіант послідовного інтерфейсу, що передбачає передачу логічних сигналів рівнями струму по замкненому колу між передавачем та приймачем.

Т

Таблиця істинності – перелік всіх можливих станів входів двійкового пристрою з відповідними вихідними значеннями.

Тактовий генератор – джерело прямокутних імпульсів постійної частоти, які використовуються для синхронізації внутрішніх команд, що виконуються центральним процесором, і передачі інформації по системній шині.

Транзисторно-транзисторна логіка – технологія виготовлення мікросхем, що передбачає використання біполярних транзисторів для реалізації логічних функцій, інвертування і посилення сигналу на виході.

Тригер – елементарна комірка пам'яті, що призначена для зберігання одного біта інформації.

Ц

Цикл виконання команди – послідовність операцій, що здійснюються мікропроцесором при виконанні однієї машинної команди

Цифро-аналоговий перетворювач – мікроелектронна схема, що призначена для перетворення цифрового сигналу в аналогову форму.

Цифровий вимірювальний орган – мікропроцесорний пристрій, що здійснює цифрове оброблення сигналів за заданим алгоритмом для аналізу стану електроенергетичного об'єкта, що захищається.

Цифровий релейний захист – мікропроцесорний пристрій, що забезпечує захист електромереж та електротехнічних об'єктів від аварійних режимів.

Цифрова мікросхема – мікросхема, що призначена для оброблення сигналів, зміна яких відповідає дискретній функції.

Ч

Черга команд – область надоперативної пам’яті мікропроцесора, до якої записуються одна або декілька команд безпосередньо перед їх виконанням.

Чип – напівпровідникова структура, на поверхні якої сформовані контактні площинки.

Ш

Шина – група паралельних провідників, за допомогою яких дані передаються від одного пристрою мікропроцесорної системи до іншого.

Шина адреси – шина у складі мікропроцесорної системи, що використовується для визначення адреси елемента в оперативній пам’яті, до якого у даний момент відбувається звернення з боку центрального процесора або пристроїв введення-виведення.

Шина даних – шина у складі мікропроцесорної системи, що використовується для обміну команд і даних між центральним процесором (пристроями введення-виведення) та оперативною пам’яттю.

Шина управління – шина у складі мікропроцесорної системи, що використовується для передачі спеціальних сигналів, які синхронізують роботу усіх пристроїв, підключених до системної шини.

Шифратор (кодер, encoder) – логічний пристрій, який забезпечує перетворення позиційного n-розрядного коду в m-розрядний двійковий код.

ЛІТЕРАТУРА

1. Колонтаєвський Ю. П., Сосков А. Г. Електроніка та мікросхемотехніка : підручник. 2-е вид. К. : Каравела, 2009. 416 с.

2. Жуйков В. Я., Терещенко Т. О., Ямненко Ю. С., Заграничний А. В. Мікропроцесорна техніка : підручник ; ред. О. В. Борисов. Київ : НТУУ «КПІ», 2016. 440 с.

3. Електроніка і мікропроцесорна техніка / Сенько В. І. та ін. К. : «Агроосвіта», 2015. 676 с.

4. Сучасні мікроконтролери в електронній та інформаційно-вимірювальній техніці : навч. посіб. / Вовна О. В. та ін. Покровськ : ДВНЗ «ДонНТУ», 2020. 311 с.

5. Промислові мережі та інтеграційні технології в автоматизованих системах : навч. посіб. / Пупена О. М. та ін. К. : Вид-во «Ліра-К», 2011. 552 с.

6. Яндульський О. С., Дмитренко О. О. Релейний захист. Цифрові пристрої релейного захисту, автоматики та управління електроенергетичних систем : навч. посіб. К. : НТУУ «КПІ», 2016. 102 с.

7. Thorpe E. Arduino: Advanced Methods and Strategies of Using Arduino. Independently Published, 2020. 224 p.

8. Geddes M. Arduino Project Handbook. 25 Practical Projects to Get You Started. San Francisco, 2016. 275 p.

9. Кідиба В. П. Релейний захист електроенергетичних систем : підручник. Львів : Видавництво Національного університету «Львівська політехнік», 2013. 533 с.

10. Хіхловська І. В., Антонов О. С. Обчислювальна техніка та мікропроцесори : підручник. 2-ге вид. Одеса, 2011. 440 с.

11. Поджаренко В. О., Кучерук В. Ю., Севастьянов В. М. Основи мікропроцесорної техніки : навч. посіб. Вінниця : ВНТУ, 2006. 226 с.

12. Грищук Ю. С. Мікроконтролери: Архітектура, програмування та застосування в електромеханіці : навч. посіб. Харків : НТУ «ХПІ», 2019. 384 с.

Навчальне видання

Василець Святослав Володимирович
Василець Катерина Сергіївна
Килимчук Антон Володимирович

МІКРОПРОЦЕСОРНА ТЕХНІКА В СИСТЕМАХ ОБЛІКУ ЕНЕРГІЇ ТА РЕЛЕЙНОМУ ЗАХИСТІ

Навчальний посібник

Друкується в авторській редакції

Технічний редактор *Г.Ф. Сімчук*

Видавець і виготовлювач
Національний університет
водного господарства та природокористування,
вул. Соборна, 11, м. Рівне, 33028.

Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру видавців, виготівників і розповсюджувачів видавничої продукції РВ № 31 від 26.04.2005 р.