МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»

В.Г. Джулгаков, К.І. Руденко

ПРОЕКТУВАННЯ ЦИФРОВИХ КОНТРОЛЕРІВ

Навчальний посібник

Харків «ХАІ» 2008

УДК 004.384:004.4’236:004.454

Джулгаков В.Г. Проектування цифрових контролерів: навч. посібник / В.Г. Джулгаков, К.І. Руденко. – Х.: Нац. аерокосм. ун-т «Харк. авіац. ін-т», 2008. – 100 с.

Детально розглянуто різноманітні варіанти побудови систем збирання даних і керування технічними об’єктами на основі однокристальних мікроконтролерів. Особливу увагу приділено поєднанню апаратних і програмно-логічних аспектів проектування контролерів систем керування. Всі наведені приклади апаратних і програмних рішень перевірені авторами на практиці. Даний навчальний посібник є логічним продовженням і доповненням посібника «Микроконтроллерные системы: структуры и практическое применение».

Для студентів технічних спеціальностей при вивченні принципів побудови та програмування цифрових систем керування на основі мікроконтролерів як матеріал лекцій і лабораторних занять, а також при виконанні курсових і дипломних проектів.

Іл. 52. Табл. 6. Бібліогр.: 12 назв.

Р е ц е н з е н т и: академік Академії наук вищої освіти України,
д-р техн. наук, проф. І.О. Фурман,
канд. техн. наук, доц. В.С. Кортнєва

1 ЕТАПИ ПРОЕКТУВАННЯ КОНТРОЛЕРІВ СИСТЕМ КЕРУВАННЯ

1.1 Принципи розробки мікропроцесорних контролерів

Процес проектування контролерів на основі мікропроцесорів і мікроконтролерів формують відповідно до концепції єдності проектування і налагодження апаратної й програмної складових.

Важлива особливість застосування контролерів – робота в масштабі реального часу (МРЧ), тобто гарантоване забезпечення реакції на зовнішні події протягом визначеного інтервалу часу. Вирішення задачі комплексної розробки апаратури і програмного забезпечення (ПЗ) із урахуванням роботи в МРЧ при довільній структурі та схемотехніці контролера є складним, коштовним і довготривалим.

Тому найбільш ефективною є методика проектування на основі типових функціонально-топологічних і програмних (ФТП) модулів. Це означає, що кожний елемент і модуль апаратури виконує завершену типову функцію, що реалізована у вигляді усталеної сукупності компонент (мікросхем, пасивних елементів тощо) на печатній платі з визначеними технологічними нормами, а процедура керування апаратурою реалізована як програмний модуль. Топологія є невід'ємною складовою модуля, оскільки характеристики сучасної прецизійної елементної бази можуть бути реалізовані тільки при правильному конструюванні печатної плати.

Передумовою застосування ФТП-модулів є принцип магістрально-модульної побудови керуючих обчислювачів на основі мікропроцесорів (МП) і мікроконтролерів (МК), як показано на рис. 1.1.

Магістрально-модульна організація передбачає, що пам'ять та інтерфейс введення/виведення виконані у вигляді модулів, що обмінюються інформацією з МП через магістраль. Обробку даних здійснює МП, обмін даними, як правило, виконується під його керуванням та через його внутрішні регістри. На логічному рівні мікропроцесорна система подається у вигляді регістрової моделі. Доступ МП до комірок пам'яті та регістрів модулів введення/виведення здійснюється за адресами, які зведені розробниками у карту пам'яті й пристроїв введення/виведення. Параметри магістралі (розрядність шин адреси, даних і керування, протокол і діаграми обміну) визначаються або певним офіційним стандартом, або параметрами аналогічного МП, який є в конкретному випадку стандартом «де-факто» (наприклад, спільним для МК MCS-51). Параметри магістралі значною мірою визначають граничні характеристики продуктивності мікропроцесорної системи.

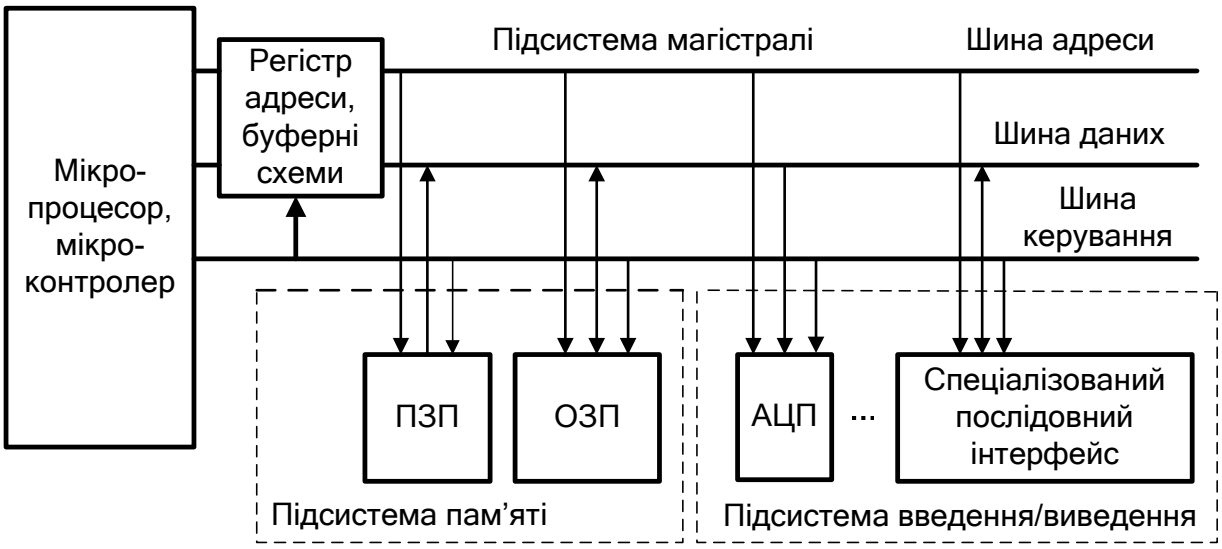

Рисунок 1.1 – Магістрально-модульна побудова МП-системи

Проектування мікропроцесорних контролерів із використанням ФТП-модулів здійснюється в два етапи.

На першому етапі на макетній платі створюють і випробовують сам модуль, проектують ПЗ для його обслуговування. При цьому макетну плату проектують з повноцінною топологією печатних провідників, що забезпечує узгодження хвильового опору і екранування від перешкод, але з додатковими елементами, що підтримують налагодження. Модуль тестується і налагоджується у взаємодії з працездатним процесорним модулем, з яким він надалі буде інтегруватися на платі чи в пристрої контролера.

На другому етапі на основі процесорного ядра та відпрацьованих ФТП-модулів оточення проектується замовлений контролер. Очевидно, що набір ФТП-модулів значною мірою прив'язаний до того процесорного ядра, з яким від налагоджувався. Типовими прикладами підходу на основі ФТП-модулів є сучасні промислові серії контролерів і блоків введення-виведення в стандартах MicroPC, PCI, PC-104.

Особливість МП-контролерів полягає в тому, що вони самі інтегруються у певний об'єкт (тобто належать до категорії "embedded controllers"). Це означає, що перед розробниками МП-системи такого типу стоїть задача повного циклу проектування, починаючи від розробки алгоритму функціонування і закінчуючи комплексними випробуваннями у складі об'єкта, а можливо, і супроводження при виробництві. Основні етапи розробки МП-контролера показані на рис. 1.2.

Формування технічних вимог розпочинає цикл проектування. Можливість програмування МП-контролерів спонукає замовника закласти максимально широкі функції, щоб мати можливість застосовувати контролер для керування цілою низкою однотипних пристроїв чи

об’єктів. Критерієм вибору контролера є економічна обґрунтованість будь-якого збільшення обсягів апаратних засобів. Це визначають в результаті дослідження ринку пристроїв даного типу і максимального поліпшення показника ціна/функціональність. На цьому етапі формують вимоги до типу МП чи МК, що буде використаний в розробці.

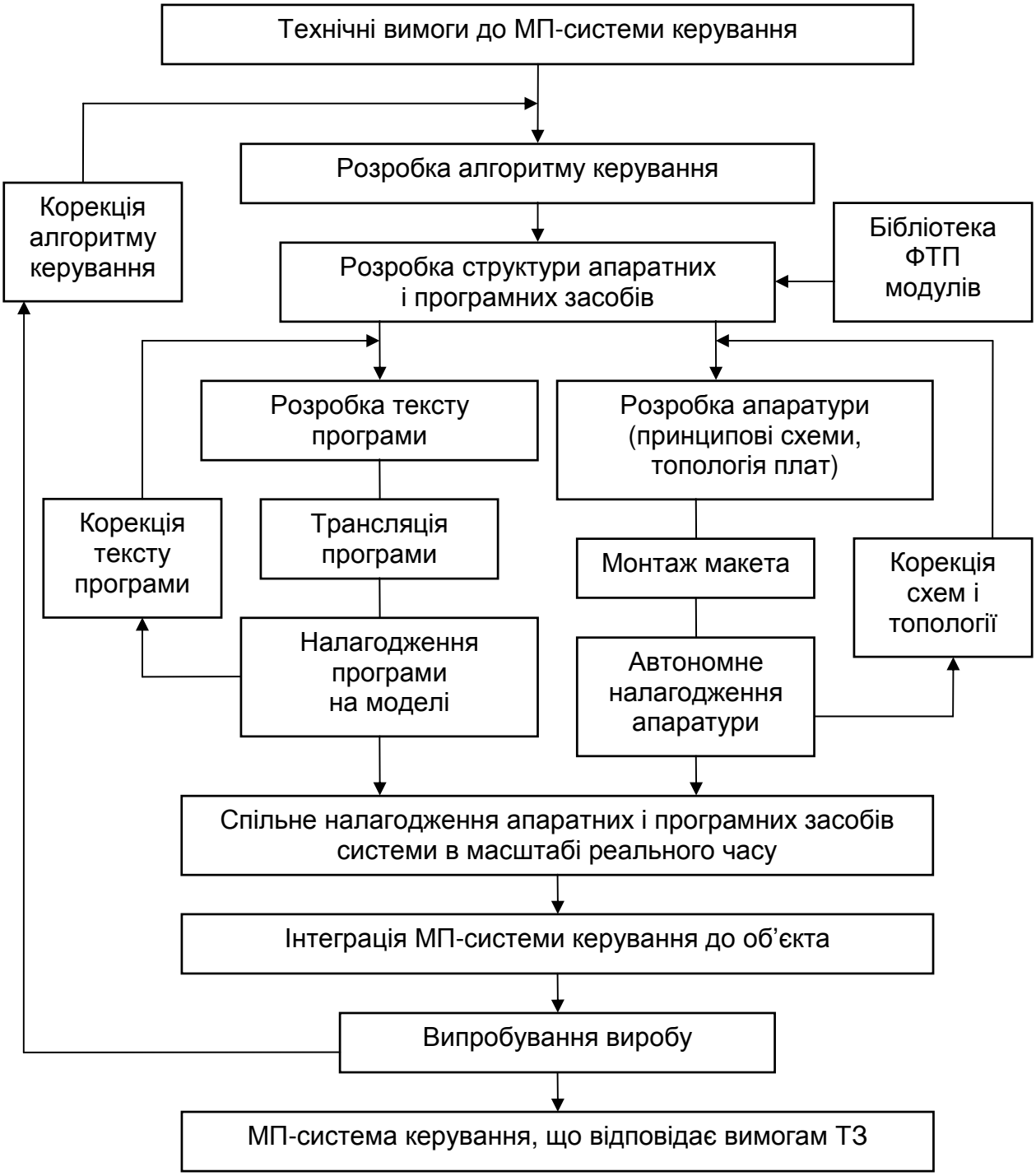

Рисунок 1.2 – Етапи проектування мікроконтролерної системи

Етап розробки алгоритму керування є найвідповідальнішим, оскільки помилки цього етапу виявляються при випробуваннях завершеного виробу і призводять до коштовної переробки всієї системи керування. Зазвичай формують та опрацьовують декілька варіантів алгоритму, що забезпечують виконання технічних вимог з використанням напрацьованих раніше ФТП-модулів. Основні варіанти відрізняються співвідношенням обсягів програмного забезпечення і апаратури. Критерієм вибору є максимальне збільшення програми і зменшення апаратури при забезпеченні заданих показників швидкодії та надійності в повному діапазоні експлуатаційних параметрів. Часто визначною вимогою є можливість розміщення коду програми керування у внутрішній пам'яті МК, що дозволяє зменшити апаратні витрати на зовнішню пам'ять і забезпечити захист програми. На цьому ж етапі остаточно визначають тип МП чи МК і найважливіших схем оточення (флеш-пам'ять, ПЛІС, контролери інтерфейсів, АЦП і т. ін.).

На **етапі розробки структури МП-контролера** остаточно визначають склад наявних апаратних модулів і таких, що їх слід спроектувати, протоколи обміну між модулями, типи роз'ємів. Для вбудованих контролерів здійснюють попередню розробку конструкції плат. У частині ПЗ визначають склад і зв'язки програмних модулів, мови програмування. На цьому ж етапі вибирають засоби проектування та налагоджування апаратури і ПЗ.

Зміст **етапів розробки тексту програми, трансляції та налагодження** логічних зв'язків на моделі апаратури суттєво залежить від використаних системних засобів. Сьогодні існує підтримка програмування 8-розрядних МК мовами високого рівня (С, Паскаль). Це дозволяє використовувати всі переваги структурного програмування, формувати ПЗ як проект із застосуванням модулів, що транслюються окремо. В той же час продовжують широко використовувати мови рівня Асемблера, особливо коли слід забезпечити контрольовані інтервали часу. Задачі попередньої обробки даних часто потребують застосування обчислень з плаваючою точкою, складних функцій. Тому зараз потужними засобами розробки ПЗ для контролерів є інтегровані середовища програмування (крос-системи) на основі мов високого рівня та візуального формування алгоритмів (детальніше див. розд. 3). Такі системи дозволяють суттєво скоротити витрати часу на створення та корекцію ПЗ, оскільки з рис.1.2 видно, що ці етапи формують внутрішній, найчастіше повторюваний цикл у послідовності етапів розробки МП-системи.

Інший внутрішній цикл, що виконується паралельно, складають **етапи створення апаратури**: розробка загальної принципової схеми

і топології плат, монтаж макета і його автономне налагодження. Ці етапи можна вважати закінченими після того, як «оживає» магістраль мікропроцесорної системи і через неї можна звертатися до пам'яті та блоків введення/виведення. Час виконання цих етапів залежить від наявного набору апробованих ФТП-модулів і кваліфікації розробників. Поширеними системами проектування, що застосовують на етапі формування принципової схеми і розробки топології, є ACCEL EDA та OrCad. Ефективність їх використання значною мірою залежить від наявності потрібних бібліотек елементів, що застосовують у схемах.

Етап спільного налагодження апаратури і ПЗ у масштабі реального часу є найбільш трудомістким і обов'язково потребує застосування таких високотехнологічних засобів розробки (development tools), як схемний емулятор, емулятор ПЗП, логічний аналізатор і генератор програмованих послідовностей. Вибір засобів обумовлений застосованим методом налагодження. Етап завершується, коли апаратура й програмне забезпечення разом дозволяють виконувати всі кроки алгоритму роботи системи. Наприкінці етапу код програми керування «зашивають» за допомогою програматора в енергонезалежну пам'ять, і перевіряється робота контролера без участі емулятора. Налагодження на цьому етапі ведуть в лабораторних умовах із живленням від джерела, що забезпечує максимальний захист апаратури. Частину зовнішніх джерел і споживачів інформації моделюють.

Етап інтеграції контролера в об'єкт полягає в повторенні робіт із спільного налагодження апаратури та керуючої програми, але при роботі у власному відсіку об'єкта, живленні від штатного джерела, з інформацією від штатних пристроїв і датчиків. Проблеми виникають через електромагнітну несумісність виконавчих пристроїв, що розроблені раніше, з МП-системою керування. Багато часу витрачається на ліквідацію однократних відмов. Цю проблему вирішують за допомогою програмного резервування, але за наявності резерву пам'яті програм. На цьому ж етапі здійснюють калібрування пристрою із занесенням параметрів до флеш-пам'яті.

Випробування об'єкта з МП-контролером можна розділити на комплексні та спеціальні. Особливість комплексних випробувань полягає у тому, що для спостереження за МП-контролером у реальних умовах не завжди можна застосувати лабораторні засоби налагодження. Автономні налагоджувальні засоби менш розвинені й суттєво дорожчі. Спеціальні випробування (на електромагнітну сумісність, кліматичні та ін.) проводять за звичайними методиками. Після успішного проведення випробувань формують файл з остаточною версією коду програми для програматора або заводу-виробника МК.

1.2 Загальна характеристика однокристальних мікроконтролерів

Однокристальні мікроконтролери (МК) є однією із функціональних груп сучасної мікропроцесорної техніки. Історично поява МК була викликана розвитком систем малої автоматизації промисловості й побутових пристроїв і необхідністю застосування малогабаритних повнофункціональних систем збирання і обробки інформації, пристосованих для вирішення задач керування обладнанням. Саме тому за даним класом мікропроцесорних пристроїв закріпилась назва “мікроконтролер” – мініатюрний пристрій для керування.

Основною ознакою МК є інтеграція на одному кристалі всіх функціональних блоків, що зазвичай наявні у мікропроцесорній системі:

– процесор для обробки цифрових даних під керуванням програми;

– генератор тактових імпульсів для синхронізації роботи блоків МК;

– пам'ять програм для зберігання кодів програми, що виконується;

– пам'ять даних для зберігання оперативної інформації;

– порти паралельного введення-виведення для обміну даними із зовнішніми пристроями (перетворювачами сигналів, джерелами бітової інформації, різноманітними індикаторами);

– порт послідовного зв’язку для пересилання даних між мікроконтролерами чи ПЕОМ на значній відстані;

– таймери для реалізації функцій вимірювання часу та підрахунку зовнішніх імпульсів;

– контролер переривань для обробки сигналів про асинхронні події.

Всі блоки МК інтегровані між собою через внутрішню шину і мають спільні принципи програмного настроювання. Завдяки цьому досягається висока ефективність роботи МК і зручність його програмування. Традиційно МК поділяють на декілька функціональних груп:

МК для задач логічного керування – мають спеціальні апаратні ресурси для ефективного зберігання бітової інформації та спеціалізовані команди для роботи із бітами.

Системи збирання даних (аналогові процесори) – МК, що містять на кристалі додаткові блоки для введення, перетворення та виведення аналогових сигналів: багатоканальні аналого-цифрові перетворювачі, цифро-аналогові перетворювачі та вихідні широтно-імпульсні модулятори (ШІМ).

Процесори цифрової обробки сигналів (DSP) – спеціалізовані МК для ефективної реалізації алгоритмів цифрової фільтрації. Такі алгоритми формують як послідовність операцій множення та додавання. Отже, в структурі DSP традиційно наявні спеціальні апаратні блоки множення-додавання, побудовані за конвеєрним принципом,

щоб результат попередньої операції автоматично ставав вхідним операндом наступної. Сучасні DSP містять також на одному кристалі із процесором багатоканальні швидкодіючі АЦП.

Більшість моделей мікроконтролерів традиційно належать до класу 8-розрядних пристроїв. Це пояснюється тим, що в багатьох задачах керування обладнанням цілком достатньо точності, що забезпечується при такій розрядності вхідних і вихідних даних контролера. За необхідності розрядність самої цифрової обробки підвищують алгоритмічним шляхом.

Обсяги пам’яті в МК логічного керування відносно малі.

Резидентна (внутрішня) пам'ять програм (РПП) – реалізована на кристалі МК як ППЗУ, РППЗУ або Flash з обсягами від 4 до 128 Кбайт.

Резидентна (внутрішня) пам'ять даних (РПД) – статичний ОЗП на кристалі МК із різними функціональними областями загальним обсягом від 256 байт до 1 Кбайт.

Зовнішня пам'ять програм (ЗПП) і зовнішня пам'ять даних (ЗПД) – адресні області, які можуть бути додатково реалізовані в обчислювальній системі на базі МК. Не всі моделі 8-розрядних МК підтримують роботу із зовнішньою пам’яттю.

Системи збирання даних (аналогові процесори) мають збільшені обсяги пам’яті для зберігання інформаційних масивів.

Традиційна сфера використання МК логічного керування та МК збирання даних – малогабаритні системи автоматичного керування технологічним обладнанням і рухомими об’єктами (автотранспорт, мала авіація), а також побутова техніка.

В останні роки були розроблені моделі 16-бітових і 32-бітових МК (наприклад, групи TMS та ARM). За швидкодією і обсягами ресурсів пам’яті, що може бути адресована, ці МК вже наближаються до молодших моделей процесорів універсальних ПЕОМ. Сферою використання МК груп TMS та ARM є багатофункціональні контролери складних технічних об’єктів у промисловості та енергетиці, маршрутизатори інформаційних мереж, системи обробки зображень на рухомих об’єктах і роботах.

Обов’язковим елементом всіх типів МК є таймери. Навіть у перших моделях МК були наявні два таймери із програмним вибором функцій. В сучасних моделях МК кількість таймерів доведено до 10-12. Вони функціонують у режимах вимірювання часу, підрахунку сигналів, генерації частотно- та широтно-модульованих імпульсних послідовностей.

2 АРХІТЕКТУРА І ПРИНЦИПИ РОБОТИ МІКРОКОНТРОЛЕРІВ

Архітектуру та функціонування мікроконтролерів (МК) коротко розглянемо на прикладі групи MCS-51 – популярних пристроїв для побудови цифрових регуляторів, особливо для задач логічного керування. Базовий МК цієї групи розробки Intel має позначення 80C51.

2.1 Електричний інтерфейс і архітектура базового МК MCS-51

Основними електричними параметрами МК MCS-51 є напруга живлення +5В, рівні вхідних і вихідних сигналів – ТТЛ і базова тактова частота – 12 МГц. Тип корпусу для моделі 80C51 – DIP-40 (із дворядними виводами). Розподіл і призначення виводів МК 80C51 показано на рис. 2.1.

На кристалі базового МК групи MCS-51 реалізовані обчислювальне ядро (внутрішній восьмирозрядний процесор), спільне для всіх МК даної групи, та блоки периферійних функцій (рис. 2.2). Для всіх периферійних блоків у складі МК можна програмно встановлювати потрібні режими роботи.

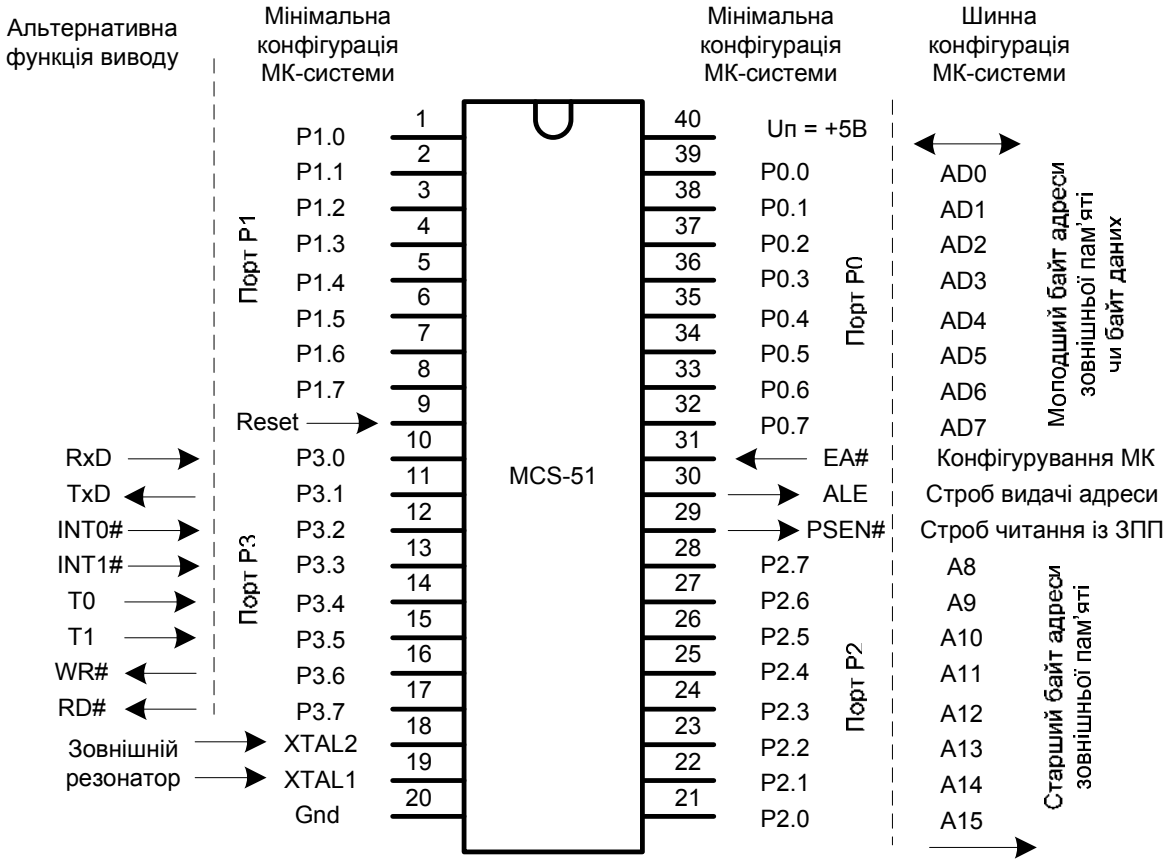

Рисунок 2.1 – Електричний інтерфейс базового МК MCS-51

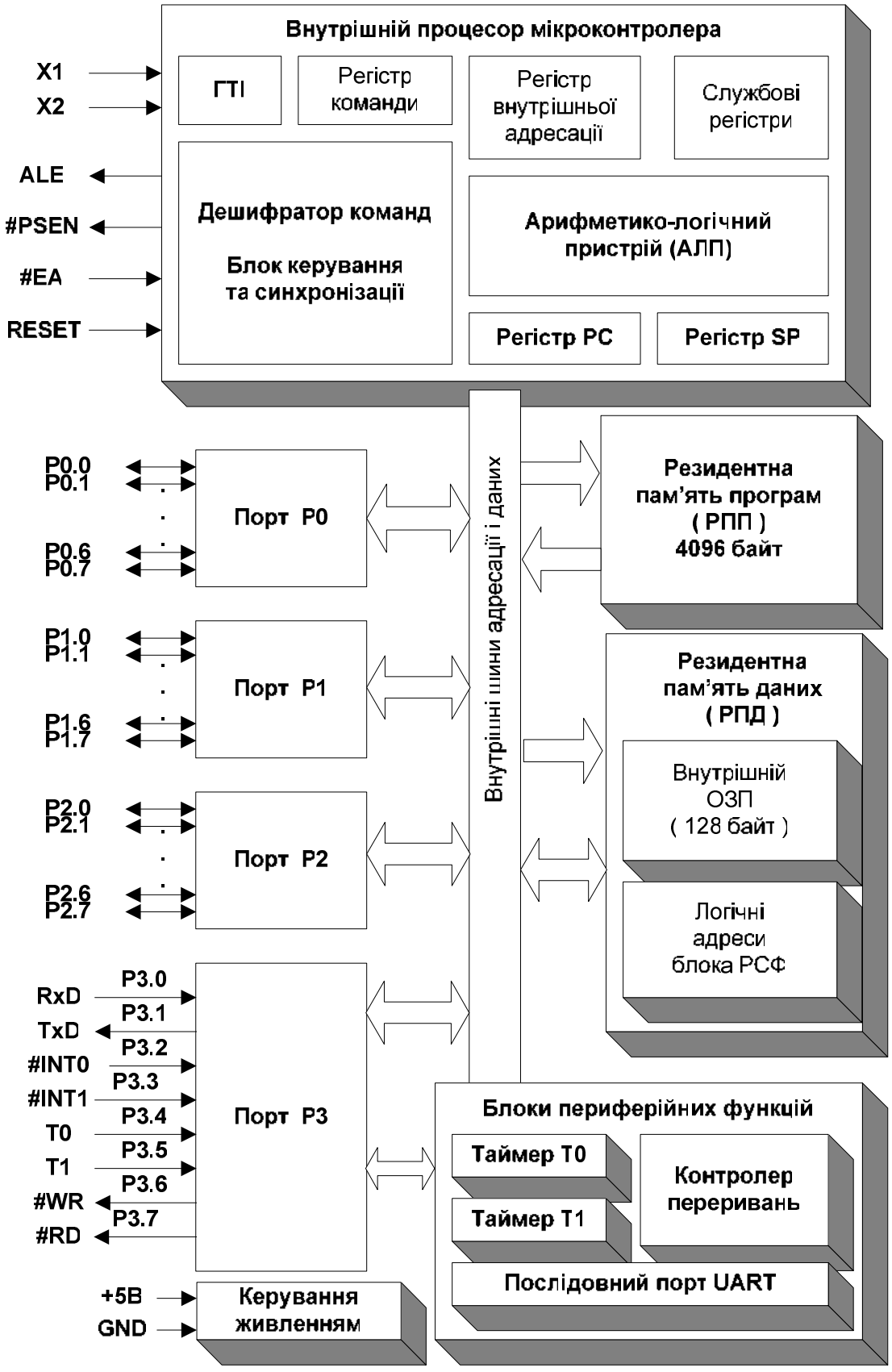

Рисунок 2.2 – Структура базового мікроконтролера групи MCS-51

Як видно із рис. 2.1, обчислювальна система на основі мікроконтролера (МК-система) може бути побудована у двох варіантах:

1) *мінімальна конфігурація* – всі зовнішні пристрої системи (датчики, виконавчі пристрої, елементи індикації та керування) підключаються безпосередньо до портів МК чи до окремих виводів портів через додаткові буферні елементи (наприклад, шинні формувачі); така конфігурація може бути реалізована, якщо кількість зовнішніх пристроїв співвідноситься із кількістю портів;
2) *шинна конфігурація* – на основі портів P0 і P2 можуть бути організовані системні шини адреси і даних; всі зовнішні пристрої системи підключаються до системної шини даних через регістри, що адресуються процесором за допомогою шини адреси; у такий же спосіб у системі може бути реалізована зовнішня додаткова пам'ять.

У мінімальній конфігурації всі лінії портів безпосередньо доступні для використання в програмі.

У шинній конфігурації функції портів P0 і P2 регламентовані (видача адреси та передача байта даних). Крім того, дві лінії порту P3 використовують для керування зовнішніми пристроями:

- сигнал WR# – строб запису у зовнішній пристрій виведення;
- сигнал RD# – строб читання із зовнішнього пристрою введення.

Незалежно від конфігурації інші лінії порту P3 можуть виконувати так звані альтернативні (або периферійні) функції:

- RxD – вхідний сигнал послідовного порту UART (*receive data*);
- TxD – вихідний сигнал послідовного порту UART (*transmit data*);
- INT0#, INT1# – входи зовнішніх запитів на переривання програми;
- T0, T1 – входи для імпульсів, які можуть підраховувати таймери.

Призначення інших сигналів (ALE, Reset тощо) розглянуто у [3].

Як видно з рис. 2.2, всередині МК реалізовані дві окремі області пам'яті – резидентні пам'ять програм і пам'ять даних (РПП і РПД). Крім того, до мікроконтролера у разі необхідності можна підключити зовнішні мікросхеми пам'яті або регістри введення-виведення у логічному обсязі 64К для програм і 64К для даних.

Резидентна пам'ять програм має неоднорідну структуру, в якій виділяють області внутрішнього ОЗП (адреси 00H–7FH) та область адрес, що відносяться до регістрів спеціальних функцій (адреси 80H–FFH). Детально склад РПД описано у [3].

Регістри спеціальних функцій (РСФ) – це логічна сукупність комірок пам'яті, які фізично входять до складу різних блоків мікроконтролера – АЛП, таймерів, портів, контролера переривань та ін. Програмне настроювання периферійних блоків МК детально розглянуто у посібнику [3].

МК групи MCS-51 належать до класу систем з апаратним принципом керування. Це означає, що набір елементарних дій процесора з обробки даних є фіксованим і визначається тільки апаратною структурою дешифратора команд і блока керування МК. Повна система команд МК MCS-51 наведена у посібнику [3].

2.2 Структура й робота портів введення-виведення МК

Порти введення-виведення (ВВ) – це буферні комірки пам'яті в адресному просторі МК. Функціонально вони схожі на внутрішні регістри чи звичайні комірки пам'яті МК, але одна група інформаційних ліній порту підключена до внутрішньої шини МК, а інша – до зовнішніх виводів МК (див. рис. 2.2).

Порти ВВ використовують для тимчасового зберігання даних під час передавання інформації між МК і зовнішніми пристроями.

Для програміста порти ВВ виглядають як звичайні комірки пам'яті. Таким чином, у програмі звертання до будь-якого зовнішнього пристрою реалізується як звертання до порту введення-виведення, до якого підключений цей зовнішній пристрій (див. рис. 2.3).

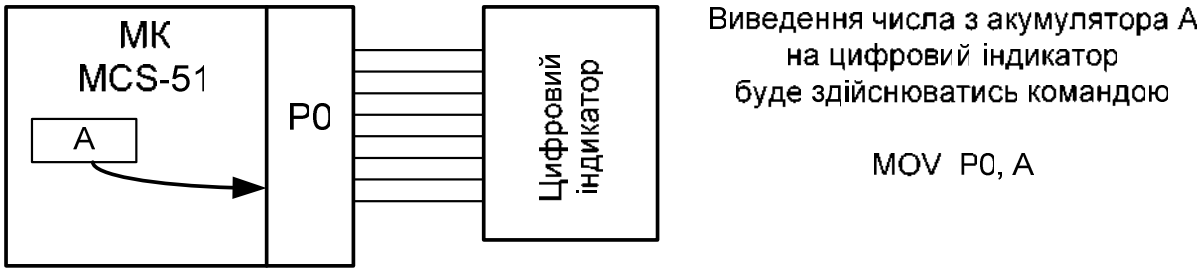

Рисунок 2.3 – Виведення даних через порт на зовнішній пристрій

МК MCS-51 має 32 лінії введення-виведення, об'єднані в чотири восьмирозрядних паралельних порти P0, P1, P2 та P3. Кожну лінію можна використовувати як для введення, так і для виведення даних під керуванням програми.

Кожний порт ВВ містить буферний підсилювач з третім станом і регістр-фіксатор. Саме цей регістр має ім'я P0, P1, P2 чи P3.

2.2.1 Використання портів у мінімальній конфігурації МК-системи

Як було зазначено у розд. 2.2, обчислювальна система на основі МК (МК-система) може бути побудована у двох варіантах – мінімальна та шинна конфігурації.

У мінімальній конфігурації всі зовнішні пристрої системи (датчики, виконавчі пристрої, елементи індикації та керування) підключаються безпосередньо до портів МК чи до окремих виводів портів.

В мінімальній конфігурації МК-системи застосування портів повністю визначається програмістом (що передавати і в якому напрямку). Тому розглянемо два базових режими: введення з порту (*читання порту*) та виведення через порт (*запис у порт*).

Введення з порту – використовується, якщо до порту підключені джерела сигналів, наприклад, кнопки, тумблери, кінцеві датчики чи виходи інших мікросхем.

Дії програміста:

1. Настроїти порт на режим введення – слід записати в порт число FFh (всі “1”): наприклад, для порту P0 це можна здійснити один раз у програмі командою **`MOV P1, #0FFh`**

Таке пересилання настроїть порт на режим введення, тільки якщо до порту підключено джерело сигналу.

2. Далі можна багато разів читати поточну інформацію із зовнішніх ліній порту (тобто від джерела сигналів), наприклад, командами

`MOV A, P1` чи **`MOV R3, P2`**

Слід розуміти, що така команда пересилає в (A) чи в (R3) саме рівні сигналів із зовнішніх ліній МК, а не “одиниці”, що були записані в регістр-фіксатор порту (див. рис. 2.4).

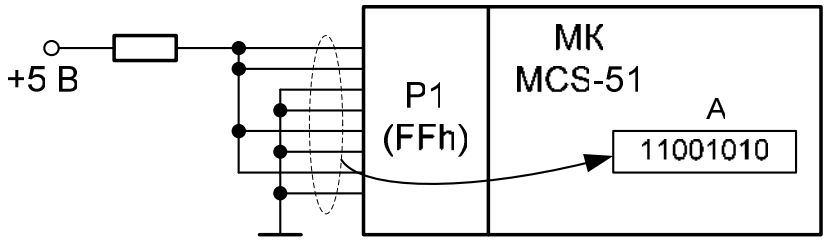

Рисунок 2.4 – Принцип читання інформації з порту

Аналогічні дії можна виконувати для окремих ліній порту, звертаючись до них за допомогою бітового селектора в команді:

```
setb P1.3    ; настроїти лінію 3 порту P1 на введення
mov  C, P1.3; прочитати сигнал на лінії P1.3 в біт C
```

чи

```
jb  P1.3, m1 ; якщо сигнал на P1.3 = “1”, перейти на m1
```

Виведення через порт – використовується, коли до порту підключені приймачі сигналів, наприклад, індикатори, виконавчі пристрої (реле, крокові двигуни) чи входи інших мікросхем.

Дії програміста:

Настроювання порту здійснювати не треба, оскільки режим виведення забезпечується підключенням приймача сигналу до порту.

Виведення через порт здійснюється командами байтового пересилання:

```
MOV P2, A  ; Виведення байта з акумулятора в порт P2
MOV P1, #0FFh  ; Виведення "1" на всі лінії порту P1
```

Порівняйте: при підключенні джерел сигналів остання команда потрібна для настроювання порту на режим введення.

Можна використовувати команди бітових операцій, наприклад:

```
SETB  P0.2 ; Установлення "1" на лінії 2 порту P0
CLR   P3.4 ; Установлення "0" на лінії 4 порту P3
```

Важливо, що інформація, видана в порт, утримується на зовнішніх виводах, доки нова інформація не буде записана в той самий порт.

2.2.2 Використання портів у шинній конфігурації МК-системи

На основі портів P0 і P2 можуть бути сформовані системні шини адреси і даних; всі зовнішні пристрої системи підключають до системних шин через регістри, які адресуються процесором; так само в системі може бути реалізована додаткова зовнішня пам'ять.

Розглянемо програмний аспект використання шинної конфігурації. Основний спосіб пересилання даних у зовнішній адресний простір (у зовнішню пам'ять чи зовнішні регістри введення-виведення) – це використання спеціальних команд:

```
MOVX A,@DPTR ; Читання із зовнішнього АП в акумулятор
MOVX @DPTR,A ; Запис у зовнішній АП із акумулятора
```

У цих командах для вказування комірки зовнішнього АП застосована непряма адресація на основі регістра DPTR [3]. Цей регістр 16-бітовий, тому в нього можна записати довільну адресу комірки зовнішнього АП від 0000h до FFFFh. В таких пересиланнях з боку МК можна використовувати тільки акумулятор.

Наприклад, для пересилання значення 15 в комірку зовнішнього АП з адресою 01A7h слід написати такі команди:

```
MOV  A,#15      ;  Задавання значення 15 в акумуляторі
MOV DPTR,#01A7h; Задавання адреси комірки в регістрі DPTR
MOVX @DPTR,A   ;  Запис у зовнішній АП з акумулятора
```

Для читання даних з комірки 028Ch зовнішнього АП:

```
MOV  DPTR,#028Ch; Задавання адреси комірки в регістрі DPTR
MOVX A,@DPTR    ; Читання із зовнішнього АП в акумулятор
```

Тепер перейдемо до апаратного аспекту шинної конфігурації.

Як тільки МК виконує команду MOVX, адреса, задана в регістрі DPTR, автоматично видається із МК через порти P2 і P0:

- порт P2 видає старшу частину адреси (розряди A8…A15);
- порт P0 видає молодшу частину адреси (розряди A0…A7).

Для наведеного прикладу з адресою 01A7h через порти буде виведена інформація: (P2) = 01h = 00000001b, (P0) = A7h = 10100111b

Важливо, що момент видачі адреси через порти P2 и P0 супроводжується активним рівнем сигналу ALE = 1 (див. рис. 2.5).

Наступні дії процесора МК залежать від команди:

- за командою **`MOVX @DPTR,A`** число із акумулятора видається через порт P0 на зовнішню шину; встановлюється низький рівень сигналу WR# = 0, за рахунок цього здійснюється запис інформації у комірку зовнішнього АП (пам'ять чи регістр); потім встановлюється високий (неактивний) рівень сигналу WR# = 1;
- за командою **`MOVX A,@DPTR`** встановлюється низький рівень сигналу RD# = 0; за рахунок цього здійснюється читання інформації із зовнішнього пристрою чи пам’яті; прочитаний байт через порт P0 надходить в акумулятор; потім встановлюється високий (неактивний) рівень сигналу RD# = 1.

Звертання до зовнішнього АП здійснюється також при читанні кодів команд, якщо в МК-системі реалізована зовнішня пам’ять програм. При цьому адреса комірки зовнішнього АП береться із регістра PC, а замість сигналу RD# мікроконтролер формує сигнал PSEN#=0 (див. рис. 2.5). Порти P2 і P0 використовують так, як при виконанні команд **`MOVX`**. Це автоматично виконується тоді, коли адреса у регістрі PC більше верхньої адреси резидентної пам’яті програм МК.

Таким чином, порт P0 здійснює дві функції – передача молодшої частини адреси зовнішнього АП і пересилання байта даних (такий режим роботи називається мультиплексним). Для того, щоб виділити інформацію про адресу і дані на окремі зовнішні шини, до портів мікроконтролера слід підключити буферні регістри (БР) і шинний формувач (ШФ). Функціональна схема шинного інтерфейсу для MCS-51 і часова діаграма виконання команд типу **`MOVX`** показані на рис. 2.5.

Сигнал ALE використовують для запису адреси в буферні регістри. Таким чином, на входах нижнього БР адреса (A0…A7) присутня тільки на частині інтервалу виконання команди, а на виходах цього регістра – весь проміжок часу звертання до зовнішнього АП.

Блок керування ШФ – це логічна схема, яка на основі сигналів PSEN#, RD# та WR# формує сигнали для переключення напрямку передавання через ШФ (сигнал T) и дозволу його роботи (сигнал

OE#). Логіка побудови блока керування ШФ виражається логічними рівняннями

T = not WR# та *OE# = PSEN# & RD# & WR#.*

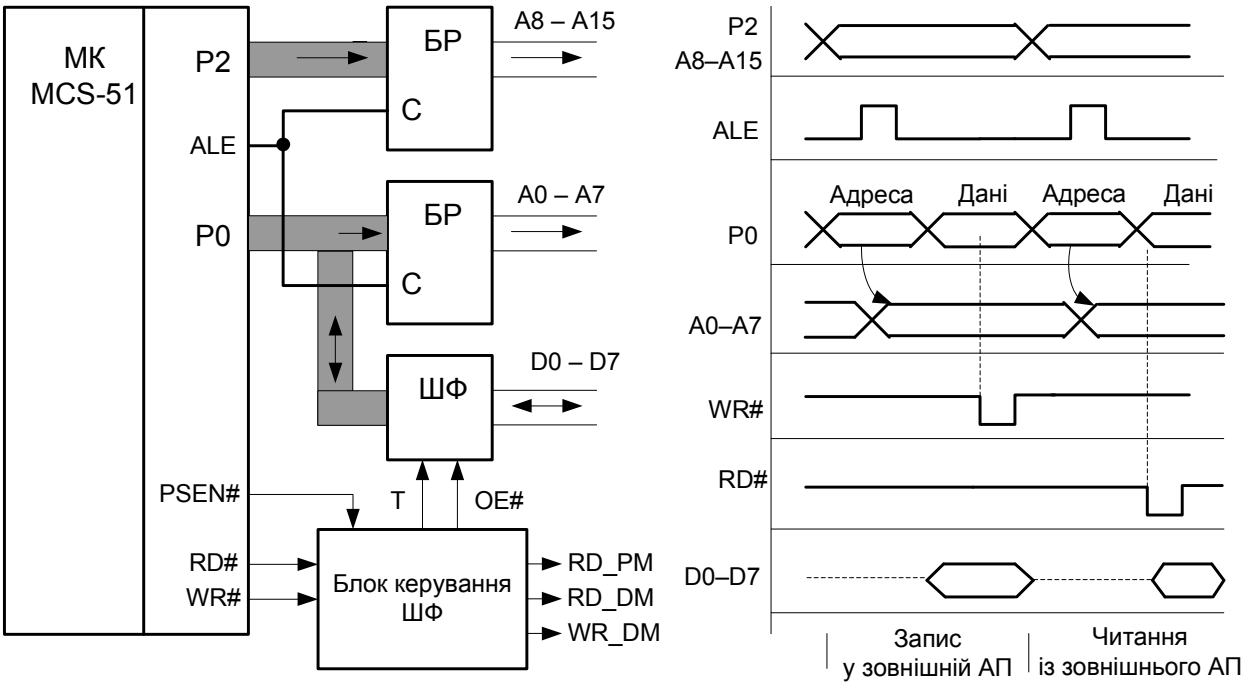

Рисунок 2.5 – Принцип побудови і роботи шинного інтерфейсу МК

Фази запису та читання зовнішнього АП, показані на часовій діаграмі, в реальній системі зазвичай не йдуть одна за одною, оскільки команди запису і читання зовнішнього АП не обов'язково стоять поруч у програмі.

2.3 Сучасні моделі мікроконтролерів MCS-51

Перші МК з ядром MCS-51 були розроблені у 70-х роках XX століття фірмою Intel. З тих часів ядро MCS-51 набуло значної популярності, і сьогодні більше десятка корпорацій виготовляють до півтора сотень моделей МК цієї групи. Хоча фірма Intel згорнула виробництво МК MCS-51, інші відомі виробники (Atmel, Philips, Siemens, Analog Devices, Winbond, Cygnal) продовжують розвивати цей напрямок.

Базові моделі групи відрізнялися типом РПП, що позначалося останніми цифрами у назві моделі: 31 – МК без РПП; 51 – з РПП однократного програмування; 751 – з РПП на основі ПЗП багаторазового програмування з ультрафіолетовим стиранням.

У МК з цифрами 52/752 реалізовано ще один таймер – T2, який функціонує в 16-бітовій конфігурації і забезпечує такі режими, як ав-

топерезавантаження 16-бітового стартового числа, фіксація поточного вмісту лічильних регістрів за зовнішнім сигналом і генерація періодичних імпульсів.

Щоб забезпечити високу зручність перепрограмування МК, сучасні фірми-розробники, зокрема Atmel, створили низку моделей, в яких РПП реалізована як Flash-пам'ять. Такі моделі позначаються цифрами 89 (наприклад, у розробках Atmel – **AT89**xxxxx). Застосування Flash-пам'яті дозволяє здійснювати фізичне перепрограмування МК безпосередньо в цільовій системі. Така технологія отримала назву ISP – *in-system programmability*.

Технологія ISP може бути реалізована в двох варіантах:

- через послідовний інтерфейс SPI – *serial peripheral interface*, для якого застосовані декілька ліній порту P1; SPI реалізований в моделях AT89**S**xxxx; його можна також використовувати для обміну даними між двома та більше мікроконтролерами;
- через послідовний порт UART, який переводиться в режим програмування; ця технологія підтримується додатковим програмним забезпеченням і реалізована, зокрема, в моделі AT89C5115.

Мікроконтролери із позначенням AT89**C**xxxx програмують через паралельний інтерфейс, для якого застосовують практично всі лінії портів. Процедура здійснюється на зовнішніх програматорах, тобто можливість ISP не забезпечена.

Єдиний недолік популярних і зручних МК серії AT89S – відсутність вбудованого АЦП.

Тому значної популярності для побудови систем збирання аналогових даних набули МК **ADuC8xx** фірми Analog Devices. Як правило, в цих моделях, окрім вже названих таймера T2 і розширеної РПД, реалізовані 8-канальний 12-розрядний швидкісний АЦП, два потенціальних 12-розрядних ЦАП, шинний послідовний інтерфейс I^2C.

Багато Flash-мікроконтролерів з ядром MCS-51 розроблено фірмою Philips. У МК групи P89C51Rx+ реалізовані Flash-пам'ять обсягом від 8 до 64К, масиви таймерів, 8- та 10-розрядні АЦП, 8-розрядний ЦАП, ШІМ, інтерфейси програмування – SPI, шинні – I^2C і CAN.

2.3.1 Характеристика мікроконтролера моделі AT89S8252

Дана модель застосована у навчальній мікроконтролерній системі УУМС, що створена і використовується на кафедрі систем управління Національного аерокосмічного університету «ХАІ» [4].

У моделі AT89S8252 реалізовані:

- Flash-пам'ять програм обсягом 8К, що програмується через SPI;

- EEPROM даних обсягом 2К із внутрішнім програмуванням;
- розширений внутрішній ОЗП обсягом 256 байт;
- три 16-бітових таймери (T0, T1 і T2);
- підтримка дев'яти джерел запитів на переривання;
- сторожовий таймер;
- два регістри DPTR для роботи із зовнішнім адресним простором.

Нижні 128 байт ОЗП мають структуру, стандартну для MCS-51. Верхні 128 байт (адреси 80h–FFh) збігаються з адресами області РСФ. Тому регістри з РСФ доступні через пряму адресацію (за іменами регістрів), а байти ОЗП – тільки через непряму адресацію.

Flash-пам'ять програм займає адреси 0000h–1FFFh. При програмуванні через інтерфейс SPI перезаписується весь масив пам'яті.

2.3.2 Керування сторожовим таймером та EEPROM

Сторожовий таймер WDT (*Watchdog Timer*) призначений для попередження «зависання» програми; сигнал переривання від нього викликає «скидання» процесора. Настроювання WDT і керування записом до EEPROM даних здійснюється за допомогою регістра WMCON, який не має бітової адресації (табл. 2.1).

Особливість використання сторожового таймера полягає в тому, що він має бути настроєний на період $T_{WD} > T_{0\ max}$, де $T_{0\ max}$ – максимальний з періодів дискретності обчислень, що реалізуються в даній системі.

У кожному періоді дискретності $T_{0\ max}$ слід виконувати скидання сторожового таймера для того, щоб він починав новий період контролю T_{WD}.

Нехай в мікроконтролерній системі $T_{0\ max} = 100$ *мс*. Тоді період сторожового таймера має становити як мінімум $T_{WD} = 128$ *мс*. Відповідна двійкова комбінація для розрядів [PS2…PS0] = 011.

У початковому блоці програми слід виконати настройку:

```
mov  WMCON,#01100000b
```

Перед початком циклічних дій по алгоритму запускаємо WDT установкою в "1" біта WDTEN:

```
orl  WMCON,#00000001b
```

У підпрограмі, що обслуговує завершення основного періоду дискретності $T_{0\ max}$, треба "скинути" WDT (біт WDTRST = 1):

```
orl  WMCON,#00000010b
```

Таблиця 2.1 – Призначення бітів регістра WMCON

Біт	Номер	Призначення біта
PS2 PS1 PS0	7 6 5	Задавання періоду переповнення сторожового таймера: при “000” – 16 мс, “001” – 32 мс, “010” – 64 мс, “011” – 128 мс, “100” – 256 мс, “101” – 512 мс, “110” – 1024 мс і при “111” – 2048 мс
EEMWE	4	Дозвіл запису в EEPROM. Перед записом біт слід встановити в “1”. По закінченні запису біт треба скинути в “0”
EEMEN	3	Дозвіл доступу до EEPROM. При EEMEN = 1 команди MOVX (від DPTR) працюють з EEPROM, при скиданні в ”0” – із зовнішнім адресним простором
DPS	2	Вибір активного регістра DPTR. При DPS = 0 (базове значення) активний основний DPTR (DP0H:DP0L). При DPS = 1 активний допоміжний DPTR (DP1H:DP1L)
WDTRST **RDY/BSY**	1	Скидання сторожового таймера / ознака зайнятості EEPROM. При запису в цей біт “1” сторожовий таймер скидається, а даний біт автоматично переходить в “0” (це не впливає на стан EEPROM). При читанні цього біта аналізується стан EEPROM: якщо біт дорівнює “0”, то запис у EEPROM не закінчено, якщо “1” – то EEPROM доступна для дій
WDTEN	0	Біт запуску сторожового таймера. При WDTEN = 1 таймер запускається і генерує скидання МК через інтервал часу, визначений бітами (PS2-PS1-PS0)

Доступ до блока пам’яті даних EEPROM здійснюється, як до зовнішнього адресного простору, командами MOVX A,@DPTR та MOVX @DPTR,A за адресами від 0000h до 07FFh (2К). Для вказування, що доступ виконується саме до EEPROM, треба в програмі встановити біт EEMEN:

```
mov  WMCON,#00001000b
```

Особливість доступу до EEPROM полягає в тому, що цикл запису здійснюється довше, ніж до зовнішньої пам’яті (близько 2,5 мс). Тому в програмі треба чекати завершення поточного циклу запису до EEPROM перед наступним доступом. Запис стробується встановленням біта EEMWE = 1. Читання з EEPROM займає стільки ж часу, як і із

зовнішнього АП (2 МЦ). Приклад процедури запису одного байта в EEPROM такий:

```
  mov  A,#data        ; Дані для запису в акумуляторі
  mov  DPTR,#Address  ; Задаємо адресу пам'яті
  orl  WMCON,#00011000b; Установлення EEMWE = EEMEN = 1
  movx @DPTR,A        ; Ініціюємо запис
wt:  mov A,WMCON      ; Цикл чекання завершення запису
  . . .      ;
  jnb  ACC.1,wt       ; Перевіряємо біт RDY/BSY
  anl  WMCON,#11100111b; Скидання бітів EEMWE і EEMEN
```

В циклі очікування можна виконувати інші дії з обробки даних, не пов'язані із доступом до EEPROM чи зовнішнього АП.

2.3.3 Структура й використання таймера T2

Програмна модель таймера T2 містить такі регістри:

- лічильні регістри TH2 та TL2 ("H"– старший, "L"– молодший);
- регістри RCAP2H і RCAP2L для стартового числа або фіксації поточного вмісту лічильних регістрів;
- регістр T2CON для вибору режимів і керування таймером T2;
- регістр T2MOD для задавання додаткових режимів.

Регістри (TH2, TL2) та (RCAP2H, RCAP2L) завжди використовують парами, тому таймер T2 є повністю 16-розрядним.

Для роботи таймера T2 від зовнішніх сигналів служать лінії порту P1: лінія P1.0 (ім'я T2) – вхід імпульсів для режиму лічильника або вихід генератора імпульсів заданої частоти; лінія P1.1 (ім'я T2EX) – вхід керування захватом/перезавантаженням таймера T2.

Основне керування таймером T2 здійснюють за допомогою регістра T2CON, який має бітову адресацію. Структура регістра показана на рис. 2.6, а призначення бітів дано в табл. 2.2. У додатковому регістрі T2MOD використовують тільки два розряди: T2OE (поз. 1) – дозвіл генерації сигналу програмованої частоти від таймера T2 на зовнішній вивід P1.0 та DCEN (поз. 0) – дозвіл задавання напряму лічення таймера T2 в режимі автоперезавантаження стартового числа (при "0" – лічення у бік зростання, детальніше див. [1]).

	7	6	5	4	3	2	1	0
T2CON:	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C_T2	CP_RL2

Рисунок 2.6 – Бітова структура регістра T2CON

Таблиця 2.2 – Призначення бітів регістра T2CON таймера T2

Біт	Призначення біта
TF2	Ознака переповнення таймера T2 (при переході з FFFFh в 0000h). Має скидатися програмно. У режимі генератора для УАПП (при встановленні RCLK=1 або TCLK=1) ознака TF2 не встановлюється
EXF2	Ознака зовнішньої події для таймера T2. Встановлюється по перепаду 1→0 на вході T2EX (лінія P1.1), якщо біт EXEN2=1. Обслуговується як переривання. Має скидатися програмно
RCLK	Вибір джерела синхрочастоти для приймача УАПП у режимах 1 і 3. При RCLK =1 використовують таймер T2, при RCLK=0 – таймер T1
TCLK	Вибір джерела синхрочастоти для передавача УАПП у режимах 1 і 3. При TCLK =1 використовують таймер T2, при TCLK=0 – таймер T1
EXEN2	Дозвіл зовнішньої події для таймера T2. При EXEN2=1 дозволено сприйняття перепаду на вході T2EX і установлення біта EXF2.
TR2	Біт запуску/зупину таймера T2. При TR2=1 таймер запускається і працює відповідно до бітів C_T2, CP_RL2, RCLK і TCLK. При TR2=0 таймер зупиняється
C_T2	Вибір джерела імпульсів для підрахунку: при C_T2=0 – ГТІ МК, при C_T2=1 – лічення імпульсів по входу T2 (лінія P1.0)
CP_RL2	Вибір режиму роботи таймера T2. При CP_RL2 = 0 – режим автоперезавантаження. Стартове число із пари (RCAP2H, RCAP2L) пересилається в лічильні регістри (TH2, TL2) при переповненні таймера (при EXEN2=0) або за перепадом 1→0 зовнішнього сигналу на вході T2EX (при встановленому EXEN2=1). При CP_RL2 = 1 – режим фіксації. По перепаду 1→0 зовнішнього сигналу на вході T2EX (при встановленому EXEN2=1) здійснюється пересилання поточного стану лічильних регістрів (TH2, TL2) в пару регістрів (RCAP2H, RCAP2L). Якщо встановлено RCLK=1 чи TCLK=1 (робота на ПАПП), біт CP_RL2 ігнорується, а таймер T2 працює в режимі перезавантаження стартового числа при переповненні

Приймач і передавач послідовного порту в МК AT89S8252 можуть бути синхронізовані від різних джерел (таймер T1 чи T2).

При застосуванні таймера T2 як генератора синхросигналів для УАПП (при RCLK=1 чи TCLK=1) або генератора зовнішніх імпульсів (при T2OE=1) значення частоти генерації визначається виразами

$$F_{УАПП} = F_Q / [32 \cdot (65536 – (RCAP2H,RCAP2L))]$$

та $F_{OUT} = F_Q / [4 \cdot (65536 – (RCAP2H,RCAP2L))]$ відповідно.

При тактовій частоті $F_Q = 11{,}0592$ *МГц* значення стартових чисел: для швидкості 9600 біт/с – FFDCh, 19200 біт/с – FFEEh, 38400 біт/с – FFF7h; 56800 біт/с – FFFAh. При $F_Q = 12$*МГц* для швидкості 9600 біт/с – FFD9h, інші значення точно не реалізуються. Переривання TF2 від таймера T2 в обох випадках не формується.

Перепад 1→0 на вході T2EX фіксується в розряді EXF2 (при EXEN2=1) та обслуговується як додаткове зовнішнє переривання.

Таймер T2 в принципі може формувати два запити на переривання – TF2 (від таймера) та EXF2 (від входу P1.1 – T2EX). Обидва запити пов'язані з одним вектором – 002Bh. Тому ідентифікацію джерела запиту слід виконувати програмно. Із цієї причини ознаки обох запитів мають скидатися програмою (аналогічно із запитами від УАПП).

У регістр дозволу переривань IE додано біт із номером 5 – ім'я ET2, який дозволяє обробку переривань від таймера T2. У регістр пріоритетів IP додано біт PT2 для задавання високого пріоритету цих переривань. Всередині шкали запити TF2 і T2EX мають нижчий пріоритет, ніж УАПП.

2.3.4 Інтерфейс внутрішньосистемного програмування SPI

У мікроконтролерах AT89Sxxxx реалізовано можливість запису програми в РПП за технологією ISP на базі інтерфейсу SPI. Крім цілей програмування, цей інтерфейс у деяких моделях мікроконтролерів може забезпечувати обмін даними між МК зі швидкістю до 1,5 Мбіт/с.

Стандарт інтерфейсу SPI регламентує такі сигнали (в дужках вказано відповідну фізичну лінію порту МК):

MOSI (P1.5) – *master output slave input* – інформаційний вихід контролера, що ініціює обмін (ведучого) – вхід підпорядкованого МК;

MISO (P1.6) – *master input slave output* – інформаційний вхід контролера, що ініціює обмін (ведучого) – вихід підпорядкованого МК;

SCK (P1.7) – *serial clock* – лінія синхронізації; тактові імпульси формує ведучий контролер;

SS (P1.4) – *slave select* – лінія вибору підпорядкованого контролера; такий контролер має формувати на цьому виході сигнал “0”.

Настроювання інтерфейсу SPI для обміну даними з іншими МК здійснюється шляхом встановлення бітів трьох РСФ: SPCR – регістр керування SPI, SPSR – регістр стану SPI та SPDR – регістр даних SPI. (детальніше див. [1]).

Для фізичного програмування Flash-пам’яті використовують інтерфейс SPI спільно із сигналом Reset, а лінія SS не задіяна. При подаванні на вхід Reset сигналу “1” МК переходить у режим скидання і одночасно в режим запису інформації в пам’ять програм.

Фізичне програмування Flash-пам’яті МК може бути здійснено безпосередньо з ПЕОМ. Для видачі сигналів програмування можна застосовувати LPT-порт, а необхідний протокол (послідовність команд, адрес і даних для запису) реалізується спеціальним програмним забезпеченням. Таким чином, через лінії паралельного LPT-порту здійснюється побітова передача даних.

Програмування виконують через кабель, побудований за принципом, що наведений в табл. 2.3.

Кабель виготовляють таким чином, щоб кожний сигнальний провідник утворював виту пару з провідником GND. Довжина кабелю не може перевищувати 1,5 м.

Таблиця 2.3 – Принцип побудови кабелю для інтерфейсу SPI

Лінія інтерфейсу SPI	**Лінія МК (номер виводу)**	**Лінія LPT-порту (контакт роз’єму DB-25)**
MOSI	P1.5 (6)	7
MISO	P1.6 (7)	10
SCK	P1.7 (8)	8
RST	Reset (9)	6
GND	GND (20)	18…25 (всі з’єднати)

Для здійснення фізичного програмування можна застосувати, наприклад, програму-завантажувач **AEC_ISP.exe** (програма MS DOS), яка використовує скомпільований файл у форматі Intel HEX Одночасно з пам’яттю програм можна виконати запис даних до EEPROM.

Ще один варіант програмування МК – застосування вбудованого завантажувача через інтерфейс SPI у складі інтегрованого середовища Visual MCStudio (див. розд. 3). Перевагою цього підходу є здійснення фізичного програмування МК безпосередньо із середовища розробки програм.

3 ТЕХНОЛОГІЯ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МІКРОКОНТРОЛЕРІВ

Ефективність роботи систем керування із цифровими контролерами значною мірою залежить від якості розробки та надійності програмного забезпечення (ПЗ). Це, в свою чергу, визначається ефективністю застосованих алгоритмів, логічними принципами побудови програм і технологією проектування ПЗ (сукупністю етапів і програмних засобів підтримки процесу проектування).

3.1 Процедурний та модульний підходи до розробки програми

Процес проектування ПЗ починається з детального аналізу алгоритмів, задачами якого є:
– визначення структур даних, розподіл сукупності змінних параметрів алгоритмів на вхідні, глобальні, локальні та вихідні;
– визначення автономних частин в алгоритмах, які виконують схожі дії або реалізують певні завершені функціональні перетворення;
– формування на основі визначених автономних алгоритмів низки підпрограм і способів передачі параметрів у них і отримання результатів;
– поєднання підпрограм, за допомогою яких вирішують близькі за функціональністю задачі, у програмні модулі, визначення способів міжмодульної взаємодії;
– встановлення принципів виклику підпрограм – програмний чи як реакція контролера на події (сигнали переривань);
– формування структури головного модуля програми та модуля ініціалізації системи, що виконується першим при запуску контролера; визначення необхідних настройок для таймерів і портів контролера;
– розподіл адресного простору мікроконтролерної системи для безконфліктного розташування модулів із підпрограмами.

Передача параметрів у підпрограми може здійснюватись у двох формах: передача значень параметрів чи передача адрес розташування змінних. Типовими способами передачі параметрів є:

1) використання обмінних зон пам’яті – регістрів чи комірок пам’яті; цей спосіб найширше застосовують для однокристальних мікроконтролерів;

2) передача через стек – спосіб універсальний, але ефективний лише для процесорів, які допускають одночасно стекову та базову адресацію у стеку;

3) використання області коду програми для передачі адрес змінних – спосіб застосовується тільки для підпрограм однорівневого виклику, але не потребує часу на передачу параметрів.

Для забезпечення міжмодульної взаємодії застосовують спеціальні директиви опублікування імен. Наприклад, якщо модуль М1 використовує підпрограми PP1, PP2 та змінну S1, описані в модулі М2, то в декларативній частині модуля М2 мають бути записані директиви

```
public PP1, PP2
public S1  ,
```

а в модулі М1 обов'язковими будуть директиви

```
extern code PP1, PP2
extern data S1
```

3.2 Технологічні кроки при проектуванні ПЗ (текстова реалізація)

Традиційним підходом при проектуванні ПЗ МК є розробка програмних модулів у текстовому вигляді мовою Асемблер. Саме цю апаратно-орієнтовану мову низького рівня застосовують для розробки ПЗ МК, оскільки програміст має ефективно оперувати з пам'яттю, регістрами, периферійними пристроями.

Програма мовою Асемблер являє собою текст, кожний рядок якого визначає одну команду для процесора. Текст програми може бути сформований у будь-якому текстовому редакторі. Для перетворення цього тексту у послідовність двійкових кодів машинних команд застосовують спеціальну програму – *компілятор* з мови Асемблер, або просто Асемблер. Для текстового файла кожного з модулів програми компілятор створює окремий двійковий виконавчий файл. Для об'єднання окремих файлів модулів в одну виконавчу програму застосовують іншу програму – *лінкер* (редактор міжмодульних зв'язків).

Важливим етапом розробки ПЗ МК є тестування створених прикладних програм, пошук і виправлення помилок. Однак для ПЗ МК існують певні проблеми тестування, пов'язані з тим, що МК-системи керування, як правило, не мають розвинених засобів введення та відображення тестової інформації та вмісту пам'яті, покомандних режимів виконання. Для розв'язання цієї проблеми застосовують низку апаратних і програмних засобів.

1. *Програми-симулятори*, які на ПЕОМ імітують для програміста виконання прикладної програми, що підлягає тестуванню. Програми-симулятори працюють на основі логічної моделі МК певної групи, яка містить опис усіх регістрів МК, структури пам'яті, блоків периферійних функцій. Всі елементи логічної моделі МК змінюють свої значення

внаслідок імітації виконання команд прикладної програми. Програміст може модифікувати значення елементів моделі МК з метою тестування програми. Застосування програм-симуляторів є найпоширенішим і найдешевшим підходом до тестування та налагоджування ПЗ МК.

2. *Внутрішньосхемні емулятори* (ВСЕ) – складні апаратно-програмні засоби для фізичної імітації роботи МК у схемі пристрою, що проектується. (рис. 3.1,а). Апаратну основу ВСЕ складає комплект швидкісних імітуючих модулів, який підключається замість МК у схему цільового пристрою та з'єднується з ПЕОМ. На комп'ютері виконується спеціальне ПЗ, через яке програміст має повний доступ до ресурсів фізичної моделі МК і схеми пристрою [1].

Рисунок 3.1 – Апаратно-програмні засоби тестування ПЗ МК

3. *Моніторинг цільового мікроконтролера* безпосередньо в схемі пристрою, що проектується. У прикладну програму для МК додають спеціальні функції, що забезпечують пересилання на ПЕОМ для відображення вмісту основних регістрів МК і змінних у пам'яті й

прийом сигналів керування від ПЕОМ (наприклад, для покрокового режиму виконання програми). При цьому не вдається контролювати абсолютно всі ресурси МК. Сучасним розвитком технології моніторингу є апаратно-програмний інтерфейс JTAG, реалізований в деяких групах МК (наприклад, AVR) – рис. 3.1, б, в.

3.3 Інтегровані середовища розробки ПЗ для МК

Інтегровані середовища розробки (ІСР) для різних мов програмування набули великої популярності в останні 10-15 років. ІСР – це спеціалізована програмна система, що поєднує в одній програмній оболонці всі засоби для розробки ПЗ – текстовий редактор, компілятор, лінкер і налагоджувач. В таких системах для МК налагоджувач, як правило, працює на основі програмного симулятора та логічної моделі МК.

Практично для всіх груп МК на сьогодні створені та доступні широкому колу програмістів спеціалізовані ІСР. Ураховуючи, що функціональні групи МК суттєво відрізняються структурою процесорного ядра, ресурсами та системою команд, для кожної групи існують власні ІСР. Так, найбільш відомими ІСР є: для групи MCS-51 – Raisonaice IDE-51 (RIDE-51), Keil µVision, ProView; для групи AVR – AVR Studio. Узагальнена структура ІСР ПЗ МК показана на рис. 3.2.

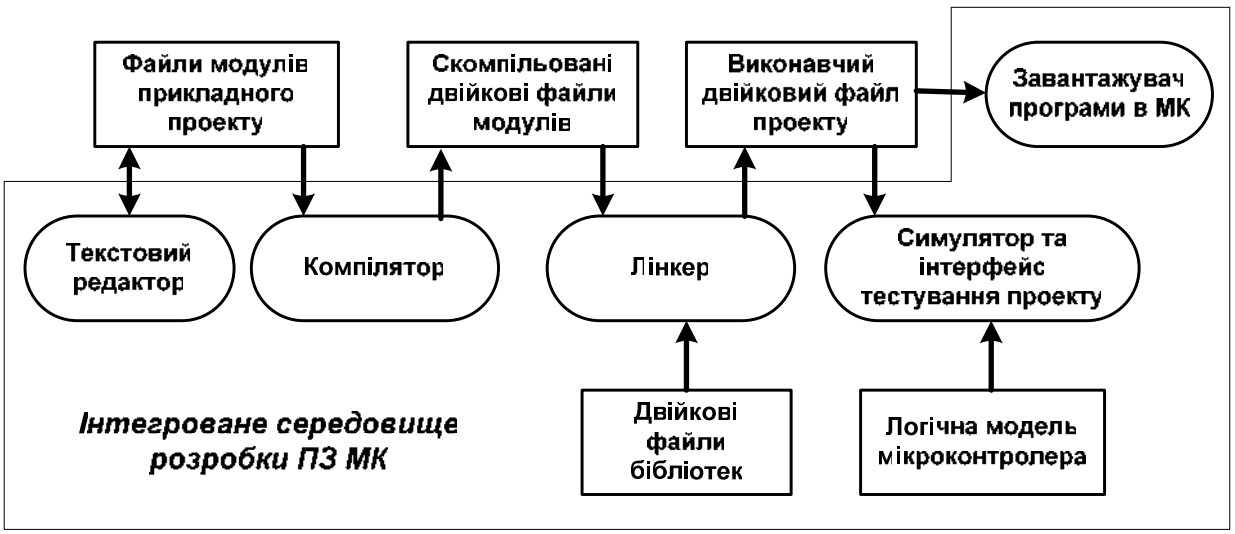

Рисунок 3.2 – Узагальнена структура ІСР ПЗ МК

Слід зазначити, що деякі ІСР ПЗ МК підтримують розробку програм мовою високого рівня, як правило, Сі. У цьому випадку компілятор перетворює програму на Сі в Асемблер (фаза трансляції), а потім

у виконавчий двійковий код. За таким принципом працюють Keil µVision та AVR Studio.

Як правило, інтерфейс симулятора виконання прикладної програми побудований на основі текстового редактора, в якому створювались початкові програмні модулі проекту мовою Асемблер.

Якщо певна група МК підтримує тестування прикладної програми у фізичній системі на основі інтерфейсу JTAG, то відповідне ІСР містить компоненти для реалізації такого тестування.

3.4 Сучасна технологія візуального проектування ПЗ із застосуванням інженерних мов програмування

Широке застосування програмованих логічних контролерів на основі МК у системах промислової автоматизації вимагає швидкої розробки та супроводження значних обсягів програмного забезпечення. У той же час кількість кваліфікованих прикладних програмістів для МК досить обмежена. З іншого боку, супроводження та доробка ПЗ МК вимагають високої якості документування та “прозорості”програм. Виконати названі вимоги на основі традиційної технології розробки ПЗ мовою Асемблер (і навіть Сі) практично неможливо. Саме тому в останні 10 років значної популярності набули так звані інженерні, або графічні мови програмування та програмні системи, що дозволяють їх використовувати.

Інженерні мови програмування надають розробникові сукупність графічних елементів для побудови схем алгоритмів майбутньої програми. Відповідно до міжнародного стандарту IEC-61131-3 основними графічними мовами є FBD (*functional block diagram*) – мова функціональних блоків, та FC (*flow chart*) – мова традиційних блок-схем.

Отже, розробник програми для МК формує її як сукупність наочних алгоритмічних схем у спеціалізованих графічних редакторах, тобто отримує так звану “графічну програму”. Задачею інтегрованого середовища розробки в даному випадку є генерація текстових еквівалентів для графічних програмних модулів, їх подальша компіляція та отримання виконавчого коду для МК. Підсистема симуляції виконання програми має забезпечувати покрокове виконання за алгоритмічними схемами FBD чи FC.

Технологію візуального проектування ПЗ МК розглянемо на основі сучасного інтегрованого середовища **Visual MCStudio**, що створене науково-виробничою фірмою “Інформатика” (м. Харків) [5].

Інтегроване середовище Visual MCStudio складається з підсистем розробки ПЗ (рис. 3.3) і тестування програм (рис. 3.4) і включає в себе такі програмні компоненти:

- модуль адміністрування проектів (менеджер проектів), за допомогою якого створюються проекти й відстежується їх файлова структура;
- спеціалізований багатовіконний текстовий редактор для розробки програмних модулів мовами Сі та Асемблер;
- модуль конфігурації середовища відповідно до модифікації фізичного МК, що використовується (менеджер моделей);
- спеціалізований графічний редактор програм мовою FBD;
- спеціалізований графічний редактор програм мовою FC;

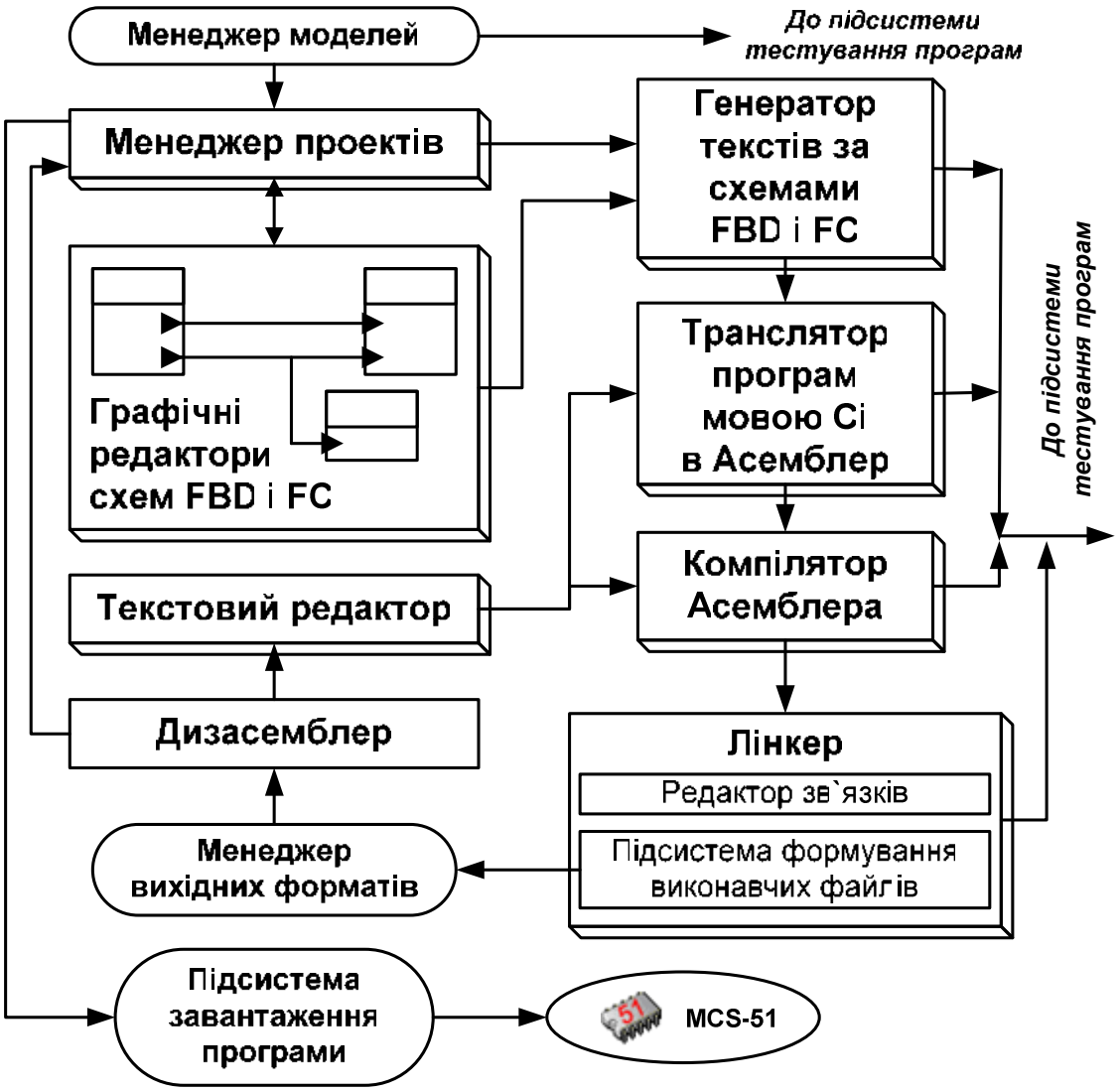

Рисунок 3.3 – Структура підсистеми розробки ПЗ
у складі середовища Visual MCStudio

- генератор текстових еквівалентів графічних програм мовою Сі;
- транслятор текстів програмних модулів мовою Сі в Асемблер-51;
- компілятор з підтримкою макровизначень і пересувних сегментів з мови Асемблер-51 у виконавчий код МК групи MCS-51;
- підсистема формування виконавчих файлів у стандартних (BIN, HEX) або у довільних форматах, описаних користувачем;
- дизасемблер для відтворення тексту програми на основі BIN-, HEX- чи довільного формату виконавчих файлів для МК;
- підсистема завантаження програми у фізичний мікроконтролер, яка реалізує ці дії відповідно до типових чи заданих користувачем протоколів, спираючись на застосування ”плагінів”;

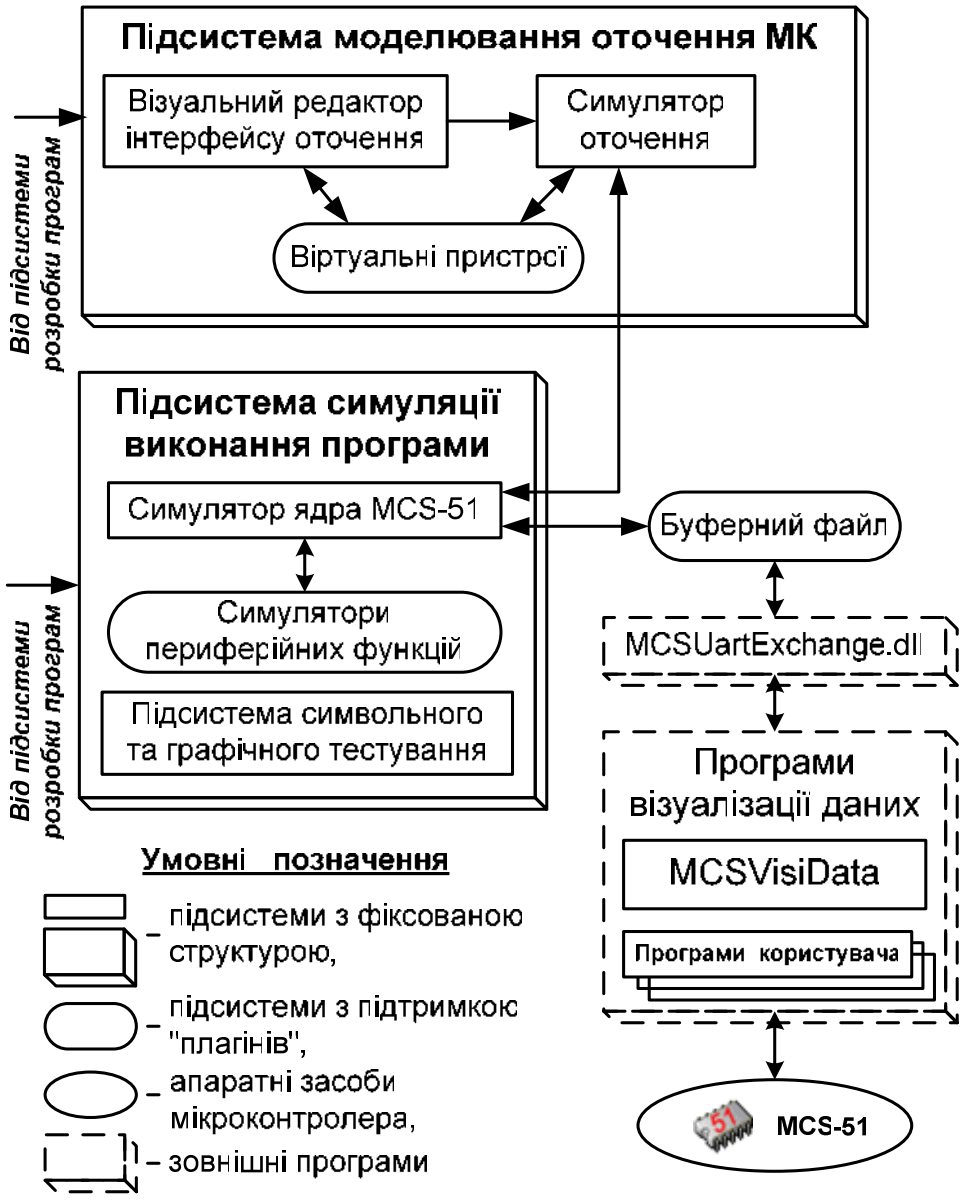

Рисунок 3.4 – Структура підсистеми тестування ПЗ у складі MCStudio

- візуальний редактор опису моделей типових апаратних засобів оточення мікроконтролера у реальній системі та розробки інтерфейсу для тестування програми з імітацією роботи апаратури формування сигналів і відображення даних (так званих віртуальних пристроїв);
- підсистема симуляції виконання прикладних програм, що потрібна для автономного тестування програми для мікроконтролера на ПЕОМ; симулятор виконання програми відображає користувачеві стан апаратних ресурсів мікроконтролера на основі його програмної моделі; додаткові функції певних моделей мікроконтролерів користувач може реалізувати у вигляді DLL і підключити до середовища Visual MCStudio; цей спосіб має назву ”плагін” (від англ. plug-in);
- підсистема символьного та графічного тестування, яка дозволяє вести налагоджування за графічними програмами чи програмами мовою Сі, оперуючи їх елементами (змінними, типами, функціями тощо з текстовим і графічним відображенням їх значень);
- симулятор роботи апаратури зовнішнього оточення мікроконтролера, який в процесі симуляції виконання прикладної програми імітує функціонування певних елементів реальної апаратури (зокрема перемикачів, генераторів, індикаторів тощо);
- автономна програмна система MCSVisiData, що дозволяє описувати структуру пакетів для двостороннього обміну даними між ПЕОМ і мікроконтролерною системою, підтримувати формування та пересилання цих пакетів як з фізичною системою, так і зі симулятором виконання прикладної програми у складі Visual MCStudio, а також відображати прийняті дані у вигляді, визначеному користувачем.

Процес побудови прикладного проекту у системі Visual MCStudio дещо відрізняється, коли проект складається тільки з текстових модулів чи містить модулі, розроблені графічними мовами.

Якщо проект складається тільки з модулів мовою Асемблер, то користувач створює необхідну кількість текстових файлів, що містять текст основної програми та сукупності підпрограм. Після написання тексту модулів здійснюється їх компіляція. За наявності помилок видаються відповідні повідомлення про їх причини та місцезнаходження. У разі успішної компіляції проекту створюється бінарний файл, який може бути завантажений у пам’ять мікроконтролера чи системи на його основі.

Процес формування проекту із модулів, розроблених графічними мовами та мовою високого рівня Сі, показано на рис. 3.5. Слід за-

значити, що в одному проекті можна створювати програмні модулі різними мовами відповідно до задач, що вирішуються. В овальних блоках показані типи файлів, в яких зберігається інформація, сформована користувачем чи згенерована підсистемами ICP Visual MCStudio.

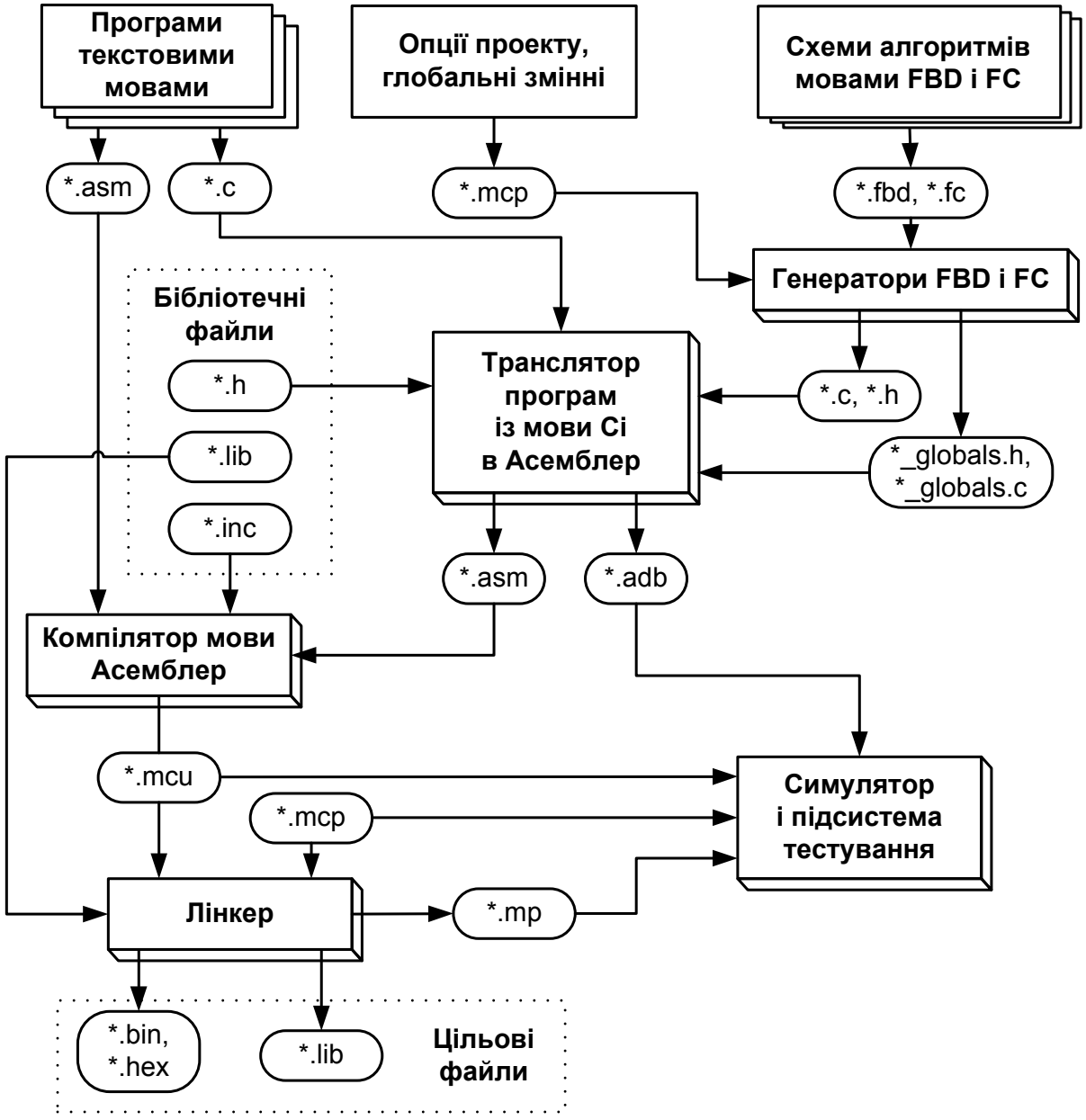

Рисунок 3.5 – Процес формування проекту в системі Visual MCStudio

Формування проекту починається із аналізу поставленої задачі та розподілу її на сукупність автономних підзадач. На основі цих підзадач слід формувати модулі майбутнього програмного проекту. Слід також сформувати (за необхідності) перелік підпрограм, які будуть використовуватися деякими чи всіма модулями. Для кожного модуля чи підпрограми слід сформувати перелік вхідних і вихідних параметрів.

Особливу увагу слід приділяти списку глобальних змінних проекту. Оптимальним можна вважати варіант, коли глобальні змінні відповідають тільки вхідним і вихідним параметрам всього програмного проекту, тобто пов'язані з адресами фізичних пристроїв введення та виведення інформації. У цьому випадку передача всіх даних всередині проекту між окремими модулями та підпрограмами здійснюється в явному вигляді через їх вхідні та вихідні параметри. Відсутність неявної передачі параметрів через глобальні змінні всередині проекту дозволяє суттєво полегшити налагоджування програми та її тестування.

Головний модуль проекту й модуль ініціалізації не повинні мати параметрів. В принципі, у проекті може бути декілька основних безпараметричних модулів; при цьому послідовність їх виконання визначається спеціальними засобами.

Створення проекту у системі Visual MCStudio здійснюється через меню **Файл-Создать-Проект** (або кнопкою ). При створенні проекту слід вказати його назву, а також групу і модель мікроконтролера, для якого проектується програмне забезпечення.

Головне вікно системи Visual MCStudio в режимі розробки проекту має структуру, показану на рис. 3.6.

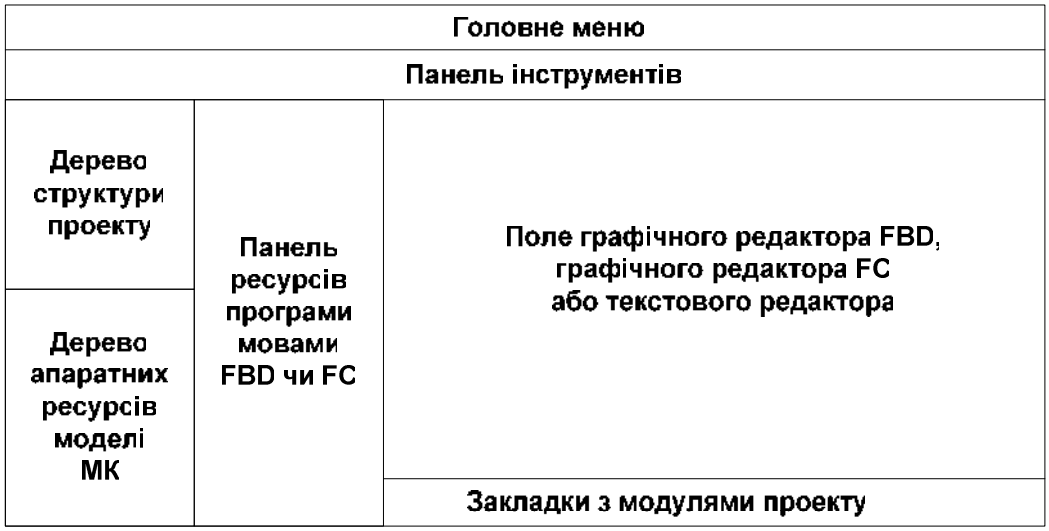

Рисунок 3.6 – Вікно системи Visual MCStudio при редагуванні проекту

Дерево структури проекту показує перелік модулів, з яких складається поточний проект, і призначене для швидкого переключення між редакторами окремих модулів. Для модулів FBD і FC у дереві присутні файли .C та .asm, що генеруються системою.

Дерево апаратних ресурсів моделі МК відображає у структурованому вигляді всі програмно-доступні апаратні засоби МК вибраної моделі. У режимі симуляції виконання прикладної програми в цьому

дереві відображаються поточні значення всіх ресурсів МК (регістрів, портів тощо).

На панелі ресурсів програми мовами FBD чи FC показані програмні компоненти для формування схем алгоритмів цими мовами – оператори, структурні блоки FC, модулі поточного проекту для виклику, бібліотечні функції, апаратні ресурси МК, глобальні та локальні змінні проекту, а також вхідні та вихідні параметри модуля, що редагується.

Редактор програм інженерною мовою FBD є інструментом для швидкого створення графічних програм у вигляді схем алгоритмів з акцентом на структуру програми та передачу даних між блоками обробки, а не на синтаксис мови програмування.

Програма мовою FBD виглядає як сукупність операторних і функціональних блоків (ФБ), з'єднаних лініями даних і керування. Процес програмування мовою FBD – це побудова схеми алгоритму.

Елементами FBD-схеми є оператори, змінні, константи, літерали та функціональні блоки (ФБ), які можуть бути розроблені мовами FBD, FC, C чи Асемблер, а також лінії даних і керування. На рис. 3.7 показано схему для алгоритму *V2 := K1 * (V1 – V1hyst)*, після чого здійснюється збереження значення вхідної змінної *V1hyst := V1*.

Дуги даних з'єднують виходи та входи операторів і ФБ, а також ці елементи із змінними та константами. Дуга даних означає передачу значення від одного елемента схеми до інших. Послідовність виконання блоків FBD-схеми визначається готовністю даних на їх входах.

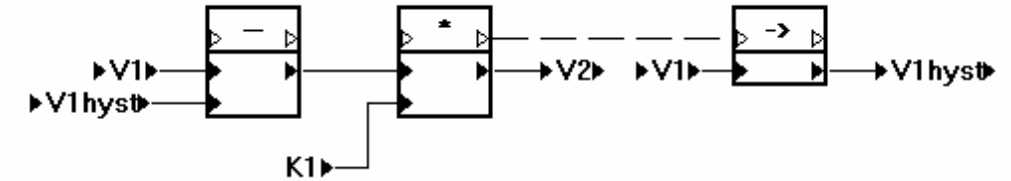

Рисунок 3.7 – Приклад FBD-схеми з лініями даних і керування

Дуги керування явно визначають порядок виконання операторів і функціональних блоків і використовуються у випадках, коли порядок виконання не визначається готовністю даних на входах блоків (паралельні часозалежні гілки алгоритму) або немає однозначності у послідовності виконання операторів. Загалом використання дуг керування не є обов'язковим. Дуги керування завжди прокладають між входами та виходами керування, що знаходяться у заголовках операторів і функціональних блоків.

Виходи блоків чи операторів логічного типу також можна використовувати для задавання порядку виконання блоків, тобто підключати на входи керування. Якщо значення логічної змінної чи виходу дорів-

нює *true*, то підключений блок буде виконуватися. Наприклад, схема на рис. 3.8 відповідає алгоритму: *якщо (Var1+Var2) > 40, то Var3 := 55, інакше Var3 := 0.*

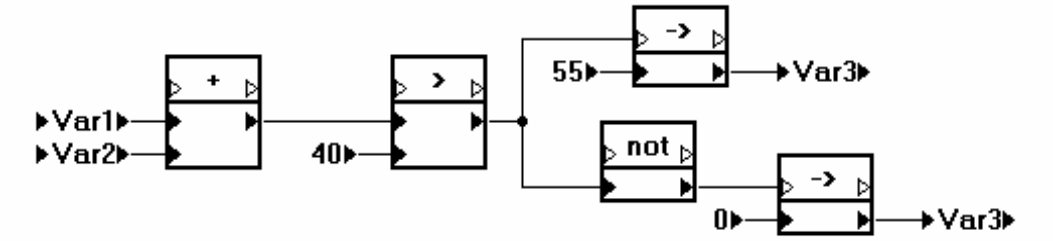

Рисунок 3.8 – Приклад FBD-схеми з лініями умовного виконання

Особливість виконання FBD-схеми полягає у тому, що всі блоки схеми можуть бути виконані тільки один раз або не виконані жодного разу, якщо входять у неактивну гілку умовного виконання. Якщо треба виконати FBD-схему декілька разів, тобто циклічно, її треба викликати декілька разів на зовнішніх схемах (наприклад, на FC-схемі). Всередині FBD-схеми не може бути циклів за лініям даних чи керування.

Схеми FC схожі на традиційні блок-схеми алгоритмів, тобто складаються із прямокутних блоків-дій та ромбоподібних блоків-предикатів. Оператори всередині блоків FC-схеми є текстом, який записується відповідно до синтаксису мови Сі. Цей текст безпосередньо підставляється у текстовий еквівалент програми, що генерується за FC-схемою.

Як на FBD-, так і на FC-схемах можна ставити виклики бібліотечних функцій системи (математичні функції тощо) та виклики функціональних блоків (модулів), що сформовані користувачем у рамках проекту. При викликах за необхідності у функціональні блоки передаються параметри (рис. 3.9).

FBD-схеми застосовують для опису алгоритмів, які містять велику кількість обчислювальних і логічних дій над взаємопов'язаними даними. FC-схеми ефективні для опису алгоритмів, в яких важлива послідовність дій над відносно незалежними даними, або для алгоритмів із великою кількістю логічних перевірок (предикатів).

Загальну структуру проекту формують у вікні керування викликом блоків (пункт у дереві проекту). В цьому вікні програміст задає безпараметричні блоки (модулі), які виконуються одноразово (блоки ініціалізації), та послідовність виконання основних модулів проекту. Така структура проекту базується на тому, що програми керування технологічним обладнанням чи технічними об'єктами, як правило, виконуються циклічно. Можна також асоціювати виклик безпараметричного модуля із апаратним перериванням.

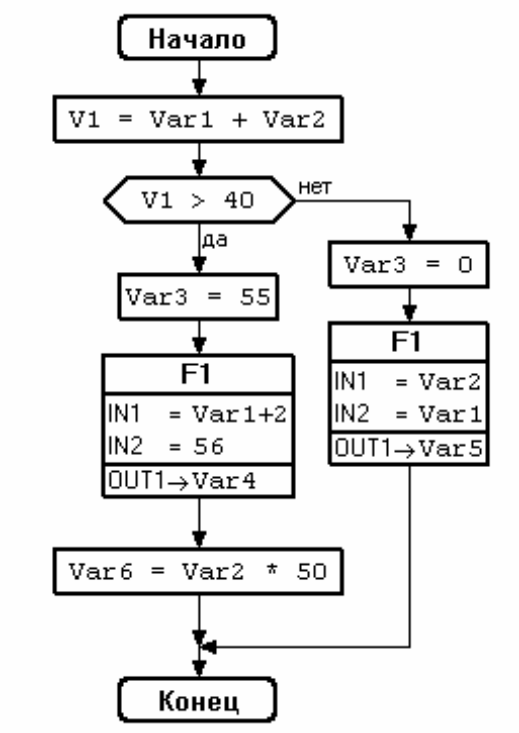

Рисунок 3.9 – FC-схема із викликом раніше визначеного блока F1

Компіляція проекту здійснюється при натисканні Ctrl-Shift-F9 або Ctrl-F9. Всі етапи, пов'язані із генерацією текстових еквівалентів графічних програм, трансляцією файлів мовою Сі у Асемблер, компіляцією у виконавчий код і компонуванням вихідного файла, здійснюються послідовно і автоматично. За наявності повідомлень про помилки програміст має їх виправити й повторити компіляцію.

Наступним етапом проектування ПЗ МК є тестування створеної програми шляхом симуляції її виконання на основі логічної моделі МК у середовищі Visual MCStudio. Процес симуляції запускається натисканням F9 чи  , при цьому курсор симуляції – синя смуга – встановлюється на перший рядок текстової програми або перший блок FBD- чи FC-схеми виділяється синім кольором.

Далі користувач може здійснити:

– автоматичну симуляцію із максимальної швидкістю – F9 чи  ;

– симуляцію "автокрок" із регульованою швидкістю – Alt-F7 чи  ;

– покрокову симуляцію із заходженням у підпрограму – F7 чи  , або із виконанням підпрограми за один крок – F8 чи  .

Під час симуляції незалежно від режиму користувач може переглядати та навіть вручну змінювати з метою тестування вміст всіх регістрів чи комірок пам'яті логічної моделі МК. Регістри загального призначення (R0…R7) і спеціальних функцій доступні через дерево ре-

сурсів (рис. 3.6), а вміст пам'яті можна відображати як масиви комірок в окремих вікнах для різних областей (РПП, РПД тощо), так і у вигляді змінних (), що мають імена.

Одночасно із переглядом числових значень змінних завдяки функції трасування (кнопка ) можна переглядати значення окремих змінних чи виходів МК у вигляді динамічних графіків (трендів).

Для контролю інтервалів часу при симуляції для користувача відображається сукупний час виконання команд програми (імітація відліку реального часу). Також є функція таймінгу, яка фіксує час виконання вказаного фрагмента програми.

За допомогою спеціального редактора (кнопка ) можна сформувати так звану модель апаратного оточення МК. Редактор моделі оточення містить панель віртуальних пристроїв (кнопки, різноманітні індикатори, генератор, джерело аналогових сигналів), які можна розмістити на спеціальній формі та асоціювати із зовнішніми виводами МК. Під час симуляції виконання програми по кнопці  можна імітувати надходження сигналів від зовнішніх джерел і відображення даних на індикаторах. Таким чином, значно підвищується наочність тестування програми для МК.

Окремою опцією симуляції є імітація роботи послідовного порту МК (кнопка ). При цьому в окремому вікні побітово відображається стан буферів прийому та передачі UART, статистика передачі даних.

Для візуалізації даних, що будуть прийматися на ПЕОМ під час виконання програми на фізичному МК, і для передачі даних від ПЕОМ на МК у складі системи є автономна програма MCSVisiData. Структура пакетів прийому та передачі даних довільно формується користувачем. Можливий двосторонній обмін даними між програмою MCSVisiData та моделлю послідовного порту в середовищі Visual MCStudio під час симуляції виконання прикладної програми (фізичні порти ПЕОМ при цьому не задіяні).

До складу Visual MCStudio входять також модулі для завантаження прикладної програми через COM-порт комп'ютера до універсальної мікроконтролерної системи УУМС-2 та для фізичного програмування МК по SPI.

3.5 Особливості формування функціональних блоків

При формуванні модулів проекту в системі Visual MCStudio програміст може створити довільну кількість функціональних блоків (ФБ) мовами FBD, FC чи Сі. Будь-який ФБ можна використовувати (тобто

викликати) в інших алгоритмічних схемах модулів проекту. Як правило, у вигляді ФБ оформлюють функціонально завершені або типові алгоритми керування чи обробки даних (ПІД-регулятор, нелінійності тощо). У системі Visual MCStudio кожний ФБ оформлюється як окремий програмний модуль.

ФБ характеризуються такими параметрами:

1) *входи* – дані, значення яких формуються в процесі виконання прикладної програми зовнішніми відносно даного ФБ алгоритмами (сигнали від зовнішньої апаратури чи виходи інших ФБ);

2) *виходи* – дані, які формуються самим ФБ у процесі виконання та передаються іншим ФБ чи на зовнішню апаратуру по завершенні виконання ФБ;

3) *власні дані* (змінні, що зберігаються між викликами) – формуються в процесі виконання ФБ та обов'язково зберігаються по закінченні алгоритму ФБ до його наступного виклику; якщо в проекті використовують декілька викликів ФБ одного типу, то значення власних даних є унікальними для кожного виклику;

4) *локальні змінні* та *константи* – параметри ФБ, значення яких існують і доступні тільки в процесі виконання даного ФБ до моменту завершення.

Розглянемо формування ФБ на прикладі алгоритму ПІД-регулятора, який описується такими залежностями:

$$E(k) = G(k) - Y(k); \qquad U_I(k) = U_I(k-1) + K_I \cdot E(k);$$

$$U_C(k) = K_P \cdot E(k) + K_D \cdot (Y(k) - Y(k-1)) + U_I(k); \quad Y(k-1) = Y(k),$$

де $G(k)$ – потрібне значення регульованого параметра;

$Y(k)$ – поточне значення виміряного регульованого параметра;

$E(k)$ – поточне значення похибки регулювання;

K_P, K_D, K_I – коефіцієнти пропорційної, диференціальної та інтегральної частин регулятора відповідно;

$U_I(k)$ – інтегральна складова вихідного параметра регулятора;

$U_C(k)$ – вихідний параметр регулятора – керуючий вплив на об'єкт;

$k, k-1$ – поточний та попередній моменти дискретного часу.

Відповідно до вказаних залежностей можна сформувати перелік вхідних, вихідних і власних параметрів ФБ ПІД-регулятора:

G, Y, K_P, K_I, K_D – вхідні параметри; U_C – вихідний параметр, $Y1$ – власний параметр ФБ, що відповідає $Y(k-1)$ в алгоритмі, U_I – власний параметр, що відповідає $U_I(k)$ та $U_I(k-1)$ (достатньо однієї змінної, оскільки ці параметри стоять у різних частинах рівняння).

Можливий варіант реалізації алгоритму ПІД-регулятора у вигляді FBD-схеми показано на рис. 3.10.

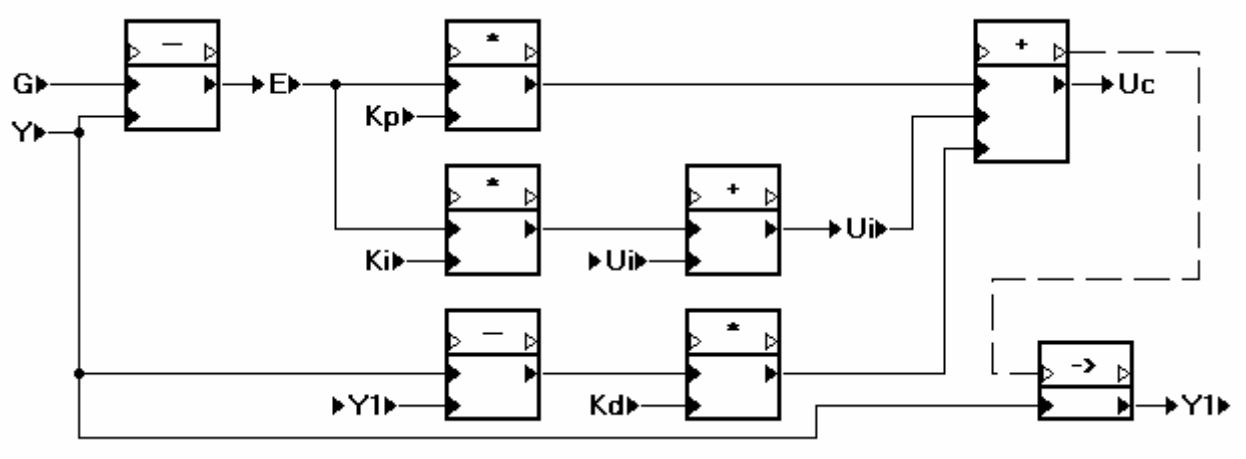

Рисунок 3.10 – Реалізація FBD-схеми алгоритму ПІД-регулятора

З'єднання блока суматора із блоком присвоєння значення лінією керування необхідне для того, щоб поточне значення вхідного параметра *Y* зберігалось у власній змінній $Y1$ тільки після обчислень за алгоритмом. Локальна змінна E в даному випадку застосована для наочної відповідності заданому початковому рівнянню ПІД-регулятора.

При використанні розробленого ФБ на схемах інших модулів вхідні параметри можна задавати як числові константи або як змінні; такий приклад застосування показано на рис. 3.11.

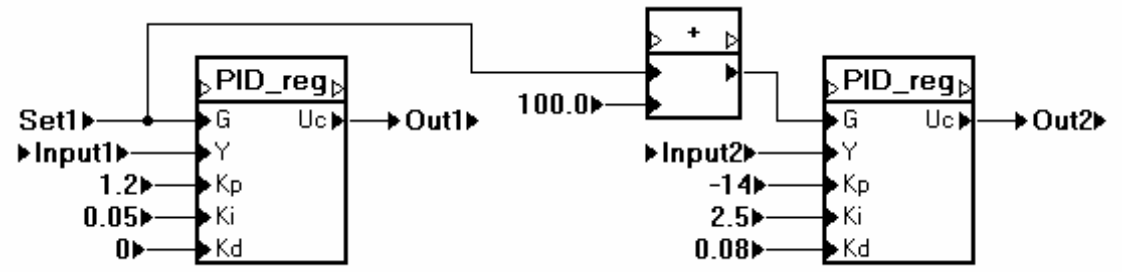

Рисунок 3.11 – Приклад застосування ФБ на схемі іншого модуля

Зазначимо, що не існує обмежень на кількість використань розроблених ФБ на одній чи декількох інших схемах алгоритмів, що можуть бути створені мовами FBD або FC. Власні дані конкретного ФБ є унікальними для кожного використання (тобто для кожного виклику ФБ). Це означає, що змінні $Y1$ та U_I є унікальними для кожного із двох екземплярів ПІД-регулятора на схемі рис. 3.11, хоча алгоритми екземплярів блока **PID_reg** однакові.

4 РОЗВ'ЯЗАННЯ ТИПОВИХ ЗАДАЧ КОНТРОЛЮ ТА КЕРУВАННЯ НА БАЗІ МК

У практичних задачах застосування мікроконтролерів з ядром MCS-51 у технічних системах контролю та керування часто виникає потреба під'єднати до МК зовнішні пристрої для формування, введення чи виведення дискретних та аналогових сигналів.

4.1 Під'єднання простих електричних пристроїв до МК

1. Мікросхеми, виготовлені за технологією ТТЛ, можна під'єднувати безпосередньо до портів МК. При цьому слід ураховувати паспортну навантажувальну здатність портів по виходу: вісім входів ТТЛ для порту P0 та чотири входи ТТЛ для інших портів.

2. Під'єднання кнопок і перемикачів слід виконувати так, щоб на входах МК не було «невизначеного потенціалу» (рис. 4.1):

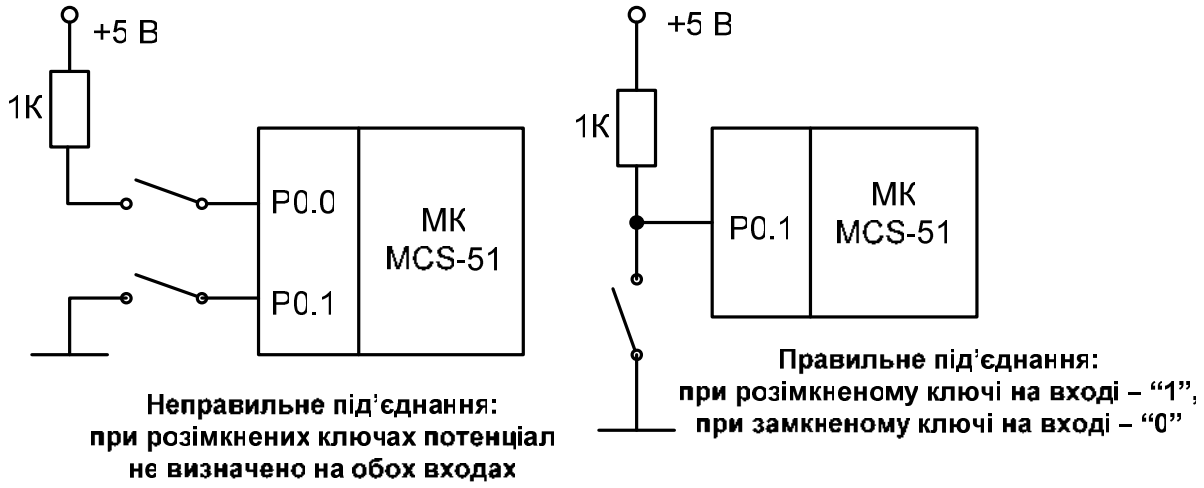

Рисунок 4.1 – Приклади під'єднання перемикачів до портів МК

3. Для сигналів, які формуються за допомогою механічних перемикачів і потребують швидкої реакції МК (це сигнали переривань чи вхідні сигнали таймерів), слід передбачити захист від ефекту «вібрації» контактів (рис. 4.2).

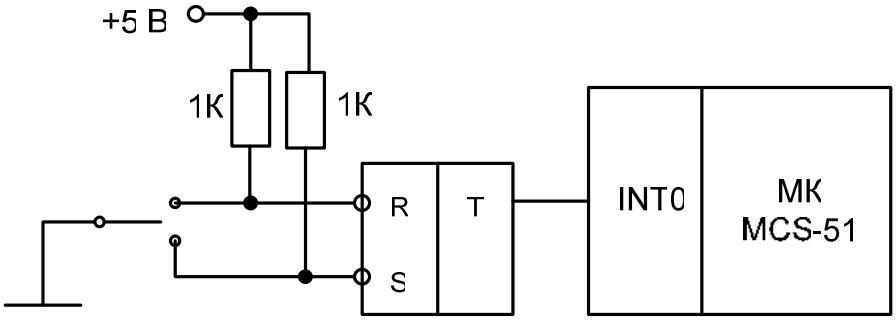

Рисунок 4.2 – Схема усунення ефекту «вібрації» сигналів

Наявність RS-тригера в такій схемі дозволяє сформувати на вході INT0 одноразовий перехід в “0” при перемиканні тумблера вверх (R=0) та одноразовий перехід в “1” при перемиканні тумблера вниз (S=0). Якщо ключ знаходиться в середньому положенні із-за вібрації, то на обох входах буде неактивний рівень (S=R=1), тобто сигнал на виході тригера не буде змінюватись.

4. Елементи індикації можна під’єднати так, щоб світіння здійснювалось або при “1”, або при “0” на виході порту (рис. 4.3):

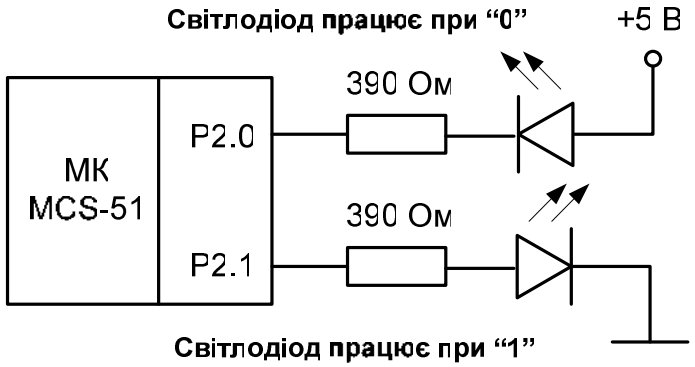

Рисунок 4.3 – Приклади під’єднання елементів індикації

5. При під’єднанні семисегментних індикаторів (ССІ) треба:

– на лінію керування кожним сегментом установити обмежувальний резистор (390 Ом) чи застосувати шинний формувач;

– врахувати, що ССІ бувають виконані за схемою із спільним анодом (сегмент світиться при сигналі “0”) чи із спільним катодом (сегмент світиться при сигналі “1”) (рис. 4.4);

– для відображення числових даних треба сформувати таблицю перекодування: наприклад, для відображення “0” слід передати на індикатор код “hgfedcba” = 00111111, для відображення “1” – “hgfedcba” = =00000110, якщо сегмент “a” під’єднано до молодшого біта порту.

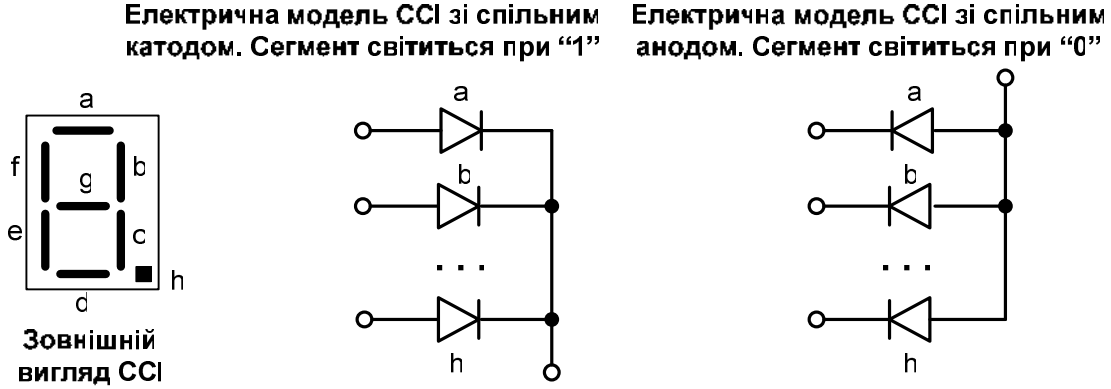

Рисунок 4.4 – Різновиди семисегментних індикаторів

6. При керуванні реле рекомендується встановити транзисторний ключ між виходом контролера і обмоткою реле. Паралельно з обмоткою реле обов'язково треба включити діод для погашення струму самоіндукції обмотки (рис. 4.5), а для обмеження струму через відкритий транзистор слід застосувати резистор в колекторному ланцюзі.

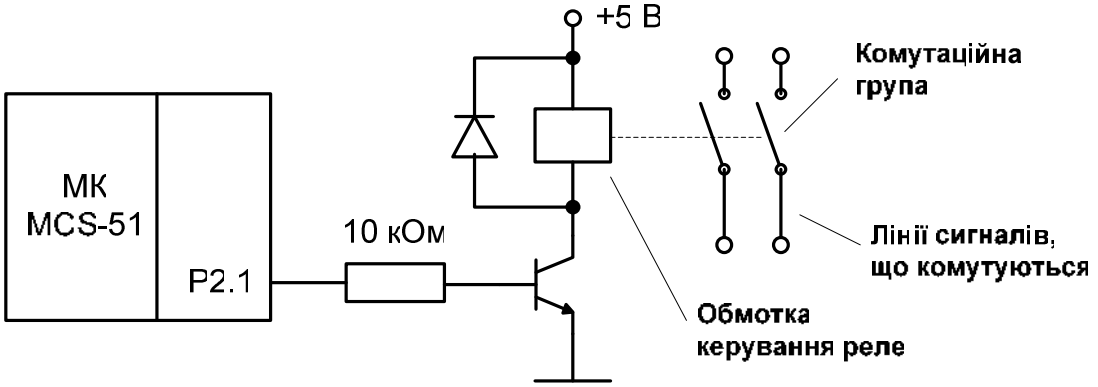

Рисунок 4.5 – Принцип під'єднання реле до мікроконтролера

Слід зазначити, що незалежно від пристрою, що під'єднаний до виводів МК, опитування сигналів здійснюється за допомогою звичайних команд роботи з портами. Так, для чекання значення "1" на лінії P0.1 відповідно до рис. 4.1, можна використати команду

```
m1: JNB P0.1, m1 .
```

Для відображення значення "0" на СІ із спільним катодом відповідно до рис. 4.4 можна застосувати команду

```
mov P1, #00111111b ,
```

якщо індикатор під'єднано до порту P1. При використанні СІ із спільним анодом керування сегментами здійснюється в інверсному коді, тобто для відображення на такому СІ цифри "0" треба скористатися командою

```
mov P1, #11000000b .
```

Відповідно до схеми на рис. 4.5, для замикання контактів реле можна скористатися командою **`setb P2.1`** , а щоб перевести реле в розімкнений стан, – командою **`clr P2.1`** .

Слід пам'ятати, що реле є фізичним пристроєм із обмеженою швидкодією, особливо у порівнянні із МК. Наведені вище команди виконуються процесором МК приблизно за 1 мкс. Але реле потребує у сотні разів більшого часу на перемикання. Тому при програмуванні почергового замикання та розмикання реле слід ураховувати ці фізичні часові обмеження.

4.2 Реалізація інтервалів часу та вимірювання часових параметрів

Важливий клас задач для цифрових контролерів – це забезпечення інтервалів часу між подіями чи вимірювання часових параметрів сигналів (інтервалів часу між сигналами, частоти, кількості й ширини імпульсів). Як правило, такі задачі вирішують за допомогою таймерів, що входять до складу мікроконтролера.

4.2.1 Програмна затримка виконання функціональної програми

Забезпечення певного інтервалу часу між подіями при виконанні функціональної програми найбільш просто реалізується за допомогою порожніх циклів, що виконуються відповідну кількість разів. Для розрахунку таких циклів використовують відому тривалість виконання команд МК. Протягом інтервалу, що реалізується, процесор не здійснює алгоритмічну обробку даних, оскільки зайнятий виконанням команд циклів.

Як приклад розглянемо програмну реалізацію інтервалу часу тривалістю 1 с. При цьому зазначимо, що команда організації циклів **djnz** виконується 2 МЦ, що при тактовій частоті 11.0592 МГц становить $T_{DJNZ}=2.17$ *мкс*. На основі ручного підбору кількості повторення циклів можна запропонувати варіант реалізації підпрограми:

```
Delay1s: mov   r2,#8
         mov   r3,#1
         mov   r4,#244
Z1s:     djnz  r4,Z1s  ; внутрішній вкладений цикл
         djnz  r3,Z1s  ; середній за вкладенням цикл
         djnz  r2,Z1s  ; зовнішній цикл
         ret
```

Час виконання підпрограми становить (з урахуванням call та ret):

$$T_{1S} = 4 \cdot (2T_{МЦ}) + 245 \cdot T_{DJNZ} + 7 \cdot (256 \cdot (257 \cdot T_{DJNZ})) + 8 \cdot T_{DJNZ} + 2T_{МЦ} .$$

Наприклад, якщо треба на лінії P1.0 сформувати сигнал “1” тривалістю 1 с, то фрагмент програми може бути таким:

```
setb  P1.0
call  Delay1s
clr   P1.0
```

Очевидним недоліком такого підходу є непродуктивні дії процесора протягом цілої секунди, однак перевага – у простоті програмної реалізації.

4.2.2 Реалізація інтервалів часу на основі таймерів

Застосування таймерів у складі МК дозволяє вивільнити процесор від виконання непродуктивних дій, зокрема порожніх циклів чекання. При використанні таймера реалізація інтервалу часу здійснюється шляхом підрахунку кількості імпульсів від ГТІ МК із періодом $T_{ГТІ} = T_{МЦ} = 12 / F_Q$.

Інтервал часу, що реалізується, триває від моменту запуску таймера до моменту переповнення. Таким чином, для задавання тривалості інтервалу слід розрахувати стартове число для таймера відповідно до методики, наведеної в [2].

Приклад 1. Реалізувати інтервал 50 мс на таймері T0 при F_Q = *12 МГц.*

Стартове число $N_{ST} = (65535 + 1) - \left[\frac{50 \cdot 10^{-3}}{1 \cdot 10^{-6}}\right] = 15536.$ Прийнятна конфігурація лічильних регістрів таймера – 16 біт. Фрагмент програми може виглядати так:

```
org 0000h
  jmp   start     ; Вектор початкового переходу
org 000Bh
  jmp   proc_T0   ; Перехід на обробку переривання
org 0030h         ; Блок ініціалізації
start: mov  TMOD,#00000001b
    mov  IE,#10000010b
    mov  TL0,#Low(Nst) ; Задавання стартового числа
    mov  TH0,#High(Nst)
    mov  SP,#6Fh      ; Ініціалізація стека
    . . .         ; Інші дії
    setb TR0 ; Запуск таймера – початок інтервалу
    . . .         ; Інші дії в програмі
    . . .    ;    !!! Мітка Proc_T0 програмно недосяжна
"int"   . . .  ; На одній з команд виникає переривання
    . . .
Proc_T0: ; Процедура обробки переривання від таймера T0
    clr  TR0 ; Зупин таймера – кінець інтервалу
    . . .         ; Інші дії при закінченні інтервалу
    reti          ; Повернення в основну програму
```

Відповідно до принципів обробки переривань [3], якщо переривання від таймера T0 виникло, наприклад, при виконанні команди, що позначена “int”, буде зроблено перехід на адресу 000Bh, потім на про-

цедуру Proc_T0; повернення здійснюється на команду, наступну після команди, позначеної як “int”.

Якщо тривалість інтервалу перевищує максимально можливе значення для таймера, то можна реалізувати інтервал як сукупність дискрет часу: $T_3 = m \cdot T_D$, де T_D – тривалість дискрети, що реалізується таймером, а значення *m* можна контролювати програмно, наприклад, підраховуючи на регістрі Ri (i=0,1, …, 7) моменти переповнення таймера.

Приклад 2.

Реалізувати інтервал $T_3 = 2c$ на таймері T1 при $F_Q = 12$ МГц.

Тривалість дискрети вибираємо як $T_D = 50$ мс. Тоді $m = 40$.

```
org 0000h
  jmp   start     ; Вектор початкового переходу
org 001Bh
  jmp   proc_T1   ; Перехід на обробку переривання
org 0030h         ; Блок ініціалізації
start: mov  TMOD,#00000001b
    mov  IE,#10000010b
    mov  TL1,#Low(Nst) ; Задавання стартового числа
    mov  TH1,#High(Nst)
    mov  SP,#6Fh      ; Ініціалізація стека
    mov  R7,#40       ; Коефіцієнт дискрети
    . . .             ; Інші дії
    setb TR1 ; Запуск таймера – початок інтервалу
    . . .         ; Інші дії в програмі
    . . .    ;    !!! Мітка Proc_T1 програмно недосяжна
“int”    . . .  ; На одній з команд виникає переривання
    . . .
Proc_T1: ; Процедура обробки переривання від таймера T1
    djnz  r7, m_out ; Перевірка, чи пройшли 40 дискрет
    clr  TR1   ; Зупин таймера – кінець інтервалу
    . . .           ; Інші дії при закінченні інтервалу
    jmp  m_ret      ; Стрибок на вихід з процедури
m_out: mov TL1,#Low(Nst) ;Задавання стартового числа
    mov  TH1,#High(Nst) ;для наступної дискрети
m_ret:  reti          ; Повернення в основну програму
```

Особливість даної реалізації полягає в тому, що при кожному входженні в процедуру Proc_T1 по закінченні дискрети команда `djnz` зменшує стан регістра R7. Якщо R7>0 (не всі 40 дискрет пройшли),

таймер завантажується на наступну дискрету. Тільки коли R7=0 (витримано 40 дискрет, тобто інтервал у 2 с), таймер буде зупинено і далі здійснено дії в процедурі відповідно до завершення інтервалу.

4.2.3 Підрахунок кількості й вимірювання частоти імпульсів

Однією з типових задач у системах цифрового керування є підрахунок кількості імпульсів від зовнішніх пристроїв. У разі, коли здійснюється підрахунок періодичних імпульсів за обмежений інтервал часу, можна казати про вимірювання частоти (чи періоду) імпульсів.

1. Задача простого підрахунку зовнішніх імпульсів. Її вирішують шляхом подавання таких імпульсів на зовнішній вхід таймерів T0, T1 чи T2. При цьому вибраний таймер слід настроїти на режим підрахунку зовнішніх імпульсів [3]. Як правило, таймер використовують у 16-бітовій конфігурації. Наприклад, при застосуванні таймера T0 треба записати в регістр керування таймерами TMOD число 00000101b.

2. Вимірювання частоти імпульсів на основі підрахунку їх кількості за фіксований інтервал часу. Можливі два підходи до такого вимірювання – з програмною реалізацією інтервалу та на основі іншого таймера.

Для програмної реалізації інтервалу вимірювання можна скористатися підпрограмою затримки Delay1s, що була розглянута на початку розд. 4.2:

```
cseg
mov  TMOD, #00000101b  ; Настроювання таймера T0
. . .   ; на режим підрахунку зовнішніх імпульсів
setb  TR0      ; Початок вимірювання
call  Delay1s  ; Затримка на 1с
clr   TR0      ; Кінець вимірювання.
. . . ; Результат міститься в регістрах TL0 і TH0
```

Переваги такого підходу – простота й наочність, однак недоліком є непродуктивні дії процесора в інтервалі підрахунку.

Максимальна частота зовнішніх імпульсів при такому підході до вимірювання обмежується розрядністю таймерних регістрів: $F_{MAX} = 2^{16} - 1 = 65535\ Гц$, а мінімальна частота пов'язана з інтервалом вимірювання: $F_{MIN} = 1/T_B = 1\ Гц$.

Більш ефективним є застосування іншого таймера, наприклад, T1, для реалізації інтервалу вимірювання. Для простоти розглянемо постановку задачі із прикладу 1, де інтервал становить $T_B = 50$ *мс* при $F_Q = 12$ *МГц.*

```
org 0000h
  jmp   start     ; Вектор початкового переходу
org 001Bh
  jmp   proc_T1   ; Перехід на обробку переривання
org 0030h         ; Блок ініціалізації
start: mov  TMOD,#00010101b ; Настроювання таймерів
    mov  IE,#10001000b    ; Дозвіл переривання від T1
    mov  TL1,#Low(Nst)    ; Задавання стартового числа
    mov  TH1,#High(Nst)
    . . .         ; Інші дії
    mov  TL0,#0    ; Обнулення таймера, що підраховує
    mov  TH0,#0    ; зовнішні імпульси
    setb TR1 ; Запуск таймерів – початок інтервалу
    setb TR0 ; і початок підрахунку
    . . .         ; Інші дії в програмі
"int". . .  ; На одній з команд виникає переривання при
    . . .  ; закінченні інтервалу вимірювання (50мс)
Proc_T1: ; Процедура обробки переривання від таймера T1
    clr  TR1 ; Зупин таймерів – кінець інтервалу
    clr  TR0 ; та кінець підрахунку
    mov  R3,TL0  ; Фіксація результату вимірювання
    mov  R4,TH0
    . . .         ; Інші дії при закінченні інтервалу
    reti          ; Повернення в основну програму
```

В даному прикладі, оскільки T_B незначний, слід враховувати апаратні обмеження МК на частоту зовнішніх імпульсів: $F_{MAX} = 2/T_{МЦ} = 500\ кГц$, тобто за інтервал T_B може бути підраховано тільки 25000 імпульсів. Мінімальна частота визначається так само: $F_{MIN} = 1/T_B = 20\ Гц$.

Інтервал підрахунку може бути збільшено на основі підходу, що був наведений у прикладі 2 – програмний підрахунок дискрет часу.

Загалом принцип вимірювання частоти визначається часовою діаграмою на рис. 4.6.

Абсолютне значення частоти вимірюваних імпульсів може бути розраховане як $F_B = K/T_B$. Слід зазначити, що зі збільшенням T_B точність результату зростає.

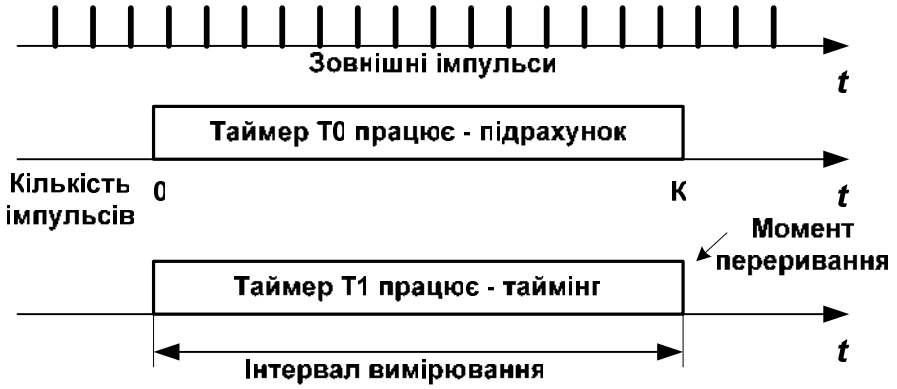

Рисунок 4.6 – Принцип вимірювання частоти імпульсів

3. Вимірювання періоду слідування імпульсів є одним із способів визначення їх частоти. Такий підхід застосовують, коли частота імпульсів незначна, отже, для її безпосереднього вимірювання шляхом підрахунку імпульсів потрібен значний інтервал часу (декілька секунд). Принцип вимірювання періоду зовнішніх імпульсів T_B полягає у підрахунку синхронних імпульсів від еталонного генератора (наприклад, ГТІ МК), що вмістилися у період T_B. Для такого підрахунку застосовують таймер МК у режимі таймінгу, а зовнішні імпульси визначають моменти фіксації значення з таймера (два способи показані на рис. 4.7).

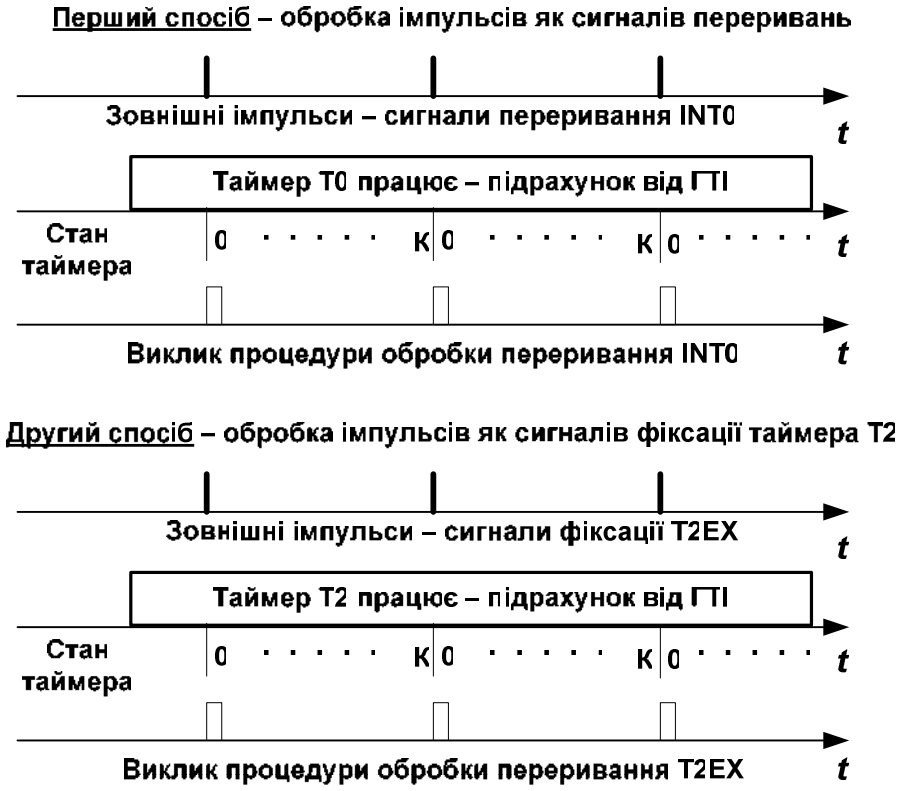

Рисунок 4.7 – Ілюстрація способів вимірювання періоду імпульсів

Перший спосіб – зовнішні імпульси фіксують як сигнали переривання (INT0 чи INT1), а в процедурі обробки переривання поточне значення таймера T0 чи T1 можна “скидати” в регістри для подальшої обробки. Як приклад наведемо програму із застосуванням переривання INT0 і таймера T0.

```
org 0000h
  jmp   start     ; Вектор початкового переходу
org 0003h
  jmp   pr_INT0   ; Перехід на обробку переривання
org 0030h         ; Блок ініціалізації
start: mov  TMOD,#00000001b ; Настроювання таймера T0
    mov  TL0,#0   ; Обнулення таймера T0, що підраховує
    mov  TH0,#0   ; імпульси від ГТІ
    setb IT0 ; Переривання фіксуватиметься за зрізом
    mov  IE,#10000001b   ; Дозвіл переривання від INT0
    . . .    ; Інші дії в програмі
“int”. . . ; На одній з команд виникає переривання INT0
    . . .    ; Інші дії в програмі

Pr_INT0: ; Процедура обробки переривання INT0
    clr  TR0 ; Зупин таймера T0
    mov  R3,TL0  ; Фіксація результату вимірювання (К)
    mov  R4,TH0
    mov  TL0,#0   ; Обнулення таймера T0, що підраховує
    mov  TH0,#0   ; імпульси від ГТІ
    setb TR0      ; Запуск таймера T0
    reti          ; Повернення в основну програму
```

Зазначимо, що перший прохід через процедуру Pr_INT0 буде “холостим”, оскільки таймер ще не запущено. Крім того, для підвищення точності треба враховувати час від появи сигналу INT0 до зупину таймера.

Другий спосіб базується на тому, що в мікроконтролерах 80x52 є можливість автоматичної фіксації поточного стану таймера T2. При цьому вхідні імпульси, період яких вимірюється, слід подавати на вхід T2EX (лінія P1.1). Сигнал на цьому вході керує копіюванням поточного стану таймера T2 в регістри RCAP2H, RCAP2L (див. розд. 2.10). Після кожного копіювання таймер слід обнулити.

Для настроювання таймера T2 у регістрі T2CON слід встановити біти EXEN2 (дозвіл сприйняття сигналу по входу T2EX) та CP_RL2 (режим фіксації). Байт для настроювання таймера T2 буде мати вигляд 00001001b.

Частота вимірюваних імпульсів може бути виражена через кількість підрахованих імпульсів К від ГТІ як $F_B = 1 / (K \cdot T_{МЦ})$, причому $T_{МЦ} = T_{ГТІ}$.

Максимальне значення вимірюваної частоти становить $F_{B\ max} = 1 / T_{МЦ}$, а мінімальне – $F_{B\ min} = 1 / (N_{max} \cdot T_{МЦ})$, де $N_{max} = 65535$ для 16-бітного таймера.

Програмна реалізація вказаного способу може бути такою:

```
  FdataL  equ  40h    ; Результат вимірювання (2 байти)
  FdataH  equ  41h
org 0
  jmp    start        ; Вектор початкового переходу
org 002Bh    ; Обробка переривання від входу T2EX
  mov  FdataL,RCAP2L ; Копіювання значень
  mov  FdataH,RCAP2H ; із регістрів фіксації
  mov  TL2,#0        ; Обнулення таймера T2
  mov  TH2,#0
  clr  EXF2          ; Скидання ознаки переривання
  reti       ; Повернення в основну програму
org XXXX
start:            ; Основна програма
  mov  T2CON,#00001001b  ; Настроювання таймера T2
  mov  IE,#10100000b ; Настроювання переривань
  mov  TL2,#0        ; Обнулення таймера
  mov  TH2,#0
  setb TR2           ; Запуск таймера T2
main_calc:           ; Обчислення за алгоритмами,
    . . .            ; що здійснюються циклічно
  jmp  main_calc
```

Важливо, що в наведеному прикладі застосовано тільки таймер Т2. Фактична максимальна вимірювана частота обмежена часом виконання процедури обробки переривання (14 машинних циклів з урахуванням виклику) і становить $F_{ВФ\ max} = 1 / (14\ T_{ГТІ})$.

4.2.4 Вимірювання тривалості (ширини) імпульсів

Задача вимірювання тривалості (ширини) імпульсів розв'язується шляхом підрахунку кількості імпульсів фіксованої частоти (наприклад, від ГТІ МК), що вмістилися в активну фазу вимірюваного імпульсу. Можна запропонувати два варіанти вимірювання тривалості зовнішніх імпульсів. У будь-якому з варіантів апаратним засобом

вимірювання є таймер у складі МК, який запускається відповідно до активної фази імпульсу (рис. 4.8).

1. Програмно-кероване вимірювання тривалості імпульсів

Підхід базується на програмній перевірці рівня сигналу на вхідній лінії МК, на яку надходять імпульси. Коли програмно фіксується високий (активний) рівень сигналу, слід включити таймер. При програмній фіксації низького рівня – завершенні активної фази імпульсу – таймер слід зупинити, а накопичене в таймері значення виражає тривалість імпульсу в одиницях періоду $T_{ГТІ}$ імпульсів від ГТІ МК.

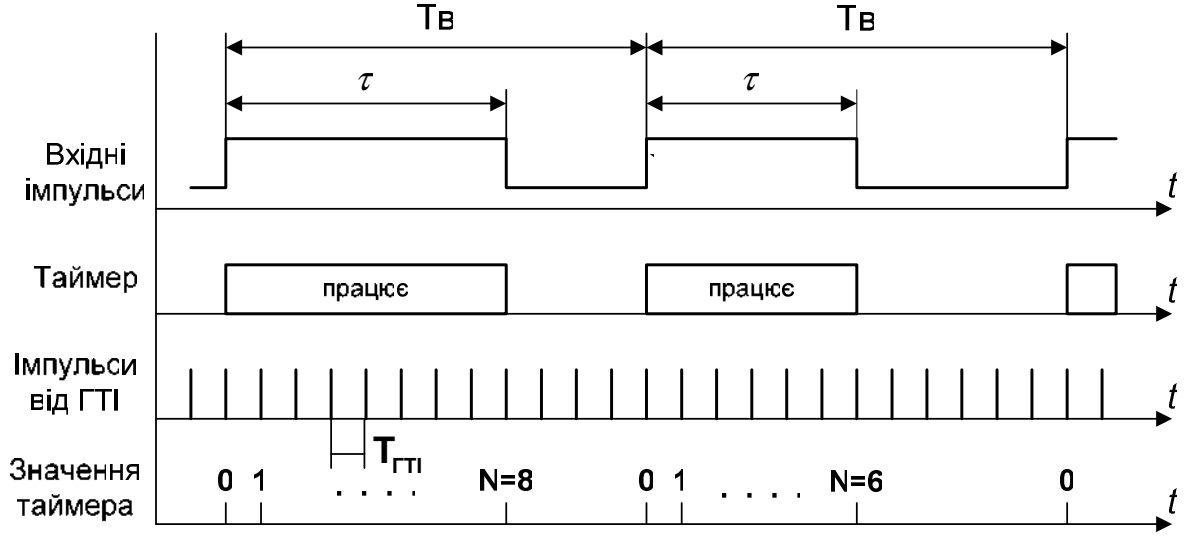

Рисунок 4.8 – Принцип вимірювання ширини імпульсів

Зовнішні імпульси можна подавати на будь-який вхід МК, який можна програмно перевіряти – наприклад, P1.0. Вимірювання здійснюють циклічно. Оскільки зазвичай крім вимірювання ширини імпульсів треба виконувати певні обчислення, то повернення до чергового вимірювання може припадати на середину активної фази поточного імпульсу. Тому в програмі слід забезпечити запуск таймера якраз на початку активної фази імпульсу.

```
  mov  TMOD, #00000001b   ; Настроювання таймера T0
  mov  IE, #0             ; Всі переривання заборонені
  . . .
M1: mov  TL0,#0           ; Цикл вимірювання
  mov  TH0,#0
  jb   P1.0,$ ; Перевірка завершення попереднього
              ; імпульсу
  jnb  P1.0,$ ; Перевірка початку нового імпульсу
  setb TR0    ; Початок вимірювання – запуск таймера
  jb   P1.0,$ ;Перевірка завершення поточного імпульсу
```

```
  clr  TR0     ; Закінчення вимірювання – зупин таймера
  mov  R6,TL0    ; Фіксація результату вимірювання
  mov  R7,TH0
  . . .          ; Необхідна обробка даних
  jmp  M1        ; Повернення до початку вимірювань
```

Примітка: символ $ означає перехід на адресу поточної команди, тобто здійснюється зациклювання на поточній команді.

Очевидна перевага такого підходу – простота програмної реалізації завдяки лінійності програми та чіткій послідовності перевірок.

Серйозний недолік – непродуктивне чекання зміни фаз імпульсу, причому тривалість чекання залежить від ширини імпульсу.

2. Вимірювання з автоматичним запуском таймера

У МК групи MCS-51 є можливість керувати запуском таймера за допомогою рівня сигналу, що подається на вхід INTx. Ця можливість є активною при установленні біта GATE=1 у регістрі TMOD. При цьому таймер слід запустити програмно один раз у блоці ініціалізації, а поточні запуски і зупини будуть здійснюватись відповідно до фази вхідного імпульсу на INTx [3].

Оскільки імпульси, що вимірюються, надходять на вхід запиту переривання, можна автоматизувати реакцію на завершення активної фази імпульсу – як обробку переривання INTx за зрізом сигналу. В процедурі обробки зовнішнього переривання можна здійснити фіксацію значення таймера, нескладну обробку інформації. В основній програмі слід реалізувати потрібний період дискретності системи і обчислення за алгоритмами. Результати вимірювання оновлюються автоматично, паралельно з основними обчисленнями.

```
org 0
  jmp main_prog  ; Перехід на основну програму
org 0003h        ; Обробка переривання від INT0
  mov  R6,TL0    ; Фіксація результату вимірювання
  mov  R7,TH0
  mov  TL0,#0  ;Обнулення таймера для наступного циклу
  mov  TH0,#0
  reti           ; Повернення в основну програму

main_prog:           ; Основна програма
  mov  TMOD,#00001001b   ; Настроювання таймера
  mov  IE,# 10000001b    ; Дозвіл переривання INT0
  setb IT0           ; INT0 сприймається за зрізом
  mov  TL0,#0        ; Початковий стан таймера
  mov  TH0,#0
```

```
  setb TR0   ; Таймер включено, але він ще не працює
main_calc:       ; Обчислення за алгоритмами
  . . .          ; з використанням R6 та R7
  jmp  main_calc
```

Для синхронізації оновлення результату вимірювання з основною програмою в процедурі обробки переривання INT0 можна передбачити установлення бітової ознаки, яка буде перевірятися основною програмою.

Процедура обробки переривання INT0 в даному прикладі розташована безпосередньо на векторі переривання (адреса 0003h). У складних програмах, де є обробка інших переривань, слід перемістити цю процедуру на інше місце в пам'яті програм, а за адресою 0003h розмістити команду переходу на цю процедуру.

Важливо, що період вимірюваних імпульсів, як правило, значно менше, ніж період дискретності алгоритмів керування обладнанням. Тому запропонований підхід значно вивільняє ресурси процесора, оскільки не потребує програмної перевірки фаз імпульсів при вимірюванні. Процедура обробки переривання в мінімальній реалізації потребує всього 12 машинних циклів.

Максимальне значення ширини вимірюваних імпульсів при 16-бітовій конфігурації таймера становить $T_{B\,max} = 65536\,T_{МЦ}$, а мінімальне $T_{B\,min} = T_{МЦ}$.

Таймер Т2 не має можливості запуску від зовнішнього сигналу, тому його бажано застосовувати для формування періоду дискретності роботи контролера чи для синхронізації послідовного порту, якщо таймер Т1 потрібен для вимірювання ширини зовнішніх імпульсів.

4.2.5 Забезпечення виконання програми із заданим періодом

Одним із параметрів алгоритмів керування обладнанням, сформованих у процесі синтезу цифрового регулятора, є значення періоду (періодів) дискретності T_0, наприклад

$$U[kT_0] = A_1 \cdot X[kT_0] + A_2 \cdot X[(k-1)T_0] + B_1 \cdot U[(k-1)T_0].$$

Хоча параметр T_0 не є коефіцієнтом алгоритму чи вхідною змінною, його врахування і точна програмна реалізація необхідні, оскільки від цього залежить якість керування.

З урахуванням періоду дискретності узагальнена часова діаграма реалізації алгоритмів керування має вигляд, показаний на рис. 4.9.

Найпростішим способом реалізації періоду дискретності є обчислення точного часу виконання дій з введення, обробки та виведення

інформації $T_{ВОВ}$ і часу чекання завершення періоду дискретності $T_{ЧК} = T_0 - T_{ВОВ}$. Чекання можна реалізувати на основі фіксованої, заздалегідь розрахованої кількості програмних циклів (реалізація залежить від тактової частоти і принципів виконання команд МК [3]).

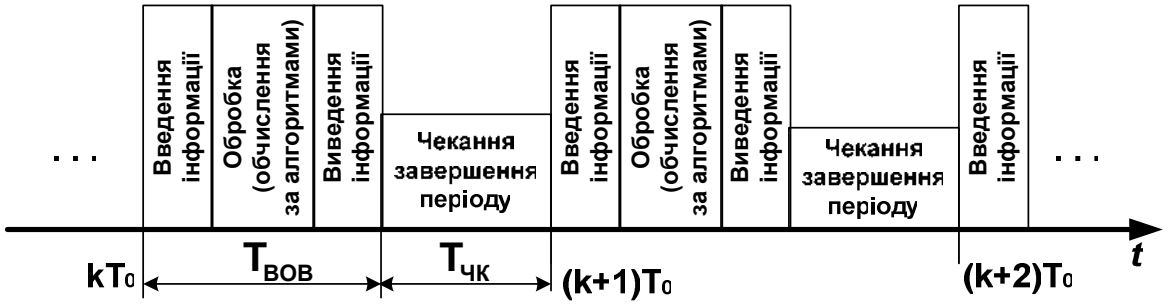

Рисунок 4.9 – Заповнення періоду дискретності роботи контролера

Для МК MCS-51 часто застосовують програмні цикли на основі команди **`DJNZ Rg,мітка`**, причому в регістр Rg завантажують потрібну кількість повторень. Виконання команди **`DJNZ`** потребує двох машинних циклів.

Застосовувати такий програмний підхід має сенс тільки для простих алгоритмів керування, що не містять великої кількості розгалужувань, тобто гілок програми з умовним виконанням (для таких алгоритмів $T_{ВОВ}$ буде приблизно однаковим у сусідніх періодах, що дозволяє приблизно дотримуватися реального часу). У високоточних системах реального часу (*real-time systems*) керування енергетичними, металургійними чи хіміко-технологічними процесами для реалізації періодів дискретності завжди застосовують таймери.

Обробка завершення періоду T_0 як переривання від таймера є найбільш ефективним способом, оскільки дозволяє вивільнити процесор від непродуктивного чекання (що може бути важливо в багатозадачних системах). Всі розрахунки для вибору конфігурації й стартового числа таймера здійснюються так само, як описано в [3].

Чекання завершення періоду T_0 здійснює апаратний контролер переривань. Відповідно, дії, пов'язані із завершенням періоду дискретності, виконуються в процедурі обробки переривання від таймера.

Не рекомендується здійснювати обчислення за основними алгоритмами керування обладнанням у процедурі обробки переривання від таймера. У такому випадку процедура стає великою, а при зациклюванні чи апаратних перешкодах єдиним засобом повернення до нормального функціонування є "скидання" МК, оскільки реагування на інші переривання стає неможливим.

При чеканні завершення періоду на основі переривань найбільш складною є задача повернення до початку блока основних обчислень. Переривання від таймера і виклик процедури його обробки, як правило, припадають на момент, коли процесор закінчив основні обчислення і знаходиться в циклі чекання. Тому після переривання у стеку буде збережено саме адресу цього циклу. Повернення з процедури обов'язково слід виконувати за командою RETI, однак не в цикл чекання, а на початок блока обчислень. Розв'язанням цієї проблеми може бути заміна адреси повернення в стеку, яку можна здійснити в самій процедурі обробки переривання.

Приклад 1. Період T_0 повністю реалізується на таймері T0 (16 біт)

```
Nst equ 12000 ; Стартове число для періоду T0
org 0
  jmp  main_prog ; Перехід на початок основної програми
org 000Bh        ; Вектор переривання від таймера T0
  jmp  proc_T0   ; Перехід на процедуру обробки
main_prog:              ; Основна програма
  mov  TMOD,#00000001b;Таймер T0 в конфігурації 16 біт
  mov  IE,#10000010b  ; Дозвіл переривання
                      ; від таймера T0
  mov  TL0,#Low(Nst)    ; Завантаження стартового числа
  mov  TH0,#High(Nst)   ; для таймера T0
  setb TR0            ; Запуск таймера T0
calc:. . .    ; Обчислення за основними алгоритмами,
   . . .      ; що здійснюються з періодом T0
     . . .
  jmp  $     ; Цикл чекання завершення періоду

Proc_T0: ; Процедура обробки переривання від таймера T0
  clr  TR0        ; Зупин таймера
  mov  TL0,#Low(Nst); Запис стартового числа у T0
  mov  TH0,#High(Nst)    ;
  setb TR0   ; Запуск таймера T0
  pop  ACC   ; Видалення зі стека старої
  pop  ACC   ; адреси повернення
  mov  a,#Low(calc)  ; Запис у стек нової адреси:
  push ACC   ; спочатку молодша частина адреси CALC,
  mov  a,#High(calc)
  push ACC           ; потім старша частина
  reti ; Повернення в основну програму на обчислення
```

Якщо для реалізації періоду дискретності застосувати таймер T2, то перезавантаження стартового числа буде здійснюватися автоматично в момент переповнення таймера із регістрів RCAP2H: RCAP2L. У процедурі обробки переривання треба скидати ознаку переривання TF2. Дії, що стосуються задавання нової адреси повернення з процедури, можна здійснювати так само, як і в наведеному прикладі. Замість двох команд **pop ACC** можна двічі застосувати команду **dec SP**.

Якщо період дискретності неможливо реалізувати безпосередньо і повністю на таймері МК, слід скористатися підходом, що базується на апаратній реалізації малих дискрет часу і програмному підрахунку їх кількості.

Приклад 2. Період T_0 реалізується дискретами $T_Д$ на таймері T2

```
Nst equ  15536 ; Стартове число для дискрети Тд=50мс
M equ 20 ; Відношення T0 до Тд
DC data 30h  ; Змінна – лічильник дискрет
org 0
  jmp  main_prog ; Перехід на основну програму
org 002Bh        ; Вектор переривання від таймера T2
  clr  TF2 ;Скидання ознаки переривання від таймера T2
jmp  proc_T2 ; Перехід на процедуру
             ; обробки переривання
main_prog:           ; Основна програма
  mov  T2CON,#00000000b  ; Настроювання таймера T2
  mov  IE,# 10100000b    ; Дозвіл переривання від T2
  mov  TL2,#Low(Nst)     ; Задавання стартового числа
  mov  TH2,#High(Nst)    ; для таймера T2
  mov  RCAP2L,#Low(Nst)   ; Число перезавантаження
  mov  RCAP2H,#High(Nst) ; для таймера T2
  mov  DC,#M ; Початкове значення лічильника дискрет
  setb TR2           ; Запуск таймера T2
calc:. . .           ; Обчислення за алгоритмами,
  . . .          ; що здійснюються з періодом T0
    jmp  $   ; Цикл чекання завершення періоду

proc_T2: ; Процедура обробки переривання від таймера T2
  djnz DC,out ; Цикл до відпрацювання (M*Тд) дискрет
  mov  DC,#M   ; Початкове значення лічильника дискрет
    . . .      ; Інші дії при завершенні періоду T0
  pop  ACC     ; Видалення зі стека адреси повернення
  pop  ACC     ;(можна зробити двома командами dec SP)
```

```
  mov  a,#Low(calc)  ; Нова адреса повернення – у стек
  push ACC           ; Молодша частина адреси CALC,
  mov  a,#High(calc) ; старша частина адреси CALC
  push ACC
out: reti          ; Повернення в основну програму
                   ; на початок обчислень
```

У даному прикладі слід відмітити важливу особливість. Команда **djnz** в процедурі обробки переривання **proc_T2** реалізує цикл, «розподілений у часі». Це означає, що поки значення змінної DC не стане дорівнювати 0 в результаті декремента, після перевірки значення DC одразу здійснюється вихід із процедури. Наступний декремент DC і перевірка її значення будуть виконані через інтервал $T_Д$. Таким чином, процедура обробки переривання і повернення до початку обчислень (мітка **calc**) повністю виконуються один раз на M інтервалів $T_Д$. Проміжні (M–1) викликів процедури **proc_T2** можуть припадати у часі як на виконання основних обчислень, так і на фазу чекання завершення періоду T_0 (рис. 4.10). Повернення з процедури обробки **proc_T2** при проміжних викликах буде здійснюватися звичайним шляхом – у ту точку основної програми, звідки було виконано перехід на процедуру обробки **proc_T2**.

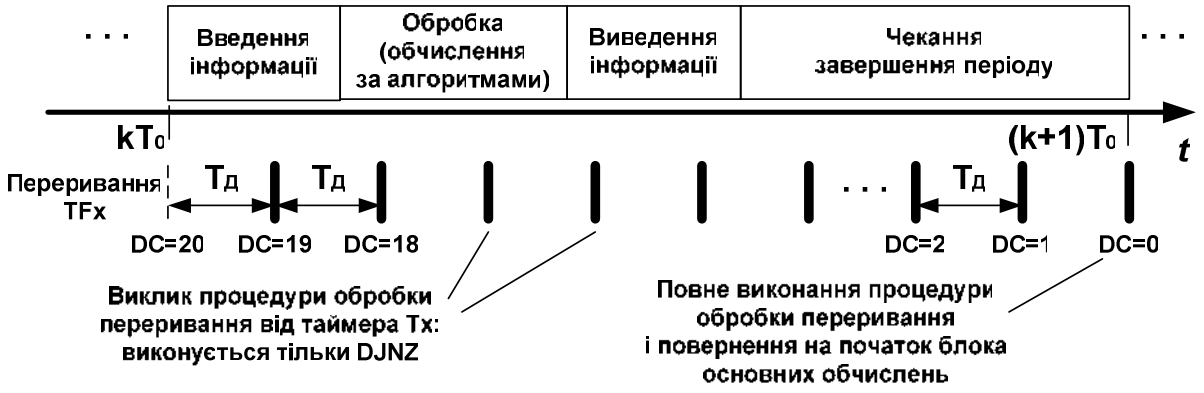

Рисунок 4.10 – Часова діаграма реалізації періоду дискретизації на основі програмного підрахунку часових дискрет

Принципова особливість програм для систем автоматичного керування – це циклічний принцип їх виконання, тобто обчислення здійснюються з періодом T_0 фактично безкінечно (до вимкнення живлення МК). Саме тому алгоритми, наведені у вигляді блок-схем, можуть не мати традиційного блока завершення «Кінець». Природно, що в кожній конкретній системі відповідно до вирішуваних задач можуть здійснюватися програмні переходи на різні режими функціонування системи, у тому числі й на нециклічні однократні дії.

4.3 Прийом аналогових сигналів у МК та їх перетворення

У сучасних системах програмного керування технологічним обладнанням чи рухомими об'єктами однією із задач, що виконують МК, є взаємодія із датчиками (вимірювачами) фізичних параметрів, що безперервно змінюються у часі, тобто введення аналогових сигналів у цифрову систему. Обов'язковим елементом у цьому процесі є перетворювач аналогових неперервних сигналів у дискретну чи цифрову форму. Тому в цей час використовують декілька способів введення аналогових сигналів у цифровий контролер:

1) використання зовнішніх АЦП та аналогових мультиплексорів;

2) застосування МК із вбудованими АЦП і мультиплексорами;

3) на основі перетворювачів "напруга – ширина імпульсів" (ШІМ) із наступним вимірюванням у МК тривалості вхідних імпульсів;

4) на основі перетворювачів "напруга – частота" (ЧІМ) із наступним вимірюванням у МК частоти чи періоду вхідних імпульсів.

Зазначимо, що деякі фірми-розробники пропонують пристрої вимірювання аналогових сигналів із цифровим виходом, як правило, у вигляді послідовного коду. В цьому випадку МК взаємодіє з вимірником як із будь-яким іншим дискретним пристроєм відповідно до протоколу його роботи.

4.3.1 Використання зовнішнього інтегрального АЦП

Зараз існує велика кількість АЦП в інтегральному вигляді, які можна безпосередньо під'єднувати до МК чи шин обчислювальних систем.

Умовне схемне зображення типового АЦП показано на рис. 4.11. Вхідними лініями пристрою є: аналоговий інформаційний вхід AI, вхід керування Start для запуску перетворення та скидання АЦП, вхід U_{REF} для задавання опорної напруги. Вихідними сигналами є: N-розрядна цифрова шина для видачі результату перетворення і сигнал Ready, який показує, що перетворення завершено і вихідні дані готові.

Сучасні модифікації АЦП мають не паралельну, а послідовну вихідну шину. Це дозволяє економити виводи портів МК, забезпечивши при цьому високу розрядність результату перетворення (16 чи 24 біти). Додатковим сигналом керування в таких моделях є сигнал синхронізації, імпульси якого супроводжують біти вихідного коду АЦП.

Основні характеристики АЦП:

- діапазон вхідного аналогового сигналу, $\boldsymbol{U_{BX\ MIN}}$ ... $\boldsymbol{U_{BX\ MAX}}$, В (визначається відповідно до U_{REF});
- розрядність вихідного коду ***N***, біт;

- час перетворення $T_{ПР}$, мкс;
- нелінійність статичної характеристики, В/біт.

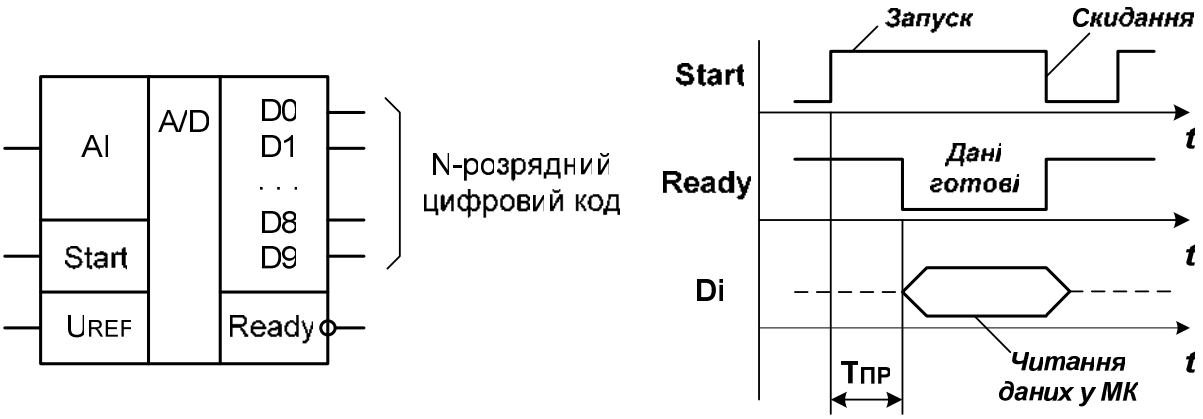

Рисунок 4.11 – Схемне зображення типового 10-розрядного АЦП і часова діаграма його роботи

У практичних розробках має сенс застосовувати АЦП імпортного виробництва, наприклад фірм Analog Devices, Philips чи Texas Instruments.

Типовим рішенням при розробці програмно-керованого каналу введення даних є безпосереднє під'єднання АЦП до портів МК. При цьому слід задіяти декілька ліній МК для видачі програмним шляхом сигналів керування і прийому коду від АЦП. Для прикладу застосуємо такий розподіл ліній портів МК (рис. 4.12):

P0.0...P0.7, P1.0, P1.1 – прийом вихідного коду АЦП – D0...D9;

P1.2 – прийом сигналу *Ready* (готовність даних на виходах АЦП);

P1.3 – формування сигналу *Start* (запуск перетворення).

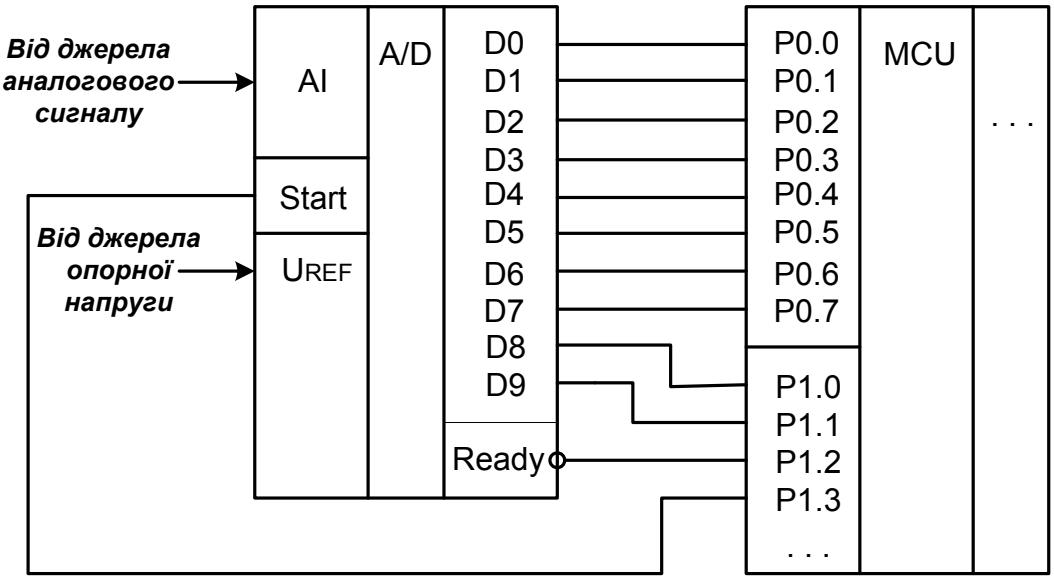

Рисунок 4.12 – Приклад під'єднання інтегрального АЦП до портів МК

Програмний фрагмент обслуговування АЦП має підтримувати протокол керування АЦП відповідно до часової діаграми його роботи (рис. 4.11). У даному прикладі програма може мати вигляд

```
mL data 40H  ; комірки для розміщення 10-бітового коду,
mH data 41H  ; отриманого від АЦП
CSEG
; Настроювання портів
  clr  P1.3      ; Скидання АЦП
  mov  P0, #0FFH ; Настроювання порту P0 на введення
  setb P1.0 ; Настроювання ліній P1.0-P1.2 на введення
  setb P1.1
  setb P1.2
loop: call ADT   ; Виклик процедури керування АЦП
     . . .       ; Обробка даних
    jmp  loop    ; Цикл основних алгоритмів
; Процедура керування АЦП
ADT: setb P1.3   ; Запуск перетворення
WAIT: jb Ready, WAIT ; Чекання сигналу готовності даних
   mov  mL, P0   ; Читання вихідного коду АЦП
   mov  A, P1    ; Читання байта з розрядами N8, N9
   anl  A, #00000011B ; Виділення потрібних розрядів
   mov  mH, A
   clr  P1.3     ; Видача сигналу скидання АЦП
   ret
```

Можлива логіка керування АЦП показана на рис. 4.13.

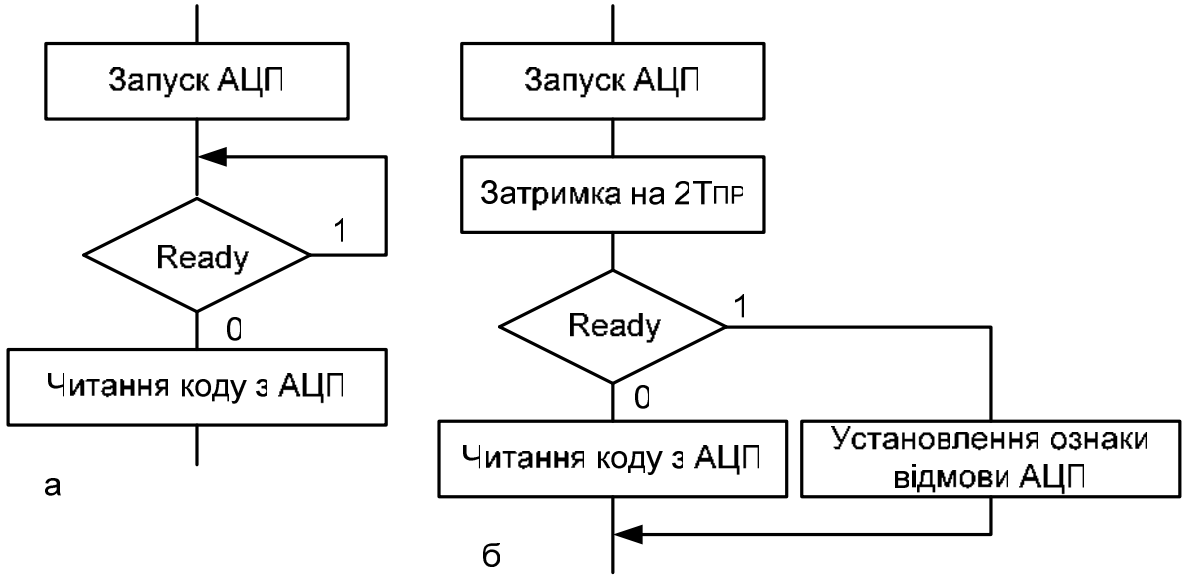

Рисунок 4.13 – Логіка керування АЦП із чеканням сигналу готовності

Варіант на рис. 4.13,а – це найпростіший принцип аналізу сигналу *Ready*. Такий підхід чутливий до відмови АЦП (чи іншого пристрою, обмін з яким здійснюється за аналогічним протоколом), оскільки програма “зациклюється”, якщо сигнал *Ready* не дорівнює “0”. Альтернативою є принцип, показаний на рис. 4.13,б. Після запуску АЦП здійснюється затримка на подвійний час $T_{ПР}$, щоб при нормальній роботі АЦП гарантовано отримати результат. Якщо *Ready* не дорівнює “0”, це буде означати відмову пристрою, що можна врахувати в програмі.

Процедура керування АЦП, взята з попереднього прикладу, для варіанта (б) може виглядати так (для $T_{ПР} = 20$ мкс, біт F0 – ознака відмови АЦП):

```
ADT: setb P1.3   ; Запуск перетворення
   clr  F0
   mov  r7,#20
W1:  djnz r7,W1  ; Затримка на 2*Тпр, тобто на 40мкс
   jb Ready,Fault;Перевірка сигналу готовності Ready=0
   mov  mL, N0_7 ; Читання вихідного коду АЦП (N0…N7)
   mov  A, P1    ; Читання байта з розрядами N8, N9
   anl  A, #00000011B ; Виділення бітів N8,N9
   mov  mH, A
   jmp  mout
Fault: setb F0   ; Установлення ознаки відмови АЦП
mout: clr  P1.3  ; Видача сигналу скидання АЦП
   ret
```

Алгоритм, показаний на рис. 4.13,б, можна використати для керування будь-якими пристроями, від яких слід чекати відклику чи сигналу готовності даних.

4.3.2 Принципи побудови багатоканальних систем збирання даних

У цифрових системах керування часто виникає необхідність прийому даних від великої кількості датчиків. У зв’язку із цим блоки введення і АЦ-перетворення мають бути побудовані як багатоканальні модулі:

- містити декілька АЦП (за кількістю вхідних сигналів);
- забезпечувати часове мультиплексування вхідних аналогових сигналів для обробки в єдиному АЦП.

Типова аналого-цифрова система збирання даних (СЗД) складається з аналогового мультиплексора (АМХ), інтегрального АЦП і схем сполучення з портами МК. Будь-який з аналогових входів (зазвичай 4 чи 8) через АМХ за допомогою сигналів керування від МК

може бути під'єднаний до входу АЦП, і таким чином буде отриманий і прочитаний в МК цифровий код, що відповідає рівню аналогового сигналу. Можлива схема системи зображена на рис. 4.14,а.

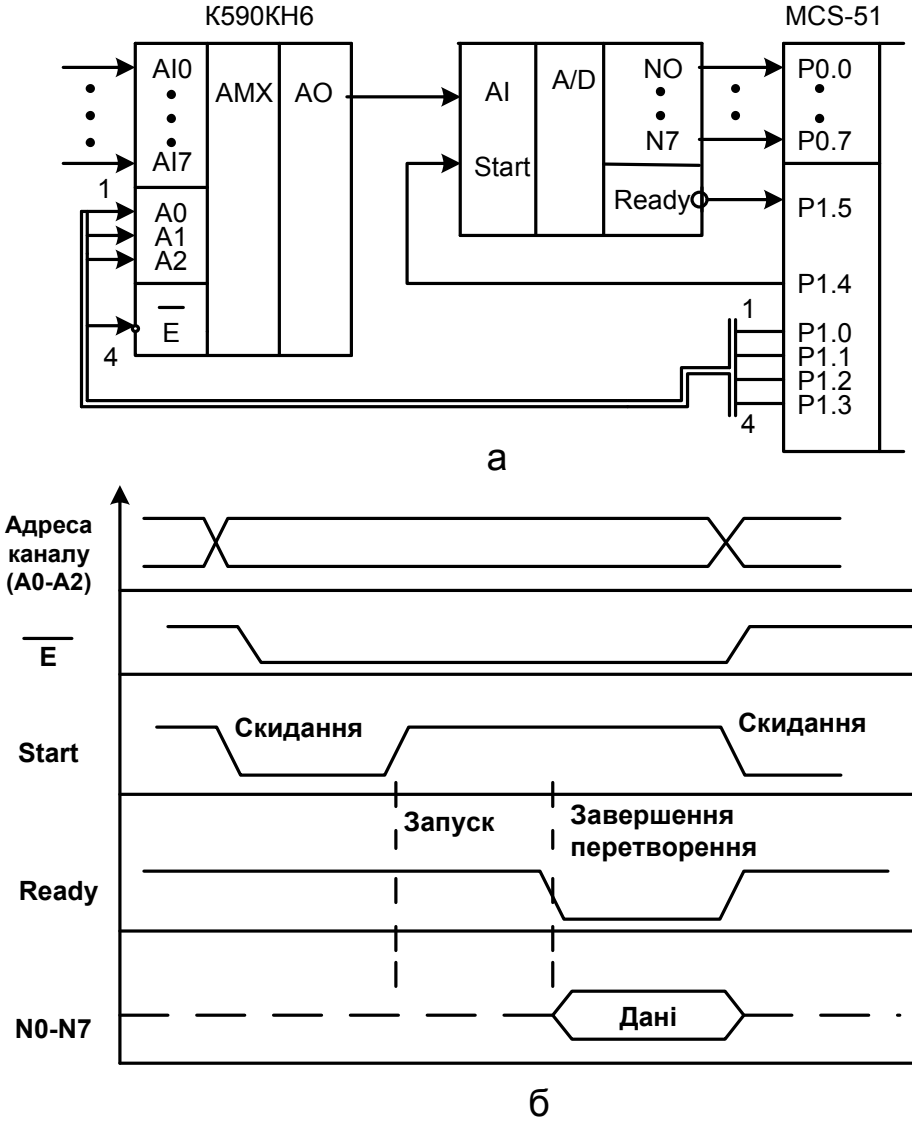

Рисунок 4.14 – Приклад побудови багатоканальної СЗД

Аналоговий мультиплексор – схема комутації, яка дозволяє у довільному порядку під'єднати будь-який з аналогових входів до аналогової вихідної лінії. Номер входу, який під'єднується до АЦП, задається для АМХ як двійковий код на спеціальних цифрових входах. Часто як АМХ застосовують інтегральні схеми (ІС) аналогових КМОП-ключів із цифровим вибором каналу (К590КН1, К590КН6, К561КП2), які обслуговують вісім аналогових ліній. Такі ІС мають цифровий вхід дозволу роботи E (див. схему рис. 4.14,а).

У наведеному прикладі для простоти під'єднання до МК використано восьмирозрядний інтегральний АЦП.

Часові діаграми роботи всього блока показані на рис. 4.14,б. Рівень "0" на вході Start забезпечує "скидання" АЦП, а подача "1" на вхід Start ініціює початок перетворення, яке закінчується видачею сигналу Ready=0. Тоді ж можливе зчитування вихідного коду. Аналіз сигналу готовності слід здійснювати за логікою, наведеною в попередньому прикладі.

Подібна схема керування АЦП та АМХ дозволяє реалізувати їх використання із розміщенням у зовнішньому АП (під'єднання до зовнішніх регістрів, доступ командами звертання до зовнішньої пам'яті (MOVX) або безпосереднє під'єднання до портів МК і доступ командами введення-виведення, що працюють з портами.

Під'єднання ІС АЦП до портів МК (рис. 4.14,а) в основному подібне показаному на рис. 4.12. Слід передбачити додаткові лінії портів МК для видачі сигналів керування на АМХ і сигналу E. Через те, що час перетворення в АЦП малий, застосування обміну за перериванням є неефективним; тому розробимо протокол програмно-керованого обміну з перевіркою готовності даних. Приймаємо такий розподіл ліній портів:

порт P0 – прийом у МК байта даних з АЦП (результат перетворення);
P1.0...P1.2 – видача з МК коду адреси каналу АМХ (0...7);
P1.3 – видача з МК сигналу дозволу роботи E для АМХ;
P1.4 – видача сигналу Start – скидання/початку роботи для АЦП;
P1.5 – прийом у МК сигналу Ready – готовності даних від АЦП.

Розробка програми керування СЗД полягає в точному дотриманні послідовності видачі й аналізу сигналів, яка визначається часовою діаграмою на рис. 4.14,б. Програма може бути наведена в такому вигляді:

```
; Ініціалізація
MOV P0,#0FFH  ; Порт P0 настроєно на введення
MOV P1,#00101000B ; Початкова видача сигналів через P1
```

Процедура керування перетворенням по одному каналу чекає в акумуляторі (A) адресу каналу (0...7) і повертає через акумулятор результат перетворення по цьому каналу.

```
ADT_chnl:
  orl  A, #00101000B ; Об'єднання стану порту P1
  mov  P1, A     ; з номером каналу і видача через порт
  clr  F0        ; Обнулення ознаки відмови АЦП
  nop
  clr  P1.3      ; Включення аналогового мультиплексора
  nop            ; Інтервал – 2 мкс
  nop
```

```
  setb P1.4       ; Запуск перетворення в АЦП
  mov  r7,#20
W1: djnz r7,W1    ; Затримка на 2*Тпр, тобто на 40мкс
  jb P1.5, Fault  ; Перевірка сигналу готовності
  mov  A,P0       ; Прийом результату перетворення в МК
  jmp  m_ret
Fault: setb F0    ; Встановлення ознаки відмови АЦП
  clr  A
m_ret: CLR P1.4   ; Скидання АЦП
  setb  P1.3      ; Відключення мультиплексора АМХ
  ret             ; Повернення в основну програму
```

При розробці прикладу основної програми застосовуємо процедуру ADT_chnl. Нехай канали треба опитувати по черзі від нульового до сьомого чи від сьомого до нульового. Дані, які приймаються в МК, будемо розміщувати у вигляді одновимірного масиву в РПД, починаючи з адреси, наприклад, 40H. Додатково використовуємо регістри:

R0 – для непрямої адресації масиву даних у РПД;

R3 – як лічильник каналів у режимі зворотного лічення (8→0).

Наведемо два варіанти програмної реалізації:

```
; Декларативна частина – опис таблиці в РПД
  tabl_start  DATA  40H  ; Початкова адреса
  tabl_size   EQU   8    ; Розмір таблиці
  tabl_end DATA tabl_start+tabl_size  ; Кінцева адреса


; 1) заповнення з початку  2) заповнення з кінця (7→0)
mov R0, #tabl_start          mov R0, #tabl_end
mov R3, #tabl_size           mov R3, #tabl_size
; Реалізація опитування восьми каналів
R_Tabl:          R_Tabl:
mov A, #tabl_size            mov  A, R3
clr   C                      dec  A     ;номер каналу
subb A, R3 ; номер каналу    call ADT_chnl
call ADT_chnl                mov  @R0, A
mov  @R0, A                  dec  R0
inc   R0                     djnz  R3, R_Tabl
djnz  R3, R_Tabl             ret
ret
```

4.3.3 Принципи використання АЦП із послідовним інтерфейсом

За типом цифрового інтерфейсу АЦП можна класифікувати на паралельні та послідовні. Як правило, АЦП із розрядністю 6…16 біт бувають паралельними та послідовними, а перетворювачі з розрядністю більше 16 — тільки послідовними. Це зумовлено тим, що АЦП із цифровим паралельним інтерфейсом незручно під'єднувати до обчислювача: наприклад, для під'єднання паралельного АЦП з розрядністю 24 біти потрібно використати якнайменше три восьмибітові порти обчислювача. В АЦП із послідовним інтерфейсом застосовані незалежно від розрядності всього від двох до п'яти сигнальних ліній.

Аналогово-цифрові перетворювачі з послідовним інтерфейсом можуть включати в собі такі пристрої: масив підсилювачів, що можна настроювати програмно (PGA — Programmable Gate Array); джерело опорної напруги (Ref. — Reference voltage source); засоби перетворення сигналу (це можуть бути: модулятори, фільтри, компаратори, ЦАП та ін.); послідовний інтерфейс, засоби синхронізації й тактування; схема управління або мікроконтролер. Розглянемо, наприклад, послідовний АЦП фірми Texas Instruments (Burr-Brown) ADS1210. Спрощена структурна схема перетворювача показана на рис. 4.15.

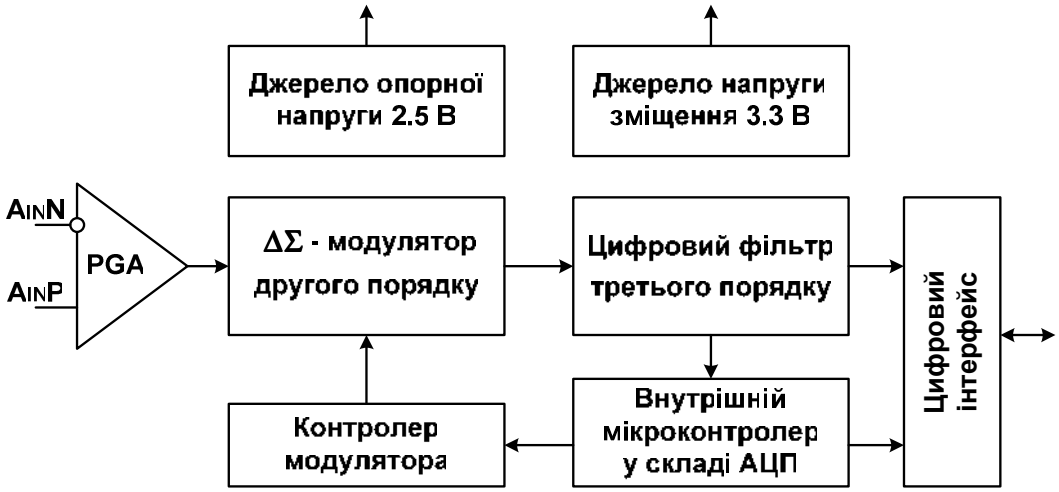

Рисунок 4.15 — Структурна схема АЦП ADS1210

Модель ADS1210 — точний, 24-бітовий сігма-дельта АЦП, що має широкий динамічний діапазон, функцію самокалібрування. Пристрій ADS1210 дозволяє отримувати результати цифрового перетворення високої якості й має вбудовані джерела опорної напруги та напруги зміщення. Цей АЦП має потужний інтегрований мікроконтролер, який дозволяє програмувати різноманітні режими роботи системи. Сігма-дельта архітектура забезпечує широкий динамічний діапазон і гарантує 22 біти точності перетворення. Ефективна точність 23 біти мо-

же бути досягнута шляхом використання малошумного операційного підсилювача на частоті перетворення 10 Гц. Умовне графічне зображення АЦП ADS1210 показано на рис. 4.16.

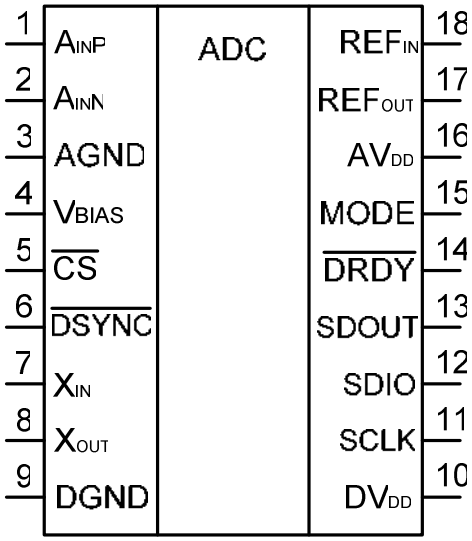

Рисунок 4.16 —Умовне графічне зображення АЦП ADS1210

Відповідно до типу сигналів виводи перетворювача можна розділити на аналогові та цифрові, а їх функціональне призначення таке:

- AINP — неінвертований вхід аналогового перетворювача;
- AINN — інвертований вхід аналогового перетворювача;
- AGND — аналогова земля (спільна точка аналогових елементів схеми, що містить АЦП);
- VBIAS — вихід напруги зміщення для схем масштабування вхідного сигналу;
- $\overline{CS}$ — вхід дозволу роботи послідовного інтерфейсу;
- $\overline{DSYNC}$ — вхід синхронізації початку аналого-цифрового перетворення;
- XIN — вхід генератора тактових імпульсів;
- XOUT — вихід генератора тактових імпульсів;
- DGND — цифрова земля (спільна точка цифрових елементів схеми, що містить АЦП);
- DVDD —живлення цифрових схем перетворювача (+5 В);
- SCLK — вхід/вихід синхронізації послідовного інтерфейсу;
- SDIO — вхід/вихід даних послідовного інтерфейсу;
- SDOUT — вихід даних послідовного інтерфейсу;
- $\overline{DRDY}$ — вихід, ознака завершення перетворення;
- MODE — вхід, вибір режиму роботи послідовного інтерфейсу;

- AVDD — живлення аналогових схем перетворювача (+5 В);
- REFOUT — вихід джерела опорної напруги живлення (+2.5 В);
- REFIN — вхід опорної напруги живлення.

Послідовний інтерфейс мікросхеми АЦП ADS1210 може працювати в двох основних режимах: ведений та ведучий. Розглянемо ведений режим. Для переведення мікросхеми в цей режим необхідно подати на вхід MODE *низький рівень сигналу*. Мінімальний набір сигналів, необхідних для роботи з АЦП, включає в себе лінії SDIO та SCLK. Однак у такому режимі ускладнюється протокол обміну між АЦП і МК. Тому на практиці використовують повний набір сигналів для обміну даними та командами з АЦП. При цьому будуть задіяні лінії SCLK, SDIO, SDOUT, $\overline{\text{DRDY}}$. Приклад схеми під'єднання АЦП до мікроконтролера групи MCS-51 показано на рис. 4.17.

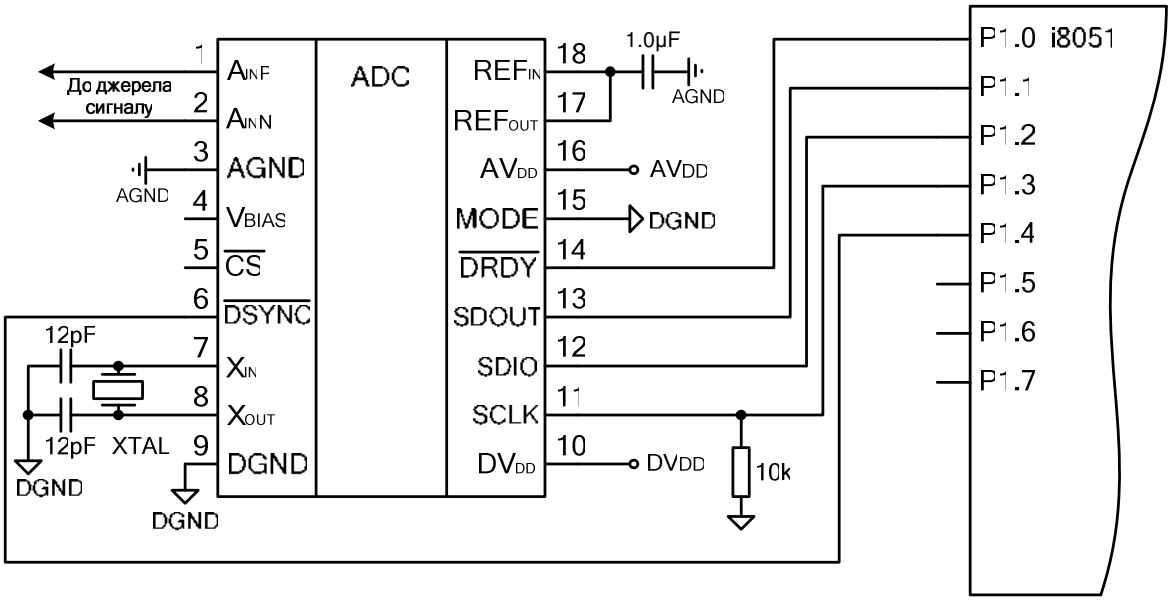

Рисунок 4.17 — Приклад схеми використання АЦП ADS1210

Лінії SDOUT і SDIN використовують для передачі й прийому даних, лінію SCLK – для синхронізації передачі. Низький рівень на лінії $\overline{\text{DRDY}}$ вказує на готовність даних у буфері перетворювача, а низьким рівнем на лінії $\overline{\text{DSYNC}}$ мікроконтролер може ініціювати початок перетворення в АЦП. Вхідний аналоговий сигнал подається на лінії AINP та AINN у диференціальному вигляді. Якщо цей сигнал змінюється в діапазоні від нуля вольт до напруги живлення перетворювача, то вивід AINN можна з'єднати або із загальним проводом аналогової частини пристрою, або з виходом джерела опорної напруги 2.5В; у першому випадку цифровий код на виході АЦП буде змінюватися в діапазоні [0,0xFFFFFF] в форматі «ціле беззнакове», а у другому випадку

— від [0x800000,0x7FFFFF], тобто буде задаватися у форматі «ціле із знаком». При роботі АЦП у вибраному режимі використовують протокол послідовної передачі даних SPI, причому з боку мікроконтролера цей протокол можна реалізувати як апаратно, так і програмно. У будь-якому випадку необхідно настроїти приймач-передавач SPI мікроконтролера на такий режим: Master, CPOL=0, CPHA=1, частота сигналу синхронізації має бути менше тактової частоти перетворювача. Інтерфейс АЦП реалізовано як набір з п'яти регістрів:

- **INSR** (Instruction Register) — регістр інструкцій (8 біт);
- **DOR** (Data Output Register) — регістр видачі даних (24 біти);
- **CMR** (Command Register) — регістр команд (32 біти);
- **OCR** (Offset Calibration Register) — регістр калібрування зміщення (24 біти);
- **FCR** (Full-Scale Calibration Register) — регістр загального калібрування (24 біти).

Названі регістри утворюють масив байт з адресами: 0x00 — 0x0E. Регістр інструкцій **INSR** керує читанням і записом даних через послідовний інтерфейс. Запис байта в цей регістр ініціює процедуру читання або запису декількох (від одного до чотирьох) байтів інформації у задану адресу. Регістр команд **CMR** настроює АЦП, запис інформації в цей регістр дозволяє керувати:

- роботою схеми формування напруги зміщення V_{BIAS};
- роботою схеми формування опорної напруги (REFO);
- форматом даних у регістрі DOR (DF): із знаком чи без;
- режимом роботи АЦП ($U/\overline{B}$): однополярний чи двополярний;
- порядком передачі байтів (BD) у послідовному каналі: старший вперед чи молодший вперед;
- порядком передачі бітів (MSB) у послідовному каналі: старший вперед чи молодший вперед;
- режимом роботи ліній передачі даних SDI, SDIO;
- синхронізацією початку перетворення;
- режимом роботи АЦП: нормальний, самокалібрування, калібрування системного зміщення, загальне системне калібрування, фонове калібрування, «сплячий режим»;
- коефіцієнтом підсилення перетворювача: 1, 2, 4, 8, 16;
- режимом прискорення перетворення, раз: 1, 2, 4, 8, 16;
- коефіцієнтом децимації (ціле число в діапазоні 19 … 8000).

При читанні регістра **CMR** можна отримати інформацію про стан ознаки завершення перетворення, як альтернатива використанню додаткової лінії порту (сигнал $\overline{DRDY}$). Регістр **DOR** містить результат

попереднього перетворення. Регістри **OCF** і **FCR** містять коефіцієнти, що були отримані в результаті операцій калібрування АЦП. Ці дані адекватні тільки для режиму роботи АЦП, при якому здійснювалося калібрування. Тому при зміні швидкості перетворення, коефіцієнтів підсилення, прискорення, децимації, а також умов експлуатації (температура, величина напруги живлення) слід здійснити калібрування знову. Результат калібрування може бути зчитаний із регістрів і записаний до енергонезалежної пам'яті контролера, що дозволить у деяких випадках не виконувати калібрування АЦП при включенні. Приклад програми мовою С, яка здійснює настроювання, калібрування і введення даних з АЦП, наведено нижче.

```
// Байти ініціалізації АЦП
// Включення виводу REF і вибір режиму Slave
#define CMR_INIT_3 (CMR_REFO | CMR_SDL)
// Ініціалізація режиму (MODE) Self-Calibartion (001);
// підсилення (GAIN) = 1; Канал (CH) = 0
#define CMR_INIT_2 (CMR_MD0)
// Встановлюється Turbo Mode = 16; Decimation Rate = 255
#define CMR_INIT_1 (CMR_SF2)
#define CMR_INIT_0 255

void ADC_Init(void) {
  // Дозвіл роботи SPI в режимі MSB, Master, CPOL=0, CPHA=1,
  // частота fXin/16
  SPCR = (1 << SPE) | (1 << MSTR) | (1 << CPHA) | (1 << SPR0);
  // Конфігурування ADC
  // Запис у командний регістр
  SPDR = INSR_MB1 | INSR_MB0 | ADC_CMR;
  // запис 4 байт у регістр CMR
  while (!(SPSR & (1 << SPIF)));
  SPDR = CMR_INIT_3; // старший байт
  while (!(SPSR & (1 << SPIF)));
  SPDR = CMR_INIT_2;
  while (!(SPSR & (1 << SPIF)));
  SPDR = CMR_INIT_1;
  while (!(SPSR & (1 << SPIF)));
  SPDR = CMR_INIT_0; // молодший байт
  while (!(SPSR & (1 << SPIF)));
  // Чекання завершення режиму самокалібрування
  while (P0 & ADC_DRDY);
}

int32_t ADC_In(uint8_t channel) {
  // Чекання завершення чотирьох циклів перетворення
  while (!(P0 & ADC_DRDY));
  while (P0 & ADC_DRDY);
```

```
    // Зчитування даних
  int32_t res;
  // Запис у командний регістр
  // Команда читання трьох байтів із регістра DOR
  SPDR = INSR_RW | INSR_MB1 | ADC_DOR;
  while (!(SPSR & (1 << SPIF)));
  // Читання трьох байтів результату перетворення:
  // другий байт
  SPDR = 0;
  while (!(SPSR & (1 << SPIF)));
  res = ((uint32_t)SPDR) << 16;
  // перший байт
  SPDR = 0;
  while (!(SPSR & (1 << SPIF)));
  res += ((uint32_t)SPDR) << 8;
  // нульовий байт
  SPDR = 0;
  while (!(SPSR & (1 << SPIF)));
  res += SPDR;

  return (res);
}
```

4.3.4 Використання МК із вбудованими АЦП

Аналого-цифровий перетворювач у складі МК групи MCS-51/52 (наприклад, у моделі SAB80515 фірми Siemens) забезпечує 8-бітове перетворення і має вісім мультиплексних аналогових входів. Крім того, АЦП має вбудовану схему вибирання-зберігання та можливість програмного задавання опорних напруг, що підвищує точність при звуженні меж вимірювання сигналів. Перетворення здійснюється методом послідовного наближення. Тривалість циклу перетворення – від 15 до 29 МЦ.

У складі вбудованого АЦП є три регістри, доступних для програміста:

- ADCON – регістр керування АЦП;
- ADDAT – регістр даних для результату перетворення;
- DAPR – регістр програмування опорних напруг.

Регістр керування АЦП ADCON розміщений у логічній області РСФ за адресою 0D8h і має бітову адресацію, тобто імена бітів можна використовувати в командах програми. Призначення бітів наведено в табл. 4.1.

Регістр ADDAT фіксує 8-бітовий результат перетворення в АЦП. Попередній результат зберігається в регістрі до появи нового результату. Регістр доступний як для зчитування, так і для запису.

Таблиця 4.1 – Призначення бітів регістра керування АЦП ADCON

Біт	Призначення біта
MX0 (0) MX1 (1) MX2 (2)	Вибір каналу аналогового вхідного сигналу. Номер активного каналу відповідає двійковому значенню, записаному в біти (MX2)-(MX1)-(MX0). Наприклад, ‘100’ – канал 4
ADM (3)	Вибір режиму перетворення. При ADM = 1 – безперервне перетворення. При ADM = 0 – одноразове перетворення
BSY (4)	Ознака зайнятості АЦП. Коли BSY=1, триває перетворення, при BSY=0 перетворення немає (попереднє перетворення було завершене)
ADS (5)	Запуск/зупин перетворення. При запису ‘1’ в ADS починається перетворення (циклічно повторюється при ADM=1). При ADM=0 для нового перетворення слід скинути ADS в ‘0’, а потім знову встановити в ‘1’
CLK (6)	Використовується не для керування АЦП. При ‘1’ активується генерація імпульсів із Ft/12 на виводі P1.6
BD (7)	Використовується не для керування АЦП. Включення режиму передачі через УАПП зі швидкістю в бодах

Регістр DAPR дозволяє змінювати внутрішні опорні напруги IVAREF та IVAGND. Вони можуть програмуватися з кроком 1/16 відносно зовнішніх опорних напруг VAREF і VAGND (як такі можуть використовуватися напруга джерела живлення VCC і GND).

Біти з 0 по 3 визначають IVAGND, а біти 4-7 – значення IVAREF. При цьому ці напруги обов’язково мають відрізнятися на 1В.

Параметри IVAGND та IVAREF визначаються формулами:

$$IVAGND = VAGND + (DAPR_{0\text{-}3} / 16)*(VAREF - VAGND),$$

$$IVAREF = VAGND + (DAPR_{4\text{-}7} / 16)*(VAREF - VAGND),$$

причому $0< DAPR_{0\text{-}3} <13$ та $3<DAPR_{4\text{-}7}$.

Перетворення починається при встановленні ADS=1 і триває від 15 до 29 МЦ при тактовій частоті 12МГц залежно від встановлених опорних напруг. Під час перетворення ознака BSY=1. По завершенні перетворення результат записується в регістр ADDAT, ознака BSY=0 і встановлюється запит переривання IADC у регістрі IRCON – додатковому регістрі керування перериванням у МК цієї моделі.

Особливість реалізації АПЦ у МК групи MCS-51 полягає в тому, що в моделях різних виробників застосовані різні регістри для керування АЦП (за назвами, структурою та фізичними адресами).

Так, у МК виробництва Philips результат перетворення по кожному з восьми каналів фіксується в окремому 8-бітовому регістрі, тому загальна кількість регістрів, що обслуговують АЦП, дорівнює 10. Крім того, є можливість запуску АЦП від зовнішнього сигналу. На базі блока АЦП може здійснюватися порівняння будь-якого з вхідних аналогових сигналів з окремою опорною напругою (режим компаратора).

У мікроконтролерах серії ADuC8xx розробки Analog Devices блок АЦП є 8-канальним і 12-бітовим, тому в структурі МК реалізовано два регістри для фіксації результату перетворення. Крім того, для задавання різних режимів роботи АЦП передбачено три регістри керування АЦП. Ефективним є режим циклічного перетворення із прямим доступом у зовнішню пам'ять системи на основі МК. У цьому режимі результати АЦ-перетворення автоматично записуються як масив даних у ЗПД без участі процесора, тобто без відповідних програмних дій. Тому для використання АЦП завжди слід ретельно вивчити документацію на конкретну вибрану модель МК. Певним недоліком АЦП, вбудованих у МК, є невисока точність перетворення.

4.3.5 Прийом аналогових сигналів у МК на основі ШІМ

Існує декілька способів введення аналогових сигналів у цифрову систему, альтернативних аналого-цифровому перетворенню. Одним із них є формування широтно-модульованих імпульсів за аналоговим сигналом і вимірювання їх тривалості мікропроцесорним пристроєм. Перевагами такого підходу є:

- простота й дешевизна схемотехнічної реалізації, особливо у випадках, коли використовують МК без вбудованого АЦП;
- можливість передачі ШМ-сигналу на значну відстань без суттєвих перешкод (оскільки частота є невеликою, а крутизну фронтів легко забезпечити тригером Шмітта); це зручно, якщо МК-регулятор неможливо розмістити поблизу від датчика аналогового сигналу;
- застосування будь-яких датчиків, що формують вихідний сигнал у вигляді напруги (на відміну від перетворювачів на основі ЧІМ, які працюють тільки з датчиками резистивного чи ємнісного типу – див. розд. 4.3.6).

Передумовою застосування перетворювачів ШІМ є наявність у складі будь-яких мікроконтролерів таймерів, на основі яких доволі легко здійснити вимірювання ширини імпульсів.

Узагальнений принцип апаратного перетворення аналогового сигналу в послідовність ШМ-імпульсів показано на рис. 4.18.

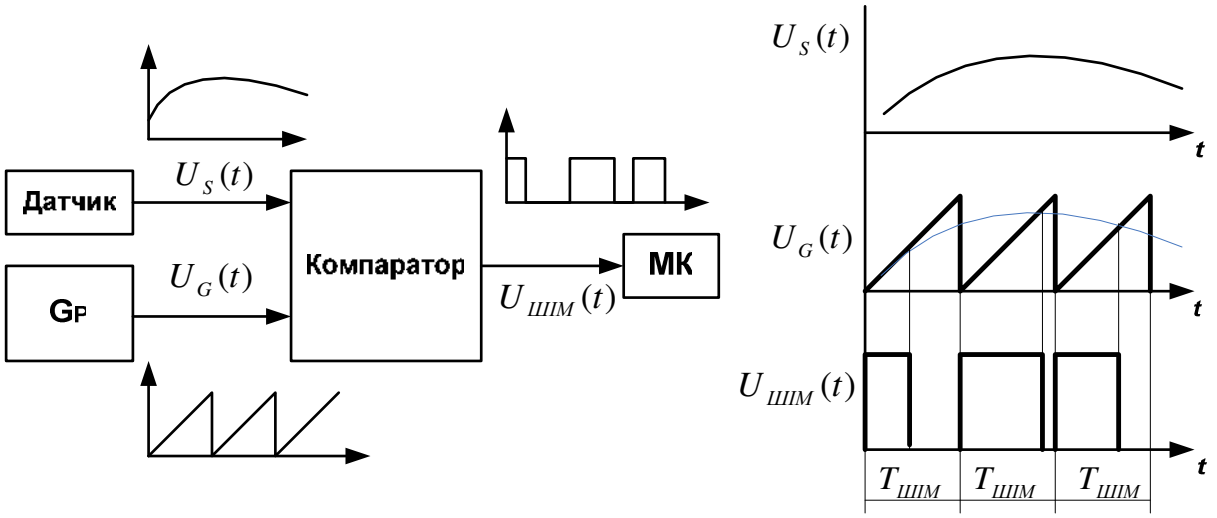

Рисунок 4.18 – Принцип формування вхідних ШМ-імпульсів для МК

Блок **Gp** – це генератор трикутних імпульсів, які задають період ШМ-сигналів, що подаються на МК.

Якщо тривалість активної (“1”) фази ШМ-імпульсу в кожному періоді $T_{ШІМ}$ позначити як $\tau(t)$, то можна сформувати такі залежності: $\tau(t) = K_{ШІМ} \cdot U_S(t)$ – зв'язок ширини імпульсу і рівня аналогового сигналу з датчика; $K_{ШІМ} = T_{ШІМ} / U_{G\max}$ – визначення коефіцієнта широтно-імпульсної модуляції.

Проста схема формувача ШМ-імпульсів на операційних підсилювачах (ОП) показана на рис. 4.19. До його складу додатково введено пристрій узгодження для приведення діапазону сигналу $U_S(t)$ до вихідного діапазону генератора трикутних імпульсів $U_G(t)$, тобто $U_{G\,MAX}$.

Період імпульсів, що формуються, визначається виразом

$$T_{ШІМ} = 2R_4C_1 \ln(1 + 2\frac{R_5}{R_6}),$$ а частота ШІМ $$F_{ШІМ} = \frac{1}{T_{ШІМ}}.$$

Бажано підбирати співвідношення U_S до $U_{G\,MAX}$ так, щоб забезпечити $U_{S\,MAX} < U_{G\,MAX}$. При цьому буде досягнуто наявність інтервалу часу Δt:

$$\Delta t = T_{ШІМ} - \tau_{\max}, \quad де \ \ \tau_{\max} = K_{ШІМ} \cdot U_{S\max}.$$

Цей інтервал необхідний для гарантованого виконання процедури обробки переривання від таймера МК, який здійснює вимірювання ширини вхідних ШМ-імпульсів. Принципи та приклади програмної реалізації такого вимірювання детально описані у розд. 4.2.

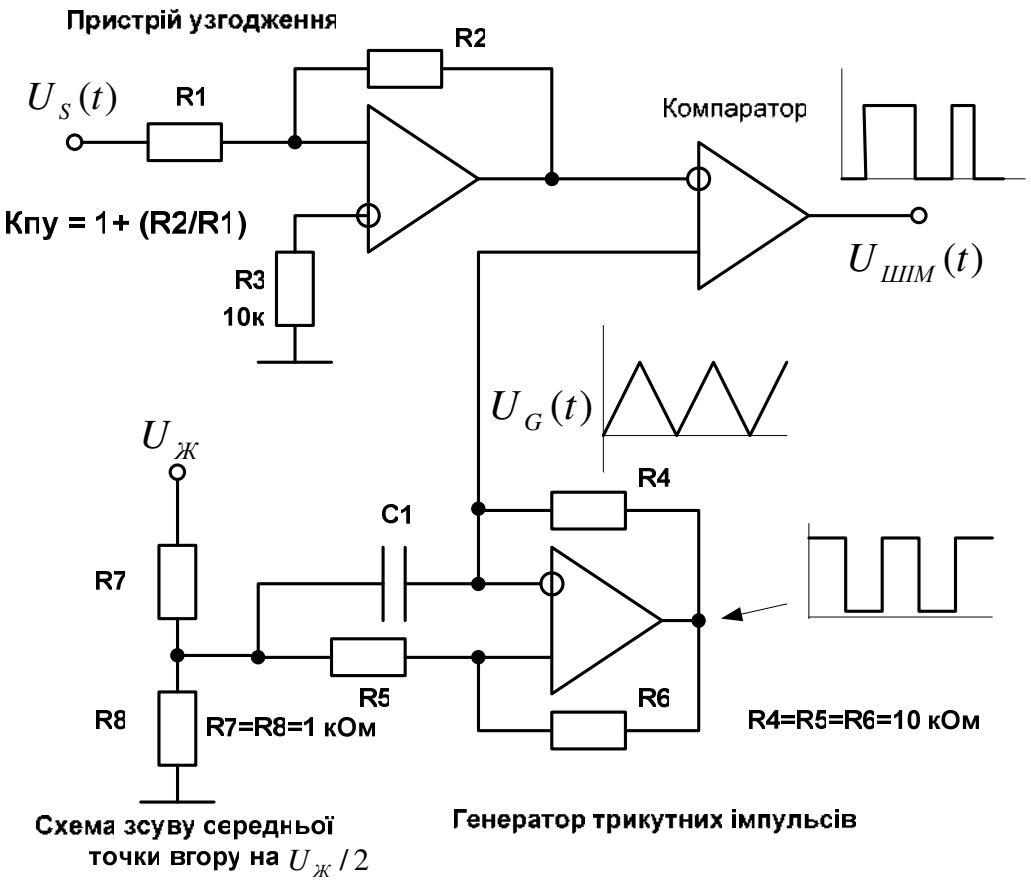

Рисунок 4.19 – Формувач ШМ-імпульсів за аналоговим сигналом

4.3.6 Прийом аналогових сигналів у МК на основі ЧІМ

Інформація про деякий фізичний параметр може надходити в МК у вигляді послідовності імпульсів змінної частоти. Схемна реалізація ЧІМ (рис. 4.20) простіша за ШІМ. Але якщо вхідний аналоговий сигнал для ШІМ можна отримувати від будь-якого джерела напруги, то датчиком у ЧІМ може бути лише пристрій зі змінним опором (датчик потенціометричного або резистивного типу).

У випадку застосування ЧІМ задачею МК є вимірювання інтервалу часу між двома послідовними зовнішніми імпульсами (вимірювання періоду зовнішніх імпульсів) або підрахунок кількості імпульсів за фіксований інтервал часу (вимірювання частоти зовнішніх імпульсів). Вихід формувача типу ЧІМ зазвичай подають на вхід зовнішнього переривання (для вимірювання періоду) або на лічильний вхід таймера МК (для вимірювання частоти). Детально всі варіанти алгоритмічної і програмної реалізації для вимірювання частоти або періоду імпульсів розглянуті у розд. 4.2.

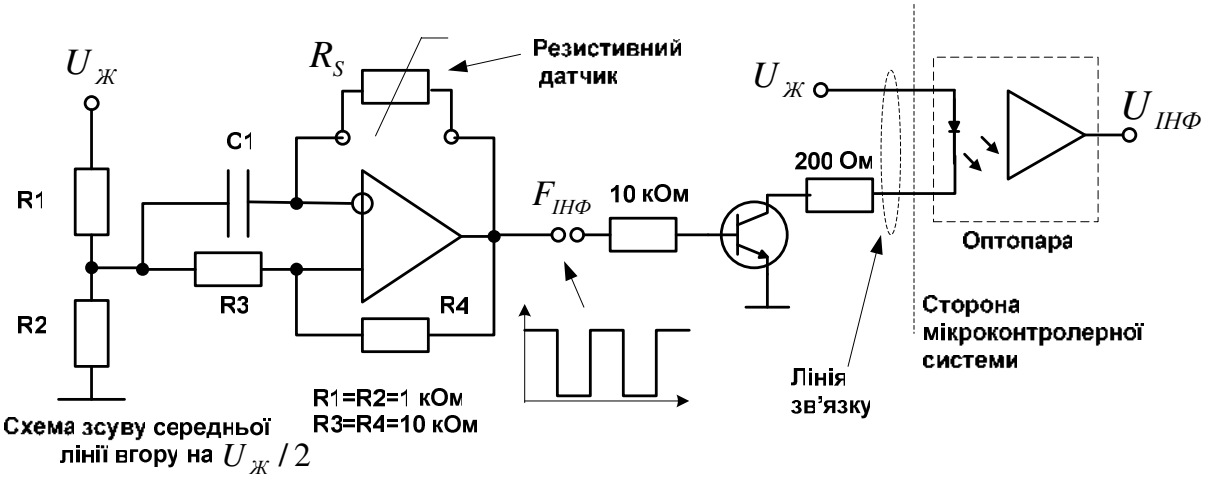

Рисунок 4.20 – Формувач ЧМ-імпульсів від резистивного датчика із вихідним інтерфейсом «ланцюг струму»

Період імпульсів, що формуються пристроєм, визначається аналогічно схемі на рис. 4.19:

$$T_{ЧІМ} = 2R_S C_1 \ln(1 + 2\frac{R_3}{R_4}) = 2{,}197 \cdot R_S C_1 \ .$$

Як правило, частота інформаційних імпульсів $F_{ІНФ}$ від вимірювальних перетворювачів (ЧІМ) значно менше частоти еталонних імпульсів від ГТІ МК: $F_{ІНФ} << F_{ГТІ}$. Тому вимірювання частоти $F_{ІНФ}$ здійснюють шляхом підрахунку кількості еталонних імпульсів від ГТІ МК, що вмістилися в інтервал $T_{ЧІМ}$ між двома зовнішніми імпульсами.

У інтерфейсі «ланцюг струму» сигнал передається як імпульси струму (0…20 мА). Це полегшує обмеження на довжину лінії зв'язку (до сотень метрів), захищає від електромагнітних перешкод. Оптопара реалізує гальванічне розгалуження перетворювача та МК, що дозволяє застосувати для перетворювача джерело живлення з напругою, що відрізняється від живлення МК.

4.3.7 Відновлення значення вимірюваного фізичного параметра

Цифрові дані, отримані в результаті аналого-цифрового перетворення або вимірювання тривалості ШМ-імпульсу (періоду ЧМ-імпульсів), необхідно перерахувати в значення того фізичного параметра, який вимірюється системою.

У даному розділі розглянемо системи з датчиками, які мають лінійну характеристику перетворення. Функція перетворення вихідного коду АЦП у значення фізичного параметра для лінійної залежності має вигляд

$$P = K_P \cdot C + B_P ,$$

де ***C*** – код, отриманий з АЦП, ***P*** – числове значення фізичного параметра.

$$K_P = \frac{P_{MAX} - P_{MIN}}{C_{MAX} - C_{MIN}}; \qquad B_P = P_{MIN} - K_P \cdot C_{MIN}.$$

C_{MIN}, C_{MAX} – діапазон вихідного коду АЦП, відповідний конкретному діапазону вихідного сигналу датчика D_{MIN} ... D_{MAX}. Якщо ввести до розгляду I^M_{MIN}, I^M_{MAX} – максимально можливий діапазон вхідних сигналів АЦП, C^M_{MIN}, C^M_{MAX} – максимально можливий діапазон відповідного вихідного коду АЦП, та коефіцієнт передачі АЦП $K_{АЦП}$, то

$$C_{MIN} = C^M_{MIN} + (D_{MIN} - I^M_{MIN}) \cdot K_{АЦП};$$

$$C_{MAX} = C^M_{MAX} + (D_{MAX} - I^M_{MAX}) \cdot K_{АЦП};$$

$$K_{АЦП} = \frac{C^M_{MAX} - C^M_{MAX}}{I^M_{MAX} - I^M_{MAX}}.$$

При цьому слід дотримуватись співвідношень

$$I^M_{MIN} < D_{MIN} < D_{MAX} < I^M_{MAX};$$

$$C^M_{MIN} < C_{MIN} < C_{MAX} < C^M_{MAX}.$$

Якщо датчик має суттєво нелінійну характеристику, то функція перетворення може бути задана у вигляді апроксимуючого полінома на всьому інтервалі вимірювання параметра чи на окремих інтервалах значень.

При вимірюванні фізичного параметра на основі ШІМ і ЧІМ для відновлення значення фізичного параметра необхідно знати діапазон значень ширини або частоти імпульсів, які визначаються схемотехнікою модулятора (рис.4.19 і 4.20). Необхідно також знати мінімальне та максимальне значення вимірюваного параметра P_{MIN}, P_{MAX}.

Після розрахунку схеми модулятора та її виготовлення треба здійснити калібрування каналу вимірювання. Бажано використовувати датчики, які працюють на лінійній ділянці у вимірюваному діапазоні фізичного параметра. Необхідно експериментально визначити кількість імпульсів *N* для граничних значень вимірювання параметра: N_{MIN}, N_{MAX}. Тоді значення виміряного фізичного параметра ***P*** можна відновити як

$$P = K_{PN} \cdot N + B_{PN},$$

де $K_{PN} = \dfrac{P_{MAX} - P_{MIN}}{N_{MAX} - N_{MIN}}$ та $B_{PN} = P_{MIN} - K_{PN} \cdot N_{MIN}$.

Важливо, що коефіцієнти K_{PN}, B_{PN} у даному випадку враховують як характеристику датчика, так і коефіцієнт перетворення ШІМ або ЧІМ. Їх значення можуть бути підставлені в остаточний варіант програми керування об'єктом чи обладнанням у вигляді констант.

4.4 Виведення аналогових сигналів із МК

4.4.1 Використання зовнішніх паралельних ЦАП

Типові структури блоків цифро-аналогового перетворення (ЦАП) і виведення сигналів показані на рис. 4.21.

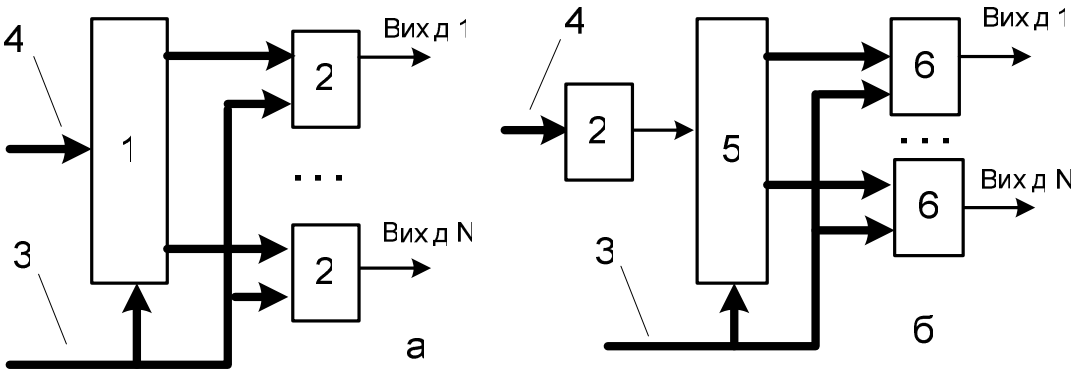

Рисунок 4.21 – Типові структури цифро-аналогових вихідних блоків: 1 – цифровий комутатор; 2 – цифро-аналоговий перетворювач (ЦАП) із вхідним регістром даних; 3 – сигнали керування від обчислювача; 4 – цифровий код; 5 – аналоговий комутатор; 6 – екстраполятор

Схема на рис. 4.21,а з паралельно включеними перетворювачами має більшу вартість, але забезпечує високу точність формування вихідних сигналів завдяки можливості індивідуального підбору характеристик кожного ЦАП. Цифровий комутатор (демультиплексор) забезпечує з'єднання паралельних цифрових вхідних ліній кожного ЦАП із шиною даних обчислювача. Функцію цифрового демультиплексора може виконувати адресний селектор. Необхідно, щоб блоки ЦАП при цьому мали вбудовані засоби фіксації цифрового коду на весь період дискретності – паралельні регістри.

Схема з послідовним перетворенням (рис. 4.21,б) дешевше внаслідок використання одного ЦАП у режимі розподілу часу. Однак до складу схеми мають входити засоби фіксації і згладжування аналогових сигналів на основі екстраполяторів чи ПВЗ, які забезпечують по-

стійний рівень аналогового сигналу на кожній вихідній лінії протягом всього періоду дискретності, поки ЦАП обслуговує інші канали.

Блоки (інтерфейси) ЦА-перетворення зазвичай характеризуються тим, що мікросхеми ЦАП є пристроями безперервної дії, тобто не потребують спеціальних сигналів керування для запуску чи скидання. Двійковий код, поданий на цифрові входи ЦАП, безперервно перетворюється у рівень струму чи напруги на його аналоговому виході.

Мікросхема ЦАП може бути підключена безпосередньо до портів МК або до портів виведення (регістрів, ППІ), логічно розміщених у зовнішньому адресному просторі МК. Якщо розрядність ЦАП перевищує розрядність порту чи регістра, необхідно застосовувати додатковий порт чи регістр для зберігання молодших (чи старших) розрядів коду, що видається (рис. 4.22).

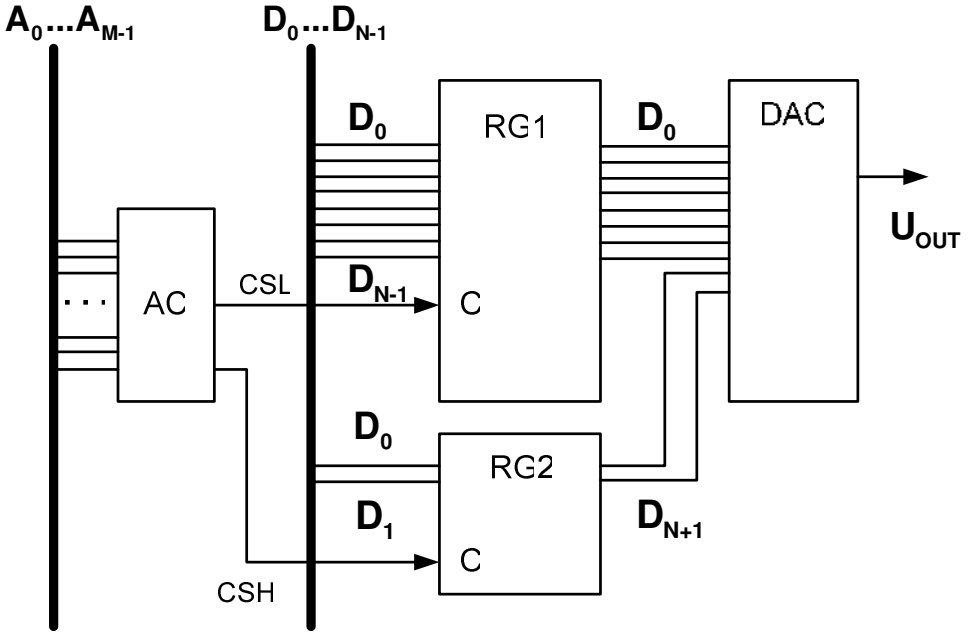

Рисунок 4.22 – Приклад вихідного цифро-аналогового інтерфейсу

При цьому пересилання двійкового коду на ЦАП слід виконувати в два етапи – по черзі молодший і старший байти у регістри RG1 та RG2 за відповідними адресами.

Нехай для прикладу результат для виведення на ЦАП як двобайтове число міститься за адресою 40h у РПД МК, а регістри ЦАП мають системні адреси 8000h і 8001h. Тоді виведення коду на ЦАП можна здійснити командами програми:

```
mov  dptr,#8000h   ; Задавання адреси регістра RG1
mov  a,40h
movx @dptr,a    ; Виведення молодшого байта на ЦАП
inc  dptr       ; Формування адреси регістра RG2
```

```
mov  a,41h
anl  a,#00000011b  ; Виділення розрядів
movx @dptr,a   ; Виведення старшого байта на ЦАП
```

4.4.2 Використання ЦАП із послідовним інтерфейсом

Застосування ЦАП із послідовним інтерфейсом дозволяє зменшити кількість ліній при підключенні ЦАП до мікроконтролера. Крім того, це дозволяє розробникам мікросхеми розмістити декілька перетворювачів в одному корпусі без суттєвого збільшення кількості виводів мікросхеми. Як приклад розглянемо ЦАП фірми Texas Instruments (Burr-Brown) DAC7512. Ця мікросхема являє собою 12-бітовий послідовний ЦАП, побудований за схемою «ланцюг резисторів». Структурна схема ЦАП показана на рис. 4.23. Цифровий код передається на ЦАП від мікроконтролера за протоколом SPI (виводи $\overline{\text{SYNC}}$, SCLK, D_{IN}). Вихідна напруга знімається з виводу V_{OUT}. Вивід V_{DD} одночасно є лінією живлення схеми і входом опорної напруги перетворювача. Вивід GND підключається до «загального» проводу схеми.

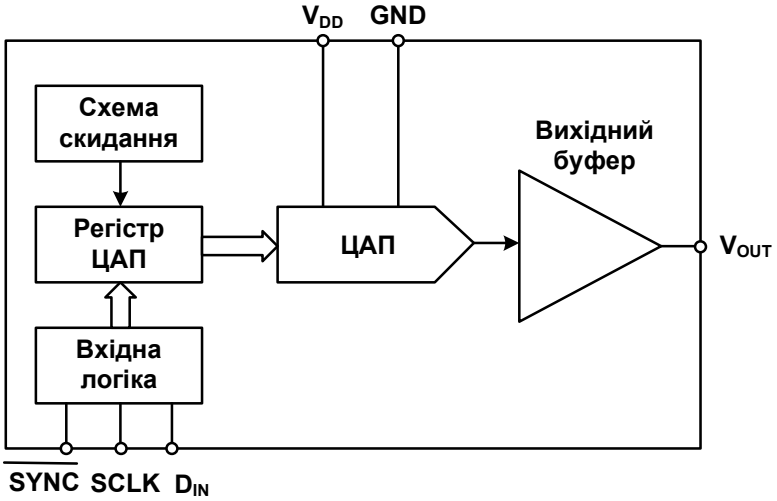

Рисунок 4.23 — Структурна схема ЦАП DAC7512

Приклад схеми використання ЦАП DAC7512 із МК MCS-51 показано на рис. 4.24.

Для передачі даних на ЦАП послідовний інтерфейс SPI мікроконтролера слід перевести в такий режим: Master, MSB, CPOL=0, CPHA=1. Передача бітів даних здійснюється протягом наявності низького рівня сигналу дозволу $\overline{\text{SYNC}}$. Кожний біт даних на лінії D_{IN} супроводжується зрізом на лінії синхронізації SCLK. Таким чином, передаються 16 бітів даних, чотири старші з яких – холості.

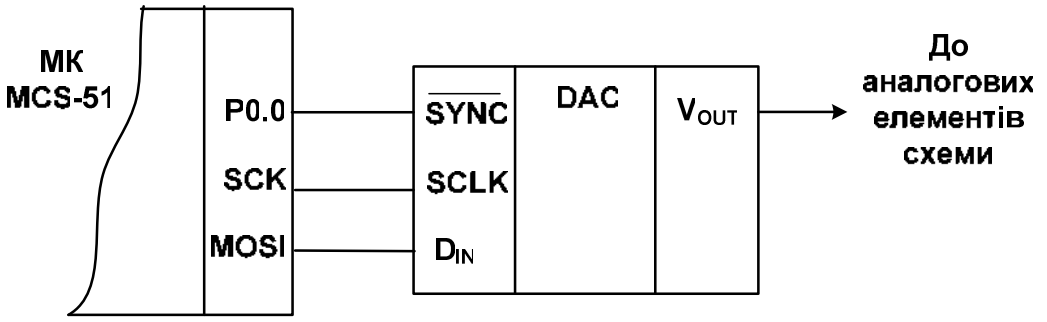

Рисунок 4.24 — Схема використання ЦАП DAC7512

Приклад програми, що виконує передачу цифрового коду на ЦАП, дано нижче.

```
void DAC_Out(uint16_t value) {
  // Настроювання SPI на режим MSB, Master, CPOL=0, CPHA=1
  SPCR = (1 << SPE) | (1 << MSTR) | (1 << CPHA);
  // Запис старшого байта цифрового коду у ЦАП
  SPDR = (uint8_t)(value >> 8);
  while (!(SPSR & (1 << SPIF)));
  // Запис молодшого байта цифрового коду у ЦАП
  SPDR = (uint8_t)(value & 0xff);
  while (!(SPSR & (1 << SPIF))); }
```

4.4.3 Використання МК із вбудованими ЦАП

У деяких моделях МК групи MCS-51 наявні вбудовані ЦАП, що дозволяють видавати аналогові сигнали у вигляді напруги безпосередньо із мікросхеми МК. Як приклад розглянемо МК ADuC812 (Analog Devices), який містить два ідентичних 12-розрядних ЦАП. Керування обома ЦАП здійснюється регістром DACCON (табл. 4.2).

Таблиця 4.2 – Призначення бітів регістра керування АЦП ADCON

Біт	Призначення біта
Mode (7)	Встановлює режим роботи обох ЦАП. Якщо Mode=1, то 8-розрядний режим (тільки на DACxL), при Mode=0 – 12-розрядний режим
RNG1 (6) RNG0 (5)	Біти вибору діапазону для ЦАП-1 і ЦАП-0 відповідно. При RNGx=1 діапазон 0...Vcc, при RNGx=0 – 0…Vref
CLR1 (4) CLR0 (3)	Біти обнулення ЦАП-1 і ЦАП-0 відповідно. При CLRx=0 на виході ЦАП-х встановлюється 0В
SYNC (2)	Біт синхронізації ЦАП-0 і ЦАП-1 (див. коментар нижче)
PD1 (1) PD0 (0)	Біти запуску/вимикання ЦАП=1 і ЦАП-0 відповідно. При PDx=1 ЦАП-х працює, при PDx=0 ЦАП-х вимкнено

Дані для цих ЦАП у вигляді беззнакових чисел задають у регістрах (DAC0L,DAC0H) для ЦАП-0 і (DAC1L,DAC1H) для ЦАП-1. Синхронне оновлення аналогового сигналу на виході обох ЦАП задається бітом SYNC у регістрі DACCON. Якщо встановити SYNC=0, то можна записати нові дані в DACxL і DACxH окремими командами, і тоді при задаванні SYNC=1 відповідні аналогові сигнали будуть сформовані на виходах обох ЦАП. Якщо SYNC=1 статично, то вихідні сигнали ЦАП оновлюються при записі байта в молодший регістр DACxL окремо для кожного ЦАП. Тому для 12-розрядного режиму при індивідуальному керуванні кожним ЦАП слід спочатку записувати дані в старший регістр DACxH, а потім – в DACxL. У регістрах DACxH використовують молодшу тетраду.

4.4.4 Використання ШІМ для формування аналогових сигналів

У сучасних моделях МК як групи MCS-51, так і інших груп для формування аналогових вихідних сигналів застосовують широтно-імпульсні модулятори (ШІМ). Це пов'язано як із простою схемотехнікою вихідних ШІМ (таймери, цифрові компаратори), так і із зручністю використання широтно-модульованих (ШМ) сигналів для керування виконавчими пристроями. За необхідності ШМ-сигнал може бути згладжений.

Формувати вихідні ШМ-сигнали можна такими способами:
1) побудова зовнішніх схем ШІМ;
2) застосування апаратних ШІМ у складі деяких моделей МК;
3) програмна підтримка формування ШМ-сигналів на таймері МК.

Більшість МК групи MCS-51 не мають вбудованих ШІМ. Тому треба будувати зовнішні схеми та під'єднувати їх до портів МК.

1. Функціональна схема одного з можливих варіантів зовнішнього ШІМ показана на рис. 4.25. Генератор **G** формує неперервну послідовність імпульсів із частотою F_G. Ці імпульси підраховує лічильник **Cnt** розрядністю N_C. Стан лічильника (код $\mathbf{C_C}$) дискретно лінійно змінюється від 0 до 2^{N_C}. Формування ШМ-імпульсів здійснюється цифровим багаторозрядним компаратором. Група входів A приймає двійковий код від МК – C_{MK}, що виражає значення параметра керування виконавчим пристроєм, а група входів B – двійковий код C_C, тобто поточний стан лічильника. Розрядність компаратора також дорівнює N_C .

Якщо $C_{MK} > C_C$, то вихід компаратора **Out** = 1, якщо $C_{MK} < C_C$, то вихід компаратора **Out** = 0. Розрядність лічильника N_C і вихідного коду від МК N_{MK} мають збігатися. Частота ШМ-сигналу, що формується

$F_{ШІМ} = \frac{F_G}{2^{N_C}}$, а період $T_{ШІМ} = 2^{N_C} \cdot T_G$. Точність формування становить $\delta = 2^{-(N_C+1)} \cdot 100\%$.

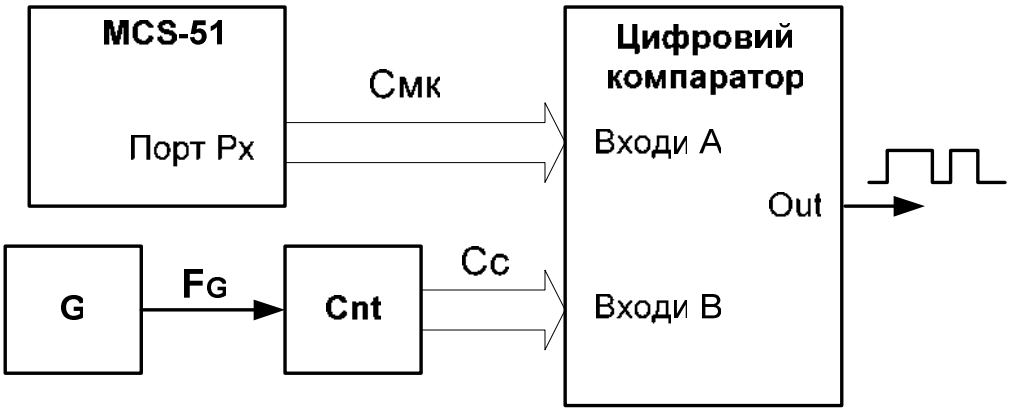

Рисунок 4.25 – Структура зовнішнього ШІМ-генератора для МК

Програмна підтримка формування ШМ-сигналів полягає у видачі коду керування $C_{МК}$ через порт Px, до якого підключений ШІМ.

Імпульсами для лічильника **Cnt** можуть бути навіть вихідні імпульси ALE, що автоматично формуються МК. Їх частота становить $F_G = F_{ALE} = F_Q/6$, де F_Q – тактова частота МК.

2. У багатьох МК групи MCS-51 розробки Philips (8xC51Fx, GB) чи Siemens, а також Analog Devices реалізовані так звані блоки PCA (programmable counter array) – масиви таймерів-лічильників для формування ШМ-імпульсів. У складі блока PCA є таймер-лічильник і декілька однотипних модулів фіксації-порівняння (МФП) на основі регістрів **CAPx** і компараторів. Кожний із МФП формує імпульси на окремій лінії порту МК **Pi.k**. Таким чином, структура кожного лічильника у PCA близька до показаної на рис. 4.25.

Робота із модулем PCA для формування ШМ-імпульсів полягає у програмному задаванні частоти F_G внутрішніх імпульсів і завантаженні коду керування $C_{МК}$ до регістра **CAPx.** У процесі роботи таймера його стан циклічно змінюється від 00H до FFH. Коли $C_{МК}$ < CAPx, відповідна лінія порту **Pi.k=0**, а коли $C_{МК} \geq$ CAPx, тоді **Pi.k=1.**

3. Програмно-апаратна реалізація вихідних ШМ-сигналів

Незважаючи на те, що в багатьох МК групи MCS-51 відсутні вбудовані апаратні ШМ-генератори, сформувати ШМ-сигнал на виводі МК доволі просто. Для цього використовують таймер у конфігурації автоперезавантаження стартового числа. Розрядність таймера N_T відповідає розрядності лічильника N_C із схеми на рис. 4.25.

Стан таймера змінюється від 0 до 2^{N_T}. Значення періоду ШМ-імпульсів визначимо як $T_{ШІМ} = 2^{N_T} \cdot T_G$, де T_G – період імпульсів від внутрішнього ГТІ МК. Тоді в кожному інтервалі $T_{ШІМ}$ N1 періодів T_G відповідають активній фазі ШМ-імпульсу, а N0 періодів T_G – неактивній (нульовій) фазі. Фактично число N1 – це код керування $C_{МК}$.

Важливо, що $(N1 + N0) \cdot T_G = T_{ШІМ}$, або $N1 + N0 = 2^{N_T} - 1$. У двійковій арифметиці $2^{N_T} - 1 = FFh$, тому значення N1 та N0 є зворотними і можуть бути отримані одне з одного шляхом логічної інверсії.

Таким чином, при використанні таймера в режимі автоперезавантаження необхідно по черзі завантажувати в лічильний регістр таймера значення стартового числа для відпрацювання таймером N1 або N0 періодів T_G. Після переривання поточний вміст регістра стартового числа буде переміщатися у лічильний регістр. У процедурі обробки переривання від таймера слід змінювати вміст регістра стартового числа на інверсний, а також змінювати фазу вихідного ШМ-імпульса (0/1). Стартові числа для фаз імпульсу

$$N_{ST1} = 2^{N_T} - N1 = N0 \quad \text{та} \quad N_{ST0} = 2^{N_T} - N0 = N1.$$

В основній програмі слід розраховувати чергове значення коду керування $C_{МК}$ та записувати його в регістр стартового числа. Період роботи основної програми має бути значно більше періоду генерації ШІМ. Так, для $N_T = 8$ і $T_G = 1мкс$ частота ШІМ становить $F_{ШІМ} = 3{,}9\ кГц$. У прикладі програми таймер T0 застосовано в конфігурації автоперезавантаження для генерації імпульсів на лінії P1.7. Основна програма записує розраховане значення тривалості імпульсу в комірку **dPWM** у РПД МК.

```
dPWM  equ  40h
org 0
  jmp   main_prog    ; Перехід на основну програму
org 000Bh          ; Обробка переривання від таймера T0
  cpl  TH0         ; Формування тривалості наступної фази
  cpl  P1.7        ; Зміна фази ШМ-імпульсу
  reti             ; Повернення в основну програму
org XXXX
  main_prog:       ; Основна програма
  mov TMOD,#00000010b     ; Настроювання таймера T0
  mov IE,#10000010b       ; і дозволу переривань
  mov  TL0,#254  ; Початкове стартове число для фази 1
  mov  TH0,#1    ; Початкове стартове число для фази 0
```

```
  setb P1.7       ; Початкове значення імпульсу
  setb TR0        ; Запуск таймера T0
main_calc:        ; Обчислення за алгоритмами,
  . . .       ; що здійснюються із потрібним періодом
  . . .           ; Розрахунок коду керування для ШІМ
  clr  TR0        ; Зупин таймера T0
  mov  TL0, dPWM ; Формування нового стартового числа
  xrl  TL0, #0FFh    ; для фази «1»
  mov  TH0, dPWM ; Нове стартове число для фази «0»
  setb P1.7       ; Початкове значення імпульсу
  setb TR0        ; Запуск таймера T0
    . . .
  jmp  main_calc
```

При обробці переривання не здійснюється зупин таймера, отже, ШМ-імпульси формуються безперервно.

Зупин таймера для задавання нових стартових чисел займає всього вісім машинних циклів (3% періоду $T_{ШІМ}$, один раз за період $T_0 >> T_{ШІМ}$) і за рахунок інерційності виконавчих пристроїв не вплине на рівномірність роботи керованого обладнання.

4.4.5 Формування вихідного коду для керування ЦАП або ШІМ

Значення вихідного параметра алгоритму керування, обчислене в алгоритмі у фізичних одиницях, має бути перераховано у відповідний код керування *C* для ЦАП або ШІМ. Функція перерахунку є лінійною і виглядає аналогічно функції перетворення вхідного коду АЦП у значення параметра

$$C = Int(K_P \cdot P - B_P),$$

де *Int* – функція отримання цілої частини; P – розраховане в програмі значення вхідного параметра в фізичних одиницях, а коефіцієнти функції перерахунку визначають як

$$K_P = \frac{C_{MAX} - C_{MIN}}{P_{MAX} - P_{MIN}}; \quad B_P = C_{MIN} - K_P \cdot P_{MIN}.$$

Ці значення коефіцієнтів розраховують при розробці алгоритмів і підставляють у програму у вигляді констант.

4.4.6 Згладжування ШМ-сигналів і формування полярності

Для отримання вихідних аналогових сигналів на основі широтно-модульованих імпульсів застосовують згладжування на основі актив-

ного фільтра. Цей підхід можна використовувати як для МК із вбудованими ШІМ, так і у випадку, коли вихідний ШМ-сигнал формується програмно.

Проста схема фільтра для отримання аналогового сигналу на основі ШМ-імпульсів показана на рис. 4.26.

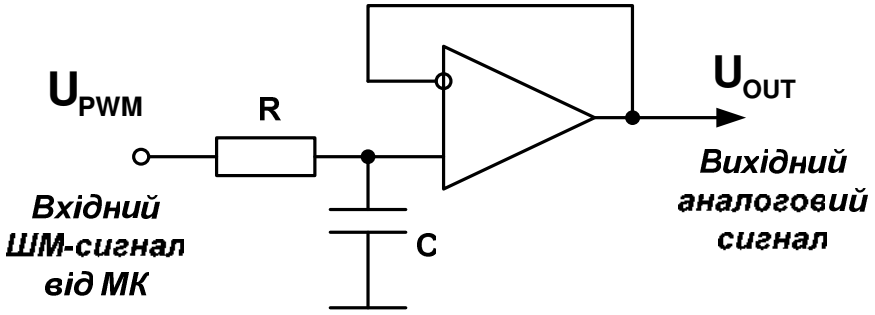

Рисунок 4.26 – Приклад схеми згладжування ШМ-імпульсів

Постійна часу фільтра $T_Ф=RC$, а частота зрізу $\omega_Ф=1/(RC)$. Значення параметрів R і C слід вибирати так, щоб частота зрізу фільтра була менше частоти ШМ-імпульсів: $F_{ШІМ} / \omega_Ф = 3…5$ (але не набагато менше, інакше фільтр буде повільно реагувати на зміну тривалості ШМ-імпульсів).

Для зміни полярності аналогового сигналу від МК (тобто від ЦАП) можна застосувати схему керованого інвертора на основі операційного підсилювача та електронного ключа, показану на рис. 4.27.

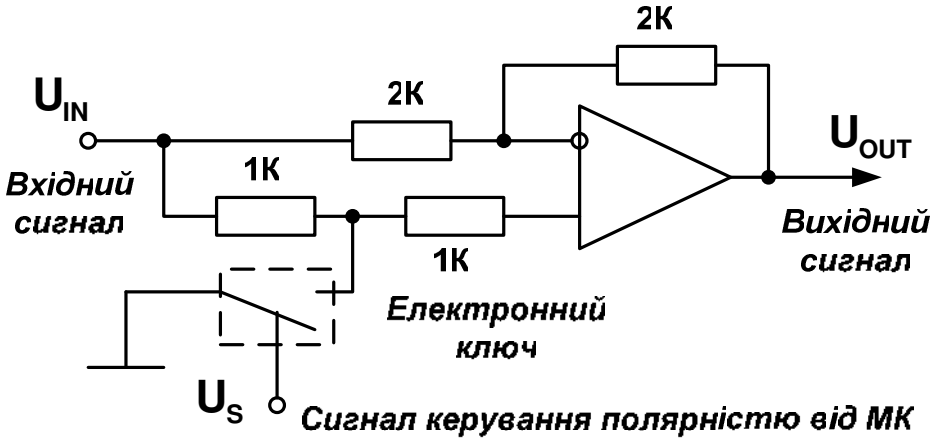

Рисунок 4.27 – Типова схема для зміни полярності аналогового сигналу

При U_S = 1 ключ замкнено, і схема працює як інвертор: $\boldsymbol{U_{OUT} = – U_{IN}}$. При U_S = 0 ключ розімкнено, схема працює як повторювач: $\boldsymbol{U_{OUT} = U_{IN}}$.

4.5 Принципи реалізації дискретного автомата на МК MCS-51

4.5.1 Постановка задачі формування дискретного автомата

Дискретні (кінцеві) автомати часто застосовують у системах керування у випадках, коли набір станів об'єкта керування та набір значень сигналів керування фіксовані, тобто утворюють кінцеву дискретну множину. Такі пристрої мають елементи пам'яті для зберігання певних значень логічних змінних навіть за відсутності вхідних сигналів, що були причиною їх встановлення. Прикладом найпростіших автоматів є тригери.

Дискретні автомати поділяють на такі:

– автомати Мура, в яких вихідні сигнали Y є функціями тільки стану автомата Q: $Q_H = f_1(Q, X),\ Y = f_2(Q);$

– автомати Мілі, в яких виходи залежать як від вхідних сигналів X, так і від стану автомата: $Q_H = f_1(Q, X),\ Y = f_2(Q, X);$

– автономні автомати, які не мають інформаційних входів X, а переключаються тільки під впливом тактових імпульсів.

Підходи до реалізації дискретних автоматів:

– апаратний, коли пристрій проектують на основі тригерів і комбінаційних схем (у даному посібнику цей підхід не розглядається);

– програмний, коли використовують універсальний процесор чи мікроконтролер і алгоритм роботи автомата реалізують програмно.

Розглянемо технологію розробки програмного автономного автомата. У цьому випадку переходи між станами і формування вихідних сигналів визначаються тільки тактовими сигналами (тобто інтервалами часу). Одним із способів опису роботи такого автомата може бути часова діаграма (як альтернатива традиційній діаграмі переходів у вигляді графа).

Наприклад, автомат має формувати набір двійкових сигналів керування A, B, C і D у такій послідовності (рис. 4.28):

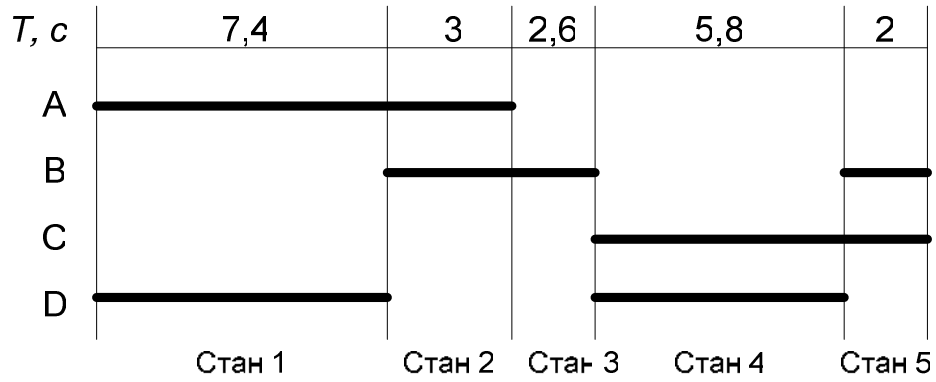

Рисунок 4.28 – Приклад часової діаграми роботи автомата

Як видно, робота автомата складається з *P = 5* станів такої тривалості: $T_1 = 7{,}4$ *с;* $T_2 = 3$ *с;* $T_3 = 2{,}6$ *с;* $T_4 = 5{,}8$ *с* і $T_5 = 2$ *с.*

Крок 1. Визначаємо тривалість дискрети часу для роботи автомата T_D (часовий інтервал перебування в одному такті) – як найбільший спільний дільник тривалості перебування в окремих станах: $T_D = 0{,}2$ *с.*

Крок 2. Розраховуємо загальну кількість тактів у циклі автомата як

$$N = \sum_{i=1}^{P} N_i = \sum_{i=1}^{P} \frac{T_i}{T_D},$$

Для нашого прикладу отримаємо

$$N = \frac{7{,}4}{0{,}2} + \frac{3}{0{,}2} + \frac{2{,}6}{0{,}2} + \frac{5{,}8}{0{,}2} + \frac{2}{0{,}2} = 37 + 15 + 13 + 29 + 10 = 104 \text{ такти.}$$

Крок 3. Визначаємо кількість і номери тактів і сукупність вихідних сигналів для кожного стану роботи автомата відповідно до часової діаграми

Номер стану	T_i, с	Кількість тактів $N_i = T_i / T_D$	Номери тактів для стану	Вихідні сигнали			
				A	**B**	**C**	**D**
1	7,4	37	1..37	1	0	0	1
2	3	15	38..52	1	1	0	0
3	2,6	13	53..65	0	1	0	0
4	5,8	29	66..94	0	0	1	1
5	2	10	95..104	0	1	1	0

Крок 4. Для формування алгоритму роботи автомата слід застосувати змінну – лічильник тактів *K*. Через кожний інтервал часу $T_D = 0{,}2$ *с* (що відповідає черговому тактовому імпульсу автомата) збільшуємо *K* на одиницю і перевіряємо, якому стану відповідає значення *K*. При переході між станами слід змінювати значення вихідного коду ABCD, що формується програмою, відповідно до таблиці (крок 3). Лічильник тактів *K* змінює своє значення від 1 до *N*, після чого знову набуває початкового значення 1.

Крок 5. Найбільш зручно сформувати алгоритм роботи автомата у вигляді блок-схеми. Дискрету (тобто такт) роботи автомата можна реалізувати на основі програмної затримки або за допомогою таймера. У другому варіанті весь алгоритм роботи автомата виконується в процедурі обробки переривання від таймера. Блок-схема алгоритму автомата показана на рис. 4.29.

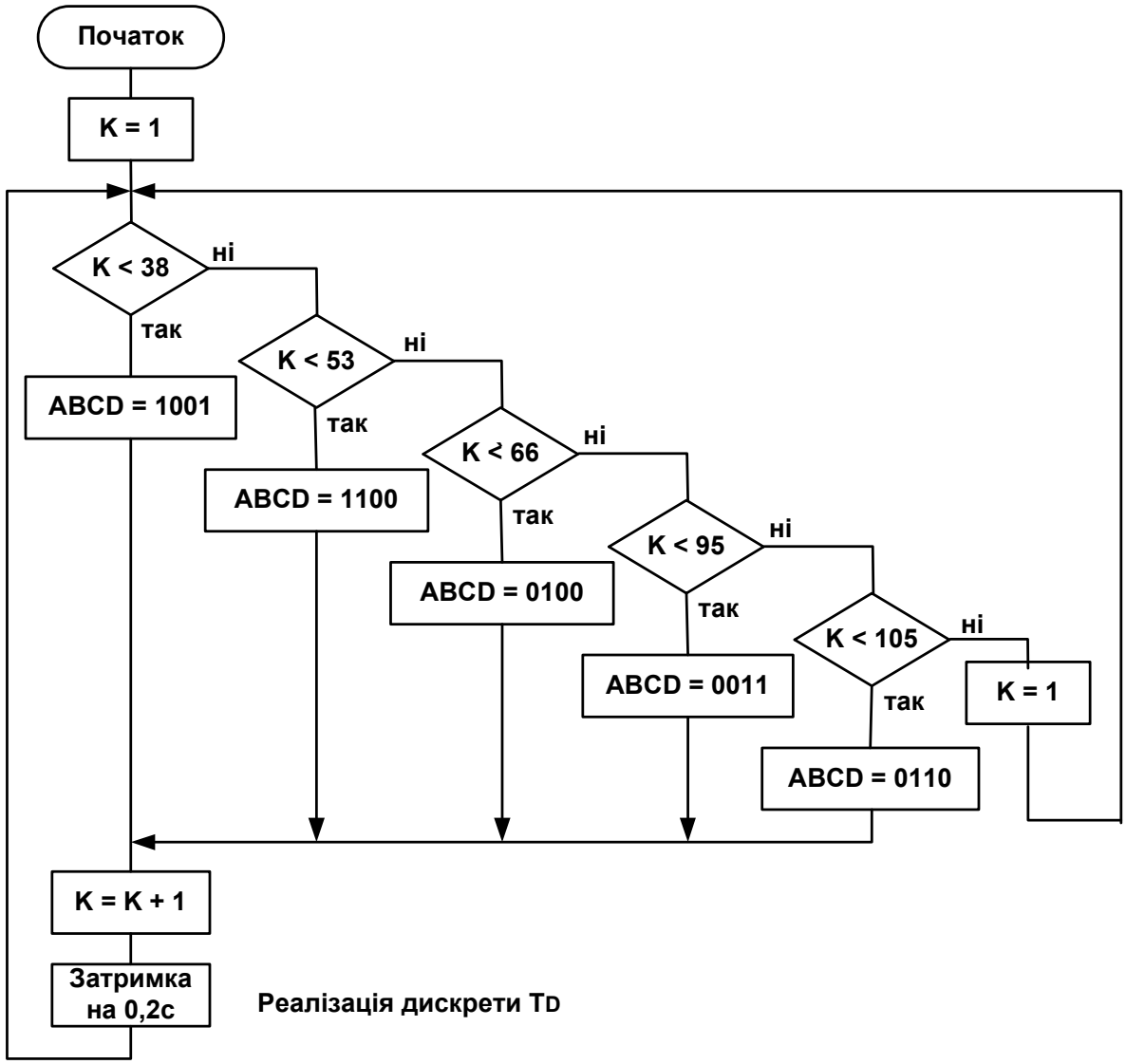

Рисунок 4.29 – Приклад блок-схеми алгоритму дискретного автомата

Зазначимо, що в алгоритмах такого типу обов'язково має бути холостий такт ("105" у прикладі) чи перевірка досягнення кінцевого такту *N*. У холостому такті не реалізується інтервал T_D, отже, на перебування в ньому практично не витрачається час, але здійснюється перехід до стану і такту 1.

Ще одна особливість алгоритму автомата – у нього немає традиційного блока завершення, оскільки припинити роботу автомата може тільки зовнішній вплив (вимкнення контролера, переривання від зовнішнього сигналу тощо), але не програмні дії.

Програмна реалізація блок-схеми (рис. 4.29) може бути здійснена принаймні двома способами: безпосереднє програмування блок-схеми у вигляді послідовності перевірок значення *K* і на основі табличного подання сукупності вихідних сигналів для кожного стану. Роз-

глянемо детальніше другий спосіб, як більш швидкісний у виконанні процесором.

4.5.2 Програмна реалізація дискретного автомата на основі таблиці станів

Відповідно до цього підходу у пам'яті програм формують таблицю у вигляді одновимірного масиву, що містить набори вихідних сигналів для всіх станів і тактів роботи автомата. Доступ до потрібного елемента масиву здійснюється через індекс на основі значення лічильника тактів *K*.

Найефективніша реалізація для МК MCS-51 базується на використанні відносної адресації в таблиці із застосуванням команди **MOVC A,@A+PC**. Створимо спеціальну підпрограму F1 для отримання значення елемента таблиці відповідно до його номера. Нехай номер елемента передається у підпрограму через регістр-акумулятор і значення елемента також повертається через акумулятор. Таблиця значень вихідних сигналів автомата має бути розташована у пам'яті програм безпосередньо після команди RET, тобто адреса таблиці може бути визначена відносно команди читання MOVC всередині підпрограми (абсолютну адресу таблиці знати не потрібно). Адреса комірки пам'яті програм, із якої читається потрібний елемент, буде обчислена як сума вмісту регістра A і поточного значення регістра PC. Програмна реалізація може бути такою:

```
dseg   ; Сегмент даних
org 30h  ; Визначення змінної – лічильника тактів
   K: ds 1
cseg   ; Сегмент програми
start:   mov K,#1 ; Початкові стан і такт автомата
loop:    mov a,K  ; Передача індексу в підпрограму
     call F1   ; Отримання вихідного коду автомата
     mov P0,a ; Видача коду через порт на об'єкт
     inc K     ; Перехід до наступного такту
     mov a,K   ; Перевірка досягнення кінцевого такту
     cjne a,#105,mTd ; Якщо K≠105, то на реалізацію Td,
     jmp start ; інакше – перехід до початкового стану
mTd: call Delay02s ; Реалізація дискрети Td
     jmp loop  ; Цикл роботи автомата

; підпрограма отримання вихідного коду автомата
F1:  movc a,@a+pc ; читання елемента таблиці
     ret          ; повернення з підпрограми
```

```
table: db 09h,<...>,09h ; 37 однакових байт для стану 1
       db 0Ch,<...>,0Ch ; 15 однакових байт для стану 2
       db 04h,<...>,04h ; 13 однакових байт для стану 3
       db 03h,<...>,03h ; 29 однакових байт для стану 4
       db 06h,<...>,06h ; 10 однакових байт для стану 5
```

У даному прикладі директиву **db <байт>** застосовано для визначення вмісту послідовності байт.

При читанні команди **MOVC A,@A+PC** із пам'яті перед її виконанням у регістрі PC формується адреса наступної команди, тобто однобайтової команди **RET**. Адреса початку таблиці на одиницю більше адреси команди **RET**. Елементи таблиці індексуються своїм зміщенням від її початку, тобто перший елемент має індекс 0. Оскільки номер такту на одиницю більше індексу потрібного елемента, то це як раз компенсує наявність однобайтової команди **RET**. Час доступу до даних не залежить від розміру таблиці й поточного індексу. Підпрограма F1 не використовує ніяких додаткових регістрів.

4.5.3 Реалізація дискретного автомата із вхідними сигналами

У випадку, якщо робота дискретного автомата залежить не тільки від тактів часу (дискрет T_D), але і від вхідних сигналів, на початку кожного такту у верхній точці циклу слід аналізувати сукупність вхідних сигналів і примусово змінювати значення лічильника тактів *K* таким чином, щоб перевести автомат у потрібний стан.

Нехай автомат із попереднього прикладу має два вхідні сигнали – S1 і S2. За наявності S1=1 із стану C1 чи C2 треба переходити у стан C3, а за наявності S2=1 із стану C4 треба переходити до стану C5, не чекаючи завершення поточного стану. Послідовність зміни станів автомата можна описати графом станів на рис. 4.30.

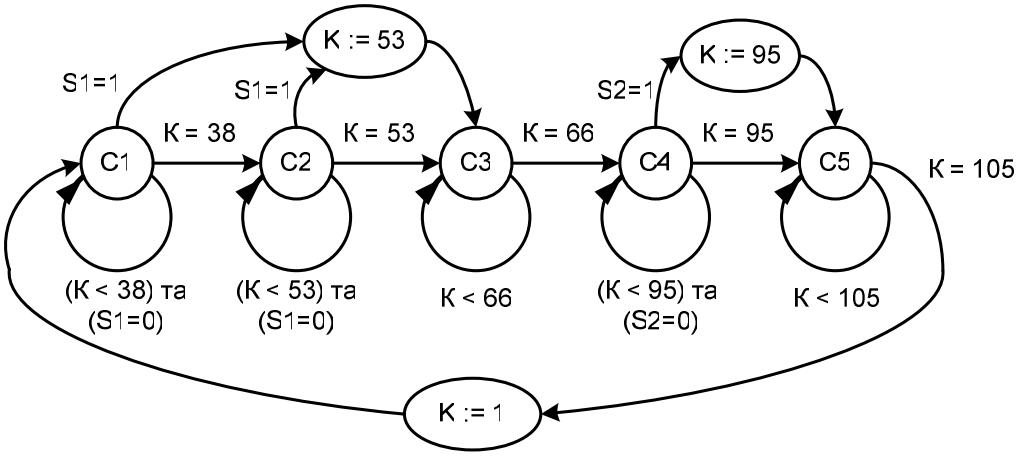

Рисунок 4.30 – Діаграма переходів автомата із вхідними сигналами

У кожному із станів С1…С5 на кожному такті роботи здійснюється інкремент лічильника тактів *K* і видача відповідного коду на об'єкт керування. Крім того, перевіряються вхідні сигнали S1 і S2. Додатковий фрагмент блок-схеми автомата показано на рис. 4.31.

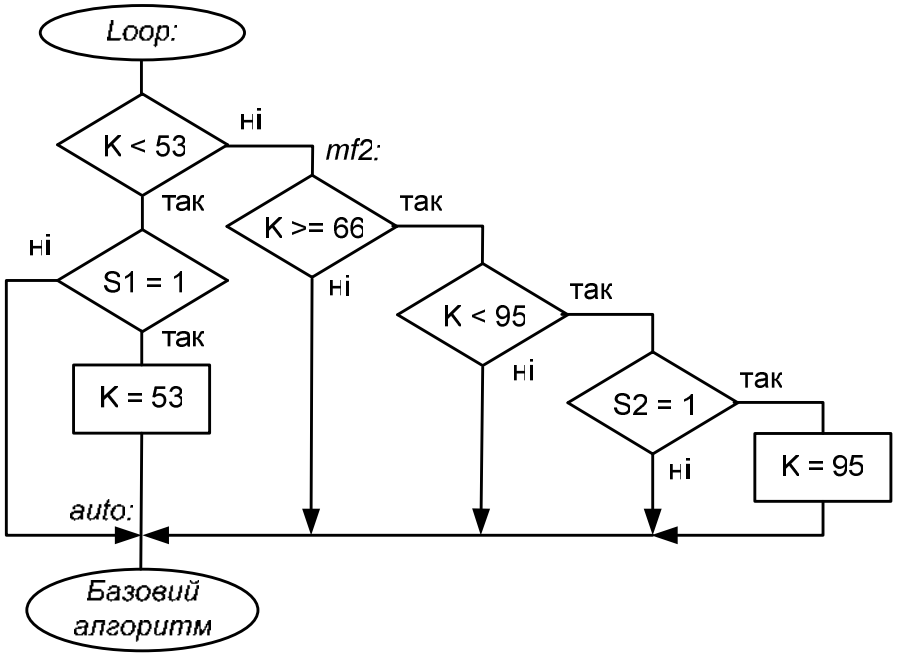

Рисунок 4.31 – Частина алгоритму для перевірки вхідних сигналів

```
S1   equ  P1.0     ; Визначення ліній порту
S2   equ  P1.1     ; для вхідних сигналів
. . .          ; Визначення змінної K – лічильника тактів
cseg   ; Сегмент програми
start:   mov K,#1 ; Початковий стан і такт автомата
loop:    clr c
     mov a,K       ; Перевірка знаходження у С1 і С2
     subb a,#53    ; K < 53 ?
     jnc mf2       ; При K ≥ 53 перехід до перевірки С4
     jnb S1,auto ; Перевірка сигналу S1
     mov K,#53     ; При S1=1 – перехід на С3 (до 53)
     jmp auto      ; Перехід до базового алгоритму
mf2: clr c
     mov a,K       ; Перевірка знаходження у стані С4
     subb a,#66    ; K ≥ 66 ?
     jc  auto      ; "K" ще у С3 – на базовий алгоритм
     subb a,#95    ; K < 95 ?
     jnc auto      ; При K ≥ 95 – до базового алгоритму
     jnb S2,auto ; Перевірка сигналу S2
     mov K,#95     ; При S2=1 – перехід на С5 (до 95)
```

```
auto:    mov a,K ; Передача індексу в підпрограму
    . . .        ; Базовий алгоритм, наведений раніше
    jmp loop     ; Цикл роботи автомата
```

4.6 Зв'язок МК з ПЕОМ через послідовний порт

При застосуванні мікроконтролерних регуляторів часто виникає потреба у відображенні даних оператору на ПЕОМ. Оскільки мікроконтролерний блок, як правило, знаходиться на значній відстані від ПЕОМ, може бути реалізований тільки послідовний тип зв'язку. З боку мікроконтролера зазвичай використовують послідовний порт UART, а на стороні ПЕОМ – COM-порт. На фізичному рівні COM-порт працює у стандарті RS-232 (лог. "0" – "+12В", лог. "1" – "-12В"), що не збігається з рівнями сигналів ТТЛ.

Сучасним рішенням для узгодження рівнів сигналів є застосування спеціальної мікросхеми MAX232 (рис. 4.32). Ця мікросхема містить два однотипних канали перетворення.

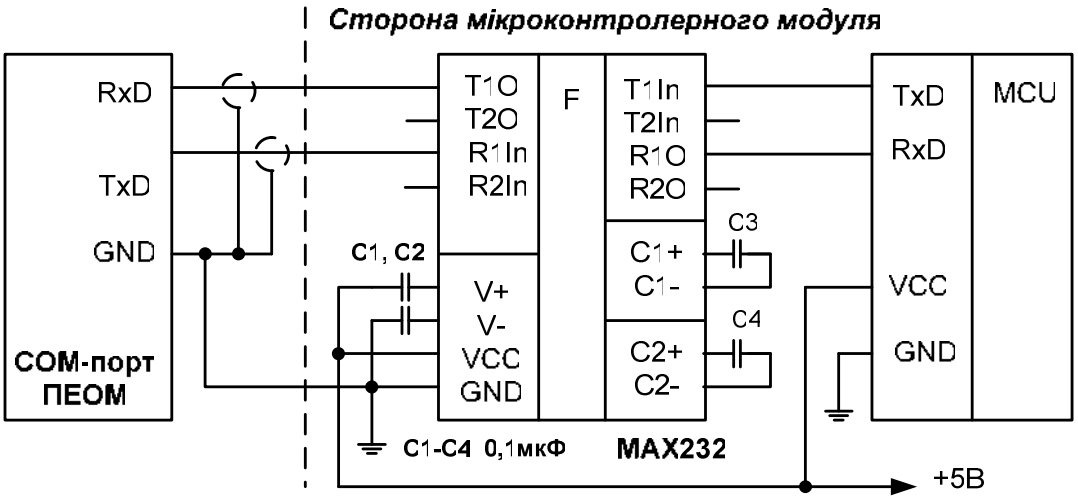

Рисунок 4.32 – Схема використання перетворювача ТТЛ – RS-232

Недолік інтерфейсу RS-232 – взаємодія тільки двох пристроїв. Тому для побудови мережі контролерів застосовують інтерфейс RS-485, який визначається міжнародним стандартом EIA RS-422/485 (регламентує тільки електричні параметри) – рис. 4.33.

Фізичний канал передачі в стандарті RS-485 – це симетрична лінія з двох провідників (вита пара). На кінцях лінії встановлюють *термінатори* (Т) – резистори 120 Ом. Обидва провідники (А і В) мають *ненульовий та незбіжний* потенціал. Формування та ідентифікація логічних рівнів здійснюється за правилом: логічний "0" при $U_A < U_B$, логічна "1" при $U_A > U_B$. Це суттєво знижує вплив зовнішніх електромагнітних полів.

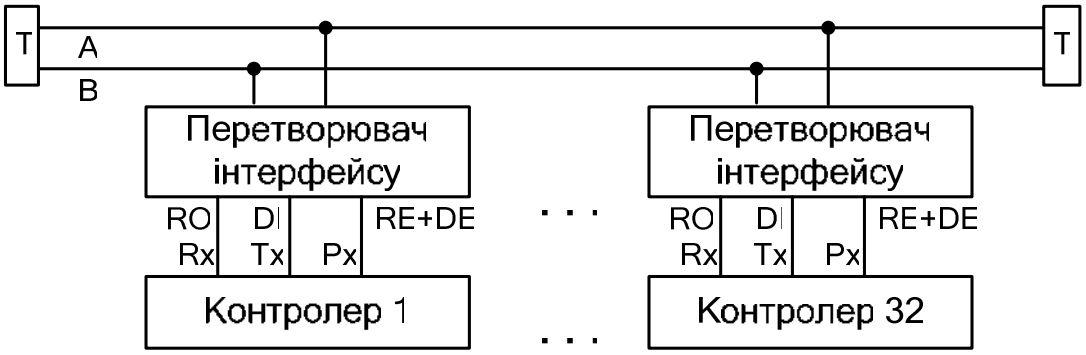

Рисунок 4.33 – Принцип побудови мережі МК стандарту RS-485

Для формування сигналів у стандарті RS-485 застосовують перетворювачі інтерфейсу (ТТЛ – RS-485) – спеціалізовані мікросхеми, наприклад ADM485 фірми Analog Devices Inc. При під'єднанні цієї мікросхеми до мікроконтролера застосовують такий розподіл ліній:

- на стороні контролера – вхід Rx послідовного порту до виводу RO перетворювача, вихід Tx послідовного порту до виводу DI перетворювача, одну з ліній довільного порту МК – одночасно до ліній RE і DE перетворювача; сигнал керування формують за принципом: "0" – робота на прийом, "1" - на передачу;
- на стороні мережі – виводи A та B застосовують як контакти для під'єднання провідників, що формують мережу.

На один сегмент мережі можна встановлювати до 32 пристроїв. Протокол обміну даними формують відповідно до потреб системи.

4.7 Критерії вибору типу аналого-цифрового інтерфейсу

У попередніх розділах було детально розглянуто різновиди аналого-цифрових інтерфейсів, що застосовують для введення або виведення аналогових сигналів у МК-системах. Для обґрунтованого вибору інтерфейсу особливості різних варіантів зведені у табл. 4.3.

Таблиця 4.3 – Переваги та недоліки аналого-цифрових інтерфейсів

Тип інтерфейсу	Переваги, недоліки, особливості застосування
АЦП із паралельним вихідним кодом	*Переваги*: швидке перетворення сигналів (одиниці-десятки мікросекунд), вихідні дані у цифровому вигляді, може бути двополярний вхідний сигнал. *Недоліки*: датчик та АЦП мають знаходитися близько від плати МК, відносно висока ціна АЦП. *Особливості*: точність перетворення залежить від розрядності АЦП: $\Delta=2^{-N}$, сигнали керування від МК

Продовження табл. 4.3

Тип інтерфейсу	Переваги, недоліки, особливості застосування
АЦП із послідовним вихідним кодом	*Переваги*: такі ж, як у АЦП із паралельним виходом; крім того, датчик та АЦП можуть знаходитися на значній відстані від МК, висока розрядність АЦП. *Недоліки*: відносно висока ціна АЦП, додаткові сигнали керування та синхронізації від МК, складний протокол прийому даних, потрібен додатковий час на прийом даних у МК. *Особливості*: точність перетворення залежить від розрядності АЦП: $\Delta=2^{-N}$; де N – від 16 до 24 бітів
Вхідний перетворювач ШІМ	*Переваги*: проста схемотехнічна реалізація; перетворення сигналів від датчиків із потенціальним виходом чи джерел напруги (на відміну від ЧІМ); можливість віддалення ШІМ від МК на значну відстань; при застосуванні «ланцюга струму» – гальванічне розгалуження ШІМ і МК. *Недоліки*: час вимірювання залежить від значення вхідного параметра (змінна ширина імпульсів); можливість «насичення» перетворювача при зависокому чи занизькому вхідному сигналі; максимальна та мінімальна ширина імпульсів обмежена параметрами таймера МК. *Особливості*: необхідне застосування таймера МК для перетворення імпульсів у цифрову форму; точність вимірювання залежить від частоти тактових імпульсів, що надходять на таймер
Вхідний перетворювач ЧІМ	*Переваги*: дуже проста схемотехнічна реалізація; можливість віддалення ШІМ від МК на значну відстань; при застосуванні «ланцюга струму» – гальванічне розгалуження ШІМ і МК. *Недоліки*: час вимірювання залежить від значення вхідного параметра (змінний період імпульсів); максимальна та мінімальна частота імпульсів обмежена параметрами таймера МК; при вимірюванні частоти (а не періоду) необхідні додаткові розрахунки через нелінійну залежність від параметра

Закінчення табл. 4.3

Тип інтерфейсу	Переваги, недоліки, особливості застосування
	Особливості: необхідне застосування таймера МК для перетворення імпульсів у цифрову форму; точність вимірювання залежить від частоти тактових імпульсів, що надходять на таймер
ЦАП із паралельним вхідним кодом	*Переваги*: пряме перетворення сигналів практично без затримки, вхідні дані у цифровому вигляді; як правило, невисока ціна. *Недоліки*: ЦАП має знаходитися близько від плати мікроконтролера, розрядність – до 10-12 бітів. *Особливості*: точність перетворення залежить від розрядності ЦАП: $\Delta=2^{-N}$, складна схема формування декількох вихідних сигналів з одного порту МК
ЦАП із послідовним вхідним кодом	*Переваги*: більша розрядність (16-24 біти) порівняно із паралельними ЦАП, пристрій може знаходитись на відстані від мікроконтролера. *Недоліки*: відносно висока ціна ЦАП, додаткові сигнали керування та синхронізації від МК, складний протокол передачі даних, потрібен додатковий час на передачу даних у ЦАП. *Особливості*: точність перетворення залежить від розрядності ЦАП: $\Delta=2^{-N}$, де N – від 16 до 24 бітів
Вихідний перетворювач ШІМ	*Переваги*: відносно проста схемотехнічна реалізація; перетворення сигналів практично без затримки; можливість керування силовим обладнанням через підсилювачі у ключовому режимі; один таймер можна застосувати для декількох перетворювачів; наявність апаратних блоків ШІМ у складі МК. *Недоліки*: якщо МК не містить ШІМ, перетворювач має знаходитися близько від плати МК. *Особливості*: точність формування ШМ-імпульсів залежить від розрядності таймера та компаратора (рис. 4.25); період ШІМ має бути у 100-1000 разів менший за період дискретизації системи керування

Бібліографічний список

1. Бродин В.Б. Системы на микроконтроллерах и БИС программируемой логики / В.Б. Бродин, А.В. Калинин. – М.: ЭКОМ, 2002. – 400 с.
2. Бродин В.Б. Микроконтроллеры. Архитектура, программирование, интерфейс / В.Б. Бродин, И.И. Шагурин. – М.: ЭКОМ, 1999. – 400 с.
3. Микроконтроллерные системы: структуры и практическое применение: учеб. пособие: в 2 ч. / В.Г. Джулгаков, В.В. Нарожный, К.И. Руденко, А.Н. Таран. – Х.: Нац. аэрокосм. ун-т “Харьк. авиац. ин-т”, 2003. – Ч.1. – 126 с.
4. Микроконтроллерные системы: структуры и практическое применение: учеб. пособие: в 2 ч. / В.Г. Джулгаков, В.В. Нарожный, К.И. Руденко, А.Н. Таран. – Х.: Нац. аэрокосм. ун-т “Харьк. авиац. ин-т”, 2005. – Ч.2. – 90 с.
5. Сайт програмної системи Visual MCStudio: http://mcstudio.h15.ru
6. Угрюмов Е.П. Цифровая схемотехника / Е.П. Угрюмов. – СПб.: БХВ-Петербург, 2001. – 528 с.
7. Гутников В.С. Интегральная электроника в измерительных устройствах / В.С. Гутников. – Л.: Энергоатомиздат, 1988. – 304 с.
8. Гук М. Интерфейсы ПК: справ. / М. Гук. – СПб.: ЗАО «Издательство «Питер», 1999. – 416 с.
9. Густав Оллсон. Цифровые системы автоматизации и управления / Оллсон Густав, Пиани Джангуидо. – СПб.: Невский Диалект, 2001. – 557 с.
10. Евстифеев А.В. Микроконтроллеры AVR семейств Tiny и Mega фирмы “ATMEL” / А.В. Евстифеев. – М.: Изд. дом “Додэка-XXI”, 2004. – 560 с.
11. Евстифеев А.В. Микроконтроллеры AVR семейства Classic фирмы “ATMEL” / А.В. Евстифеев. – М.: Изд. дом “Додэка-XXI”, 2002. – 228 с.
12. Кузьминов А.Ю. Интерфейс RS232. Связь между компьютером и микроконтроллером / А.Ю. Кузьминов. – М.: Радио и связь, 2004. – 168 с.

Зміст

Джулгаков Віталій Георгійович
Руденко Кирило Ігорович

ПРОЕКТУВАННЯ ЦИФРОВИХ КОНТРОЛЕРІВ

Редактор Т.Г. Кардаш

Зв. план, 2008
Підписано до друку 26.03.2008
Формат 60x84 1/16. Папір офс. №2. Офс. друк
Ум. друк. арк. 5,5. Обл.- вид. арк. 6,25. Наклад 100 прим.
Замовлення 164. Ціна вільна

Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»
61070, Харків-70, вул. Чкалова,17
http://www.khai.edu
Видавничий центр «ХАІ»
61070, Харків-70, вул. Чкалова,17
izdat@khai.edu