Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

А. О. Новацький

ЕЛЕКТРОНІКА ТА МІКРОПРОЦЕСОРНА ТЕХНІКА

Частина 2

Мікропроцесорні системи

Підручник

*Затверджено Вченою радою КПІ ім. Ігоря Сікорського
як підручник для студентів, які навчаються за освітньою програмою
«Інтегровані інформаційні системи» за спеціальністю
126 «Інформаційні системи та технології»*

Київ
КПІ ім. Ігоря Сікорського
2023

.

УДК 004.3(075.8)
Н72

Рецензенти:
А. Ю. Дорошенко, д-р фіз.-мат. наук, проф.,
Інститут програмних систем НАН України

О. А. Чемерис, д-р техн. наук, ст. наук. співроб.,
Інститут проблем моделювання в енергетиці ім. Г. Є. Пухова НАН України

Відповідальний редактор
В. П. Полторак, канд. техн. наук, доц.,
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Гриф надано Вченою радою КПІ ім.Ігоря Сікорського (протокол № 4 від 3 квітня 2023 р.)

Автор
Новацький Анатолій Олександрович, кандидат технічних наук, доцент

Н72 Електроніка та мікропроцесорна техніка : підручник. У 2 ч. Ч. 2. Мікропроцесорні системи [Електронний ресурс] / А. О. Новацький. – Електронні текстові дані (1 файл: 20,2 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2023. – 489с.

Стисло викладено основи систем числення, кодування та двійкової арифметики. Розглянуто характеристику мікропроцесорної системи, її структуру, програмну модель та характеристику команд типового мікропроцесора та мікроконтролера. Наведено матеріал про мікроконтролерні мережі та основні периферійні модулі мікроконтролерів: таймери/лічильники; широтно-імпульсний модулятор; паралельні порти; підсистема переривань; універсальний асинхронний послідовний інтерфейс; пам’ять; аналого-цифровий та цифро-аналоговий перетворювачі; інтерфейс RS-232. Розглянуто моделювання у пакеті «PROTEUS» модулів: універсального асинхронного інтерфейсу, широтно-імпульсного модулятора, годинника реального часу, аналого-цифрового перетворювача, цифрового вольтметра, TWI, SPI та 1-WIRE.

Для студентів освітньої програми «Інтегровані інформаційні системи» за спеціальністю 126 «Інформаційні системи та технології», які вивчають проектування та використання мікропроцесорних систем. Вона може бути також корисною студентам відповідних спеціальностей під час вивчення дисциплін, які пов’язані з проектуванням та використанням мікропроцесорних систем, а також під час виконання курсових проектів, бакалаврських робіт та магістерських дисертацій, в яких використовуються мікропроцесорні пристрої.

УДК004.3(075.8)

© А. О. Новацький, 2023
© КПІ ім. Ігоря Сікорського, 2023

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

АЦП – аналого-цифровий перетворювач
БШД – буфер шини даних
ВІС – велика інтегральна схема
ВПД – внутрішня пам’ять даних
ВПП – внутрішня пам’ять програм
ГТІ – генератор тактових імпульсів
ДК – двійковий код
ДШ – дешифратор
ЕОМ – електронна обчислювальна машина
ЗШД – зовнішня шина даних
ІНЛ – інтегральна нелінійність
КМОН – комплементарний метал-оксид-напівпровідник
МІНТ – мережевий інтерфейс
МК – мікроконтролер
МКС – мікроконтролерна система
МП – мікропроцесор
МПС – мікропроцесорна система
МПСК – мікропроцесорна система керування
ОЗП – оперативний запам’ятовувальний пристрій
ПДП – прямий доступ до пам’яті
ПЗП – постійний запам’ятовувальний пристрій
ПФР – пристрій формування рівнів
СГ – системний генератор
ТК – транзисторний ключ
УАПП – універсальний асинхронний програмований приймач/передавач
УСАПП – універсальний синхронно/асинхронний приймач/передавач
ЦАП – цифро-аналоговий перетворювач
ШІМ – широтно-імпульсна модуляція

УМОВНІ ПОЗНАЧЕННЯ

АВЕ – аналоговий виконавчий елемент
АГР – агрегат
АК – аналоговий компаратор
АЛП – арифметико-логічний пристрій
АМПС – аналоговий мультиплексор
АС – адресний селектор
БР – буферний регістр
БРА – буферний регістр адреси
БСШ – блок сполучення з шиною
ВМ – виконавчий механізм
ВАК – власне аналоговий компаратор
ВБ – виконавчий блок
ВІ – вузол індикації
ВШД – внутрішня шина даних
Д – датчик
ДВЕ – дискретні виконавчі елементи
ДЖ – джерело живлення
ДОН – джерело опорної напруги
ДПД – дозвіл передачі даних
$\overline{\text{ДПРМ}}$ – дозвіл прийому даних
ДШ – дешифратор
ЗВПР – зовнішній пристрій
ЗП – запам’ятовувальний пристрій
КЗ – канал зв’язку
КЕОМ – керуюча ЕОМ
КОП – код операції
КП – комірка пам’яті
КПДП – контролер прямого доступу до пам’яті
КПЕР – контролер переривань
ЛЗ – лінія зв’язку
ЛК – лічильник команд
МБ – молодший байт
МЗР – молодший значущий розряд
МІНТ – мережевий інтерфейс
МПЛ – мультиплексор
НП – нормувальний перетворювач
ОК – об’єкт керування

ОП – операційний підсилювач
ПВВ – пристрій введення/виведення
ПВЗ – пристрій вибірки-зберігання
ПВІ – пристрій відображення інформації
ПД – передавач
ПЕР – переривання
ПК – персональний комп’ютер
ПП – пам’ять програм
ППД – приймач/передавач
ППІ – паралельний програмований інтерфейс
ППР – пристрій перетворення рівнів
ППС – пристрій перетворення сигналів
ПРМ – приймач
ПТ – програмований таймер
РА – регістр адреси
РВВ – регістр введення/виведення
РГ – регістр
РГПД – регістр-покажчик даних
РЗП – регістр загального призначення
РК – регістр команд
РКП – регістр керування потужністю
РКПП – регістр керування приймачем-передавачем
РКС – регістр керувального слова
РКСТ – регістр керування-статусу
РМП – регістр масок переривань
РП – регістр пріоритетів
РПД – резидентна пам’ять даних
РПН – регістр послідовного наближення
РПП – резидентна пам’ять програм
РПС – регістр-покажчик стека
РШД – резидентна шина даних
СБ – старший байт
СД – світлодіод
СЗР – старший значущий розряд
СК – скидання
СМА – суматор адреси
СОЗП – статичний оперативний запам’ятовувальний пристрій
СПД – статична пам’ять даних
СУР – схема узгодження рівнів

СФР – схема формування рівнів
СШ – системна шина
США – системна шина адреси
СШД – системна шина даних
СШК – системна шина керування
ТЗЧ – точка зчитування
ТТЛ – транзисторно-транзсторна логіка
УП – узгоджуючий пристрій
ФНЧ – фільтр нижніх частот
ЦД – цифровий давач
ЦПП – центральний процесорний пристрій
ША – шина адреси
ШД – шина даних
ШК – шина керування
ШМК – шлюзовий мікроконтролер
ШФ – шинний формувач
ACK – підтвердження
ASCII – американський стандартний код для обміну інформацією
BCD – двійково-десяткове кодування
BHE – передавання старшого байта
BT – час передачі одного біта (bit time)
CAN – мережа контролерів (Controller Area Network)
CPHA – фаза тактового сигналу
CPOL – полярність тактового сигналу
DMA – прямий доступ до пам’яті (direct memory access)
EEPROM – енергонезалежна пам’ять, що може бути електрично стерта та перепрограмована
ETIMSK – розширений регістр маски переривань від таймерів/лічильників
FILO – першим зайшов, останнім вийшов (First In, Last Out)
FIFO – першим зайшов, першим вийшов (First In, First Out)
GTCCR – загальний регістр керування таймером/лічильником
IEEE – інститут інженерів з електротехніки та електроніки
IE – регістр дозволу переривань
I^2C – двонаправлена двопровідна шина
IP – регістр пріоритетів переривань
IR – регістр команд
$\overline{\text{IORD}}$ – читання пристрою введення/виведення

$\overline{\text{IOWR}}$ – запис у пристрій введення/виведення

JTAG – інтерфейс, призначений для підключення складних цифрових мікросхем або пристроїв рівня друкованої плати до стандартної апаратури тестування і налагодження

LSB – молодший значущий розряд (Least Significant Bit)

MSB – старший значущий розряд (Most Significant Bit)

$\overline{\text{MEMR}}$ – читання пам’яті

$\overline{\text{MEMW}}$ – запис у пам’ять

MIPS – мільйон інструкцій на секунду (Million Instructions Per Second)

MISO – сигнал входу для ведучого, виходу для веденого

MOb – об’єкт повідомлення (Message Objects)

MOSI – сигнал виходу для ведучого, входу для веденого

MSPI – режим ведучого SPI (Master SPI Mode)

NACK – не підтвердження

NRZ – код без повернення до нуля (Non Return to Zero)

NRWW – немає читання під час запису (No Read-While-Write)

OCI – відкрита система взаємозв’язку (Open System Interconnection)

OCR – регістр порівняння

PAROUT – значення вихідного контрольованого параметра

PC – лічильник команд (Program counter)

RAM – пам’ять з довільним доступом (Random Access Memory)

ROM – пам’ять тільки для читання (Read-Only Memory)

RF – регістр прапорців

RESET – системне скидання

RWW – читання під час запису (Read-While-Write)

SCK – лінія тактового сигналу

SCL – послідовна лінія тактування

SDA – послідовна лінія даних

SFIOR – регістр спеціальних функцій введення/виведення

SP – покажчик стека

SPI – послідовний периферійний інтерфейс

SPM – зберігання пам’яті програм (Store Program Memory)

SPMCR – керуючий регістр зберігання пам’яті програм (Store Program Memory Control Register)

SRAM – статичний оперативний запам’ятовувальний пристрій

SS – стековий сегмент

TAP – порт тестового доступу

TCCR – регістр керування таймером/лічильником
TCON – регістр керування таймерами
TTCAN – CAN-мережа, керована часом (Time-Triggered CAN)
TIFR – регістр прапорців переривань від таймерів/лічильників
TIMSK – регістр маски переривань від таймерів/лічильників
TMOD – регістр режимів таймерів
TWI – двопровідний інтерфейс
UDR – буферні регістри приймача і передавача
USART – універсальний синхронний/асинхронний приймач/передавач
USI – універсальний послідовний інтерфейс (Universal Serial Interface)
WDTCR – регістр керування вартовим таймером

ВСТУП

Підручник орієнтовано на підготовку фахівців зі спеціальності «Інформаційні системи та технології», для успішного опановування якою необхідно засвоїти принципи побудови та використання мікропроцесорних пристроїв, на основі яких розробляються та функціонують мікропроцесорні та обчислювальні системи. Підручник охоплює теоретичний матеріал та практичні завдання, які необхідні для вивчення базової частини дисципліни «Електроніка та мікропроцесорна техніка». Підручник складається із десяти розділів, в яких розглядаються питання, що пов'язані з програмуванням та вивченням архітектури основних апаратних засобів мікропроцесорних та мікроконтролерних систем (МПС/МКС). Також у підручнику розглянуто такі питання: основні поняття та особливості мікропроцесорної техніки; структурна та функціональна схеми МПС; структури типового мікропроцесора (МП) та мікроконтролера (МК); програмні моделі МП та МК; мова Асемблера та її використання у МПС; характеристика команд МП та МК; способи адресації операндів і команди типового МП та МК; організація підсистеми переривань; формування інтервалів часу та підрахунок зовнішніх подій у МПС; організація пам'яті та модуля введення/виведення; інтерфейси: I^2C, SPI, CAN, 1-WIRE та RS-485; зв'язок МП та МК з аналоговим об'єктом керування; зв'язок МП/МК з модемом і т. ін.

Матеріал викладено так, що кожний наступний розділ є логічним продовженням попереднього. В кінці кожного розділу наведено контрольні запитання та завдання для самоконтролю.

Тематика підручника відповідає робочій програмі дисципліни «Електроніка та мікропроцесорна техніка», яка є важливою дисципліною у навчальному плані підготовки бакалаврів і магістрів у Національному технічному університеті України «Київський політехнічний інститут імені Ігоря Сікорського» за освітньою програмою «Інтегровані інформаційні системи» зі спеціальності 126 «Інформаційні системи та технології».

Студент має знати: принципи побудови та функціонування мікропроцесорних та мікроконтролерних пристроїв та систем, принципи вибору методів аналізу і розрахунку цих пристроїв та систем із заданими характеристиками, а також вміти: розраховувати, програмувати та моделювати мікропроцесорні та мікроконтролерні пристрої та системи; розробляти структурні, функціональні та принципові схеми та схеми алгоритмів роботи.

Підручник написано на основі досвіду викладання відповідних дисциплін згідно з програмами підготовки бакалаврів і магістрів спеціальності 126 «Інформаційні системи та технології» на кафедрі інформаційних систем та технологій у Національному технічному університеті України «Київський політехнічний інститут імені Ігоря Сікорського».

Курс забезпечується такими дисциплінами: математика, фізика, програмування, основи системної інженерії, комп'ютерні мережі, теорія інформації та кодування, комп'ютерна електроніка.

Зі свого боку, курс забезпечує засвоєння студентами більшості наступних дисциплін спеціальності, оскільки в ньому розглянуто архітектурні особливості сучасних мікропроцесорних та мікроконтролерних пристроїв та систем, які є основою сучасних інтегрованих інформаційних систем.

Підручник може бути корисним студентам відповідних спеціальностей під час вивчення дисциплін, які пов'язані із проектуванням та використанням МПС, а також під час виконання курсових проектів, курсових робіт, бакалаврських та магістерських робіт, в яких використовуються мікропроцесорні пристрої.

1. ХАРАКТЕРИСТИКА МІКРОПРОЦЕСОРНИХ СИСТЕМ

1.1. Основні поняття мікропроцесорної техніки

1.1.1. Мікропроцесор

Мікропроцесор – це пристрій, що обробляє інформацію згідно з програмою, яка подається на його входи у вигляді команд, і реалізований в одній великій інтегральній схемі (ВІС). У МПС МП не може функціонувати без інших інтегральних схем, які виконують функції пам'яті, введення/виведення, формування часових інтервалів, аналого-цифрового та цифро-аналогового перетворення і т. ін.

Мікропроцесор реалізує функції, які зазвичай має виконувати центральний процесор електронної обчислювальної машини (ЕОМ), та здійснює арифметичні і логічні операції, аналізує і приймає рішення, що змінюють процес обчислення, керує процесом обміну інформацією і т. ін.

1.1.2. Мікро-ЕОМ

Мікро-ЕОМ – універсальний блок обробки даних і формування керувальних сигналів, призначений для вбудовування в різні спеціалізовані системи контролю і керування для розширення їх функціональних можливостей. Мікро-ЕОМ містить: МП; оперативний запам'ятовувальний пристрій (ОЗП); постійний запам'ятовувальний пристрій; пристрій введення/виведення і т. ін. Нині цей термін фактично не використовується. Його замінено на термін «мікроконтролер».

1.1.3. Однокристальна мікро-ЕОМ

Однокристальна мікро-ЕОМ – це мікро-ЕОМ, яку побудовано на одній мікросхемі (одному кристалі). Нині цей термін також фактично не використовується. Його замінено на термін «мікроконтролер».

1.1.4. Мікропроцесорний комплект

Мікропроцесорний комплект – це сукупність інтегральних схем, сумісних за конструктивно-технологічним виконанням і призначених для спільного використання під час побудови модуля МП, мікро-ЕОМ і МПС.

Поява МП передусім пов'язана з успіхами інтегральної технології.

З моменту «виходу у світ» великої інтегральної схеми (ВІС), одразу стало очевидним те, що тільки ті схеми цифрової обчислювальної техніки можуть бути ефективно реалізовані в них, які мають такі властивості: регулярність структури; серійність, тобто будуть вироблятися великим тиражем; обмежену кількість входів і виходів, відношення якої до загальної

складності ВІС, тобто до кількості транзисторів, реалізованих у ньому, має бути невеликим.

Серед стандартних пристроїв обчислювальної техніки ці властивості притаманні запам'ятовувальним пристроям, які першими почали випускатися у вигляді ВІС. Що стосується керувальних автоматів, то реалізація їх у ВІС пов'язана з великими витратами через те, що кожна керувальна схема є унікальною і, отже, вимагає створення спеціалізованої (замовленої) ВІС в єдиному екземплярі. Прагнення виробників ВІС уникнути великих витрат і разом з тим задовольнити попит на будь-які різноманітні пристрої, здатні виконувати як обчислювальні, так і керувальні функції, привело до думки про створення програмно-керованих універсальних кристалів, тобто наділити логічний елемент властивістю програмуватися, яка залишається з ним у процесі його експлуатації. Виділивши в керувальному пристрої центральний процесор, пам'ять, схеми керування обміном із зовнішнім світом, різні фірми і промислові об'єднання почали виготовляти універсальні елементи, здатні виконувати різноманітні послідовності команд, тобто які мають властивість програмуватися. Ідея ЕОМ була реалізована на рівні кристала. Перші МП були призначені відповідно для застосування у калькуляторах і відеотерміналах, однак завдяки такій універсальності їх почали широко застосовувати і в інших пристроях.

Центральною ланкою будь-якого МП є арифметико-логічний пристрій. Окрім того, до МП можуть належати: швидкодіюча пам'ять для збереження операндів і проміжних результатів – регістри загального призначення (інколи серед них виділяється акумулятор – регістр, в якому зберігається операнд і поміщається результат операції); регістри спеціального призначення – лічильник команд (англ. PC), регістр команд (англ. IR), прапорці та покажчик стека (англ. SP); дешифратор (ДШ) команд і схеми організації машинних циклів; схеми синхронізації; схема формування сигналів, необхідних для керування обміном з пам'яттю і зовнішніми пристроями, а саме реалізації обміну за перериванням і в режимі прямого доступу до пам'яті (ПДП) МПС; формувачі – схеми, необхідні для збільшення навантажувальної здібності виводів, та узгодження схем з різноманітним типом технологій. Прийнято ВІС, що виконують функції МП або деяку частину з них і що вміщують деяку підмножину перерахованих пристроїв, називати мікропроцесорними ВІС.

1.1.5. Мікроконтролер

У пристрої, які містяться в МК (окрім власне МП) можуть бути вміщені: пам'ять для збереження даних, програм і результатів обробки (ОЗП

та постійний запам’ятовувальний пристрій); порти введення/виведення зі схемами керування; таймери; аналого-цифрові та цифро-аналогові перетворювачі (АЦП та ЦАП) і т. ін. Такі пристрої належать до класу МК, в яких усі основні складові мікро-ЕОМ розміщені на одному кристалі.

1.1.6. Основні характеристики мікропроцесора і мікроконтролера

До основних характеристик МП і МК належить: довжина слова даних (розрядність); кількість комірок пам’яті, що адресуються, і швидкодія.

Довжина слова даних визначається максимальною кількістю розрядів оброблюваних даних, що розглядаються апаратною частиною МП/МК, як єдине ціле. У більшості вживаних нині МП/МК довжина слова даних становить 8, 16 та 32 біти.

Кількість адресованих комірок пам’яті залежить від кількості адресних виводів МП/МК. Наприклад, 8-розрядний МП і8080 має 16 адресних ліній (виводів), що дозволяє адресувати $2^{16} = 65536$ комірок пам’яті, завдовжки 8 біт (1 байт), тобто 65536 байт. Частіше для вказівки об’єму пам’яті використовують скорочення 1 К = 1024. Тоді МП і8080 адресує 64 Кбайт пам’яті. 16-розрядний МП і8086 має 20 адресних виводів, тобто здатен адресувати $2^{20} = 1$ Мбайт пам’яті. 8-розрядний МК типу AT89C51 містить 16 адресних виводів, що забезпечує адресацію 64 Кбайт пам’яті.

Швидкодія (продуктивність) МП/МК визначається часом виконання окремих команд і програм. Для оцінювання швидкодії МП/МК можна використовувати:

а) максимальне значення тактової частоти ($f_{\text{такт}}$) або мінімальне значення періоду тактової частоти (тривалість такту машинного циклу);

б) час виконання найпростішої операції, наприклад, пересилання із регістра у регістр;

в) час виконання тестових програм.

Іноді, як основну характеристику, МП/МК називають *потужність*, яка значною мірою визначається названими раніше параметрами: довжиною слова даних, кількістю адресованих комірок пам’яті та швидкодією.

1.2. Системи числення, коди та двійкова арифметика

1.2.1. Системи числення та коди

Загальна характеристика

Система числення (англ. Numeral system або system of numeration) – символічний метод запису чисел, представлення чисел за допомогою письмових знаків.

В обчислювальній і мікропроцесорній техніці використовуються позиційні системи числення – двійкова, як основна, десяткова, шістнадцяткова і вісімкова, як допоміжні [2].

Окрім названих систем числення, у мікропроцесорній техніці для представлення дискретної інформації застосовуються коди з тими ж назвами:

– двійковий;

– шістнадцятковий і т. ін.

Про коди кажуть у тих випадках, коли в тій чи іншій системі числення представлені дискретні повідомлення (дискретна інформація), що називаються даними.

Десяткова система числення

У повсякденному житті використовується десяткова система числення, в якій числа утворюються за допомогою цифр від 0 до 9. Слово «цифра» перекладається від латинського *digitus*, що означає «палець». Така кількість цифр – десять, – пояснюється тим, що у нас десять пальців. Десяткова система є прикладом *позиційної системи числення*, коли положення (позиція, розряд) кожної цифри в числі визначає її значення, або вагу. Наприклад, число 374.29 є скороченим записом виразу

$$(3\cdot 10^2)+(7\cdot 10^1)+(4\cdot 10^0)+(2\cdot 10^{-1})+(9\cdot 10^{-2}).$$

Положення будь-якої цифри визначає степінь числа з основою 10 (вага позиції/розряду числа), на яке ця цифра помножується. В цьому прикладі цифра 3 знаходиться в розряді сотень (10^2), 7 – десятків (10^1), 4 – одиниць (10^0), 2 – десятих часток (10^{-1}) і 9 – сотих часток (10^{-2}).

Основою десяткової системи є число 10.

Позиційні системи числення мають три важливі характеристики:

– кількість цифр системи дорівнює її основі;
– найбільша цифра на одиницю менше основи;
– для визначення значення кожної цифри остання помножується на вагу позиції (основа в степені, значення якої визначається положенням цифри).

Крапка в позиційному представленні чисел називається *десятковою крапкою* і використовується для відділення цілої частини числа від дробової частини. Тому можна записати, що

$$N = N_I + N_F,$$

де N_I та N_F – відповідно ціла і дробова частини числа. В нашому прикладі N_I = 374, а N_F = 0.29.

Двійкова система числення

Найпростішою позиційною системою числення є двійкова. Як зрозуміло з назви, система має основу 2. Для представлення чисел використовуються дві десяткові цифри – 0 і 1. Наприклад, число 1011.1101_2 еквівалентно десятковому числу:

$$(1\times 2^3)+(0\times 2^2)+(1\times 2^1)+(1\times 2^0)+(1\times 2^{-1})+(1\times 2^{-2})+(0\times 2^{-3})+(1\times 2^{-4})=11,8125_{10}.$$

Записуючи числа в тій чи іншій системі числення, підрядкові символи числа визначають основу системи числення. Їх зазвичай опускають, якщо відомо, про яку основу йдеться.

Окрім підрядкових символів для вказівки системи числення, в якій записано число, можуть використовуватися латинські букви: *B* – для двійкової (бінарної); *D* – для десяткової; *H* – для шістнадцяткової. Наприклад: 0110*B*; 7548*D*; 7598*H*; *ABBAH*.

Необхідно зауважити, що в наведеному прикладі крапка в позиційному (двійковому) представленні числа відокремлює його цілу і дробову частину, як і в випадку десяткової крапки.

Під час подання інформації у двійковому коді (ДК) праворуч записують значення молодшого (нульового) значущого розряду, ліворуч – значення старшого (*Np*-1-го) значущого розряду, де N_p – кількість розрядів ДК.

Переведення чисел із десяткової системи числення у двійкову

Припустимо, треба податити десяткове число у двійковій формі. Одним із методів виконання такого перетворення буде метод *ділення-множення*. Під час використання цього методу цілі числа переводяться в систему числення із заданою основою за допомогою послідовного ділення на цю основу. У попередньому прикладі ціла частина числа, яка дорівнює 11_{10}, перетвориться у двійкову форму таким чином:

11 : 2 = 5, залишок 1, молодший значущий розряд,
5 : 2 = 2, залишок 1, ↑
2 : 2 = 1, залишок 0, ↑
1 : 2 = 0, залишок 1, старший значущий розряд.

Процес ділення продовжується доти, поки не отримаємо результат, що дорівнює 0. Залишки від ділення потім виписуються, починаючи з останнього, або старшого значущого розряду, і закінчуючи першим, або молодшим значущим розрядом.

У розглянутому випадку $N_I = 11_{10} = 1011_2 = 1011B$.

Перетворення дробової частини числа в систему числення із заданою основою здійснюється за допомогою виконання низки послідовних операцій множення на цю основу. В розглянутому прикладі дробова частина числа N_F, яка дорівнює 0.8125_{10}, переводиться у двійкову систему таким чином:

$0.8125 \cdot 2 = 1.6250 = 0.6250$, перенесення 1 (старший значущий розряд),
$0.6250 \cdot 2 = 1.2500 = 0.2500$, 1,
$0.2500 \cdot 2 = 0.5000 = 0.5000$, 0,
$0.5000 \cdot 2 = 1.0000 = 0.0000$ 1 (молодший значущий розряд).

Процес послідовного множення продовжується до одержання нульового результату (що не завжди можливо) або до заповнення двійкових розрядів, виділених для представлення числа. Цифри перенесення потім виписують, починаючи зі старшого значущого розряду. У розглянутому прикладі $N_F = 0.8125_{10} = 0.1101_2$. Тому маємо $N = 11.8125_{10} = 1011.1101_2$.

Двійкове лічення

Лічба у двійковій системі числення менш складна, ніж в інших системах, оскільки тут використовується усього дві цифри – 0 і 1. Якщо трапляється деяка послідовність подій, то першій події відповідає цифра 1. Однак вже для другої події цифри 2 у двійковій системі не має. У цьому випадку відбувається перенесення в наступний розряд, що дає дворозрядне двійкове число. Якщо перевести це число в десяткову систему числення, то отримаємо бажаний результат:

$$10_2 = (1 \cdot 2^1) + (0 \cdot 2^0) = 2_{10}.$$

У табл. 1.1 подано послідовність двійкових чисел, яка відповідає десятковим числам від 0 до 9.

Таблиця 1.1. **Подання чисел у двійковій та десятковій системах числення**

Двійкове число	Десяткове число
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9

Зауважимо, що перенесення одиниці у наступний більш старший розряд відбувається під час зображення усіх парних десяткових чисел.

Під час зображення десяткових чисел 2, 4, 8, 16 і т. д. відбуваються відповідно перенесення із розряду двійок (2^1), четвірок (2^2), вісімок (2^3) і т. д.

Вісімкова система числення

Одна з систем числення – вісімкова система, яка також інколи використовується під час роботи з МП/МК. Це система з основою 8 і набором цифр від 0 до 7.

Як і у випадку двійкової і десяткової системи числення, крапка під час позиційного вісімкового представлення числа відокремлює його цілу частину від дробової.

Шістнадцяткова система числення

Двійковими числами великої розрядності важко оперувати, тому використовують їх більш компактне подання, яке тісно пов’язано з двійковим. Наприклад, у шістнадцятковій системі числення числа подаються за степенями основи 16 ($16 = 2^4$), що еквівалентно кількості комбінацій 4-розрядного двійкового числа. В шістнадцятковій системі числення використовуються такі символи: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *A*, *B*, *C*, *D*, *E*, *F* (табл. 1.2).

Переведення із шістнадцяткової системи в інші системи числення здійснюється аналогічно двійковій і десятковій. Також аналогічно виконуються арифметичні дії над шістнадцятковими числами.

Двійково-десяткові числа (коди)

Є розбіжність між машинним поданням чисел (двійкова система числення) і поданням чисел у нашому повсякденному житті (десяткові числа).

Перетворення між ними, у випадку великого об’єму вхідних даних і вихідних результатів, призводить до помітних втрат процесорного часу. Разом з тим, є велике коло задач, що характеризуються значними об’ємами числових даних і порівняно простою їх обробкою (економічні задачі, статистичні та бухгалтерські розрахунки і т. ін.). Тому було розроблено такі форми подання чисел, в яких поєднувалися двійкова і десяткова системи числення. Такі форми отримали загальну назву *двійково-десяткового кодування* (Binary-Coded-Decimal) або BCD-кодування. Загальна властивість цих форм виявляється в тому, що за основу береться десяткове число і кожна його цифра зображується тим чи іншим двійковим еквівалентом.

До теперішнього часу з численних систем двійково-десяткового кодування практично застосовуються тільки дві: двійково-десяткові числа в запакованому і незапакованому форматах.

Таблиця 1.2. **Подання чисел у десятковій, двійковій та шістнадцятковій системах числення**

Десяткове число	Двійкове число	Шістнадцяткове число
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Двійково-десяткові числа у запакованому форматі

У запакованому форматі байт (вісім біт) містить дві десяткові цифри. Менша цифра займає праву тетраду (біти 3…0), старша – ліву тетраду (біти 7…4). Обидві цифри представлені своїми двійковими еквівалентами та називаються також кодом 8421 – за двійковими вагами окремих розрядів тетради. Розміщення десяткових цифр у байті показано на рис. 1.1*а*.

Багаторозрядні запаковані десяткові числа займають декілька суміжних байтів. У разі необхідності роботи зі знаковими числами старша тетрада (іноді – молодша) старшого байта призначається для знака числа (рис. 1.1*б*).

Для кодування знака можна використовувати шість заборонених у разі *двійково-десяткового кодування* тетрад 1010…1111 (або у шістнадцятковій системі символів від *A* до *F*), які не є десятковими цифрами.

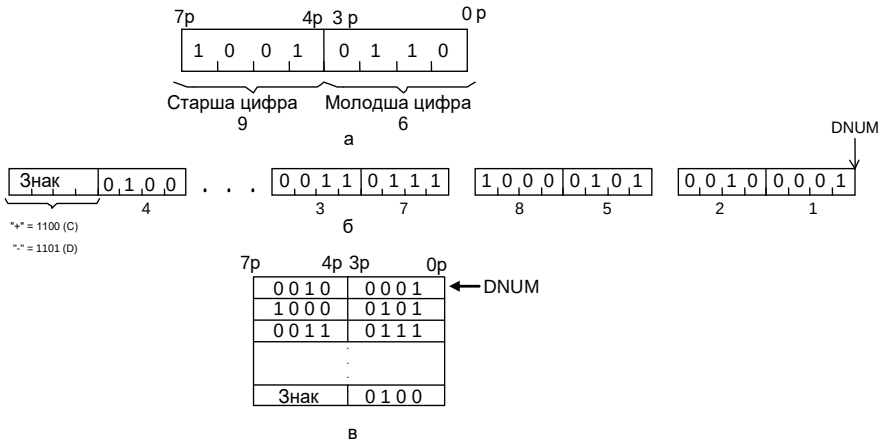


Рис. 1.1. Формат запакованих двійково-десяткових чисел: *а* – кодування байта; *б* – багаторозрядне число; *в* – розміщення багаторозрядного числа в пам'яті

Для зображення знака «плюс» застосовується код 1100 (*C*), а знака «мінус» – код 1101 (*D*). Через необхідність зображення знака багатобайтові запаковані десяткові числа мають непарну кількість розрядів (байт).

Під час програмування запаковані десяткові числа визначаються початковою адресою – DNUM (це адреса молодшого байта) і кількістю байтів – *N*. Розміщення запакованого десяткового числа в пам'яті показано на рис. 1.1*в*.

Двійково-десяткові числа у незапакованому форматі

Приклад чисел у незапакованому двійково-десятковому форматі наведено на рис. 1.2.

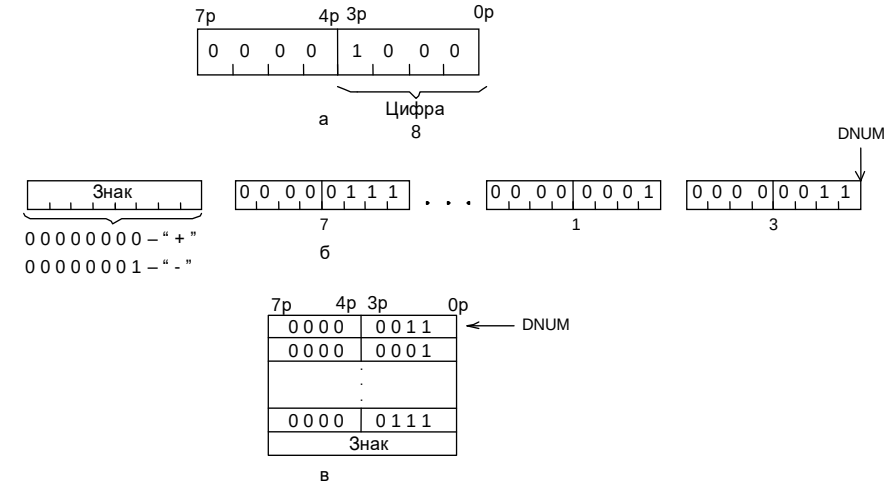


Рис. 1.2. Формат незапакованих двійково-десяткових чисел: *а* – кодування байта; *б* – багаторозрядне число; *в* – розміщення багаторозрядного числа в пам'яті

У цьому форматі десятковим числам відповідають двійкові набори від 0000 0000 (цифра 0) до 0000 1001 (цифра 9), або в шістнадцятковій системі – від 00H до 09H. Отже, можна вважати, що власне значення десяткової цифри

займає молодшу тетраду і подано її двійковим еквівалентом, а в старшій тетраді знаходиться комбінація 0000.

Багаторозрядні незапаковані двійково-десяткові числа займають суміжні байти (рис. 1.2*б*). Для знака числа призначається старший байт, в якому комбінація 0000 0000 (00H) означає знак плюс, а комбінація 0000 0001 (01H) – знак мінус. Такі числа визначаються в програмах їх початковою адресою (DNUM) і кількістю розрядів (байт) або довжиною (*N*). Розміщення незапакованого двійково-десяткового числа в пам'яті показано на рис. 1.2*в*.

Подання алфавітно-цифрової інформації в коді ASCII

Для кодування алфавітно-цифрової інформації або символів використовується 7-розрядний код ASCII (американський стандартний код для обміну інформацією), який у російськомовній літературі називають кодом КОІ-7 (табл. 1.3). Крім цього, може бути декілька інших кодувань [2].

У цьому коді десятковим цифрам відповідають двійкові набори від 00110000 (цифра 0) до 00111001 (цифра 9), або в шістнадцятковій системі від 30H до 39H.

Ця форма під час кодування десяткових цифр відрізняється від незапакованого формату тим, що старша тетрада байта містить 0011B = 3D замість 0000B = 0D.

У табл. 1.3, починаючи з № 128 по № 255, коди символів залежать від національного кодування (власна для кожної країни).

Формати подання чисел. Подання чисел зі знаком

Усі системи числення допускають використання як додатних, так і від'ємних чисел, для позначення яких, як відомо, застосовують знаки плюс (+) або мінус (–). Число в такому поданні називається числом зі знаком.

Оскільки ЕОМ побудовані на схемах двійкової логіки та для подання двійкових станів застосовуються символи нуль і одиниця, тому ці ж символи необхідно використовувати і для позначень знака числа. Зазвичай додатковий розряд для подання знака числа називається знаковим розрядом і розміщується з лівого боку найбільшого значущого розряду числа. Для позначення додатних і від'ємних значень використовують відповідно: нуль та одиниця.

Нижче описано три способи подання двійкових чисел зі знаком, які відомі як прямий, зворотний і додатковий код числа.

Таблиця 1.3. **Код ASCII**

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
0	0		53	35	5	106	6A	j	159	9F	Я
1	1		54	36	6	107	6B	k	160	A0	а
2	2		55	37	7	108	6C	l	161	A1	б
3	3		56	38	8	109	6D	m	162	A2	в
4	4		57	39	9	110	6E	n	163	A3	г
5	5		58	3A	:	111	6F	o	164	A4	д
6	6		59	3B	;	112	70	p	165	A5	е
7	7		60	3C	<	113	71	q	166	A6	ж
8	8		61	3D	=	114	72	r	167	A7	з
9	9		62	3E	>	115	73	s	168	A8	и
10	A		63	3F	?	116	74	t	169	A9	й
11	B		64	40	@	117	75	u	170	AA	к
12	C		65	41	A	118	76	v	171	AB	л
13	D		66	42	B	119	77	w	172	AC	М
14	E		67	43	C	120	78	x	173	AD	н
15	F		68	44	D	121	79	y	174	AE	о
16	10		69	45	E	122	7A	z	175	AF	п
17	11		70	46	F	123	7B	{	176	B0	▒
18	12		71	47	G	124	7C	|	177	B1	▒
19	13		72	48	H	125	7D	}	178	B2	▓
20	14		73	49	I	126	7E	~	179	B3	│
21	15		74	4A	J	127	7F		180	B4	┤
22	16		75	4B	K	128	80	А	181	B5	╡
23	17		76	4C	L	129	81	Б	182	B6	╢

Продовження табл. 1.3

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
24	18		77	4D	M	130	82	В	183	B7	╖
25	19		78	4E	N	131	83	Г	184	B8	╕
26	1A		79	4F	O	132	84	Д	185	B9	╣
27	1B		80	50	P	133	85	Е	186	BA	║
28	1C		81	51	Q	134	86	Ж	187	BB	╗
29	1D		82	52	R	135	87	З	188	BC	╝
30	1E		83	53	S	136	88	И	189	BD	╜
31	1F		84	54	T	137	89	Й	190	BE	╛
32	20		85	55	U	138	8A	К	191	BF	┐
33	21	!	86	56	V	139	8B	Л	192	C0	└
34	22	“	87	57	W	140	8C	М	193	C1	┴
35	23	#	88	58	X	141	8D	Н	194	C2	┬
36	24	$	89	59	Y	142	8E	О	195	C3	├
37	25	%	90	5A	Z	143	8F	П	196	C4	─
38	26	&	91	5B	[	144	90	Р	197	C5	┼
39	27	′	92	5C	\	145	91	С	198	C6	╞
40	28	(	93	5D	]	146	92	Т	199	C7	╟
41	29	)	94	5E	^	147	93	У	200	C8	╚
42	2A	*	95	5F	–	148	94	Ф	201	C9	╔
43	2B	+	96	60	‘	149	95	Х	202	CA	╩
44	2C	′	97	61	a	150	96	Ц	203	CB	╦
45	2D	D	98	62	b	151	97	Ч	204	CC	╠
46	2E	.	99	63	c	152	98	Ш	205	CD	═
47	2F	/	100	64	d	153	99	Щ	206	CE	╬
48	30	0	101	65	e	154	9A	Ъ	207	CF	╧
49	31	1	102	66	f	155	9B	Ы	208	D0	╨
50	32	2	103	67	g	156	9C	Ь	209	D1	╤
51	33	3	104	68	h	157	9D	Э	210	D2	╥
52	34	4	105	69	i	158	9E	Ю	211	D3	╙

Закінчення табл. 1.3

DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR	DEC	HEX	CHAR
212	D4	╙	224	E0	р	236	EC	ь	248	F8	Ї
213	D5	╒	225	E1	с	237	ED	э	249	F9	ї
214	D6	╓	226	E2	т	238	EE	ю	250	FA	Ў
215	D7	╫	227	E3	у	239	EF	я	251	FB	ў
216	D8	╪	228	E4	ф	240	F0	Ё	252	FC	ⁿ
217	D9	┘	229	E5	х	241	F1	ё	253	FD	²
218	DA	┌	230	E6	ц	242	F2	Ґ	254	FE	■
219	DB	▉	231	E7	ч	243	F3	ґ	255	FF	
220	DC	▄	232	E8	Ш	244	F4	Є			
221	DD	▌	233	E9	Щ	245	F5	є			
222	DE	▐	234	EA	Ъ	246	F6	І			
223	DF	▀	235	EB	Ы	247	F7	і			

Прямий код числа

Найпростішим способом подання числа зі знаком є прямий ДК числа. Число у прямому ДК називають також двійковим числом зі знаком. Для числа, що займає N двійкових розрядів, старший значущий розряд використовується для подання знака, а в інших розрядах, що залишилися, записується абсолютне значення числа у двійковій системі. Розглянемо, наприклад, запис числа -13_{10} за допомогою п’яти двійкових розрядів. Маємо

$$-13_{10} = 1.1101,$$

знаковий розряд (–).

У разі такого запису для відділення знакового розряду від цифрових розрядів використовувалась кома, яка, зазвичай, опускається. Наведемо ще один приклад:

$$+27_{10} = 011011,$$

знаковий розряд (+).

Недолік цього способу полягає в тому, що процедури додавання чисел з різними знаками ускладнені та існує дві форми представлення нуля: зі знаком плюс – 00000000 та зі знаком мінус – 10000000.

Для зберігання інформації та під час виконання арифметичних операцій у МПС такий спосіб подання чисел зі знаком не використовується.

Зворотний код числа

Іншим способом подання від'ємних чисел є подання числа за допомогою зворотного коду. Для одержання зворотного коду числа береться його додатне двійкове подання і зліва доповнюється нулями до числа розрядів, що відповідають одному машинному слову, наприклад, вісім. Після цього кожний нуль замінюється на одиницю, а кожна одиниця – на нуль:

$$(0\ 0\ 0\ 0\ 0\ 0\ 1\ 1)_2 \Rightarrow (1\ 1\ 1\ 1\ 1\ 1\ 0\ 0)_2.$$

Прямий код зворотний код.

Зворотний код від'ємного числа називають також доповненням до одиниці. Перевагами такого кодування є легкість одержання зворотного коду і можливість не враховувати знаки у разі додавання та віднімання.

Недоліки такого кодування полягають у такому: утворюються два різних подання нульового результату: комбінацією двійкових нулів або двійкових одиниць; необхідне циклічне перенесення у разі додавання для одержання правильного результату.

Перетворення від'ємного числа у зворотному коді у десяткову систему числення можна здійснити двома способами. Один з них є зворотним процедурі порозрядного доповнення числа до одиниці. Розглянемо, наприклад, 4-розрядне число 1.011. Утворюючи порозрядне доповнення до одиниці, одержуємо число 0.100 або +4. Таким чином, 1.011 = –4.

В іншому способі використовується те, що знаковий розряд N-розрядного двійкового числа має від'ємну вагу, яка дорівнює $2^{N-1}–1$. У нашому випадку $n = 4$; $2^{4-1}–1 = 7$ і вага знакового розряду є (–7). Таким чином, маємо

$$1.011 = –7 + 3 = –4.$$

Цей спосіб дуже зручний, оскільки тут лише переводимо в десяткову систему вміст цифрових розрядів, і до отриманого результату додаємо від'ємну вагу знакового розряду.

Як недолік слід зазначити, що у зворотних ДК є два коди числа 0: «додатний нуль» 0000 0000 і «від'ємний нуль» 1111 1111 (наведені 8-розрядні зворотні коди).

Додатковий код числа

Додатковий код (англ. two's complement, іноді twos-complement) – найпоширеніший спосіб представлення від'ємних цілих чисел у мікропроцесорній техніці. Він дозволяє замінити операцію віднімання операцією додавання і зробити операції додавання і віднімання однаковими для знакових і беззнакових чисел, що спрощує архітектуру ЕОМ.

В англомовній літературі «зворотний код» називають «доповненням одиниць» (англ. one’s complement), а «додатковий код» називають «доповненням двійок» (англ. two’s complement). Додатковий код додатного числа збігається із прямим кодом. Додатковий код для від’ємного числа можна отримати інвертуванням його двійкового модуля та додаванням до інверсії одиниці, або віднімання числа з нуля.

Подання чисел у вигляді додаткового коду має такі переваги порівняно з іншими формами подання:

– операції додавання і віднімання на практиці виконуються просто;
– є єдине подання нуля.

Як видно з табл. 1.4 за допомогою 8 біт можна подати у двійковій формі десяткові числа зі знаком від –128 до +127, включаючи 0.

У таблиці показано два типові для МП способи використання ДК: як двійкових чисел зі знаком, так і без знака. Лівий стовпчик містить двійкові числа (двійкові коди) від 0000 0000 до 1111 1111, правий стовпчик – їх десяткові еквіваленти від 0 до 255, отримані з припущення, що розглядаються числа без знака. У центральному стовпчику знаходяться десяткові еквіваленти двійкових чисел лівого стовпчика, отримані з припущення, що від’ємні числа записувалися у додатковому коді.

Тут додатним двійковим числам (від 0000 0000 до 0111 1111) відповідають десяткові числа від 0 до +127, а тим, що вміщують одиницю в восьмому розряді (сьомому, якщо рахувати від нуля) від’ємним двійковим числам (від 1000 0000 до 1111 1111) – десяткові від: –128 до –1.

Використовуючи вісім двійкових розрядів і подаючи від’ємні числа в додатковому коді, можна записати 256 різних чисел: 127 додатних, нуль і 128 від’ємних. Процедуру формування додаткового коду покажемо на такому прикладі:

– число 4_{10} у двійковій формі: 0000 100;
– зворотний код числа 4_{10}: 1111 011;
– додавання до зворотного коду: 1;
– число -4_{10} у додатковому коді: 111 1100.

Порівняємо отриманий результат з відповідним кодом у табл. 1.4.

Для подання двійкового числа у додатковому коді можна користуватися іншим способом, відмінним від описаного вище і коротшим за кількістю операцій. У пошуках першого біта, що дорівнює одиниці, переглядають справа наліво розряди двійкового подання модуля числа, починаючи з найменшого за значенням. Доти, поки трапляються нулі, їх копіюють в однойменні розряди результату. Перша одиниця, що трапилася,

також копіюється у відповідний розряд результату, але кожний наступний біт модуля вихідного числа замінюється на зворотний.

Таблиця 1.4. **Десяткові еквіваленти двійкових чисел**

Двійковий еквівалент	**Десятковий еквівалент**	
8-розрядні двійкові числа без знака 7р 0р	Двійкові числа зі знаком (від'ємні числа у додатковому коді)	Двійкові числа без знака
0000 0000	+0	0
0000 0001	+1	1
0000 0010	+2	2
0000 0011	+3	3
.	.	.
.	.	.
.	.	.
0111 1100	+124	124
0111 1101	+125	125
0111 1110	+126	126
0111 1111	+127	127
1000 0000	–128	128
1000 0001	–127	129
1000 0010	–126	130
1000 0011	–125	131
.	.	.
.	.	.
1111 1100	–4	252
1111 1101	–3	253
1111 1110	–2	254
1111 1111	–1	255

Відповідно до табл. 1.4 арифметичні операції над двійковими числами без знака не відрізняються від подібних операцій над двійковими числами зі знаком, від'ємні з яких подані своїми доповненнями.

Це істотно спрощує апаратну реалізацію подібних операцій у МП/МК. Однак слід звернути увагу на те, з якими числами ви маєте справу в цей момент: без знака чи зі знаком. Наприклад, у разі додавання двох чисел без знака результат – число без знака подається у вигляді деякої послідовності бітів, котру можна інтерпретувати і як від'ємне число у додатковому коді. В загальному випадку у разі додавання або віднімання чисел зі знаком

результатом буде число зі знаком. У цьому разі, якщо біт старшого розряду дорівнює одиниці, то результат – від'ємне число у додатковому коді. Якщо потрібно визначити абсолютне значення (величину) від'ємного результату, останній необхідно подати у зворотному коді, а потім додати одиницю.

Існує простіший спосіб визначення абсолютного значення від'ємного двійкового числа: необхідно додати ваги двійкових розрядів, які вміщують нулі, і до суми додати одиницю (визначення абсолютного значення від'ємного числа 11111000: $2^0 + 2^1 + 2^2 + 1 = 1 + 2 + 4 + 1 = 8$).

У простоті операцій над від'ємними числами у вигляді доповнень до двох можна переконатися на прикладі операції віднімання, яка виконується таким чином: визначається додатковий код від'ємника і виконується додавання цього коду зі зменшуваним. Якщо різниця – число додатне (біт старшого розряду дорівнює нулю), то біт перенесення необхідно відкинути: отримана послідовність бітів і буде ДК. Якщо різниця – число від'ємне (біт старшого розряду дорівнює одиниці), то вона подана в додатковому коді. Вище вказувалося, що необхідно зробити для визначення абсолютної величини від'ємного числа, поданого в такому вигляді.

Наведемо декілька прикладів виконання операції віднімання в МП/МК.

Приклад 1.1. Обчислити різницю чисел 58 – 23:

а) визначення додаткового коду числа 23
0 0 0 1 0 1 1 1 число 23_{10}
1 1 1 0 1 0 0 0 зворотний код числа 23_{10}
0 0 0 0 0 0 0 1 одиниця, що додається до зворотного коду
1 1 1 0 1 0 0 1 додатковий код числа 23_{10} (-23_{10})
б) обчислення різниці

Десяткова арифметика	Двійкова арифметика	
58	0 0 1 1 1 0 1 0	число 58_{10}
–	+	
23	1 1 1 0 1 0 0 1	додатковий код числа 23_{10}
35	1 0 0 1 0 0 0 1 1	різниця 35_{10}.

Одиниця перенесення, що відкидається у випадку додатного результату.

Приклад 1.2. Обчислити різницю чисел 26 – 34:

а) визначення додаткового коду числа 34
0 0 1 0 0 0 1 0 число 34_{10}
1 1 0 1 1 1 0 1 зворотний код числа 34_{10}
0 0 0 0 0 0 0 1 одиниця, що додається до зворотного коду
1 1 0 1 1 1 1 0 додатковий код числа 34_{10} (-34_{10})

б) обчислення різниці

Десяткова арифметика	Двійкова арифметика	
26	0 0 0 1 1 0 1 0	число 26_{10}
–	+	
34	1 1 0 1 1 1 1 0	додатковий код числа 34_{10} (-34_{10})
–08	1 1 1 1 1 0 0 0	різниця -08_{10}

в) визначення абсолютного значення різниці

1 1 1 1 1 0 0 0 різниця у додатковому коді
0 0 0 0 0 1 1 1 зворотний код різниці
0 0 0 0 0 0 0 1 одиниця, що додається до зворотного коду
0 0 0 0 1 0 0 0 абсолютне значення різниці (8_{10}).

Формати чисел з нефіксованою крапкою

Окрім розглянутих форматів чисел з фіксованою крапкою, в техніці МП застосовуються також формати з нефіксованою крапкою, розглянуті у [2].

Угруповання біт

Окремі двійкові біти (розряди) можуть об'єднуватися в групи, що мають назви:

- тетрада (група з 4-х біт);
- байт (група з 8-ми біт);
- слово (група з 16-ти біт);
- подвійне слово (група з 32-х біт).

1.2.2. Двійкова арифметика

Двійкове додавання

Двійкове додавання виконується за такими правилами [2]:

а) 0 + 0 = 0;
б) 0 + 1 = 1;
в) 1 + 1 = 0 плюс перенесення у наступний розряд;
г) 1 + 1 + 1 = 1 плюс перенесення у наступний розряд.

Для ілюстрації цих правил розглянемо додавання чисел 1110 і 1100:

1 1 1 0 перший доданок;
+1 1 0 0 другий доданок;
1 1 0 1 0 сума.

Двійкове віднімання

Двійкове віднімання виконується за такими правилами:

а) 0 – 0 = 0;

б) 1 – 0 = 1;

в) 1 – 1 = 0;

г) 0 – 1 = 1 – позика одиниці зі старшого розряду.

Використання цих правил ілюструється на прикладі віднімання числа 1100 із 10110.

1 0 1 1 0 зменшуване; (+22)

– 1 1 0 0 від'ємник; (+12)

0 1 0 1 0 різниця. (+10)

У МП/МК операція віднімання замінюється додаванням зменшуваного з додатковим кодом від'ємника. Тому операція 10110B – 1100B виконується таким чином:

00010110 зменшуване; (+22)

+

11110100 додатковий код від'ємника; (–12)

00001010 різниця. (+10)

Двійкове множення

Під час двійкового множення частковий добуток зсувається на один розряд вліво для обробки кожного наступного розряду множника. Множення чисел здійснюється за такими правилами [2]:

а) 0 x 0 = 0;

б) 0 x 1 = 0;

в) 1 x 0 = 0;

г) 1 x 1 = 1.

Наприклад, 5 x 3 = 15 у десятковому представленні чисел:

$0\ 1\ 0\ 1_2$ ($+5_{10}$);

x $0\ 0\ 1\ 1_2$ ($+3_{10}$);

0 1 0 1

+ 0 1 0 1

$0\ 1\ 1\ 1\ 1_2$ ($+15_{10}$).

Двійкове ділення

Операція ділення є зворотною відносно множення і реалізується схожими циклічними діями. Алгоритм ділення наведено у [2].

Двійково-десяткова арифметика

Вище було розглянуто два формати представлення двійково-десяткових чисел: запакований і незапакований. У сучасних МП/МК немає команд, що дійсно оперують названими числами. Виконання операцій над двійково-десятковими числами здійснюється за два етапи:

– на першому етапі операнди обробляються як цілі двійкові числа командами двійкової арифметики;

– на другому – спеціальні команди корекції перетворюють проміжний результат у двійково-десятковий формат.

Команди корекції для обох форматів двійково-десяткових чисел представлені, наприклад, у 16-розрядному МП i8086 [2].

1.3. Функціональна схема мікропроцесорної та структурна схема мікроконтролерної систем керування

1.3.1. Функціональна схема мікропроцесорної системи керування

Роботу і застосування будь-якої МПС необхідно розглядати як на апаратному, так і на програмному рівні, оскільки ці дві властивості МПС нероздільні одна від одної [2].

Аналіз МПС на апаратному рівні починають з вивчення структурної або функціональної схеми системи.

Розглянемо приклад функціональної схеми гіпотетичної мікропроцесорної системи керування (МПСК) на 8-розрядному МП, яку наведено на рис. 1.3.

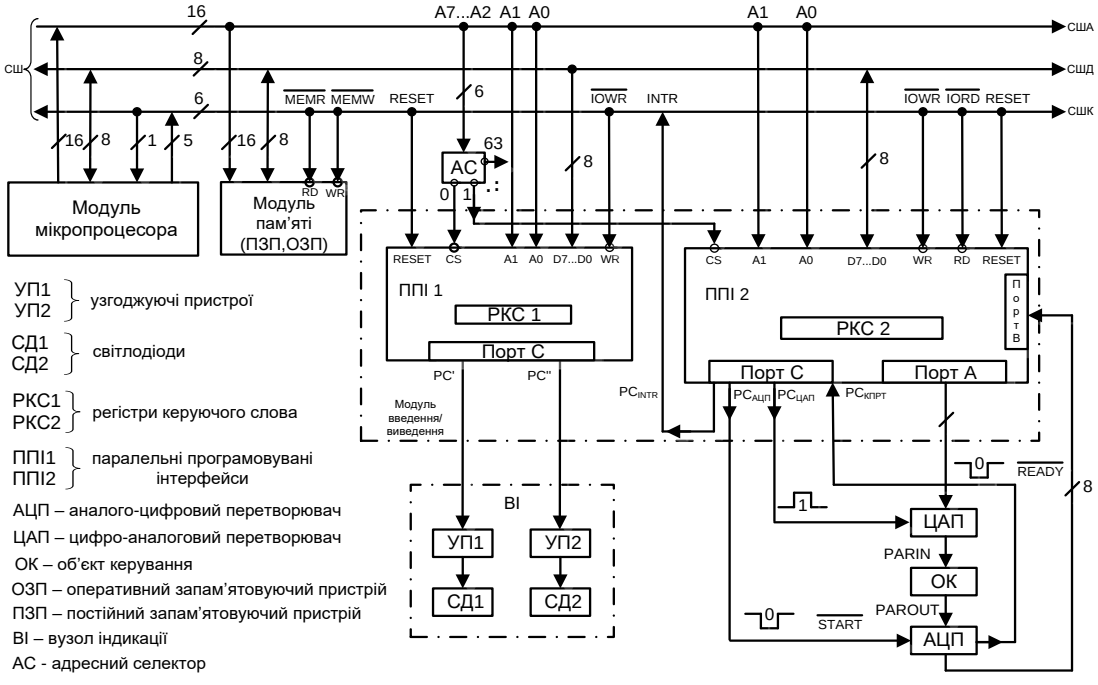


Рис. 1.3. Функціональна схема гіпотетичної МПСК

Ця МПСК орієнтована на 8-розрядний МП, наприклад і8080, і містить такі складові:

- МП;
- модуль пам’яті: постійний запам’ятовувальний пристрій та ОЗП;
- модуль введення/виведення – два паралельні програмовані інтерфейси (1 та 2);
- ЦАП;
- АЦП;
- об’єкт керування;
- вузол індикації.

У постійному запам’ятовувальному пристрої зберігається робоча програма, під керуванням якої працює МПСК, а в ОЗП зберігаються дані та знаходиться стек.

Модуль введення/виведення, що складається з двох паралельних програмованих інтерфейсів, забезпечує обмін інформацією між МП, об’єктом керування та індикацією.

АЦП і ЦАП призначені для поєднання аналогового об’єкта керування з цифровими вузлами МПСК.

Цифрові вузли МПС зв’язані між собою системною шиною, що складається із трьох окремих системних шин:

- системної шини адреси;
- системної шини даних;
- системної шини керування.

Системною шиною адреси передається адреса від МП до пристрою, з яким у цей момент часу здійснюється обмін інформацією: пам’яттю або пристроєм введення/виведення.

Системною шиною даних МП обмінюється інформацією (даними) між пам’яттю і пристроєм введення/виведення. У системі, що розглядається, із пам’яті в МП по системній шині даних передаються команди робочої програми, яка керує роботою МПСК.

Через паралельний програмований інтерфейс 1 від МП до вузла індикації по системній шині даних передається інформація, що відображає стан об’єкта керування.

Від паралельного програмованого інтерфейсу 2 у МП по системній шині даних передається інформація про поточне значення контрольованого параметра PAROUT.

Від МП до паралельного програмованого інтерфейсу 2 по системній шині даних пересилається керувальна інформація для підтримки контрольованого параметра об’єкта керування на заданому рівні.

Системною шиною керування передаються керувальні сигнали, які виробляються модулем МП.

У розглянутому прикладі таких сигналів п’ять:

– RESET – системне скидання (здійснює початкове налагодження (ініціалізацію) паралельного програмованого інтерфейсу 1 і 2);
– $\overline{\text{MEMR}}$ – читання пам’яті (керує читанням програми з постійного запам’ятовувального пристрою або даних з ОЗП);
– $\overline{\text{MEMW}}$ – запис у пам’ять (керує записом до ОЗП);
– $\overline{\text{IORD}}$ – читання пристрою введення/виведення (керує введенням інформації від зовнішнього пристрою до МП);
– $\overline{\text{IOWR}}$ – запис у пристрій введення/виведення (керує виведенням інформації з МП у зовнішній пристрій).

Один керувальний сигнал (INTR) передається по системній шині керування у МП і дозволяє організувати обмін даними через пристрій введення/виведення (ППІ2) за перериванням.

У реальних системах кількість керувальних сигналів, що передаються по системній шині керування, може бути більше названих п’яти.

У конкретний момент часу МП виконує передачу інформації в одному з двох напрямків:

– із МП у пам’ять (для МПСК, що мають ОЗП) або пристрій введення/виведення;
– у МП із пам’яті або пристрою введення/виведення.

У першому випадку кажуть, що відбувається запис (виведення, передача), а в другому – читання (введення, прийом). В обох випадках інформація передається по системній шині даних.

Окрім того, необхідно зазначити, що МП не може одночасно працювати з пам’яттю і пристроєм введення/виведення, а взаємодіє тільки з одним із них, який обраний адресним селектором і керувальним сигналом $\overline{\text{MEMRD}}/\overline{\text{MEMW}}$ або $\overline{\text{IORD}}/\overline{\text{IOWR}}$.

Оскільки передбачається, що в прикладі (рис. 1.3) об’єкт керування – аналоговий, він містить аналогові давачі і виконавчі елементи. Тоді для зв’язку такого об’єкта керування з цифровими вузлами МПС використовуються: АЦП у разі введення в МПСК значення контрольованого параметра від давача, і ЦАП у разі виведення керувального впливу на виконавчий елемент.

До ЦАП на рис. 1.3 умовно входять:

– власне ЦАП, що містить резисторну матрицю R-2R та операційний підсилювач [1; 2];
– буферний регістр, в який у ДК переписується керувальний вплив із порту А паралельного програмованого інтерфейсу 2 у момент

надходження синхроімпульсу, що програмно сформований на лінії порту С паралельного програмованого інтерфейсу 2 ($РС_{ЦАП}$).

У реальній системі буферний регістр може не використовуватися, а його функцію буде виконувати буферний регістр порту А паралельного програмованого інтерфейсу 2.

Для запуску АЦП на лінії $РС_{АЦП}$ порту С паралельного програмованого інтерфейсу 2 програмно формується керувальний імпульс ($\overline{START}$) низького рівня. Після завершення аналого-цифрового перетворення АЦП формує сигнал «Готовність» ($\overline{READY}$), який також має низький рівень та надходить у систему по лінії $РС_{КПРТ}$ порту С паралельного програмованого інтерфейсу 2. Дані від АЦП у МП передаються на системну шину даних через порт В паралельного програмованого інтерфейсу 2.

Є два способи введення даних у МП через порт В:

– програмно-керований;

– за перериванням.

У першому випадку програма постійно перевіряє значення сигналу $РС_{КПРТ}$ і, у разі виявлення логічного нуля, керує введенням даних у МП. Очевидно, що в цьому разі МП використовується не дуже ефективно, оскільки витрачає свій машинний час на постійне опитування біта $РС_{КПРТ}$, доки АЦП не закінчить перетворення.

У другому випадку МП може виконувати необхідну роботу з оброблення даних до надходження сигналу «INTR – запит переривання», що формується в момент закінчення перетворення АЦП і по системній шині керування передається на однойменний вивід МП.

У цьому разі відбувається переривання основної програми, і черговий байт вводиться в МП (у нашому прикладі через порт В паралельного програмованого інтерфейсу 2).

Після введення чергового байта формується сигнал RESET для АЦП, що передається тією ж лінією, що і сигнал $\overline{START}$, але має інверсне до нього одиничне значення.

Зображений на рис. 1.3 адресний селектор може бути реалізовано за допомогою двійкового ДШ, що перетворює вхідний паралельний 6-розрядний ДК (розряди А7…А2 системної шини адреси) у багатопозиційний унітарний код (число позицій становить $2^6 = 64$) з активними нульовими вихідними сигналами.

Активний логічний нуль буде з’являтися на тому виході ДШ, номер якого є десятковим еквівалентом вхідного ДК.

У розглянутому прикладі використовуються тільки два виходи з 64-х (наприклад, 0 і 1), що у потрібний момент обирають один із двох

паралельних програмованих інтерфейсів. Інші виходи в реальній системі можуть застосовуватися для адресації інших зовнішніх пристроїв.

У реальній системі функції модуля введення/виведення можуть бути виконані за допомогою тільки одного паралельного програмованого інтерфейсу. У наведеній функціональній схемі не використовуються чотири розряди порту С паралельного програмованого інтерфейсу 2, два з яких можна використати для керування вузлом індикації.

Робота МПСК (рис. 1.4) зводиться до підтримування аналогового вихідного сигналу об’єкта керування, позначеного PAROUT, у межах заданого діапазону:

PARMIN $\leq$ PAROUT < PARMAX, де PARMIN і PARMAX – відповідно мінімальне і максимальне значення регульованого параметра.

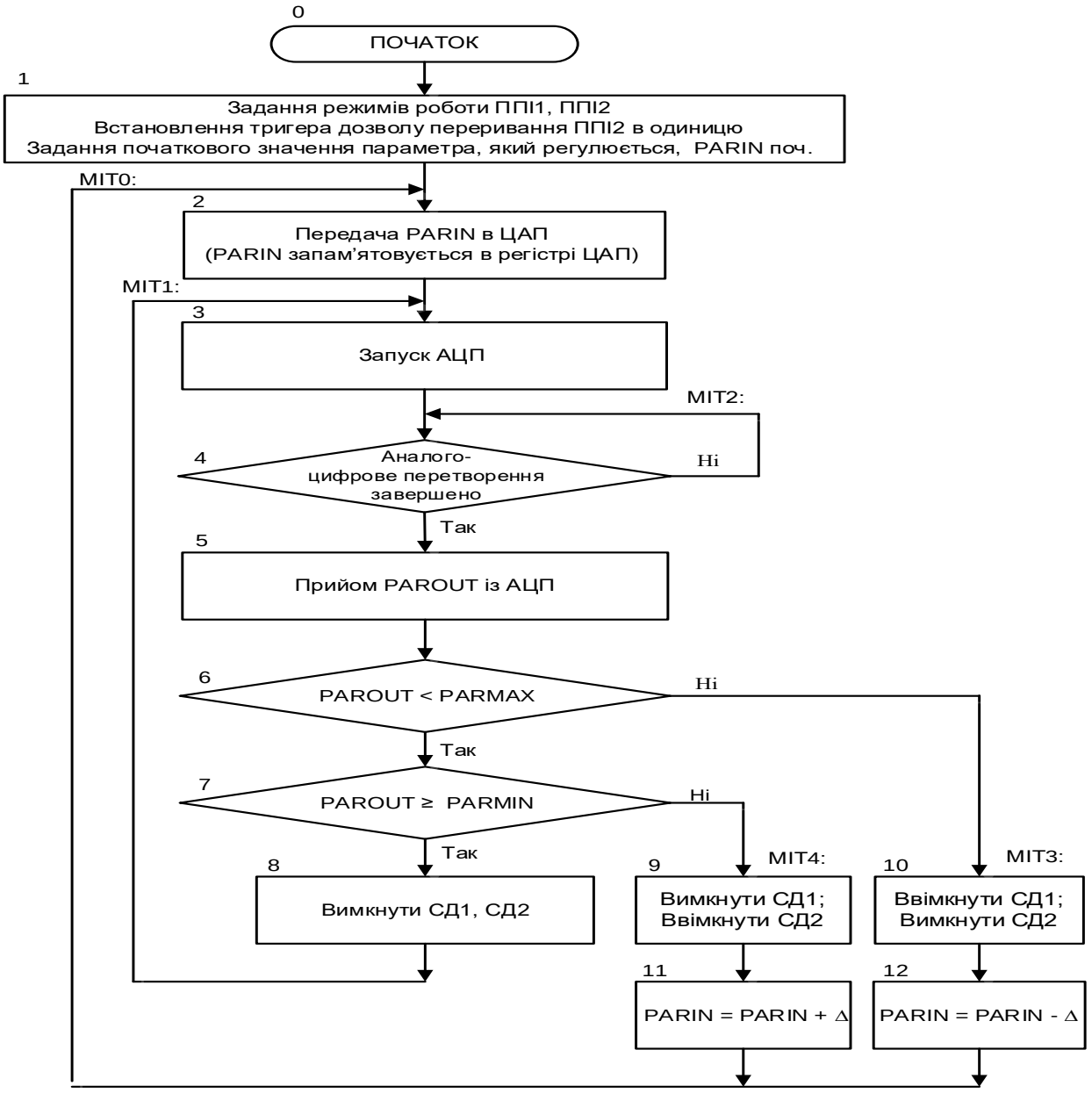


Рис. 1.4. Схема алгоритму роботи МПСК

На початку роботи для виведення об’єкта керування на робочий режим через порт А ППІ2 у ЦАП передається початкове значення регульованого параметра $PARIN_{поч}$.

Якщо в процесі роботи системи значення PAROUT перебуває в межах зазначеного вище діапазону, то значення $PARIN_{поч.}$ не змінюється. Якщо PAROUT вийшов за межі діапазону з боку мінімуму чи максимуму, то поточне значення PARIN змінюється з кроком $\pm\Delta$, розмір якого може програмно змінюватися.

Ця зміна буде вироблятися доти, поки PAROUT знову не опиниться в межах заданого діапазону.

Порушення чи не порушення границь діапазону у розглянутій системі відображаються ВІ (двома світлодіодами: СД1 і СД2).

Індикацією відображаються три стани регульованого об’єкта:

– норма (PAROUT знаходиться в межах допуску) – обидва світлодіоди вимкнено;
– PAROUT ≥ PARMAX (вихід за межі допуску з боку максимуму) – вмикається СД1 і вимикається СД2;
– PAROUT < PARMIN (вихід за межі допуску з боку мінімуму) – вимикається СД1 і вмикається СД2.

Розробку програмного забезпечення звичайно починають зі схеми алгоритму роботи системи, яку для нашого прикладу наведено на рис. 1.4.

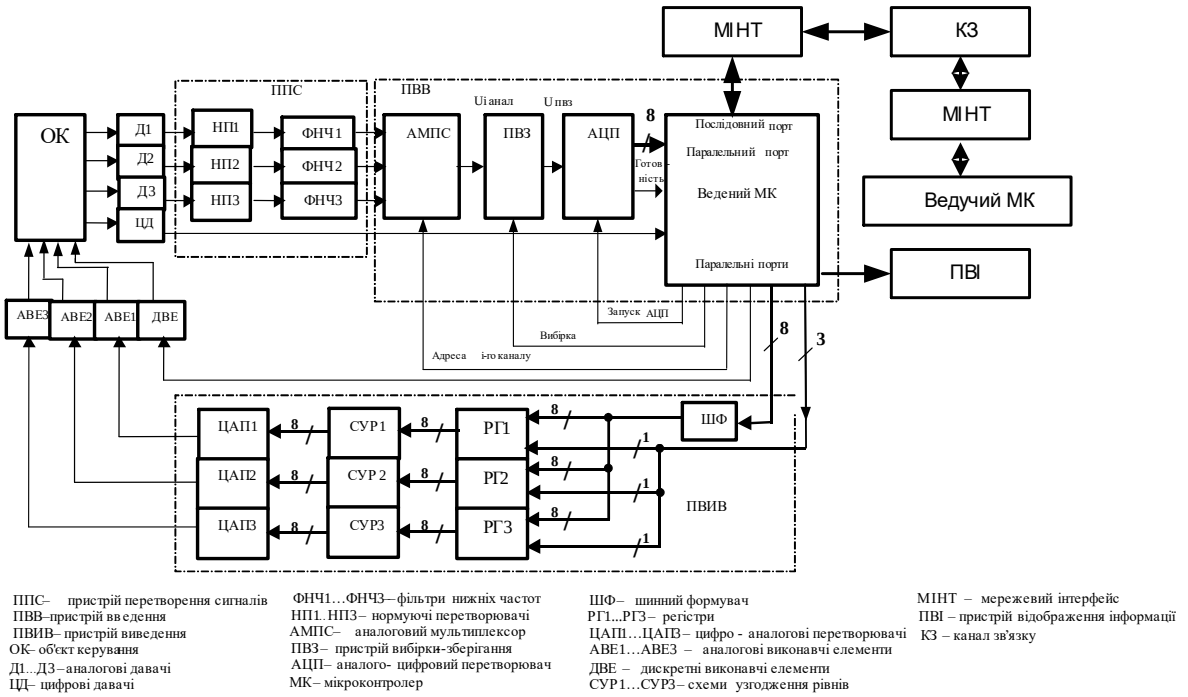
Після цього розробляється і налагоджується керувальна програма мовою Асемблера або мовою С обраного МП або МК.

Після налагодження ця програма компілюється – транслюється в машинні коди процесора і створюється об’єктний модуль, що на спеціальному пристрої – програматорі – завантажується у постійний запам’ятовувальний пристрій МПСК.

1.3.2. Структурна схема мікроконтролерної системи керування

1.3.2.1. Опис структурної схеми системи

Приклад структурної схеми типової гіпотетичної локальної мікроконтролерної системи керування наведено на рис. 1.5 [2].



ППС– пристрій перетворення сигналів
ПВВ–пристрій ввеедення
ПВИВ– пристрій виведення
ОК- об'єкт керування
Д1…ДЗ – аналогові давачі
ЦД– цифрові давачі

ФНЧ1…ФНЧЗ—фільтри нижніх частот
НП1..НПЗ– нормуючі перетворювачі
АМПС– аналоговий мультиплексор
ПВЗ– пристрій вибірки-зберігання
АЦП– аналого- цифровий перетворювач
МК– мікроконтролер

ШФ– шинний формувач
РГ1...РГЗ– регістри
ЦАП1…ЦАПЗ – цифро - аналогові перетворювачі
АВЕ1…АВЕ3 – аналогові виконавчі елементи
ДВЕ – дискретні виконавчі елементи
СУР1…СУРЗ – схеми узгодження рівнів

МІНТ – мережевий інтерфейс
ПВІ – пристрій відображення інформації
КЗ – канал зв'язку

Рис. 1.5. Структурна схема типової локальної мікроконтролерної системи керування

Локальна мікроконтролерна система керування керує визначеним об'єктом керування (агрегатом) за декількома параметрами, наприклад, температурою, тиском, кутом повороту, переміщенням і т. ін.

Система названа локальною, оскільки керування виробляється і здійснюється на нижньому (локальному) рівні складної розподіленої системи керування, що включає декілька різних агрегатів (об'єктів керування).

Основним елементом локальної мікроконтролерної системи керування є МК, який називається веденим, оскільки передбачається, що в складній розподіленій системі мається декілька подібних ведених МК, які керують окремими агрегатами на локальному рівні.

На вищому рівні ієрархії розподіленої системи керування може знаходитися ведучий МК, який на основі інформації про стан окремих агрегатів виробляє необхідні значення заданих керувальних впливів для ведених МК. Ведучий і ведений МК можуть бути пов'язані між собою мікроконтролерною мережею (див. розд. 8).

Локальна мікроконтролерна система керування підтримує кожний з конкретних параметрів на заданому рівні. Інформація про поточне значення параметрів контролю може, наприклад, зніматися з аналогових давачів (Д1…Д3) і проходити через нормувальні перетворювачі (НП1…НП3), які перетворюють діапазон зміни електричних сигналів, що знімаються з давачів, до діапазону вхідних сигналів, що відповідає обраному АЦП.

Оскільки інформаційні сигнали в більшості систем керування низькочастотні, то для придушення високочастотних завад використовуються фільтри нижніх частот.

Аналоговий мультиплексор по черзі підключає до АЦП один з трьох аналогових електричних сигналів, які відображають поточні значення контрольованих параметрів.

У випадку, якщо за час перетворення АЦП, зміна вхідного сигналу відповідає зміні вихідного ДК на виході АЦП більше, ніж на одиницю молодшого значущого розряду, то для зменшення «апертурної» похибки, яка виникає у цьому разі, в систему включають пристрій вибірки-зберігання [2; 17].

Пристрій вибірки-зберігання запам’ятовує миттєві значення вхідних аналогових сигналів у момент часової вибірки і підтримує їх постійними на вході АЦП протягом часу перетворення останнього.

З виходу АЦП інформація у паралельному двійковому коді надходить у ведений МК, який порівнює поточне значення контрольованого параметра з заданим значенням і виробляє керувальний вплив відповідно до сигналу розузгодження та обраним законом керування (П, ПІ, ПІД).

Сигнали керування, наприклад, для трьох аналогових виконавчих елементів знімаються з виходу одного з паралельних портів МК та запам’ятовуються у зовнішніх регістрах РГ1…РГ3.

Для підвищення навантажувальної здатності виходів паралельного порту МК у системі використано шинний формувач.

Виходи РГ1…РГ3 через схеми узгодження рівнів СУР1…СУР3 пов’язано з входами ЦАП1…ЦАП3, що формують аналогові керувальні впливи, спрямовані на усунення сигналу розузгодження і призначені для відпрацьовування аналоговими виконавчими елементами (АВЕ1…АВЕ3).

Схеми узгодження рівнів (СУР1…СУР3) необхідні в тому разі, коли рівні одиничних логічних сигналів, що знімаються з виходів регістрів, не відповідають необхідним рівням одиничних сигналів на входах ЦАП [1; 2].

В якості схеми узгодження рівнів зазвичай використовують логічні елементи з відкритим колектором [1; 2].

Загалом локальні мікроконтролерні системи керування окрім аналогових давачів і виконавчих елементів можуть містити цифрові давачі і дискретні виконавчі елементи, які через паралельні порти та відповідні інтерфейси поєднуються з МК.

Багато сучасних давачів можуть на своїх виходах формувати сигнали у форматах інтерфейсів: SPI, I^2C, RS-485, 1-WIRE, RS-232 і т. ін.

В цьому випадку для їх обробки в МК останній повинен мати відповідний модуль, або використовувати підпрограми.

У випадку, коли ведучий та ведені МК знаходяться на відстані, наприклад, сотень метрів, для їх зв'язку використовують мікроконтролерні мережі: CAN, TWI, SPI, RS-485, 1-WIRE і т. ін. У разі більшої відстані застосовуються модеми.

У разі необхідності в локальній мікроконтролерній системі керування використовуються пристрої відображення інформації. Для цього можуть застосовуватися семисегментні світлодіодні або рідкокристалічні індикатори, рідкокристалічні дисплеї і т. ін.

Нижче наведено приклади реалізації деяких вузлів локальної мікроконтролерної системи керування, структуру якої наведено вище.

1.3.2.2. Аналоговий мультиплексор

Аналоговий мультиплексор використовується для почергової передачі поточного значення одного з трьох аналогових контрольованих параметрів на вхід пристрою вибірки-зберігання. Для цього може бути, наприклад, використано мікросхему CD4052A [1].

На рис. 1.6 зображено позначення цієї мікросхеми на електричних схемах і пояснюється як аналоговий мультиплексор пов'язаний з іншими частинами локальної мікроконтролерної системи керування.

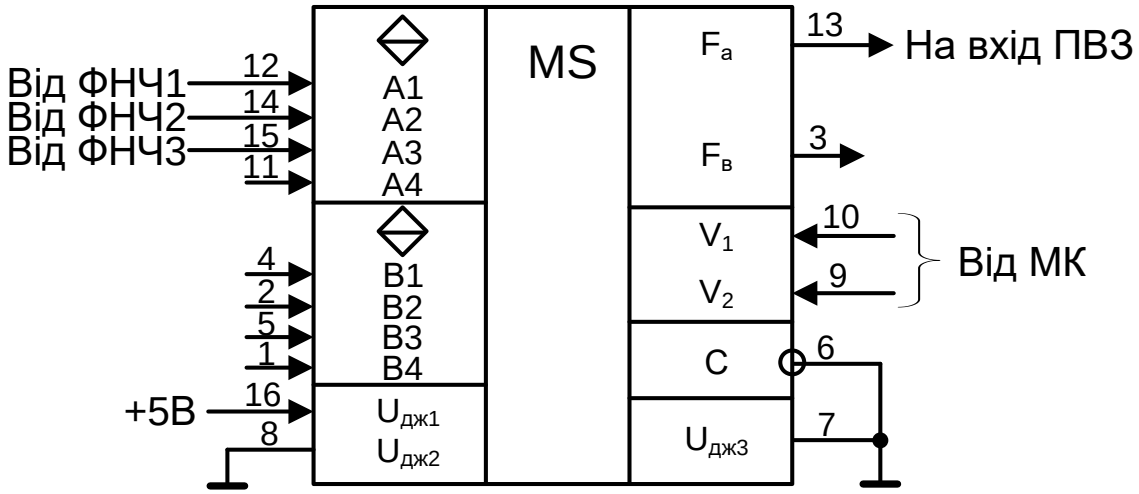


Рис. 1.6. Схема включення аналогового мультиплексора

Розглянутий пристрій належить до класу мультиплексорів-селекторів (мультиплексорів-демультиплексорів). Мікросхема містить два мультиплексори-селектори. В нашому прикладі використано половину мікросхеми як мультиплексор. Залежно від значень адресних сигналів, що надходять від МК на входи V1, V2, у мультиплексорі утворюється

наскрізний низькоомний канал між виходом F_a та одним із входів А1, А2, А3, на які подаються інформаційні сигнали від фільтру нижніх частот. З виходу F_a обраний сигнал надходить на вхід пристрою вибірки-зберігання.

1.3.2.3. Пристрій вибірки-зберігання

В якості пристрою вибірки-зберігання може бути, наприклад, використано мікросхему К1100СК2 [1]. На рис. 1.7 зображено позначення цієї мікросхеми на електричних схемах і пояснюється, як пристрій вибірки-зберігання пов'язаний з іншими частинами локальної мікроконтролерної системи керування. Тривалість імпульсу вибірки – $t_{зап}$ ($t_в$), який записує інформацію в пристрої вибірки-зберігання, у разі ємності збереження $C_{ЗБ}$ = 1 нф дорівнює 5 мкс.

1.3.2.4. Аналого-цифровий перетворювач

АЦП виконує перетворення аналогової напруги у 8-розрядний паралельний ДК, що вводиться у МК.

В якості АЦП може бути використано, наприклад, мікросхему AD571 [1; 2]. На рис. 1.8 наведено позначення цієї мікросхеми на електричних схемах і показується як АЦП пов'язаний з іншими частинами локальної мікроконтролерної системи керування.

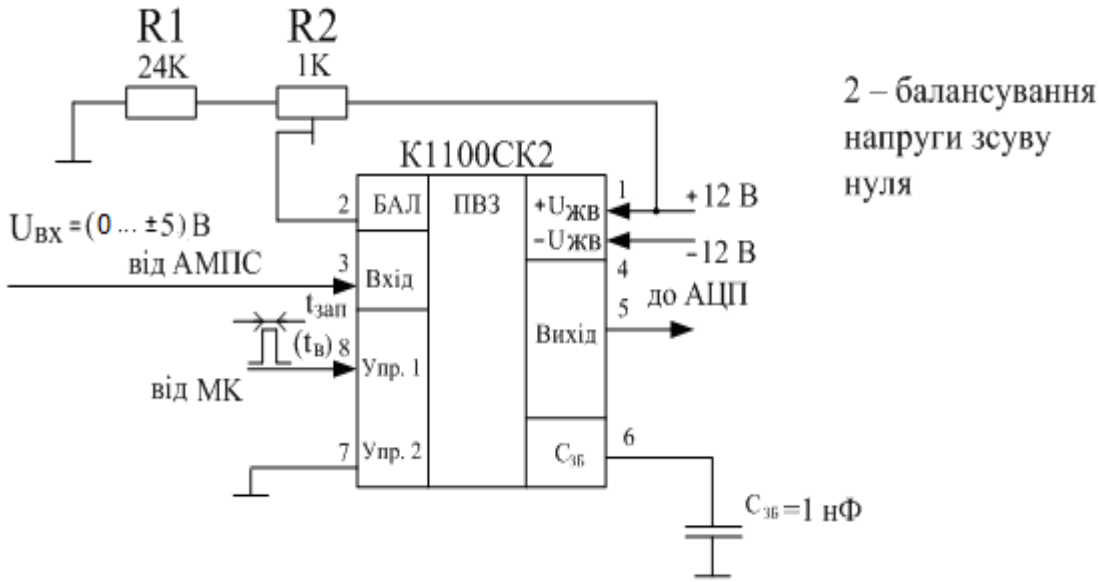


Рис. 1.7. Схема включення пристрою вибірки-зберігання

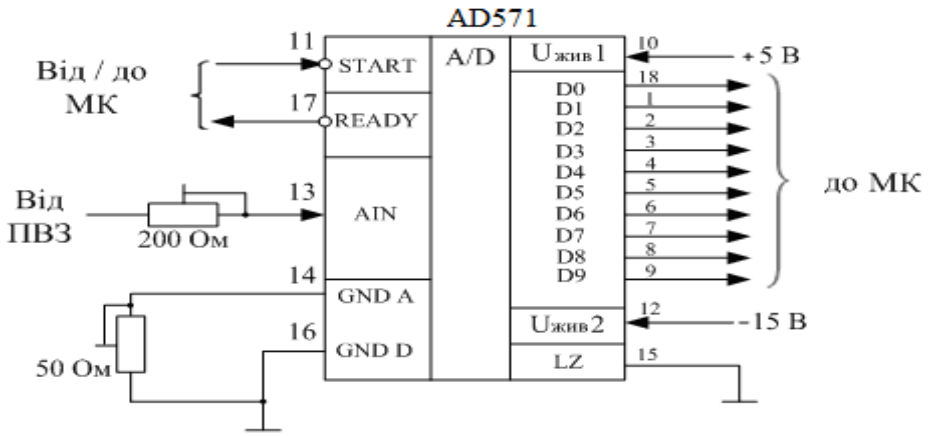


Рис. 1.8. Схема включення АЦП

Особливості взаємодії та програмування АЦП і МК пояснюють часові діаграми роботи АЦП (рис. 1.9).

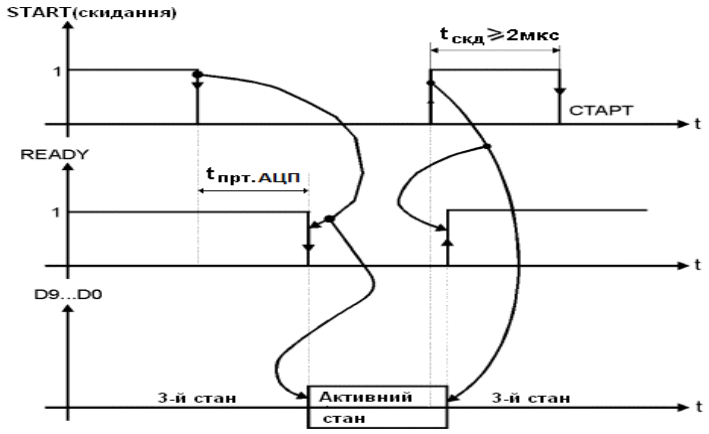


Рис. 1.9. Часові діаграми роботи АЦП

Запуск АЦП відбувається під час перемикання сигналу на вході START (СТАРТ) з логічної одиниці в нуль.

Під час перетворення на виході READY (ГОТОВНІСТЬ) є логічна одиниця, а шина даних знаходиться у третьому (високоімпедансному) стані.

Після закінчення перетворення вихідні сигнали на виводах даних D0…D9 переходять в активний стан, а сигнал на виході READY перемикається з одиниці в нуль. Отримавши сигнал готовності, МК зчитує (вводить) дані від АЦП і переводить сигнал на вході START у стан одиниці на час, не менший ніж 2 мікросекунди. У відповідь сигнал на виході READY перемикається з нуля в одиницю, а вихідні сигнали на виводах даних D0…D9 переходять у неактивний (третій) стан. Цим здійснюється «скидання» АЦП, після якого може вироблятися наступний «запуск» АЦП і т. д.

1.3.2.5. Ведений мікроконтролер

Ведений МК отримує інформацію про поточний стан об’єкта керування, робить порівняння цього стану із заданим, виробляє сигнали

розузгодження, реалізує необхідні закони керування і видає керувальні впливи на виконавчі елементи.

Як ведений МК може бути використано, наприклад, мікросхему AT89C51 [2]. На рис. 1.10 наведено позначення цієї мікросхеми на електричних схемах і пояснюється як вона пов’язана з іншими частинами локальної мікроконтролерної системи керування.

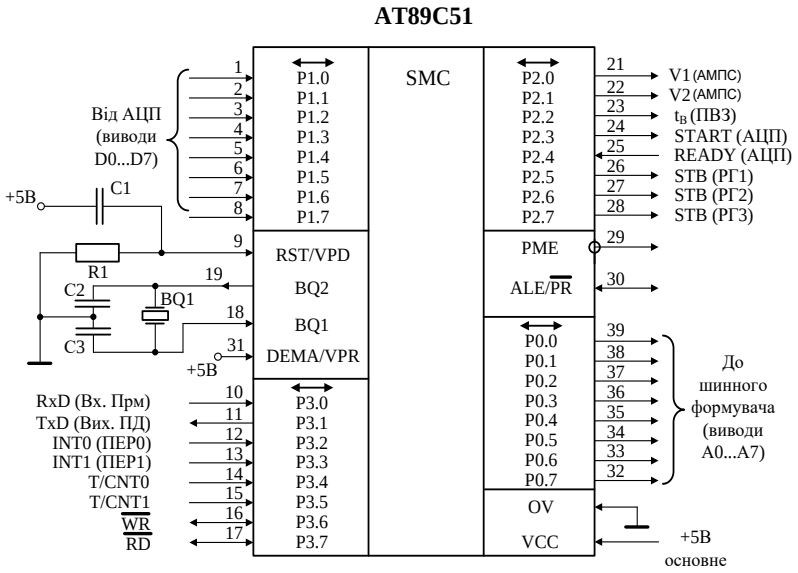


Рис. 1.10. Схема включення МК

За допомогою ланцюжка C1, R1 виробляється сигнал автоматичного «скидання» МК після включення напруги живлення.

1.3.2.6. Шинний формувач

Шинний формувач застосовується для підвищення навантажувальної здатності виводів МК, що для порту P0 дорівнює двом входам логічного елемента типу транзисторно-транзисторна логіка з переходами Шоткі, або комплементарний метал-оксид-напівпровідник [1; 2]. Оскільки виводи порту P0 повинні підключатися до інформаційних входів трьох регістрів (див. рис. 1.5), то для підсилення сигналів використовується шинний формувач.

В якості шинного формувача може бути обрано, наприклад, мікросхему IC8286.

На рис. 1.11 наведено позначення цієї мікросхеми на електричних схемах і пояснюється, як шинний формувач пов’язаний з іншими частинами локальної мікроконтролерної системи керування.

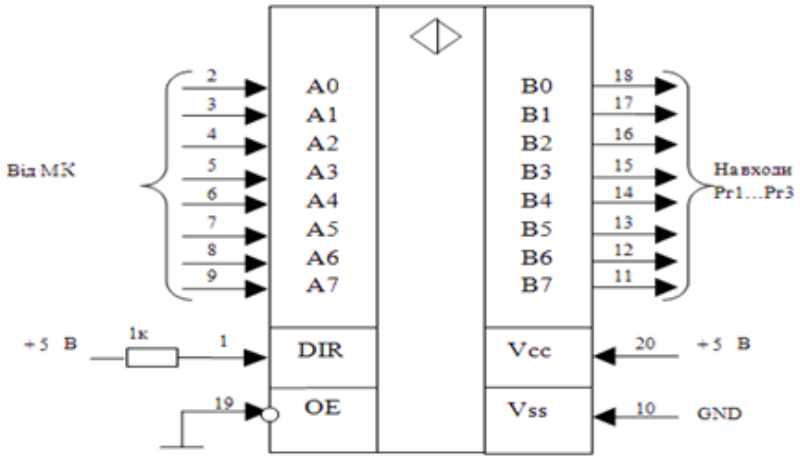


Рис. 1.11. Схема включення шинного формувача

1.3.2.7. Паралельні регістри

Паралельні регістри РГ1…РГ3 призначено для запам'ятовування значень керувальних впливів по кожному з трьох каналів. Ці впливи видаються з порту МК у паралельному двійковому коді і супроводжуються стробуючим сигналом (STB), що записує сформований керувальний вплив у необхідний регістр.

Вміст регістрів залишається незмінним до нового запису, що ініціюється подачею на відповідний вхід регістра стробуючого імпульсу. Регістри РГ1…РГ3 виконують функцію демультиплексора керувальних сигналів, які виводяться через відповідний паралельний порт МК.

В якості регістрів може бути використано, наприклад, мікросхему SN74ALS374. На рис. 1.12 наведено позначення цієї мікросхеми на електричних схемах і пояснюється, як регістри пов'язані з іншими частинами локальної мікроконтролерної системи керування.

1.3.2.8. Схеми узгодження рівнів

Схеми узгодження рівнів (СУР1…СУР3) необхідно застосовувати в тих випадках, коли мінімальні рівні напруг логічної одиниці, що з'являються на виходах регістрів, не відповідають діапазону вхідних напруг логічної одиниці ЦАП, який виконано, наприклад, на мікросхемі AD7520, та якщо останній живиться напругою: +15 В.

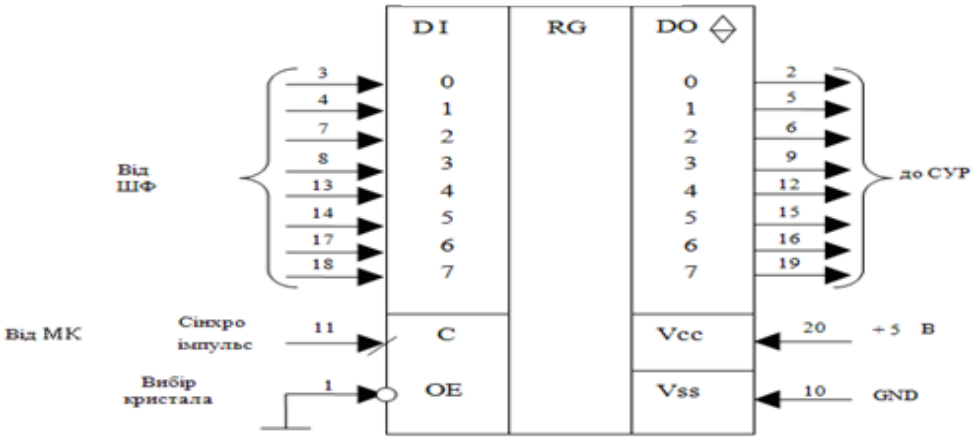


Рис.1.12. Схема включення регістра

Схеми узгодження рівнів не здійснюють жодних логічних перетворень і містять виходи з відкритим колектором/стоком [1], які через зовнішні резистори підключаються до напруги живлення, значення якої визначається необхідними величинами рівнів вхідних напруг логічної одиниці ЦАП (у нашому прикладі від 4,5 до 15 В).

В якості схеми узгодження рівнів може бути використано, наприклад, мікросхему 74LS07. На рис. 1.13 наведено позначення цієї мікросхеми на електричних схемах і пояснюється, як схеми узгодження рівнів пов’язані з іншими частинами локальної мікроконтролерної системи керування.

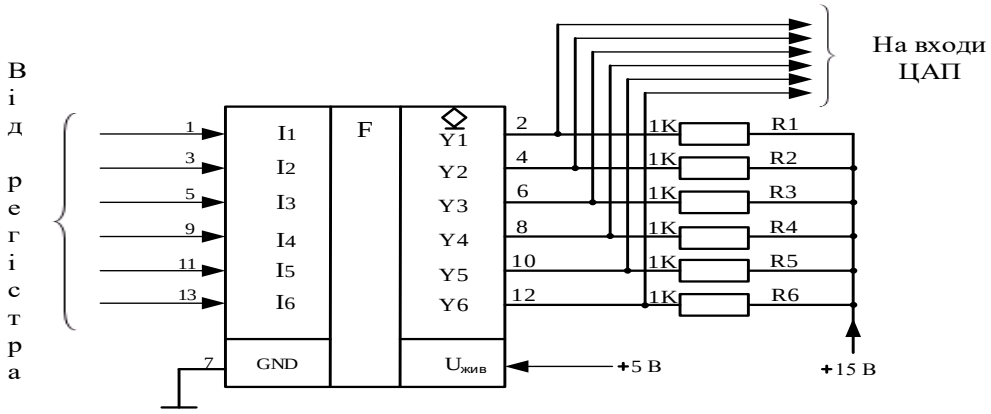


Рис. 1.13. Включення схеми узгодження рівнів

Подібних мікросхем у розглянутому прикладі (рис. 1.5) потрібно чотири, оскільки одна мікросхема включає шість повторювачів з відкритим колектором, а загальна кількість логічних сигналів, що вимагають перетворення рівнів, становить 3 · 8 = 24.

1.3.2.9. Цифро-аналогові перетворювачі

Цифро-аналогові перетворювачі (ЦАП1…ЦАП3) здійснюють перетворення цифрових керувальних сигналів, які формує МК, в аналогові

керувальні впливи, що відпрацьовуються аналоговими виконавчими елементами (АВЕ1…АВЕ3).

Як приклад, у роботі використано мікросхему AD7520, яка реалізує ЦАП на основі резистивної матриці *R*-2*R* з підсумовуванням струмів [1; 2].

Схему включення ЦАП показано на рис. 1.14.

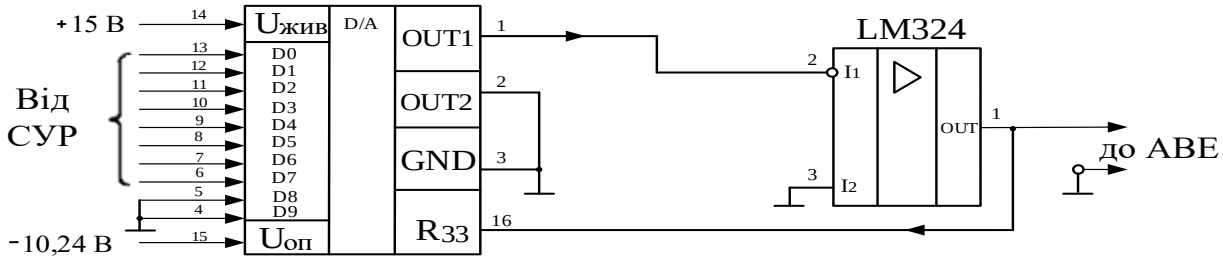


Рис. 1.14. Схема включення ЦАП

Коефіцієнт передачі цього ЦАП – $К_{пер}$ = 10 Мв/МЗР, діапазон зміни вихідної аналогової напруги у разі 8-розрядного вхідного двійкового коду, який подається на входи D0…D7, становить $U_{вих.ан}$ = 0…2,55 В. У якості операційного підсилювача може бути застосовано мікросхему LM324.

1.3.3. Модульна структура мікропроцесорної системи

1.3.3.1. Структурна схема системи

На рис. 1.15 наведено модульну структуру МПС, яку побудовано на основі 8-розрядного МП, наприклад, і8080 [2].

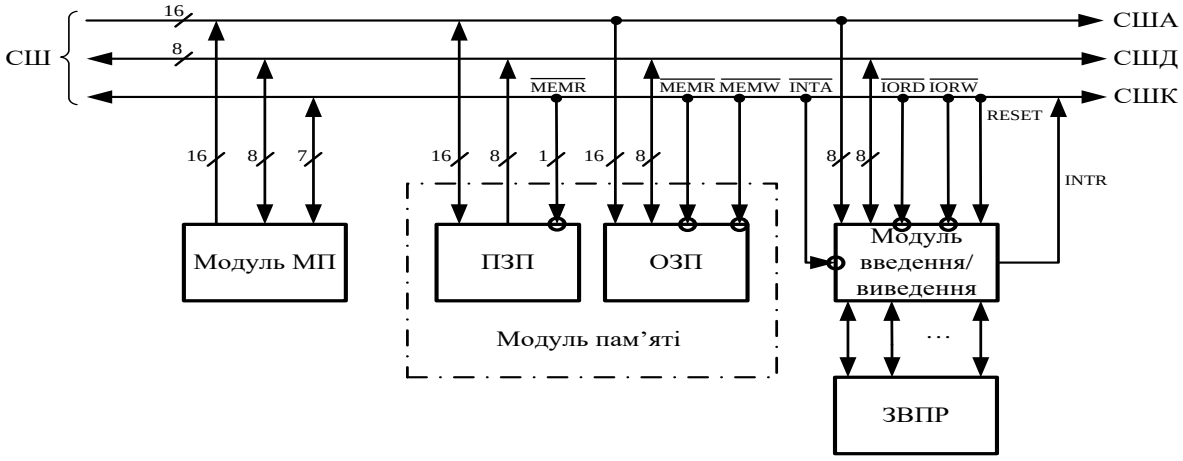


Рис. 1.15. Модульна структура МПС

До МПС належать модулі: МП, пам'яті та введення/виведення.

Окрім цього, до МПС можуть входити модулі: переривань, таймерів, АЦП, ЦАП, ПДП тощо.

1.3.3.2. Модуль 16-розрядного мікропроцесора

На рис. 1.16 наведено функціональну схему модуля МП на основі 16-розрядного МП, наприклад, МП і8086. Цей модуль вирішує такі задачі: розподіл (демультиплексування) шини адреси/даних, буферування шини адреси і шини даних, а також формування системних керувальних сигналів для блоків пам'яті і зовнішніх пристроїв.

Перша задача вирішується, наприклад, за допомогою інтегральних мікросхем і8282, що виконують функції регістрів-защіпок.

Зображені на рис. 1.16 два 8-бітових регістри і8282 запам'ятовують 15 молодших розрядів адреси (А0…А14). Це дозволяє адресувати 2^{15} = 32 Кбайт комірок пам'яті.

Оскільки сигнал $\overline{BHE}$ формується в тому ж інтервалі часу, що і адресні сигнали, то він також запам'ятовується у защіпці.

Для доступу до пам'яті максимальної ємності 1 Мбайт необхідно під'єднати ще один регістр-защіпку, на який подаються старші розряди адреси, що залишилися: виходи AD15, A16/S3…A19/S6 МП.

Друга задача вирішується за допомогою двоспрямованих 8-бітових шинних формувачів: і8286, які підсилюють сигнали для передачі на системну шину даних.

Третя задача може бути вирішена, наприклад, за допомогою комбінаційних логічних схем, які на основі сигналів: $\overline{RD}$, $\overline{WR}$ та M / $\overline{IO}$, що виробляються МП, формують необхідні керувальні сигнали: $\overline{MEMW}$, $\overline{MEMR}$, $\overline{IORD}$ та $\overline{IOWR}$.

Ці сигнали доцільно сформувати, якщо в системі використовується адресний простір введення/виведення, ізольований від простору пам'яті.

Названі вище сигнали керують запам'ятовувальними та зовнішніми пристроями подібно тому, як це робиться в системах, побудованих на основі МП і8080 [2].

Якщо в МПС введення/виведення організовано із відображенням на пам'ять, то сигнал M/ $\overline{IO}$ не використовується і на зовнішні пристрої подаються сигнали $\overline{RD}$ та $\overline{WR}$ після підсилення.

Шинні формувачі та регістри повинні забезпечувати три стани вихідних сигналів, щоб можна було організувати прямий доступ до пам'яті, якщо він використовується в системі. Виходи регістрів та шинних формувачів переводяться у третій стан за зняттям сигналу $\overline{\text{ДОЗВІЛ ШИН}}$ ($\overline{BUSEN}$) – переходом цього сигналу з нуля в одиницю. Цей сигнал надходить від контролера прямого доступу до пам'яті. Після цього останній виконує функцію формування керувальних сигналів.

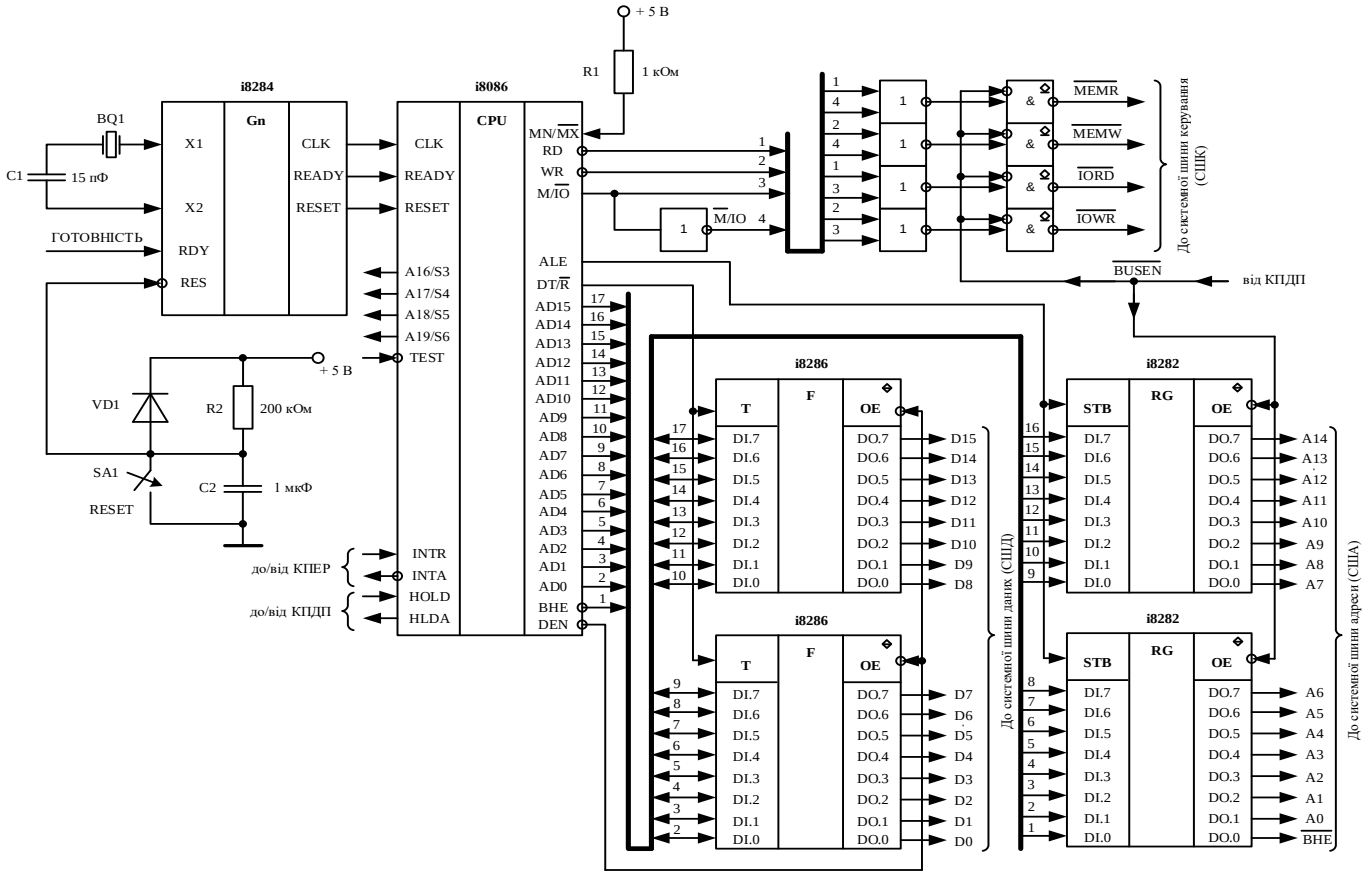


Рис. 1.16. Функціональна схема модуля МП i8086

Керувальні сигнали для системної шини керування знімаються з логічних елементів з відкритим колектором/стоком. Це зроблено для того, щоб поєднати ці сигнали з аналогічними сигналами, які формуються контролером прямого доступу до пам'яті. У цьому разі кожен з аналогічних керувальних сигналів від модуля МП і контролера прямого доступу до пам'яті поєднуються та за допомогою зовнішнього резистора підключаються до напруги живлення: +5 В. Таке з'єднання у літературі зветься монтажне «АБО» для нульових вихідних сигналів, або монтажне «І» для одиничних вихідних сигналів [1]. Якщо захоплення шин і обмін даними за режимом ПДП не передбачено, то необхідності в такому поєднанні немає.

1.3.3.3. Модуль пам'яті

Модуль пам'яті призначено для збереження програм і даних. До нього належить постійний запам'ятовувальний пристрій і ОЗП.

Постійний запам'ятовувальний пристрій призначений для збереження програм і даних, що беруть участь в операціях. Інформація до постійного запам'ятовувального пристрою заноситься на етапі виготовлення або перепрограмування системи. Постійний запам'ятовувальний пристрій – це енергонезалежна пам'ять.

ОЗП призначений для збереження даних, що обробляються, та для збереження програм, які зазвичай змінюються.

Більш детально види ОЗП та постійних запам'ятовувальних пристрів розглянуто у розд. 2.

1.3.3.4. Модуль введення/виведення

На технічні характеристики МПСК вагомо впливають засоби обміну інформацією між обчислювальним ядром і різноманітними периферійними пристроями. Ці засоби створюють підсистему введення/виведення інформації (інтерфейс), який містить у собі програмні та апаратні засоби.

Модуль введення/виведення забезпечує взаємозв'язок МП із зовнішніми пристроями. В ньому, наприклад, можуть міститися: паралельний програмований інтерфейс й послідовний програмований інтерфейс – універсальний синхроний/асинхронний приймач/передавач.

Окрім розглянутих вище трьох основних модулів: МП, пам'яті та введення/виведення до МПС можуть належати: контролер переривань, контролер прямого доступу до пам'яті, таймер, АЦП, ЦАП і т. ін. Ці модулі будуть розглянуто нижче у відповідних розділах.

Більш детально модуль введення/виведення розглянуто у розд. 6.

1.3.3.5. Системна шина

Окремі модулі МПС об’єднуються між собою внутрішньосистемним інтерфейсом і взаємодіють за адресним принципом – усі підлеглі пристрої та їх складові частини мають адреси, що не повторюються, за якими до них звертаються пристрої, які виконують функції керування.

Внутрішньосистемний інтерфейс реалізується найчастіше на основі єдиної системної шини, якою передаються адреси, дані, команди та сигнали керування. У цьому разі для передачі даних і команд використовується системна шина даних. Адреси передаються окремою системною шиною адреси, яка керується МП.

Системна шина керування призначена для обміну керувальними сигналами між МП, пам’яттю або зовнішнім пристроєм.

Основні сигнали системної шини керування:

– $\overline{MEMR}$ ($\overline{ЧТП}$) – читання з пам’яті;

– $\overline{MEMW}$ ($\overline{ЗПП}$) – запис у пам’ять;

– $\overline{IORD}$ ($\overline{ЧТВВ}$) – читання введення/виведення;

– $\overline{IOWR}$ ($\overline{ЗПВВ}$) – запис введення/виведення;

– $INTR$ – сигнал переривання до МП;

– $\overline{INTA}$ ($\overline{ППЕР}$) – підтвердження переривання (із МП до контролера переривань).

Процесор обробляє інформацію трьох типів: дані, адреси та команди програми. Над даними виконуються арифметичні та логічні операції, реалізовані процесором. Обробка адреси визначається способом збереження і доступу до даних і команд, і також ґрунтується на виконанні обчислювальних операцій.

Обробка команд складається з перетворення коду операції команди у послідовність керувальних впливів (мікрооперацій) відповідно до алгоритму виконання команди в МП. Кожна мікрооперація (або їх сукупність – мікрокоманда) виконується у фіксовані інтервали часу, а вся команда – за термін повного циклу (командного циклу). Мікрокоманди утворюють мікропрограму. Таким чином, обробка команд складається в їх представленні (інтерпретації) у формі мікропрограм. Під керуванням мікропрограми виконується обробка даних і адрес, а також керування іншими пристроями МПС через внутрішньосистемний інтерфейс.

1.3.3.6. Підключення мікропроцесора до системної шини мікропроцесорної системи

У разі використання МП, наприклад, i8080, i8086 використовується архітектура з трьома системними шинами: системна шина адреси, системна шина даних і системна шина керування (рис. 1.17).

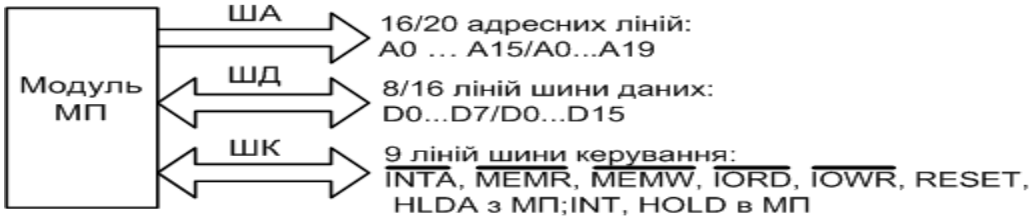


Рис. 1.17. Структурна схема системи з трьома шинами для МП i8080 або i8086

Системна шина адреси є однонапрямною, тому сигнали передаються лише від зовнішніх виводів МП. Системна шина адреси в МПС пов'язана з безліччю адресних входів, що підключені паралельно, тому сумарний струм для адресних ліній МПС може за величиною перевершувати припустиме значення струму вихідних ліній МП. Отже, для збільшення навантажувальної здатності вихідних адресних ліній МП їх необхідно буферизувати.

Для цього може використовуватися формувач шини адреси – мікросхема, виходи якої мають три стани, що дозволяє відключати шину адреси від МП у режимі прямого доступу до пам'яті.

Під час підключення системної шини даних до МП необхідно враховувати можливість перевантаження виводів МП і те, що системна шина даних є двонапрямною. Передача даних може проводитися в обох напрямках. Однак у кожний момент часу дані передаються лише в одному напрямку. Тому системна шина даних підключається до МП через двонаправлений буфер шини даних (БШД), функції якого може виконувати, наприклад, мікросхема i8286.

Системна шина керування є фактично однонапрямною, тому за одними лініями шини сигнали надходять у МП, а за другими від нього, тому необхідності організації двостороннього обміну за одними і тими самими лініями немає. Системна шина керування підключається до МП через формувач керувальних сигналів, який виробляє, шість основних сигналів: $\overline{\text{INTA}}$, $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IORD}}$, $\overline{\text{IOWR}}$, INTR. Формувач керувальних сигналів може бути виконано на твердій логіці, або з використанням спеціалізованого системного контролера типу, наприклад, intel 8228, що виконує також функцію шинного формувача.

1.4. Структурні схеми мікропроцесорів та мікроконтролерів

1.4.1. Структурна схема «інтелоподібного» 8-розрядного мікропроцесора

Структурну схему типового 8-розрядного МП, наприклад, і8080, зображено на рис. 1.18 [2].

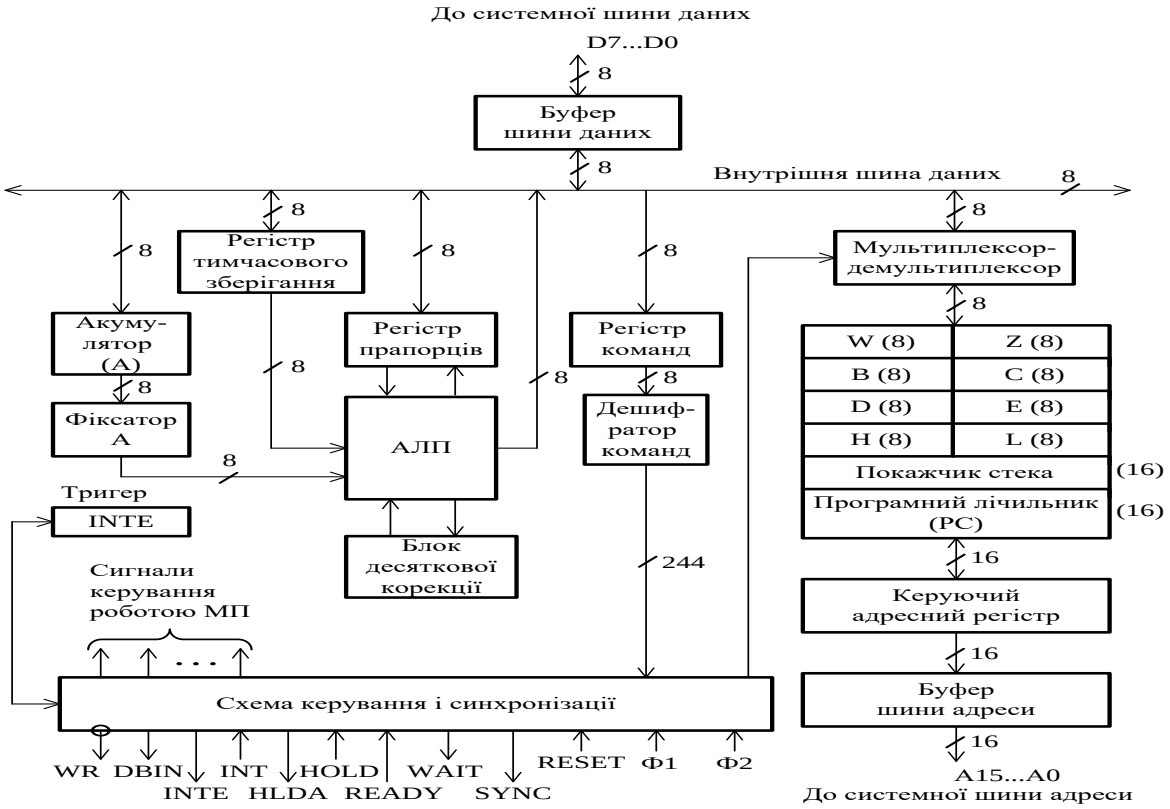


Рис. 1.18. Структурна схема МП і8080

МП і8080 є 8-розрядним монолітним МП, виготовленим у корпусі з 40 виводами за *n*-канальною метал-окисел-напівпровідниковою технологією. МП має 78 основних команд та частоту синхронізації: 0,5…4 МГц.

Нижче описано основні вузли МП.

Арифметико-логічний пристрій

Арифметико-логічний пристрій є центральним вузлом кожного МП і призначений для виконання арифметичних, логічних операцій, а також операцій зсуву. Вихідні дані для виконання операцій можуть передаватися з акумулятора і регістрів загального призначення або через БШД з пам’яті, причому один з операндів міститься в акумуляторі. Результат виконаної операції повертається в акумулятор. Вбудований у МП блок десяткової корекції дозволяє виконувати операцію додавання чисел у запакованому BCD-форматі [2].

Регістри

МП має декілька регістрів, які виконують наведені нижче функції.

Акумулятор (А) – регістр, який є найуніверсальнішим і таким, що часто використовується в командах. У ньому завжди міститься один з операндів, що беруть участь в арифметичних, логічних операціях або операціях зсуву, а також результат операцій, що виконуються в арифметико-логічному пристрої. Обмін інформацією із зовнішніми пристроями в цьому МП виконується тільки через акумулятор, що є його певним недоліком.

Регістри загального призначення (РЗП) – призначені для забезпечення швидкого доступу до операндів, що в них зберігаються, оскільки знаходяться всередині МП і фактично є швидкодіючим надоперативним запам'ятовувальним пристроєм. Вони позначаються буквами B, C, D, E, H, L і, як видно з рис. 1.18, об'єднані попарно BC, DE, HL, що дозволяє обробляти операнди завдовжки як 8 біт, так і 16 біт. Остання можливість використовується, якщо необхідно організувати збереження адреси пам'яті або виконати обчислення над двобайтовими операндами.

Програмний лічильник (ПЛ, англ. PC) – 16-розрядний регістр, призначений для збереження адреси поточної виконуваної програмою команди.

Регістр команд (РК) – регістр, призначений для зберігання коду операції поточної виконуваної команди. Код операції завжди міститься в першому байті машинного коду команди. Програмно недоступний.

Регістр стану (РС) (прапорців) – регістр, що містить прапорці – тригери, значення яких залежить від результату виконання деяких команд в арифметико-логічному пристрої.

У наведеному МП використовується п'ять прапорців, а інші три біти 8-бітного регістра прапорців не використовуються. Формат регістра стану наведено на рис. 1.19.

$7_р$	6	5	4	3	2	1	$0_р$
S	Z	0	AC	0	P	1	C

Рис. 1.19. Регістр стану (регістр прапорців) МП i8080

Нижче наведено опис прапорців:

– S – прапорець знака, встановлюється в одиницю, якщо результат виконання команди має від'ємне значення. Прапорець дорівнює старшому розряду коду результату;

– Z – прапорець нуля, який встановлюється в одиницю, якщо після виконання операції усі розряди акумулятора мають нульове значення;

– C – прапорець перенесення/запозичення, встановлюється в одиницю, якщо у разі виконання операції додавання відбулося перенесення з 7-го розряду

в уявний 8-ий або у разі віднімання відбулася позика одиниці з уявного 8-го розряду;

– AC – прапорець допоміжного перенесення, встановлюється в одиницю, якщо під час виконання команди відбулося перенесення з 3-го в 4-й розряд або позика з 4-го в 3-й розряд. Цей прапорець використовується спеціальною командою – DAA під час корекції результату додавання чисел у запакованому BCD-форматі;

– P – прапорець парності, встановлюється в одиницю, якщо результат виконання операції містить парну кількість одиниць.

Покажчик стека (SP) – 16-розрядний регістр, який призначений для адресації комірок стекової пам'яті. Для цього МП у ньому зберігається адреса верхівки стека, останньої комірки стекової пам'яті, куди записана корисна інформація. Запис чергового байта в стек ведеться з декрементом регістра-покажчика стека.

Регістри тимчасового зберігання операндів: фіксатор A, безпосередньо регістр тимчасового зберігання, регістри W і Z є програмно недоступні та використовуються для проміжного зберігання операндів і адрес під час вибірки команд з пам'яті та їх виконання.

Буферні регістри – використовуються для узгодження МП з системної шиною, програмно недоступні. БШД – двонапрямний, а буфер шини адреси – однонапрямний.

Керувальний адресний регістр складається з:

– регістра адреси пам'яті;

– схеми інкремента-декремента адреси.

Регістр адреси пам'яті – 16-розрядний, приймає і зберігає адресу від кожного 16-розрядного регістра. Вихід регістра адреси через схему інкремента-декремента адреси зв'язано з буфером шини адреси та зі входом кожного 16-розрядного регістра, який містить адресу.

Схема інкремента-декремента адреси пов'язана з виходом регістра адреси пам'яті та призначена для зміни вмісту регістра адреси пам'яті на ±1 (для формування поточних адрес пам'яті програм і стека). У процесі вибірки коду операції команди з пам'яті вміст регістрів стану і регістрів адреси збігається. Після декодування коду операції поточної команди у програмний лічильник записується адреса коду операції наступної команди, а вміст регістра адреси змінюється згідно з виконуваною командою.

Блок десяткової корекції призначений для корекції результату після додавання чисел у запакованому BCD-форматі. Для корекції використовується команда DAA.

Мультиплексор забезпечує введення інформації з декількох паралельних каналів в один послідовний під час передачі даних, наприклад, з блоку регістра загального призначення на внутрішню шину даних.

Демультиплексор служить для прийому інформації з одного послідовного каналу, наприклад з внутрішньої шини даних, і виведення її на один з паралельних каналів, наприклад в один з регістрів загального призначення.

Дешифратор команд призначений для декодування коду операції команди, який знаходиться у першому байті команди. За результатом декодування виробляється потрібна послідовність сигналів керування, що призводить до зчитування наступних байтів команди для її виконання.

Схема керування і синхронізації – є однією з найважливіших вузлів МП, що забезпечує правильну послідовність подій у МП. Після зчитування команди з пам’яті та її декодування пристрій керування генерує послідовність сигналів, необхідних для виконання команди. Робота схеми програмується під час виготовлення МП.

Схема керування містить маленький МП всередині МП. Ще однієї з основних функцій схеми керування є організація зв’язку МП із зовнішніми пристроями та системною шиною.

1.4.2. Структура 16-розрядного «інтелоподібного» мікропроцесора

В якості прикладу 16-розрядного МП у роботі розглянуто мікросхему i8086 (рис. 1.20).

Під час розробки цього МП було застосовано нові порівняно з МП i8080 архітектурні рішення, до яких, зокрема, належить розділення функцій сполучення з шиною та виконання команд. Відповідно до цього структуру МП i8086 можна умовно розділити на дві частини: виконавчий блок і блок сполучення з шиною (рис. 1.20) [2].

Виконавчий блок призначений для зберігання даних, що обробляються, і виконання операцій над ними. Він включає вісім 16-розрядних регістрів загального призначення, 16-розрядний арифметико-логічний пристрій, регістр прапорців (ознак) (англ. RF), регістр команд, пристрій керування і синхронізації.

Регістри загального призначення належать до надоперативних запам’ятовувальних пристроїв. Вони повністю доступні програмісту, можуть використовуватися довільно, але в багатьох командах вони мають специфічне призначення. Згідно з цим регістри загального призначення поділяються на дві групи.

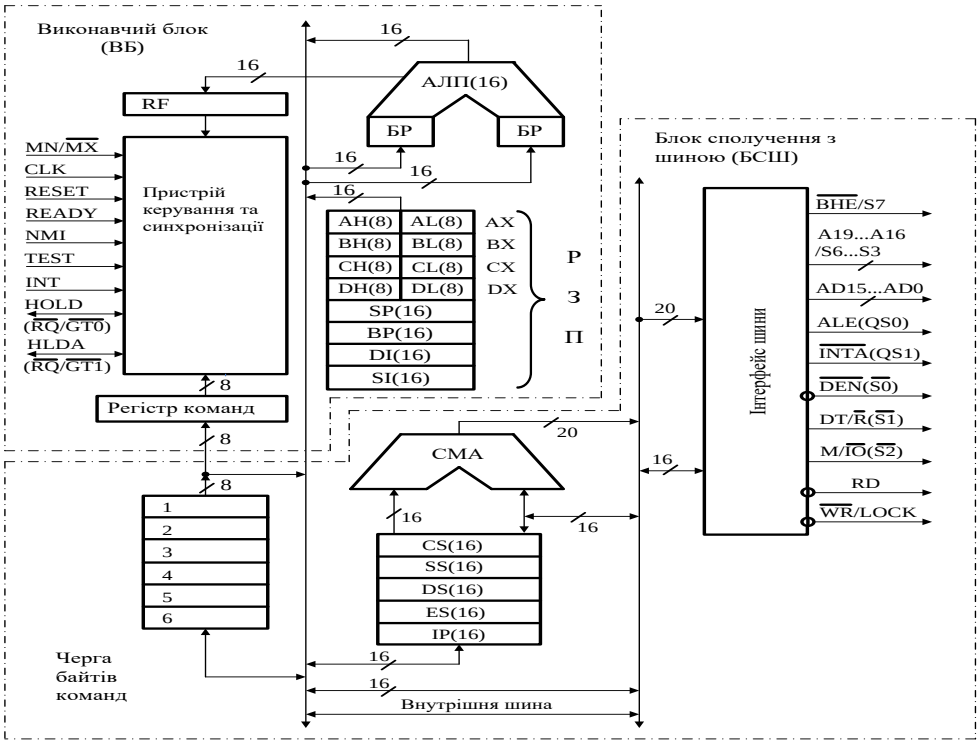


Рис. 1.20. Структура МП i8086

Перша група регістрів загального призначення: *AX, BX, CX, DX* в арифметичних і логічних командах може використовуватися для зберігання операндів і результатів виконання операцій.

Переважно для цієї мети застосовується регістр *AX* – акумулятор.

У регістрі BX – регістрі бази зазвичай зберігається початкова адреса структури даних, що обробляється, яка розміщена у сегменті даних DS.

РГ CX у низці команд використовується як лічильник під час організації циклічних обчислень.

Регістр даних DX у командах множення і ділення зберігає половину 32-розрядних операндів, а в командах введення/виведення може містити адресу зовнішніх пристроїв. У процесорі передбачена можливість роботи з 8-розрядними «половинками» названих регістрів.

Регістр AX розпадається на два: регістр молодшого байта акумулятора *AL* і регістр старшого байта акумулятора *AH*.

Аналогічно *регістр BX* – розпадається на два регістри *BL* і *BH*, регістр *CX* – на два регістри *CL* і *CH*, а *регістр DX* – на регістри *DL* і *DH*.

Друга група регістрів загального призначення складається з двох 16-розрядних регістрів-покажчиків: *SP* і *BP* та двох 16-розрядних індексних регістрів: *SI* і *DI*.

Регістри-покажчики використовуються для роботи зі стековою пам’яттю (стековим сегментом пам’яті): покажчик стека *SP* містить адресу верхівки стека,

а покажчик бази *BP* зберігає адресу початкового елемента, наприклад, структури даних у стековому сегменті (англ. SS).

Індексні регістри використовуються для доступу до елементів рядків даних. Під час читання елемента рядка-джерела регістр індексу *SI* зберігає зсув адреси в сегменті даних *DS*, а під час запису елемента рядка-приймача регістр індексу *DI* містить зсув адреси в екстра-сегменті *ES*.

Арифметико-логічний пристрій призначений для виконання арифметичних, логічних операцій та операцій зсуву над 8- та 16-розрядними операндами – даними, що беруть участь в операціях. Операнди надходять в арифметико-логічний пристрій з регістрів загального призначення і/або пам'яті. Результат операції повертається в регістр загального призначення чи в пам'ять.

Арифметико-логічний пристрій окрім схем для виконання операцій містить два програмно недоступних буферних регістри для тимчасового збереження операндів.

Регістр прапорців (англ. RF) (рис. 1.21) містить 9 ознак (прапорців), що характеризують стан програми, виконуваної МП, та керують роботою МП.

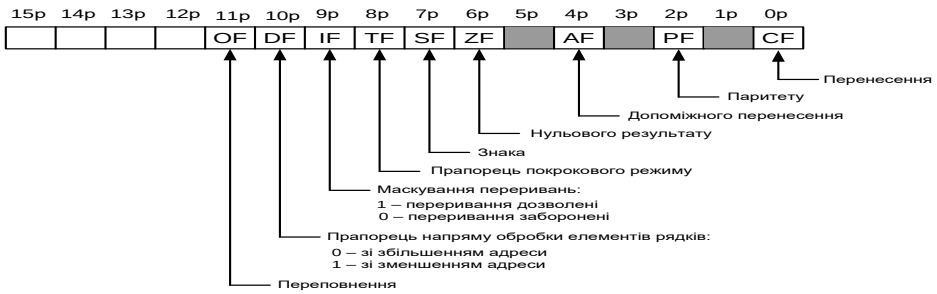


Рис. 1.21. Формат регістра прапорців МП і8086

Шість прапорців встановлюються відповідно до результату виконаної в арифметико-логічному пристрої чергової операції.

Якщо розрядність оброблюваних кодів позначити *n*, а нумерацію розрядів здійснити справа наліво від 0 до (*n*–1), то:

– *прапорець перенесення CF* встановлюється в одиницю, якщо в результаті виконання операції виникло перенесення з (*n*–1)-го розряду результату в уявний більш старший *n*-й розряд або була потрібна позика з уявного *n*-го розряду;

– *прапорець паритету PF* доповнює код результату до непарного числа одиниць, тому встановлюється в одиницю, якщо кількість одиниць у коді результату парна;

– *прапорець допоміжного перенесення AF* встановлюється в одиницю, якщо під час виконання операції виникло перенесення з 3-го розряду результату в 4-й або виконалася позика з 4-го розряду в 3-й. Прапорець

AF зазвичай використовується для виконання команд десяткової корекції результату під час виконання операцій над BCD-числами;

– *прапорець нульового результату ZF* приймає одиничне значення, якщо всі розряди результату нульові. Якщо хоча б один розряд результату відмінний від нуля, то прапорець *ZF* скидається в нуль;
– *прапорець знака SF* встановлюється рівним старшому (n–1)-му розряду коду результату. Під час виконання операцій над числами зі знаком прапорець *SF* відповідає знаку результату. Якщо результат більший за нуль або дорівнює нулю (додатний), то *SF* скидається в нуль, якщо результат менший за нуль (від'ємний), то *SF* буде встановлено в одиницю;
– *прапорець переповнення OF* встановлюється в одиницю, якщо в результаті виконання арифметичних операцій над числами зі знаком відбувається переповнення розрядної сітки, тому результат не укладається в діапазон, виділений n-розрядному числу зі знаком (перенесення з (n–2)-го розряду в (n–1)-й не збігається з перенесенням з (n–1)-го розряду в уявний n-й). Наприклад, якщо n = 8, то діапазон чисел зі знаком: –128…+127, якщо n = 16, то діапазон чисел зі знаком: –32768…+32767.

Три прапорці, що залишилися, характеризують стан пристрою керування МП:

– *прапорець напряму передачі DF* визначає спосіб зміни адрес джерел/приймачів операндів у командах роботи з рядками даних. Якщо *DF* = 0, то адреси збільшуються (автоінкремент адреси), якщо *DF* = 1, то адреси зменшуються (декремент адреси);
– *прапорець дозволу переривання IF*, що скидається в нуль, забороняє (маскує) обробку зовнішнього переривання, а встановлений в одиницю дозволяє його;
– *прапорець покрокового режиму TF*, що знаходиться в одиничному стані, задає спеціальний режим виконання програми, в якому після обробки кожної команди генерується програмне переривання.

У регістр команд вміщується код операції команди, що підлягає виконанню (програмно недоступний). Цей код декодується дешифратором, що дозволяє ідентифікувати тип виконуваної команди.

Пристрій керування і синхронізації забезпечує функціонування окремих вузлів і МП у цілому. Він аналізує вхідні і генерує вихідні керувальні сигнали, керує обміном інформацією в самому МП та виконанням команд. Після декодування коду операції чергової виконуваної команди пристрій керування

формує послідовність керувальних сигналів, що забезпечують виконання цієї команди.

Блок сполучення із шиною забезпечує обмін інформацією між МП і зовнішніми відносно нього приймачами або джерелами даних (пам'ять, пристрій введення/виведення і т. ін.).

До *блока сполучення із шиною* належать: група сегментних регістрів, регістр адреси команди, черга байтів команд, суматор адреси та інтерфейс шини.

Сегментні регістри призначено для збереження початкових (базових) логічних адрес сегментів пам'яті. Процесор містить чотири 16-розрядних сегментних регістрів, що відповідає чотирьом поточним сегментам пам'яті:

- *регістр сегмента команд CS:* визначає сегмент пам'яті, в якому зберігається виконувана програма (містить логічну адресу початку поточного сегмента коду);
- *регістр сегмента даних DS:* задає сегмент пам'яті, що містить оброблювані дані (містить логічну адресу початку поточного сегмента даних);
- *регістр сегмента стека SS:* визначає сегмент пам'яті, доступ до елементів якого можливий за допомогою стекової адресації (містить логічну адресу початку поточного сегмента стека);
- *регістр екстра-сегмента ES:* задає додатковий сегмент, в якому також можуть зберігатися оброблювані дані (містить логічну адресу початку поточного додаткового сегмента даних).

Регістр адреси команд IP (покажчик команд) зберігає 16-розрядну адресу (зміщення) чергової команди, що повинна витягатися із сегмента команд, тому є зсувом відносно початку поточного сегмента команд.

Шість 8-розрядних регістрів черги команд складають внутрішню проміжну пам'ять програми зі швидким доступом, яку призначено для збереження послідовності команд, що виконуються.

Суматор адреси призначено для формування 20-розрядної фізичної адреси зовнішньої пам'яті. Для цього використовуються дві 16-розрядні логічні адреси: початку сегмента та зміщення у сегменті (зсуву відносно початку сегмента).

Інтерфейс шини забезпечує обмін інформацією із системною шиною і формує керувальні сигнали.

У МП i8086 передбачена асинхронна паралельна робота виконавчого блока та блока сполучення з шиною. У процесі виконання програми блок сполучення з шиною вибирає з пам'яті послідовність команд і розміщує їх у черзі байтів команд (до шести байт). Після завершення виконання попередньої операції у виконавчий блок чергова команда вибирається з молодших регістрів

черги і направляється на обробку у виконавчий блок. Виконавчий блок виконує запропоновану операцію, а блок сполучення із шиною завантажує чергу новою командою чи командами, залежно від кількості регістрів черги, що звільнилися. Якщо виконавчому блоку потрібно прочитати дані з пам'яті або записати їх у пам'ять, то процес вибірки команд переривається і блок сполучення з шиною виконує цикл читання чи запису даних. Після цього відновлюється вибірка команд, що продовжується до заповнення черги. Таке перекриття етапів вибірки і виконання команд значно зменшує час читання команд із сумарного часу виконання програми, підвищуючи тим самим продуктивність МП. Якщо виконавчий блок виконує команду, що передає керування іншій комірці пам'яті команд (наприклад, команди CALL, JMP), то стара черга команд зникає та блок сполучення з шиною знову встановлює чергу команд, починаючи заповнювати її з нового значення. У більшості випадків виконавчий блок може відразу одержувати команду з черги, а не чекати, доки вона буде вибрана з пам'яті.

1.4.3. Структура 8-розрядного «інтелоподібного» мікроконтролера

Структурну схему одного з перших 8-розрядних «інтелоподібних» МК сімейства MCS-51 – AT89C51 наведено на рис. 1.22 [2].

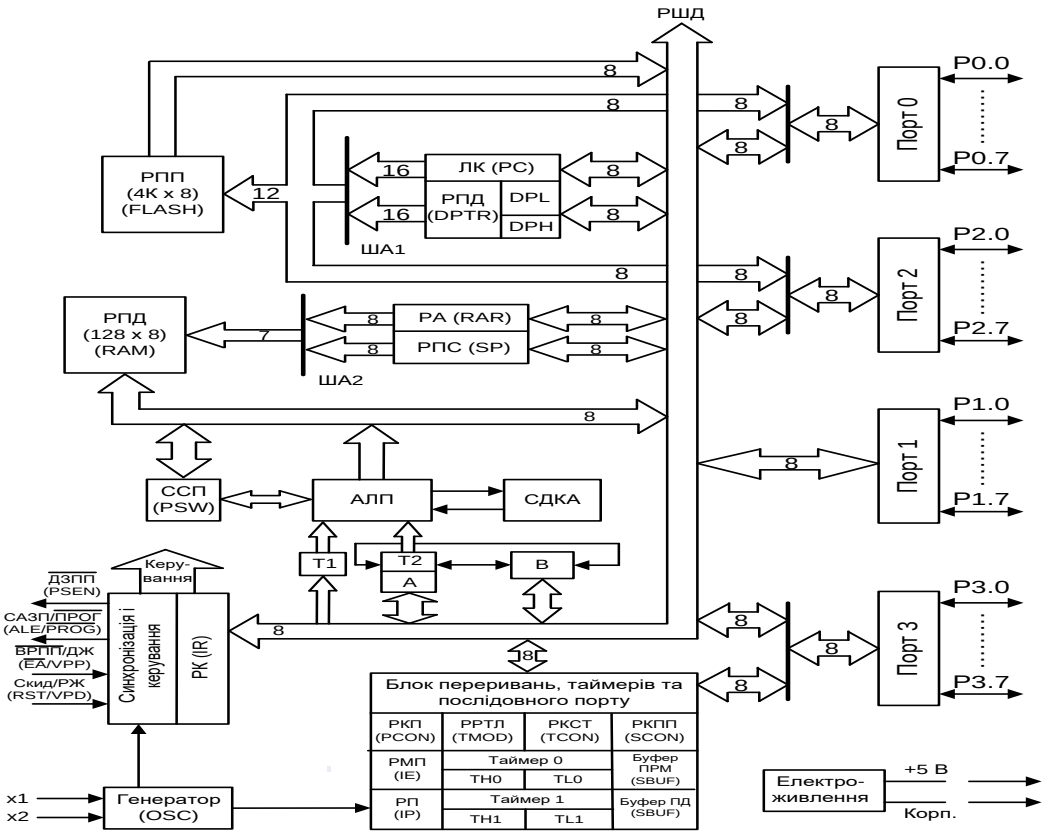


Рис. 1.22. Структурна схема МК AT89C51

МК складається з таких основних функціональних вузлів: блока керування і синхронізації; блока арифметико-логічного пристрою (англ. Arithmetic-Logic Unit); резидентної (внутрішньої) пам’яті даних (англ. Resident Data Memory) об’ємом 128 байт; резидентної (внутрішньої) пам’яті програм (англ. Resident Program Memory) об’ємом 4 Кбайт; блока переривань (англ. Interrupt System), таймерів (англ. Timer); послідовного порту (англ. Serial Port); чотирьох паралельних портів введення/виведення (англ. Parallel Port), які програмуються; схеми десяткової корекції акумулятора; внутрішнього генератора тактових імпульсів (англ. Oscillator); резидентної (внутрішньої) шини даних (англ. Resident Data Bus) і групи регістрів:

- A – акумулятор;
- B – регістр розширення акумулятора;
- T1, T2 – регістри тимчасового збереження операндів;
- РСП (PSW) – регістр стану програми (регістр прапорців);
- РК (IR) – регістр команд;
- ЛК (PC) – лічильник команд (програмний лічильник);
- РГПД (DPTR) – регістр-покажчик даних, що складається з двох частин: молодшої – DPL і старшої – DPH;
- РПС (SP) – регістр-покажчик стека;
- РА (RAR) – регістр адреси;
- РРТЛ (TMOD) – регістр режимів таймерів/лічильників;
- РКСТ (TCON) – регістр керування-статусу таймерів/лічильників;
- РКПП (SCON) – регістр керування приймачем/передавачем послідовного порту;
- SBUF – буфер приймача або передавача послідовного порту;
- РМП (англ. IE) – регістр масок переривань;
- РПП (англ. IP) – регістр пріоритетів переривань;
- РКП (PCON) – регістр керування потужністю (споживанням енергії від джерела живлення).

Опис окремих вузлів МК наведено у [2].

1.4.4. Порівняльна характеристика «інтелоподібних» мікропроцесорів і мікроконтролерів

Порівняємо характеристики «інтелоподібних» МП і МК.

Спільні риси: до структури обох пристроїв належать (рис. 1.18, 1.20, 1.22):

- арифметико-логічний пристрій;
- регістри тимчасового збереження операндів (програмно недоступні, на структурі МК позначені T1, T2);

– один з основних регістрів – акумулятор;
– регістр прапорців (ознак), на структурі МК позначено слово стану програми (англ. PSW);
– схема десяткової корекції акумулятора після додавання двійково-десяткових кодів (BCD-кодів) у запакованому форматі;
– лічильник команд (англ. PC);
– регістр-покажчик стека (англ. SP);
– схема керування та синхронізації;
– внутрішня 8-розрядна шина даних;
– регістр команд (англ. IP) – програмно недоступний.

Відмінні риси: структура МК (рис. 1.22) додатково містить:

– резидентну пам’ять даних. Для МК типу AT89C51 її об’єм становить 128 комірок пам’яті завдовжки 8 біт, тобто 128 байт;
– резидентну пам’ять програм. Для МК типу AT89C51 має об’єм 4 Кбайт (FLASH);
– чотири паралельних 8-розрядних програмованих порти введення/виведення (P0, P1, P2, P3);
– регістр розширення акумулятора B (використовується під час виконання команд множення і ділення, а також як 8-розрядний регістр загального призначення);
– послідовний програмований асинхронний інтерфейс (для нього в режимі «альтернативних функцій» використовується дві лінії порту 3: P3.0 і P3.1);
– два 16-розрядних багаторежимних таймери/лічильники T/CNT0; T/CNT1, які можуть виконувати функції програмованих таймерів або лічильників зовнішніх подій. У цьому разі імпульси, що ідентифікують зовнішні події, надходять через дві лінії порту 3 (P3.4; P3.5), який використовується в режимі «альтернативних функцій»;
– 16-розрядний регістр-покажчик даних (англ. DPTR). Складається зі старшого байта – DPH і молодшого – DPL. Може використовуватися як 16-розрядний або як два незалежних 8-розрядних регістри. Може зберігати 16-бітову адресу під час звертання до пам’яті програм або зовнішньої пам’яті даних;
– внутрішній тактовий генератор, частота якого задається за допомогою зовнішнього кварцового резонатора;
– систему переривань – для AT89C51 з п’ятьма векторами (адресами підпрограм) і двома рівнями пріоритетів. Два переривання можуть викликатися зовнішніми джерелами та надходять через дві лінії

порту 3 (P3.2, P3.3) у режимі «альтернативних функцій». Інші три переривання є внутрішніми: два від переповнення таймерів/лічильників і одне від завершення передачі або прийому послідовним каналом;

– регістри керування, що керують роботою:
  1) регістр режимів таймерів/лічильників (англ. TMOD);
  2) регістр керування-статусу таймерів/лічильників (англ.TCON);
  3) регістр керування приймачем-передавачем (англ. SCON);
  4) буфер ПД (англ. SBUF) – буфер передавача; буфер приймача (англ. SBUF). Обидва буфери мають однакове ім’я під час програмного звертання до них – SBUF. Якщо команда записує дані в буфер, то звертання надходить до буфера ПД, а якщо – читає, то до буфера приймача;
  5) регістр масок переривань (англ. IE);
  6) регістр пріоритетів переривань (англ. IP);
  7) регістр керування потужністю – один з регістрів керування (англ. PCON), керує процесом споживання енергії (потужності) від джерела живлення і дозволяє переводити МК у режими: «холостого ходу» і «зниженого споживання енергії». Один з бітів цього регістра – PCON.7 (SMOD) дозволяє програмно змінювати швидкість передачі послідовним каналом.

1.4.5. Опис 8-розрядних мікроконтролерів сімейства AVR

1.4.5.1. Загальна характеристика мікроконтролерів

Розглянутий вище МК AT89C51 є представником сімейства МК-51 (MCS-51). МК цього сімейства були створені на основі існуючих на момент їх створення «інтелоподібних» МП. Тому вони мали відповідну архітектуру та значний час широко застосовувалися у МПС різного призначення.

Однак одним з недоліків цих МК є наявність регістра-акумулятора, який обов’язково брав участь під час виконання частини команд. Це частково зменшувало час виконання робочої програми.

Цей недолік було усунуто в AVR-мікроконтролерах, які нині широко застосовуються серед 8-розрядних МК.

«Традиційні» AVR-мікроконтролери, що починали свій розвиток з серії Classic (позначення AT90S), зараз розділилися на три сімейства: Tiny (позначення AT Tiny), Mega (позначення AT Mega) і Xmega (позначення AT XMega).

Розробники МК AVR поклопоталися про інженерів, що застосовують цю платформу, зробивши МК різних серій повивідно сумісними [3; 4]. Таке рішення забезпечує свободу вибору кристала і можливість подальшого переходу між різними серіями.

Слід зазначити, що для AVR-мікроконтролерів також характерна програмна сумісність «знизу вверх», що забезпечується збереженням найменувань регістрів спеціального призначення в різних серіях і фактично єдиним набором команд.

Простежуючи розвиток AVR-мікроконтролерів, можна звернути увагу на тенденції нарощування складності системи, збільшення кількості периферійних блоків, апаратну підтримку різних інтерфейсів, розширення кількості представників у сімействах.

Слід зазначити постійне поліпшення характеристик енергоспоживання AVR-мікроконтролерів від покоління до покоління.

На додаток зауважимо, що платформа AVR, як і раніше, продовжує привертати увагу розробників збалансованим за функціональністю і вартістю набором МК у поєднанні з доступністю програмних і апаратних засобів підтримки розробок для них.

Нині широко використовуються МК сімейства Mega [3; 4], які є 8-бітними МК, призначеними для використання у вбудованих системах. Вони виготовляються за малоспоживаючою КМОН-технологією, яка у поєднанні з удосконаленою RISC-архітектурою дозволяє досягти найкращого співвідношення вартість/швидкодія/енергоспоживання.

AVR-мікроконтролери нині широко застосовуються у платах Arduino, які використовуються у вбудованих системах [6].

1.4.5.2. Структура ядра AVR-мікроконтролерів

Ядро AVR-мікроконтролерів, наприклад сімейства Mega виконано за вдосконаленою RISC-архітектурою (enhanced RISC) (рис. 1.23), в якій використовується низка рішень, спрямованих на підвищення швидкодії МК.

Арифметико-логічний пристрій, що виконує усі обчислення, підключений безпосередньо до 32 робочих регістрів, що об’єднані в єдиний регістровий файл. Завдяки цьому арифметико-логічний пристрій може виконувати одну операцію (читання вмісту регістрів, виконання операції і запис результату назад у регістровий файл) за один такт. Крім того, фактично кожна з команд (за винятком команд, в яких одним з операндів є 16-бітна адреса) займає одну 16-розрядну комірку пам’яті програм.

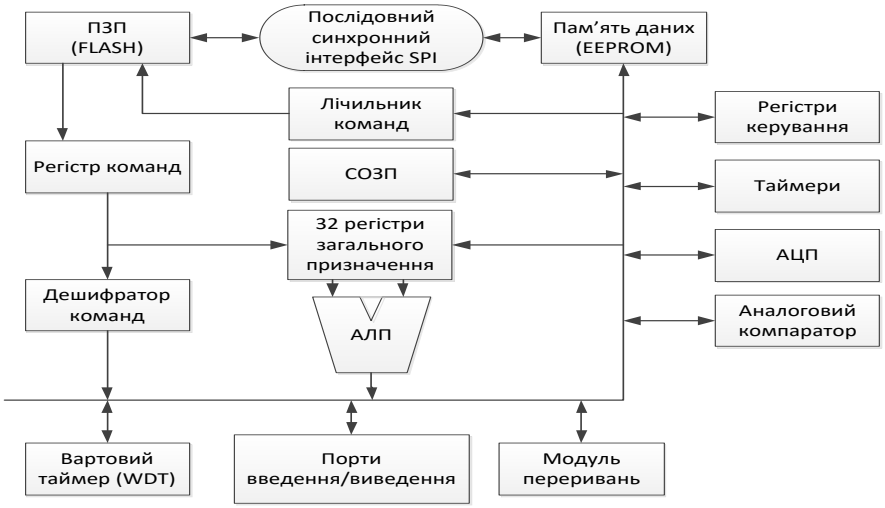


Рис. 1.23. Структура ядра AVR-мікроконтролерів

У [4] наведено деякі структури МК сімейства Mega.

Наприклад, до моделей AT Mega1281x/2561x крім ядра належать:

– 7 портів введення/виведення (порти A…F – 8-бітні, порт G – 6-бітний);
– два 8-бітних (T0, T2) та чотири 16-бітних (T1, T3, T4, T5) таймери/лічильники;
– 6 каналів широтно-імпульсної модуляції;
– 8-канальний 10-бітний АЦП;
– два інтерфейсні модулі USART, які можуть працювати в режимі SPI;
– по одному інтерфейсному модулю SPI та TWI;
– інтерфейс JTAG.

До моделей AT Mega640x/1280x/2560x належать:

– 11 портів введення/виведення (порти A…F, H, J…L – 8-бітні, порт G – 6-бітний);
– два 8-бітних (T0, T2) та чотири 16-бітних (T1, T3, T4, T5) таймери/лічильники;
– 6 каналів широтно-імпульсної модуляції;
– 8-канальний 10-бітний АЦП;
– два інтерфейсні модулі USART, які можуть працювати в режимі SPI;
– по одному інтерфейсному модулю SPI та TWI;
– інтерфейс JTAG.

Контрольні запитання та завдання

1. Що таке МП?
2. Що таке мікро-ЕОМ? Що називається мікропроцесорним комплектом?
3. Що таке системна шина керування? Наведіть приклад керувальних сигналів.

4. Опишіть обмін даними між МП та об’єктом керування через паралельний програмований інтерфейс.
5. Від чого залежить кількість адресованих комірок пам’яті МП або МК?
6. Скільки адресних виводів містить МК AT89C51? Який обсяг пам’яті він дозволяє адресувати?
7. За значеннями яких параметрів можна оцінювати швидкодію МП?
8. Що таке система числення? Назвіть існуючи системи числення.
9. Що таке позиційна система числення? Наведіть приклад позиційної системи числення.
10. Запишіть беззнакове число:
    - 01011101_2 у десятковій системі числення;
    - 11001110_2 у шістнадцятковій системі числення;
    - 723_{10} у двійковій системі числення;
    - 596_{10} у шістнадцятковій системі числення;
    - $8C_{16}$ у двійковій системі числення;
    - $F5A_{16}$ у десятковій системі числення;
    - 735_{10} у запакованому BCD-форматі;
    - 571_{10} у незапакованому BCD-форматі.
11. Запишіть у прямому, зворотному та додатковому кодах число зі знаком: -11_{10}.
12. Знайдіть суму та різницю беззнакових чисел: 01001101_2 і 00000111_2.
13. Яку роль виконує регістр-акумулятор у МП i8080?
14. Що являє собою лічильник команд?
15. Для чого призначені мультиплексор та демультиплексор у МП i8080?
16. Назвіть компоненти, що належать до виконавчого блоку МП i8086.
17. Яку розрядність має регістр BH у МП i8086?
18. Що зберігається в регістрі SI у МП i8086?
19. Назвіть умови встановлення прапорця CF в одиницю у МП i8080.
20. Для чого призначені сегментні регістри в МП i8086?
21. Дайте визначення командного циклу в МП i8080.
22. Що потрібно знати для визначення повного часу виконання команди?
23. Що являє собою арифметико-логічний пристрій у МП та МК?
24. Для чого служить лічильник команд (PC) у МП та МК?
25. Що зберігається у постійному запам’ятовувальному пристрої?
26. Назвіть основні задачі, які виникають під час розробки модуля МП на основі МП i8086.
27. Для чого призначений ОЗП?
28. Назвіть основні сигнали системної шини керування.
29. Для чого використовується аналоговий мультиплексор у МП?

30. Що таке АЦП? Для чого призначені АЦП?
31. Для чого призначені паралельні регістри РГ1…РГ3 у структурі локальної мікропроцесорної системи керування?
32. Для чого використовуються схеми узгодження рівнів?
33. Що являє собою ЦАП?
34. На які три сімейства нині розділився випуск «Традиційних» AVR-мікроконтролерів?
35. Опишіть рівень сумісності AVR-мікроконтролерів різних серій.
36. Опишіть особливості «Гарвардської архітектури» МК.
37. За скільки тактів виконуються більшість команд МК?
38. Яку розрядність має одна комірка пам’яті програм МПС на основі МП?
39. Яку розрядність має одна КП програм AVR-мікроконтролерів?
40. Опишіть структуру ядра AVR-мікроконтролерів.
41. Як обчислюється тривалість одного такту?
42. Назвіть спільні риси 8-розрядних МП та МК.
43. Назвіть відмінні риси 8-розрядних МП та МК.
44. Опишіть регістр прапорців МП i8086.
45. Наведіть та опишіть структуру 8-розрядного МП.
46. Наведіть та опишіть структуру 16-розрядного МП.
47. Наведіть та опишіть структуру ядра 8-розрядного МК.
48. Чим відрізняється призначення регістра команд від лічильника команд?
49. Назвіть основний недолік «інтелоподібних» МП.
50. Поясніть для чого в структурі МП i8086 використовуються логічні елементи з відкритим колектором/стоком?
51. Яке з’єднання у літературі зветься монтажне «АБО» для нульових вихідних сигналів, або монтажне «І» для одиничних вихідних сигналів?
52. Як у МП i8086 обчислюється фізична адреса комірки пам’яті?
53. Чим відрізняються поняття: логічна адреса комірки пам’яті від фізична адреса комірки пам’яті?
54. Для чого у МПС використовуються шинні формувачі?
55. Назвіть та поясніть основні сигнали, які передаються системною шиною керування?
56. Як обчислюється об’єм пам’яті програм у МПС?

2. ОРГАНІЗАЦІЯ ПАМ’ЯТІ МІКРОПРОЦЕСОРНИХ СИСТЕМ

2.1. Особливості архітектури пам’яті мікропроцесорних систем

2.1.1. Призначення та місце модуля пам’яті в мікропроцесорних системах

Під час вивчення модульної структури МПС зазначалося, що одним з основних її модулів є пам’ять (рис. 2.1).

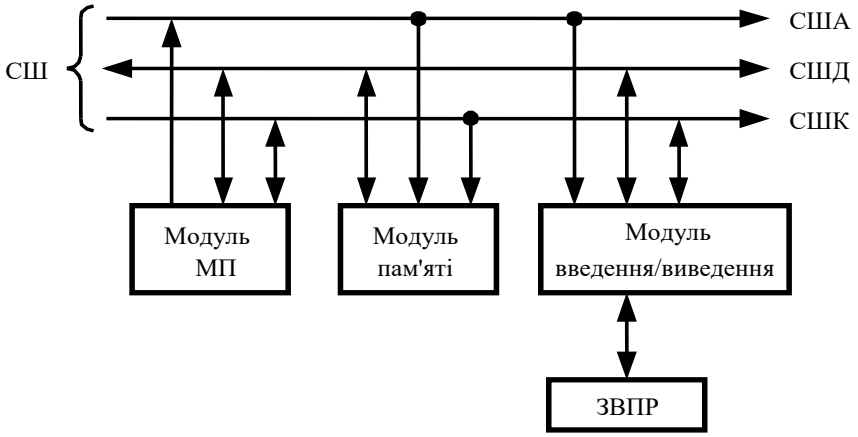


Рис. 2.1. Модульна структура МПС

У пам’яті, яку також називають запам’ятовувальним пристроєм, зберігаються програми і дані, що використовуються для керування роботою МПС [2; 3].

Пам’ять поділяється на:

– оперативний запам’ятовувальний пристрій (ОЗП);
– постійний запам’ятовувальний пристрій (ПЗП).

Керувальні програми повинні розміщуватися в енергонезалежному ПЗП, щоб зберігатися у разі вимикання живлення, тому його іноді називають пам’яттю програм. В ОЗП зберігаються дані, що беруть участь у виконанні операцій, тому його називають пам’яттю даних. Це робиться дуже умовно, оскільки в процесі роботи складної системи в ОЗП із зовнішнього носія інформації може завантажуватися керувальна програма.

Під час роботи системи ПЗП допускає головним чином читання записаної в нього інформації, а в ОЗП можна спочатку записати поточні дані, а потім їх прочитати. В сучасних МК є можливість змінювати вміст ПЗП під час роботи системи. Для цього використовується додатковий модуль та команда самопрограмування (див. підрозд. 10.4) [3].

2.1.2. Основна та зовнішня пам’ять

Пам’ять стосовно МПС зазвичай розділяють на основну і зовнішню. Кожен МП/МК може адресувати деяке число комірок пам’яті, що утворюють

єдиний адресний простір МПС, реалізованої на основі цього МП/МК. Ці комірки утворюють основну (внутрішню) пам’ять МПС.

Будь-яку комірку цієї пам’яті можна адресувати в конкретній системі для читання або для запису інформації. Наприклад, якщо системна шина адреси має кількість ліній $n = 16$, то основна пам’ять МПС може мати до $2^{16} = 65\,536$ комірок.

Основна пам’ять МПС на МК поділяється на: внутрішню пам’ять програм/даних і зовнішню пам’ять програм/даних.

Зовнішню пам’ять МПС утворюють пристрої тривалого збереження інформації, наприклад, магнітні стрічки, диски, дискети тощо.

У цьому розділі розглянуто основну пам’ять, яка використовується переважно на напівпровідникових великих інтегральних схем.

2.1.3. Пам’ять з довільним та послідовним доступом

Залежно від способу звернення до окремих комірок пам’яті розрізняють:

– пам’ять з довільним доступом;

– пам’ять з послідовним доступом.

У разі довільного доступу (вибірки) допускається будь-який порядок надходження адрес, що перебувають в адресному просторі цього процесора.

У разі адресації комірок пам’яті другого типу зміна адрес можлива тільки у визначеному порядку – збільшення чи зменшення адрес. Прикладом такого доступу є запис і читання інформації на магнітну стрічку. Основна пам’ять МПС є запам’ятовувальний пристрій із довільним доступом.

2.1.4. Енергозалежна та енергонезалежна пам’ять

Залежно від того, як впливає наявність джерела живлення на збереження інформації, пам’ять поділяється на:

– енергозалежну;

– енергонезалежну.

В енергозалежній пам’яті у разі відсутності живлення дані руйнуються, а в енергонезалежній – ні. Прикладом енергозалежної пам’яті є статичний оперативний запам’ятовувальний пристрій (англ. SRAM), а прикладом енергонезалежної – FLASH-пам’ять програм та EEPROM-пам’ять даних.

2.1.5. Статична та динамічна пам’ять

Основна пам’ять МПС будується здебільшого на метал-оксид-напівпровідникових (КМОН)-транзисторах і поділяється на статичну і динамічну. ОЗП динамічного типу зазвичай виконуються на метал-оксид-

напівпровідникових транзисторах. Особливістю цих транзисторів є дуже високий вхідний опір, що дозволяє використовувати як елемент пам'яті конденсатор, функції якого виконує вхідна паразитна ємність метал-оксид-напівпровідника транзистора. В результаті одержуємо динамічну пам'ять.

Вхідний сигнал надходить на затвор транзистора через електронний ключ [2]. У разі розмикання ключа заряд на конденсаторі зберігається, підтримуючи транзистор або у відкритому, або в закритому стані залежно від логічного рівня вхідного сигналу. Внаслідок дуже повільного розряду конденсатора через високий опір затвору рівень вхідного сигналу зберігається, тобто інформація запам'ятовується до наступного замикання ключа або до регенерації пам'яті. Вихідний сигнал знімається зі стоку метал-оксид-напівпровідника транзистора, тому конденсатор практично не шунтується навантаженням.

Однак практично не можна зберегти інформацію протягом тривалого часу. Метал-оксид-напівпровідниковий транзистор має деякий кінцевий опір, що шунтує конденсатор. Ізоляція затвору транзистора також має кінцевий опір. Тому максимальний час запам'ятовування, чи мінімальна частота регенерації, визначається ємністю конденсатора і цими двома опорами (здебільшого опором ключа транзистора). Необхідно періодично регенерувати записану у динамічну пам'ять інформацію. Регенерація може проводитися процесором програмно за допомогою спеціальних переривань основної програми чи апаратно за допомогою контролера регенерації або контролера прямого доступу до пам'яті.

Динамічну пам'ять використовують у персональних комп'ютерах для зберігання досить великих обсягів інформації.

У статичних ОЗП кожен елемент пам'яті об'ємом 1 біт являє собою симетричний тригер, складений із двох транзисторних ключів на метал-оксид-напівпровідникових транзисторах з перехресними зв'язками [2].

Для зберігання даних у МПС здебільшого використовуються статичні ОЗП та енергонезалежна EEPROM-пам'ять даних.

2.1.6. Основні характеристики пам'яті

Основними характеристиками (параметрами) пам'яті є:

- об'єм;
- розрядність;
- швидкодія;
- споживана потужність;
- габарити;
- вартість;
- технологія виготовлення та програмування.

Об’єм пам’яті вимірюється кількістю комірок, що вона містить. Розрядність відображає кількість біт інформації, що містяться в одній комірці пам’яті.

З метою зменшення споживаної потужності від джерел живлення, зменшення габаритів та вартості, сучасні великі інтегральні схеми пам’яті виконуються за метал-оксид-напівпровідниковою (КМОН)-технологією. Основним параметром, що характеризує швидкодію пам’яті, є час доступу до пам’яті.

Розрізняють час доступу у разі читання та запису. Час, необхідний для виведення інформації з пам’яті на шину даних після одержання адреси потрібної комірки, називається часом доступу у разі читання. Час доступу у разі запису – час необхідний для запису даних в область, що адресується. Час доступу сучасних напівпровідникових великих інтегральних схем запам’ятовувальних пристроїв складає одиниці наносекунд.

2.1.7. Фізична та логічна організація пам’яті

Під фізичною розуміється організація пам’яті на апаратному рівні, а під логічною – на програмному.

У МПС на основі, наприклад, 8-розрядного МП і8080 ОЗП і ПЗП логічно сполучено (для звернення до їхніх комірок використовуються ті самі команди). Фізично ОЗП і ПЗП розділено, оскільки використовують різні мікросхеми, що адресуються способом «карти пам’яті» [2].

У МПС на основі 16-розрядного МП, наприклад, і8086 логічно весь адресний простір пам’яті (1 Мбайт) поділено на сегменти, кожен з який може мати довжину від 16 до 65 536 байт, а кількість сегментів складає від 65 536 до 16.

У конкретний момент часу МП доступні тільки чотири сегменти: коду, даних, стека і додатковий сегмент даних, що називаються поточними.

Для адресації комірок пам’яті окремих сегментів використовуються дві 16-розрядні логічні адреси: перша визначає початок сегмента і зберігається в одному з чотирьох сегментних регістрів: CS, DS, SS чи ES, а друга адресує комірку в обраному сегменті (є зміщенням відносно початку сегмента).

У разі обчислення 20-розрядної фізичної адреси комірки пам’яті 16-розрядна логічна адреса початку сегмента зсувається вліво на чотири розряди, у молодші біти записуються чотири нулі та до отриманої таким чином 20-розрядної фізичної адреси початку сегмента додають другу 16-розрядну логічну адресу (зсув у сегменті) (рис. 2.2).

В якості зсуву можуть бути:

– вміст регістра-покажчика команд (IP) у разі витягування (читання)

команд із кодового сегмента;

– вміст регістра SP у разі виконання команд роботи зі стеком (PUSH/POP);
– ефективна (виконавча) адреса операнда ЕА (ВА), що обчислюється під час виконання команди залежно від способу адресації операндів.

Більш детально це питання розглянуто у підрозд. 3.3.

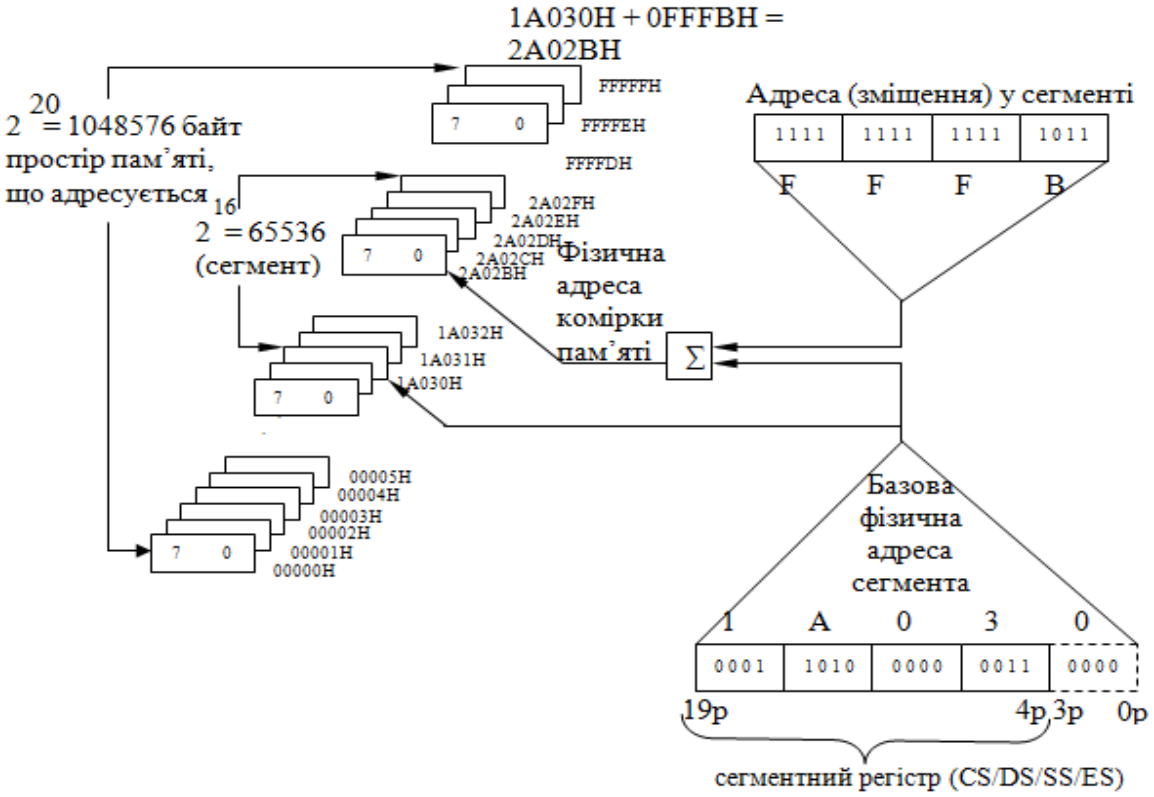


Рис. 2.2. Формування фізичної адреси комірки пам'яті

Розглянемо 8-розрядний МК, наприклад AT89C51, який містить чотири види пам'яті:

– ВПД, об'ємом 128 байт;
– ВПП, об'ємом 4 Кбайт;
– зовнішню пам'ять даних, об'ємом 64 Кбайт;
– зовнішню пам'ять програм, об'ємом 60/64 Кбайт.

Сумарний об'єм пам'яті даних становить (128 + 65536) байт, а пам'яті програм – 65 536 (64 К) байт.

Пам'ять даних і програм поділено фізично та логічно. ВПД і зовнішню пам'ять даних поділено фізично та логічно. Резидентну пам'ять програм і зовнішню пам'ять програм фізично поділено, а логічно сполучено – використовують ті самі команди. За умови необхідності резидентну пам'ять програм можна апаратно відключити і весь діапазон пам'яті програм (64 Кбайт) передати зовнішній пам'яті програм [2].

Організацію пам’яті 8-розрядного МК сімейства AVR описано у підрозд. 2.3.

2.1.8. Особливості проектування пам’яті великого об’єму

Запам’ятовувальні пристрої великих об’ємів звичайно поділяються на кілька модулів (сторінок), кожна з яких має об’єм, що обирається виходячи з можливостей реалізації на ВІС запам’ятовувальних пристроїв, і виконується у вигляді автономного модуля. Адреси комірок цих модулів можуть знаходитися в довільному місці адресного простору МПС.

Під час сторінкової адресації спочатку обирається сторінка, а потім комірка на сторінці. Сторінки обираються, наприклад за допомогою тумблерів (перемикачів), чи за допомогою дешифратора.

У першому випадку кожна сторінка конструктивно виконується у вигляді окремого модуля (плати). Плати містять перемикачі, якими задається діапазон адресного простору, що відповідає цій сторінці.

У другому випадку використовується дешифратор двійкового коду в унітарний. На вхід дешифратора подаються старші розряди адреси пам’яті, що задають номер сторінки. З виходів дешифратора знімається унітарний код, що обирає одну зі сторінок, адреса якої у двійковому коді надійшла на вхід дешифратора. Молодші розряди адреси одночасно надходять на всі сторінки. Проте активним сигналом з виходу дешифратора в кожний момент часу буде обрано лише одну зі сторінок. Виходи мікросхем пам’яті не обраних сторінок будуть знаходитися у високоімпедансному стані.

2.1.9. Призначення та організація стека

Стеком називається спеціальна область оперативної пам’яті, що здебільшого створюється і використовується під час організації та виконання підпрограм. Стек у МПС умовно поділяється на:

– апаратний;
– програмний.

Наявність виду стека залежить від типу МП або МК. У МПС, в яких стек реалізовано апаратно, відсутні команди запису в стек або читання зі стека (PUSH/POP) і стек використовується тільки для зберігання адрес повернення із підпрограм.

Більшість МПС мають програмний стек, який виконує функцію апаратного стека, а також має можливість роботи зі стеком командами PUSH/POP. Для адресації комірок стека МП/МК містять спеціальні регістри SP. До виконання команди роботи зі стеком SP адресує «верхівку» стека – останню

комірку стекової пам’яті, в яку записано інформацію, або першу вільну комірку пам’яті (залежить від типу МП/МК).

Для запису даних у стек є команда з мнемонікою PUSH, а для читання – POP. Механізм виконання цих команд у «інтелоподібних» МП дуже схожий. По-перше, стек заповнюється в бік менших адрес. По-друге, обмін зі стеком відбувається словами (2 байти). У разі запису у стек спочатку записується старший байт слова, а потім – молодший, а у разі читання – навпаки. По-третє, виконується правило обміну зі стеком – першим увійшов (був записаний) у стек – останнім вийшов (був прочитаний) зі стека, тобто стек типу FILO («FIRST IN LAST OUT»).

У 8-розрядних МП для стека виділяється спеціальна частина загального адресного простору пам’яті, яка адресується способом «карти пам’яті».

У 16-розрядних МП для стека використовується окремий сегмент, максимальний об’єм якого становить 64 Кбайт.

Виконання команд PUSH і POP у МК типу MCS-51 відрізняється від описаного вище: по-перше, стек заповнюється в бік збільшення адреси, по-друге, обмін зі стеком відбувається байтами. У МК типу AVR стек заповнюється в сторону менших адрес та обмін також відбувається байтами. Принцип «першим увійшов, останнім вийшов» у МК зберігається.

2.1.10. Режим прямого доступу до пам’яті

2.1.10.1. Загальна характеристика

Один зі способів обміну даними в МПС є режим ПДП, коли обмін даними між зовнішнім пристроєм і пам’яттю МПС відбувається без участі МП. Обміном у режимі ПДП керують зовнішні стосовно МПС електронні схеми, а не програма, що виконується МП. Ці схеми розміщуються у спеціальному пристрої, що називається контролером ПДП.

Використання режиму ПДП у МПС викликано декількома факторами. По-перше, під час використання цього режиму з’являється можливість початкового завантаження програм в основну пам’ять МПС з пристроїв введення. По-друге, що є найбільш важливим, режим ПДП забезпечує можливість обміну даними між пам’яттю МПС і зовнішнім пристроєм блоками фіксованого розміру. Причому цей обмін може відбуватися у визначеній послідовності з великою швидкістю, котра обмежена швидкодією пам’яті.

Забезпечити необхідну швидкість обміну блоків даних між пам’яттю МПС і зовнішнім пристроєм за допомогою програмно-керованого обміну за участю МП неможливо, оскільки на обмін кожним байтом між пам’яттю і зовнішнім пристроєм через МП потрібно звичайно кілька команд процесора, сумарний час виконання яких зазвичай перевищує максимальний припустимий

час на обмін одним байтом із зовнішнім пристроєм. Необхідна швидкість обміну забезпечується режимом ПДП, для реалізації якого необхідно забезпечити безпосередній зв'язок контролера ПДП і пам'яті МПС. Для цього можна було б використовувати спеціально виділені шини адоеси і даних, що зв'язують контролер ПДП із пам'яттю. Однак це призведе до значного ускладнення системи, особливо у разі підключення декількох зовнішніх пристроїв. Контролер ПДП підключається до пам'яті за допомогою системних шин МПС.

Проблема спільного використання системних шин МП і контролером ПДП найчастіше вирішується у такий спосіб: за системною шиною керування МПС від контролера ПДП на вхід HOLD МП передається керувальний сигнал «Вимога прямого доступу до пам'яті (ВПДП)». Процесор, отримавши сигнал «ВПДП», припиняє виконання програми, видає з виходу HLDA на системну шину керування керувальний сигнал контролера «Надання прямого доступу до пам'яті (НПДП)» і відключається від системної шини адреси, системної шини даних і системної шини керування.

Для відключення МП від системної шини використовується можливість переведення його виходів, які під'єднано до цих шин, у третій (високоімпедансний) стан [1]. Керувальні сигнали від контролера ПДП до системної шини керування підключаються за допомогою елементів з відкритим колектором (стоком), як це описано у пп. 1.3.3.2. З цього моменту системна шина МПС передається у розпорядження контролера ПДП. Контролер ПДП, використовуючи системну шину, обмінюється блоками даних з пам'яттю, а потім, знявши сигнал «ВПДП», повертає керування системною шиною МП.

2.1.10.2. Контролер прямого доступу до пам'яті

Типовий представник сучасного контролера ПДП – мікросхема i8237, яка виконує функцію чотириканального програмованого контролера ПДП, призначеного для організації безпосереднього зв'язку між зовнішніми пристроями та пам'яттю у МПС на основі МП i8080 та i8086 [2].

Основна функція контролера – формування адреси пам'яті та керувальних сигналів зчитування/запис пам'яті від зовнішнього пристрою. Контролер приймає запити ПДП від зовнішніх пристроїв, здійснює їх пріоритетну обробку, формує МП сигнал захоплення шин, у результаті чого системна шина МПС відключається від МП і формується послідовність адрес пам'яті та керувальних сигналів зчитування/запис. Після закінчення передачі даних контролер видає керувальний сигнал МП про закінчення обміну. Кожен з чотирьох каналів контролера ПДП містить 16-розрядний регістр, який дає змогу адресувати пам'ять об'ємом 64 Кбайт, і 14-розрядний регістр кількості циклів

обміну (лічильник циклів), який дає змогу здійснювати пересилання масивів слів даних об’ємом до 16 Кбайт.

Більш детальну інформацію про режим ПДП наведено у [2].

2.2. Організація пам’яті мікропроцесорної системи на основі мікропроцесора

2.2.1. Організація пам’яті мікропроцесорної системи на основі мікропроцесора типу i8080

8-розрядний МП, наприклад, i8080 має фон-Нейманівську організацію пам’яті [2].

Під час проектування МПС на основі цього МП використовується адресація за допомогою «карти пам’яті», в якій кожному фізичному пристрою системи (ОЗП, ПЗП) ставляться у відповідність визначені адреси.

2.2.2. Організація пам’яті мікропроцесорної системи на основі мікропроцесора типу i8086

2.2.2.1. Організація адресного простору пам’яті

На рис. 2.3 наведено склад та організацію адресного простору пам’яті МП i8086.

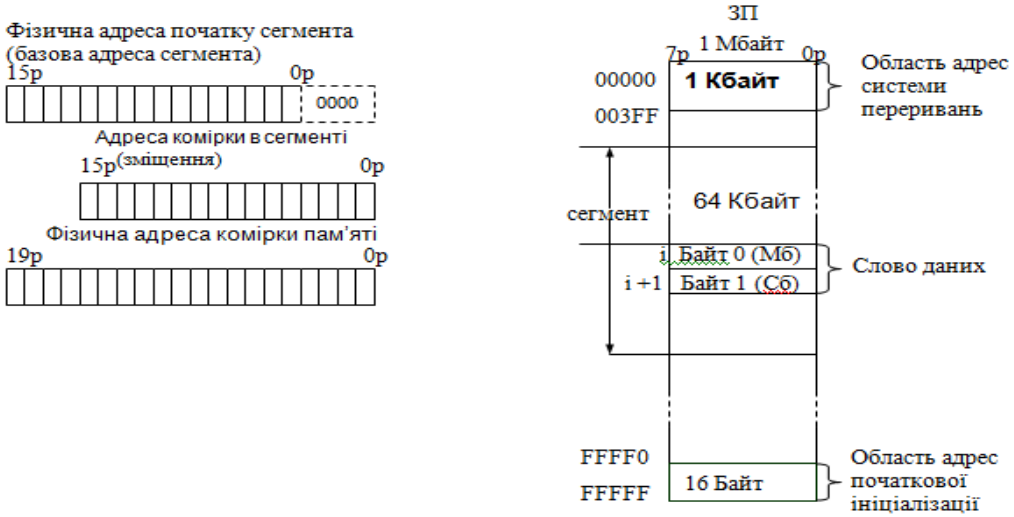


Рис. 2.3. Організація адресного простору пам’яті МП i8086

МП i8086 має 20-розрядну адресну шину, що дозволяє адресувати 2^{20} = 1 048 576 комірок пам’яті [2]. Розрядність комірки прийнято рівною 8 біт (1 байт), тобто МП доступно 1 Мбайт пам’яті. Схемні рішення, що будуть розглянуті нижче, забезпечують обмін з пам’яттю байтами або словами (16-розрядними даними) з автоматичним вибором необхідного формату. 20-розрядна фізична адреса може змінюватися від 00000H до FFFFFH.

Дві частини простору пам’яті, які розміщено в молодших адресах 00000…003FFH (1 Кбайт) і старших адресах FFFF0…FFFFFH (16 байт), використовуються відповідно для обслуговування переривань і початкової ініціалізації системи. Слова даних у пам’яті розміщуються в порядку збільшення номерів байтів: молодші байти за меншими адресами, старші байти – за більшими адресами.

Логічно весь адресний простір пам’яті розбито на сегменти, кожен з яких може мати довжину від 16 до 65 536 байт. Відповідно адресний простір МП (1 Мбайт) може бути розділено на кількість сегментів від 65 536 до 16.

Для адресації конкретної комірки в сегменті необхідна 16-розрядна адреса (зміщення відносно початку сегмента, що змінюється від 0000H до FFFFH).

Розміщення сегмента в пам’яті задається 20-розрядною фізичною адресою початку сегмента (базовою адресою сегмента), молодша шістнадцяткова цифра якого повинна дорівнювати 0H. Тобто базова адреса сегмента має вигляд XXXX0H, де X – будь-яка шістнадцяткова цифра (див. вище рис. 2.2).

Шістнадцять старших значущих розрядів 20-розрядної фізичної базової адреси сегмента зберігаються в сегментних регістрах та називаються ***логічною адресою початку сегмента*** (логічною базовою адресою сегмента, логічною адресою сегмента).

У конкретний момент часу МП доступні тільки чотири сегменти пам’яті, що називаються ***поточними***:

– поточний сегмент команд (адреса початку сегмента визначається регістром CS);
– поточний сегмент даних (адреса початку сегмента задається регістром DS),
– поточний сегмент стека (адреса початку сегмента визначається регістром SS);
– поточний додатковий (екстра) сегмент даних (адреса початку сегмента задається регістром ES).

Сегменти можуть збігатися, якщо збігається вміст сегментних регістрів, можуть перекриватися, якщо вміст регістрів відрізняється менш чим на 1000H, чи бути цілком різними, якщо коди в регістрах сегментів відрізняються більш ніж на 1000H.

Формування 20-розрядної фізичної адреси комірки пам’яті за двома 16-розрядними логічними адресами: початку сегмента та зсуву в сегменті ілюструють наведені вище рис. 2.2, 2.3.

У розглянутому на рис. 2.2 прикладі у сегментному регістрі міститься код 1A03H. Адреса комірки у сегменті (зміщення) дорівнює FFFBH. Повна 20-розрядна фізична адреса комірки утворюється в МП як сума вмісту сегментного регістра, що зсунуто вліво на 4 двійкових розряди та доповнено з правого боку кодом 0000B, і адреси в сегменті (зміщення):

1A030H + 0FFFBH = 2A02BH.

Таким чином, у прикладі обирається КП з фізичною адресою 2A02BH.

Сегментна організація пам'яті має свої переваги та недоліки. До її переваг належить модульність програм (у них чітко виділені області даних, стека і власне коди команд), просте переміщення програм у просторі пам'яті (що є важливим у мультипрограмному середовищі), просте перемикання з однієї програми на іншу. Один з недоліків сегментної організації пам'яті полягає у труднощах маніпуляції фізичними адресами, наприклад, у разі порівняння двох адрес нерівність логічних адрес не свідчить про нерівність фізичних адрес.

2.2.2.2. Вибір типу сегмента пам'яті під час обчислення фізичної адреси

Оскільки в кожен момент часу програмі доступні чотири поточних сегментних регістри: CS, DS, SS чи ES, то має бути встановлено відповідність між джерелом адреси в сегменті (зміщенням) і типом сегмента.

У разі відсутності додаткових вказівок прийняті наступні призначення сегментних регістрів (рис. 2.4), що задають правила, відповідно до яких МП для обчислення фізичної адреси комірки пам'яті обирає сегментний регістр за відомим джерелом зміщення в сегменті.

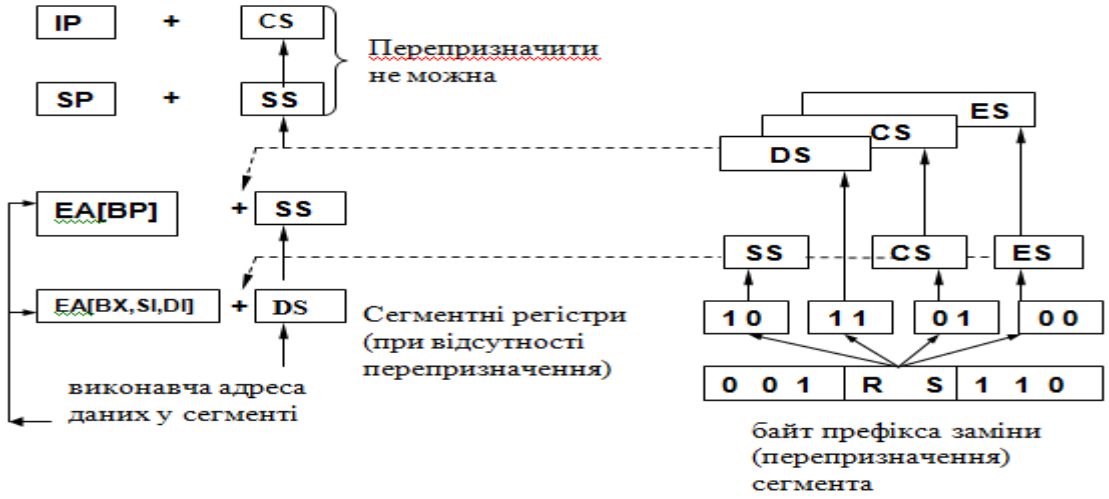


Рис. 2.4. Вибір типу сегмента пам'яті під час обчислення фізичної адреси

Під час читання команд із пам'яті зміщення витягається з регістра-покажчика команд (програмного лічильника) IP. У цьому випадку в разі обчислення фізичної адреси використовується сегментний регістр CS.

Якщо відбувається звернення до стека командами PUSH чи POP, то значення зміщення в сегменті залежить від вмісту регістра SP, а положення стекового сегмента в пам’яті визначається вмістом сегментного регістра SS.

Під час звернення до даних, які розміщено у поточних сегментах даних чи стека, МП формує виконавчу (ефективну) 16-розрядну адресу операнда ефективної адреси, що є зміщенням відносно початку обраного сегмента.

Ефективна адреса являє собою ціле беззнакове число. Значення ефективної адреси залежно від зазначеного у машинному коді команди способу адресації операнда може визначатися вмістом одного з регістрів BX, BP, SI, DI, їхніми комбінаціями, та/або безпосередніми значеннями.

Якщо для обчислення ефективної адреси використовувався регістр-покажчик бази BP, то отримана ефективна адреса є зміщенням у сегменті стека, положення якого в пам’яті визначається вмістом сегментного регістра SS.

Якщо для обчислення ефективної адреси використовувалися регістри BX, SI, DI, або ефективна адреса дорівнює 16-розрядному значенню, яке входить у машинний код команди, то ефективна адреса є зміщенням у сегменті даних, положення якого в пам’яті визначається вмістом сегментного регістра DS.

Перші два призначення: CS:IP і SS:SP змінити неможливо.

Перепризначити можна лише сегментні регістри, а також сегменти пам’яті під час розрахунку фізичної адреси за виконавчою адресою (ефективною адресою).

Це робиться за допомогою однобайтного префікса заміни сегмента (див. рис. 2.4), що повинен передувати машинному коду команди, яка звертається до нестандартного сегмента пам’яті.

Наприклад, мнемокоду команди mov [cs:codeByte], bh; M(CS*16+offset codeByte)<-bh у машинному коді передує префікс заміни сегмента: 2Eh=00101110b, який перепризначає сегмент DS на сегмент CS.

2.2.2.3. Розподіл пам’яті на банки

Коли об’єм однієї комірки пам’яті становить 8 біт, а системна шина даних має більшу кількість ліній, наприклад для МП i8086 – 16 біт, пам’ять фізично розділено на два банки по 512 Кбайт (рис. 2.5). Один банк називається молодшим і містить комірки з парними адресами (обирається сигналом $\overline{A0}$), а другий банк (старший) – з непарними адресами (обирається сигналом $\overline{BHE}$).

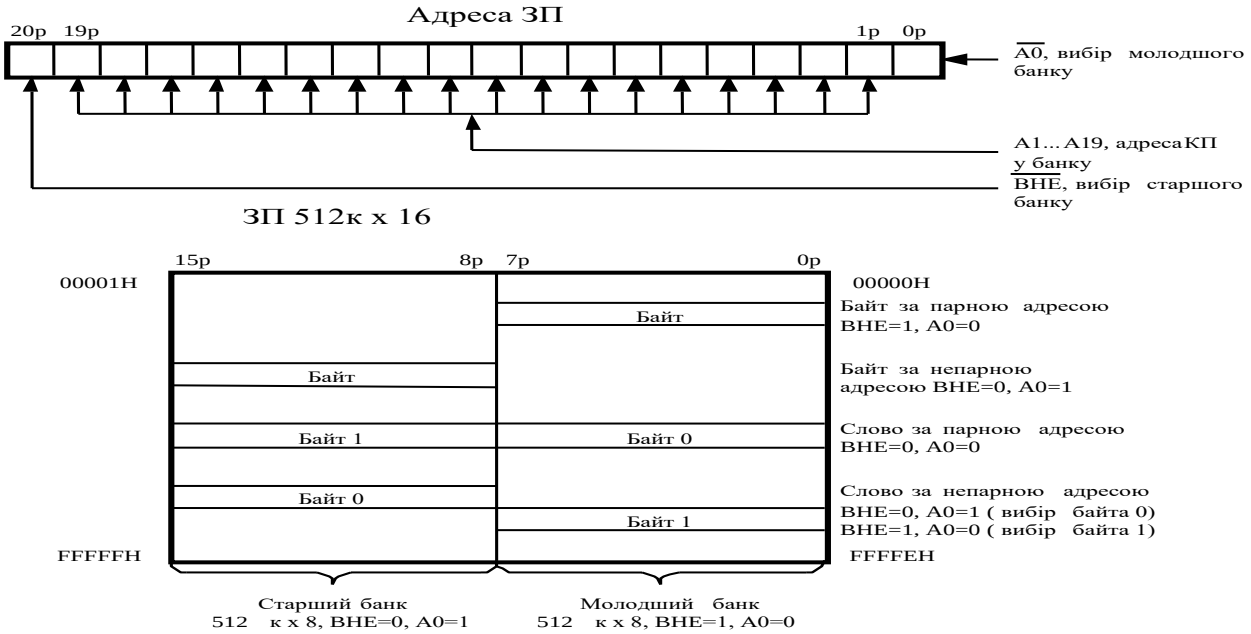


Рис. 2.5. Розподіл пам’яті МП i8086 на банки

Комірки пам’яті обох банків запам’ятовувального пристрою адресуються паралельно 19-розрядними адресами – A19…A1 (2^{19} = 512 Кбайт), а доступ до банків здійснюється сигналами: $\overline{BHE}$ – передавання за системною шиною даних старшого байта і $\overline{A0}$ – нульове значення молодшого біта адреси комірки пам’яті – передавання за системною шиною даних молодшого байта.

Якщо 16-розрядне слово розміщене за парною адресою, то звернення до нього відбувається за один машинний цикл МП. Більш докладно ці питання описано у [2].

2.2.2.4. Структурна схема модуля пам’яті мікропроцесорної системи на основі мікропроцесора i8086

На рис. 2.6 наведено структурну схему модуля пам’яті МПС мінімальної конфігурації на основі МП i8086, яка реалізує розподіл пам’яті на банки, що описано вище.

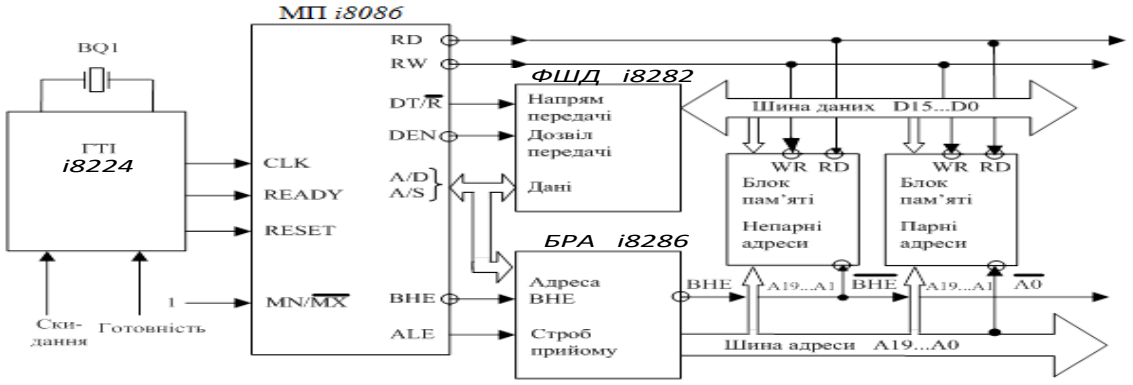


Рис. 2.6. Структурна схема модуля пам’яті МПС на основі МП i8086

Оскільки в МП використовуються мультиплексовані лінії адреси/даних і адреси/стану, значення адреси, що видаються по цих лініях на початку машинного циклу, необхідно запам'ятовувати.

Цю функцію виконують буферні регістри адреси, що запам'ятовують 20 розрядів адреси та значення сигналу $\overline{BHE}$ і видають їх на вихід стабільними на весь період обміну. Підключення ліній системної шини даних D15…D0 до МП здійснюється за допомогою формувачів шини даних.

Сигнал CLK, що синхронізує роботу МП, надходить від генератора тактових імпульсів, що синхронізує також зовнішні сигнали готовності READY і скидання RESET.

Кожен цикл обміну з пам'яттю складається з чотирьох тактів Т1, Т2, Т3 і Т4, тривалість яких збігається з періодом CLK. Якщо пам'ять знімає сигнал готовності READY, то між тактами Т3 і Т4 вставляються такти чекання $T_{чек}$, під час яких МП чекає більш повільного партнера за обміном.

Часову діаграму роботи МП і8086 для машинних циклів читання і запис приведено в [2]. Нижче подано короткий перелік дій МП і8086 у кожному такті машинного циклу.

У такті Т1 на виходи $\overline{BHE}$/S7, A19…A16/S6…S3 та AD15…AD0 видаються адреса пам'яті чи зовнішнього пристрою та сигнал $\overline{BHE}$. У такті Т1 встановлюється також сигнал DT/$\overline{R}$, що визначає напрямок передачі даних через формувач шини даних. Якщо сигнал DT/$\overline{R}$ низького рівня, то формувач шини даних налаштовується на прийом даних з пам'яті чи зовнішній пристрій у МП, якщо ж рівень сигналу високий, то формувачі перебудовуються на передачу даних із МП у пам'ять або зовнішній пристрій.

У такті Т2 з'являється сигнал $\overline{DEN}$, що включає формувач шини даних, з'єднуючи тим самим виводи AD15…AD0 МП із системною шиною даних.

Протягом такту Т3 дані приймаються в МП за сигналом «Читання» – $\overline{RD}$, чи видаються із МП паралельно із сигналом «Запис» – $\overline{WR}$.

У такті Т4 обмін даними закінчується.

У режим апаратного чекання (такти $T_{чек}$) МП переходить за відсутністю сигналу готовності *READY*, який повинен з'явитися в такті Т2 і зберігатися в такті Т3. Якщо рівень сигналу на вході READY – низький, то після такту Т3 з'являється такт чекання $T_{чек}$, в якому всі сигнали МП залишаються незмінними. Причому у такті Т3 і в кожному такті $T_{чек}$ перевіряється рівень сигналу на вході READY. Поки він залишається низьким, повторюються такти $T_{чек}$.

Для виходу зі стану чекання значення READY = 1 повинно з’явитися на початку такту Т3 чи $T_{чек}$. Тривалість такту $T_{чек}$ дорівнює періоду T_{clk}. Сигнал READY не повинен змінюватися під час перевірок, тому він синхронізується генератором тактових імпульсів.

2.2.2.5. Функціональна схема модуля пам’яті мікропроцесорної системи на основі мікропроцесора i8086

На рис. 2.7 наведено функціональну схему модуля пам’яті МПС на основі МП i8086, яка більш докладно пояснює організацію доступу до окремих 8-розрядних комірок пам’яті та 16-розрядної шини даних.

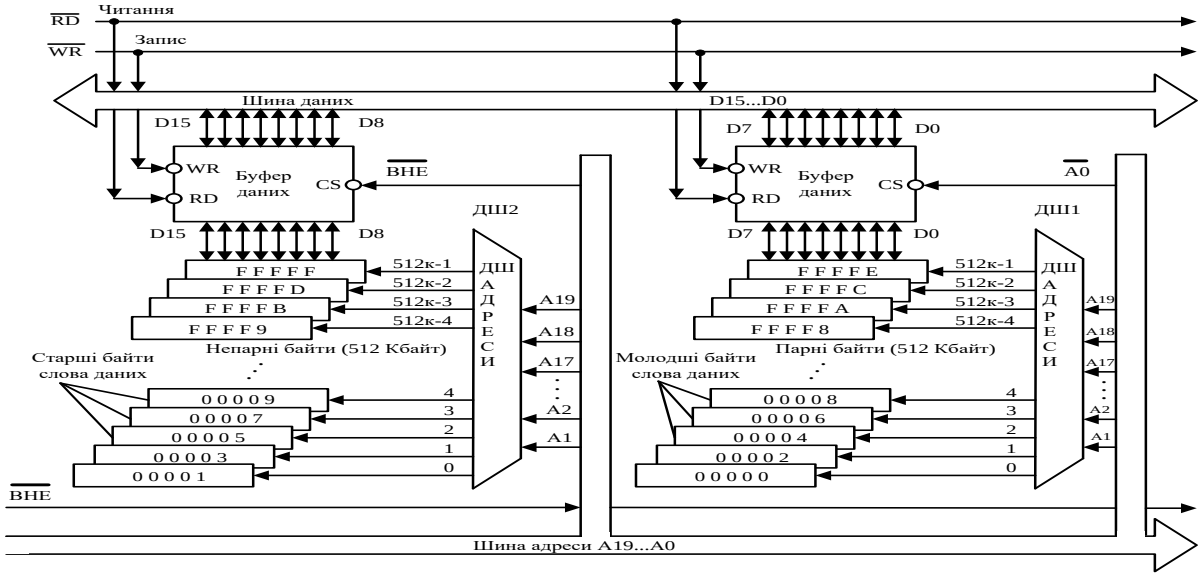


Рис. 2.7. Функціональна схема модуля пам’яті МПС на основі МП i8086

У банку пам’яті з парними адресами комірок (молодший банк) зберігаються дані, що передаються через буфер даних по восьми молодшим лініям системної шиної даних: D7…D0, а в банку пам’яті з непарними адресами (старший банк) знаходяться дані, що обмінюються з процесором по восьми старшим лініям системної шиної даних: D15…D8. Буфер даних непарного банку пам’яті підключається сигналом $\overline{BHE}$, парного – сигналом $\overline{A0}$.

2.3. Організація пам’яті мікропроцесорної системи на основі мікроконтролера

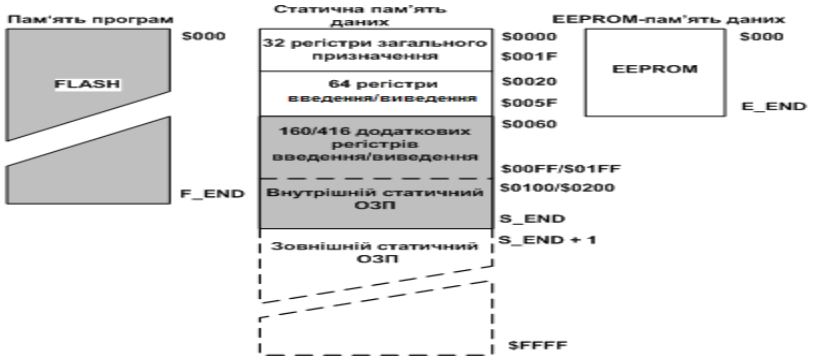
2.3.1. Загальна характеристика пам’яті

Нижче розглянуто організацію пам’яті МПС на основі 8-розрядного AVR-мікроконтролера. В цих МК реалізовано Гарвардську архітектуру, відповідно до якої розділено не лише адресні простори пам’яті програм та

пам’яті даних, але також і шини доступу до них [3; 4]. Способи адресації та доступу до цих областей пам’яті також є різними.

Така структура дає змогу центральному процесору одночасно працювати як з пам’яттю програм, так і з пам’яттю даних, що суттєво збільшує продуктивність. Кожна з областей пам’яті даних – статична пам’ять даних та EEPROM-пам’ять також розміщені у своєму адресному просторі.

На рис. 2.8 наведено карту пам’яті деяких AVR-мікроконтролерів сімейства Mega.



Модель	Пам'ять програм (FLASH)		Статична пам'ять даних (СПД)				Пам'ять даних (EEPROM)	
	Верхня границя [F_END]	Об'єм [слів]	Верхня Границя [S_END]	Об'єм СОЗП [байт]	Кількість дод. регістрів введ./виведення	Зовнішній СОЗП	Верхня границя [F_END]	Об'єм [байт]
ATmega48x	$007FF	2 K	$02FF	512	160		$0FF	256
ATmega8515x	$00FFF	4 K	$025F	512	0	•	$1FF	512
ATmega8535x	$00FFF	4 K	$025F	512	0		$1FF	512
ATmega8x	$00FFF	4 K	$045F	1 K	0		$1FF	512
ATmega88x	$00FFF	4 K	$04FF	1 K	160		$1FF	512
ATmega16x	$01FFF	8 K	$045F	1 K	0		$1FF	512
ATmega162x	$01FFF	8 K	$04FF/$045F	1 K	160/0	•	$1FF	512
ATmega164x	$01FFF	8 K	$04FF	1 K	160		$1FF	512
ATmega165x	$01FFF	8 K	$04FF	1 K	160		$1FF	512
ATmega168x	$01FFF	8 K	$04FF	1 K	160		$1FF	512
ATmega32x	$03FFF	16 K	$085F	2 K	0		$3FF	1 K
ATmega324x	$03FFF	16 K	$08FF	2 K	160		$3FF	1 K
ATmega325x	$03FFF	16 K	$08FF	2 K	160		$3FF	1 K
ATmega3250x	$03FFF	16 K	$08FF	2 K	160		$3FF	1 K
ATmega64x	$07FFF	32 K	$10FF	4 K	160	•	$7FF	2 K
ATmega640x	$07FFF	32 K	$21FF	8 K	416	•	$FFF	4 K
ATmega644x	$07FFF	32 K	$10FF	4 K	160		$7FF	2 K
ATmega645x	$07FFF	32 K	$10FF	4 K	160		$7FF	2 K
ATmega6450x	$07FFF	32 K	$10FF	4 K	160		$7FF	2 K
ATmega128x	$0FFFF	64 K	$10FF	4 K	160	•	$FFF	4 K
ATmega1280x	$0FFFF	64 K	$21FF	8 K	416	•	$FFF	4 K
ATmega1281x	$0FFFF	64 K	$21FF	8 K	416	•	$FFF	4 K
ATmega2560x	$1FFFF	128 K	$21FF	8 K	416	•	$FFF	4 K
ATmega2561x	$1FFFF	128 K	$21FF	8 K	416	•	$FFF	4 K

Примітка. Позначкою «•» позначені моделі, до яких можна підключити зовнішній статичний ОЗП.

Рис. 2.8. Карта пам’яті деяких МК сімейства Mega

Оскільки AVR-мікроконтролери мають 16-бітну систему команд, об’єм пам’яті програм на рисунку вказано не у байтах, а в 16-бітових словах. Символ «$» є ознакою шістнадцяткової системи числення.

2.3.2. Організація пам’яті програм

У МПС у пам’яті програм зберігаються команди, що керують роботою МК, та константи, що не змінюються під час роботи програми. Пам’ять програм «інтелоподібних» МП та МК, розглянутих вище, має 8-розрядну організацію.

Пам’ять програм AVR-мікроконтролерів має 16-бітну організацію, для якої довжина кожної команди кратна одному слову – 16 біт. Наприклад, об’єм

пам’яті програм МК сімейства Mega складає від 2 К (2 ·1024) до більш ніж 128 К (128 ·1024) 16-бітних слів.

У частини моделей МК сімейства Mega пам’ять програм логічно поділено на 2 рівні частини: область прикладної програми і область завантажувача [3; 4].

В останній може розміщуватися спеціальна програма – завантажувач, що дозволяє МК самостійно керувати завантаженням та вивантаженням прикладних програм. Якщо самопрограмування МК не використовується, прикладна програма може розміщуватися і в області завантажувача.

Для адресації пам’ять програм використовується лічильник команд (англ. Program Counter – PC). Розмір лічильника команд залежить від об’єму пам’яті, що адресується.

За адресою $0000 пам’яті програм знаходиться вектор скидання. Після ініціалізації (скидання) МК виконання програми починається з цієї адреси, за якою повинна розміщуватися команда переходу до основної частини ініціалізації програми. Починаючи з адреси $0001 пам’яті програм (моделі з пам’яттю програм 8 Кбайт і менше) або $0002 (останні моделі) розміщується таблиця векторів переривань. Розмір цієї області залежить від моделі МК. Розподіл області векторів переривань розглянуто окремо у [3; 4].

У разі виникнення переривання після збереження у стеку поточного значення лічильника команд відбувається виконання команди переходу до підпрограм обробки переривань. У моделях з пам’яттю програм невеликого об’єму (8 Кбайт і менше) в таблицях векторів переривань використовуються команди відносного переходу – RJMP, а в останніх моделях – команди абсолютного переходу – JMP.

У більшості МК сімейства Mega положення вектора скидання і таблиці векторів переривань може бути змінено. Вони можуть розміщуватися не лише на початку пам’яті програм, але і на початку області завантажувача.

В якості пам’яті програм AVR-мікроконтролерів використовується FLASH-ПЗП, який розрахований, що найменше, на 10000 циклів очищення/запису.

2.3.3. Організація пам’яті даних

2.3.3.1. Загальна характеристика пам’яті даних

Пам’ять даних AVR-мікроконтролерів розділено на дві частини: енергозалежну статичну пам’ять даних і енергонезалежну EEPROM-пам’ять.

Статична пам’ять даних включає регістрову пам’ять та статичний ОЗП.

Регістрова пам’ять має 32 регістри загального призначення, об’єднаних у регістровий файл, і 64 основні службові регістри введення/виведення.

У моделях з розвиненою периферією є також область додаткових (extended) регістрів введення/виведення.

Під основні регістрів введення/виведення у пам’яті МК відводиться 64 байти, а під додаткові регістри введення/виведення − 160 або 416 байт (залежно від моделі).

В обох областях регістри введення/виведення розміщуються різні службові регістри: регістри керування, стану МК і т. ін., а також регістри керування периферійними пристроями, що належать до МК. Загальна кількість регістрів введення/виведення і додаткових регістрів введення/виведення залежить від конкретної моделі МК.

Для зберігання змінних окрім регістрів загального призначення використовується також статичний ОЗП. Низка МК сімейства, крім того, мають можливість підключення зовнішнього статичного ОЗП об’ємом до 64 Кбайт.

Регістрову пам’ять разом із статичним ОЗП називають статичною пам’ятю даних. Фізично таку пам’ять виконано на основі тригерів, тому вона є енергозалежною – зберігає інформацію, доки є живлення.

Для довготривалого зберігання різної інформації, яка може змінюватися у процесі функціонування готової системи (калібрувальні константи, серійні номери, ключі і т. ін.), у МК сімейства може використовуватися вбудована енергонезалежна EEPROM-пам’ять. Її об’єм становить для різних моделей від 256 до більш ніж 4 Кбайт. EEPROM-пам’ять розміщено в окремому адресному просторі, а доступ до неї здійснюється за допомогою відповідних РВВ.

2.3.3.2. Статична пам’ять даних

В AVR-мікроконтролерах використовується лінійна організація статичної пам’яті даних (рис. 2.9).

До статичної пам’яті даних належать регістри загального призначення, регістри введення/виведення, а також статичний ОЗП. Максимальний об’єм статичної пам’яті даних становить 65536 байт та залежить від конкретної моделі сімейства.

2.3.3.3. Регістри загального призначення

32 регістри загального призначення об’єднано в регістровий файл швидкого доступу, структуру якого показано на рис. 2.10.

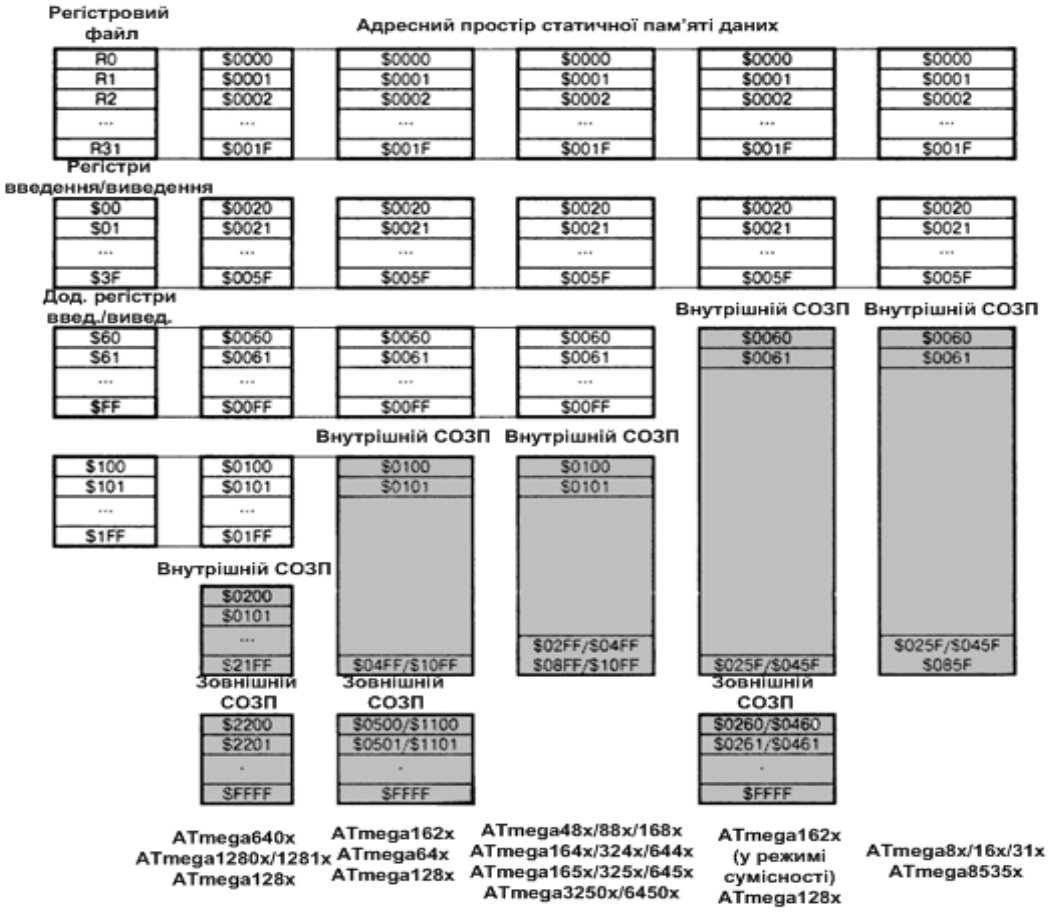


Рис. 2.9. Організація статичної пам'яті даних

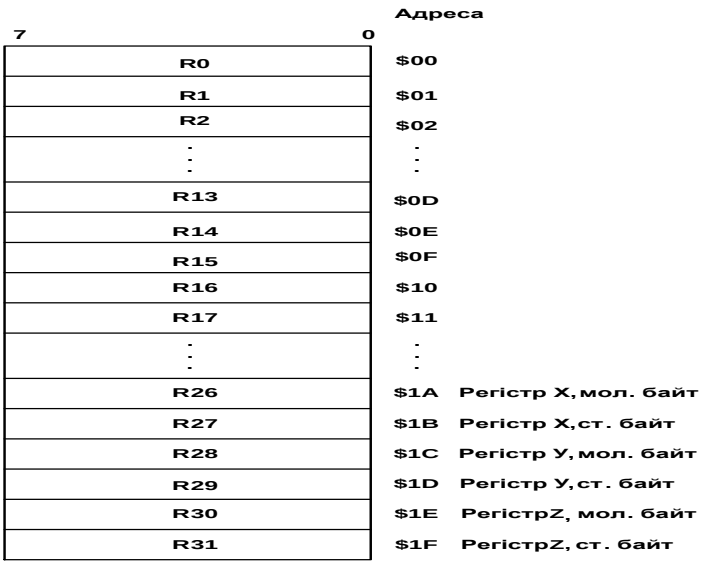


Рис. 2.10. Структура регістрів загального призначення

В AVR-мікроконтролерах всі регістри загального призначення безпосередньо доступні арифметико-логічному пристрою, на відміну від 8-бітових МК деяких інших фірм, в яких є лише один такий регістр − робочий регістр A/W (акумулятор). Завдяки цьому будь-який регістр загального

призначення може використовуватися практично у всіх командах і як операнд-джерело, і як операнд-приймач. Таке рішення (у поєднанні з конвеєрною обробкою, див. підрозд. 3.5 дозволяє арифметико-логічному пристрою виконувати за один такт одну операцію: читання операндів з регістрового файлу, виконання команди і запис результату назад у регістровий файл.

Як показано на рис. 2.9, кожен регістр файлу має свою власну адресу в просторі статичної пам'яті даних. Тому до них можна звертатися двома способами – як до регістрів та як до частини статичної пам'яті даних. Таке рішення є ще однією відмінною особливістю архітектури AVR, що підвищує ефективність роботи МК та його продуктивність.

Останні 6 регістрів файлу (R26…R31) можуть об'єднуватися у три 16-бітні регістри: X, Y та Z (рис. 2.11), що використовуються як покажчики за непрямої адресації статичної пам'яті даних.

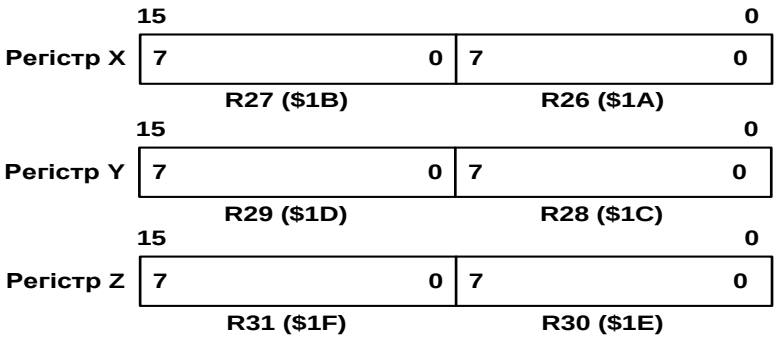


Рис. 2.11. Регістри-покажчики X, Y та Z

2.3.3.4. Стек

Призначення та організацію стека в МПС описано у п. 2.1.9. AVR-мікроконтролери мають обидва різновиди стека (залежно від моделі) апаратний і програмний. Стек розміщується у статичному ОЗП, і його глибина визначається тільки розміром вільної області цієї пам'яті.

Залежно від ємності статичного ОЗП, як покажчик стека використовується або один регістр введення/виведення SPL (SP), розміщений за адресою $3D ($5D), або пара регістрів SPH:SPL, розміщених за адресами $3E ($5E) і $3D ($5D) відповідно.

Регістри-покажчики стека є звичайними регістрами введення/виведення і, відповідно, програмно доступні.

У наборі команд AVR-мікроконтролерів є команди занесення в стек – PUSH і зчитування зі стека – POP, що дозволяє програмі використовувати стек для своїх потреб. На початку програми покажчик стека необхідно проініціалізувати, записавши в нього значення верхньої вільної адреси стекової

пам’яті даних (зазвичай, максимальна адреса статичної пам’яті даних). Стек заповнюється у напрямку зменшення значення регістра SP. Початкове значення SP адресує верхівку стека – першу вільну комірку стекової пам’яті.

Під час програмного виклику підпрограм, або виклику підпрограм за перериванням, у стеку зберігається адреса наступної команди перерваної програми (значення лічильника команд). Значення покажчика стека при цьому зменшується на 2, тому що для збереження лічильника команд потрібно 2 байти. Під час повернення з підпрограми ця адреса відновлюється зі стека і завантажується в лічильник команд. Значення покажчика стека відповідно збільшується на 2.

2.3.3.5. Регістри введення/виведення

Усі регістри введення/виведення можна умовно розділити на дві групи: службові регістри МК і регістри, що належать до конкретних периферійних пристроїв (зокрема регістри портів введення/виведення).

Частина регістрів введення/виведення розміщуються в основному просторі введення/виведення, розміром 64 байти. У більшості моделей сімейства є також простір додаткових регістрів введення/виведення розміром 160 або 416 байт. Введення додаткових регістрів введення/виведення пов’язане з тим, що для підтримки всіх периферійних пристроїв, наявних у цих моделях, адрес перших основних 64-х регістрів введення/виведення недостатньо.

Кількість та розподіл адрес простору введення/виведення (як основного, так і додаткового) залежить від конкретної моделі МК або, якщо точніше, від складу і можливостей периферійних пристроїв цієї моделі [3; 4].

Під час вказівки адрес деяких регістрів введення/виведення у дужках вказуються відповідні їм адреси додаткових регістрів введення/виведення – комірок статичної пам’яті даних. Відповідно, якщо адреса регістра вказується лише в дужках, цей регістр розміщено в просторі додаткових регістрів введення/виведення. Якщо адреса в таблиці регістрів введення/виведення не вказана, це значить, що для цієї моделі він зарезервований, і запис за цією адресою заборонено (для сумісності з майбутніми моделями).

Якщо адреса, наприклад регістр SREG вказана як $3F ($5F), то це означає, що за адресою $3F можна відповідними командами звернутися до регістрів введення/виведення основного простору регістрів введення/виведення, розміром 64 байти, а за адресою $5F можна звернутися до регістрів введення/виведення, як частині простору статичної пам’яті даних.

Різниця між цими адресами становить $20, тому що у просторі статичної пам’яті даних перед основними регістрами введення/виведення знаходяться 32 регістри загального призначення.

Приклади регістрів введення/виведення деяких моделей Mega-AVR-мікроконтролерів наведено у [3; 4].

До регістрів введення/виведення, розміщених в основному просторі введення/виведення, можна безпосередньо звернутися за допомогою команд IN і OUT, що виконують пересилання даних між одним з 32-х регістрів загального призначення і регістром основного простору введення/виведення. У системі команд є також чотири команди побітового доступу, що використовують як операнди регістри введення/виведення: команди встановлення/скидання окремого біта – SBI і CBI і команди перевірки стану окремого біта – SBIS і SBIC. Ці команди можуть звертатися лише до першої молодшої половини основного простору введення/виведення – адреси $00…$1F.

Окрім адресації регістрів основного простору введення/виведення за допомогою команд IN і OUT, до регістрів введення/виведення можна звертатися як до комірок статичної пам’яті даних за допомогою команд ST/SD/SDD і LD/LDS/LDD (для додаткових регістрів введення/виведення тільки цей спосіб є можливим).

Серед регістрів введення/виведення є один регістр, що часто використовується у процесі виконання програм – регістр стану SREG. Він належить до основного простору введення/виведення та розміщується за адресою $3F ($5F) і містить набір прапорців, що показують поточний стан МК під час виконання програми. Більшість прапорців у разі настання певних подій відповідно до результату виконання команд автоматично встановлюються в одиницю або скидаються в нуль. Усі біти цього регістра доступні як для читання, так і для запису. Після скидання МК вони скидаються в нуль.

Формат регістра SREG показано на рис. 2.12, а опис його його окремих розрядів – в табл. 2.1.

	7	6	5	4	3	2	1	0
	I	T	H	S	V	N	Z	C
Читання(R)\Запис(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Початкове значення	0	0	0	0	0	0	0	0

Рис. 2.12. Формат регістра стану SREG

Інші регістри введення/виведення буде описано нижче у відповідних розділах цього підручника та у [3; 4].

Таблиця 2.1. **Формат стану SREG**

Розряд	Назва	Опис
7	I	**Загальний дозвіл переривань.** Щоб дозволити переривання цей прапорець необхідно встановити в одиницю. Дозволити/заборонити окремі переривання можна встановленням або скиданням відповідних розрядів регістрів масок переривань. Якщо прапорець скинуто в нуль, то переривання заборонено незалежно від стану цих розрядів. Прапорець скидається апаратно після входу у підпрограму обробки переривання і відновлюється командою RETI, дозволяючи обробку наступних переривань
6	T	**Зберігання копійованого біта.** Цей розряд регістра використовується як джерело або приймач командами копіювання біта BLD (Bit Load) та BST (Bit Store). Вказаний розряд будь-якого регістра загального призначення можна скопіювати у цей розряд командою BST або змінити відповідно до вмісту даного розряду командою BLD
5	H	**Прапорець половинного перенесення**. Цей прапорець встановлюється в одиницю під час виконання деяких арифметичних операцій, якщо було або перенесення з молодшої тетради байта (з 3-го розряду в 4-й), або позика зі старшої тетради
4	S	**Прапорець знака.** Цей прапорець дорівнює результату операції «виключне АБО» – XOR між прапорцями N (від'ємний результат) і V (переповнення числа у додатковому коді). Прапорець дозволяє збільшити розрядність результату арифметичних операцій у додатковому коді з семи значущих біт до восьми із прапорцем S у якості дев'ятого знакового біта. Відповідно цей прапорець встановлюється в одиницю, якщо результат виконання арифметичної операції менший від нуля
3	V	**Прапорець переповнення додаткового коду.** Цей прапорець встановлюється в одиницю у разі переповнення розрядної сітки 8-розрядного знакового результату. Використовується під час роботи зі знаковими числами, які подано у додатковому коді
2	N	**Прапорець від'ємного значення**. Цей прапорець встановлюється в одиницю, якщо старший 7-й розряд результату операції дорівнює одиниці. Інакше прапорець дорівнює нулю
1	Z	**Прапорець нуля.** Цей прапорець встановлюється в одиницю, якщо результат виконання операції дорівнює нулю
0	C	**Прапорець перенесення.** Цей прапорець встановлюється в одиницю, якщо в результаті виконання операції відбувся вихід за межі байта

2.3.3.6. Використання зовнішнього оперативного запам'ятовувального пристрою

Частина AVR-мікроконтролерів, наприклад, Atmega8515x, Atmegal62x, Atmega64x/128x і Atmega640x/1280x/1281x/2560x/2561x мають можливість підключення зовнішнього статичного ОЗП об'ємом до 64 Кбайт.

Виводи, що використовуються для підключення зовнішнього ОЗП, схема його підключення до МК, конфігурація, часові діаграми звернення до зовнішнього ОЗП та програмна реалізація описані у [3; 4].

2.3.3.7. Енергонезалежна пам’ять даних EEPROM

Частина МК сімейства Tiny та всі МК сімейств Mega та Xmega мають енергонезалежну пам’ять даних (EEPROM-пам’ять). Цю пам’ять розміщено у власному адресному просторі, а її об’єм становить від 60 байт (для сімейства Tiny) та від 256 байт до більш ніж 4 Кбайт для сімейств Mega та Xmega.

Для запису та читання EEPROM-пам’яті в готовому пристрої використовуються три регістри введення/виведення: регістр адреси – EEAR, регістр даних – EEDR та регістр керування – EECR. РА EEAR фізично розміщується у двох регістрах введення/виведення: EEARH:EEARL. Адреси, формати та опис окремих розрядів регістрів керування EEPROM-пам’яті наведено у [3; 4].

Процедура запису одного байта в EEPROM-пам’ять складається з таких етапів:

1. Для визначення готовності пам’яті до запису даних треба дочекатися, коли скинеться прапорець EEPE (EEWE) регістра EECR.
2. Для визначення завершення запису у FLASH-пам’ять програм дочекатися, коли скинеться прапорець SPMEN регістра SPMCR (Store Program Memory Control Register).
3. Завантажити байт даних у регістр EEDR, а необхідну адресу – у регістр EEAR.
4. Встановити в одиницю прапорець EEMPE (EEMWE) регістра EECR.
5. Протягом чотирьох машинних циклів записати в розряд EEPE (EEWE) регістра EECR одиницю. Після встановлення цього розряду процесор пропускає 2 такти перед виконанням наступної інструкції.

Другий пункт введено через те, що запис в EEPROM-пам’ять не може виконуватися одночасно із записом у FLASH-пам’ять. Тому перед виконанням запису в EEPROM-пам’ять варто переконатися, що програмування FLASH-пам’яті завершено. Якщо в програмі відсутній завантажувач, тобто МК ніколи не змінює вміст пам’яті програм, другий крок можна пропустити.

На процес звернення до EEPROM-пам’яті впливає внутрішній RC-генератор, що калібрується. Відповідно, тривалість циклу запису залежить від частоти цього генератора, напруги живлення та температури [3; 4].

За закінченням циклу запису розряд EEPE (EEWE) апаратно скидається, після чого програма може почати запис наступного байта.

Під час запису в EEPROM можуть виникнути деякі проблеми, які викликані перериваннями:

1. У разі виникнення переривання між 4-м та 5-м етапами описаної вище послідовності, запис в EEPROM буде перервано, тому що за час обробки переривання прапорець EEMPE (EEMWE) скинеться в нуль.

2. Якщо в підпрограмі обробки переривання, що виникло під час запису в EEPROM-пам’ять, також відбувається звернення до неї, то буде змінено вміст регістрів адреси та даних EEPROM. У результаті перший перерваний запис буде зірвано.

Для запобігання описаних проблем рекомендовано забороняти всі переривання (скидати біт I регістра SREG) на час виконання пунктів 2…5 описаної вище послідовності.

Два приклади реалізації функції запису в EEPROM-пам’ять та процедуру читання наведено у [3; 4].

2.3.4. Програмування FLASH- та EEPROM-пам’яті

2.3.4.1. Загальна характеристика програмування

Одним з питань, яке треба вирішувати під час розробки мікроконтролерної системи (МКС), є програмування FLASH- та EEPROM-пам’яті. AVR-мікроконтролери підтримують такі режими програмування [3; 4]:

- режим послідовного програмування (за SPI-інтерфейсом);
- режим паралельного програмування за високої напруги;
- режим програмування за JTAG-інтерфейсом.

2.3.4.2. Сторінкова організація пам’яті програм і даних

В AVR-мікроконтролерах використовується сторінкова організація FLASH-пам’яті програм та EEPROM-пам’яті даних [3; 4].

Під час програмування FLASH-пам’яті весь її об’єм розбивається на сторінки, розмір і кількість яких залежить від об’єму пам’яті програм МК (табл. 2.2). Дані спочатку завантажуються в буфер сторінки і тільки потім заносяться безпосередньо в пам’ять програм. Прошивка всіх комірок сторінки при цьому відбувається одночасно.

Аналогічно організовано EEPROM-пам’ять. Розмір сторінок, а також їх кількість наведено в табл. 2.3.

Таблиця 2.2. **Параметри сторінкової організації пам’яті програм**

Параметр	Об’єм пам’яті програм (байт)						
	4 К	8 К	16 К	32 К	64 К	128 К	256 К
Розмір сторінки, слів	32	64	64	128	128	128	256
Кількість сторінок	128	128	256	256	512	1024	1024

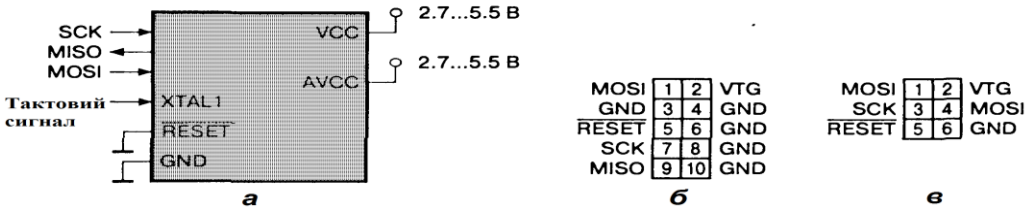
Таблиця 2.3. **Параметри сторінкової організації EEPROM-пам’яті**

Параметр	Об’єм EEPROM-пам’яті (байт)				
	256 К	512 К	1 К	2 К	4 К
Розмір сторінки, байт	4	4	4	8	8
Кількість сторінок	64	128	256	256	512

У багатьох моделях сторінкова організація EEPROM-пам’яті використовується тільки під час програмування в паралельному режимі, а програмування за послідовним каналом відбувається побайтно.

2.3.4.3. Програмування пам’яті за послідовним каналом

Для програмування або перепрограмування пам’яті програм і даних безпосередньо у пристрої використовують послідовний SPI-інтерфейс (рис. 2.13).



Примітка. Якщо в якості тактового використовується внутрішній RC-генератор, вивід XTAL1 залишають непідключеним.

Рис. 2.13. Включення МК у режимі програмування за послідовним каналом

На рис. 2.13*а* наведено схему включення мікроконтролера в цьому режимі. На рис. 2.13*б, в* показано два варіанти розводки колодки для підключення програматора, які рекомендовано компанією Atmel. Зокрема, перший варіант розводки (рис. 2.13*б*) використовується в програматорі AVRISP і налагоджувальних платах STK200/300 компанії Atmel. Другий варіант розводки (рис. 2.13*в*) використовується в більш нових пристроях компанії – програматорі AVRISP mkll і платі STK500.

Як видно з рис. 2.13*а*, для обміну даними між програматором і пристроєм використовується три лінії SPI-інтерфейсу: SCK – лінія тактового сигналу,

MOSI – вхід даних і MISO – вихід даних. Відповідність між цими лініями інтерфейсу і контактами портів введення/виведення деяких МК наведено в табл. 2.4.

Таблиця 2.4. **Виводи, які використовуються під час програмування за послідовним каналом**

Назва лінії інтерфейсу	**Atmega8515x/8535x**	**Atmega8x**	**Atmega16x/32x**	**Atmega64x/128x**	**Atmega48x/88x/168x**	**Atmega162x**	**Atmega164x/324x/644x**	**Atmega165x, Atmega325x/3250x, Atmega645x/6450x**	**Atmega640x/1280x/2560x**	**Atmega1281x/2561x**	**Призначення виводів**
SCK	PB7	PB5	PB7	PB1	PB5	PB7	PB7	PB1	PB1	PB1	Вхід тактового сигналу
MISO	PB6	PB4	PB6	PE1	PB4	PB6	PB6	PB3	PE1	PB3	Вихід даних
MOSI	PB5	PB3	PB5	PE0	PB3	PB5	PB5	PB2	PE0	PB2	Вхід даних

У деяких моделях МК виводи, що використовуються для програмування, не збігаються з виводами портів введення/виведення модуля SPI.

Часові діаграми, параметри сигналів, команди та послідовність програмування наведено в [3].

Контрольні запитання та завдання

1. Опишіть призначення та місце модуля пам’яті в МПС.
2. Опишіть підключення модуля пам’яті до системної шини.
3. Наведіть класифікацію пам’яті.
4. Чим відрізняються енергонезалежна та енергозалежна пам’ять? Наведіть приклади.
5. Чим відрізняються статична та динамічна пам’ять? Наведіть приклади.
6. Назвіть призначення ОЗП та ПЗП.
7. Наведіть та поясніть основні характеристики пам’яті.
8. Назвіть основні задачі, які вирішуються під час проектування пам’яті МПС.
9. Як виконується сторінкова адресація?
10. Дайте визначення FLASH ПЗП.

11. Поясніть фізичну та логічну організацію пам’яті в МПС на основі і8086.
12. Як адресуються комірки пам’яті у МПС на основі МП і8086?
13. Поясніть призначення та організацію стека у МПС на основі МП і8086 і МК типу AT89C51.
14. Які ви знаєте способи обміну даними в МПС?
15. Що означає режим ПДП і в чому його необхідність?
16. Назвіть розрядність однієї комірки пам’яті програм МК AVR.
17. Опишіть організацію та призначення пам’яті програм.
18. На які три частини розділено пам’ять даних МК сімейства? Опишіть призначення та структуру кожного з них.
19. Які види стека ви знаєте? Який вид стека використовується у МК і де він розміщується?
20. За якою архітектурою виконано організацію пам’яті AVR-мікроконтролерів сімейства Mega?
21. Опишіть структуру регістрового файлу.
22. Де в AVR-мікроконтролерах розміщуються регістри введення/виведення? Який з них використовується найчастіше?
23. Що таке енергонезалежна пам’ять даних (EEPROM)? Які регістри використовуються для керування EEPROM?
24. Опишіть процедуру запису одного байта в EEPROM-пам’ять даних.
25. Опишіть процедуру зчитування інформації з EEPROM-пам’яті даних.
26. Які режими програмування підтримують МК сімейства Mega?
27. Які лінії використовуються для обміну даними між програматором і пристроєм під час програмування за послідовним каналом?
28. Опишіть формат та призначення окремих розрядів регістра SREG.
29. Що означає символ «$» під час наведення адрес пам’яті?
30. Для чого використовується лічильник команд (Program Counter) та від чого залежить його розмір?

3. ПРОГРАМУВАННЯ МІКРОПРОЦЕСОРНИХ СИСТЕМ

3.1. Роль керувальної програми у роботі мікропроцесорних систем та програмна модель мікропроцесора/мікроконтролера

3.1.1. Послідовність розробки робочої керувальної програми

Якщо МПС виконана на основі МП, то керувальна програма знаходиться у зовнішній пам'яті програм (комірках ПЗП), а якщо основним вузлом МПС є МК, то програма може знаходитися у резидентній пам'яті програм і/або зовнішній пам'яті програм. Без керувальної програми жодна МПС працювати не буде.

Основні етапи створення керувальної програми:

– розробка схеми алгоритму роботи, який повинна реалізувати керувальна програма;
– якщо задача складна, то програма може бути реалізована і відлагоджена мовою високого рівня або мовою Асемблера;
– компіляція – переведення програми з мови високого рівня або Асемблера в машинні коди конкретного МП/МК (створення об'єктного модуля);
– введення керувальної програми у ПЗП за допомогою програматора;
– початкова ініціалізація системи, в процесі якої у програмний лічильник завантажується початкова адреса програми.

Керувальна програма може включати підпрограми – частини основної програми, які можуть бути викликані відповідною командою з основної програми або за перериванням.

Однією з основних характеристик архітектури будь-якого МП/МК є програмна (програмістська) модель, що включає частину структури, що доступна програмісту за допомогою системи команд [2; 3].

3.1.2. Програмна модель 8-розрядного мікропроцесора

На рис. 3.1 наведено програмну (програмістську) модель 8-розрядного МП, наприклад, i8080.

До програмістської моделі МП належать:

– регістри загального призначення: B, C, D, E, H, L, які є надоперативним ОЗП;
– слово-стану програми – Program Status Word (рис. 3.2), яке складається з акумулятора A і регістра прапорців (англ. RF): A – старший байт, RF – молодший байт.

Рис. 3.1. Програмна (програмістська) модель МП i8080

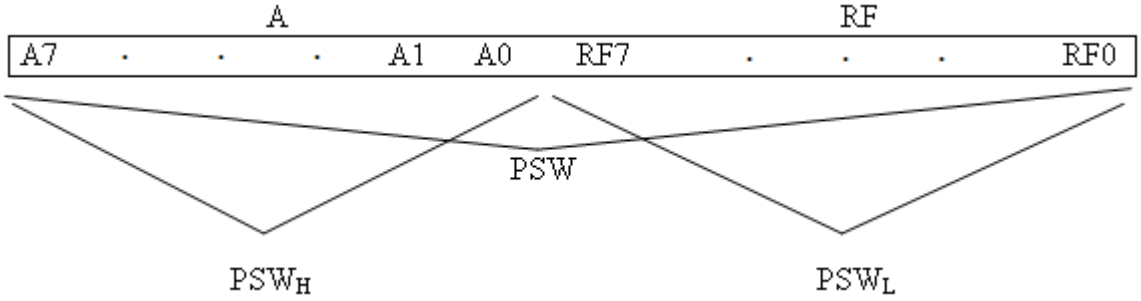


Рис. 3.2. Слово-стану програми (PSW) МП i8080

Регістр прапорців є набором тригерів, які показують стан програми після виконання команд, що впливають на прапорці (див. п. 1.4.1).

3.1.3. Програмна модель 16-розрядного мікропроцесора

У програмній моделі 16-розрядного МП, наприклад, i8086, яку показано на рис. 3.3, є кілька груп регістрів із різним функціональним призначенням.

До групи регістрів загального призначення належать такі регістри: AX, BX, CX і DX. Їхня особливість полягає в тому, що в командах допускається вказувати старшу (H – High) і молодшу (L – Low) половини цих регістрів. Такий подвійний характер регістрів AX…DX дозволяє багатьом командам оперувати байтами і словами. Інші регістри можна використовувати тільки словами. Усього

є вісім 16-розрядних регістрів загального призначення і для вибору кожного з них у машинних кодах команд досить трьох бітів.

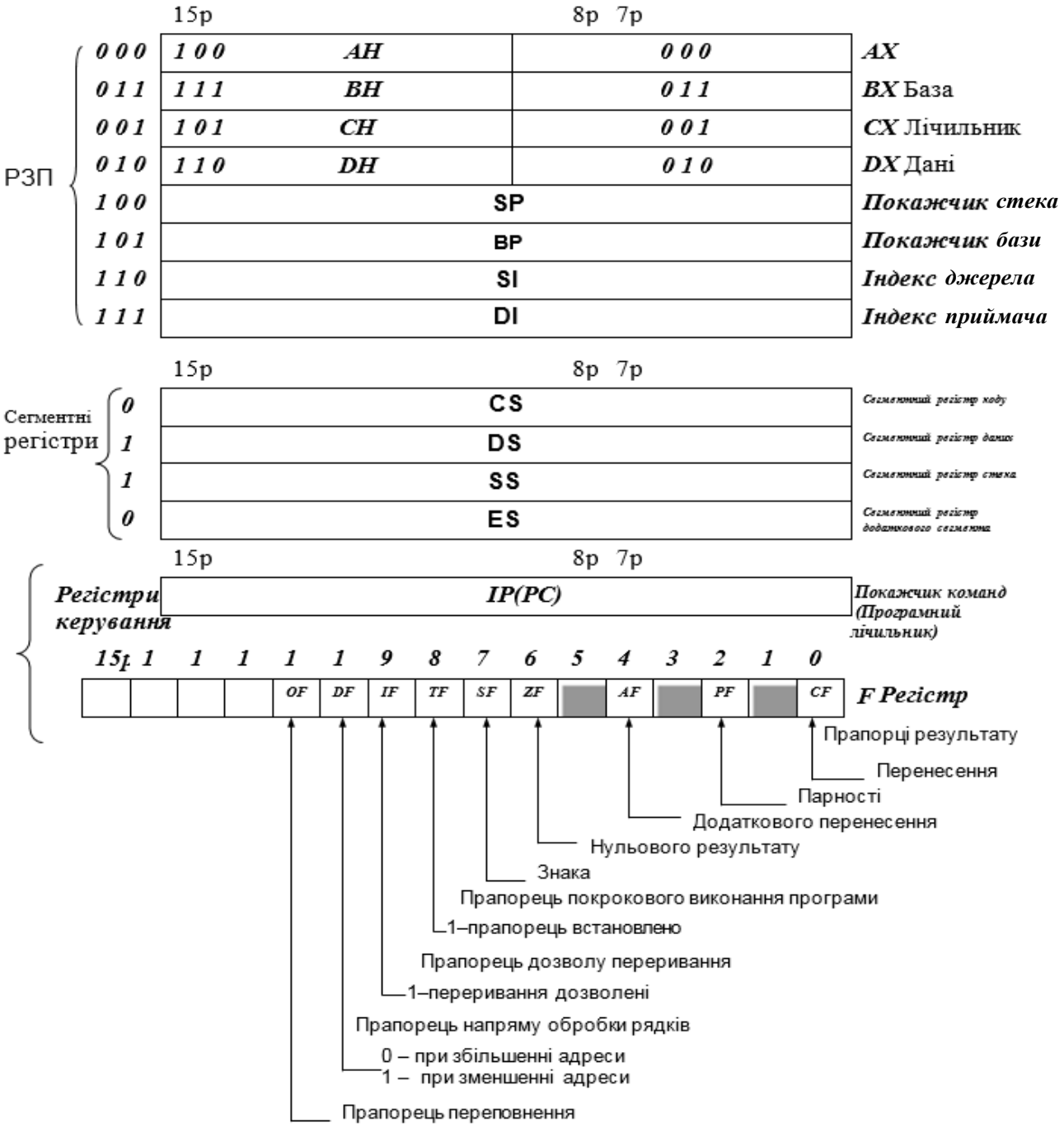


Рис. 3.3. Програмна модель МП i8086

Регістри AX…DX беруть участь в арифметичних і логічних операціях, але у деяких командах вони спеціалізовані, що відбито в їхніх назвах.

Регістр AX використовується в операціях множення, ділення, введення/виведення слів і у деяких операціях з рядками.

Регістр AL бере участь в аналогічних операціях з байтами, в операціях перетворення і десяткової арифметики.

Регістр BX широко застосовується для адресації структур даних у сегменті даних за замовчуванням.

Регістр CX виконує функції лічильника повторень у програмних циклах і в операціях з рядками. Регістр CL служить лічильником зсувів.

Регістр DX бере участь в операціях множення і ділення слів. Окрім того, він зберігає адресу порту у командах введення/виведення з непрямою адресацією.

Регістри BP, SP, SI і DI утворюють групу покажчиків та індексних регістрів. Вони можуть використовуватися для збереження адрес, забезпечуючи непряму адресацію пам'яті, а також беруть участь в обчисленнях ефективної адреси (див. п. 3.3.2). Однак ці регістри можуть залучатися для арифметичних і логічних операцій так само, як і *регістри загального призначення*. Покажчик стека SP використовується для адресації верхівки стека у сегменті стека SS. Обидві стекові операції (PUSH і POP) автоматично модифікують SP. За допомогою покажчика бази BP забезпечується простий доступ до даних, що знаходяться у сегменті стека (не тільки у верхівці стека). Зазвичай регістр BP залучається для адресації параметрів, що передаються через стек підпрограмам. Індексні регістри SI і DI застосовуються для адресації даних, а також елементів рядків у командах роботи з рядками.

Досить нерегулярна структура загальних регістрів вимагає, щоб програміст (чи транслятор з мови високого рівня) ретельно розподіляв регістри і стежив за їх використанням. Водночас неявна вказівка деяких регістрів в операціях і режимах адресації дозволяє компактніше закодувати команди.

У нижній частині рис. 3.3 розміщено *регістр прапорців (ознак)*. Шість прапорців CF, PF, AF, ZF, SF і OF фіксують визначені ознаки результату відповідної арифметичної чи логічної операції (перші п'ять прапорців аналогічні прапорцям МП i8080):

– прапорець перенесення CF фіксує значення біта перенесення, що виникає під час додавання (віднімання) байтів чи слів, а також значення біта, що висувається в операціях зсуву. Він також показує особливість результату операцій множення та ділення. Прапорець перенесення відіграє важливу роль у разі написання програм;
– прапорець паритету (парності) PF переходом в одиницю реєструє наявність парного числа одиниць у восьми молодших бітах результату операції. Він застосовується для контролю достовірності під час передачі даних;
– прапорець допоміжного перенесення AF аналогічний прапорцю CF, але фіксує перенесення з молодшої тетради результату (чи позики). Цей прапорець використовується в операціях двійково-десяткової арифметики;

– прапорець нуля ZF сигналізує про одержання нульового результату операції;
– прапорець знака SF повторює значення старшого біта результату, що у додатковому коді відповідає знаку числа;
– прапорець переповнення OF відзначає втрату старшого біта результату операції додавання чи віднімання над знаковими числами. Переповнення виникає, коли значення перенесень у старший біт і зі старшого біта не збігаються та результат операції не відповідає діапазону чисел зі знаком. Прапорець OF показує також зміну старшого (знакового) біта у разі арифметичних зсувів вліво.

Три останніх прапорці керують деякими діями МП. Програміст може відповідними командами задати стан кожного з них:

– прапорець напрямку DF визначає сканування (перегляд) елементів рядків від менших адрес до більших (DF = 0) чи навпаки (DF = 1);
– прапорець переривання IF задає реакцію МП на запит переривання на вході INT. Якщо IF = 0, запит переривання ігнорується, а якщо IF = 1, МП розпізнає і відповідно реагує на нього. Прапорець IF не впливає на сприйняття переривань на вході NMI, які не маскуються, і внутрішніх переривань;
– прапорець трасування (покрокового виконання програми) TF під час встановлення в одиницю переводить МП у покроковий режим роботи, в якому МП автоматично генерує внутрішнє переривання після виконання кожної команди.

Покажчик команд IP виконує функції програмного лічильника PC (далі використовується назва – IP). У процесі вибірки команд із програмної пам’яті відбувається відповідна модифікація IP для того, щоб він адресував наступну команду, яка повинна виконуватись.

Наявність у програмній моделі чотирьох сегментних регістрів: коду – CS, даних – DS, стеку – SS і додаткових даних – ES пояснюється їхньою участю в адресації пам’яті.

МП має 20-розрядну шину зовнішньої фізичної адреси пам’яті, але у програмній моделі немає жодного регістра завдовжки 20 біт.

Всередині МП фізична адреса пам’яті подана двома 16-бітовими словами, одне з яких називається *логічною базовою (початковою) адресою сегмента*, а друге – *внутрішньосегментним зсувом*. Ці два слова являють собою 32-бітову логічну адресу комірки пам’яті. Пристрій перетворення адрес у шинному інтерфейсі МП у разі кожного звертання до пам’яті перетворює логічну адресу у фізичну. Для програми адресний простір пам’яті складається з чотирьох поточних сегментів, які є логічними одиницями пам’яті з максимальним розміром 64 Кбайт.

Реальний розмір сегмента необов’язково має бути максимальним. Мінімальний розмір сегмента дорівнює 16 байтам. На розміщення сегментів у просторі 1 Мбайт накладається тільки одне обмеження: базова 20-бітова фізична адреса сегмента має бути кратна 16, тому його 4 молодших біти мають бути нульовими. Інакше кажучи, фізична базова адреса сегмента має вигляд: XXXX0H. Нульові біти можна не зберігати, а мати на увазі, і тоді для логічної базової адреси сегмента досить 16 біт. Саме такі «урізані» логічні базові адреси сегментів знаходяться у регістрах CS, DS, SS і ES. Отже, у разі фіксованого вмісту сегментних регістрів максимальний робочий простір, до якого МП має доступ у будь-який момент часу, складається з 64 Кбайт для коду (власне програми), 64 Кбайт для стека і 128 Кбайт для даних. Якщо програмі потрібний більший робочий простір, вона повинна модифікувати вміст сегментних регістрів. Для адресації конкретного байта в сегменті служить другий компонент логічної адреси – зсув. Він є 16-бітовим цілим беззнаковим числом та показує відстань цього байта від початку сегмента. Отже, для утворення фізичної адреси з пари: початок сегмента – зсув необхідно зсунути логічну 16-бітову базову адресу сегмента вліво на 4 біти (у цьому разі молодша тетрада адреси отримає нулі) і додати зсув.

3.1.4. Програмна модель мікроконтролера

На рис. 3.4 наведено програмну модель одного з AVR-мікроконтролерів. Ця модель має регістри загального призначення, регістри введення/виведення, статичний ОЗП, який у технічній літературі англійською мовою називають SRAM, ємністю 128 байт, EEPROM-пам’ять даних, ємністю 512 Кбайт та ПЗП FLASH-типу, ємністю 2 К слів (4 Кбайт). Регістри загального призначення, регістри введення/виведення та статичний ОЗП називають статичною пам’яттю даних, тому що вони є енергозалежними. Відповідно EEPROM-пам’ять даних є енергонезалежною. Опис окремих складових моделі наведено у підрозд. 2.3.3.

3.2. Характеристика команд мікропроцесорів і мікроконтролерів

3.2.1. Код операції команди

Код операції команди – комбінація бітів (не більше восьми для 8-розрядних МП/МК), що знаходяться на початку машинного коду команди та визначають тип операції, що підлягає виконанню у конкретний момент часу. Код операції витягається з пам’яті та розміщується у програмно недоступний регістр команд. Потім він декодується і визначається тип команди, яка має бути виконана. Для 16-розрядних МП код операції частково може знаходитися у другому байті машинного коду команди (постбайті). Окрім кодів операції у машинний код команди можуть входити адреси та операнди.

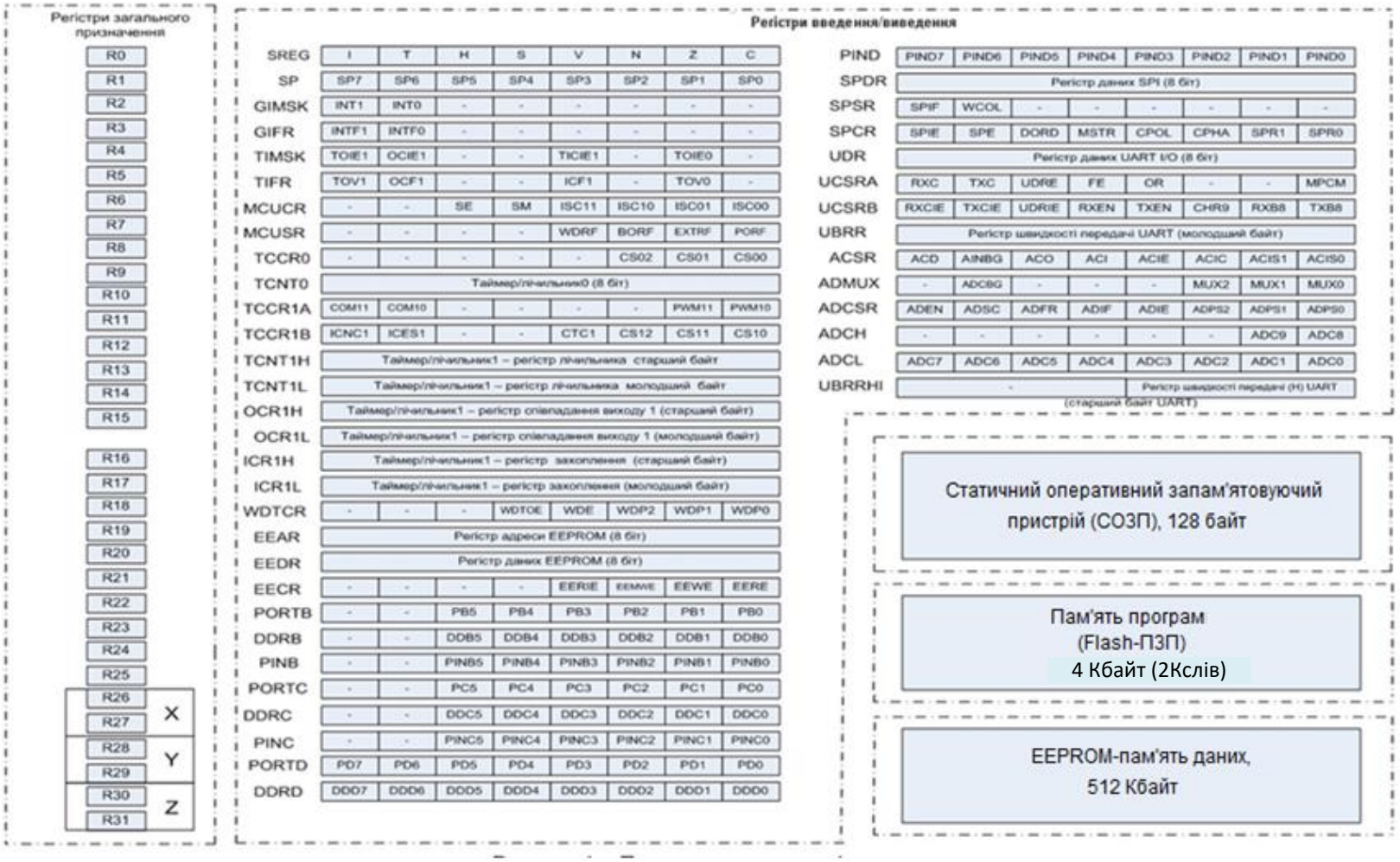


Рис. 3.4. Програмна модель AVR-мікроконтролера

3.2.2. Мнемоніка команди та мнемокод

Мнемоніка команди – представлення коду операції у вигляді сполучення латинських літер, що мають визначений зміст (використовуються англійські слова або скорочення, наприклад, MOV, PUSH, POP, JMP, CLR, NOP). Мнемоніки введені для полегшення написання програм і складають основу мови Асемблера.

Мнемокод включає мнемоніку команди та описання операндів, які беруть участь в операції.

3.2.3. Машинний код команди

Машинний код команди є двійковим кодом команди, який складається з одного або декількох байтів залежно від типу команди конкретного МП/МК.

3.2.4. Операнди

Операндами у мікропроцесорній техніці називають дані, які беруть участь у виконанні тієї чи іншої команди (операції). Залежно від способу адресації операндів дані можуть знаходитися в регістрах, пам’яті, в машинному коді команди і т. ін.

3.2.5. Коментар

Коментар використовується для полегшення читання програми. Компілятор (Асемблер) пропускає коментарі, що показані в програмі через “;” після або до мнемоніки команди, не генеруючи у цьому разі ніякого машинного коду.

3.2.6. Формати команд

3.2.6.1. Команди 16-розрядного мікропроцесора

Основні формати команд 16-розрядного МП, наприклад, і8086, наведено на рис. 3.5.

Команди розміщуються у пам’яті побайтово в комірках з послідовно зростаючими адресами. Вони вибираються з пам’яті словами (по два байти) і завантажуються у чергу команд блока сполучення із системною шиною МП.

Перший байт команди завжди містить код операції, причому в окремих командах його нульовий розряд несе інформацію про довжину операндів (байт чи слово), а перший розряд задає тип джерела і/чи приймача інформації.

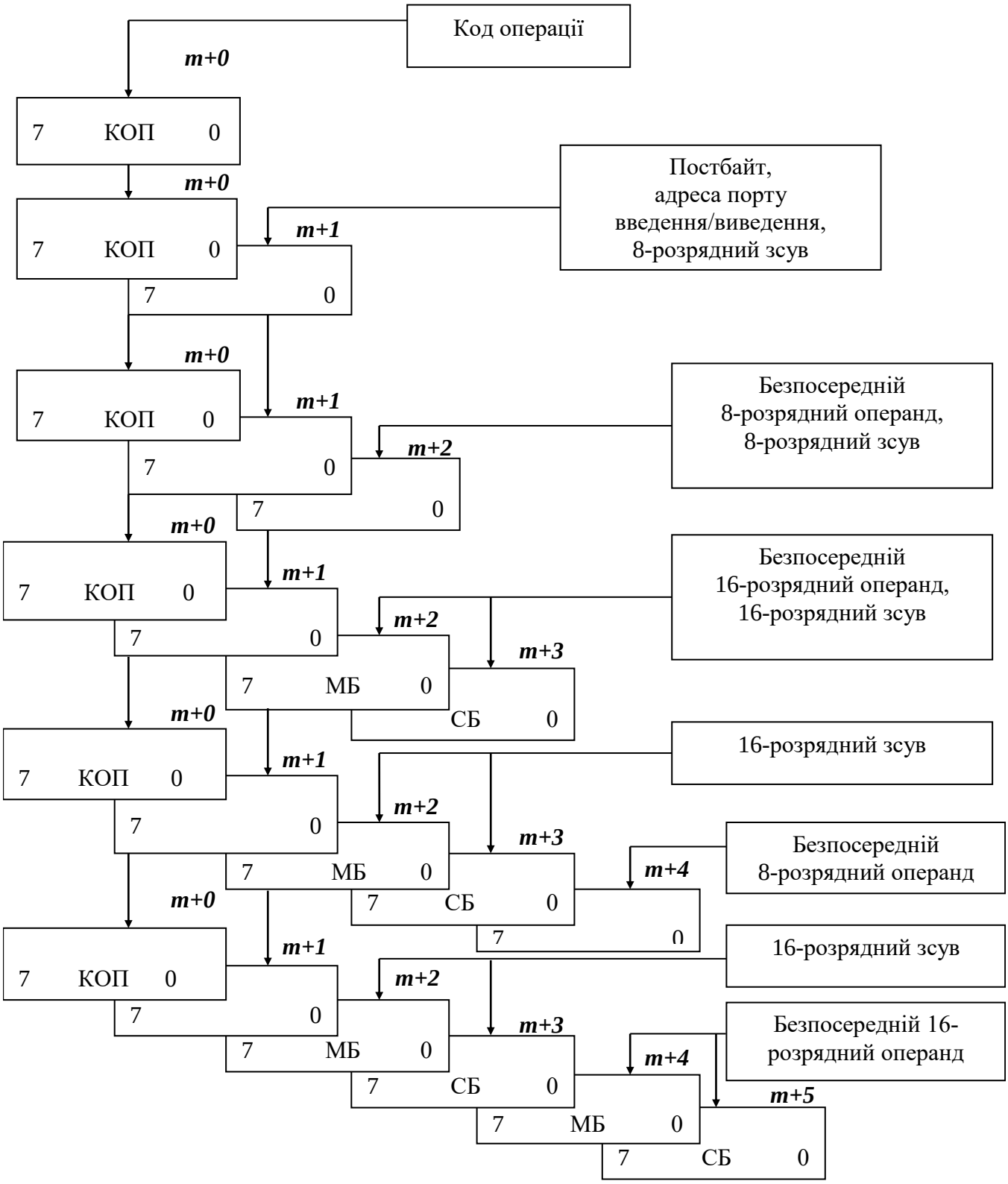


Рис. 3.5. Формати команд 16-розрядного МП

Другий байт у більшій частині команд може бути трьох типів:

– постбайт (див. рис. 3.9), що визначає, звідки брати операнд (операнди) і/чи куди направляти результат операції. У деяких командах розряди 5…3 постбайта (поле reg) використовуються для розширення коду операції команди;
– адреса порту введення/виведення, що задає пряму адресу порту у діапазоні 00H…FFH у двобайтових командах;
– 8-розрядний зсув, що визначає відносну адресу переходу в командах умовної передачі керування.

Наступні байти у довгих командах містять 8-розрядний чи 16-розрядний зсув і/чи безпосередній операнд, також 8-розрядний чи 16-розрядний. Якщо в командах є зсув і безпосередній операнд, то спочатку розміщується зсув, а потім операнд. Для 16-розрядного зсуву, і для 16-розрядного операнда завжди першим розміщується молодший байт, а другим – старший (адреса старшого байта на одиницю більше адреси молодшого). 8-розрядний зсув під час формування ефективної адреси розширюється до 16-розрядного значенням старшого (знакового) розряду. Наприклад, якщо задано байт зсуву 7EH = 01111110B, то він розширюється до 007EH. Якщо задано байт A3H = 10100011B, то він розширюється до FFA3H. Зсув сприймається МП як ціле двійкове число зі знаком у додатковому коді, що перебуває у діапазоні: –128…+127 для 8-розрядного чи: –32768…+32767 для 16-розрядного зсуву.

3.2.6.2. 8-розрядний мікроконтролер сімейства AVR

Більшість МК сімейства AVR мають 14 форматів (типів) базового набору команд, наведених на рис. 3.6.

На рис. 3.6 використано такі умовні позначення: КОП – код операції; K – константа даних; k – адресна константа; b – номер біта в регістрі введення/виведення або регістрового файлу (регістр загального призначення) – 3 біти; s – номер біта в регістрі стану – 3 біти; A – адреса регістра у просторі введення/виведення; q – зміщення для непрямої адресації – 6 біт; d(r) – регістр-приймач (джерело) з області регістрового файлу.

До першого типу належать команди, код операції яких займає всю довжину команди – 16 біт. До цієї групи належить, наприклад, команда LPM – завантаження регістра загального призначення R0 з пам’яті програм за адресою, яка міститься у регістровій парі Z = R31:R30. Такий формат мають також команди IJMP, ICALL, RET, RETI, NOP, SLEEP та WDT.

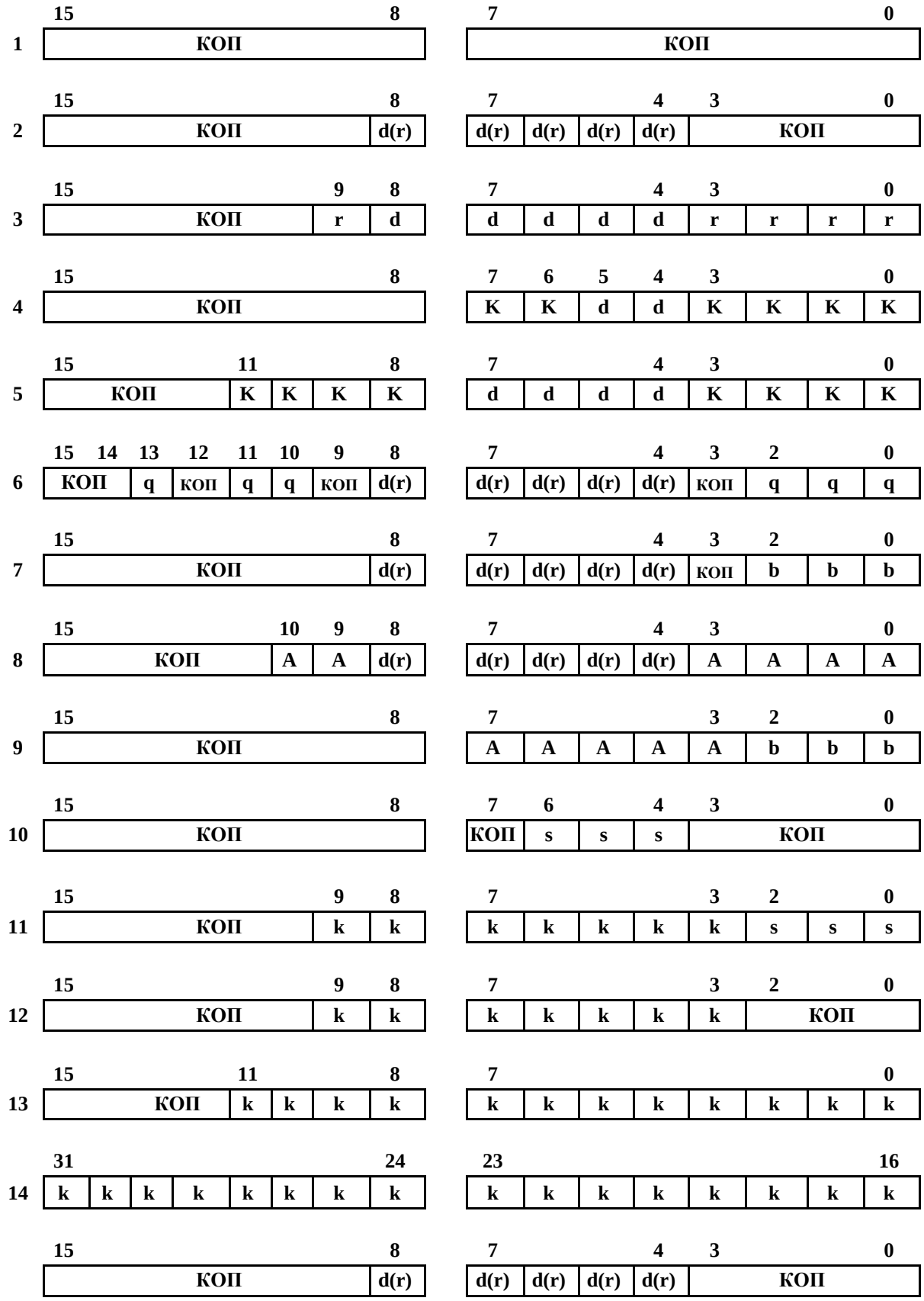


Рис. 3.6. Формати базового набору команд

До другого типу належать команди, які крім коду операції містять п’ятирозрядну адресу одного з регістрів загального призначення. Наприклад, це команди DEC Rd; LD Rd, -X; ST Y, Rr і т. ін.

Третій тип мають команди, в яких адресуються два операнди – регістри загального призначення: Rd – приймач, Rr – джерело. Це, наприклад, команди ADD Rd, Rr; CPSE Rd, Rr; AND Rd, Rr і т. ін

До четвертого типу належать двоопераднді команди, в яких один операнд: K6 = 6 біт (K = 0…63) входить у саму команду – безпосередня адресація, а другим є пара регістрів: R24, R25; R26, R27; R28, R29; R30, R31, які кодуються двома бітами команди – dd: 00 – R24, R25; 01 – R26, R27; 10 – R28, R29; 11 – R30, R31. Наприклад, команди ADIW Rdl, K6; SBIW Rdl, K6, де Rdl = R24/R26/R28/R30.

П’ятий тип мають двоопераднді команди, наприклад, SBCI Rd*, K; ORI Rd*, K, в яких Rd* – один із шістнадцяти регістрів загального призначення (R16…R31), а другий K – 8-бітна константа (безпосередній операнд): $0 \le K \le 255$.

До шостого типу належать команди, в яких один з двох операндів може бути регістром загального призначення (Rd – приймач, Rr – джерело), а другий міститься у статичній пам’яті даних і адресується за допомогою непрямої відносної адресації, коли вміст індексного регістра Y або Z додається до зміщення q ($0 \le q \le 63$). Це, наприклад, команди LDD Rd, z + q; STD Y + q, Rr і т. ін.

Сьомий тип мають команди, в яких адресується один із восьми бітів регістра загального призначення. Наприклад, команди BLD Rd, b; BST Rr, b; SBRC Rr, b і т. ін., де Rd – приймач, Rr – джерело (один з 32-х РЗП), а b = 0…7 – номер біта регістра загального призначення.

До восьмого типу належать команди, в яких одним операндом є регістр загального призначення (Rd – приймач, Rr – джерело), а другий міститься в одному з 64-х регістрів введення/виведення. Наприклад, команди IN Rd, P; OUT P, Rr, де P – адреса регістра введення/виведення (P = 0…63).

Дев’ятий тип мають команди, в яких адресуються окремі біти молодшої половини регістрів введення/виведення (P* = 0…31). Наприклад, команди SBI P*, b, де b = 0…7 – номер біта регістра введення/виведення.

До десятого типу належать команди, в яких адресується один із 8-ми бітів регістра прапорців SREG. Наприклад, команди BSET s; BCLR s, де s = 0…7 – номер біта у регістрі стану.

Одинадцятий тип мають команди умовного переходу залежно від значення вказаних бітів регістра стану. Наприклад, команди BRBS s, k;

BRBC s, k, де s = 0…7 – номер біта регістра стану SREG, а k = 7 біт (–64 ≤ k ≤ 63) під час переходу додається до адреси наступної команди.

Дванадцятий тип мають команди умовного відносного переходу залежно від значень окремих прапорців регістра стану (прапорці адресуються неявно). Наприклад, команди BREQ k; BRCC k і т. ін., де k = 7 біт (–64 ≤ k ≤ 63) під час переходу додається до адреси наступної команди.

До тринадцятого типу належать команди відносного безумовного переходу: RJMP k та відносного безумовного виклику підпрограм: RCALL k, де k = 12 біт. Діапазон переходів та виклику підпрограм: –2048 + 2047.

Чотирнадцятий тип мають команди: LDS Rd, k – пряме завантаження та STS k, Rr – пряме збереження, в яких одним операндом є регістр загального призначення (Rd – приймач, Rr – джерело), а другий міститься у статичній пам'яті даних (регістр загального призначення, регістр введення/виведення та статичний ОЗП). Оскільки k = 16 біт, то кількість комірок статичної пам'яті даних становитиме 2^{16} = 65536.

3.2.7. Формати даних

3.2.7.1. Загальна характеристика даних

Під час роботи з МП або МК необхідно знати не тільки формати команд, які керують роботою МП/МК, але й формати (типи) даних (операндів), що беруть участь у виконанні тієї або іншої команди (операції).

3.2.7.2. Формати (типи) даних 8-розрядного мікропроцесора/мікроконтролера

Формати даних 8-розрядного МП/МК наведено на рис. 3.7.

3.2.7.3. Формати (типи) даних 16-розрядного мікропроцесора

У 16-розрядному МП окрім названих форматів даних використовуються 16-розрядні операнди, які можуть бути (рис. 3.8):

– цілим додатним числом у діапазоні: 0…65535D;
– числом зі знаком у діапазоні: –32768…+32767;
– шістнадцятьма незалежними логічними змінними: X1, X2,…,X16.

16-розрядний МП може виконувати операції над двійково-десятковими числами у запакованому та незапакованому форматах, а також числами, що представлено в коді ASCII.

Окрім того, у ролі операндів можуть бути рядки (ланцюжки) даних, елементами яких можуть бути байти або слова.

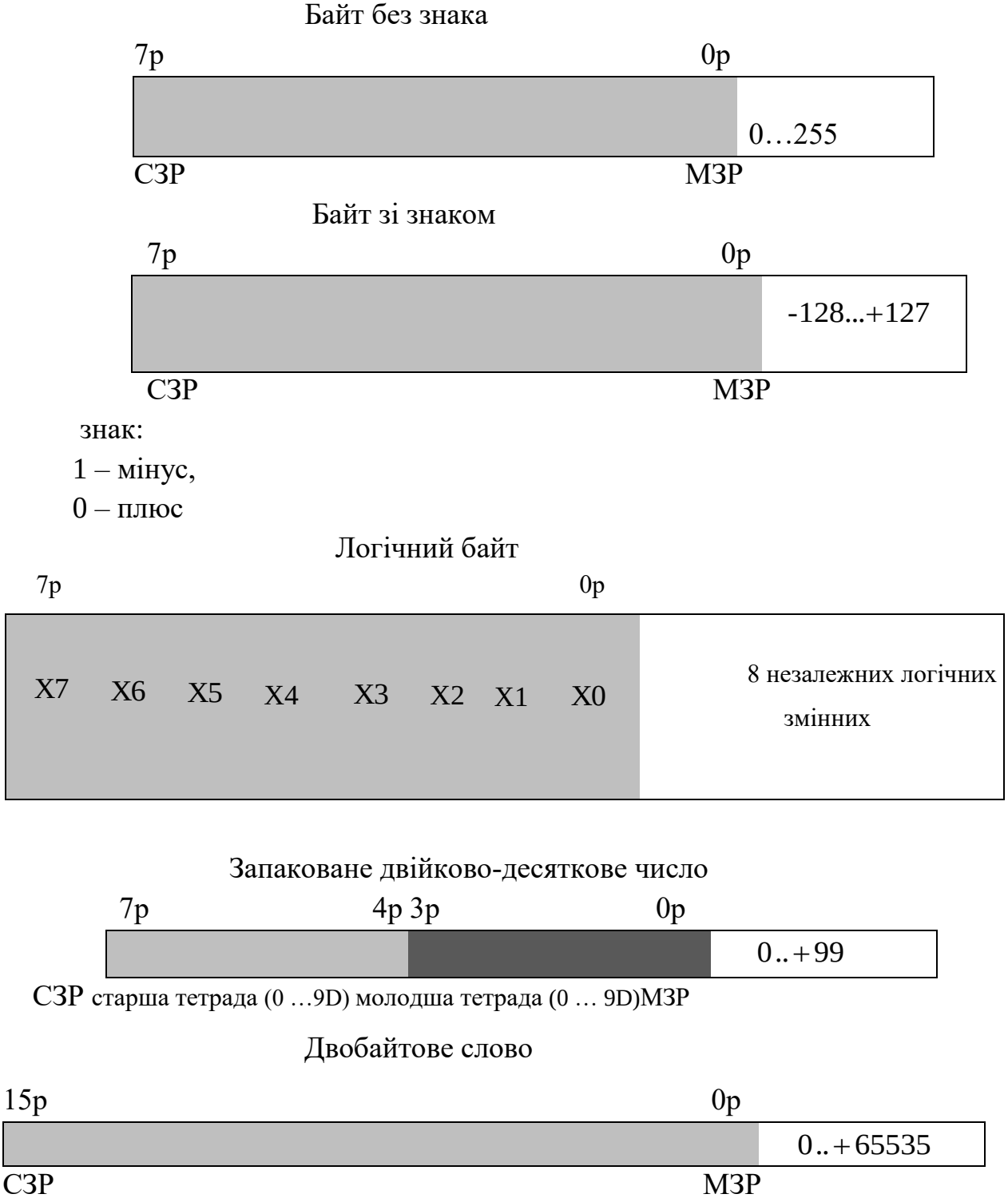


Рис. 3.7. Формати (типи) даних 8-розрядного МП/МК

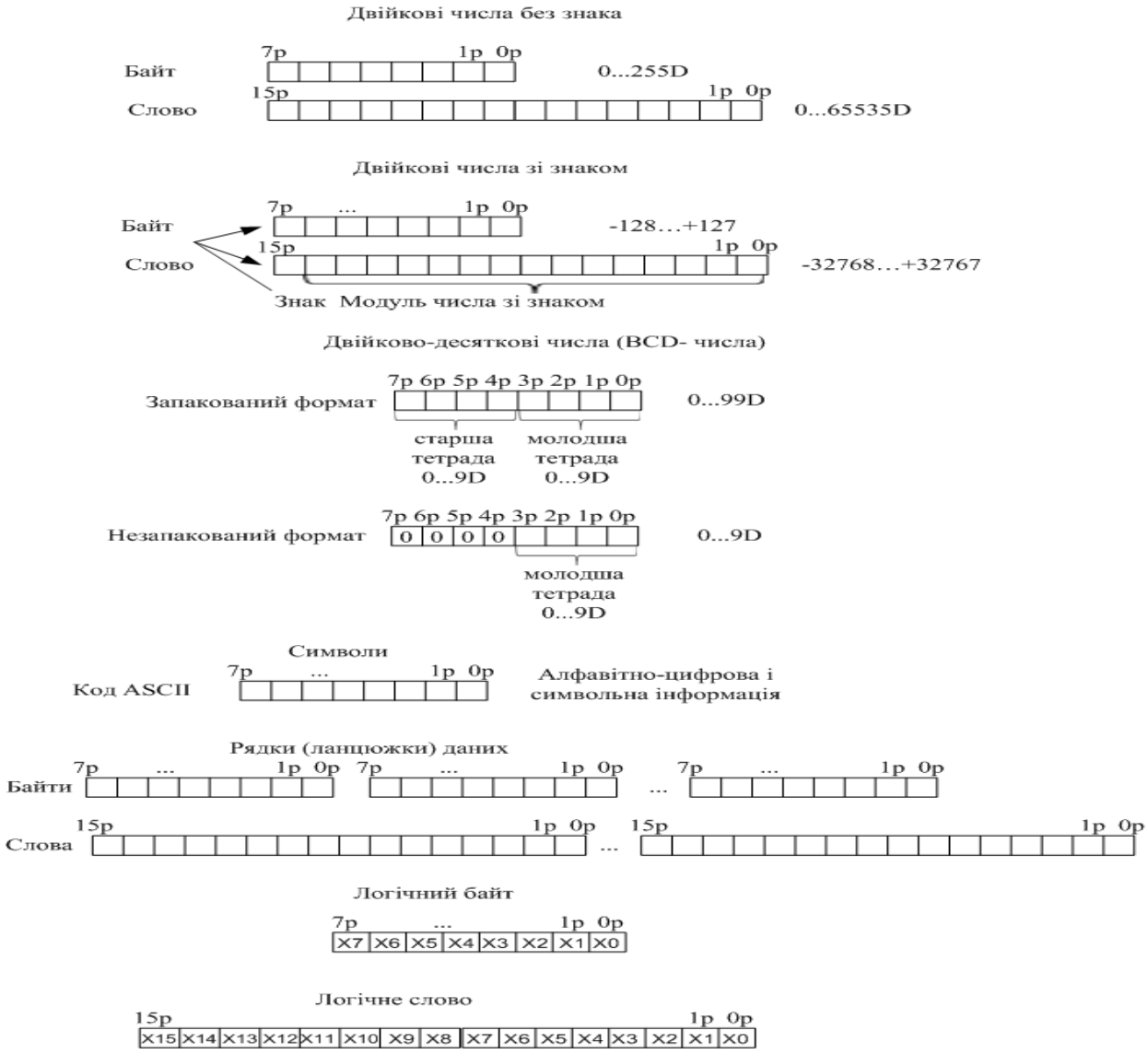


Рис. 3.8. Типи та формати даних МП i8086

3.2.8. Довжина команд у байтах і їх розміщення у пам’яті програм

Довжина команд 16-розрядного МП i8086 складається від одного до шести байт (див. рис. 3.5), які розміщуються в пам’яті так само, як у 8-розрядному МП. Так, наприклад, байти найдовшої команди (6 байт) розміщуються в такому порядку: 1-й байт – код операції; 2-й байт – постбайт (використовується для адресації операндів); 3-й байт – молодший байт зсуву, який бере участь в обчисленні ефективної адреси операнда, яка формується згідно з форматом постбайта; 4-й байт – старший байт зсуву; 5-й байт – молодший байт безпосереднього операнда; 6-й байт – старший байт безпосереднього операнда.

Відповідно до наведеного рис. 3.6 команди МК сімейства AVR мають довжину – одне слово (2 байти), за винятком команди чотирнадцятого типу, яка має довжину 2 слова (4 байти).

Програма, яка керує роботою МК, послідовно, команда за командою, розміщується в сусідніх комірках пам’яті у порядку зростання їх адрес. Адреса команди, яка повинна виконуватись, знаходиться у програмному лічильнику. Пристрій керування МК на основі прочитаного коду операції, що міститься в першому байті команди, визначає, скільки ще байтів міститься в команді, та керує їх читанням з пам’яті за допомогою збільшення на одиницю адреси, яка видається у разі кожного звернення до запам’ятовувального пристрою. Після читання з пам’яті чергової команди МК формує адресу коду операції наступної команди.

3.2.9. Вплив команд на прапорці

Як зазначалося раніше, одним з основних регістрів МП/МК є регістр прапорців (ознак).

Прапорці регістра ознак встановлюються під час виконання низки команд МП/МК. Під час опису системи команд, яка зазвичай оформлюється у вигляді таблиці, в одній з колонок показується вплив окремих команд на ті або інші прапорці.

Якщо команда не змінює прапорець, то ставлять риску, якщо змінює, то ставлять плюс або *x*, а якщо встановлює у будь-який стан, то пишуть нуль (0) або одиницю (1). Зазвичай команди пересилань не змінюють прапорці, окрім команд, які пересилають дані у регістр прапорців. Частіше прапорці змінюють арифметичні, логічні команди і команди порівняння.

Окремі прапорці (ознаки) аналізуються під час виконання команд умовних переходів, виклику і повернення з підпрограм, що дозволяє передавати керування в програмі залежно від поточного значення прапорців.

Вплив на прапорці команд МК типу AT89C51 сімейства МК51 показано у [2].

Для AVR-мікроконтролерів під час опису системи команд, яка зазвичай оформлюється у вигляді таблиці, в одній з колонок показується вплив окремих команд на ті або інші прапорці (див. табл. 3.16).

Вплив команд на прапорці в 16-му МП i8086 зображено у табл. 3.1, 3.2.

3.2.10. Час виконання команд

Час, який необхідний для зчитування команди з пам’яті та її виконання, називається командним циклом. Командний цикл для МП типу i8080 складається з декількох машинних циклів, кількість яких залежить від

складності виконуваної команди і дорівнює кількості звернень МП до пам’яті або одного з пристроїв введення/виведення [2].

Таблиця 3.1. **Команди МП i8086, які впливають на прапорці стану**

Операція	**Команди**	**Прапорці стану**					
		OF	**CF**	**AF**	**SF**	**ZF**	**PF**
Додавання, віднімання, порівняння	ADD ADC SUB SBC	+	+	+	+	+	+
	CMP NEG CMPS SCAS	+	+	+	+	+	+
	INC DEC	+	-	+	+	+	+
Множення	MUL IMUL	+	+	?	?	?	?
Ділення	DIV IDIV	?	?	?	?	?	?
Десяткова корекція	DAA DAS	?	+	+	+	+	+
	AAA AAS	?	+	+	?	?	?
	AAM AAD	?	?	?	+	+	+
Зсуви	AND OR XOR TEST	0	0	?	+	+	+
	SHL SHR [1)]	+	+	?	+	+	+
	SHL SHR [2)]	?	+	?	+	+	+
	SAR	+	+	?	+	+	+
	ROL ROR RCL RCR	+	+	-	-	-	-
	ROL ROR RCL RCR	?	+	-	-	-	-
Відновлення прапорців	POPF IRET	+	+	+	+	+	+
	SAHF	-	+	+	+	+	+
Керування прапорцем перенесення	STC	-	1	-	-	-	-
	CLC	-	0	-	-	-	-
	CMC	-	Г	-	-	-	-

Примітка. «+» – результат операції впливає на прапорець; «–» – не впливає; 1 – встановлюється в одиницю ; 0 – скидається в нуль; Г – інвертується; ? – невизначений;

[1)] – зсув на один розряд; [2)] –зсув на декілька розрядів.

Таблиця 3.2. **Вплив команд МП i8086 на прапорці керування**

Операції	Команди	Прапорці керування		
		DF	IF	TF
Відновлення прапорців	POPF IRET	+	+	+
Переривання	INT INTO	-	0	0
Керування прапорцями	STD	1	–	–
	CLD	0	–	–
	STI	–	1	–
	CLI	–	0	–

Примітка. «+» – результат операції впливає на прапорець; «–» – не впливає; 1 – встановлюється в одиницю; 0 – скидається в нуль.

Машинний цикл складається з деякої кількості елементарних дій, що називаються станами (тактами), і виконуються за один період системного сигналу тактування.

В одній з колонок таблиці, які описують систему команд, записують кількість тактів, що необхідно для виконання команди. Знаючи кількість тактів (n_T) і значення тактової частоти МП/МК (f_T) можна визначити час виконання команди $t_K = n_T/f_T$.

Більшість команд МК типу AT89C51 виконуються за 1 або 2 машинних цикли, а множення і ділення – за чотири машинних цикли. Якщо, наприклад, f_{BQ} = 12 МГц, то $T_{BQ} = 1/12 \cdot 10^{-6}$ с, $T_{МЦ} = 1/12 \cdot 12 \cdot 10^{-6} = 1$ мкс, а окремі команди виконуються за 1, 2 або 4 мкс.

В AVR-мікроконтролерах окремі команди виконуються за 1, 2, 3 або 4 такти. Тривалість одного такту дорівнює одному періоду тактової частоти f_{BQ}.

Для 16-розрядного МП, наприклад, i8086 одна з колонок таблиць, в яких наведено команди та їх характеристики, містить інформацію про час виконання команд у кількості тактів. У цій колонці може стояти або конкретне число, що відображає час виконання цієї команди в тактах, або наводиться вираз:

$$n + T_{BA},$$

де n – конкретне значення; T_{BA} – додаткова кількість тактів, необхідна для обчислення ефективної (виконавчої) адреси операнда в пам’яті.

Це значення визначається згідно з наведеною нижче табл. 3.3 за відомим способом адресації операнда, що застосовується у відповідній команді.

Таблиця 3.3. **Визначення часу обчислення виконавчої (ефективної) адреси T_{BA} (T_{EA})**

Значення T_{EA} у тактах залежно від способу адресації операндів							
mod	**r/m**	**Тип адресації**	**Такти**	**mod**	**r/m**	**Тип адресації**	**Такти**
00	110	Пряма	6	00	000	Базово-індексна (BX)+(SI)	7
00	111	Непряма (BX)	5	00	001	(BX)+(DI)	8
				00	010	(BP)+(SI)	8
				00	011	(BP)+(DI)	7
00	100	Непряма (SI)	5				
00	101	(DI)	5			Базово-індексна з зсувом	
01	110	Базова (BP)+DATA8	9	01	000	(BX)+(SI)+DATA8	11
10	110	(BP)+DATA16	9	10	000	(BX)+(SI)+ DATA16	11
01	111	(BX)+DATA8	9	01	001	(BX)+(DI)+ DATA8	12
10	111	(BX)+DATA16	9	10	001	(BX)+(DI)+ DATA16	12
01	100	Індексна (SI)+DATA8	9	01	010	(BP)+(SI)+ DATA8	12
10	100	(SI)+DATA16	9	10	010	(BP)+(SI)+ DATA16	12
01	101	(DI)+DATA8	9	01	011	(BP)+(DI)+ DATA8	11
10	101	(DI)+DATA16	9	10	011	(BP)+(DI)+ DATA16	11

3.3. Способи адресації операндів

3.3.1. Визначення способів адресації

Тип звернення (адресації) до операндів (даних, що беруть участь в операції) називають способом адресації. До способів адресації операндів також належить те, як у командах передавання керування дається вказівка на адресу переходу.

3.3.2. 16-розрядний мікропроцесор

3.3.2.1. Види способів адресації

У 16-розрядному МП, наприклад, і8086 застосовуються такі способи адресації операндів: неявна, регістрова, безпосередня, пряма, непряма, базова, індексна, базово-індексна, базово-індексна зі зміщенням, стекова, відносна, адресація рядків та адресація портів введення/виведення [2].

У багатьох командах МП і8086 для адресації операндів використовується постбайт, формат якого розглянуто нижче.

3.3.2.2. Формат постбайта та його вплив на обчислення ефективної (виконавчої) адреси

Адреса, що обчислюється МП для адресації операнда в пам’яті, називається ефективною (виконавчою) адресою і позначається виконавчою адресою (ефективною адресою). Виконавча адреса (ефективна адреса) являє собою зсув у сегменті (ціле беззнакове число) і обчислюється залежно від способу адресації, що для багатьох команд визначається постбайтом (рис. 3.9).

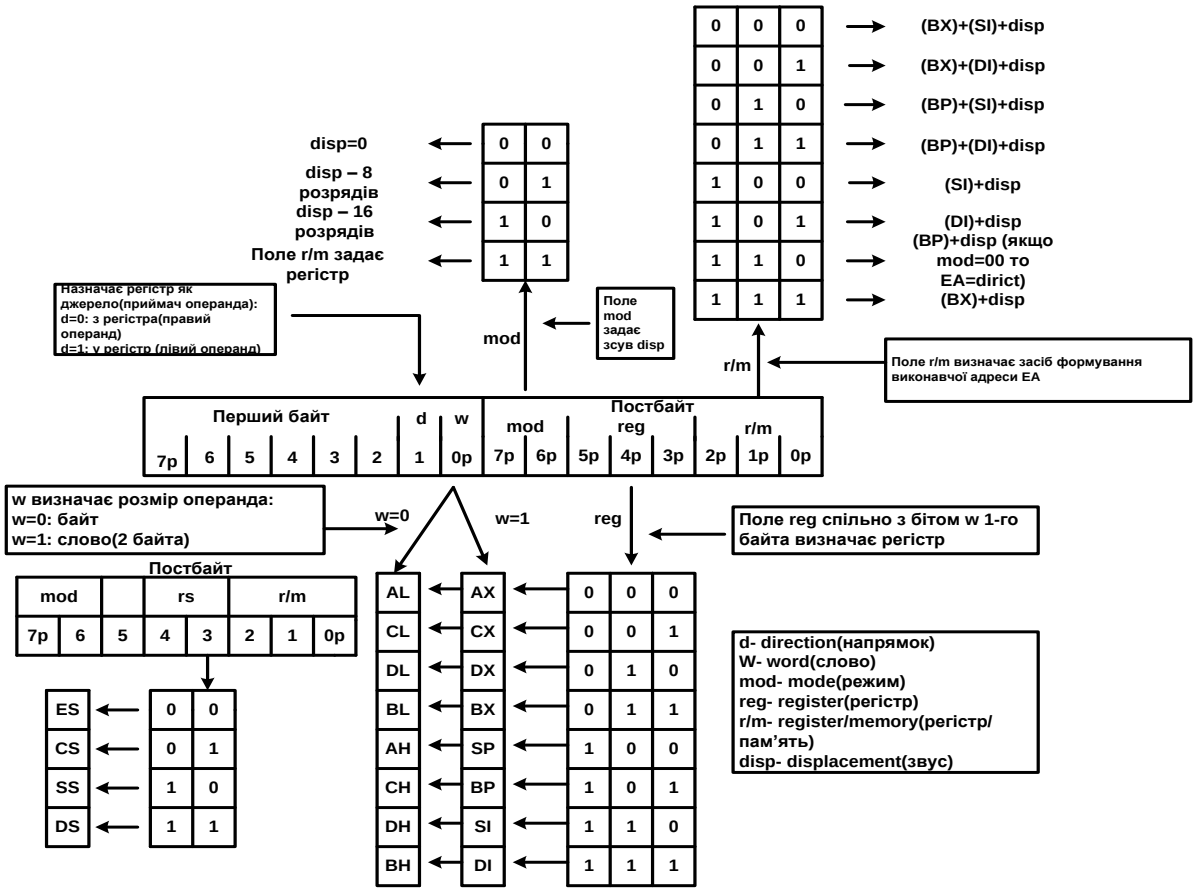


Рис. 3.9. Формат постбайта команди та його вплив на спосіб адресації

Дворозрядне поле mod (md) разом з полем r/m задають спосіб формування виконавчої адреси. Поле mod визначає зсув (disp), а r/m – регістри, вміст яких бере участь в обчисленні ефективної адреси. Виконавча адреса є 16-розрядним зсувом у сегменті пам'яті (сегментах даних, стеку, коду та екстра).

Якщо під час обчислення ефективної адреси виникає перенесення, то воно ігнорується відповідно до правил додавання чисел у додатковому коді.

Таким чином фізична адреса завжди перебуває у межах одного сегмента. Окрім ефективної адреси постбайт може задавати адреси регістра/регістрів, що беруть участь в операції. Для цього 3-розрядне поле reg разом з полем w першого байта команди задає або 8-розрядний регістр (якщо w = 0), або 16-розрядний регістр (якщо w = 1). Наприклад, reg = 110 і w = 1 визначають регістр SI, а той же код у полі reg, якщо w = 0, указує на регістр DH (старший байт регістра DX).

Якщо поле mod = 11, то поле r/m аналогічно полю reg задає регістр.

У табл. 3.4 наведено різні варіанти адресації за допомогою постбайта.

Таблиця 3.4. **Режими адресації за допомогою постбайта**

mod / r/m	0 0	0 1	1 0	11,r/m=reg	
				W=0	W=1
000	(BX) + (SI)	(BX)+(SI)+disp8	(BX)+(SI)+disp16	AL	AX
001	(BX) + (DI)	(BX)+(DI)+disp8	(BX)+(DI)+disp16	CL	CX
010	(BP) + (SI)	(BP)+(SI)+disp8	(BP)+(SI)+disp16	DL	DX
011	(BP) + (DI)	(BP)+(DI)+disp8	(BP)+(DI)+disp16	BL	BX
100	(SI)	(SI) + disp8	(SI) + disp16	AH	SP
101	(DI)	(DI) + disp8	(DI) + disp16	CH	BP
110	direct = disp16	(BP) + disp8	(BP) + disp16	DH	SI
111	(BX)	(BX) + disp8	(BX) + disp16	BH	DI

Постбайт визначає дві адреси: регістра та ефективної адреси. У командах пересилання, наприклад, один з них може бути адресою джерела інформації, другий – адресою приймача. Конкретний вибір регістра, що задається полем reg в якості приймача/джерела інформації, залежить від вмісту біта d першого байта команди. Якщо d = 0 поле reg задає джерело, а ефективна адреса – приймач. Якщо d = 1 поле reg задає приймача, а ефективна адреса – джерело.

У двомісних арифметико-логічних операціях поле reg, якщо d = 1, задає лівий операнд і приймач результату, а ефективна адреса – правий операнд.

Якщо d = 0, ефективна адреса задає лівий операнд і приймач результату, а reg – правий операнд.

У низці команд (із другим безпосереднім операндом або з одним операндом) поле reg використовується для розширення коду операції. У цьому випадку трактувати це поле як адресу регістра не можна.

3.3.2.3. Неявна адресація

Під час неявної адресації в команді відсутня адреса операнда в явному вигляді. Інформацію про потрібну адресу операнда МП одержує з коду операції команди, тому вважають, що операнд адресується неявно. Так, наприклад, у розглянутому процесорі можуть адресуватися: регістр-акумулятор, регістр прапорців і т. ін.

Наприклад, команда XCHG AX, BX; AX↔BX здійснює обмін словами між двома регістрами AX і BX. З машинного коду команди

$$\underbrace{1\,0\,0\,1\,0\,0\,1\,1}_{\text{BX}} = 93\text{H}$$

можна побачити, що один з регістрів – AX – адресується неявно (за кодом операції), а другий – BX – адресується за допомогою регістрової адресації (див. нижче).

3.3.2.4. Регістрова адресація

Під час регістрової адресації операнд міститься в одному з регістрів МП.

Код регістра вказується або у першому байті команди, або у постбайті. Команди такого типу найкоротші та виконуються за мінімальний час.

На рис. 3.10 наведено приклади команд із регістровою адресацією.

Регістри загального призначення можуть бути джерелами операндів, їх приймачами чи обома одночасно. Сегментні регістри можуть використовуватися як джерела або приймачі операндів, тому пересилати вміст одного із сегментних регістрів в інший однією командою неможливо.

Для виконання цієї задачі необхідно використовувати один з регістрів загального призначення як проміжний регістр.

Наприклад, команда INC reg (рис. 3.10*а*) виконує інкремент вмісту одного з 16-розрядних регістрів загального призначення: AX, CX, DX, BX, SP, BP, SI чи DI.

У поле reg компілятор розміщує 3-розрядний код регістра, що вказується у мнемоніці команди.

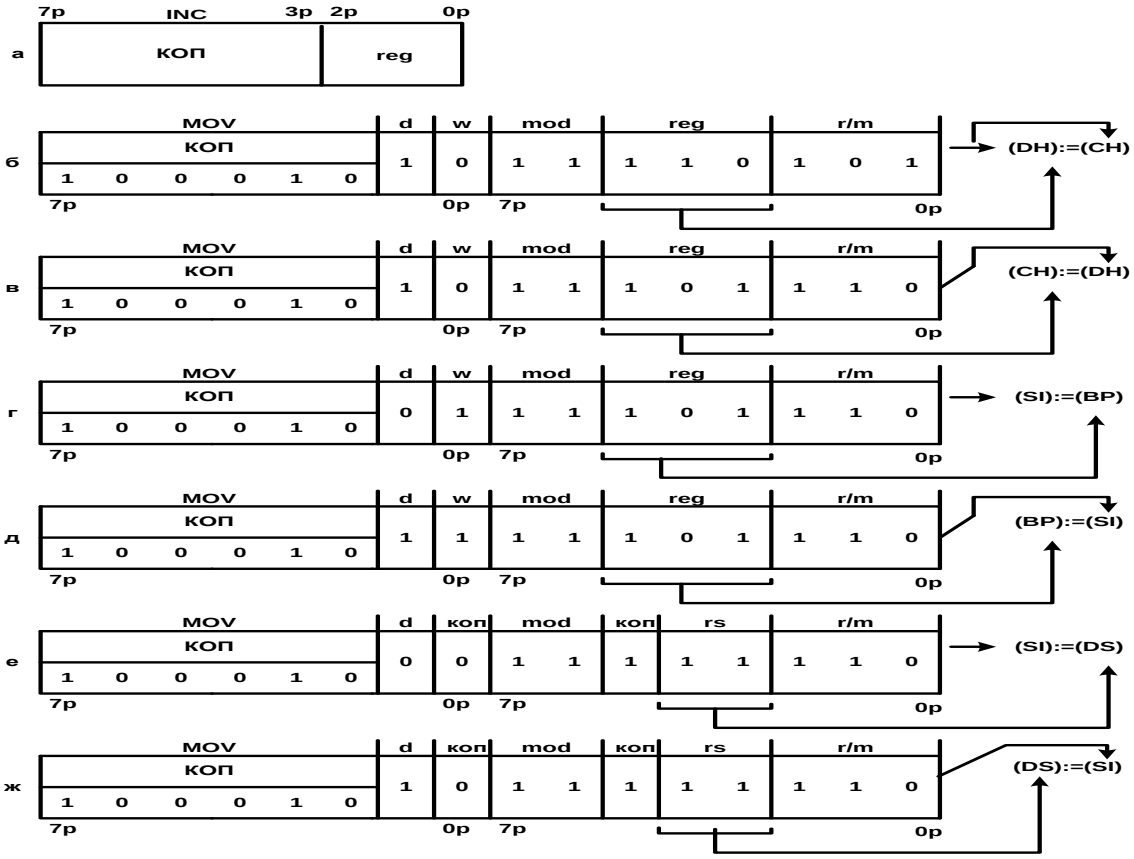


Рис. 3.10. Приклади команд з регістровою адресацією

Наприклад, мнемокоду INC BX буде відповідати машинний код

$$\underbrace{01000}_{\text{КОП команди}}\ \underbrace{011}_{\text{код регістра BX}}\ .$$

На рис. 3.10*б* показано машинний код команди MOV DH, CH; DH ← CH. У цій команді біт w = 0, оскільки операнд має довжину 8 біт. Біт d = 1 вказує на те, що операнд – регістр DH, що адресується полем reg постбайта, є приймачем інформації (у команді пересилання стоїть зліва).

На рис. 3.10*в* наведено машинний код аналогічної команди, де операнди помінялися місцями: MOV CH, DH; CH ← DH.

На рис. 3.10*г, д* відповідно показано машинні коди команд: MOV SI, BP; SI ← BP та MOV BP, SI; BP ← SI. Довжина операндів дорівнює шістнадцяти бітам, тому поле w = 1.

На рис. 3.10*е, ж* відповідно наведено коди команд: MOV SI, DS; SI ← DS і MOV DS, SI; DS ← SI, в яких один з операндів знаходиться в сегментному регістрі DS.

3.3.2.5. Безпосередня адресація

Під час безпосередньої адресації операнд (data) входить у саму команду і витягується з пам'яті під час виконання команди відразу ж за її кодом операції.

Безпосередні операнди можуть мати довжину 8 чи 16 біт (рис. 3.11).

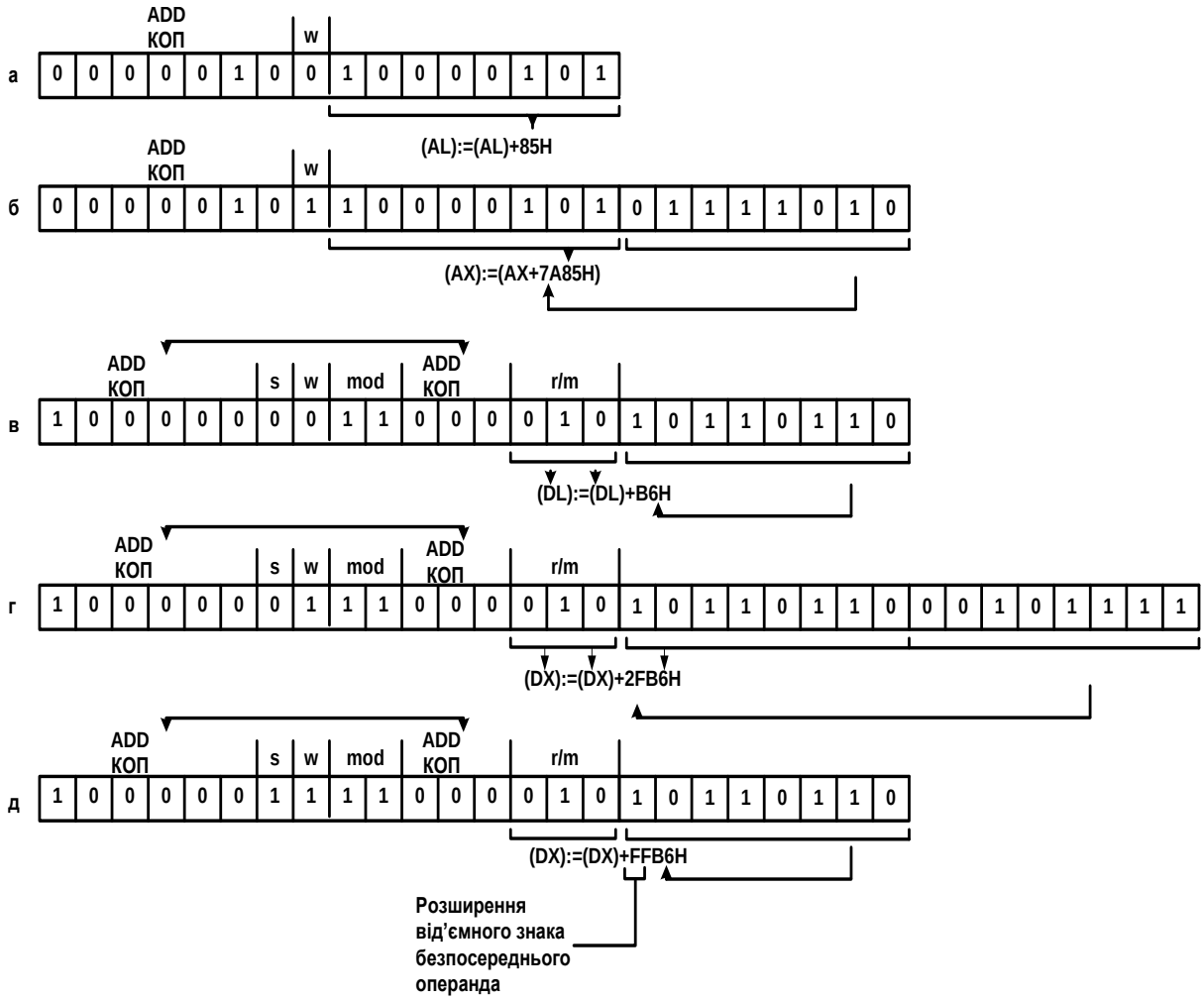


Рис. 3.11. Приклади команд з безпосередньою адресацією

На рис. 3.11*а* показано машинний код команди додавання ADD AL, 85H; AL ← AL + 85H. Безпосередній операнд 85H додається до вмісту акумулятора AL і результат поміщається в AL, який адресується неявно. Оскільки у мнемокоді команди зазначено 8-розрядний регістр AL, то компілятор встановив біт W = 0.

На рис. 3.11*б* наведено машинний код команди додавання ADD AX, 7A85H; AX ← AX + 7A85H. Безпосередній операнд 7A85H підсумовується з вмістом акумулятора AX, що адресується неявно, і результат додавання розміщується в AX. Оскільки у мнемокоді команди зазначено 16-розрядний регістр AX, компілятор встановлює біт W = 1. Команда має

довжину 3 байти, що зберігаються в пам'яті в такому порядку: код операції, молодший байт безпосереднього операнду, старший байт безпосереднього операнду.

На рис. 3.11*в* показано машинний код команди додавання ADD DL, 0B6H; DL ← DL + 0B6H. Безпосередній операнд 0B6H додається до вмісту регістра DL і результат операції розміщується в DL. Другий операнд (регістр DL) адресується за допомогою полів mod = 11 і r/m = 010 постбайта.

Через те, що поле reg у цьому випадку не використовується, воно доповнює код операції команди.

Оскільки в мнемокоді команди зазначено 8-розрядний регістр DL, то компілятор установить біти $S = W = 0$.

На рис. 3.11*г* наведено машинний код команди додавання ADD DX, 2FB6H; DX ← DX + 2FB6H. Безпосередній 16-розрядний операнд 2FB6H додається до вмісту 16-розрядного регістра DX і результат розміщується в DX. Другий операнд (регістр DX) адресується за допомогою полів mod = 11 і r/m = 010 постбайта. Через те, що поле reg у цьому випадку для адресації не використовується, воно доповнює код операції команди. Оскільки у мнемокоді команди обидва операнди зазначені як 16-розрядні, то компілятор встановлює біти: $S = 0$, $W = 1$.

На рис. 3.11*д* показано машинний код команди додавання ADD DX, 0FFB6H; DX ← DX + 0FFB6H. Вміст регістра DX, що адресується за допомогою полів mod = 11 і r/m = 010 постбайта, додається до 16-розрядного безпосереднього операнда. Старший байт безпосереднього операнда – FFH є розширенням знакового розряду молодшого байта, якщо розглядати його як число зі знаком (B6H = 10110110B).

У машинному коді команди розміщується тільки молодший байт B6, який під час виконання команди розширюється до 16-розрядного операнда значенням знакового (старшого) розряду безпосереднього операнда (у нашому прикладі дорівнює 1). У цьому випадку компілятор встановлює біти $S = W = 1$, що вказує процесору на необхідність виконання описаних вище дій. Застосований механізм дозволяє економити одну комірку пам'яті.

Безпосередня адресація не може використовуватися для завантаження констант у сегментні регістри і включення констант у стек. У цьому випадку використовується проміжне завантаження констант в один з регістрів загального призначення. Зазначимо, що у разі запису констант (безпосередніх операндів) у шістнадцятковому коді мовою Асемблера перед константою, що починається з букви, повинен стояти нуль. Безпосередні операнди

застосовуються для ініціалізації регістрів загального призначення та комірок пам’яті, як еталонні значення у командах порівняння і т. ін.

3.3.2.6. Пряма адресація

Під час прямої адресації ефективна адреса операнда міститься у машинному коді самої команди (рис. 3.12).

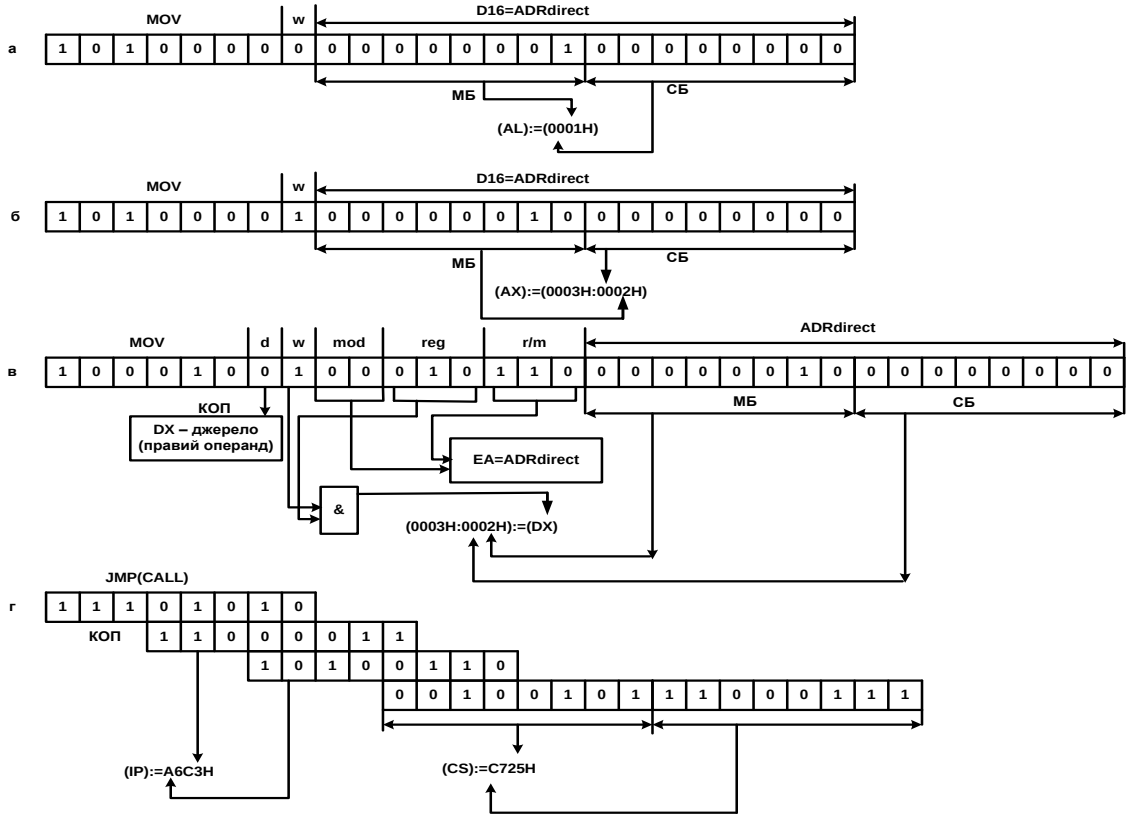


Рис. 3.12. Приклади команд з прямою адресацією

На рис. 3.12*а* наведено машинний код команди MOV AL, [dataByte]; AL ← M(DS ·16 + offset dataByte).

Дана команда пересилає в акумулятор AL значення 8-розрядної змінної з ім’ям dataByte, що розміщена в сегменті даних зі зміщенням offset dataByte відносно початку сегмента даних, що в прикладі дорівнює 0001H. У машинному коді цієї команди адреса (зміщення) змінної у сегменті даних зберігається у другому і третьому байтах команди.

Рис. 3.12*б* містить машинний код команди MOV AX, [dataWord]; AL ← M(DS ·16 + offset dataWord), AH ← M(DS ·16 + offset dataWord+1).

У цій команді в регістр AX пересилається значення 16-розрядної змінної dataWord, розміщеної у двох сусідніх комірках сегмента даних зі зміщенням відносно початку сегмента: offset DataWord. Це зміщення в прикладі дорівнює

0002H та адресує молодший байт змінної, що пересилається в молодшу частину акумулятора AX (AL).

Зміщення offset dataWord + 1, яке дорівнює 0003H, адресує старший байт змінної, що пересилається у старшу частину акумулятора AX (AH).

Символічні імена dataWord, dataByte і т. ін. мають бути визначені у програмі мовою Асемблера.

У розглянутих двох прикладах пряма адреса (зміщення у сегменті даних) була розміщена у другому і третьому байтах машинного коду команди.

На рис. 3.12*в* показано машинний код команди MOV [dataWord], DX; M(DSx16 + offset dataWord)←DL, M(DSx16 + offset dataWord + 1) ← DH.

У цій команді пряма адресація виконується з використанням постбайта, в якому поле mod = 00, а r/m = 110. За постбайтом розміщується 16-розрядна пряма адреса (зміщення) операнда у сегменті даних.

Ця команда пересилає вміст 16-розрядного регістра DX у дві комірки сегмента даних, виділених для збереження змінної dataWord. Молодший байт цієї змінної має зміщення offset dataWord у сегменті даних, що дорівнює 0002H, а старший байт – зміщення у сегменті даних – 0003H.

Регістр DX адресується за допомогою регістрової адресації (поле reg постбайта, що дорівнює 010, та w = 1).

Слід зазначити особливість прямої адресації у разі написання програм мовою Асемблера. У деяких командах, наприклад, INC, MUL, ROR і т. ін. за іменем змінної неможливо визначити розмір операнда (змінної) – байт або слово. Це приклади «анонімних» звернень до пам’яті. Щоб усунути цю неоднозначність, компілятору (Асемблеру) потрібна додаткова інформація, яку повідомляє спеціальний оператор. Якщо, наприклад, необхідно зробити інкремент 8-розрядної змінної dataByte, необхідно записати INC [Byte dataByte]. Щоб зсунути на один розряд вправо 16-розрядну змінну data Word необхідно записати ROR [Word data Word], 1.

Пряма адреса в командах передачі керування JMP, CALL (наприклад, JMP NEAR LABEL) задає адресу (зсув) у поточному сегменті команд CS.

Існують команди передачі керування з довгою (4 байти) прямою адресою для переходів між сегментами. В таких командах, наприклад, JMP FAR LABEL (рис. 3.12*г*) другий і третій байти визначають новий вміст регістра IP, а четвертий і п’ятий – адресу нового сегмента команд CS.

3.3.2.7. Непряма адресація

У цьому режимі ефективна адреса операнда знаходиться в одному з регістрів BX, SI, DI або BP (рис. 3.13).

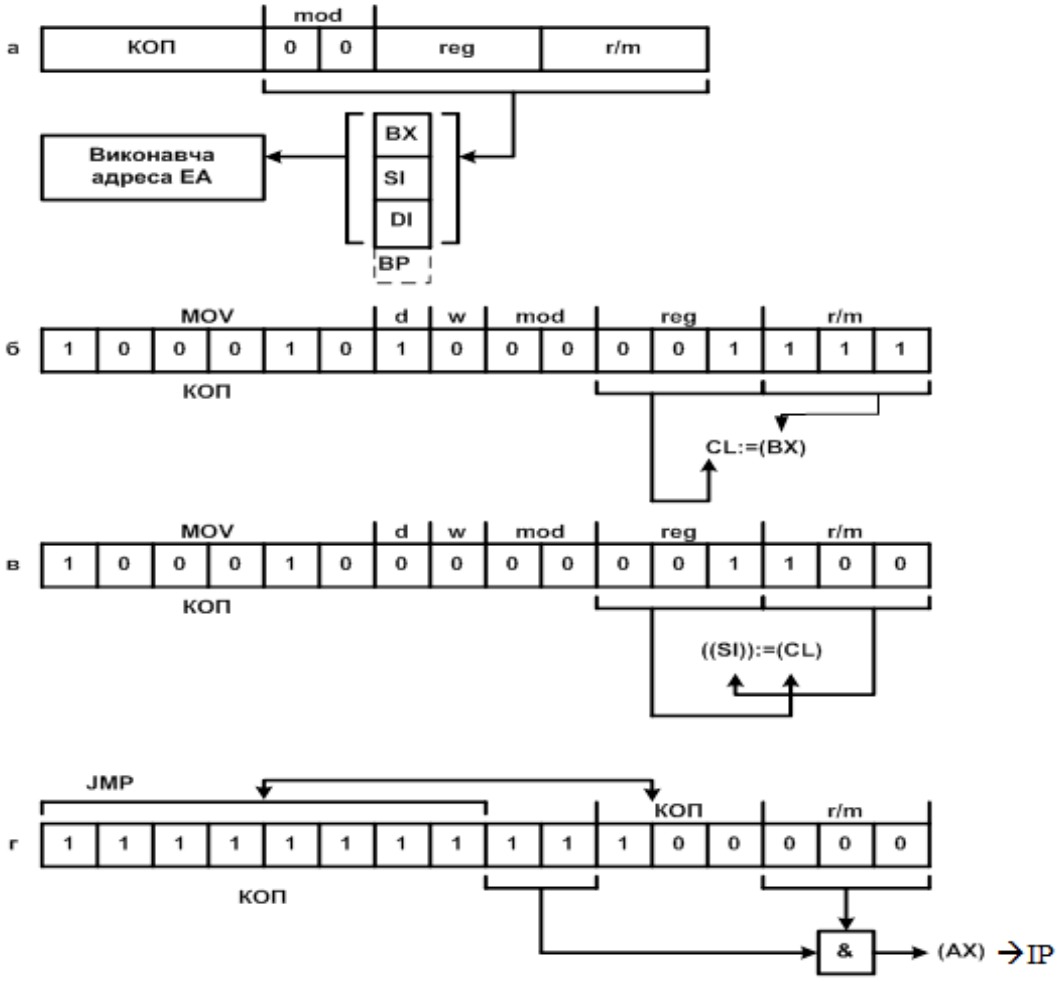


Рис. 3.13. Приклади команд з непрямою адресацією

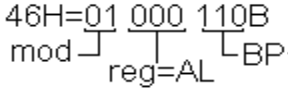
За допомогою зміни вмісту названих регістрів можна обчислювати і змінювати адреси пам’яті під час виконання програми, тобто одна та сама команда може звертатися до різноманітних комірок пам’яті. Використання регістра як джерела ефективної адреси вказується взяттям його імені у квадратні дужки, наприклад: ADD AX, [DI]; AX ← AX + M (DS ·16 + DI).

Непряма адресація реалізується за допомогою постбайта (рис. 3.13*б, в*). Якщо непряма адреса зберігається в одному із регістрів BX, SI або DI (рис. 3.13*а*), то поле mod = 00, r/m = 100 (РГ SI), 101(регістр DI) або 111 (регістр BX).

Регістр BP для непрямої адресації використовувати не рекомендовано через те, що довжина машинного коду команди збільшується на один байт.

Наприклад, якщо для збереження непрямої адреси використовується регістр BP та виконується команда OR AL, [BP]; AL ← AL ∨ M (SS ·16 + BP), то компілятор сформує такі три байти:

1-й байт (КОП) – 0АН = 00001010В;

2-й байт (постбайт) – 

;

3-й байт (8-розрядний зсув) – disp8 = 00Н = 00000000В.

Оскільки комбінація: mod = 00, r/m = 100 використовується у разі прямої адресації (див. рис. 3.9), то компілятор встановив комбінацію: mod = 01 і r/m = 110, що вказує на те, що ефективна адреса операнда ЕА = ВР + disp8.

Зсув disp8 входить у третій байт машинного коду команди, дорівнює 00Н та збільшує довжину машинного коду на один байт.

На рис. 3.13*б* наведено машинний код команди MOV CL, [BX]; CL ← M (DS·16 + BX).

На рис. 3.13*в* показано машинний код команди MOV [SI], CL; M(DS ·16 + SI) ← CL.

Непряма адресація може використовуватися також у командах передачі керування (JMP, CALL).

Вміст будь-якого з 16-розрядних регістрів загального призначення (AX, BX, CX, DX, SI, DI, BP, SP) у цьому випадку є зсувом у кодовому сегменті CS, тобто вміст названого регістра під час виконання команди завантажується у вказівник команд (програмний лічильник) – IP.

На рис. 3.13*г* показано машинний код команди JMP AX, яка передає керування команді, що знаходиться у поточному сегменті коду CS зі зсувом, що дорівнює вмісту регістра AX. У разі виконання цієї команди значення AX завантажується в регістр IP, після чого програма виконується, починаючи з цієї нової адреси.

3.3.2.8. Базова, індексна, базово-індексна і базово-індексна зі зсувом адресації

Під час обчислення ефективної адреси часто користуються поняттями: *база* та *індекс*.

У видах адресації, що розглядаються нижче, як базу та індекс можна використовувати:

- вміст одного з регістрів: BX, BP, SI або DI;
- 16-розрядний зсув (disp16).

Різниця між базовою та індексною адресацією дуже умовна. Вирази для обчислення ефективної адреси за формою однакові (рис. 3.14*а*, рис. 3.15*а*):

$$EA = \left.\begin{matrix} BP \\ BX \\ SI \\ DI \end{matrix}\right\} + зсув(disp).$$

Однак у разі базової адресації значення BP, BX, SI або DI є базою, а зсув: disp8 або disp16 – індексом.

Під час індексної адресації роль бази виконує disp16, а вміст BP, BX, SI або DI є індексом.

3.3.2.9. Базова адресація

Під час базової адресації виконавча адреса (ефективна адреса) може бути сумою вмісту регістрів BX∨BP∨SI∨DI і зсуву: disp8 або disp16 (*число зі знаком у додатковому коді*). Якщо у разі обчислення ефективної адреси використовуються регістри BX, SI або DI, то автоматично вибирається поточний сегмент даних DS, а якщо база міститься в регістрі BP, то обирається поточний сегмент стека SS.

Під час базової адресації як базу рекомендовано використовувати регістри BX або BP (рис. 3.14*а*).

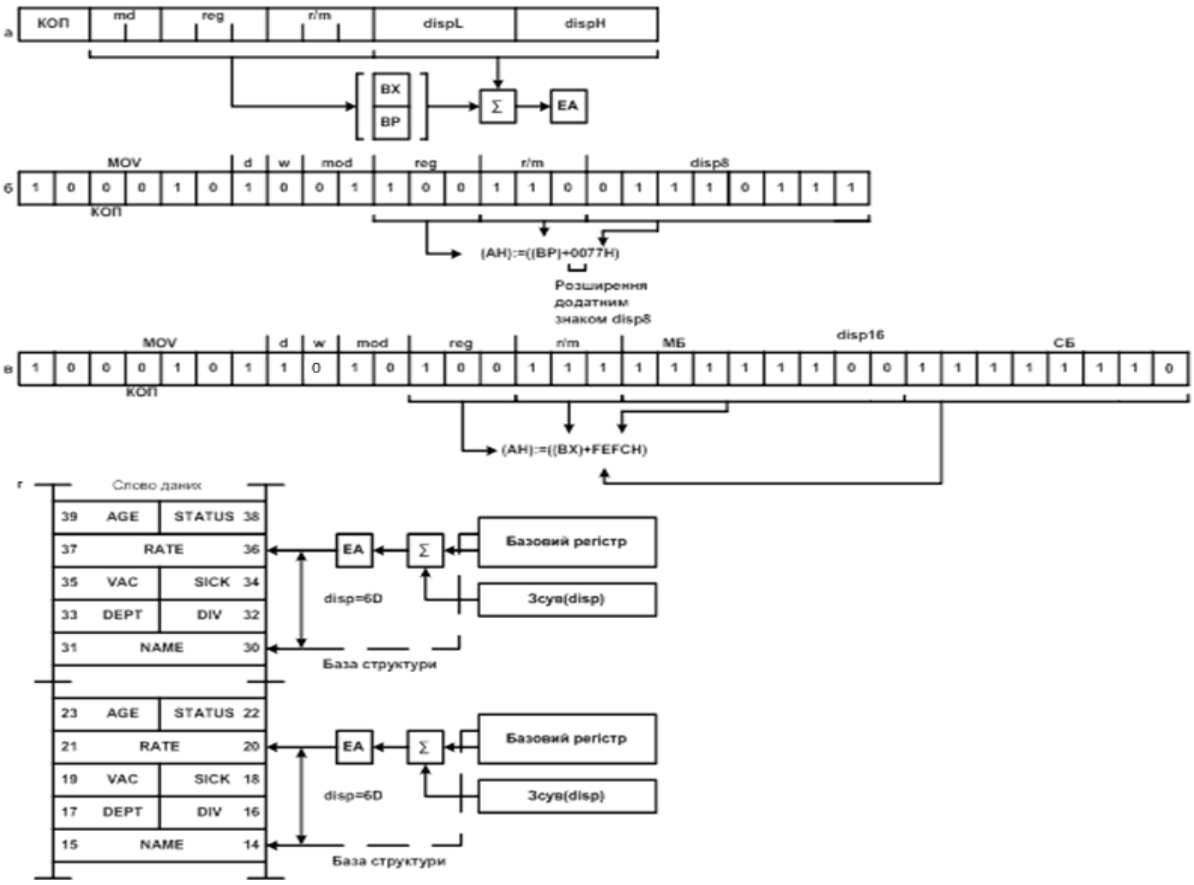


Рис. 3.14. Приклади команд з базовою адресацією

Є можливість перепризначити сегменти DS і SS командою-префіксом заміни сегмента (див. рис. 2.4).

Якщо використовується зсув disp8, то це число зі знаком у діапазоні (–128…+127), яке у разі обчислення ефективної адреси розширюється до шістнадцяти розрядів значенням знакового (старшого) біта disp8. Короткі зсуви застосовують для економії одного байта пам'яті програм, якщо індекс вкладається в діапазон: –128…+127 і немає сенсу використовувати disp16 (рис. 3.14*б*):

MOV AH, [BP + 77H]; AH←M(SS ·16 + BP + 0077H).

В іншому прикладі (рис. 3.14*в*) 16-розрядний зсув 0FEFCH безпосередньо використовується під час обчислення ефективної адреси: MOV AH, [BX + 0FEFCH]; AH ← M(DS ·16 + BX + 0FEFCH).

Головне застосування базової адресації пов'язане з обробкою однотипних структур даних, коли зсув (номер) елемента структури відомий у разі асемблювання програми. *Це дозволяє звертатися до тих самих елементів однотипних структур даних, розміщених у різних областях пам'яті.*

Модифікація вмісту базового регістра забезпечує доступ до цих областей. Таким чином, *під час базової адресації: індекс = const, а база = var* і обчислюється під час виконання програми (рис. 3.14*г*).

3.3.2.10. Індексна адресація

Під час індексної адресації ефективна адреса може обчислюватися як сума *додатного 16-розрядного зсуву (disp16)* та вмісту регістра SI∨DI∨BP∨BX.

Під час індексної адресації як індекс рекомендовано використовувати регістри SI або DI (рис. 3.15*а*, *б*).

У разі обчислення ефективної адреси додатний зсув disp16 = const виконує функцію бази, а вміст одного з названих вище регістрів виконує функцію індексу (зсуву відносно постійної бази), який змінюється. *Індексну адресацію зручно застосовувати для звернення до елементів масивів* (рис. 3.15*в*). Додатний зсув disp16 визначає відому у разі асемблювання початкову (базову) адресу масиву в сегменті, а значення регістра визначає положення потрібного елемента в масиві.

Прості маніпуляції з вмістом регістра, що виконує функцію індексу, дозволяють звертатися до будь-якого елемента масиву. Таким чином, *під час індексної адресації: база = const, а індекс = var* і обчислюється під час виконання програми.

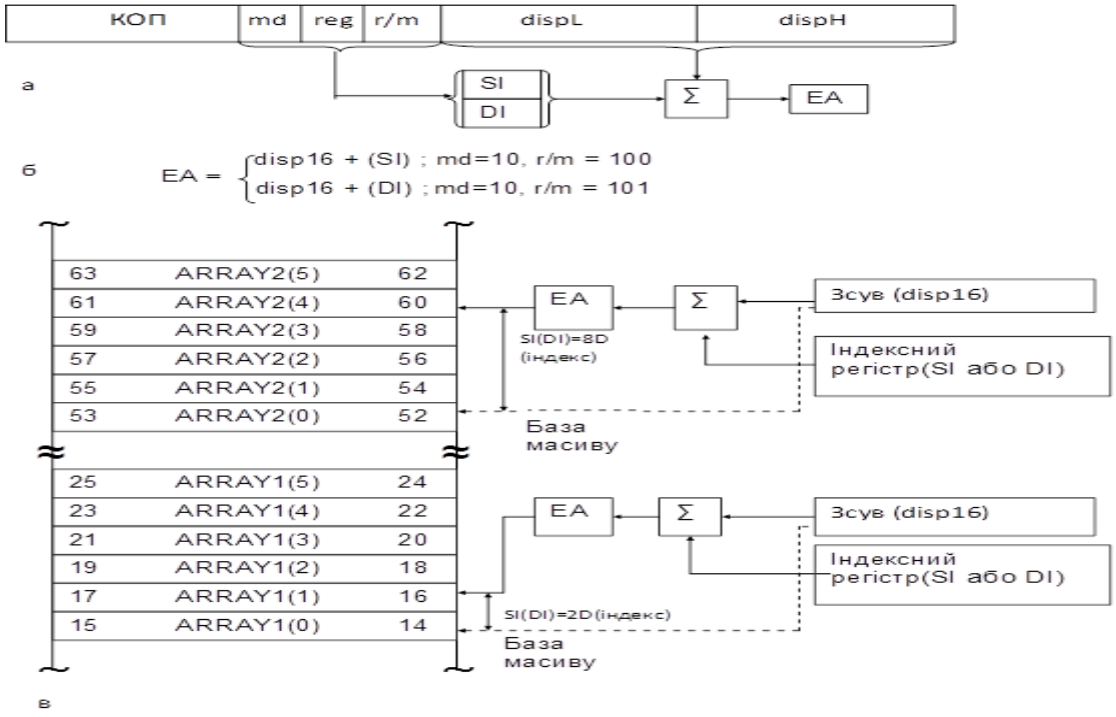


Рис. 3.15. Приклади команд з індексною адресацією

3.3.2.11. Базово-індексна адресація

Базово-індексна адресація генерує виконавчу адресу (англ. EA), яка є сумою значень базового та індексного регістрів (рис. 3.16).

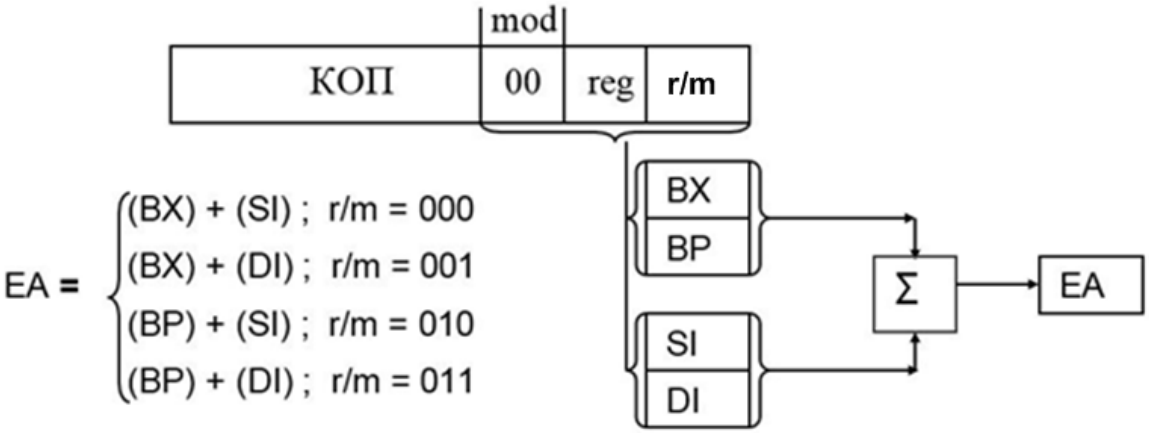


Рис. 3.16. Базово-індексна адресація

Така адресація є гнучким засобом доступу до найрізноманітніших ділянок пам’яті, оскільки дві компоненти адреси можна визначати та модифікувати під час виконання програми. У разі використання базового регістра BX виконавча адреса (EA) за замовчуванням є зсувом у сегменті даних (англ. DS), а у разі використання регістра BP – у сегменті стека (англ. SS). Прикладні програми можуть містити префіксні команди (див. рис. 2.4), які

дозволяють перепризначати ефективну адресу відносно інших сегментів пам’яті.

Під час базово-індексної адресації поле mod постбайта дорівнює 00, а r/m задає комбінації регістрів, що використовуються у разі обчислення ефективної адреси.

Таким чином, *під час базово-індексної адресації: база = var і індекс = var*, які можуть обчислюватися програмою і пересилатися у регістри, що використовуються у цьому разі як база та індекс.

Будь-який з чотирьох названих вище регістрів може бути базою або індексом, але рекомендовано використовувати:

– регістр BX∨BP як базу;

– регістри SI∨DI як індекс.

Як приклад, *базово-індексну адресацію можна використовувати для адресації елементів двовимірних масивів, що розміщені на початку сегмента даних або стека*.

3.3.2.12. Базово-індексна адресація зі зсувом (зміщенням)

У випадку базово-індексної адресації зі зсувом (зміщенням) у обчисленні ефективної адреси бере участь вміст регістрів BX, BP, SI, DI і зсув, що може бути 8- чи 16-розрядним числом зі знаком (рис. 3.17).

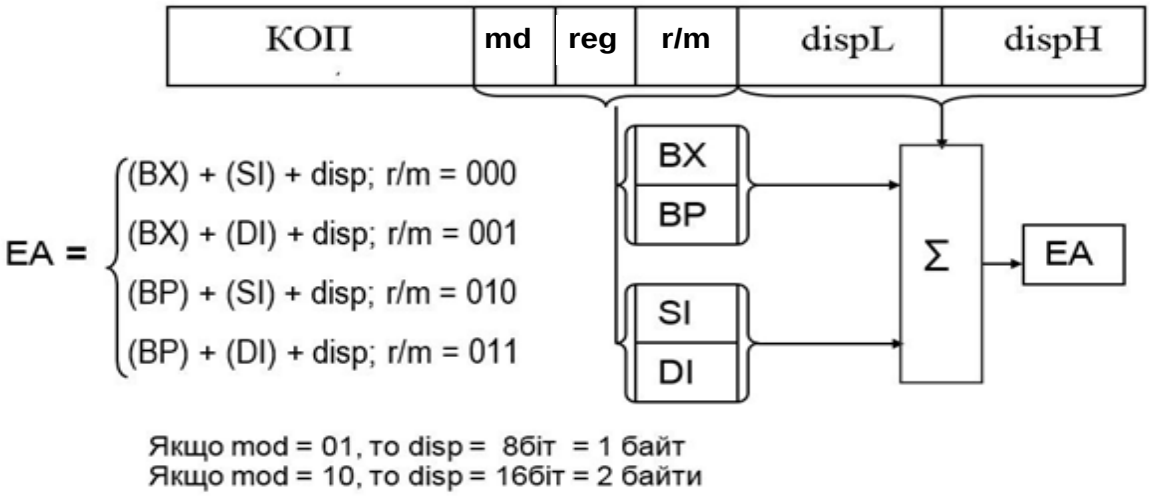


Рис. 3.17. Базово-індексна адресація зі зміщенням

Використання зсуву ще більше збільшує гнучкість програмування порівняно з базово-індексною адресацією (рис. 3.18).

У цьому прикладі регістр BP містить зсув початку структури в сегменті стека (англ. SS).

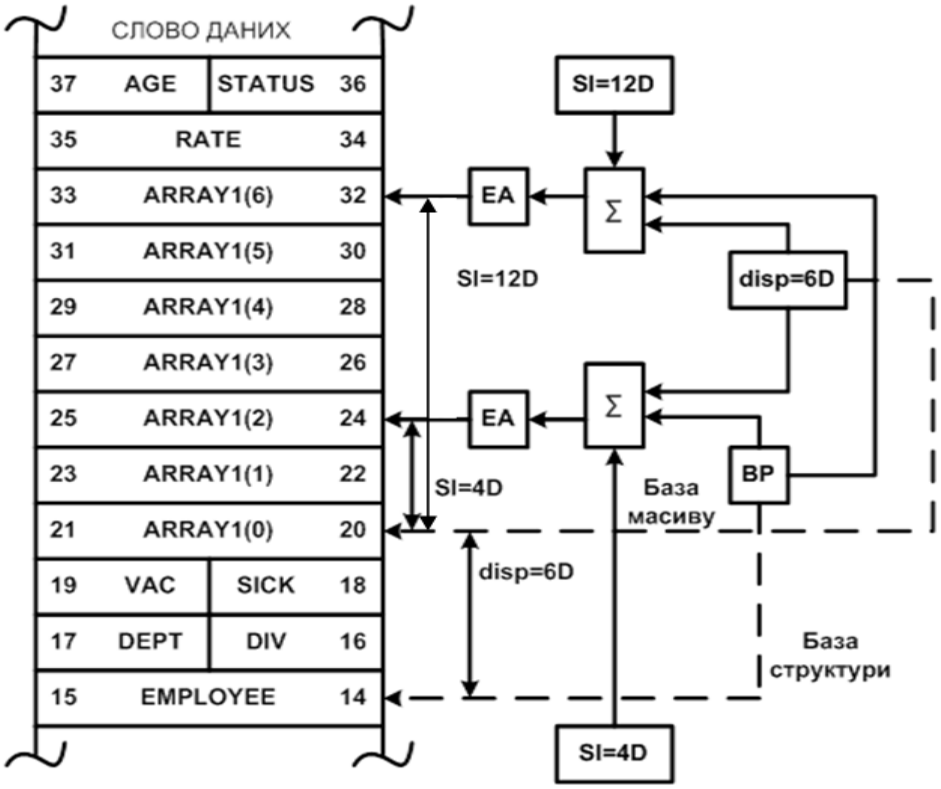


Рис. 3.18. Звернення до елемента масиву структури, розміщеної у сегменті стека за допомогою базово-індексної адресації зі зсувом

Відстань початку масиву від початку структури дорівнює зсуву (6D), який заданий у команді. Вміст індексного регістра SI обирає конкретний елемент масиву даних у структурі.

Таким чином, під час базово-індексної адресації зі зсувом можуть використовуватися:

– регістр BX∨BP = var – початок структури (база);

– disp = const – початок масиву у структурі (зсув);

– регістр SI∨DI = var – адреса елемента в масиві (індекс).

Базово-індексну адресацію зі зсувом можна використовувати для адресації елементів двовимірних масивів, що розміщені в середині сегмента, або для забезпечення доступу до елементів масивів однотипних структур даних, розміщених, наприклад, у сегменті стека (рис. 3.18), або у сегменті даних.

3.3.2.13. Стекова адресація

Стекову адресацію виділено в самостійний вид умовно, щоб зазначити специфіку роботи зі спеціальною областю пам’яті, яку називають стеком.

Під час запису до стека командою PUSH стек розширюється, а під час читання зі стека командою POP стек стискається.

У будь-якому разі операція виконується відносно крайньої (останньої) комірки пам’яті, яку називають «верхівкою стека».

Формально стек – це пам’ять з організацією типу FILO. У МП і8086 базова адреса сегмента пам’яті, що виділено під стек, зберігається у регістрі SS. Адреса верхівки стека знаходиться у регістрі-покажчику стека SP.

На рис. 3.19 показано машинний код і особливості виконання в такому МП команди PUSH AX:

$$M(SS \cdot 16+(SP_{ПОЧ} -1)) \leftarrow AH;$$
$$M(SS \cdot 16+(SP_{ПОЧ} -2)) \leftarrow AL;$$
$$SP \leftarrow SP_{ПОЧ} -2,$$

де $SP_{ПОЧ}$ – вміст регістра-покажчика стека SP до виконання команди.

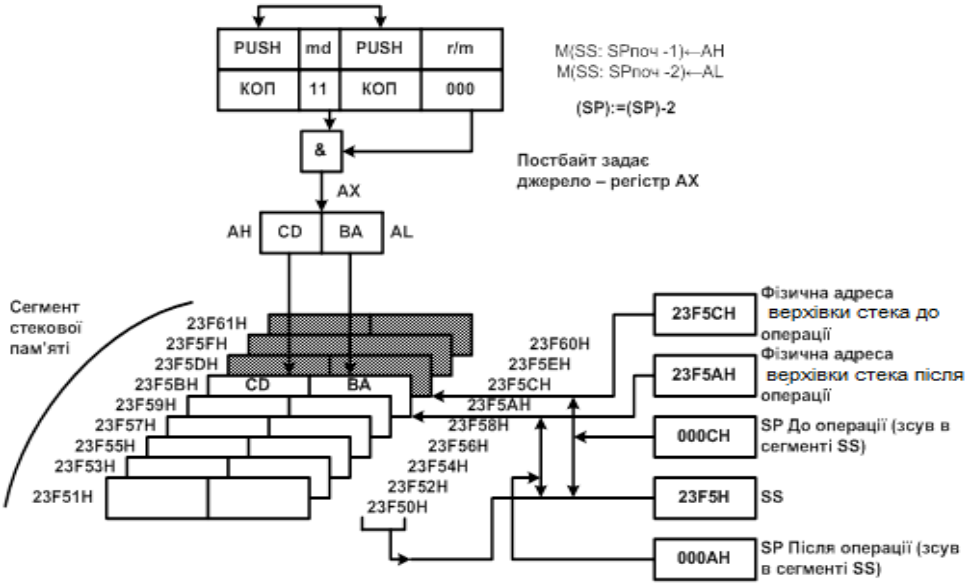


Рис. 3.19. Стекова адресація

У прикладі на рис. 3.19 $SP_{ПОЧ}$ = 000CH, а SS = 23F5H. Після виконання команди: SP = 000CH–2 = 000AH.

3.3.2.14. Відносна адресація

Відносна адресація застосовується в командах передачі керування, наприклад JMP, CALL, LOOP. У цьому разі машинний код команди містить зсув відносно вмісту вказівника команд, тобто відносно адреси наступної команди.

Як коротка відносна адреса в командах безумовного, умовного переходів та циклу використовується 8-розрядний зсув у діапазоні: –128…+127, який під час обчислення фізичної адреси розширюється до 16-ти розрядів знаковим бітом зсуву.

На рис. 3.20*а* наведено приклад команди безумовного переходу з короткою відносною адресою.

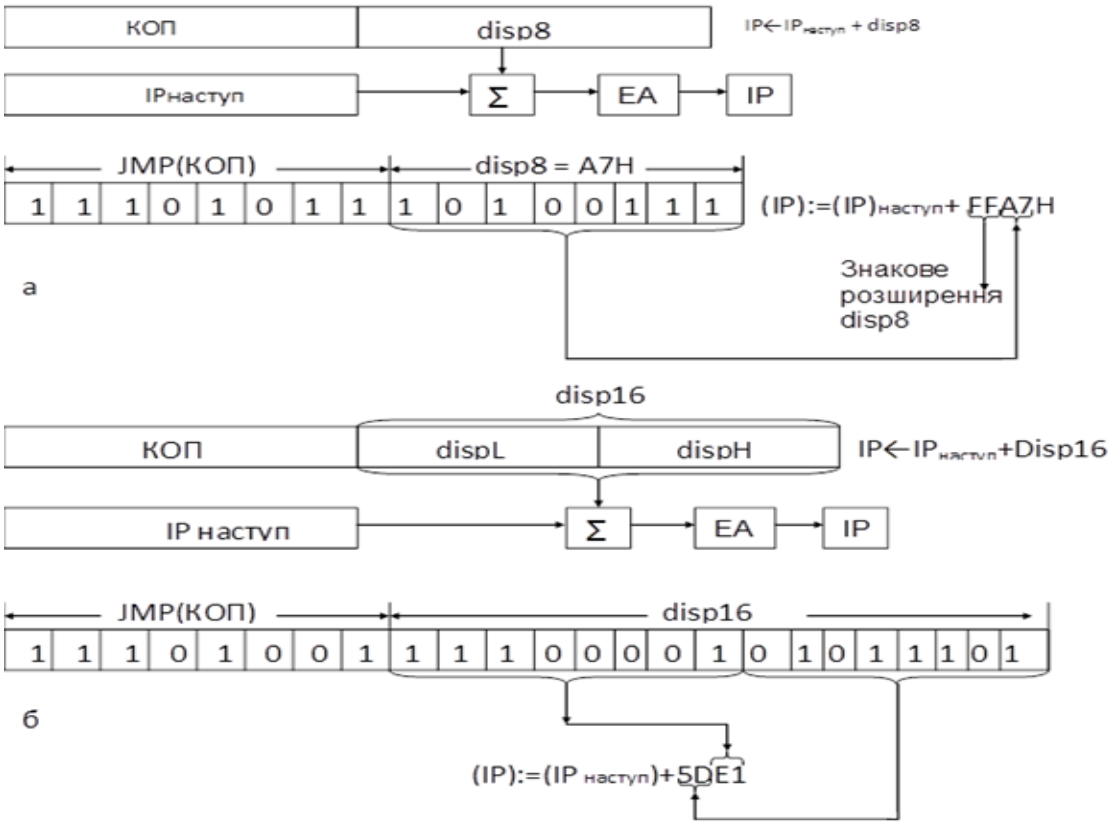


Рис. 3. 20. **Відносна адресація**

Другий байт команди (disp8 = A7H = 10100111B) розширюється до значення FFA7H і додається до вмісту вказівника команд $IP_{наступ}$, де $IP_{наступ}$ – адреса (зсув) наступної за JMP команди. Додавання FFA7H еквівалентно відніманню 59H = 01011001B = 89D з адреси, що міститься в IP (A7H є додатковим кодом числа 89D). Після виконання цієї команди МП перейде в програмі на 89 байт назад відносно адреси наступної команди ($IP_{наступ}$).

На рис. 3.20*а* в якості $IP_{наступ}$ є вміст лічильника команд IP після витягування команди JMP із пам’яті, тобто $IP_{наступ}$ адресує наступну команду.

На рис. 3.20*б* наведено приклад команди переходу з 16-розрядним зсувом, який визначає безумовний перехід вперед по програмі на 5DE1H = 24033 байти відносно адреси наступної команди.

У програмі, що написана мовою Асемблера, вказується не значення зсуву, а мітка тієї команди, якій передається керування. Необхідне значення зсуву disp автоматично обчислює системна програма-компілятор, що також, як і мова програмування, має назву «Асемблер». Зсув обчислюється за допомогою

віднімання від адреси мітки адреси наступної команди і підставляється у відповідні розряди машинного коду команди.

3.3.2.15. Адресація рядків даних

Цей вид адресації використовується у разі виконання операцій із рядками (ланцюжками) даних. Рядком (ланцюжком) називають послідовність байтів або слів, що розміщуються у сусідніх комірках пам'яті (наприклад, рядок, що вводиться з термінала). Під час обробки елементів рядків апаратно передбачається, що регістр SI адресує байт або слово рядка джерела (звідси походить його назва «індекс джерела»), а індексний регістр DI – байт або слово рядка приймача – «індекс приймача» (рис. 3.21).

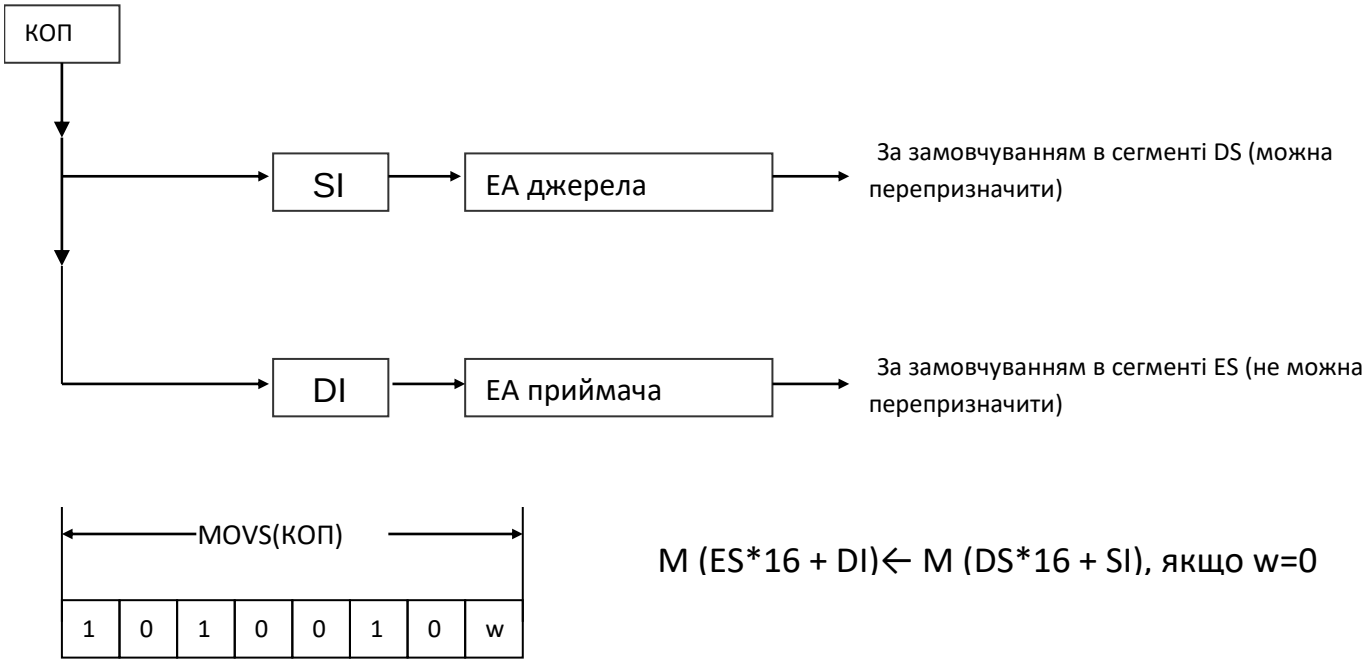


Рис. 3.21. Адресація рядків даних

Для формування адрес наступних елементів рядків МП автоматично корегує вміст регістрів SI і DI під час переходу до наступних елементів рядків (інкрементує, якщо прапорець DF = 0, або декрементує, якщо DF = 1).

Рядок-джерело може знаходитися в будь-якому сегменті (за замовчуванням приймається сегмент даних DS), а рядок-приймач тільки в додатковому сегменті даних, логічна базова адреса якого знаходиться в регістрі ES.

На рис. 3.21 показано машинний код однобайтної команди MOVS – пересилання елемента рядка-джерела (байт/слово) на місце елемента рядка-приймача.

Ця команда мовою Асемблера має вигляд:

MOVSB; (ES ·16+DI)←M(DS ·16+SI),

якщо довжина операнда дорівнює восьми бітам (1 байт) або MOVSW; (ES · 16+DI)←M(DS·16+SI), M(ES · 16+DI+1)←M(DS · 16+SI+1), якщо довжина операнда дорівнює 16-ти бітам (2 байти).

Код операції команди MOVSB: 10100100B = A4H (W = 0), а код операції команди MOVSW – 10100101B = A5H (W = 1).

Залежно від значення прапорця DF у разі виконання команди MOVS змінюється вміст регістрів SI і DI :

– у разі виконання команди MOVSB – на ±1 (DF = 0/1);
– у разі виконання команди MOVSW – на ±2 (DF = 0/1).

За допомогою однієї команди MOVS можна переслати тільки один елемент рядка. Для організації циклічного виконання команди MOVS (а також циклічного виконання деяких інших команд обробки рядків) використовується однобайтова команда-префікс REP (повторити). Вона записується перед головною командою (наприклад, REP MOVSB) і забезпечує її виконання *N* разів. Кількість повторень *N* попередньо записується у регістрі CX. Після пересилання кожного елемента вміст CX зменшується на одиницю. Коли після чергового декременту CX = 0, виконання команди закінчується.

3.3.2.16. Адресація портів введення/виведення

Мікропроцесор i8086, окрім можливості адресації 1 Мбайта комірок пам’яті, може додатково адресувати: 64 К пристроїв (портів) введення/виведення, які обмінюються 8-розрядними даними, або 32 К портів, які обмінюються 16-розрядними даними.

Для адресації зовнішніх пристроїв, розміщених в області адрес зовнішніх пристроїв, використовується пряма і непряма адресації. Під час прямої адресації адреса зовнішніх пристроїв 8-розрядна, що дозволяє адресувати: 256 8-розрядних або 128 16-розрядних портів введення/виведення (рис. 3.22*а*).

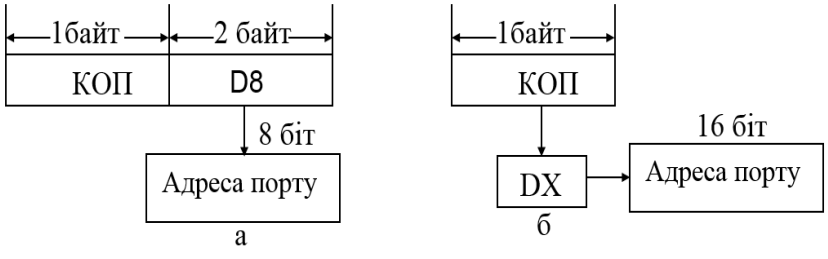


Рис. 3.22. Адресація портів введення/виведення

Наприклад, команда IN AL, 20H; AL←PORT(20H) вводить байт від зовнішнього пристрою в акумулятор AL через порт з адресою 20H. Ця команда займає 2 комірки пам’яті і має машинний код: 11100100B = E4H – перший байт; 00100000B = 20H – другий байт.

Непряма адресація портів (рис. 3.22*б*) аналогічна непрямій адресації операндів у пам’яті. Адреса зовнішнього пристрою розміщується у регістрі DX, що дозволяє адресувати 65536 8-розрядних або 32768 16-розрядних портів введення/виведення. Вміст регістра DX можна змінювати у процесі виконання програми і в такий спосіб звертатися до групи пристроїв введення/виведення у циклі.

Наприклад, команда IN AL, DX; AL ← PORT (DX) вводить байт від зовнішнього пристрою в акумулятор AL через порт, адреса якого міститься у регістрі DX. Ця команда займає одну комірку пам’яті та має машинний код 11101100B = ECH.

Окрім розглянутої, може також використовуватись адресація зовнішніх пристрів як комірок пам’яті, що виконується із застосуванням зазначених вище способів адресації запам’ятовувальних пристроїв.

3.3.2.17. Спосіб адресації операндів і час виконання команд

У разі розробки програмного забезпечення, особливо для МПС реального часу, важливим є час виконання тієї чи іншої команди. Тому для кожного формату команди МП вказується кількість тактів n, необхідних для її виконання. Той самий формат деяких команд може бути використано для задання різних способів адресації і, отже, різноманітних способів обчислення виконавчої (ефективної) адреси. Кількість тактів, необхідних для виконання такої команди: $n_T = n + T_{BA}$, де n – постійне значення, яке визначається з довідкових таблиць; T_{BA} – кількість тактів, необхідних для обчислення ефективної адреси (див. п. 3.2.10).

3.3.3. Опис 8-розрядного мікроконтролера сімейства AVR

3.3.3.1. Загальні відомості про мікроконтролер

AVR-мікроконтролери підтримують наведені нижче способи адресації операндів – даних, що беруть участь в операціях:

- неявна;
- безпосередня;
- пряма;
- непряма адресації.

Деякі способи адресації мають кілька різновидів залежно від того, до якої області пам’яті виконується звернення під час прямої адресації, або які додаткові дії виконуються над індексним регістром під час непрямої адресації.

3.3.3.2. Неявна адресація

Під час неявної адресації в команді відсутня адреса операнда в явному вигляді. Інформацію про потрібну адресу операнда МК отримує з коду операції команди, тобто вважають, що операнд адресується неявно. Так, наприклад, можуть адресуватися окремі прапорці регістра стану у командах типу BRTS k; BRVS k; BRPL k і т. ін.

3.3.3.3. Безпосередня адресація

Під час безпосередньої адресації операнд входить у саму команду та читається із пам’яті програм, що міститься у машинному коді команди. Це, наприклад, команди ADIW Rd, K6 (K6 = 6 біт); SBCI Rd*, K (K = 8 біт) і т. ін.

3.3.3.4. Пряма адресація

Під час прямої адресації адреси операндів містяться у машинному коді команди. Ця адресація використовується для звернення до комірок статичної пам’яті даних. Відповідно до її структури є такі різновиди прямої адресації: пряма адресація одного регістра загального призначення, пряма адресація двох регістрів загального призначення, пряма адресація регістра введення/виведення, пряма адресація всієї статичної пам’яті даних – регістра загального призначення, регістра введення/виведення та статичного ОЗП.

3.3.3.5. Пряма адресація одного регістра загального призначення

Цей спосіб адресації використовується в командах, що оперують з одним із регістрів загального призначення, при цьому адреса регістра-операнда (його номер) міститься в розрядах 8…4 (5 біт) машинного коду команди (рис. 3.23).

Прикладом команд, що використовують цей спосіб адресації, є команди роботи зі стеком – PUSH, POP, команди інкременту – INC, декременту – DEC, а також деякі команди арифметичних операцій.

3.3.3.6. Пряма адресація двох регістрів загального призначення

Цей спосіб адресації використовується в командах, що оперують одночасно двома регістрами загального призначення, при цьому адреса

регістра-джерела міститься в розрядах 9, 3…0 (5 біт), а адреса регістра-приймача в розрядах 8…4 (5 біт) машинного коду команди (рис. 3.24).

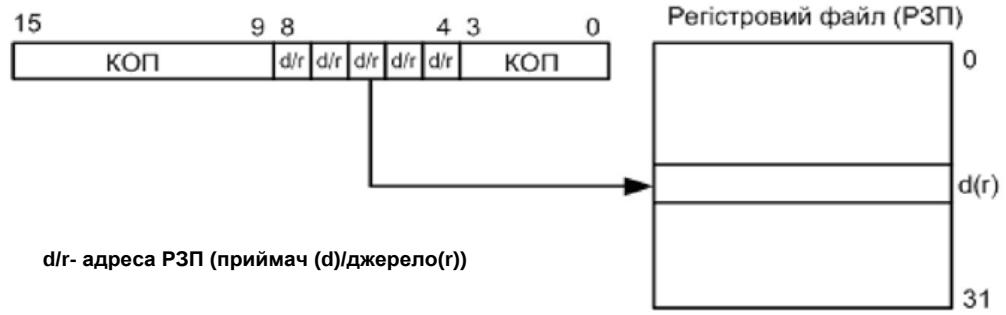


Рис. 3.23. Пряма адресація одного регістра загального призначення

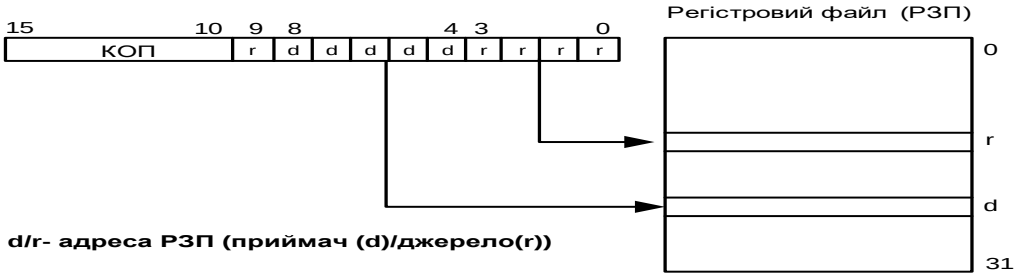


Рис. 3.24. Пряма адресація двох регістрів загального призначення

До команд, що використовують цей спосіб адресації, належать команди пересилання даних з одного регістра загального призначення у другий – MOV, а також більшість команд арифметичних операцій.

Деякі команди мають тільки один регістр-операнд, але використовують цей спосіб адресації. У цьому випадку той самий регістр є джерелом і приймачем. Як приклад можна навести команду очищення регістра – CLR Rd, яка виконує операцію «виключне АБО» регістра із самим собою – EOR Rd, Rd.

3.3.3.7. Пряма адресація регістра введення/виведення

Цей спосіб адресації використовується командами пересилання даних між регістром введення/виведення і регістром загального призначення (регістровим файлом) – IN і OUT.

У цьому разі адреса регістра введення/виведення міститься в розрядах 10, 9, 3…0 – 6 біт, а адреса регістра загального призначення – у розрядах 8…4 – 5 біт машинного коду команди (рис. 3.25).

Рис. 3.25. Пряма адресація регістрів введення/виведення

3.3.3.8. Пряма адресація статичної пам’яті даних

Цей спосіб використовується під час звернення до всього адресного простору статичної пам’яті даних, до якої належить: регістр загального призначення, регістр введення/виведення та статичний ОЗП.

Є тільки дві команди, що використовують цей спосіб адресації. Це команди пересилання байта між одним з регістрів загального призначення і коміркою статичної пам’яті даних – LDS і STS. Кожна з цих команд займає в пам’яті програм два слова (32 біти). У першому слові міститься код операції та адреса регістра загального призначення (у розрядах з 8-го по 4-й). У другому слові міститься адреса комірки пам’яті, до якої відбувається звернення (рис. 3.26).

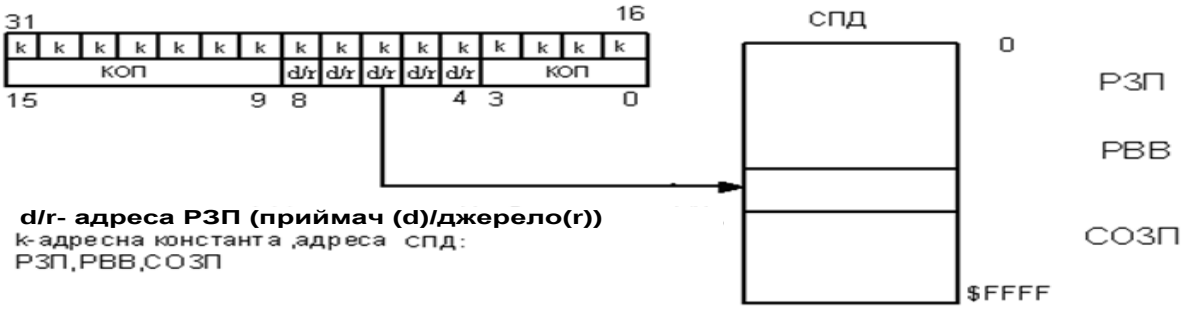


Рис. 3.26. Пряма адресація статичної пам’яті даних

За адресами $0000…$001F у СПД розміщено РЗП, за адресами $0020…$005F – основні РВВ, а за адресами $0060…$FFFF – ДРВВ та СОЗП.

3.3.3.9. Непряма адресація

Під час непрямої адресації адреса комірки статичної пам’яті даних міститься в одному з індексних регістрів X, Y і Z.

Є такі різновиди непрямої адресації:

– проста непряма адресація;
– відносна непряма адресація;

– непряма адресація з переддекрементом;
– непряма адресація з постінкрементом.

3.3.3.10. Проста непряма адресація

У разі використання простої непрямої адресації звернення виконується до статичної пам’яті даних за адресою, що міститься в одному з індексних регістрів: X, Y або Z (рис. 3.27).

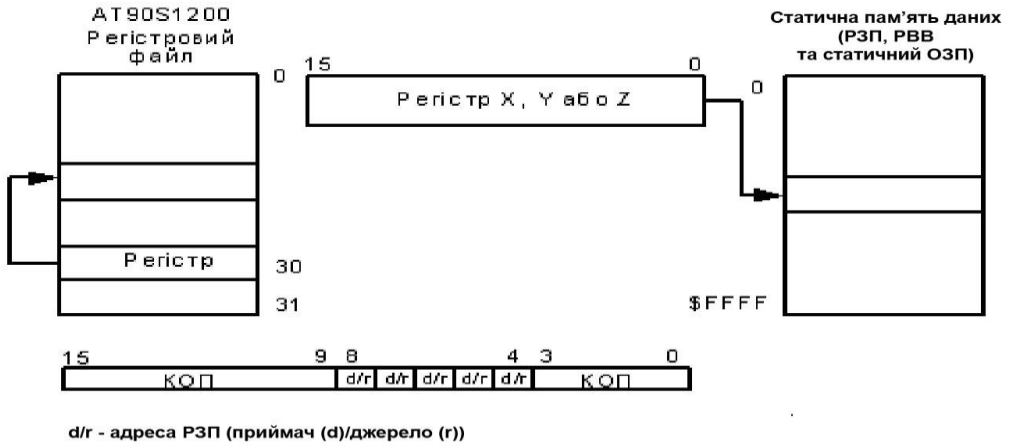


Рис. 3.27. Проста непряма адресація

Жодні дії із вмістом індексного регістра у цьому разі не виконується.

Мікроконтролери МК підтримують 6 команд (по 2 для кожного індексного регістра) простої непрямої адресації: LD Rd, X/Y/Z (пересилання байта з статичної пам’яті даних до регістра загального призначення) і ST X/Y/Z, Rr (пересилання байта з регістра загального призначення до статичної пам’яті даних). Адреса регістра загального призначення міститься в розрядах 8…4 машинного коду команди.

3.3.3.11. Відносна непряма адресація

У командах відносної непрямої адресації адреса комірки статичної пам’яті даних, до якої виконується звернення, обчислюється додаванням вмісту індексного регістра Y або Z і константи, що міститься в машинному коді команди (рис. 3.28).

МК підтримують чотири команди відносної непрямої адресації (дві для регістра Y і дві для регістра Z): LDD Rd, Y + q/Z + q (пересилання байта зі статичної пам’яті даних до регістра загального призначення) і STD Y + q/Z + q, Rr (пересилання байта з регістра загального призначення до статичної пам’яті даних).

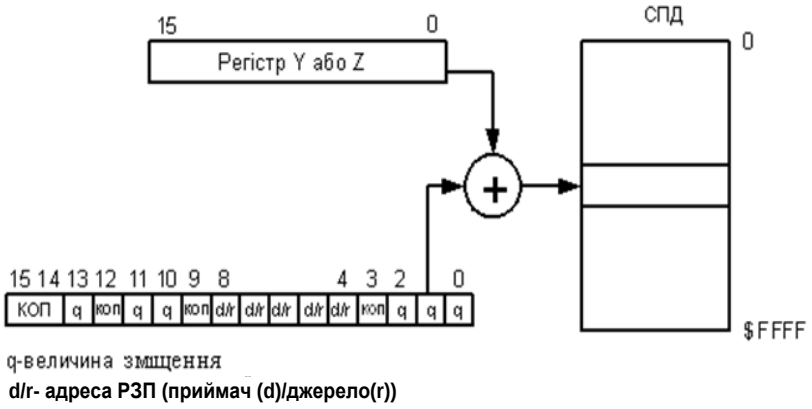


Рис. 3.28. Відносна непряма адресація

Адреса регістра загального призначення міститься в розрядах 8…4 машинного коду команди, а величина зміщення (q) – у розрядах 13, 11, 10, 2…0. Оскільки під значення зміщення виділяється тільки 6 біт, воно не може перевищувати 63 ($0 \le q \le 63$).

3.3.3.12. Непряма адресація з попереднім декрементом (переддекрементом)

Під час виконання команд непрямої адресації з переддекрементом вміст індексного регістра спочатку зменшується на одиницю, а потім виконується звернення за отриманою адресою (рис. 3.29).

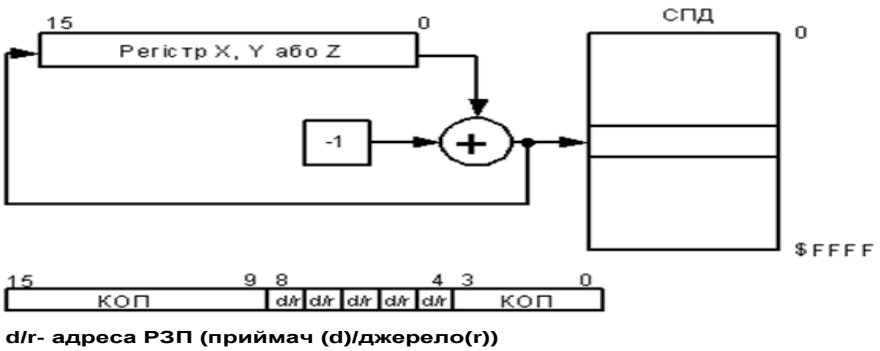


Рис. 3.29. Непряма адресація з переддекрементом

МК сімейства підтримують 6 команд (по 2 для кожного індексного регістра) непрямої адресації з переддекрементом: LD Rd, –X/–Y/–Z (пересилання байта зі статичної пам’яті даних до регістра загального призначення) і ST –X/–Y/–Z, Rr (пересилання байта з регістра загального призначення до статичної пам’яті даних).

Адреса регістра загального призначення міститься в розрядах 8…4 машинного коду команди, а регістри X/Y/Z адресуються неявно.

3.3.3.13. Непряма адресація з постінкрементом

Під час виконання команд непрямої адресації з постінкрементом (наступним інкрементом) після звернення за адресою, що міститься в індексному регістрі, вміст індексного регістра збільшується на одиницю (рис. 3.30).

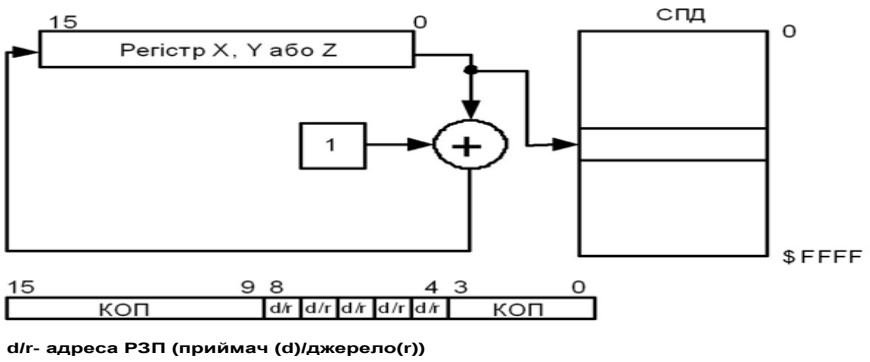


Рис. 3.30. Непряма адресація з постінкрементом

МК сімейства підтримують 6 команд (по 2 для кожного індексного регістра) непрямої адресації з постінкрементом: LD Rd, X+/Y+/Z+ (пересилання байта зі статичної пам'яті даних до регістра загального призначенн) і ST X+/Y+/Z+, Rr (пересилання байта з регістра загального призначення до статичної пам'яті даних). Адреса регістра загального призначенн міститься в розрядах 8…4 машинного коду команди, а регістри X/Y/Z адресуються неявно.

3.3.3.14. Непряма адресація пам'яті програм

Такий спосіб адресації використовується в командах IJMP, ICALL (рис. 3.31).

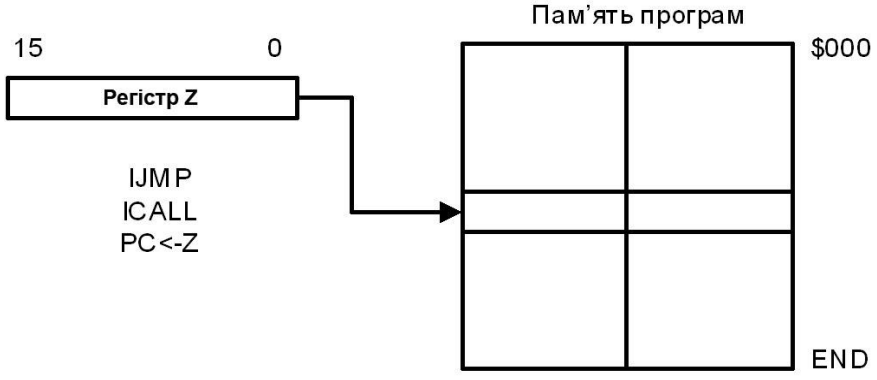


Рис. 3.31. Непряма адресація пам'яті програм

У результаті виконання такої команди програма продовжує виконуватися з адреси, що міститься в індексному регістрі Z. Таким чином, дія команди зводиться до завантаження вмісту індексного регістра у лічильник команд.

На відміну від команд відносного переходу ця команда не має обмежень за областю дії. Оскільки індексний регістр – 16-розрядний, то максимально можлива величина переходу становить 64 Кслів (128 Кбайт).

Як і команда відносного переходу, команда непрямого переходу виконується за два машинних цикли.

3.3.3.15. Непряма адресація констант у пам’яті програм

Цей спосіб адресації використовується в команді LPM, яка завантажує один байт із пам’яті програм у регістр загального призначення R0 (рис. 3.32).

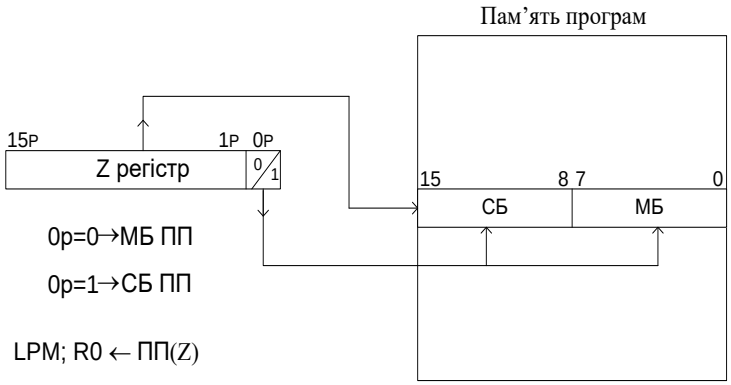


Рис. 3.32. Непряма адресація констант у пам’яті програм

Адреса комірки пам’яті, до якої відбувається звернення, міститься в індексному регістрі Z, при цьому старші п’ятнадцять розрядів регістра – Z1…Z15 адресують слово в пам’яті програм, а молодший біт – Z0 адресує байт в обраному слові. Якщо Z0 = 0, то адресується молодший байт слова, а якщо Z0 = 1 – старший байт.

Доступний об’єм пам’яті програм для цієї команди не може перевищувати 2^{15} = 32768 слів. Для звернення до розширеної пам’яті програм може використовуватися команда ELPM.

3.3.3.16. Відносна адресація пам’яті програм

Цей спосіб адресації використовується в командах RJMP, RCALL (рис. 3.33).

Під час виконання команди до поточного вмісту лічильника команд (адреси наступної команди) додається 12-розрядне число зі знаком – k, яке подано у додатковому двійковому коді.

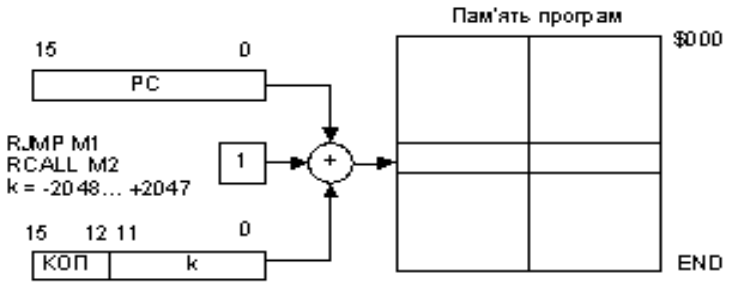


Рис. 3.33. Відносна адресація пам’яті програм

Ця команда має обмеження за областю дії. Через те, що операнд являє собою 12-розрядне число зі знаком, максимальна величина переходу відносно адреси наступної команди становить від –2048 до +2047 слів (приблизно ±4 Кбайт).

У програмах як операнди цієї команди замість адрес використовуються мітки.

Компілятор обчислює величину зміщення відносно адреси наступної команди – k. Для цього він віднімає від адреси мітки адресу наступної команди і підставляє це значення у відповідні розряди машинного коду команди.

3.4. Команди мікропроцесорів та мікроконтролерів

3.4.1. 16-розрядний мікропроцесор

3.4.1.1. Загальна характеристика команд

Усі команди 16-розрядного МП, наприклад, i8086, залежно від їх функціонального призначення, умовно поділено на такі групи: пересилання; робота зі стеком; введення/виведення; обмін; трансляція; завантаження; пересилання прапорців; арифметичні; корекція; порівняння; логічні; зсув; обробка рядків; умовні і безумовні переходи; організація циклів; виклик і повернення з підпрограм; програмні переривання; керування МП [2].

Усі названі групи команд об’єднано у п’ять таблиць, до яких включено основні (базові) команди (мнемоніки). По кожній команді в таблиці міститься основна та довідкова інформація: номер команди в межах таблиці; мнемоніка; функція/алгоритм, який реалізує команда (коментар); формат команди; кількість байт та час виконання в тактах.

У таблицях відсутня ще одна важлива характеристика команди – вплив на окремі прапорці (ознаки), на якій зупинимося при більш докладному розгляді кожної групи команд.

3.4.1.2. Команди пересилання даних

Більшість команд цієї групи (табл. 3.5) мають мнемоніку MOV і здійснюють *пересилання* байта або слова від джерела операнда до місця призначення приймача.

Як джерело та приймач можуть служити один з регістрів загального призначення, сегментні регістри або комірки пам'яті. Окрім того, джерелом можуть бути безпосередні операнди.

Залежно від способу адресації операндів команди пересилання мають довжину від двох до шести байтів і на прапорці не впливають.

До групи команд пересилання належать також команди роботи зі стеком, які мають мнемоніки PUSH/POP і служать для занесення 16-розрядного слова, що міститься в регістрі або пам'яті, у стек, або читання цього слова зі стека. В якості 16-розрядного операнда під час роботи зі стеком можуть бути вміст одного з регістрів загального призначення, сегментних регістрів, регістра прапорців і двох сусідніх комірок пам'яті.

Довжина команд роботи зі стеком від одного до чотирьох байт. На прапорці вони не впливають, окрім команди POPF, яка завантажує зі стека регістр прапорців.

До групи команд пересилання належать команди обміну XCHG, які викликають *обмін* байтами/словами між двома джерелами, в якості яких можуть бути регістри та пам'ять. Сегментні регістри та регістр прапорців у цьому обміні використовуватися не можуть. Довжина команд від одного до чотирьох байт. На прапорці вони не впливають.

Окрім команд роботи з пам'яттю, МП має окрему групу команд *введення/виведення* (табл. 3.5), які виконують обмін між МП та пристроями введення/виведення (портами).

Для введення байта/слова з порту служить команда IN, яка розміщує дані в акумуляторі (в AL або AX, залежно від довжини операнда). Номер порту може бути задано як безпосередньо у другому байті команди, так і непрямо у регістрі DX.

Перший формат команд введення/виведення дозволяє адресувати $2^8 = 256$ 8-розрядних портів або 128 16-розрядних.

Непряме задання порту, хоча і вимагає попереднього завантаження його номера в DX, дозволяє організувати програмні цикли, в яких використовується номер портів введення/виведення, що змінюється. Кількість портів, що адресуються в цьому випадку: 65536 8-розрядних або 32768 16-розрядних.

Таблиця 3.5. **Команди пересилання**

№ з/п	Мне-моніка	Функція (алгоритм)	Формат				Кількість байт	Кількість тактів
			1 байт	2 байти	3 байти	4 байти		
1	MOV	r/m ← reg	1000100w	mdreg r/m			2…4	9 + $T_{ва}$
2	MOV	reg ← r/m	1000101w	mdreg r/m			2…4	2/8 + $T_{ва}$
3	MOV	r/m ← sr	10001100	md0sr r/m			2…4	2/9 + $T_{ва}$
4	MOV	sr ← r/m	10001110	md0sr r/m	В якості sr не можна використовувати РГ CS		2…4	2/8 + $T_{ва}$
5	MOV	A ← M(addr)	1010000w	addr L	addr H		3	10
6	MOV	M(addr) ← A	1010001w	addr L	addr H		3	10
7	MOV	r/m ← data	1100011w	md000 r/m	data L	data H, w=1	3…6	10 + $T_{ва}$
8	MOV	reg ← data	1011wreg	data L	data H, w=1		2, 3	4
9	PUSH	стек ← r/m	11111111	md110 r/m			2…4	10/16 + $T_{ва}$
10	PUSH	стек ← reg	01010reg				1	10
11	PUSH	стек ← sr	000sr110				1	10
12	PUSHF	стек ← F	10011100				1	10
13	POP	r/m ← стек	10001111	md000 r/m			2…4	8/17 + $T_{ва}$
14	POP	reg ← стек	01011reg				1	8
15	POP	sr ← стек	000sr111				1	8
16	POPF	F ← стек	10011101				1	8
17	XCHG	r/m ↔ reg	1000011w	mdreg r/m			2…4	4/17 + $T_{ва}$
18	XCHG	AX ↔ reg	10010reg				1	3
19	XLAT	AL←M(BX+AL)	11010111				1	11
20	LEA	reg ← EA	10001101	mdreg r/m			2…4	2 + $T_{ва}$
21	LDS	DS, reg ←r/m	11000101	mdreg r/m			2…4	16 + $T_{ва}$
22	LES	ES, reg ←r/m	11000100	mdreg r/m			2…4	16 + $T_{ва}$
23	LAHF	AH ← F(7…0)	10011111				1	4
24	SAHF	F(7…0) ← AH	10011110				1	4
25	IN	A ←port(addr)	1110010w	addr			2	10
26	IN	A ←port(DX)	1110110w				1	8
27	OUT	port(addr) ←A	1110011w	addr			2	10
28	OUT	port(DX) ←A	1110111w				1	8

Примітка:

data - байт або слово даних;

addr - 16-розрядна адреса запам'ятовуючого пристрою (ЗП) або 8- розрядна адреса зовнішнього пристрою (ЗВПР);

addr L – молодший байт адреси;

addr H - старший байт адреси;

M(addr) - мітка ЗП з адресою addr;

port(addr) - порт ЗВПР з адресою addr.

SR	REG
00	ES
01	CS
10	SS
11	DS

reg	w = 0	w = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

w = 0 : data = 8 біт

w = 1 : data = 16 біт

Команда виведення OUT служить для пересилання даних (байта або слова) з акумулятора (з AL або AX) у порт виведення. Ця команда, як і команда IN, має два формати, які визначають спосіб адресації порту виведення. Довжина команд IN, OUT один/два байти. На прапорці вони не впливають.

До групи команд пересилання належить також однобайтна команда *трансляції (перетворення кодів)*: XLAT; AL←M(DS·16 + BX + AL), яку орієнтовано на табличні перетворення кодів. Таблиця розміром не більше за 256 байт розміщується в пам'яті (за замовчуванням в сегменті DS). Початкова адреса таблиці завантажується в регістр BX. Перед виконанням команди в регістрі AL у двійковому коді міститься номер рядка таблиці, з якого байт пересилається в AL. Розглянемо, наприклад, перетворення десяткових цифр (номерів рядків таблиці) у код «2 з 5» (табл. 3.6).

Таблиця 3.6. **Перетворення кодів**

Цифри	**Код 2 з 5**
0	11000
1	00011
2	00101
3	00110
4	01001
5	01010
6	01100
7	10001
8	10010
9	10100

Для перекодування двійкового представлення деякої цифри K (K = 0, 1, 2, …, 9) у код «2 з 5» треба виконати такі операції (рис. 3.34):

- завантажити початкову адресу таблиці коду «2 з 5» до регістра BX;
- завантажити двійковий код цифри K (наприклад K = 8D = 00001000B) до AL;
- виконати команду XLAT, яка пересилає вміст K-го рядка таблиці до AL (для K = 8 до AL пересилається код xxx10010B, де xxx – початкове значення 3-х старших розрядів елемента пам'яті, в якому зберігається код). У прикладі, що розглядається xxx = 000.

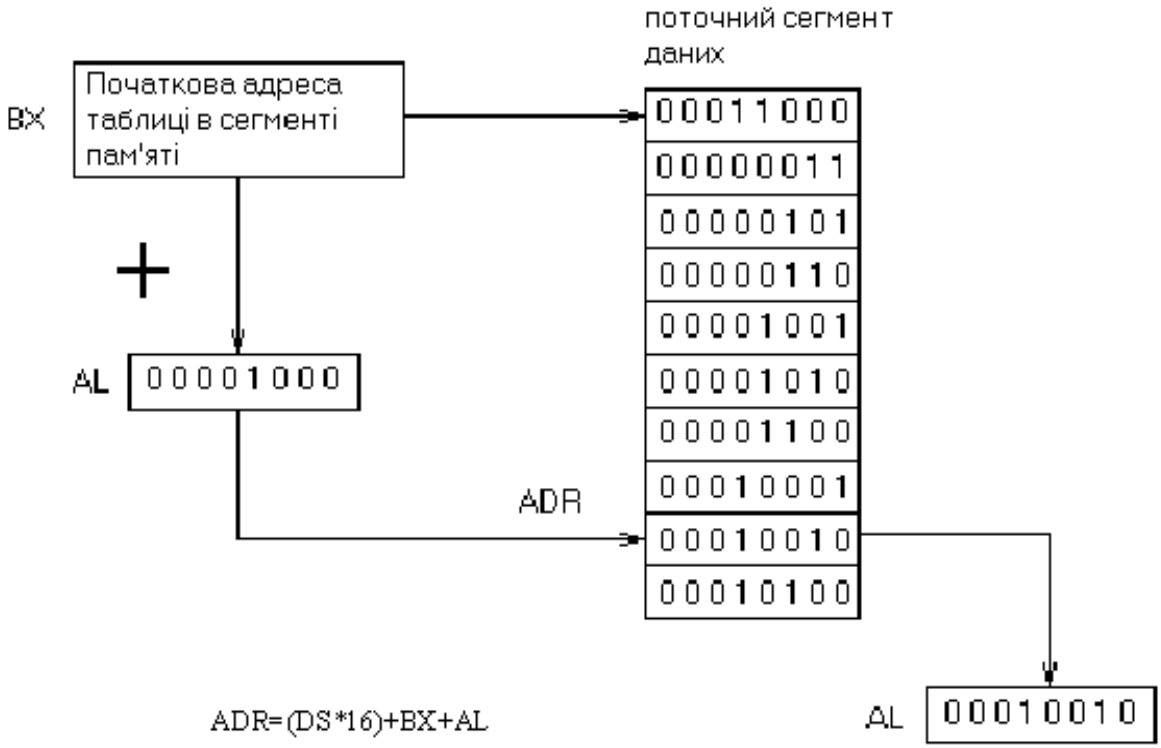


Рис. 3.34. Приклад використання команди XLAT

До групи команд пересилання належать також команди *завантаження*. Ці команди використовуються для керування адресацією операндів та включають три мнемоніки:

- LEA (завантажити виконавчу (ефективну) адресу операнда EA);
- LDS (завантажити покажчик до регістрів DS та reg);
- LES (завантажити покажчик до регістрів ES та reg),

За командою LEA у 16-розрядний регістр загального призначення, що адресується полем reg другого байта команди (постбайта), завантажується ефективна адреса операнда, яка обчислюється згідно зі значеннями полів md і r/m постбайта. Використання команди LEA зручне під час створення підпрограм, що працюють з параметрами. У цьому випадку перед викликом підпрограми у відповідний регістр завантажується адреса змінної, яку заздалегідь записано в пам’ять як параметр [2].

Потрібно зауважити, що ця команда дозволяє обчислювати ефективну адресу за значенням двох регістрів загального призначення та зміщенням. Її зручно використовувати перед виконанням команди XLAT (див. вище) для обчислення базової адреси таблиці у поточному сегменті даних і завантаження цієї адреси до регістра BX.

Команди LDS і LES використовуються переважно у разі звернення до даних, що знаходяться поза поточними сегментами DS або ES. У цьому разі

виникає необхідність змінити базову 16-розрядну адресу сегмента (базу) і 16-розрядне зміщення у сегменті, що звуться покажчиком. Для цього покажчик заздалегідь завантажується у чотири суміжні комірки пам’яті у такому порядку:

– молодший байт зміщення;
– старший байт зміщення;
– молодший байт бази (адреси сегмента);
– старший байт бази (адреси сегмента).

Під час виконання команди LDS/LES адреса (зміщення) першого байта покажчика у сегменті даних обчислюється за значеннями полів md і r/m постбайта. Після цього два перших байти покажчика (зміщення) завантажуються у регістр, що адресується полем reg постбайта, а два старших байти (база) – у регістр DS/ES.

Команди завантаження прапорців не змінюють.

До групи команд пересилання належать також команди *пересилання регістра прапорців* (регістр RF):

– LAHF (завантажити регістр AH вмістом регістра прапорців);
– SAHF (переслати вміст регістра AH до регістра прапорців);
– PUSHF (записати вміст регістра прапорців до стека);
– POPF (переслати вміст стека до регістра прапорців).

За командою LAHF здійснюється пересилання молодшого байта регістра прапорців RF у регістр AH, а за командою SAHF – зворотне пересилання. Ці команди введено в систему команд i8086 для спрощення програмної сумісності з 8-розрядним процесором i8080. Зокрема, без їх використання для реалізації команд PUSH PSW і POP PSW МП i8080 за допомогою команд МП i8086 потрібно по 9 байт пам’яті, а використання, наприклад, команди LAHF дозволяє під час реалізації команд PUSH PSW обійтися усього двома байтами: LAHF, PUSH AX.

Команда PUSHF вміщує вміст регістра (англ. RF) до стека, причому спочатку записується старший байт RF, а потім – молодший.

Очевидно, що дві з цих чотирьох команд (SAHF і POPF) впливають на прапорці.

3.4.1.3. Арифметичні команди

МП i8086 виконує чотири основні арифметичні операції: додавання, віднімання, множення і ділення над 8-розрядними та 16-розрядними даними зі знаком і без знака (табл. 3.7).

Таблиця 3.7. **Арифметичні команди**

№ з/п	Мне-моніка	Функція (алгоритм)	Формат				Кількість байт	Кількість тактів
			1 байт	2 байти	3 байти	4 байти		
1	ADD	r/m←r/m + reg	0000000w	mdreg r/m			2…4	16 + $T_{ва}$
2	ADD	reg←reg + r/m	0000001w	mdreg r/m			2…4	3/9 + $T_{ва}$
3	ADD	r/m←r/m+data	100000sw	md000 r/m	data L	data H,sw=01	3…6	4/17 + $T_{ва}$
4	ADD	A ←A + data	0000010w	data L	data H ,w=1		2…3	4
5	ADC	r/m← r/m+reg+C_F	0001000w	mdreg r/m			2…4	16 + $T_{ва}$
6	ADC	reg← reg+r/m +C_F	0001001w	mdreg r/m			2…4	3/ 9 + $T_{ва}$
7	ADC	r/m ←r/m+data+C_F	100000sw	md010 r/m	data L	data H,sw=01	3…6	4/17 + $T_{ва}$
8	ADC	A←A+data+C_F	0001010w	data L	data H, w=1		2, 3	4
9	INC	r/m ← r/m + 1	1111111w	md000 r/m			2…4	3/15 + $T_{ва}$
10	INC	reg ← reg + 1	01000reg	reg – тільки 16-розрядний			1	2
11	AAA	ASCII-корекція для додавання	00110111	Якщо ((AL)&0FH)>9 або (AC_F=1), тоді (AL)←(AL)+6; (AH)←(AH)+1; F=1; (AC_F)←1; (AL)←(AL)&0FH			1	4
12	DAA	2-10-корекція для додавання	00100111	Якщо ((AL)&0FH)>9 або (AC_F =1), тоді (AL)←(AL)+6, AC_F←1; якщо (AL)>9FH або (C_F)=1, то (AL)←(AL)+60H, C_F←1			1	4
13	SUB	r/m ← r/m – reg	0010100w	mdreg r/m			2…4	16 + $T_{ва}$
14	SUB	reg ← reg – r/m	0010101w	mdreg r/m			2…4	3 /9 + $T_{ва}$
15	SUB	r/m ← r/m – data	100000sw	md101 r/m	data L	data H,sw=01	3…6	4/17 + $T_{ва}$
16	SUB	A ← A – data	0010110w	data L	data H, w=1		2, 3	4
17	SBB	r/m←r/m – reg – C_F	0001100w	mdreg r/m			2…4	16 + $T_{ва}$

Продовження табл. 3.7

18	SBB	reg ← reg – r/m – C_F	0001101w	mdreg r/m			2…4	3/9 + $T_{ва}$
19	SBB	r/m←r/m-data-C_F	100000sw	md011 r/m	data L	data H,sw=01	3…6	4/17 + $T_{ва}$
20	SBB	A ← A – data – C_F	0001110w	data L	data H, w=1		2…3	4
21	DEC	r/m ← r/m – 1	1111111w	md001 r/m			2…4	3/15 + $T_{ва}$
22	DEC	reg ← reg – 1	01001reg	reg – тільки 16-розрядний			1	2
23	NEG	r/m ← 0 – r/m	1111011w	md011 r/g	Змінює знак (формує додатковий код)		2…4	3/16 + $T_{ва}$
24	CMP	r/m – reg	0011100w	mdreg r/m	Якщо r/m=reg, тоді Z_F=1 Якщо r/m<reg, тоді C_F=1 Якщо r/m>reg, тоді $Z_F = C_F = 0$		2…4	9 + $T_{ва}$
25	CMP	reg – r/m	0011101w	mdreg r/m	Якщо reg=r/m, тоді Z_F=1 Якщо reg<r/m, тоді C_F=1 Якщо reg>r/m, тоді $Z_F = C_F = 0$		2…4	3/9 + $T_{ва}$
26	CMP	r/m – data	100000sw	md111 r/m	data L data H, sw = 01 Якщо r/m = data, то Z_F=1 Якщо r/m <data, то C_F=1 Якщо r/m>data, то $Z_F = C_F = 0$		3…6	Регістри 4; Пам’ять w = 0 10 + $T_{ва}$; w = 1 17 + $T_{ва}$;

27	CMP	A – data	0011110w	data L	data H, w = 1 Якщо A=data, тоді $Z_F = 1$ Якщо A<data, тоді $C_F = 1$ Якщо A>data, тоді $Z_F = C_F = 0$	2, 3	4
28	AAS	ASCII-корекція для віднімання	00111111	Якщо ((AL)&0FH)>9 або (AF=1), тоді (AL)←(AL)–6, (AH)←(AH)–1, C_F←1; AF←1, (AL) ← (AL)&0FH		1	4
29	AAM	ASCII-корекція для множення	11010100	00001010	(AL)←залишок від (AL):10 (AH)←частка від (AL):10	2	83
30	DAS	2-10-корекція для віднімання	00101111	Якщо ((AL)&0FH)>9 або (AC_F=1), тоді (AL)←(AL)–6, AC_F←1; Якщо (AL)>9FH або (C_F) = 1, тоді (AL)←(AL)–60H, C_F ←1		1	4
31	MUL (множення цілих чисел без знака)	AX←AL·r/m, w=0 DX, AX←AX·r/m, w = 1	1111011w	md100 r/m	Якщо AH<>0, тоді $C_F = O_F = 1$, Якщо AH = 0, тоді $C_F = O_F = 0$, Якщо DX<>0, тоді $C_F = O_F = 1$, Якщо DX = 0, тоді $C_F = O_F = 0$	від 2 до 4	reg8·reg8 70–77; reg16·reg16 118–133; Mem8· reg8 76...83 + $T_{ва}$; Mem16·reg16 124...139 + $T_{ва}$;
32	IMUL (множення цілих чисел зі знаком)	AX←AL·r/m, w=0 DX,AX←AX·r/m, w = 1	1111011w	md101 r/m	Якщо AH<>0, тоді $C_F = O_F = 1$, Якщо AH = 0, тоді $C_F = O_F = 0$, Якщо DX<>0, тоді $C_F = O_F = 1$, Якщо DX = 0, тоді $C_F = O_F = 0$	від 2 до 4	reg8·reg8 80–98; reg16·reg16 128–154; Mem8· reg8 86...104 + $T_{ва}$; Mem16·reg16 138...164 + $T_{ва}$;

Закінчення табл. 3.7

33	DIV (ділення цілих чисел без знака)	AX: r/m , w = 0 AH – залишок AL – частка DX,AX : r/m, w =1 DX – залишок AX – частка	1111011w	md110 r/m	Якщо частка перевищує розміри AL або AX, чи якщо дільник дорівнює 0, то генерується переривання типу 0	від 2 до 4	reg16:reg8 80...90; reg32:reg16 144...162; reg16:mem8 86...96+$T_{ва}$; reg32:mem16 150...168+$T_{ва}$
34	IDIV (ділення цілихчисел зі знаком)	AX: r/m , w = 0 AH – залишок AL – частка DX,AX : r/m, w =1 DX – залишок AX – частка	1111011w	md111 r/m	Якщо частка перевищує розміри AL або AX, чи якщо дільник дорівнює 0, то генерується переривання типу 0	від 2 до 4	reg16:reg8 101...112; reg32:reg16 165...184 reg16:mem8 107...118+$T_{ва}$; reg32:mem16 175...194+$T_{ва}$;
35	AAD	ASCII-корекція для ділення	11010101	00001010	(AL) ← (AH)x10+(AL) (AH) ← 0	2	60
		Перетворює два незапакованих BCD-числа, що містяться в AX, у двійкове значення					
36	CBW	AH ← sign AL	10011000	Знак AL переходить в усі біти AH		1	2
37	CWD	DX ← sign AX	10011001	Знак AX переходить в усі біти DX		1	5

Примітка. w = 0 : data = 8 біт; w = 1 : data = 16 біт; s = 0 : data = 16 біт; s = 1 : sign data = 8 біт.

Числа зі знаком подано в МП у додатковому коді і їх діапазон у разі, наприклад, шістнадцяти розрядів становить від –32768 до +32767. Для реалізації арифметичних дій над числами, які подано у двійково-десятковому коді, введено спеціальні команди корекції результатів. Усі арифметичні команди впливають на прапорці.

Операції *додавання* включають три мнемоніки: ADD (додати), ADC (додати з урахуванням перенесення) і INC (збільшити на одиницю).

Команда ADD (№ 1…4) здійснює додавання операнда джерела до операнда місця призначення. Джерелом і місцем призначення звичайно є регістр або пам'ять, причому джерелом можуть бути також дані (константи), які безпосередньо представлено в команді. У першому форматі (№ 1, 2) біт машинного коду команди *d* визначає місце призначення результату додавання: якщо *d* = 1 це регістр, а якщо *d* = 0 – пам'ять. У другому форматі (№ 3) поле машинного коду команди *s* визначає кількість байтів даних, які безпосередньо подано в команді. Якщо *s* = 0 команда складається з трьох (w = 0) або чотирьох (w = 1) байт, причому у двох останніх записано 16-розрядну константу.

Якщо *s* = 1 та w = 1, то команда складається з трьох байт і останній, третій, містить 8-розрядну константу, яка перед додаванням розширюється знаковим розрядом до шістнадцяти розрядів, якщо довжина чисел, що додаються дорівнює шістнадцяти бітам (w = 1). Нагадаємо, що розширення знаком виконується копіюванням значення знакового розряду молодшого байта на весь старший байт, наприклад константу 5AH буде розширено до 005AH, а константу 8AH – до FF8AH.

Встановлення прапорців SF, ZF, PF, AF, і CF відповідно до результату операції додавання здійснюється так само, як і в МП i8080, але додатково введено прапорець переповнення OF, що використовується під час виконання дій над числами зі знаком.

У табл. 3.8 наведено приклад додавання 8-розрядних даних, що показує відмінність у модифікації прапорців CF і OF залежно від результату.

Команда ADC (№ 5…8) виконує операцію додавання двох операндів з додаванням значення прапорця CF. Вона використовується для організації додавання багаторозрядних чисел, двійкове представлення яких перевищує шістнадцять розрядів. Спочатку командою ADD підсумовуються молодші розряди, а потім під час додавання більш старших розрядів чисел використовується команда ADC, яка дозволяє враховувати значення можливого перенесення. Джерелом та місцем призначення операндів можуть бути регістр

або пам’ять. Команда ADC має формати, які аналогічні форматам команди ADD. За допомогою команди ADC здійснюється також побайтове додавання, якщо w = 0 багатобайтових чисел, як це робилося в i8080.

Таблиця 3.8. **Приклади встановлення прапорців CF і OF**

Доданки і результат у двійковій системі	**Числа без знака у десятковій системі**	**Числа зі знаком у десятковій системі**	**Доданки і результат у двійковій системі**	**Числа без знака у десятковій системі**	**Числа зі знаком у десятковій системі**
00000101 + 00101010 00101111	5 + 42 47 CF = 0	5 + +42 +47 OF = 0	00100101 + 01101001 10001110	37 + 105 142 CF = 0	+37 + +105 −114 *OF* = 1
00000011 + 11111111 00000010	3 + 255 2 CF = 1	3 + −1 +2 OF = 0	10000101 + 11110111 01111100	133 + 247 124 CF = 1	−123 + −9 +124 *OF* = 1

Команда *INC* (№ 9, 10) збільшує на одиницю (виконує інкремент) вмісту регістра або пам’яті. Необхідно зазначити, що ця команда, так само як і команда *DEC*, впливає на всі прапорці, окрім CF.

Операції *віднімання* мають чотири мнемоніки: SUB (відняти), SBB (відняти з урахуванням позики), DEC (зменшити на одиницю) і NEG (змінити знак).

За командою SUB (№ 13…16) відбувається віднімання операнда джерела з операнда місця призначення. Як і в команді додавання, операнди можуть знаходитися у регістрах або пам’яті. Числом, що віднімається, може також бути операнд (константа), який знаходиться безпосередньо в команді.

Команда SBB (№ 17…20) необхідна для віднімання операндів з урахуванням позики, тому разом з операндами у відніманні бере участь значення прапорця CF.

Команда DEC (№ 21, 22) зменшує на одиницю (виконує декремент) вмісту регістра або пам’яті та має два формати, як і команда INC.

Команда NEG (№ 23) змінює знак операнда, причому отримується представлення операнда в додатковому коді. Наприклад, якщо операнд є: –1 (11111111), то команда NEG змінить його на: +1 (00000001).

Операції *множення* мають дві мнемоніки: MUL (помножити) та IMUL (помножити числа зі знаком)

Командою MUL (№ 31) відбувається множення без знака вмісту акумулятора (AL або AX) на операнд джерела. Результат подвійної довжини повертається до акумулятора і регістра, що використовується для його розширення (до AL і AH у разі 8-розрядних операндів або до AX і DX у разі 16-розрядних операндів). Команда впливає тільки на два прапорці CF і OF, які встановлюються в «1», якщо старша половина результату відмінна від нуля.

На відміну від операцій додавання і віднімання, звичайне множення чисел, які подано у двійковій системі числення, дає правильні результати тільки для чисел без знака. Наприклад, якщо розглядати множення 8-розрядних чисел: 11111111B·11111111B = 1111111000000001B, як 255·255 = 65 025, то результат буде правильним. Якщо ж ці числа розглядати як числа зі знаком (–1)·(–1), то результат –511 буде неправильним. Коли операнди, що перемножуються, і результат розглядаються як числа зі знаком, то використовується команда IMUL (№ 32).

Операції ділення містять дві мнемоніки: DIV (розділити) і IDIV (розділити числа зі знаком). За командою DIV (№ 33) відбувається ділення без знака, операнда подвійної довжини, що знаходиться в акумуляторі та регістрі, що використовується для розширення акумулятора, на операнд зі джерела, яке задано. У разі 8-розрядних операндів ділене – це регістри AL і AH, у разі 16-розрядних операндів – це регістри AX і DX. Результат – частка заноситься до акумулятора (AL або AX), а залишок – до регістра розширення акумулятора (AH або DX). Під час виконання операції ділення прапорці приймають довільні значення. У разі додатного результату, що перевищує максимально допустиме значення, або у разі від'ємного результату, що менше мінімального значення, результат – частка і залишок будуть мати невизначені значення і відбудеться переривання типу 0.

Особливість команди ділення IDIV (№ 34) полягає у тому, що результат – частка і залишок завжди мають однакові знаки. Наприклад, у разі ділення числа 47 на +3 з двох можливих результатів: 15 із залишком 2 і 16 із залишком –1, буде отримано перший результат. Дробові значення результату округлюються до найближчого цілого. Значення прапорців під час виконання команди IDIV також невизначені.

До групи арифметичних операцій належать також дві команди, які здійснюють *розширення* знаком 8-розрядних і 16-розрядних операндів. Ці команди відіграють допоміжну роль під час підготовки операнда, що використовується як ділене. Оскільки ділене у разі ділення 8-розрядних операндів розміщується в 16-розрядному акумуляторі AX, а у разі ділення

16-розрядних операндів – у 32-розрядному складеному регістрі DX, AX, то необхідно під час підготовки діленого заповнити AH або DX. Для чисел без знака вказані регістри заповнюються нулями. У разі ділення чисел зі знаком перед заповненням вказаних регістрів потрібно аналізувати знак діленого і заповнити регістр або нулями, якщо ділене є додатним, або одиницями, якщо ділене від'ємне. Для цього використовуються команди CBW (розширення знаком байта до слова) (№ 36) і CWD (розширення знаком слова до подвійного слова) (№ 37). За командою CBW (CWD) знак AL або AX, тобто старший розряд регістра, записується у всі розряди регістра AH або DX.

До групи арифметичних операцій належать спеціальні команди *десяткової корекції результатів*. Такими командами є: DAA, DAS, AAA, AAS, AAM та AAD.

В основі двійково-десяткового представлення чисел є принцип кодування кожної десяткової цифри групою з чотирьох бітів (тетрадою). Оскільки чотирма бітами можна закодувати шістнадцять різних десяткових комбінацій, а десяткових цифр тільки десять, останні шість кодів у двійково-десятковому представленні не використовуються (табл. 3.9).

Таблиця 3.9. **Двійково-десяткове представлення чисел**

№ з/п	Двійково-десятковий код	Десяткова цифра
1	0 0 0 0	0
2	0 0 0 1	1
3	0 0 1 0	2
4	0 0 1 1	3
5	0 1 0 0	4
6	0 1 0 1	5
7	0 1 1 0	6
8	0 1 1 1	7
9	1 0 0 0	8
10	1 0 0 1	9
11	1 0 1 0	не використовується
12	1 0 1 1	не використовується
13	1 1 0 0	не використовується
14	1 1 0 1	не використовується
15	1 1 1 0	не використовується
16	1 1 1 1	не використовується

Ця умова може бути причиною отримання неправильних результатів під час виконання звичайних арифметичних операцій, наприклад таких, як додавання і віднімання.

Застосування команд корекції залежить від виду арифметичної операції, що виконується над двійково-десятковими числами (додавання, віднімання, множення або ділення) і формату представлення двійково-десяткових чисел.

Є два формати представлення двійково-десяткових чисел:

– запакований;

– незапакований (див. п. 1.2.1).

Окрім названих двох форматів, що застосовуються у двійково-десятковій системі числення, є код ASCII, який застосовується під час обміну інформацією в обчислювальних системах (див. п. 1.2.1). Представлення чисел у коді ASCII показано у табл. 3.10.

Таблиця 3.10. **Представлення чисел за допомогою ASCII-коду**

Цифра	**Код**	**Шістнадцяткове представлення**	**Цифра**	**Код**	**Шістнадцяткове представлення**
0	00110000	30	5	00110101	35
1	00110001	31	6	00110110	36
2	00110010	32	7	00110111	37
3	00110011	33	8	00111000	38
4	00110100	34	9	00111001	39

Відмінність коду ASCII від незапакованого двійково-десяткового формату під час представлення чисел полягає в наявності коду 3D (0011) у старшій тетраді.

Для корекції результату після додавання двійково-десяткових чисел у запакованому форматі служить *команда DAA* (№ 12). У процесі виконання цієї команди виконуються такі дії: якщо (AL)$\wedge$(0FH) > 9, або (AF) = 1, то (AL)$\leftarrow$(AL) + 6, (AF)$\leftarrow$1. Якщо (AL) > 9FH, або (CF) = 1, то (AL)$\leftarrow$(AL) + 60H, (CF)$\leftarrow$1. Команда впливає на прапорці: AF, PF, ZF, SF та CF. Стан прапорця OF не визначено.

Приклад 3.1. Регістри AL і BL містять 29H і 13H, які відображають десяткові числа: 29 і 13 у запакованому двійково-десятковому форматі відповідно.

Виконується така послідовність команд:

ADD AL, BL; AL $\leftarrow$ AL + BL,

DAA.

Після виконання першої команди в регістрі AL буде записано 3CH, яке не є правильним двійково-десятковим числом, що відображає суму операндів. Команда DAA записує в регістр AL скориговане число 42H, яке є правильним двійково-десятковим записом десяткового числа 42.

Команда DAS (№ 30) виконує корекцію результату після віднімання двійково-десяткових чисел у запакованому форматі [2].

За командою DAS вміст регістра AL замінюється правильним двійково-десятковим числом у запакованому форматі.

Корекція після множення і ділення двійково-десяткових чисел у запакованому форматі неможлива.

Для корекції результатів арифметичних дій над BCD-числами у незапакованому форматі використовуються чотири команди: *AAA (корекція для додавання), AAS (корекція для віднімання), AAM (корекція для множення), AAD (корекція для ділення)* [2].

Команда AAA (№ 11) виконує корекцію результату після додавання BCD-чисел у незапакованому форматі.

Команда AAA підганяє чисельну суму двох розпакованих BCD-цифр до розпакованого BCD-формату, який легко перетворюється в ASCII-код.

Команда AAS (№ 28) виконує корекцію результату після віднімання BCD-чисел у незапакованому форматі.

Команда AAM (№ 29) виконує корекцію результату після множення BCD-чисел у незапакованому форматі.

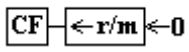
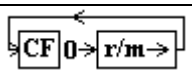
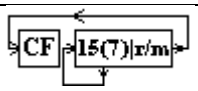
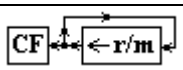
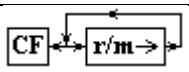
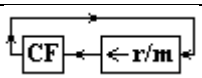
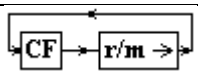
Команда AAD (№ 35) виконує перетворення двох незапакованих BCD-чисел, розміщених в AX, у двійкове 8-розрядне число.

Команди порівняння CMP необхідні для порівняння двох операндів та виконуються за допомогою віднімання значення другого операнда від першого операнда. На відміну від звичайного віднімання, отримана різниця нікуди не заноситься, а результатом операції порівняння є значення прапорців CF та ZF, які встановлюються залежно від співвідношення операндів, що порівнюються (табл. 3.9). Команда CMP має формати, які аналогічні командам віднімання: № 24…27.

3.4.1.4. Логічні команди

Логічні команди (табл. 3.11) використовуються для реалізації чотирьох основних булевих функцій: AND (порозрядне логічне І), OR (порозрядне логічне АБО), XOR (виключне АБО/порозрядна логічна сума за модулем 2/⊕ – псевдоплюс/нееквівалентність) та NOT (порозрядне логічне НЕ).

Таблиця 3.11. **Команди логічні, зсуву та обробки рядків**

№ з/п	Мне-моніка	Функція (алгоритм)	Формат				Кількість байт	Кількість тактів
			1 байт	2 байти	3 байти	4 байти		
1	NOT	r/m ← $\overline{\text{r/m}}$	1111011w	md010 r/m	(Не змінює прапорці)		2...4	3/16 + $T_{ва}$
2	AND	r/m ← r/m&reg	0010000w	mdreg r/m			2...4	16 + $T_{ва}$
3	AND	reg ← reg&r/m	0010001w	mdreg r/m			2...4	3/9 + $T_{ва}$
4	AND	r/m ← r/m&data	1000000w	md100 r/m	data L	data H, w=1	3...6	4/17 + $T_{ва}$
5	AND	A ← A&data	0010010w	data L	data H, w=1		2, 3	4
6	OR	r/m ← r/m ∨ reg	0000100w	mdreg r/m			2...4	16 + $T_{ва}$
7	OR	reg ← reg ∨ r/m	0000101w	mdreg r/m			2...4	3/9 + $T_{ва}$
8	OR	r/m ← r/m∨data	1000000w	md001 r/m	data L	data H, w=1	3...6	4/17 + $T_{ва}$
9	OR	A ← A ∨ data	0000110w	data L	data H, w=1		2, 3	4
10	XOR	r/m ← r/m⊕reg	0011000w	mdreg r/m			2...4	16 + $T_{ва}$
11	XOR	reg ← reg⊕r/m	0011001w	mdreg r/m			2...4	3/9 + $T_{ва}$
12	XOR	r/m ← r/m⊕data	1000000w	md110 r/m	data L	data H, w=1	3...6	4/17 + $T_{ва}$
13	XOR	A ← A⊕data	0011010w	data L	data H, w=1		2, 3	4
14	TEST	r/m & reg	1000010w	mdreg r/m	(Результат кон'юнкції відкидається, впливає на прапорці)		2...4	3/9 + $T_{ва}$
15	TEST	r/m & data	1111011w	md000 r/m	data L	data H, w=1	3...6	4/10 + $T_{ва}$
16	TEST	A & data	1010100w	data L	data H, w=1		2, 3	4
17	SHL / SAL		110100vw	md100 r/m	Логічний/арифметичний зсув вліво (множення на 2 чисел без знака/зі знаком)		2...4	
18	SHR		110100vw	md101 r/m	Логічний зсув вправо (ділення на 2 чисел без знака)		2...4	
19	SAR		110100vw	md111 r/m	Арифметичний зсув вправо (ділення на 2 чисел зі знаком)		2...4	2 (reg-один раз)
20	ROL		110100vw	md000 r/m	Циклічний зсув вліво з копіюванням у C_F значення старшого біта		2...4	15 + $T_{ва}$ (Mem-один раз)
21	ROR		110100vw	md001 r/m	Циклічний зсув вправо з копіюванням у C_F значення молодшого біта		2...4	4·CL + 8 (reg-CL раз)
22	RCL		110100vw	md010 r/m	Циклічний зсув вліво з включенням C_F у ланцюг зсуву		2...4	4·CL + + 20 + $T_{ва}$ (Mem-CL раз)
23	RCR		110100vw	md011 r/m	Циклічний зсув вправо з включенням C_F у ланцюг зсуву		2...4	

Закінчення табл. 3.11

24	MOVS	M(DI)←M(SI)	1010010w	(MOVSB → w=0; MOVSW→w=1)	1	18/ (9 + 17)*N
25	CMPS	M(DI) – M(SI)	1010011w	(CMPSB→ w = 0; CMPSW→w = 1)	1	
26	SCAS	A – M(DI)	1010111w	(A–M(ESx16+DI))	1	15/ (9 + 15)*N
27	LODS	A ← M(SI)	1010110w	LODSB→w = 0 A←M(DSx16+SI) LODSW→w = 1	1	12/(9 + + 13) *N
28	STOS	M(DI) ← A	1010101w	STOSB→w = 0 M(ESx16+DI)←A STOSW→w = 1	1	11/ (9 + 10)*N
29	REP	Повторювати команду, доки CX<>0	1111001x	Префікс команд MOVS or STOS	1	Додатково 6 тактів на цикл
30	REPE REPZ	Повторювати команду, доки CX<>0; ZF=1	1111001z z = 1	Префікс команд CMPS or SCAS	1	
31	REPNE REPNZ	Повторювати команду, доки CX<>0; ZF=0	1111001z z = 0	Префікс команд CMPS or SCAS	1	

Примітка. v = 0 : кількість зсувів = 1; v = 1 : кількість зсувів = (CL); z = 1: REPE/REPZ – повторювати команду, доки CX<>0; ZF = 1 (доки однакові); z = 0: REPNE/REPNZ – повторювати команду, доки CX<>0; ZF = 0 (доки не однакові); x – може приймати довільне значення; N – кількість елементів рядків, які обробляються; & – логічне «І»; ∨ – логічне «АБО»; ⊕ – додавання за модулем два/виключне АБО/нееквівалентність.

Сюди також належить команда TEST (перевірка), яка полягає у порозрядному логічному множенні (І) операндів без занесення результату множення у місце призначення і необхідна для аналізу вмісту джерела за значеннями прапорців.

Усі двоопера́ндні команди AND (№ 2…4), OR (№ 6…8), XOR (№ 10…12) і TEST (№ 14…16) мають по три однакових формати та збігаються за часом виконання.

Окрім цього, команди AND (№ 5), OR (№ 9) та XOR (№ 13) мають четвертий формат, в якому перший операнд (акумулятор: AL або AX) адресується неявно, а другий (безпосередній операнд) міститься у самій команді. Одноопера́ндна команда NOT (№ 1) здійснює інвертування операнда, має один формат і прапорці не змінює.

Команди AND, OR, XOR і TEST впливають на прапорці PF, SF і ZF. Прапорці CF і OF завжди скидаються у нуль, оскільки відсутні перенесення і переповнення відповідно. Прапорець AF не визначено. Нижче наведено приклади використання деяких логічних команд.

Приклад 3.2. Командою ANL AL, #D8; AL←AL ∧#D8 скинути у нуль парні біти акумулятора AL, а інші біти залишити без зміни.

```
            7p  6  5  4   3  2  1 0p
  AL→ X X X X   X X X X      ; x=0/1; x*0=0; x*1=x,
∧
  # D8 → 1  0  1  0   1  0  1 0      ;  #D8 = AAh,
  ------------------------------------
  AL← X  0  X 0  X  0  X  0.
```

Приклад 3.3. Командою ORL AL, #D8; AL←AL∨#D8 встановити в одиницю непарні біти акумулятора AL, а інші біти залишити без зміни.

```
            7p 6  5  4   3  2  1 0p
  AL→ X X X X   X X X X      ; x=0/1; x+1=1; x+0=x,
∨ #D8→1  0  1  0   1  0  1  0     ;  #D8 = AAh,
  ------------------------------------
  AL← 1  X  1  X   1  X 1 X.
```

Приклад 3.4. Командою XRL AL, #D8; AL←AL⊕ #D8 проінвертувати старшу тетраду акумулятора AL, а інші біти залишити без зміни.

```
            7p 6   5  4   3  2  1 0p
  AL→ X  X  X X  X X X X      ; x=0/1; x⊕0=x; x ⊕ 1 = x̄.
⊕
  #D8 →1  1  1  1   0  0  0  0      ;  #D8 = F0h,
  ------------------------------------
  AL← x̄  x̄  x̄  x̄  X X X X.
```

Приклад 3.5. Командою CPL AL, AL←$\overline{AL}$ проінвертувати вміст регістра AL.

AL → 0110 1101B – до виконання команди CPL,

AL ← 1001 0010B – після виконання команди CPL.

3.4.1.5. Команди зсуву

МП, що розглядається, може виконувати команди *зсуву* трьох типів (табл. 3.11):

– логічний (вліво/вправо), команди SHL (№ 17) і SHR (№ 18) відповідно;
– арифметичний (вліво/вправо), команди SAL (№ 17) і SAR (№ 19) відповідно;
– циклічний (вліво/вправо), команди ROL (№ 20), RCL (№ 22), ROR (№ 21), RCR (№ 23) відповідно.

Особливості виконання різних типів зсувів пояснюються у табл. 3.11 під час опису функцій, що виконуються названими командами. Довжина операнда, що зсувається, визначається бітом W першого байта машинного коду команди

(W = 0 – 8-розрядний операнд, W = 1 – 16-розрядний операнд). Значення біта V визначає кількість зсувів, що здійснюються однією командою. Якщо V = 0, то здійснюється зсув на один розряд, а якщо V = 1 відповідна команда зсуву буде виконана *n* разів, де кількість зсувів *n* вказується у регістрі-лічильнику CL. Приклад використання команди SAR, якщо *n* = 3, наведено на рис. 3.35.

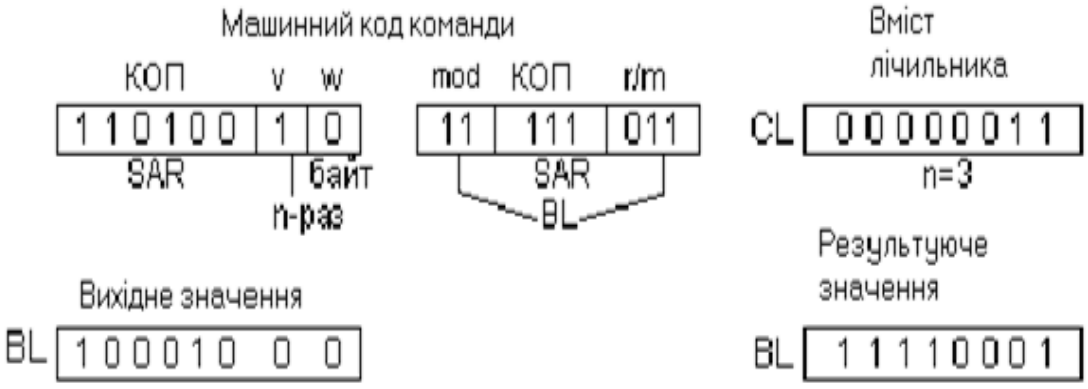


Рис. 3.35. Приклад використання команди SAR, якщо *n* = 3

Команди логічного й арифметичного зсувів можуть застосовуватися передусім для подвоєння чисел і ділення на 2. Для подвоєння числа без знака досить зсунути всі його розряди на один розряд вліво, заповнивши молодший біт нулем (команда SHL, № 17). Оскільки старший біт під час зсуву передається у розряд CF, аналізування значення цього прапорця визначає точність результату.

Наприклад, подвоєння числа 65 (01000001) за допомогою зсуву вліво дає точний результат 130 (10000010), що визначається значенням CF = 0. Подальше подвоєння дасть неправильний результат 4 (00000100), що визначається значенням CF = 1.

Аналогічно використовуючи логічний зсув вправо (команда SHR, № 18), здійснюється діленням числа без знака на 2. Наприклад, зсуваючи число 13 (00001101) вправо, отримуємо 6 (00000110) і CF = 1, що свідчить про неточний результат. Аналогічні операції для чисел зі знаком реалізовуються за допомогою команд SAL (арифметичний зсув вліво) і SAR (арифметичний зсув вправо).

Відмінність ділення на 2 чисел без знака і чисел зі знаком полягає у тому, що за командою SHR старший біт заповнюється нулем, а за командою SAR значення у старшому біті не змінюється і передається у наступний молодший розряд. Наприклад, арифметичний зсув числа –120 (10001000B)

вправо на 3 розряди (рис. 3.35) дає результат –15 (11110001B). Команди SHL і SAL реалізують ті ж самі дії, тому коди операцій цих команд однакові.

Для здійснення циклічних зсувів вліво і вправо, без участі або з участю прапорця CF використовуються команди ROL (циклічний зсув вліво), ROR (циклічний зсув вправо), RCL (циклічний зсув вліво через прапорець CF) і RCR (циклічний зсув вправо через прапорець CF).

3.4.1.6. Команди обробки рядків

МП, що розглядається, може виконувати *команди обробки рядків*. Рядком називають послідовність байтів або слів, що розміщуються у суміжних комірках пам'яті. Прикладом може бути рядок, який вводиться з термінала у МПС. Операції над рядками (табл. 3.11) виконуються над кожним елементом рядка (байтом або словом), тому час їх виконання пропорційний кількості елементів у рядку. У 16-розрядному МП, наприклад, і8086 є декілька однобайтних команд, що виконують прості операції над елементами рядків. Ці команди значно прискорюють маніпуляції над рядками завдяки зменшенню часу, що витрачається на обробку кожного елемента і на допоміжні дії, які необхідно виконувати під час обробки послідовності елементів. До допоміжних дій належать: переадресація елемента рядка, зменшення на одиницю лічильника кількості елементів рядка, що обробляється, і перевірка досягнення лічильником нуля.

Для ілюстрації часових витрат на обробку елементів рядків розглянемо процес пересилання рядка з однієї області пам'яті в іншу.

Для вказівки початкової адреси першого елемента початкового рядка і рядка результату використовуються регістри SI (індекс джерела) і DI (індекс місця призначення). Відповідно, кількість елементів рядків, що пересилаються, задається у регістрі-лічильнику CX. Пересилання рядка полягає у виконанні таких кроків (у цьому описанні кожного кроку вказано відповідну команду і кількість тактів, які необхідні для її виконання):

а) якщо CX = 0, то пересилання закінчено;

б) переслати байт з комірки з непрямою адресою, яку вказано у SI, до AL: MOV AL, [SI]; AL ← M(DS:SI); 9T + 5T;

в) запам'ятати AL в елементі пам'яті з непрямою адресою, яку вказано у DI: MOV [DI], AL; M(DS:DI) ← AL; 9T + 5T;

г) збільшити SI на «1»: INC SI; SI ← SI + 1; 2T;

д) збільшити DI на «1»: INC DI; DI ← DI + 1; 2T;

е) зменшити CX на «1»: DEC CX; CX ← CX – 1; 2T;

ж) перейти до кроку б), якщо CX ≠ 0 (ZF = 0); JNZ Мітка; 8 тактів;

и) перейти до кроку а), якщо CX = 0 (ZF = 1); 4 такти.

Власне пересилання байта виконується на кроках б) і в). На кроках г) і д) здійснюється обчислення адрес джерела і місця призначення для наступного елемента рядка. Нарешті, кроки а), ж) та и) використовуються для визначення кількості оброблених елементів. Кількість тактів, які витрачаються на виконання кроків б)…е), дорівнює 34. Дії, які виконуються за допомогою вказаних чотирьох команд протягом кроків б)…е), можуть бути виконані за допомогою однієї команди MOVSB (№ 24) (переслати однобайтний елемент рядка), що вимагає тільки 18 тактів.

Для організації циклічного виконання команди MOVS (а також інших команд обробки рядків) використовується однобайтова команда-префікс REP (повторити). Вона записується перед основною командою, наприклад, REP MOVS, і забезпечує її виконання N разів.

Кількість повторень N заздалегідь записується до регістра CX і на кожному кроці зміст CX зменшується на одиницю.

Використовуються три мнемоніки команди-префікса: REP, REPE/REPZ і REPNE/REPNZ (№ 29, 30, 31). Перша мнемоніка аналізує тільки одну умову: доки CX ≠ 0, продовжуються повторення. У тих випадках, коли виникає необхідність додатково аналізувати умову ZF = 1 (або ZF = 0), використовується друга (або третя) мнемоніка. Команда REP дозволяє ще більше скоротити час на обробку рядка шляхом за допомогою поєднання операцій (CX) <– (CX) – 1 з перевіркою досягнення лічильником нуля.

У низці випадків виникає необхідність у пересиланні рядка у зворотному порядку починаючи не з першого, а з останнього її елемента. Наприклад, потрібно переслати рядок, що містить 10 байт і розміщений у пам’яті у діапазоні адрес (SI) = 200…209 в область пам’яті у діапазоні адрес (DI) = 205…214. Зрозуміло, що таке пересилання не можна виконувати починаючи з першого елемента, оскільки області пам’яті з вихідним рядком і рядком-результатом перекриваються і перше ж пересилання «зіпсує» значення п’ятого елемента вихідного рядка. Необхідне пересилання рядків можна здійснити починаючи з останнього елемента початкового рядка. У цьому випадку потрібно встановити початкові адреси елементів рядків рівними (SI) = 209 і (DI) = 214. Окрім того, під час пересилання кожного елемента необхідно не збільшувати, а зменшувати на одиницю вміст індексних регістрів SI і DI. Напрямок передачі встановлюється за допомогою прапорця DF: якщо

значення DF = 1 відбувається автодекрементування індексних регістрів, а якщо значення DF = 0 – автоінкрементування.

Коли як елементи рядків використовуються не байти, а слова, відповідне збільшення або зменшення значень індексних регістрів здійснюється на два, тому якщо DF = 1: SI ← SI – 2; DI ← DI – 2, а якщо DF = 0: SI ← SI + 2; DI ← DI + 2.

Окрім команди MOVS для дії з рядками є ще чотири команди: CMPS (№ 25 – порівняння елементів рядків), SCAS (№ 26 – сканування елементів рядка), LODS (№ 27 – завантаження елемента рядка) і STOS (№ 28 заповнення елемента рядка).

Команда CMPS (№ 25) дозволяє здійснити поелементне порівняння двох рядків, один з яких розміщується у пам’яті з непрямою адресою, яку вказано у SI, а другий – з непрямою адресою, яку вказано у DI. За командою CMPS відбувається віднімання елемента рядка з адресою у SI з елемента рядка з адресою у DI.

Під час виконання операції порівняння результат віднімання не фіксується, а встановлюються відповідні значення прапорців: ZF та CF, завдяки яким визначається результат порівняння. Аналогічно з командою MOVS під час порівняння змінюються значення індексних регістрів за правилом: SI ← SI ± Δ; DI ← DI ± Δ, де «+» використовується, якщо DF = 0; «–», якщо DF = 1; Δ = 1, якщо W = 0; Δ = 2, якщо W = 1.

Для циклічного повторення команди CMPS використовується префікс повторення REPNE/REPNZ або REPE/REPZ (повторювати доки елементи, що порівнюються, не дорівнюють (ZF = 0) або доки елементи, що порівнюються, рівні (ZF = 1)).

За командою сканування SCAS (№ 26) відбувається порівняння значення елемента рядка, який розміщено з адреси, яку вказано у DI, зі значенням AL. У цьому разі також здійснюється операція віднімання, результат якої не фіксується. Одне з можливих застосувань команди сканування складається у пошуку елемента рядка, що дорівнює заданому зразку. Зразковий елемент завантажується в AL і потім організується цикл сканування з використанням префікса повторення REPNE/REPNZ. Як і у разі виконання попередніх команд, кожного разу за командою SCAS відбувається автоінкремент (або автодекремент) адреси елемента: (DI) ← (DI) ± Δ.

Дві наступні команди LODS (завантаження елемента рядка № 27) і STOS (запис елемента рядка № 28), дозволяють завантажувати елементи рядка в акумулятор і записувати вміст акумулятора у рядок. Під час виконання цих

команд здійснюється підготовка адреси наступного елемента рядка, тобто модифікується вміст відповідного індексного регістра. Однак повторення команди LODS звичайно не використовується, а повторення команди STOS, що організується за допомогою префікса REP, застосовується у разі завантаження рядка константою, яку заздалегідь записано в акумулятор.

Розглянуті вище команди роботи з рядками реалізують відносно прості операції, на основі яких можна організувати складніші операції обробки рядків.

3.4.1.7. Команди безумовних переходів

Команди безумовних переходів мають мнемоніку JMP (табл. 3.12).

Команда JMP дозволяє здійснити перехід у будь-яку точку програми, яку розміщено як у поточному програмному сегменті, так і в іншому сегменті програм. Під час переходу у межах поточного програмного сегмента використовуються перші три формати команди JMP (№ 1...3).

Перший формат команд безумовних переходів – це перехід у довільну точку програми в межах поточного програмного сегмента, для чого до вмісту IP додається 16-розрядний зсув у додатковому коді, старший розряд якого є знаковим (діапазон переходів відносно адреси наступної команди: –32768…+32767). Другий, скорочений формат дозволяє перейти до точки програми, що знаходиться на відстані не більш ніж на –128…+127 від адреси наступної за JMP команди. Нарешті, третій формат здійснює завантаження покажчика команд 16-розрядним числом, яке або розміщено за виконавчою адресою (ефективною адресою), що визначається постбайтом, або знаходиться в одному з регістрів загального призначення. Останній перехід називається непрямим, оскільки використовується непряма адресація (адреса переходу знаходиться в одному з регістрів загального призначення або у двох комірках пам’яті).

Для реалізації безумовного переходу до точки програми, яку розміщено поза поточним програмним сегментом та коли потрібне перезавантаження сегментного регістра CS, використовуються четвертий і п’ятий формати JMP. Четвертий формат визначає прямий міжсегментний перехід, у разі якого у другому і третьому байтах машинного коду команди вказане нове значення регістра IP (PC), а в четвертому і п’ятому байтах – нове значення регістра CS. П’ятий формат за допомогою постбайта дозволяє визначити виконавчу адресу (ефективну адресу), за якою в пам’яті знаходяться нове значення регістра IP (PC): адреси EA, EA + 1 і нове значення регістра CS: адреси EA + 2, EA + 3.

Таблиця 3.12. **Команди передачі керування**

№ з/п	Мнемо-ніка	Функція (алгоритм)	Формат					Кількість байтів	Кількість тактів
			1 байт	2 байти	3 байти	4 байти			
1	JMP	IP←IP+disp16	11101001	disp L	disp H	(disp16 – число зі знаком)		3	15
2	JMP	IP ← IP+disp8	11101011	disp8	(disp8 – число зі знаком)			2	15
3	JMP	IP ← r/m	11111111	md100 r/m				2...4	11/18 + $T_{ва}$
4	JMP	IP, CS←ip,cs	11101010	ip_L	ip_H	cs_L	cs_H (5-й байт)	5	15
5	JMP	IP, CS ← r/m	11111111	md101 r/m				2...4	24 + $T_{ва}$
6	CALL	Стек ← IP, IP ← IP + disp	11101000	disp L	disp H			3	19
7	CALL	Стек ← IP, IP ← r/m	11111111	md010 r/m				2...4	16/ 21 + $T_{ва}$
8	CALL	Стек ← IP,CS IP, CS←ip,cs	10011010	ip_L	ip_H	cs_L	cs_H (5-й байт)	5	28
9	CALL	Стек ← IP,CS IP, CS ← r/m	11111111	md011 r/m				2...4	37 + $T_{ва}$
10	RETN	IP ← стек	11000011					1	8
11	RETN	IP ← стек SP←SP + data	11000010	data L	data H			3	17
12	RETF	CS, IP ← стек	11001011					1	12
13	RETF	CS, IP ← стек SP ← SP+data	11001010	data L	data H			3	18
14	JCC (псевдо-мнемо-ніка (див. табл. 3.13)	IP ← IP+disp8, якщо виконується умова: cccc	0111cccc	disp8	disp8 – число зі знаком у додатковому коді			2	8/4

15	LOOP	CX ← CX-1 IP ← IP+disp8, якщо (CX)<>0	11100010	disp8			2	17/5
16	LOOPZ LOOPE	CX ← CX-1 IP ← IP+disp8, якщо (CX)<>0,ZF=1	11100001	disp8			2	18/6
17	LOOPNZ LOOPNE	CX ← CX-1 IP ← IP+disp8, якщо (CX)<>0,ZF=0	11100000	disp8			2	18/6
18	JCXZ	IP ← IP+disp8, якщо (CX) = 0	11100011	disp8			2	18/6
19	INTN (INT type)	Стек←F,CS,IP IP ← (N*4); CS ← (N*4+2)	11001101	N(type)	Порядок запису у стек: F(RF); CS; IP		2	52
20	INT3	Стек←F, CS, IP IP ← (3*4); CS ← (3*4+2)	11001100				1	52
21	INTO	Стек←F, CS, IP; IP ← (4*4); CS ←(4*4+2), якщо OF = 1	11001110				1	52 або 4
22	IRET	F, CS, IP←стек	11001111	Порядок читання зі стека: IP; CS; F(RF).			1	24
23	CLC	CF ← 0	11111000				1	2
24	CMC	CF ← $\overline{CF}$	11110101				1	2
25	STC	CF ← 1	11111001				1	2
26	CLD	DF ← 0	11111100				1	2

Закінчення табл. 3.12

27	STD	DF ← 1	11111101				1	2
28	CLI	IF ← 0	11111010				1	2
29	STI	IF ← 1	11111011				1	2
30	HLT	Зупинка МП	11110100				1	2
31	WAIT	Переведення МП у стан очікування	10011011				1	3
32	ESC	Зовнішня команда для співпроцесора	11011xxx	mdYYY r/m			2...4	8 + $T_{ва}$
33	LOCK	Блокування шин МПС	11110000				1	2
34	NOP	Немає операції	10010000				1	2

Примітка. disp – зсув; L – молодший байт; H – старший байт; cccc – 4-розрядний код умови (табл. 3.12); N – тип переривання (0…255); xxx – вказує номер співпроцесора; YYY – вказує номер (код операції), який повинен виконувати співпроцесор.

3.4.1.8. Команди умовних переходів

Повний список мнемонік команд умовних переходів, умови переходів, коди операцій, а також відповідні булеві комбінації прапорців і їх значення наведено у табл. 3.13.

Таблиця 3.13. **Команди умовних переходів**

№ з/п	Мнемоніка	Умова переходу	Значення прапорців	КОП команди
Для чисел зі знаком				
1	JL / JNGE	Менше / не більше і не дорівнює $(< / \not\geq)$	SF ⊕ OF = 1	1100
2	JNL / JGE	Не менше / більше або дорівнює $(\not< / \geq)$	SF ⊕ OF = 0	1101
3	JG / JNLE	Більше / не менше і не дорівнює $(> / \not\leq)$	(SF⊕ OF) ∨ ZF = 0	1111
4	JNG / JLE	Не більше / менше або дорівнює $(\not> / \leq)$	(SF⊕ OF) ∨ ZF = 1	1110
Для чисел без знака				
5	JB / JNAE / JC	Менше / не більше і не дорівнює $(< / \not\geq)$	CF = 1	0010
6	JNB / JAE / JNC	Не менше / більше або дорівнює $(\not< / \geq)$	CF = 0	0011
7	JA / JNBE	Більше / не менше і не дорівнює $(> / \not\leq)$	CF ∨ ZF = 0	0111
8	JNA / JBE	Не більше / менше або дорівнює $(\not> / \leq)$	CF ∨ ZF = 1	0110
Інші умовні переходи				
9	JE / JZ	Дорівнює / за нулем	ZF = 1	0100
10	JNE / JNZ	Не дорівнює / якщо не нуль	ZF = 0	0101
11	JS	За мінусом	SF = 1	1000
12	JNS	За плюсом	SF = 0	1001
13	JO	За переповненням	OF = 1	0000
14	JNO	За відсутністю переповнення	OF = 0	0001
15	JP / JPE	За парністю	PF = 1	1010
16	JNP / JPO	За непарністю	PF = 0	1011

Команди умовних переходів здійснюють передачу керування залежно від результатів виконання попередніх операцій. Існують 16 різновидів команд умовних переходів, які у табл. 3.12, № 14 показано одним рядком.

Усі команди умовних переходів виконуються за 8 тактів, якщо умова виконується (є перехід); 4 такти, якщо умова не виконується (немає переходу).

Мнемоніки всіх команд умовних переходів мають першу літеру J (JMP), а наступні букви, які у рядку № 14 позначено як CCCC, в конкретних командах будуть мати те, чи інше значення залежно від виконуваної команди умовного переходу (табл. 3.13).

Розрізняють три різновиди умовних переходів, які використовуються для встановлення співвідношень чисел зі знаком, або чисел без знака та переходів залежно від значень окремих прапорців. У перших двох різновидах для одних і тих самих співвідношень між числами обираються різні мнемоніки команд, оскільки одним і тим самим співвідношенням чисел зі знаком і чисел без знака відповідають різні значення прапорців. У разі порівняння чисел зі знаком для позначення умови «більше» використовується буква G (Greater – більше), а для позначення – «менше» буква L (Less – менше). Для аналогічних умов у разі порівняння чисел без знака використовуються відповідно букви A (Above – над) і B (Below – під). Умови рівності позначаються буквою E (Equal – дорівнює), а не виконання деякої умови – буквою N (Not – не).

Потрібно зазначити, що допускається використання двох різних мнемонік для кожної команди; наприклад, мнемоніки JL і JNGE – еквівалентні, оскільки умови «менше» і «не більше або дорівнює» – ідентичні.

Всі команди умовних переходів мають однаковий двобайтовий формат (табл. 3.12, № 14), у першому байті якого задається коди операцій, а у другому – 8-розрядний зсув, який розглядається як число зі знаком і, отже, дозволяє здійснювати переходи у діапазоні: –128…+127 відносно адреси наступної команди. У разі необхідності більш віддаленого («далекого») переходу під час виконання умови використовується додатково команда безумовного переходу.

Час виконання кожної з команд умовних переходів вказано для двох випадків:

– умову виконано і керування дійсно передається згідно зі значенням другого байта команди;
– умову не виконано, тому керування передається наступній команді.

3.4.1.9. Команди організації циклів

Команди організації циклів (табл. 3.12) введено в МП i8086 для зручності виконання обчислювальних циклів. До них належать такі мнемоніки (команди з номерами 15…18): LOOP (цикл, доки (CX) ≠ 0), LOOPNZ/LOOPNE (цикл, доки (CX) ≠ 0 і не нуль/не дорівнює), LOOPZ/LOOPE (цикл, доки (CX) ≠ 0 і нуль/дорівнює) і JCXZ (перехід за нулем у регістрі CX).

Кожна з цих команд має двобайтовий формат, у другому байті якого вказується 8-розрядний зсув, що використовується для організації переходу. Цей зсув розглядається як число зі знаком і перед обчисленням адреси переходу воно

розширяється знаком до шістнадцяти розрядів. Вказане першим у табл. 3.12 кількість тактів, що дорівнює 17/18, відповідає випадку, коли умова переходу в цих командах виконується. Якщо умови не виконані, названі команди виконуються за п’ять/шість тактів. Як лічильник циклів використовується регістр CX, вміст якого у процесі виконання кожної з названих команд організації циклів (окрім JCXZ) зменшується на одиницю (декрементується).

3.4.1.10. Команди виклику і повернення з підпрограм

Команда CALL дозволяє викликати підпрограму, яку розміщено як у поточному програмному сегменті, так і в іншій області пам’яті. Вона має такі ж формати, що і команда JMP, за винятком скороченого (команди з номерами 6…9 у табл. 3.12).

На відміну від команди JMP аналогічного формату за командою CALL перед зміною значень IP або IP і CS відбувається автоматичний запис у стек поточних значень цих регістрів, що забезпечує запам’ятовування точки повернення з підпрограми.

Для повернення з підпрограми використовується команда RET, під дією якої відбувається передача керування за адресою повернення, яку занесено у стек під час виконання попередньої команди CALL. Під час повернення з підпрограм, які розміщено у поточному програмному сегменті, застосовуються перші два формати команди RET (№ 10, 11), причому другий формат відрізняється від першого тим, що до вмісту покажчика стека додається константа, яку записано у другому і третьому байтах команди. Це дозволяє одночасно з поверненням з підпрограми скидати параметри, які записано у стек під час виконання цієї підпрограми і які не використовуються надалі.

Для міжсегментного повернення застосовуються третій і четвертий формати RET (№ 12, 13), які забезпечують відновлення зі стека вмісту як регістра CS, так і програмного лічильника IP.

3.4.1.11. Команди програмних переривань

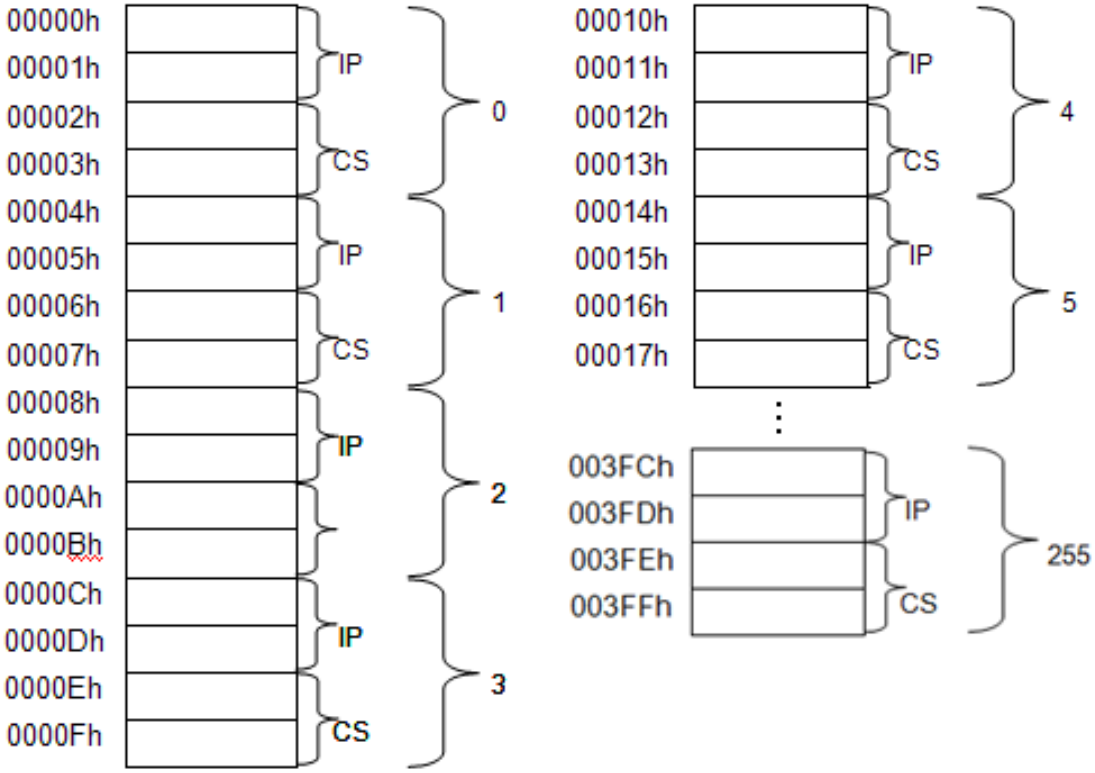
Команди програмних переривань (табл. 3.12) включають три мнемоніки: INT (переривання), INTO (переривання за переповненням) і IRET (повернення з підпрограми обробки переривання).

Команда переривання INT type (№ 19) має двобайтовий формат, другий байт якого містить 8-розрядне число, що визначає тип (type)/рівень переривання. За командою INT type процесор переходить до виконання підпрограми обслуговування переривання вказаного рівня, причому автоматично виконуються дії, необхідні для забезпечення повернення у точку переривання.

Ці дії полягають у такому: вміст регістра прапорців F записуєтьсяу стек (PUSHF), скидаються прапорці IF і TF (IF ← 0, TF ← 0), поточні значення регістра CS і покажчика команд IP записуються у стек (стек ← CS, стек ← IP).

Для визначення початкової адреси підпрограми обслуговування переривання згідно зі значенням type використовується табл. 3.14.

Таблиця 3.14. **Таблиця адрес підпрограм-обробників переривань**



Для кожного з 256 (0…255) рівнів переривань у цій таблиці відведено по чотири байти: перші два байти визначають значення покажчика команд IP, другі – значення сегментного регістра CS. Ця четвірка байтів визначає початкові адреси підпрограм обслуговування переривань (пари значень CS, IP), які мають бути заздалегідь записані до комірок пам'яті за абсолютними адресами 00000H…003FFH. Адреса входу до табл. 3.14, яка відповідає вказаному в команді INT type рівню переривання, визначається в МП таким чином. Після запам'ятовування у стеку поточних значень регістра прапорців F, регістрів CS і IP здійснюються завантаження: $IP_L \leftarrow M(type \cdot 4)$; $IP_h \leftarrow M(type \cdot 4+1)$; $CS_L \leftarrow M(type \cdot 4+2)$; $CS_h \leftarrow M(type \cdot 4+3)$. Нові значення CS і IP визначають початкову адресу необхідної підпрограми обслуговування переривання.

Команда переривання за переповненням INTO викликає перехід на обслуговування переривання четвертого рівня (type = 4) у випадку, коли значення прапорця переповнення OF = 1. Команда INTO зазвичай

використовується після арифметичних команд над числами зі знаком.

Декілька перших рівнів переривань (до 32-х) резервуються під обробку низки специфічних ситуацій, таких, наприклад, як спроба ділення на нуль, переповнення і тому подібних. Призначення перших п’яти зарезервованих рівнів вказано у табл. 3.15.

Таблиця 3.15. **Зарезервовані рівні переривань**

Номер рівня	**Адреса таблиці**	**Призначення**
0	00000H	Обробка ситуації «ділення на нуль»
1	00004H	Робота у покроковому режимі (TF = 1)
2	00008H	Обслуговування запиту переривання, що не маскується (за входом NMI)
3	0000CH	Обслуговування запиту переривання за однобайтовою командою INT3 (type = 3)
4	00010H	Обробка ситуації «переповнення» (INTO)

Особливість обробки переривань зарезервованих рівнів полягає у тому, що процесор переходить до їх обслуговування незалежно від значення прапорця дозволу переривань IF.

Однобайтова команда IRET ставиться у кінці кожної підпрограми обслуговування переривання і забезпечує повернення з переривання. За цією командою процесор витягує зі стека значення покажчика команд IP і сегментного регістра CS, а також відновлює колишній вміст регістра прапорців F (як і за командою POPF). Під час переходу на підпрограму обслуговування переривання у разі необхідності вміст інших регістрів МП, що відповідає програмі, що переривається, може бути збережено у стеку за допомогою команд PUSH – запис до стека, а потім відновлено під час повернення з підпрограми за допомогою команд POP – читання зі стека.

3.4.1.12. Команди керування мікропроцесором

Розрізняють три типи команд керування МП: операції з прапорцями, встановлення МП в особливі стани і синхронізація зі співпроцесорами (табл. 3.12).

3.4.1.13. Операції з прапорцями

Ці команди містяться в табл. 3.12 і включають 7 мнемонік: STC (№ 25, встановлення прапорця перенесення CF), CMC (№ 24, інверсія прапорця перенесення CF), CLC (№ 23, скидання прапорця перенесення CF), STD (№ 27, встановлення прапорця напрямку DF), CLD (№ 26, скидання прапорця напрямку

DF), STI (№ 29) і CLI (№ 28), відповідно встановлення і скидання прапорця дозволу переривання IF. Призначення перерахованих команд очевидне. Так команди STC, CMC і CLC дозволяють задавати необхідне початкове значення прапорця CF у разі різних арифметичних і логічних перетворень даних і зсувах. Команди STD і CLD використовуються під час обробки рядків для задання напрямку обробки: від першого елемента рядка до останнього або навпаки. Нарешті, команди STI та CLI служать для керування системою переривань і дозволяють відповідно дозволити або заборонити зовнішнє переривання у будь-якій частині програми.

3.4.1.14. Команди встановлення мікропроцесора в особливі стани

Ці команди (табл. 3.12) містять дві мнемоніки: HLT (№ 30 – зупинка) і WAIT (№ 31 – очікування), які переводять процесор відповідно у стан зупинки або очікування.

Знаходячись у будь-якому з цих станів, процесор не виконує жодних дій (на часових діаграмах роботи МП вводяться такти очікування TW), поки не з’являться певні зовнішні впливи. Зі стану зупинки процесор може бути виведено двома способами: за допомогою початкового скидання (сигналом за входом RESET) або зовнішнім перериванням (сигналом запиту за входом INT). У разі першого способу процесор перейде до виконання основної програми з початку, у другому – до виконання підпрограми обслуговування переривання відповідного рівня. Під час виконання команди HLT вміст покажчика команд IP автоматично збільшується на одиницю, тому після виконання підпрограми обслуговування переривання процесор перейде до виконання наступною за HLT команди.

Основний спосіб виведення зі стану очікування полягає у подачі сигналу низького рівня (логічного нуля) на вхід МП TEST. Таким чином, час знаходження у стані очікування визначається моментами виконання команди WAIT і появою активного нуля на вході TEST. Керування очікуванням за допомогою цього механізму дозволяє здійснити синхронізацію, тобто узгодити роботу процесора з різними зовнішніми пристроями (наприклад, зі співпроцесорами або з пристроями, що мають меншу швидкодію). У випадку, коли сигнал на вході TEST активний (логічний нуль) у момент виконання команди WAIT, процесор буде знаходитися у стані очікування протягом трьох тактів генератора тактових імпульсів (час виконання команди WAIT). Дії входу TEST схожі на дії входу READY. Відмінність у тому, що READY – вхід для апаратної перевірки готовності пам’яті або пристрою введення/виведення до обміну з МП, а TEST – вхід для програмної перевірки.

Другий спосіб виведення зі стану очікування полягає у поданні запиту переривання на вхід INT. Однак у цьому випадку процесор виходить зі стану очікування тільки тимчасово. За командою WAIT не відбувається автоматичне

нарощування покажчика команд IP, внаслідок чого після виконання відповідної підпрограми обслуговування переривання процесор знову перейде до виконання команди WAIT, тобто перейде у стан очікування. Таким чином, МП i8086, на відміну від i8080, може виконувати підпрограми переривання під час очікування сигналу готовності від зовнішнього пристрою. Важливо зазначити, що у разі поновлення роботи процесора після очікування (за сигналом TEST = 0) зовнішні переривання не будуть обслуговуватися, доки не виконається наступна за WAIT команда.

3.4.1.15. Команди синхронізації зі співпроцесорами

Ці команди використовуються під час проектування на основі МП i8086 багатопроцесорних систем. Хоча система команд МП i8086 досить розвинена, проте в ній відсутні деякі команди, характерні для високопродуктивних систем. Відсутні команди процесора можна реалізувати за допомогою відповідних підпрограм. Однак більш ефективним рішенням є використання співпроцесорів (спеціалізованих процесорів), призначених для розширення функцій основного процесора. Наприклад, відсутні у системі команд операції над числами з плаваючою комою можуть бути виконано, наприклад, за допомогою співпроцесора i8087.

Для організації сумісної роботи основного процесора системи зі співпроцесорами служить команда ESC (№ 32, табл. 3.12). Перший байт команди містить код операції ESC, що дорівнює 11011, і трирозрядне поле xxx; другий байт має структуру постбайта, в якому поле *reg* позначено як *yyy*.

Поле *xxx* вказує номер того співпроцесора багатопроцесорної системи, який повинен виконати відповідну операцію, а поле *yyy* – номер (код) цієї операції. Для організації взаємодії основного процесора зі співпроцесорами останні повинні стежити за появою у потоку команд основного процесора команди ESC, яка вкаже, який співпроцесор і яку операцію повинен виконувати. У загальному випадку поля *xxx* і *yyy* дозволяють задати 64 комбінації 6-розрядних двійкових кодів. Наприклад, можна побудувати систему з одним співпроцесором, який виконує 64 різні операції, або з вісьма співпроцесорами, кожен з яких буде виконувати до 8 операцій і т. ін.

Поля *mod* і *r/m* 2-го байта команди ESC використовуються для задання адреси операнда, який буде брати участь у виконанні команди ESC. За вмістом цих полів основний процесор витягує операнд з пам’яті та виставляє його значення на шину даних як операнд для співпроцесора. Таким чином, основний процесор видає всю необхідну інформацію для роботи відповідного співпроцесора: момент включення в роботу (поява коду операції ESC), номер співпроцесора (поле xxx), код операції (поле yyy) і операнд (видається на шину даних).

У [2] розглянуто приклад використання арифметичного співпроцесора

для перемноження двох чисел з плаваючою комою.

Кожній команді ESC передує команда WAIT, яка синхронізує роботу основного процесора зі співпроцесором. Тут припускається, що сигнал готовності співпроцесора (логічний нуль) подається на вхід TEST.

Є ще одна команда, яка може застосовуватися у разі роботи зі співпроцесором: команда LOCK. Вона є однобайтовим префіксом блокування шин (машинний код 11110000B = F0H), під час виконання якої на однойменному виході МП формується логічний нуль на час виконання наступної команди. Цей сигнал на виході LOCK повідомляє співпроцесорам багатопроцесорної системи, що під час виконання наступної команди забороняється робити запит системної шини.

Проектування багатопроцесорних систем зазвичай пов’язано з розв’язанням питання керування доступом до спільної пам’яті, яка зветься базою даних. Для запобігання конфліктним ситуаціям і помилкам у роботі системи потрібно організувати керування так, щоб у кожен момент часу тільки один процесор мав доступ до спільної бази даних. Для цього необхідно, по-перше, мати ознаку дозволу доступу і, по-друге, виключити можливість аналізу цієї ознаки одночасно декількома процесорами системи. Ознака дозволу доступу має два значення: «логічний нуль» – доступ дозволено і «логічна одиниця» – доступ заборонено. Для зберігання цієї ознаки виділяється деяка комірка пам’яті, яку позначимо як SEMAPHOR. Доки один процесор аналізує вміст комірки SEMAPHOR, робота інших процесорів системи блокується за допомогою префікса LOCK. Цей префікс може передувати будь-якій команді у програмі та спричиняє появу активного сигналу на виході LOCK центрального процесора, який використовується для блокування інших процесорів.

У [2] розглянуто приклад використання префікса LOCK для керування доступом до спільної бази даних.

3.4.1.16. Порожня операція

Окреме місце у системі команд МП, що розглядається, займає однобайтна команда NOP – відсутність операції (табл. 3.12). За цією командою МП не виконує жодних дій. Команда не впливає на прапорці та не має операндів.

Вона використовується для задання часової затримки, тривалістю два такти, або для резервування однієї комірки пам’яті.

Машинний код команди NOP збігається з кодом команди XCHG AX, AX, за якою також не виконується жодних дій.

3.4.2. Команди 8-розрядного мікроконтролера сімейства AVR

3.4.2.1. Загальні відомості про сімейство

AVR-мікроконтролери мають RISC-архітектуру, основною перевагою якої є збільшення швидкодії завдяки скороченню кількості операцій обміну з

пам’яттю програм. Практично всі команди займають одну комірку пам’яті. Виняток становлять команди, в яких одним з операндів є 16-розрядна адреса статичної пам’яті даних (регістр загального призначення, регістр введення/виведення та статичний ОЗП).

Підвищення швидкодії досягнуто не завдяки скороченню кількості команд процесора, а завдяки збільшенню розрядності комірки пам’яті програм до 16. Більшість команд виконується за один машинний цикл (1 такт).

Усі команди AVR-мікроконтролерів можна поділити на декілька груп:

– логічні операції;
– арифметичні операцій та команди зсуву;
– операції з бітами;
– команди пересилання даних;
– команди передачі керування;
– команди керування системою.

Нижче скорочено описано ці команди. Детальніша інформація є у [3].

У табл. 3.16 наведено базовий набір команд, якими можна користуватись у типовому AVR-мікроконтролері, а також основні відомості про команди, такі як мнемонічне позначення команди, її опис, тип, кількість тактів, необхідних для її виконання, а також прапорці регістра SREG, на які впливає ця команда.

У табл. 3.16 використано такі позначення:

– Rd – регістр-приймач результату, $0 \le d \le 31$;
– Rd* – регістр-приймач результату, $16 \le d \le 31$;
– Rdl – R24, R26, R28, R30 для команд ADIW і SBIW;
– Rr – регістр-джерело, $0 \le d \le 31$;
– P – адреса регістра введення/виведення;
– P* – адреса регістра введення/виведення, який адресується побітово (адреси $00…$1F);
– K – символьна або числова константа (8 біт);
– K6 – символьна або числова константа (6 біт);
– k – адресна константа;
– b – номер біта в регістрі (3 біти);
– q – константа (6 біт), може бути константний вираз;
– s – номер біта в регістрі стану (3 біти);
– X, Y, Z – індексні регістри непрямої адресації
(X = R27:R26, Y = R29:R28, Z = R31:R30).

3.4.2.2. Арифметичні операції і команди зсуву

До цієї групи належать команди, що виконують такі операції, як додавання, віднімання, зсув – вправо/вліво та інкремент/декремент.

Таблиця 3.16. **Базовий набір команд AVR-мікроконтролера**

№ з/п	Мнемонічне позначення	Операнди	Опис	Операція	Прапорці	Тип	Кільк. тактів
АРИФМЕТИЧНІ, ЛОГІЧНІ КОМАНДИ ТА КОМАНДИ ЗСУВУ							
1	ADD	Rd, Rr	Додавання без перенесення	Rd ← Rd + Rr	Z, C, N, V, H, S	3	1
2	ADC	Rd, Rr	Додавання з перенесенням	Rd ← Rd + Rr + C	Z, C, N, V, H, S	3	1
3	ADIW	Rdl, K6	Додавання двох байт із константою	Rdh:Rdl ← Rdh:Rdl + K6	Z, C, N, V, S	4	2
4	SUB	Rd, Rr	Віднімання без перенесення	Rd ← Rd – Rr	Z, C, N, V, H, S	3	1
5	SUBI	Rd*, K	Віднімання константи	Rd* ← Rd* – K	Z, C, N, V, H, S	5	1
6	SBC	Rd, Rr	Віднімання з перенесенням	Rd ← Rd – Rr – C	Z, C, N, V, H, S	3	1
7	SBCI	Rd*, K	Віднімання константи з перенесенням	Rd* ← Rd* – K – C	Z, C, N, V, H, S	5	1
8	SBIW	Rdl, K6	Віднімання з двох байт константи	Rdh:Rdl ← Rdh:Rdl – K6	Z, C, N, V, S	4	2
9	AND	Rd, Rr	Логічне І	Rd ← Rd • Rr	Z, N, V, S	3	1
10	ANDI	Rd*, K	Логічне І з константою	Rd* ← Rd* • K	Z, N, V, S	5	1
11	OR	Rd, Rr	Логічне АБО	Rd ← Rd v Rr	Z, N, V, S	3	1
12	ORI	Rd*, K	Логічне АБО з константою	Rd* ← Rd* v K	Z, N, V, S	5	1
13	EOR	Rd, Rr	Логічне виключне АБО	Rd ← Rd ⊕ Rr	Z, N, V, S	3	1
14	COM	Rd	Побітова інверсія	Rd ← $FF – Rd	Z, C, N, V, S	2	1
15	NEG	Rd	Зміна знака (додатковий код)	Rd ← $00 – Rd	Z, C, N, V, H, S	2	1
16	INC	Rd	Інкремент значення регістра	Rd ← Rd + 1	Z, N, V, S	2	1
17	DEC	Rd	Декремент значення регістра	Rd ← Rd – 1	Z, N, V, S	2	1
18	ASR	Rd	Арифметичний зсув вправо	Rd(n) ← Rd(n+1), n = 0...6, C ← Rd(0), Rd(7) ← Rd(7)	Z, C, N, V	2	1
19	LSL	Rd	Логічний/арифметичний зсув вліво	Rd(n + 1) ← Rd(n), Rd(0) ← 0, C ← Rd(7)	Z, C, N, V	2	1
20	LSR	Rd	Логічний зсув вправо	Rd(n) ← Rd(n + 1), Rd(7) ← 0, C ← Rd(0)	Z, C, N, V, S	2	1
21	ROL	Rd	Циклічний зсув вліво через С	Rd(0) ← C, Rd(n + 1) ← Rd(n), C ← Rd(7)	Z, C, N, V, H, S	2	1
22	ROR	Rd	Циклічний зсув вправо через С	Rd(7) ← C, Rd(n)← Rd(n + 1), C ← Rd(0)	Z, C, N, V, S	2	1
23	TST	Rd	Перевірка на нуль чи від'ємне значення	Rd ← Rd • Rd	Z, N, V, S	2	1
24	CLR	Rd	Очищення регістра	Rd ← Rd ⊕ Rd	Z, N, V, S	2	1
25	SWAP	Rd	Обмін тетрадами	Rd(3...0) ← Rd(7...4), Rd(7...4) ← Rd(3...0)	–	2	1
26	SER	Rd	Встановлення регістра	Rd ← $FF	–	2	1
КОМАНДИ ПЕРЕДАЧІ КЕРУВАННЯ							
1	RJMP	k	Відносний безумовний перехід	PC ← PC + k + 1	–	13	2
2	IJMP		Непрямий безумовний перехід	PC ← Z	–	1	2
3	RCALL	k	Відносний безумовний виклик підпрограми	STACK ← PC + 1, PC ← PC + k + 1, SP ← SP - 2	–	13	3
4	ICALL		Непрямий безумовний виклик підпрограми	STACK ← PC + 1, PC ← Z, SP ← SP - 2	–	1	3

Продовження табл. 3.16

№ з/п	Мнемонічне позначення	Операнди	Опис	Операція	Прапорці	Тип	Кільк. тактів
5	RET		Повернення з підпрограми	PC ← STACK, SP ← SP + 2	–	1	4
6	RETI		Повернення з підпрограми обробки переривання	PC ← STACK, SP ← SP + 2, I ← 1	I	1	4
7	CPSE	Rd, Rr	Порівняти, пропустити, якщо рівні	if (Rd = Rr), то PC ← PC + 2/3	–	3	1/2/3
8	CP	Rd, Rr	Порівняти	Rd – Rr	Z, N, V, C, H, S	3	1
9	CPC	Rd, Rr	Порівняти з перенесенням	Rd – Rr – C	Z, N, V, C, H, S	3	1
10	CPI	Rd*, K	Порівняти з константою	Rd* – K	Z, N, V, C, H, S	5	1
11	SBRC	Rr, b	Пропустити, якщо біт у регістрі скинутий	if (Rr(b) = 0), то PC ← PC + 2/3	–	7	1/2/3
12	SBRS	Rr, b	Пропустити, якщо біт у регістрі встановлений	if (Rr(b) = 1), то PC ← PC + 2/3	–	7	1/2/3
13	SBIC	P*, b	Пропустити, якщо біт у порту скинутий	if (P*(b) = 0), то PC ← PC + 2/3	–	9	1/2/3
14	SBIS	P*, b	Пропустити, якщо біт у порту встановлений	if (P*(b) = 1), то PC ← PC + 2/3	–	9	1/2/3
15	BRBS	s, k	Перейти, якщо прапорець у SREG встановлений	if (SREG(s) = 1) then C ← PC + k + 1	–	11	1/2
16	BRBC	s, k	Перейти, якщо прапорець у SREG скинутий	if (SREG(s) = 0) then PC ← PC + k + 1	–	11	1/2
17	BREQ	k	Перейти, якщо дорівнює	if (Z = 1) then PC ← PC + k + 1	–	12	1/2
18	BRCS	k	Перейти, якщо прапорець перенесення встановлений	if (C = 1) then PC ← PC + k + 1	–	12	1/2
19	BRNE	k	Перейти, якщо не дорівнює	if (Z = 0) then PC ← PC + k + 1	–	12	1/2
20	BRCC	k	Перейти, якщо прапорець перенесення скинутий	if (C = 0) then PC ← PC + k + 1	–	12	1/2
21	BRSH	k	Перейти, якщо дорівнює або більше	if (C = 0) then PC ← PC + k + 1	–	12	1/2
22	BRLO	k	Перейти, якщо менше	if (C = 1) then PC ← PC + k + 1	–	12	1/2
23	BRMI	k	Перейти, якщо мінус	if (N = 1) then PC ← PC + k + 1	–	12	1/2
24	BRPL	k	Перейти, якщо плюс	if (N = 0) then PC ← PC + k + 1	–	12	1/2
25	BRGE	k	Перейти, якщо більше або дорівнює (зі знаком)	if (N ⊕ V = 0) then PC ← PC + k + 1	–	12	1/2
26	BRLT	k	Перейти, якщо менше (зі знаком)	if (N ⊕ V = 1) then PC← PC + k + 1	–	12	1/2
27	BRHS	k	Перейти, якщо прапорець половинного перенесення встановлений	if (H = 1) then PC ← PC + k + 1	–	12	1/2
28	BRHC	k	Перейти, якщо прапорець половинного перенесення скинутий	if (H = 0) then PC ← PC + k + 1	–	12	1/2
29	BRTS	k	Перейти, якщо прапорець T встановлений	if (T = 1) then PC ← PC + k + 1	–	12	1/2
30	BRTC	k	Перейти, якщо прапорець T скинутий	if (T = 0) then PC ← PC + k + 1	–	12	1/2
31	BRVS	k	Перейти, якщо прапорець переповнення встановлений	if (V = 1) then PC ← PC + k + 1	–	12	1/2
32	BRVC	k	Перейти, якщо прапорець переповнення скинутий	if (V = 0) then PC ← PC + k + 1	–	12	1/2
33	BRIE	k	Перейти, якщо переривання дозволені	if (I = 1) then PC ← PC + k + 1	–	12	1/2
34	BRID	k	Перейти, якщо переривання заборонені	if (I = 0) then PC ← PC + k + 1	–	12	1/2

Продовження табл. 3.16

№ з/п	Мнемо-нічне позначення	Операнди	Опис	Операція	Пра-порці	Тип	Кіл. тактів
КОМАНДИ ПЕРЕСИЛАННЯ ДАНИХ							
1	MOV	Rd, Rr	Копіювання регістра	Rd ← Rr	–	3	1
2	LDI	Rd*, K	Завантаження константи	Rd* ← K	–	5	1
3	LD	Rd, X	Непряме завантаження	Rd ← (X)	–	2	2
4	LD	Rd, X+	Непряме завантаження з постінкрементом	Rd ← (X), X ← X + 1	–	2	2
5	LD	Rd, – X	Непряме завантаження з переддекрементом	X ← X – 1, Rd ← (X)	–	2	2
6	LD	Rd, Y	Непряме завантаження	Rd ← (Y)	–	2	2
7	LD	Rd, Y+	Непряме завантаження з постінкрементом	Rd ← (Y), Y ← Y + 1	–	2	2
8	LD	Rd, – Y	Непряме завантаження з переддекрементом	Y ← Y – 1, Rd ← (Y)	–	2	2
9	LDD	Rd, Y + q	Непряме завантаження зі зміщенням	Rd ← (Y + q)	–	6	2
10	LD	Rd, Z	Непряме завантаження	Rd ← (Z)	–	2	2
11	LD	Rd, Z+	Непряме завантаження з постінкрементом	Rd ← (Z), Z ← Z + 1	–	2	2
12	LD	Rd, –Z	Непряме завантаження з переддекрементом	Z ← Z – 1, Rd ← (Z)	–	2	2
13	LDD	Rd, Z + q	Непряме завантаження зі зміщенням	Rd ← (Z + q)	–	6	2
14	LDS	Rd, k	Пряме завантаження	Rd ← (k)	–	14	2
15	ST	X, Rr	Непряме збереження	(X) ← Rr	–	2	2
16	ST	X+, Rr	Непряме збереження з постінкрементом	(X) ← Rr, X ← X + 1	–	2	2
17	ST	–X, Rr	Непряме збереження з переддекрементом	X ← X – 1, (X) ← Rr	–	2	2
18	ST	Y, Rr	Непряме збереження	(Y) ← Rr	–	2	2
19	ST	Y+, Rr	Непряме збереження з постінкрементом	(Y) ← Rr, Y ← Y + 1	–	2	2
20	ST	–Y, Rr	Непряме збереження з переддекрементом	Y ← Y – 1, (Y) ← Rr	–	2	2
21	STD	Y + q, Rr	Непряме збереження зі зміщенням	(Y + q) ← Rr	–	6	2
22	ST	Z, Rr	Непряме збереження	(Z) ← Rr	–	2	2
23	ST	Z+, Rr	Непряме збереження з пост/інкрементом	(Z) ← Rr, Z ← Z + 1	–	2	2
24	ST	–Z, Rr	Непряме збереження з перед/декрементом	Z ← Z – 1, (Z) ← Rr	–	2	2
25	STD	Z+q, Rr	Непряме збереження зі зміщенням	(Z + q) ← Rr	–	6	2
26	STS	k, Rr	Пряме збереження	(k) ← Rr	–	14	2
27	LPM*		Завантаження з програмної пам’яті	R0 ← (Z)	–	1	3
28	IN	Rd, P	Читання порту	Rd ← P	–	8	1
29	OUT	P, Rr	Запис у порт	P ← Rr	–	8	1
30	PUSH	Rr	Занесення регістра у стек	STACK ← Rr; SP ← SP – 1	–	2	2
31	POP	Rd	Витягнення регістра зі стека	SP ← SP + 1,Rd ← STACK	–	2	2
КОМАНДИ РОБОТИ З БІТАМИ							
1	SBR	Rd*, K	Встановити біт (біти) у регістрі	Rd* ← Rd* v K	Z, N, V, S	5	1
2	CBR	Rd*, K	Скинути біт (біти) у регістрі	Rd* ← Rd* • ($FF – K)	Z, N, V, S	5	1
3	SBI	P*, b	Встановити біт у регістрі введеня/ виведення	I/O(P*, b) ← 1	–	9	2
4	CBI	P*, b	Скинути біт у регістрі введеня/ виведення	I/O(P*, b) ← 0	–	9	2
5	BSET	S	Встановити вказаний розряд регістра SREG	SREG(s) ← 1	SREG (s)	10	1
6	BCLR	S	Скинути заданий розряд регістра SREG	SREG(s) ← 0	SREG (s)	10	1
7	BLD	Rd, b	Завантажити біт з T у регістр	Rd(b) ← T	–	7	1
8	BST	Rr, b	Зберегти біт з регістра у T	T ← Rr(b)	T	7	1
9	SEC		Встановити прапорець перенесення	C ← 1	C	1	1
10	CLC		Скинути прапорець перенесення	C ← 0	C	1	1
11	SEN		Встановити прапорець від’ємного числа	N ← 1	N	1	1
12	CLN		Скинути прапорець від’ємного числа	N ← 0	N	1	1

Закінчення табл. 3.16

№ з/п	Мнемонічне позначення	Операнди	Опис	Операція	Прапорці	Тип	Кільк. тактів
13	SEZ		Встановити прапорець нуля	Z←1	Z	1	1
14	CLZ		Скинути прапорець нуля	Z← 0	Z	1	1
15	SEI		Встановити прапорець переривань	I←1	I	1	1
16	CLI		Скинути прапорець переривань	I←0	I	1	1
17	SES		Встановити прапорець знака	S←1	S	1	1
18	CLS		Скинути прапорець знака	S←0	S	1	1
19	SEV		Встановити прапорець переповнення	V←1	V	1	1
20	CLV		Скинути прапорець переповнення	V←0	V	1	1
21	SET		Встановити прапорець T	T←1	T	1	1
22	CLT		Скинути прапорець T	T←0	T	1	1
23	SEH		Встановити прапорець половинного перенесення	H←1	H	1	1
24	CLH		Очистити прапорець половинного перенесення	H←0	H	1	1
КОМАНДИ КЕРУВАННЯ МК							
1	NOP		Порожня операція	__	–	1	1
2	SLEEP		Переведення у режим сну	__	–	1	3
3	WDR		Скидання вартового таймера	__	–	1	1

Усі операції виконуються тільки над регістрами загального призначення. Операнди можуть бути знаковими/беззнаковими числами, а також подані у додатковому коді.

Усі команди цієї групи виконуються за один машинний цикл, за винятком команд, що оперують двобайтовими значеннями, що виконуються за два цикли.

3.4.2.3. Логічні операції

Ці команди дозволяють виконувати стандартні логічні операції над байтами: «логічне множення» – І, «логічне додавання» – АБО, операцію «виключне АБО», а також обчислення зворотного і додаткового кодів числа. До цієї групи належать також команди очищення/встановлення регістрів і команду перестановки тетрад. Усі логічні операції виконуються за один машинний цикл над регістрами загального призначення, а результат зберігається в одному з регістрів загального призначення.

3.4.2.4. Команди передачі керування

У цю групу входять команди переходу, виклику підпрограм, повернення з них і команди типу «перевірка/пропуск», що пропускають наступну за ними команду під час виконання деякої умови. Також до цієї групи належать команди порівняння, що формують прапорці регістра SREG, які призначено переважно

для роботи разом з командами умовного переходу. У командах цього типу виконується перевірка умови, результат якої впливає на виконання наступної команди. Якщо умова істинна, наступна команда ігнорується.

Наприклад, команда SBRS Rd, b перевіряє розряд b регістра Rd й ігнорує (пропускає) наступну команду, якщо цей розряд дорівнює одиниці. Перехід до наступної інструкції виконується збільшенням лічильника команд на одиницю, а пропуск команди викликає завантаження нового значення в лічильник команд. Отже, коли умова, що перевіряється, істинна, у конвеєрі виникає затримка (див. підрозд. 3.5). Тривалість затримки залежить від довжини команди, що пропускається, і становить від одного до двох машинних циклів.

Команди передачі керування порушують нормальне (лінійне) виконання основної програми. Щоразу, коли виконується команда з цієї групи (окрім команд порівняння), нормальне функціонування конвеєра порушується.

Перед завантаженням у конвеєр нової адреси він зупиняється й очищується послідовність виконуваних команд. Реініціалізація конвеєра призводить до того, що такі команди виконуються протягом декількох машинних циклів [3].

У системі команд МК є команди як безумовного, так і умовного переходів. Команди непрямого – JMP і відносного – RJMP безумовного переходу є найпростішими в цій групі. Їх функція полягає тільки у записі нової адреси в лічильник команд.

Під час виконання команди RJMP змінюється вміст лічильника команд за допомогою додавання до нього або віднімання з нього деякого значення, яке є 12-розрядним числом зі знаком. Ця команда має обмеження за областю дії, оскільки максимальна величина переходу відносно адреси наступної команди становить від –2048 до +2047 слів (приблизно ±4 Кбайт).

У програмах замість констант використовуються мітки. Компілятор (Асемблер) віднімає від адреси мітки адреси наступної команди і підставляє отримане значення у відповідне місце машинного коду команди.

Нижче наведено приклад, який ілюструє сказане:

```
    cpi r16, $42; порівння регістра r16 з числом $42;
brne error; перехід на мітку error, якщо r16 ≠ $42;
 rjmp ok ; безумовний перехід на мітку ok, якщо r16 = $42;
error:
   . . .
ok:  nop; місце переходу за командою rjmp.
```

Оскільки команда відносного переходу змінює вміст лічильника команд, вона виконується за два машинних цикли.

У результаті виконання команди непрямого переходу IJMP програма продовжує виконуватися з адреси, що міститься в індексному регістрі Z. Таким чином, дія команди зводиться до завантаження вмісту індексного регістра у лічильник команд.

На відміну від команди відносного переходу ця команда має менше обмежень за областю дії. Насправді, оскільки індексний регістр Z – 16-розрядний, максимально можлива величина переходу становить: 64 Кслів (128 Кбайт). Як і команда відносного переходу, команда непрямого переходу виконується за два машинних цикли.

3.4.2.5. Команди умовного переходу

У цих командах виконується перевірка умови, результат якої впливає на стан лічильника команд. Якщо умова істинна, відбувається перехід за вказаною адресою, якщо ж умова хибна, виконується наступна команда.

Команди умовного переходу мають обмеження за областю дії. Нове значення лічильника команд обчислюється додаванням до нього або відніманням з нього деякого зміщення. Оскільки під значення зміщення в слові команди виділяється лише 7 біт, максимальна величина переходу відносно адреси наступної команди становить від –64 до +63 слів.

Оскільки перехід за вказаною адресою здійснюється завантаженням нового значення в лічильник команд, то у випадку істинності умови, що перевіряється, у конвеєрі виникає затримка тривалістю в один машинний цикл.

Усі команди умовного переходу можна розділити на дві підгрупи. Перша підгрупа – команди умовного переходу загального призначення. У цю підгрупу входять дві команди BRBS s, k і BRBC s, k, в яких явно вказується номер прапорця регістра SREG, який перевіряється. Відповідно, перехід здійснюється під час SREG.s = 0 (BRBC) або SREG.s = 1 (BRBS).

Іншу підгрупу становлять 18 команд, кожна з яких виконує перехід за якою-небудь конкретною умовою: «дорівнює», «більше або дорівнює», «було перенесення» і т. ін. Одні з цих команд використовуються після порівняння беззнакових чисел, інші – після порівняння чисел зі знаком.

Можливі умови, що перевіряються, а також відповідні їм команди умовного переходу наведено в табл. 3.17.

Команди, які наведено в табл. 3.17, є тільки еквівалентними мнемонічними позначеннями команд BRBS s, k та BRBC s, k з визначеними значеннями операнда – s. Наприклад, команда BREQ k має, такий самий код операції, що і команда BRBS 1, k, а команда BRGE k – такий самий, що і BRBC 4, k. За останньою командою відбувається перехід за значенням k, якщо

прапорець S = 1. Значення S = (N + V), де N – прапорець від'ємного значення результату, а V – прапорець переповнення додаткового коду.

Таблиця 3.17. **Зведена таблиця команд умовного переходу**

Перевірка	Логіч. умова	Команда	Зворотна перевірка	Логіч. умова	Команда	Тип даних
Rd>Rr	(N⊕V)=0	BRLT*	Rd≤Rr	(N⊕V)=1	BRGE*	Зі знаком
Rd≥Rr	(N⊕V)=0	BRGE	Rd<Rr	(N⊕V)=1	BRLT	Зі знаком
Rd=Rr	Z=1	BREQ	Rd≠Rr	Z=0	BRNE	Зі знаком
Rd≤Rr	(N⊕V)=1	BRGE*	Rd>Rr	(N⊕V)=0	BRLT*	Зі знаком
Rd<Rr	(N⊕V)=1	BRLT	Rd≥Rr	(N⊕V)=0	BRGE	Зі знаком
Rd>Rr	C=0	BRLO*	Rd≤Rr	C=1	BRSH*	Без знака
Rd≥Rr	C=0	BRSH/BRCC	Rd<Rr	C=1	BRLO/BRCS	Без знака
Rd=Rr	Z=1	BREQ	Rd≠Rr	Z=0	BRNE	Без знака
Rd≤Rr	C=1	BRSH*	Rd>Rr	C=0	BRLO*	Без знака
Rd<Rr	C=1	BRLO/BRCS	Rd≥Rr	C=0	BRSH/BRCC	Без знака
«Перенесення»	C=1	BRCS	«Немає перенесення»	C=0	BRCC	—
«Менше за нуль»	N=1	BRMI	«Більше за нуль»	N=0	BRPL	—
«Переповнення»	V=1	BRVS	«Немає переповнення»	V=0	BRVC	—
«Нуль»	Z=1	BREQ	«Не нуль»	Z=0	BRNE	—
«Половинне перенесення»	H=1	BRHS	«Немає половинного перенесення»	H=0	BRHC	—

* Оскільки у МК відсутні команди переходів за умовами: більше та менше, або дорівнює, то для переходу за цими умовами операнди попередньої команди порівняння мають бути записані в зворотному порядку, тобто замість CP Rd, Rr –> CP Rr, Rd.

3.4.2.6. Команди виклику підпрограм

Для виклику підпрограм у базовому наборі команд є дві команди: команда відносного виклику – RCALL і команда непрямого виклику – ICALL.

Враховуючи деякі відмінності, які описано нижче, команда RCALL працює так само, як і команда відносного безумовного переходу RJMP.

Команда RCALL зберігає у стеку значення лічильника команд (адресу наступної команди). Потім вміст лічильника команд збільшується або зменшується на деяке значення, що є 12-розрядним числом зі знаком. Максимальна величина переходу відносно адреси наступної команди становить від –2048 до +2047 слів (приблизно ±4 Кбайт).

У програмах у команді RCALL, як і у випадку команди RJMP, використовуються мітки. Асемблер сам обчислює величину зміщення відносно

адреси наступної команди за допомогою віднімання від адреси мітки адреси наступної команди і підставляє це значення в машинний код команди.

Команда відносного виклику підпрограм виконується за три машинних цикли, два з яких витрачаються на збереження у стеку двох байт лічильника команд (адреси наступної команди).

Враховуючи деякі відмінності, які описано нижче, команда ICALL працює так само, як і команда непрямого безумовного переходу IJMP.

Команда ICALL зберігає у стеку значення лічильника команд (адресу наступної команди). Потім у лічильник команд завантажується вміст індексного регістра Z. Оскільки індексний регістр – 16-розрядний, максимально можлива величина переходу становить 64 Кслів (128 Кбайт). Як і команда RCALL, команда непрямого виклику підпрограм виконується за три машинних цикли.

Підпрограма повинна закінчуватися командою повернення RET, як показано в прикладі:

```
rcall   sp_test    ;  виклик підпрограми  sp_test
...          ;  текст основної програми
sp_test      ;  мітка підпрограми
push    r2       ;  збереження r2 у стеку
...          ;  виконання підпрограми
pop     r2   ;  відновити r2 зі стека
ret          ;  повернення з підпрограми
```

У наведеному вище прикладі команда ret замінює адресу, що міститься в лічильнику команд, адресою команди, наступної за командою rcall.

3.4.2.7. Команди повернення з підпрограм

У кінці кожної підпрограми обов’язково повинна міститися команда повернення з неї. У системі команд AVR-мікроконтролерів є дві таких команди.

Для повернення з підпрограми, що викликається командами RCALL і ICALL, використовується команда RET. Для повернення з підпрограми обробки переривання використовується команда RETI.

Обидві команди відновлюють зі стека вміст лічильника команд, який зберігається там перед переходом до підпрограми. Команда повернення з підпрограми RETI додатково встановлює в одиницю прапорець глобального дозволу переривань I регістра SREG, що скидається апаратно у разі виникнення переривання.

На виконання кожної з команд повернення з підпрограми потрібно чотири машинних цикли.

3.4.2.8. Команди пересилання даних

Команди цієї групи призначено для пересилання вмісту комірок, що перебувають у просторі адрес статичної пам’яті даних – регістрі загального призначення, регістрі введення/виведення та статичному ОЗП. Поділ простору адрес на три частини обумовлює різноманітність команд цієї групи. Пересилання даних може виконуватись у таких напрямках:

– регістр загального призначення <=> регістр загального призначення;
– регістр загального призначення <=> регістр введення/виведення;
– регістр загального призначення <=> статична пам’ять даних.

До цієї групи також належать команди PUSH і POP, які дозволяють зберігати у стеку і відновлювати зі стека вміст регістра загального призначення.

На виконання команд пересилання потрібно від одного до трьох машинних циклів залежно від типу команди.

3.4.2.9. Команди операцій з бітами

До цієї групи належать команди, що виконують встановлення або скидання заданого розряду регістра загального призначення або регістра введення/виведення. Є також команди для зміни розрядів регістра стану SREG, оскільки перевірка стану розрядів саме цього регістра виконується найчастіше. Умовно до цієї групи можуть належати також дві команди передачі керування типу «перевірка/пропуск», що пропускають наступну команду залежно від стану розряду регістра загального призначення або регістра введення/виведення.

Усі задіяні розряди регістра введення/виведення мають свої символічні імена. Визначення цих імен описано у тому ж файлі, що й визначення символічних імен для адрес регістрів [3; 4]. Таким чином, після додавання у програму зазначеного файлу в командах замість числових значень номерів розрядів можна буде вказувати їхні символічні імена.

Усі команди цієї групи виконуються за один машинний цикл, за винятком випадків, коли в результаті перевірки відбувається пропуск наступної команди. У цьому разі команда виконується за два або три машинних цикли залежно від довжини команди, що пропускається.

3.4.2.10. Команди керування мікроконтролером

До цієї групи належать 3 команди:

– NOP – порожня операція;
– SLEEP – переведення МК у режим зниженого енергоспоживання;
– WDR – скидання вартового таймера.

Команди NOP і WDR виконуються за один машинний цикл, а команда SLEEP – за три машинних цикли.

Вище скорочено було розглянуто 118 базових команд. Реальна кількість команд більша, оскільки за деякими номерами наведено опис команд, які виконуються за схожими алгоритмами, але відрізняються способами адресації операндів. Детальний опис базових команд наведено у [3; 4].

3.4.2.11. Нові команди

Архітектура AVR-мікроконтролерів постійно оновлювалася. Це оновлення стосується як структури МК, так і його системи команд. Кожне сімейство послідовно успадковувало набір команд попереднього сімейства. Однак є деякі винятки.

Нижче розглянуто нові команди, які останнім часом з'явилися у AVR-мікроконтролерах: Mega та XMega [3].

JMP

Код операції:

15														1	0
1	0	0	1	0	1	0	k	k	k	k	k	1	1	0	k
31														17	16
k	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k

Алгоритм виконання:

PC ← k.

Опис: для МК з 22-розрядним програмним лічильником (регістром стану), максимальна ємність пам'яті програм становить 2^{22} = 4 Мслів (8 Мбайт), наприклад у МК AT90SC6464C-USB, а для МК з 16-розрядним регістром стану, відповідно 2^{16} = 64 Кслів (128 Кбайт).

Довжина команди: 2 слова (4 байти).

CALL

Код операції:

15														1	0
1	0	0	1	0	1	0	k	k	k	k	k	1	1	1	k
31															16
k	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k

Алгоритм виконання:

STACK ← PC + 2 SP ← SP–2 PC ← k, якщо k = 16 біт	Безумовний прямий виклик підпрограми із пам'яті програм ємністю 2^{16} = 64 Кслів (128 Кбайт)
STACK ← PC + 2 SP ← SP–3 PC ← k, якщо k = 22 біт	Безумовний прямий виклик підпрограми із пам'яті програм ємністю 2^{22} = 4 Мслів (8 Мбайт)

Довжина команди: 2 слова (4 байти).

MULS

Код операції:

15							8	7							0
0	0	0	0	0	0	1	0	d	d	d	d	r	r	r	r

Алгоритм виконання:

R1:R0 ← RdxRr

PC ← PC + 1.

Операнди: Rd, Rr ($16 \le d \le 31$; $16 \le r \le 31$) – знакові величини.

Довжина команди: 1 слово (2 байти).

MULSU

Код операції:

15							8	7							0
1	0	0	1	0	0	1	1	0	d	d	d	0	r	r	r

Алгоритм виконання:

R1:R0 ← RdxRr

PC ← PC + 1.

Операнди: Rd, Rr ($16 \le d \le 23$; $16 \le r \le 23$): Rr – беззнакове число, Rd – знакове число.

Довжина команди: 1 слово (2 байти).

FMUL

Код операції:

15							8	7							0
0	0	0	0	0	0	1	1	0	d	d	d	0	r	r	r

Алгоритм виконання:

R1:R0 ← RdxRr

PC ← PC + 1.

Опис: числа подаються у вигляді N.Q, де N – значення числа до крапки у двійковому вигляді; Q – значення числа після крапки у двійковому вигляді. Таким чином операнди подаються в вигляді (N1.Q1) і (N2.Q2). Беззнакові 8-розрядні дробові числа використовують формат, в якому числа можуть перебувати у діапазоні: [0, 2]. Біти 6…0 являють собою дробову частину, а сьомий біт – цілу частину (0 або 1), тобто 1.7 формат. Результат множення (формат результату): 2.14 зсувається вліво на один розряд для приведення до формату – 1.15 та записується у пару регістрів R1:R0.

Операнди: Rd, Rr (16 ≤ d≤ 23; 16 ≤ r ≤ 23) – беззнакові дробові величини.

Довжина команди: 1 слово (2 байти).

FMULS

Код операції:

15							8	7							0
0	0	0	0	0	0	1	1	1	d	d	d	0	r	r	r

Алгоритм виконання:

R1:R0 ← RdxRr

PC ← PC + 1.

Опис: числа подаються у вигляді N.Q, де N – значення числа до крапки у двійковому вигляді; Q – значення числа після крапки у двійковому вигляді. Таким чином операнди подаються у вигляді (N1.Q1) і (N2.Q2). Дробові числа зі знаком подібно до цілих чисел зі знаком використовують формат двійкового доповнення, що дозволяє подавати перші у діапазоні: [–1, 1]. Якщо, наприклад, ціле число без знака має двійковий код: 10110010, то його десятковий еквівалент становить: 128 + 32 + 16 + 2 = 178. Відповідно, десятковий еквівалент цього числа як знакового цілого становитиме: 178 – 256 = –78. Якщо наведене вище двійкове число подати у форматі беззнакового дробового, то отримаємо:

1 + 0,25 + 0,125 + 0,015625 = 1,390625.

Аналогічно наведеному правилу обчислення цілих чисел зі знаком можна отримати десятковий еквівалент двійкового числа 10110010, якщо вважати, що воно відповідає дробовому числу зі знаком: 1,390625 – 2 = – 0,609375.

Результат множення (формат результату): 2.14 зсувається вліво на один розряд для приведення до формату: 1.15 та записується у пару регістрів R1:R0.

Операнди: Rd, Rr (16 ≤ d ≤ 23; 16 ≤ r ≤ 23) – дробові величини зі знаком.

Довжина команди: 1 слово (2 байти).

FMULSU

Код операції:

15							8	7							0
0	0	0	0	0	0	1	1	1	d	d	d	1	r	r	r

Алгоритм виконання:

R1:R0 ← RdxRr

PC ← PC + 1.

Опис: числа подаються у вигляді N.Q, де N – значення числа до крапки у двійковому вигляді, Q – значення числа після крапки у двійковому вигляді. Таким чином операнди подаються у вигляді: N1.Q1 і N2.Q2 (дивись опис команди FMULS).

Операнди: Rd, Rr ($16 \le d \le 23$; $16 \le r \le 23$) – обидві величини дробові, одна з яких у Rr – беззнакова, друга величина у Rd – знакова.

Довжина команди: 1 слово (2 байти).

MOVW

Код операції:

15							8	7							0
0	0	0	0	0	0	0	1	d	d	d	d	r	r	r	r

Алгоритм виконання:

Rd + 1:Rd ← Rr + 1:Rr

PC ← PC + 1.

Опис: команда копіює вміст однієї пари регістрів загального призначення в іншу пару регістрів загального призначення. Пара регістрів джерела (Rr + 1:Rr) залишається незмінною. Пара регістрів приймача (Rd + 1:Rd) завантажується копією регістрів Rr + 1:Rr. Номери регістрів Rr, Rd мають бути парними.

Операнди: Rd, Rr ($0 \le d \le 30$; $0 \le r \le 30$).

Довжина команди: 1 слово (2 байти).

LPM Rd, Z

Код операції:

15							8	7							0
1	0	0	1	0	0	0	d	d	d	d	d	0	1	0	0

Алгоритм виконання:

Rd ← ПП (Z)

PC ← PC + 1.

Операнди: Rd ($0 \le d \le 31$), ПП (Z) – комірка пам’яті програм, адреса якої міститься в індексному регістрі Z.

Довжина команди: 1 слово (2 байти).

LPM Rd, Z+

Код операції:

15							8	7							0
1	0	0	1	0	0	0	d	d	d	d	d	0	1	0	1

Алгоритм виконання:

Rd ←ПП (Z), Z ← Z + 1

PC ← PC + 1.

Операнди: Rd ($0 \le d \le 31$), ПП (Z) – комірка пам’яті програм, адреса якої міститься в індексному регістрі Z.

Довжина команди: 1 слово (2 байти).

SPM

Код операції:

15							8	7							0
1	0	0	1	0	1	0	1	1	1	1	0	1	0	0	0

Опис: ця команда використовується для самопрограмування МК, який має відповідний блок. Команда зберігається, і відповідно виконується, у спеціальній області пам’яті, яка називається завантажувачем (див. пп. 10.4.2).

Більш детальний опис цієї команди наведено у [3].

BREAK

Код операції:

15							8	7							0
1	0	0	1	0	1	0	1	1	0	0	1	1	0	0	0

Алгоритм виконання:

PC ← PC + 1.

Опис: ця інструкція використовується налагоджувачем, який вбудовано у МК, та зазвичай у робочих програмах не використовується. Після виконання команди BREAK процесор зупиняється, що дає можливість налагоджувачеві працювати із внутрішніми ресурсами.

Операнди: немає.

Довжина команди: 1 слово (2 байти).

EIJMP

Код операції:

15							8	7							0
1	0	0	1	0	1	0	0	0	0	0	1	1	0	0	1

Алгоритм виконання:

PC(15:0) ← Z(15:0)

PC(21:16) ← EIND (6 біт),

де EIND – ім’я додаткового регістра керування.

Довжина команди: 1 слово (2 байти).

EICALL

Код операції:

15							8	7							0
1	0	0	1	0	1	0	1	0	0	0	1	1	0	0	1

Алгоритм виконання:

STACK ← PC + 1

SP ← SP-3

PC(15:0) ← Z(15:0)

PC(21:16) ← EIND (6 біт),

де EIND – ім’я додаткового регістра керування.

Довжина команди: 1 слово (2 байти).

ELPM

Код операції:

15							8	7							0
1	0	0	1	0	1	0	1	1	1	0	1	1	0	0	0

Алгоритм виконання:

R0←ПП (RAMPZ:Z),

де RAMPZ – ім’я додаткового регістра керування;

PC ← PC + 1.

Операнди: R0 – регістр загального призначення; (RAMPZ:Z) – комірка пам’яті програм, адреса якої міститься у двох регістрах: RAMPZ (6 біт) та Z – 16 біт. Довжина команди: 1 слово (2 байти).

ELPM Rd, Z

Код операції:

15							8	7							0
1	0	0	1	0	0	0	d	d	d	d	d	0	1	1	0

Алгоритм виконання:

Rd ← ПП (RAMPZ:Z),

де RAMPZ – ім'я додаткового регістра керування;

PC ← PC + 1.

Операнди: Rd ($0 \le d \le 31$) – регістри загального призначення;

ПП (RAMPZ:Z) – комірка пам'яті програм, адреса якої міститься у двох регістрах: RAMPZ (6 біт) та Z – 16 біт.

Довжина команди: 1 слово (2 байти).

ELPM Rd, Z+

Код операції:

15							8	7							0
1	0	0	1	0	0	0	d	d	d	d	d	0	1	1	1

Алгоритм виконання:

Rd ←ПП (RAMPZ:Z), RAMPZ:Z ← RAMPZ:Z + 1,

де RAMPZ – ім'я додаткового регістра керування;

PC ← PC + 1.

Операнди: Rd ($0 \le d \le 31$) – регістри загального призначення; ПП (RAMPZ:Z) – комірка пам'яті програм, адреса якої міститься у двох регістрах: RAMPZ (6 біт) та Z – 16 біт.

Довжина команди: 1 слово (2 байти).

DES

Алгоритм виконання:

R15:R0 ← ENCRYPT(R15:R0, K), якщо прапорець H = 0,

R15:R0 ← DECRYPT(R15:R0, K), якщо прапорець H = 1.

Опис: команда DES використовується в сучасних МК сімейства XMega для шифрування інформації. Більш детальну інформацію про команду та метод кодування наведено у додатковій літературі [3].

LAC

Код операції:

15							8	7							0
1	0	0	1	0	0	1	d	d	d	d	d	0	1	1	0

Алгоритм виконання:

Temp ← Rd;
Rd ← (Z);
(Z) ← ($FF – Temp) ·(Z);
PC ← PC + 1.

Опис: команда LAC записує Rd у тимчасовий регістр Temp, пересилає (Z) в Rd, інвертоване значення Temp множить на (Z) та результат записує в (Z). Таким чином до виконання команди в Rd знаходиться маска, за якою необхідно скинути у нуль відповідні біти (Z), а після виконання команди результат записується у (Z) з обнуленими бітами на тих місцях, де у масці знаходилась одиниця. Під час виконання команди початкове значення (Z) записується у Rd.

Операнди: Rd ($0 \le d \le 31$).

Довжина команди: 1 слово (2 байти).

LAS

Код операції:

15							8	7							0
1	0	0	1	0	0	1	d	d	d	d	d	0	1	0	1

Алгоритм виконання:

Temp ← Rd;
Rd ← (Z);
(Z)← Temp v (Z);
PC ← PC + 1.

Опис: команда LAS записує Rd у тимчасовий регістр Temp, пересилає (Z) у Rd, логічно підсумовує Temp і (Z) та результат записує в (Z). Таким чином до виконання команди в Rd знаходиться маска, за якою необхідно встановити в одиницю відповідні біти (Z), а після виконання команди результат записується у (Z) з встановленими бітами на тих місцях, де у масці знаходилась одиниця. Під час виконання команди початкове значення (Z) записується у Rd.

Операнди: Rd ($0 \le d \le 31$).

Довжина команди: 1 слово (2 байти).

LAT

Код операції:

15							8	7							0
1	0	0	1	0	0	1	d	d	d	d	d	0	1	1	1

Алгоритм виконання:

Temp ← Rd;
Rd ← (Z);
(Z)← Temp xor (Z);
PC ← PC + 1.

Опис: команда LAT записує Rd у тимчасовий регістр Temp, пересилає (Z) у Rd, виконує логічне «Виключне АБО» (XOR) між Temp і (Z) та результат записує в (Z). Таким чином до виконання команди в Rd знаходиться маска, за якою необхідно проінвертувати відповідні біти (Z), а після виконання команди результат записується у (Z) з проінвертованими бітами на тих місцях, де у масці знаходилась одиниця. Під час виконання команди початкове значення (Z) записується у Rd.

Операнди: Rd ($0 \le d \le 31$).

Довжина команди: 1 слово (2 байти).

XCH

Код операції:

15							8	7							0
1	0	0	1	0	0	0	d	d	d	d	d	0	1	0	0

Алгоритм виконання:

Temp ← Rd;
Rd ← (Z);
(Z) ← Temp;
PC ←PC + 1.

Опис: команда XCH міняє місцями значення, що знаходиться за адресою, вказаною в Z, зі значенням Rd.

Операнди: Rd ($0 \le d \le 31$).

Довжина команди: 1 слово (2 байти).

3.5. Функціонування конвеєра

Для підвищення швидкодії AVR-мікроконтролерів під час виконання програми використовується дворівневий конвеєр (рис. 3.36).

Під час першого машинного циклу відбувається вибірка команди з пам’яті програм та її декодування.

У наступному циклі ця команда виконується і паралельно відбувається вибірка і декодування другої команди і так далі. Фактичний час виконання більшості команд дорівнює одному машинному циклу, що дозволяє досягати продуктивності до 1 MIPS (Million Instructions Per Second) на один мегагерц тактової частоти.

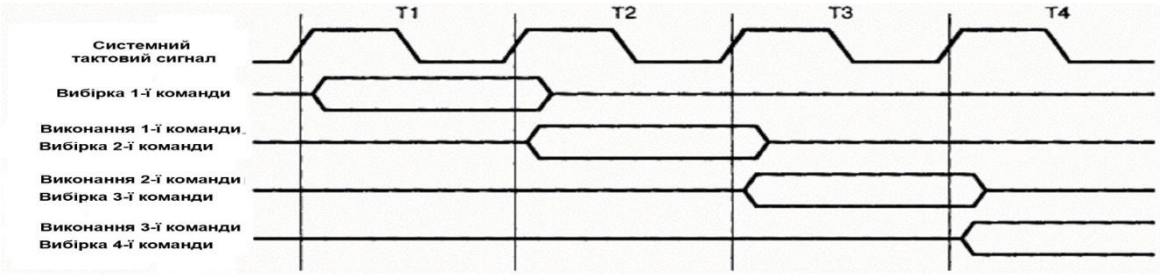


Рис. 3.36. Послідовність виконання команд у конвеєрі

Завдяки підключенню арифметико-логічного пристрою безпосередньо до регістрового файлу (регістра загального призначення) він виконує одну команду – читання вмісту двох регістрів, виконання операції і запис результату в регістр-приймач за один такт (рис. 3.37).

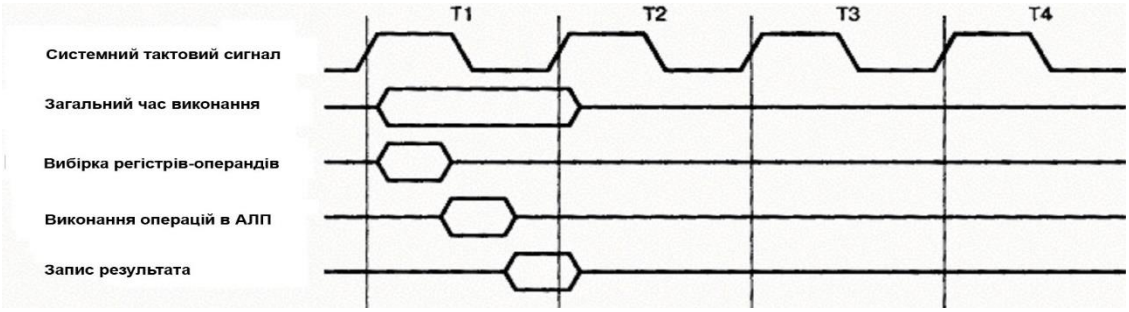


Рис. 3.37. Функціонування арифметико-логічного пристрою

Звернення до внутрішнього статичного ОЗП виконується за два такти [3].

Вище було описано послідовність виконання команд програми в ідеальному випадку. Однак у дійсності дуже часто відбувається порушення нормального порядку функціонування конвеєра. Прикладом команд, що викликають таке порушення, є команди умовного переходу, а також команди типу «перевірка/пропуск» – пропускають наступну команду, якщо результат перевірки позитивний. Якщо умова, що перевіряється командою умовного переходу, істинна, виконання програми повинно продовжуватись з деякої іншої адреси. Оскільки в конвеєрі уже відбулася вибірка команди, розміщеної після команди переходу, час виконання команди переходу збільшується на один машинний цикл, під час якого відбувається вибірка команди, розміщеної за адресою переходу.

Під час виконання команд типу «перевірка/пропуск», наступна команда не виконується у разі істинності умови, яка перевіряється. Однак вибірка команди, що пропускається, уже відбулася. Внаслідок того, що команда не виконується, у конвеєрі утвориться «дірка», що полягає в пропуску одного або двох (залежно від команди, що пропускається) машинних циклів. Відповідно, команди типу «перевірка/пропуск» виконуються за один машинний цикл, якщо результат перевірки умови негативний, і за два або три цикли, якщо він позитивний.

Команди безумовного переходу – JMP, RJMP і IJMP, команди виклику підпрограм – CALL, RCALL і ICALL і команди повернення з підпрограм – RET і RETI також змінюють вміст лічильника команд (англ. PC), що викликає перехід у пам'яті програм. У результаті виконання цих команд відбувається «розрив» у роботі конвеєра, внаслідок чого виникає затримка виконання програми на декілька машинних циклів. Залежно від команди тривалість затримки становить від двох до чотирьох машинних циклів.

З тієї самої причини порушення нормального функціонування конвеєра відбувається і під час виникнення переривання. Максимальна затримка при цьому становить чотири машинних цикли.

Контрольні запитання та завдання

1. Назвіть основні етапи створення керувальної програми.
2. Дайте визначення підпрограмі.
3. Назвіть регістри загального призначення МП i8080.
4. Що являє собою регістр прапорців?
5. Назвіть регістри загального призначення МП i8086.
6. Який регістр зберігає адресу порту у командах введення/виведення з непрямою адресацією?
7. Назвіть індексні регістри МП i8086.
8. Назвіть сегментні регістри МП i8086.
9. Для чого використовується регістр SP?
10. Що показує прапорець OF?
11. Який прапорець потрібно встановити в одиницю, щоб перевести МП у покроковий режим роботи?
12. Яку функцію виконує покажчик команд IP?
13. Що називається логічною базовою адресою сегмента?
14. Як із пари: початок сегмента – зміщення у сегменті утворити фізичну адресу комірки пам'яті?
15. Дайте визначення мові Асемблера.
16. Назвіть та дайте визначення основних керувальних програм, які використовуються у разі створення програм мовою Асемблера.
17. Поясніть такі терміни: мнемоніка команди та мнемокод; код операції команди; операнди; коментар.
18. Наведіть та поясніть формати команд 8-розрядних МП та МК і 16-розрядного МП.
19. Наведіть та поясніть формати даних 8-розрядних МП та МК і 16-розрядного МП.

20. Поясніть, що таке прапорці та яку роль вони відіграють у роботі МПС.
21. Як обчислюється час виконання окремих команд у МК та МП?
22. Як виконується перепризначення сегментів у МП i8086?
23. Що таке підпрограми і як вони використовуються під час створення асемблерних програм? Наведіть приклад.
24. Назвіть та опишіть логічні команди МП i8086.
25. Назвіть та опишіть логічні команди AVR-мікроконтролерів.
26. Поясніть використання логічних команд для: встановлення в одиницю, скидання в нуль, інвертування потрібних бітів операнда-джерела, при цьому інші біти не повинні змінюватись.
27. Назвіть та опишіть команди обробки рядків МП i8086.
28. Назвіть та поясніть команди переходів МП i8086.
29. Назвіть та поясніть команди переходів AVR-мікроконтролерів.
30. Назвіть та поясніть команди організації циклів МП i8086.
31. Назвіть та поясніть команди виклику та повернення з підпрограм МП i8086.
32. Назвіть та поясніть команди виклику та повернення з підпрограм AVR-мікроконтролерів.
33. Назвіть та опишіть команди роботи з бітами AVR-мікроконтролерів.
34. Які чотири основні булеві функції реалізуються за допомогою логічних команд?
35. Опишіть особливості виконання зсувів: логічного, циклічного, арифметичного.
36. Яку дію виконує команда CMPS?
37. В якому випадку використовуються четвертий і п’ятий формати JMP у МП i8086?
38. Які мнемоніки включають команди програмних переривань МП i8086?
39. Що називається виконавчою (ефективною) адресою у МП i8086?
40. Для чого використовується постбайт?
41. Звідки МП одержує інформацію про потрібну адресу у разі неявної адресації?
42. Де міститься операнд у разі регістрової адресації?
43. Яку довжину можуть мати безпосередні операнди у МП i8086?
44. Який тип адресації має команда MOV AL, [dataByte]?
45. Де міститься ефективна адреса у разі непрямої адресації?
46. Що виконує роль бази у разі базової та індексної адресації?
47. Яке головне застосування базової адресації?
48. В якому випадку використовується індексна адресація?

49. Для чого використовується стек?
50. В яких командах може використовуватись відносна адресація?
51. Які регістри використовуються у разі адресації рядків даних?
52. Які способи адресації застосовуються у МП i8086?
53. Скільки типів команд має більшість МК сімейства AVR? Наведіть коротку характеристику цих типів.
54. Наведіть типи (формати) даних МК AVR.
55. Опишіть особливості RISC-архітектури МК AVR.
56. Назвіть основні групи команд з базового набору МК AVR. Дайте коротку характеристику кожній з груп.
57. Назвіть та дайте характеристику новим командам МК AVR.
58. Що спільного та які вімінності під час виконання команд JMP та CALL?
59. Опишіть призначення та відмінності команд RET та RETI.
60. Поясніть особливості виконання команди RJMP.
61. Як компілятор використовує мітки в командах умовних переходів під час формування машинних кодів команд?
62. Чим відрізняються алгоритми виконання команд арифметичних та логічних зсувів? Наведіть приклади їх використання.
63. Який діапазон десяткових чисел відображає 8-розрядне число зі знаком?
64. Чим відрізняються мнемоніка та мнемокод команд?
65. Для чого використовується регістр PC/IP?
66. Опишіть роботу конвеєра під час виконання програми МК AVR.
67. Для чого використовуються регістри-покажчики X, Y і Z?

4. ОРГАНІЗАЦІЯ ПІДСИСТЕМИ ПЕРЕРИВАНЬ

4.1. Особливості архітектури підсистеми мікропроцесорної системи

4.1.1. Загальні відомості про систему переривань

Для підвищення ефективності роботи засобів підтримки обміну інформацією у мультимодульних системах кожному модулю системи надаються права ініціації обміну. Для цього обчислювальне ядро системи повинно мати можливість приймати запити на переривання від інших модулів, переривати обробку поточного процесу і переходити на обслуговування запитів з обміну інформацією.

Засоби, які реалізують у МПС запити на переривання, називаються підсистемою переривань і реалізуються або у вигляді окремих ВІС – контролерів переривань, або міститься у МК.

У МК запити на переривання можуть надходити ззовні, або від окремих периферійних модулів, які містяться у МК.

Підсистема переривань характеризується такими властивостями:

– кількістю джерел (рівнів, векторів, адрес) переривань;
– видом переривань: внутрішнє/зовнішнє; апаратне/програмне; програмно-апаратне; переривання, що маскується; переривання, що не маскується;
– можливістю програмного маскування і керування пріоритетами;
– порядком обслуговування та обчисленням адреси підпрограми обробки переривання і т. ін.

Підсистема переривань призначена для прийому, пріоритетної обробки й обслуговування запитів переривань [2; 3]. Під час проектування підсистеми переривань необхідно розробити підпрограми обслуговування переривань. Для обробки зовнішніх переривань додатково потрібно розробити зовнішні апаратні засоби.

4.1.2. Види переривань

Розглянемо види переривань на прикладі МП i8086.

МП i8086 має векторну (адресну) підсистему переривань із зовнішніми та внутрішніми джерелами запитів. Кожне джерело має свій «тип» – номер «входу» у таблиці адрес обробників переривань (рис. 4.1), за яким МП знаходить відповідну підпрограму обслуговування переривання.

Таблиця адрес має 256 входів з номерами від 0D до 255D і займає перші 1024 байти пам’яті програм – по чотири байти на кожен тип запиту переривань ($256 \cdot 4 = 1024$).

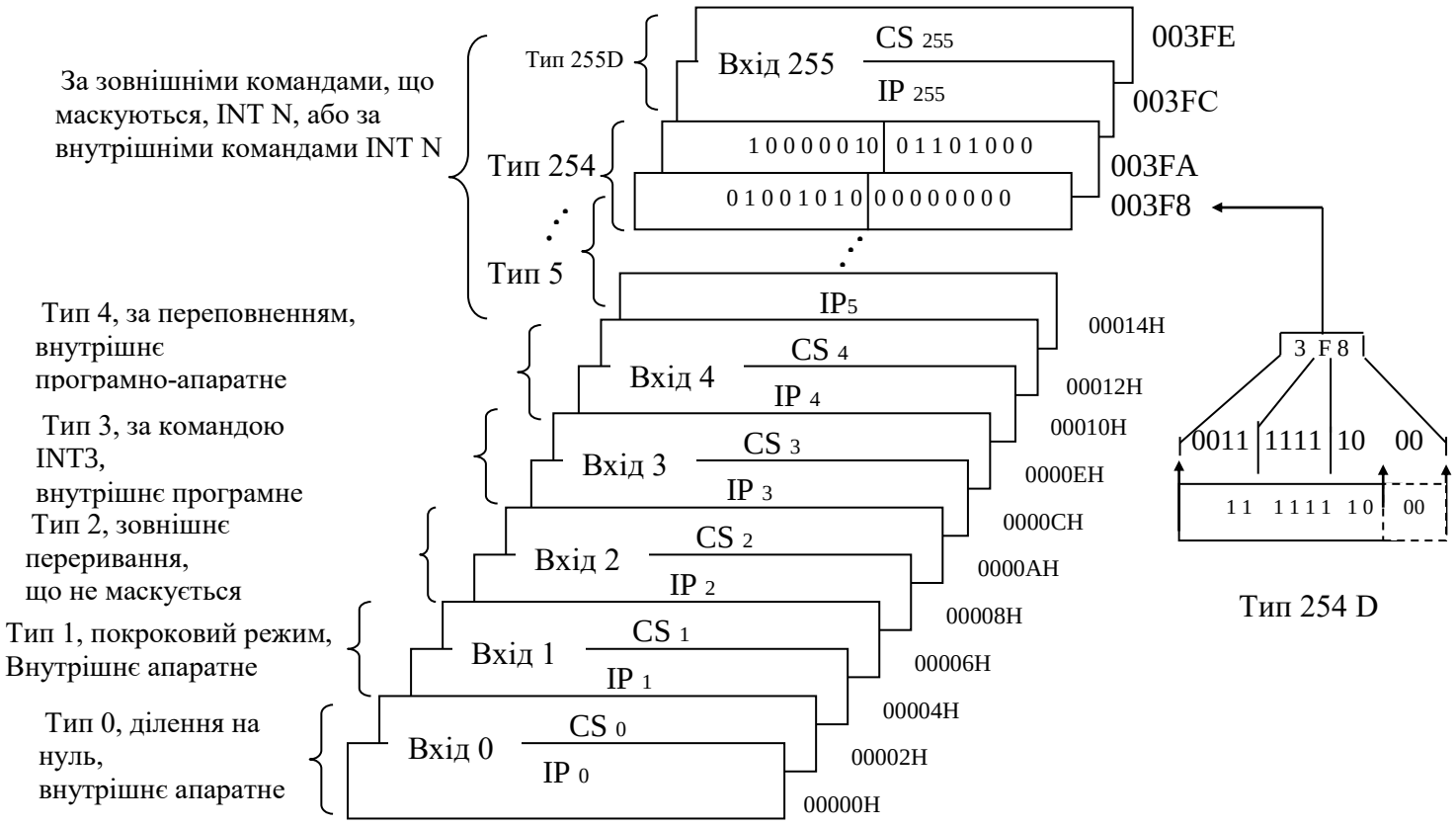


Рис. 4.1. Організація переривань у МП i8086

Два старших байти із цих чотирьох (друге слово входу у таблицю) містять логічну адресу початку сегмента команд, в якому розміщена підпрограма обробки переривання, а два молодших байти (перше слово входу у таблицю) зберігають зміщення підпрограми в обраному сегменті.

Переривання поділяються на внутрішні або зовнішні.

Внутрішні джерела переривань поділяються на апаратні, програмно-апаратні та програмні. До апаратних належать переривання від помилки ділення на нуль (тип 0) і відлагоджувальні (тип 1). Запит типу 0 генерується апаратно, якщо під час виконання команд DIV, IDIV дільник дорівнює 0. Запит типу 1 виникає за умови виконання програми в покроковому режимі після виконання чергової команди, якщо прапорець TF = 1.

До програмно-апаратного належить переривання за переповненням (тип 4), що настає, якщо виконується команда INTO і прапорець переповнення додаткового коду OF = 1.

Програмні переривання виникають під час виконання однобайтної команди INT3 (тип 3) чи двобайтової команди INT type, що імітує запит переривання будь-якого типу. Програмні переривання не маскуються.

Зовнішні запити переривання для МП i8086 надходять за двома входами МП NMI та INT. Немасковані запити NMI викликають переривання типу 2 і

сприймаються за фронтом сигналу, що зростає, на вході NMI. Запити, що маскуються, надходять за входом INT і сприймаються процесором тільки тоді, коли переривання дозволені (не замасковані) – прапорець IF = 1.

В іншому випадку, коли IF = 0, запит відкидається. Варто звернути увагу, що під час запиту за входом INT не обов'язково буде обслуговуватися переривання типу 3, як у разі виконання команди INT3. Зовнішній пристрій, що виставив запит, повинен за системною шиною даних передати у МП байт, що дозволяє ідентифікувати тип переривання.

У разі одночасного надходження декількох запитів діє система пріоритетів, що у МП i8086 виглядає так (від вищого рівня до нижчого): переривання під час ділення на нуль (вищий рівень); переривання за командами INT3, INT type, INTO; переривання за входом NMI, переривання за входом INT; переривання за прапорцем TF (покроковий режим виконання програми).

4.1.3. Особливості обробки зовнішніх переривань

Джерелами зовнішніх переривань у МПС є пристрої введення/виведення; пристрої, що задають час; аварійні ситуації (наприклад, аварія живлення) тощо.

Засоби, які реалізують обмін за зовнішніми перериваннями на основі МП, реалізуються у вигляді окремих ВІС. Останні звуться контролерами переривань. За їх допомогою можна реалізувати багаторівневу систему переривань з можливістю врахування пріоритетів і маскування входів.

У МПС на основі МП i8080 та i8086, наприклад, використовується спеціальна мікросхема i8259, за допомогою якої можна реалізувати багаторівневу систему переривань з можливістю врахування пріоритетів і маскування входів [2].

У разі проектування підсистеми переривань від зовнішніх чинників вирішують задачі: обробка запитів переривань і формування сигналу переривання для процесора; відмова або дозвіл прийому запитів переривань під час виконання програм; ідентифікація джерел запитів переривань; передача керування підпрограмам обслуговування запитів переривань; збереження поточного стану процесора у разі переходу до підпрограм обслуговування запитів переривань і його відновлення після завершення підпрограм; обробка пріоритетів запитів переривань; сполучення засобів підтримки підсистеми переривань з процесором.

Зовнішні пристрої можуть видавати сигнал на обслуговування переривань у будь-який момент часу, подаючи сигнал рівня «логічна одиниця» на вхід INT МП (апаратний запит переривань), або за допомогою встановлення ознаки (прапорця) в одному із розрядів слова стану зовнішніх пристроїв, яке зчитується МП.

В останньому випадку МПС у процесі виконання програми періодично звертається до зовнішніх пристроїв і перевіряє логічний рівень ознаки запиту на переривання. У разі підтвердження запиту МПС організовує виконання відповідної підпрограми обслуговування переривання. Підключення зовнішніх пристроїв у такому режимі нічим не відрізняється від стандартного варіанту підключення зовнішніх пристроїв до МПС (наприклад, за допомогою паралельного інтерфейсу i8055).

Якщо у системі використовується апаратний запит на обслуговування переривання, то МП, отримавши сигнал запиту INT від зовнішніх пристроїв, формує відповідний сигнал $\overline{INTA}$ – підтвердження переривання, що інформує зовнішній пристрій про те, що МП припинив виконання основної програми і перейшов до режиму обслуговування переривання.

Після завершення виконання поточної команди основної програми, МП приймає від зовнішнього пристрою системною шиною даних 8-розрядний номер – логічну адресу переходу на виконання підпрограми обслуговування переривання. МП зберігає стан поточної програми, що виконується, у стеку та переходить до підпрограми обслуговування переривання. Завершується підпрограма обслуговування переривання командою IRET, що відновлює стан перерваної основної програми зі стека та продовжує її виконання.

На наведеному рис. 4.1 розглянуто приклад обробки запиту переривання типу 254. Під час прийому запиту МП стає відомий його тип (номер у десятковому коді), множення якого на чотири (зсув вліво на 2 розряди ДК номера запиту) дає десять молодших розрядів фізичної адреси відповідного входу у таблицю адрес обробників переривань.

У старші десять розрядів адреси записуються нулі тому, що таблиця займає початкові 1024 байти пам’яті програм. Для запиту типу 254 повна 20-розрядна адреса у шістнадцятковому коді дорівнює 003F8H.

Обробка будь-якого прийнятого запиту починається із запам’ятовування вмісту регістра прапорців (англ. RF) у стеку. Прапорці IF і TF скидаються в нуль, забороняючи наступні зовнішні переривання і покроковий режим роботи.

У стек пересилається вміст регістрів CS і IP (запам’ятовується місце повернення в основну програму). За ідентифікованим типом переривання з таблиці витягається адреса входу у підпрограму обробки запиту: перше слово пересилається у регістр IP, друге – у регістр CS, після чого підпрограма виконується. Завершується вона командою IRET (повернення з переривання).

У цьому разі зі стека у зворотному порядку відновлюється вміст регістрів CS, IP та RF і керування передається перерваній програмі. У цій програмі необхідно відновити прапорці IF = 1 і TF = 1, що дозволить обробку нових зовнішніх запитів і покроковий режим роботи.

4.1.4. Маскування та призначення пріоритетів переривань

4.1.4.1. Маскування переривань

Переривання МП/МК можуть програмно маскуватися. Для цього використовуються прапорці (тригери), які приймають два значення:

– нуль, переривання заборонено;
– одиниця, переривання дозволено.

Є два види прапорців маскування переривань:

– прапорець глобального (загального) маскування всіх переривань (зазвичай знаходиться у регістрі прапорців);
– прапорці маскування переривань від окремих периферійних модулів МК.

Для МП типу i8086 та МК типу AVR це питання розглянуто у [2; 3].

4.1.4.2. Призначення пріоритетів переривань

У разі одночасного надходження переривань від декількох джерел діє система пріоритетів, особливості якої залежать від типу конкретного МП/МК. Загальний підхід призначення пріоритетів переривань полягає в тому, що зазвичай є два ступені розподілу пріоритетів. Перший ступінь реалізовано апаратно і дозволяє за умови одночасного запиту від декількох джерел обрати одне з вищим пріоритетом.

Другий ступінь реалізується програмно. У цьому разі для кожного джерела переривання у МК використовується окремий прапорець (тригер), який програмно або встановлюється в одиницю, що задає цьому перериванню високий пріоритет, або складається в нуль, що задає низький пріоритет. Очевидно, що переривання із вищим програмно заданим пріоритетом перерве підпрограму обслуговування переривання із нижчим програмно встановленим пріоритетом.

Для МК сімейства AVR, наприклад AT89C51, це питання розглянуте у [2]. Пріоритети зовнішніх переривань для МПС на МП типу i8086, можуть задаватися у разі програмування контролера переривань.

4.1.5. Визначення адреси підпрограми обробки переривання

Для кожного можливого переривання під час проектування МПС розробляється відповідна підпрограма, яка зберігається у пам’яті МПС. Для того, щоб викликати потрібну підпрограму обробки переривання треба визначити її адресу. Як це зробити залежить від типу конкретного МП/МК, який використовується у МПС.

Визначення адрес для МПС на МП типу i8086 описано вище у п. 4.1.3.

Для різних сімейств МК для визначення адрес використовують два підходи:

– всі підпрограми обробки переривань починаються з однієї адреси, а відповідь на запитання, яке конкретне переривання потрібно обробити, дається за допомогою аналізування прапорців наявності переривань;
– за кожною підпрограмою обробки переривань закріплюється конкретна адреса пам'яті, яку вказано в описі архітектури МК, наприклад AVR-мікроконтролери.

4.2. Організація підсистеми переривань мікроконтролера

4.2.1. Загальні відомості про підсистему переривань мікроконтролера

Розглянемо, як приклад, основні властивості підсистеми переривань МК типу AVR.

У разі виникнення переривання МК зберігає у стеку адресу наступної команди (вміст лічильника команд (англ. PC)) і завантажує в нього адресу відповідної підпрограми обробки переривання (вектора переривання). За цією адресою знаходиться команда відносного переходу до підпрограми обробки переривання. В кінці цієї підпрограми знаходиться команда RETI, що забезпечує повернення в основну програму і відновлення стану попередньо збереженого у стеку лічильника команд.

Оскільки основними джерелами переривань є зовнішні переривання або різні периферійні пристрої МК, то кількість переривань залежить від конкретної моделі МК.

4.2.2. Таблиця векторів переривань

AVR-мікроконтролери мають багаторівневу систему пріоритетних переривань [3; 4]. Молодші адреси пам'яті програм, починаючи з адреси $0001 або $0002 відведено під таблицю векторів переривань. У цій таблиці кожному перериванню відповідає своя адреса, яка у разі виникнення переривання завантажується в лічильник команд. Положення вектора в таблиці визначає також і пріоритет відповідного переривання: чим менше адреса, тим вище пріоритет переривання. Розмір вектора переривання залежить від обсягу пам'яті програм МК і становить 1 слово для моделей з об'ємом пам'яті менше 16 Кбайт і 2 слова для інших моделей. Для переходу до підпрограми обробки переривання використовуються команда RJMP, або – JMP.

У більшості МК сімейства Mega положення таблиці векторів переривань може бути змінено. Таблиця може розміщуватися не тільки на початку пам'яті програм, але також і на початку області завантажувача. Переміщення таблиці може бути здійснено безпосередньо в процесі виконання програми [3].

Для керування розміщенням таблиці використовується регістр керування МК MCUCR, який розміщено за адресою $35 ($55), або загальний регістр керування перериваннями GICR, розміщений за адресою $3B ($5B).

Для керування таблицею переривань у цих регістрах використовуються два молодших розряди: IVSEL (1-й розряд) і IVCE (0-й розряд). Стан прапорця IVSEL визначає положення таблиці в пам'яті програм. Якщо прапорець скинуто в нуль, то таблиця векторів переривань розміщується на початку пам'яті програм, якщо встановлено в одиницю – на початку області завантажувача.

Розмір таблиці векторів переривань залежить від моделі МК і становить від 16 до більш ніж 56 векторів. Розподіл адрес таблиці векторів переривань для різних МК сімейства наведено у [3; 4]. Під час розміщення векторів переривань в області завантажувача до значень, наведених у таблицях, слід додати значення початкової адреси області завантажувача.

Якщо переривання в роботі МК не передбачаються, то на місці таблиці векторів переривань може бути розміщена частина основної програми.

4.2.3. Обробка переривань

Для дозволу виконання відповідної підпрограми обробки переривання треба встановити в одиницю прапорець глобального дозволу переривань – I регістра SREG та встановити відповідні розряди регістрів масок переривань [3].

Обробка переривань здійснюється таким чином:

- під час виконання умов, необхідних для генерації переривання, прапорець, що відповідає цьому перериванню, встановлюється в одиницю, а прапорець I апаратно скидається, забороняючи тим самим обробку наступних переривань. Для дозволу вкладених переривань у підпрограмі обробки переривання цей прапорець треба знову встановити в одиницю;
- якщо переривання дозволено (прапорець дозволу переривання встановлено) вміст лічильника команд зберігається у стеку, після чого у лічильник команд завантажується адреса вектора відповідного переривання. При цьому прапорець переривання апаратно скидається. Ряд прапорців переривань може бути також скинуто записом одиниці в розряд регістра, який відповідає прапорцю;
- виконується підпрограма обробки переривання;
- виконується команда повернення з переривання RETI, при цьому прапорець I апаратно встановлюється в одиницю, дозволяючи обробку наступних переривань;
- автоматично зі стеку відновлюється вміст лічильника команд.

Далі основна програма продовжує своє виконання з того місця, де її було перервано.

Під час виклику підпрограм обробки переривань вміст регістра стану SREG у стеку не зберігається. Якщо це необхідно, користувач повинен самостійно запам’ятовувати вміст цього регістра під час входу в підпрограму обробки переривання і відновлювати його значення перед викликом команди RETI.

Для переривань, які викликані динамічними подіями (наприклад, для переривання, яке генерується при рівності вмісту лічильного регістра і регістра порівняння таймера), прапорець переривання встановлюється тільки в момент виникнення події. Якщо прапорець переривання скинуто, а умови генерації переривання присутні, прапорець буде встановлено тільки в момент виникнення наступної події.

Для зовнішніх переривань, які генеруються за рівнем, прапорці не передбачено, тому інформація про ці переривання буде зберігатися до наявності події, яка викликає переривання.

AVR-мікроконтролери підтримують чергу переривань. Вона працює таким чином: якщо умови генерації одного або більше переривань виникають тоді, коли прапорець глобального дозволу переривань скинуто (усі переривання заборонено), відповідні прапорці встановлюються в одиницю і залишаються в цьому стані до встановлення прапорця глобального дозволу переривань. Після цього виконується їхня обробка відповідно до пріоритету.

Час відгуку для будь-якого переривання залежить від моделі МК та становить 5 або 4 такти. Протягом цього часу відбувається збереження лічильника команд у стеку. Протягом наступних двох або трьох тактів виконується команда переходу до підпрограми обробки переривання.

Якщо переривання відбудеться під час виконання команди, яка триває кілька циклів, то генерація переривання відбудеться тільки після виконання цієї команди. Якщо ж переривання відбудеться під час перебування МК у «сплячому» режимі, то час відгуку збільшується ще на 4 або 5 тактів.

Повернення в основну програму займає 4 або 5 тактів, протягом яких відбувається відновлення лічильника команд зі стека. Після виходу з переривання процесор завжди виконує одну команду основної програми, перш ніж обслужити будь-яке відкладене переривання.

4.2.4. Зовнішні переривання AVR-мікроконтролерів

В AVR-мікроконтролерах, наприклад, сімейства Mega є два різновиди зовнішніх переривань. Переривання першого типу («звичайні») генеруються під час появи на вході зовнішнього переривання заданого сигналу.

Ці переривання присутні у всіх МК сімейства (конкретне число таких

переривань залежить від моделі). Переривання другого типу генеруються у разі будь-якої зміни стану певних виводів МК. Наявність тих чи інших зовнішніх переривань у різних моделях показано в [3; 4].

На певних входах зовнішніх переривань виявлення фронтів сигналів відбувається асинхронно, тобто не вимагає наявності тактового сигналу $f_{clkI/O}$. Наприклад, зовнішні переривання за низьким рівнем і переривання за зміною стану виводів завжди реєструються асинхронно. Відповідно, всі ці переривання можуть використовуватися для виведення МК зі «сплячих» режимів, відмінних від режиму Idle.

Для дозволу/заборони «звичайних» зовнішніх переривань залежно від моделі використовується або загальний регістр керування перериваннями GICR, або регістр маскування зовнішніх переривань EIMSK. Для дозволу/заборони переривань за зміною стану виводів використовуються або ті ж самі регістри, або окремий регістр PCICR. Формати регістрів GICR, EIMSK і PCICR різних моделей та опис їх розрядів наведено у [3; 4].

Для індикації настання «звичайних» зовнішніх переривань у МК Mega використовується або загальний регістр прапорців переривань GIFR, або регістр прапорців зовнішніх переривань EIFR. Для індикації переривань за зміною стану виводів використовуються або ті ж самі регістри, або окремий регістр PCIFR [3; 4].

Усі зовнішні переривання генеруються навіть у тому випадку, якщо відповідні виводи сконфігуровані як виходи. Ця особливість МК дозволяє генерувати переривання програмно.

4.2.5. Внутрішні переривання AVR-мікроконтролерів

Крім зовнішніх, більшість переривань в AVR-мікроконтролерах є внутрішніми і сигналізують про зміну стану окремих периферійних модулів. Кожний модуль може підтримувати одне або декілька переривань і кожне з цих переривань може бути окремо дозволено або заборонено. Запит на переривання генерується в тому випадку, якщо в будь-який момент після його дозволу встановленням відповідних прапорців виявляється відповідна йому умова. У кожного переривання є окремий вектор переривання [3].

4.2.6. Особливості використання модуля переривань у МК XMega

МК XMega мають програмований багаторівневий контролер переривань – PMIC-контролер, який відповідає за обробку запитів переривань з урахуванням їх пріоритетів і рівнів. Після того, як запит переривання підтверджується PMIC-контролером, у лічильник програми завантажується адреса вектора переривання, а потім виконується процедура обробки переривання. Більш детально робота та програмування підсистеми переривань МК XMega описано у [11].

Контрольні запитання та завдання

1. Для чого призначена підсистема переривань?
2. Назвіть основні види переривань.
3. Як відбувається маскування переривань?
4. Назвіть основні джерела переривань.
5. Які регістри потрібні для програмування та керування роботою підсистеми переривань?
6. Якими основними командами реалізується обробка переривань?
7. Як виконується команда LCALL?
8. Як виконується команда RETI?
9. Як виконується команда RET?
10. Як відбувається повернення з підпрограми обробки переривання до основної програми?
11. Яку роль відіграє стек під час проектування підсистеми переривань?
12. Як у МП i8086 обчислюється адреса підпрограми обробки переривання?
13. Де зберігаються підпрограми обробки переривань у МК типу AVR?
14. Яку 20-розрядну фізичну адресу пам’яті у шістнадцятковому коді буде сформовано у МП i8086 у разі отримання запиту зовнішнього переривання типу 165?
15. Опишіть послідовність обробки переривань у МП типу AVR.
16. Чому під час виходу із підпрограми обробки зовнішнього переривання у МК треба використовувати команду RETI?
17. Назвіть джерела переривань і адреси підпрограм, що їх обслуговують, у МК типу AVR.
18. Скільки входів має таблиця адрес обробників переривань у МПС на основі МП i8086?
19. Де зберігається таблиця адрес обробників переривань у МПС на основі МП i8086?
20. Скільки байтів виділяється на кожен тип переривань у таблиці адрес обробників переривань у МПС на основі МП i8086?
21. Яким чином виконується обробка переривань у МК типу AVR?
22. Де зберігається прапорець глобального дозволу переривань у МК типу AVR?
23. Скільки тактів становить час відгуку для переривання у МК типу AVR?
24. Назвіть два різновиди зовнішніх переривань для МК типу AVR?

5. АРХІТЕКТУРА МОДУЛЯ ПРОГРАМОВАНИХ ТАЙМЕРІВ

5.1. Способи формування інтервалів часу та підрахунок зовнішніх подій

У МПС часто треба формувати інтервали часу: часові затримки, одиничні імпульси, послідовності імпульсів і т. ін.

Для цього є такі способи:

– апаратний;

– програмний;

– апаратно-програмний.

Перший спосіб реалізується використанням спеціалізованих електронних пристроїв, наприклад, ліній затримки, одновібраторів, мультивібраторів і т. ін. [1].

У другому способі, використовуючи систему команд конкретного МП чи МК і з огляду на те, що виконання кожної команди займає деякий час, можна розробити підпрограми, що формують часові затримки, одиничні імпульси, послідовності імпульсів і т. ін.

Апаратно-програмний спосіб реалізується застосуванням програмованих таймерів.

Восьми- та 16-розрядні МП не містять вбудованих таймерів і для формування інтервалів часу використовують зовнішні мікросхеми, наприклад, і8253 [2].

МК можуть містити декілька внутрішніх 8- та 16-розрядних програмованих таймерів [2; 3].

Окрім формування часових інтервалів, таймери можуть виконувати підрахунок імпульсів, що ідентифікують настання зовнішніх подій, тобто працювати як лічильники зовнішніх подій з апаратним чи програмним запуском. Таймери сучасних МК, окрім цього, можуть працювати в режимах: захоплення, порівняння та широтно-імпульсної модуляції.

5.2. Особливості архітектури модуля таймерів мікроконтролерів

5.2.1. Загальна характеристика модуля таймерів

У цьому розділі розглянуто МК сімейства AVR, які залежно від моделі, мають до шести таймерів/лічильників загального призначення, з яких два – 8-розрядні, чотири – 16-розрядні [3; 4].

Ці таймери/лічильники можуть працювати в таких режимах:

– таймера;

– лічильника зовнішніх подій;

– захоплення;

– порівняння (збігу);
– широтно-імпульснї модуляції;
– асинхронному режимі в якості годинника реального часу;
– вартового таймера.

Назву таймери/лічильники використовують тому, що основними режимами цих пристроїв є режим таймера або лічильника зовнішніх подій. Спільним для цих режимів є те, що основу пристрою складає двійковий лічильник, який підсумовує та переключається під дією імпульсів, які надходять на його лічильний вхід. Ці імпульси змінюють стан лічильника від початкового, якій встановлюється програмно, до переповнення, коли перед приходом чергового імпульса на вхід лічильника його тригери перебувають у стані логічної одиниці. Під впливом цього імпульсу всі тригери скидаються в нуль. Цей процес називається переповненням лічильника та відображається встановленням в одиничний стан тригера (прапорця). За переповненням може викликатися відповідна підпрограма, якщо її дозволено масками (прапорцями дозволу).

Відмінність режимів таймера та лічильника зовнішніх подій полягає в джерелах надходження імпульсів на вхід лічильника.

У режимі таймера ці імпульси надходять від системного тактового генератора МК. Частота імпульсів може програмно ділитися на відповідне значення.

У режимі лічильника зовнішніх подій на вхід двійкового лічильника надходять зовнішні імпульси, які ідентифікують настання відповідної події. Ці імпульси (події) підраховуються лічильником.

Ідея використання таймерів/лічильників полягає в наступному. На початку програмування пристрою при відомій тактовій частоті МК встановлюється коефіцієнт її ділення, що задає потрібний період імпульсів на вході лічильника. Потім програмно записується початкове значення в лічильник (це може бути нульове значення). Початкове значення обирається таким чином, щоб від початку лічення до переповнення на вхід лічильника надійшла потрібна кількість імпульсів з відомим періодом. Факт переповнення відображається встановленням прапорця, що буде вказувати на те що з моменту початку лічення до переповнення надійшла потрібна кількість імпульсів, тобто формується потрібний часовий інтервал. Аналогічно може програмуватися підрахунок потрібної кількості імпульсів, які відбражають настання зовнішніх подій.

У режимі захоплення модуль таймера МК, окрім лічильника, містить додатковий регістр, в який під час надходження зовнішнього імпульсу на відповідний вхід МК захоплюється (пересилається) поточний стан лічильника. У разі захоплення встановлюється прапорець та викликається підпрограма обробки

цього стану, якщо вона дозволена. Наявність цього режиму розширює можливості програмування часових інтервалів.

У режимі порівняння (збігу) модуль таймера МК окрім лічильника містить декілька додаткових регістрів, у які програмно записуються значення, з якими буде порівнюватися поточний стан лічильника.

У момент збігу встановлюється прапорець та може викликатися підпрограма, якщо вона дозволена. Окрім цього для кожного каналу збігу виділяється окремий вивід МК, на якому в момент збігу формується відповідний сигнал.

На основі останнього режиму на виділених виводах МК може формуватися послідовність прямокутних імпульсів. Частота цих імпульсів підтримується програмно на постійному рівні, але може змінюватися їх шпаруватість. Тобто таймер може формувати декілька широтно-імпульсно-модульованих сигналів (ШІМ-сигналів), які можуть використовуватися, наприклад, для керування двигунами постійного струму.

Виводи, які використовує кожен таймер/лічильник, є лініями портів введення/виведення загального призначення, а функції, які ними реалізуються під час роботи разом з таймерами/лічильниками, є їхніми альтернативними функціями. Наприклад, виводи AVR-мікроконтролерів сімейства Mega, що використовуються таймерами/лічильниками загального призначення з описанням їх функцій, зведено у таблицю, яку наведено у [3].

Під час завершення роботи таймерів в одному з наведених вище режимів можуть виникати переривання основної програми та викликатися відповідні підпрограми обробки цих переривань. Для програмування роботи таймерів/лічильників у режимі переривань використовуються регістри прапорців переривань та регістри масок переривань. Назву та формати цих регістрів наведено у [3].

5.2.2. Попередні дільники таймерів/лічильників

Таймери/лічильники містять блок попередніх дільників, які призначено для формування тактових сигналів f_{clkTn}, де n – номер таймера.

Спрощену структурну схему блока попереднього дільника таймерів, що не мають асинхронного режиму роботи, наведено на рис. 5.1.

На рис. 5.2 наведено структурну схему блока попереднього дільника таймерів/лічильників, що мають можливість роботи в асинхронному режимі в якості годинника реального часу.

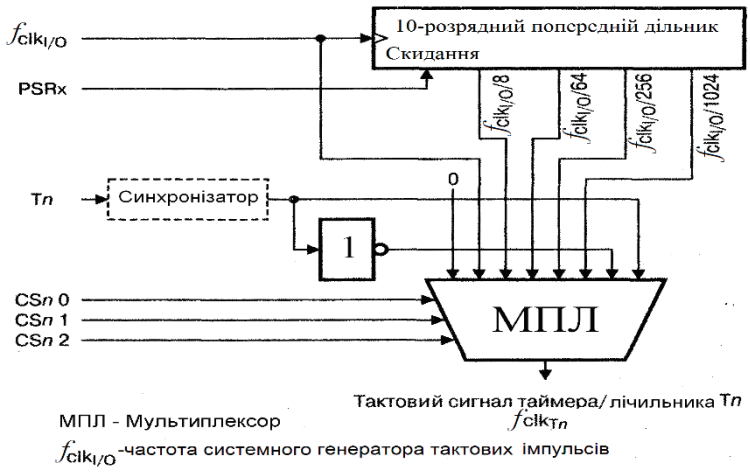


Рис. 5.1. Блок попереднього дільника таймера/лічильника без асинхронного режиму

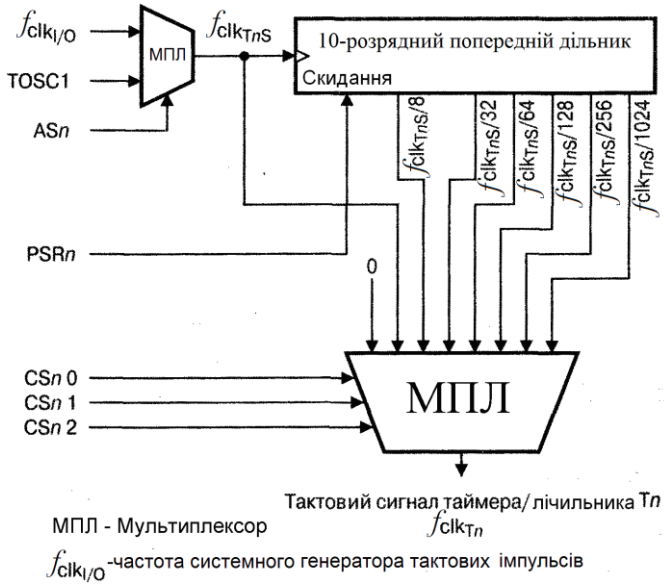


Рис. 5.2. Блок попереднього дільника таймера/лічильника з асинхронним режимом

У кожному блоку містяться власне 10-розрядний попередній дільник і вихідний мультиплексор – селектор тактового сигналу, а для таймерів, що мають можливість роботи в асинхронному режимі, – ще вхідний мультиплексор тактового сигналу. За схемою, яку наведено на рис. 5.2, виконано, наприклад, попередній дільник таймера/лічильника T0 моделей ATmega64x/128x та таймера/лічильника T2 інших моделей.

Усі таймери/лічильники кожної моделі сімейства AVR, що не мають асинхронного режиму роботи, використовують той самий 10-розрядний попередній дільник, при цьому керування тактовим сигналом кожного таймера/лічильника здійснюється індивідуально.

Попередні дільники працюють незалежно від таймерів/лічильників. Наслідком цього є, зокрема, невизначений проміжок часу між запуском таймера/лічильника та першим його відліком під час роботи разом із попереднім

дільником. Щоб уникнути цієї невизначеності, можна скористатися засобами, які описано у [3; 4].

5.2.3. Керування попередніми дільниками

Усі МК сімейства дозволяють здійснювати скидання попередніх дільників, а деякі моделі дозволяють також здійснювати їхню зупинку. Для цього використовується регістр спеціальних функцій введення/виведення (SFIOR), а для останніх моделей – регістр GTCCR. Формати цих регістрів для деяких моделей МК наведено у [3; 4].

5.2.4. Використання зовнішнього тактового сигналу

В якості тактового сигналу таймерів/лічильників, що не мають асинхронного режиму роботи, може використатися зовнішній сигнал (див. рис. 5.1).

Зовнішній сигнал надходить на вхід T*n* (*n* = 0...5) МК та перш ніж надійти на вхід селектора тактового сигналу, проходить через спеціальний вузол, що має схему синхронізації та детектор фронтів [3].

5.3. Архітектура 8-розрядних таймерів/лічильників AVR-мікроконтролерів

Більшість МК сімейства Mega мають два 8-розрядних таймери/лічильники: T0 та T2, які мають три варіанти виконання. Приклади структурних схем всіх трьох варіантів, функції, які вони можуть виконувати, регістри керування та режими роботи описано у [3]. Опис основних режимів роботи 8-розрядних таймерів/лічильників буде наведено нижче під час розгляду 16-розрядних таймерів/лічильників.

8-розрядні таймери/лічильники деяких МК можуть працювати у асинхронному режиму роботи (мають блок RTC), який у разі відповідному відповідного налаштування викликає переривання з періодом, який дорівнює 1 секунді.

На відміну від основного кварцу МК, блок RTC використовує додатковий кварц, який має частоту 32768 Гц. Це дозволяє під час ділення цієї частоти: 32768/128/256 отримати рівно одну секунду. Наприклад, 8-розрядний таймер/лічильник T0 у моделях ATmega64x і ATmega128x має блок RTC та може працювати в асинхронному режимі. В цьому режимі на вхід попереднього дільника надходить сигнал від кварцового генератора, що дозволяє використовувати таймер/лічильник як годинник реального часу [6]. Задавачем частоти сигналу може бути як кварцовий резонатор, що підключається до виводів TOSC1 та TOSC2 МК, так і сигнал від зовнішньої схеми, що подається на вивід TOSC1. Незважаючи на те що тактовий генератор таймера/лічильника

налаштовано на частоту 32768 Гц, частота кварцового резонатора або сигналу від зовнішньої схеми може перебувати в межах 0...256 кГц, при цьому вона має бути в чотири рази менше частоти тактового сигналу МК.

5.4. Архітектура 16-розрядних таймерів/лічильників AVR-мікроконтролерів

5.4.1. Загальна характеристика таймерів

У багатьох моделях мікроконтролерів сімейства Mega є шістнадцятирозрядні таймери/лічильники T1, T3, а в останніх моделях МК Mega є ще два 16-розрядних таймери: T4 та T5.

Як і 8-розрядні таймери/лічильники T0 та T2, вони можуть використовуватися для формування часових інтервалів, для підрахунку кількості зовнішніх подій, генерації ШІМ-сигналів змінної розрядності, за зовнішнім сигналом зберігати свій поточний стан в окремому регістрі введення/виведення (режим «захоплення»). Таймери/лічильники різних моделей відрізняються тільки кількістю блоків порівняння і відповідно кількістю каналів генерації ШІМ-сигналів [3]. Спрощену структурну схему одного з таймерів/лічильників, наприклад, моделі ATmega64x/ATmega128x, наведено на рис. 5.3.

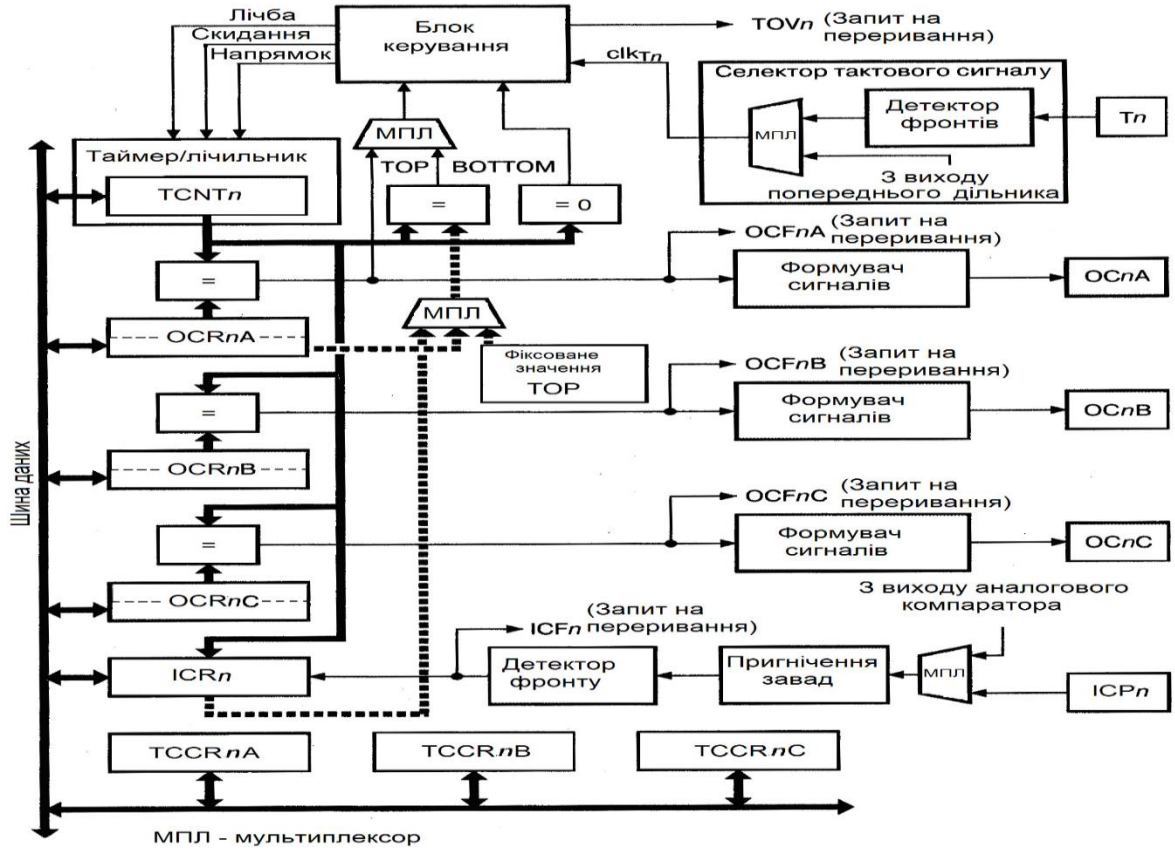


Рис. 5.3. Структурна схема 16-розрядних таймерів/лічильників T1, T3…T5

У таймері/лічильнику містяться такі регістри введення/виведення (де n = 1, 3, 4, 5):

– 16-розрядний лічильний регістр TCNTn;
– 16-розрядний регістр захоплення ICRn;
– два або три 16-розрядних регістри порівняння OCRnA, OCRnB, OCRnC;
– два або три 8-розрядних регістри керування TCCRnA, TCCRnB, TCCRnC.

Фізично кожен із 16-розрядних регістрів розміщено в двох регістрах введення/виведення, назви яких утворюються додаванням до назви регістра літери «H» (старший байт) і «L» (молодший байт). Лічильний регістр таймера лічильника TCNT1 розміщується, наприклад, у регістрах TCNT1H:TCNT1L.

Адреси регістрів 16-розрядних таймерів/лічильників наведено у [3].

Таймери/лічильники можуть генерувати переривання у разі виникнення таких подій:

– у разі переповнення лічильного регістра;
– у разі рівності лічильного регістра і регістра порівняння (по одному перериванню на кожен блок порівняння);
– у разі збереження вмісту лічильного регістра у регістрі захоплення.

Прапорці всіх переривань перебувають у регістрах TIFR/ETIFR/TIFRn, а дозвіл/заборона цих переривань здійснюється встановленням/скиданням відповідних прапорців регістрів TIMSK/ETIMSK/TIMSKn [3].

Лічильний регістр таймера/лічильника TCNTn залежно від режиму роботи модуля скидається, інкрементується, або декрементується з кожним імпульсом тактового сигналу таймера/лічильника f_{clkn}. Незалежно від того, присутній тактовий сигнал чи ні, регістр доступний у будь-який момент часу, як для зчитування, так і для запису, при цьому будь-яка операція запису в лічильний регістр блокує роботу всіх блоків порівняння на час одного періоду тактового сигналу таймера/лічильника.

Після подачі напруги живлення в регістрі TCNTn знаходиться нульове значення. У разі деяких змін стану таймера/лічильника, обумовлених режимом його роботи, встановлюється прапорець TOVn відповідного регістра. Дозвіл переривання в цьому випадку здійснюється встановленням в одиницю розряду TOIEn відповідного регістра маски.

Регістри OCRnA/OCRnB/OCRnC містяться у блоках порівняння.

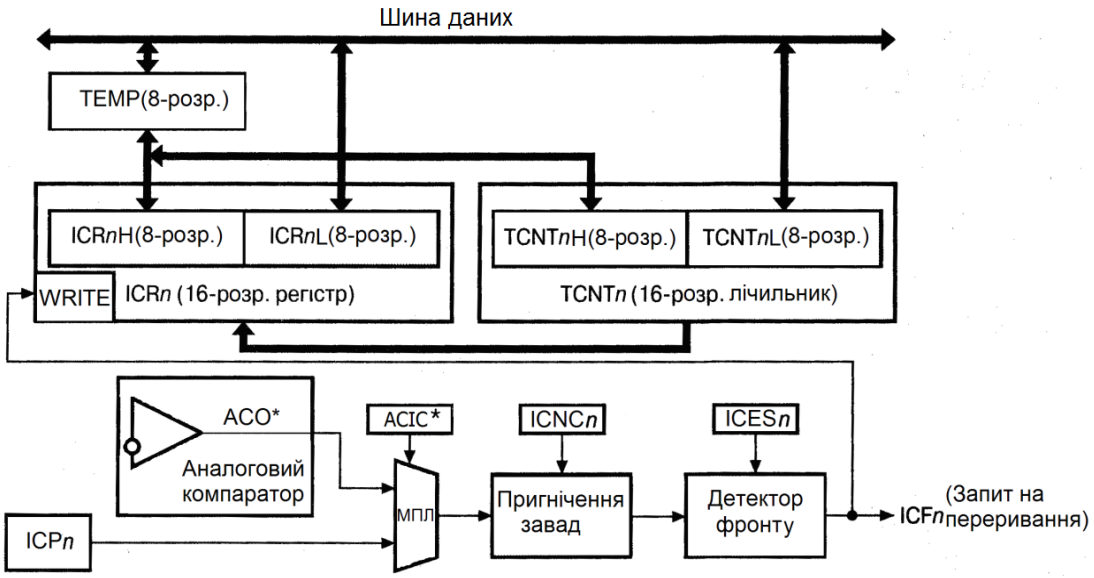
Під час роботи таймера/лічильника у кожному машинному циклі виконується порівняння цих регістрів із TCNTn. У випадку рівності вмісту регістра порівняння і лічильного регістра у наступному машинному циклі встановлюється відповідний прапорець OCFnA/OCFnB/OCFnC відповідного регістра прапорців і генерується переривання, якщо воно дозволено.

У разі виникнення цієї події може змінюватися стан виводу OCnA/OCnB/OCnC МК. Для цього цей вивід має бути сконфігурований як вихідний (відповідний розряд регістра DDRx має бути встановлений в одиницю).

Особливістю роботи блока порівняння в режимах, які призначено для формування ШІМ-сигналів, є подвійна буферизація запису в регістри порівняння. Вона полягає в тому, що число, яке записується, насправді тимчасово зберігається в спеціальному буферному регістрі. Зміна вмісту регістра порівняння відбувається тільки у разі досягнення лічильником максимального або мінімального значення.

5.4.2. Структурна схема блока захоплення

Спрощену структурну схему блока захоплення наведено на рис. 5.4.



Примітка. * Вихід аналогового компаратора може впливати на таймер/лічильник T1 (ACIC = 1, див. розд. 9).

Рис. 5.4. Структурна схема блока захоплення

У блоку захоплення міститься регістр захоплення ICRn, який призначено для збереження у визначений момент часу стану таймера/лічильника.

Ця дія може виконуватися або за активним фронтом сигналу на виводі ICPn МК, або, наприклад, для таймера/лічильника T1, за сигналом від аналогового компаратора.

Одночасно із записом у регістр захоплення встановлюється прапорець ICFn відповідного регістра прапорців і генерується запит на переривання. Дозвіл цього переривання здійснюється встановленням в одиницю розряду TICIEn відповідного регістра маски. В режимах, в яких регістр захоплення визначає модуль лічби таймера/лічильника (табл. 5.2), можна робити запис у регістр ICRn

програмно. Вивід ICPn у цих режимах відключено і функцію захоплення, відповідно, вимкнено.

Кожен 16-розрядний регістр таймерів/лічильників фізично розміщується у двох 8-розрядних регістрах. Для звернення до них потрібно виконати по дві операції зчитування або запису. Щоб запис або зчитування обох байтів вмісту 16-розрядного регістра відбувалися одночасно, у кожному таймері/лічильнику є спеціальний 8-розрядний регістр TEMP (див. рис. 5.4), який призначено для зберігання значення старшого байта. Цей регістр використовується тільки процесором і програмно недоступний.

Для виконання циклу запису 16-розрядного регістра першим має бути завантажений старший байт, що записується у регістр TEMP. Під час наступного запису молодшого байта він з'єднується з вмістом регістра TEMP, і обидва байти у тому самому машинному циклі записуються у 16-розрядний регістр. Якщо потрібно змінити кілька 16-розрядних регістрів таймера/лічильника, а старші байти всіх значень, що записуються, однакові, завантаження старшого байта достатньо виконати тільки один раз.

Для виконання циклу зчитування 16-розрядного регістра першим має бути зчитаний молодший байт. Під час його зчитування вміст старшого байта записується у регістр TEMP. Під час наступного зчитування старшого байта повертається значення, яке збережено у регістрі TEMP. Виняток становлять тільки регістри порівняння OCRn A/B/C, під час зчитування яких регістр TEMP не використовується.

Під час виконання циклу звернення до 16-розрядного регістра таймера/лічильника переривання мають бути заборонені. В іншому випадку, якщо переривання відбудеться між двома командами звернення до 16-розрядного регістра, а в підпрограмі обробки цього переривання також буде зроблено звернення до одного із 16-розрядних регістрів того ж таймера/лічильника, вміст регістра TEMP буде змінено. Як наслідок, результат звернення до 16-розрядного регістра в основній програмі буде неправильним.

Для керування схемою захоплення використовуються два розряди регістра TCCRnB – ICNCn і ICESn. Розряд ICNCn керує схемою пригнічення завад. Якщо цей розряд скинуто у нуль, схему пригнічення завад вимкнено і захоплення виконується за першим активним фронтом на виході мультиплексора (див. рис. 5.4).

Якщо цей розряд встановлено в одиницю, то з появою активного фронту виконується 4 вибірки з частотою, рівною тактовій частоті МК. Захоплення буде виконано тільки в тому випадку, якщо усі вибірки мають рівень, що відповідає активному фронту сигналу (логічна одиниця для наростаючого та логічний нуль для спадаючого).

Активний фронт сигналу, тобто фронт, за яким буде виконано збереження вмісту лічильного регістра у регістрі захоплення, визначається станом розряду ICESn. Якщо цей розряд скинуто у нуль, то активним є спадаючий фронт. Якщо розряд встановлено в одиницю, то активним є наростаючий фронт.

Для захоплення за сигналом з виводу ICPn, цей вивід має бути сконфігурований як вхідний (розряд регістра DDRx, що відповідає виводу, має бути скинутий у нуль). Якщо ж він буде сконфігурований як вихідний, захоплення можна буде здійснювати програмно, керуючи відповідним розрядом порту.

Між зміною стану входу блока захоплення і копіюванням лічильного регістра у регістр захоплення таймера/лічильника проходить деякий час. Цю затримку вносить каскад, що складається із синхронізатора (на рис. 5.3, 5.4 не показано) і детектора фронту. Величина затримки становить 2,5...3,5 тактів. У разі ввімкнення схеми пригнічення завад затримка збільшується ще на 4 такти. Для керування таймером/лічильником використовуються три регістри керування: TCCRnA, TCCRnB, TCCRnC. Формат та опис цих регістрів наведено у [3].

5.4.3. Керування тактовим сигналом

Формування тактового сигналу для таймерів/лічильників f_{clkTn} (n = 1, 3, 4, 5) здійснюється блоком попереднього дільника, що був розглянутий у п. 5.2.2. В якості тактового сигналу f_{clkTn} таймерів/лічильників T1, T3, T4 і T5 більшості МК може використовуватися:

- системний тактовий сигнал ($f_{clkTn} = f_{clk\ I/0}$);
- масштабований системний тактовий сигнал ($f_{clkTn} = f_{clkI/0}/N$, де N – коефіцієнт ділення попереднього дільника);
- зовнішній сигнал, що надходить на вхід Tn МК ($f_{clkTn} = f_{clk\ EXT}$).

Вибір джерела тактового сигналу, а також запуск і зупинка таймерів/лічильників здійснюються за допомогою розрядів CSn2...CSn0 регістра керування таймером TCCRnB відповідно до табл. 5.1.

5.4.4. Режими роботи 16-розрядних таймерів/лічильників

5.4.4.1. Загальні відомості про режими роботи

Режим роботи 16-розрядних таймерів/лічильників визначається станом відповідних розрядів регістра TCCRnA (табл. 5.2). У деяких моделях режими 8...15 відсутні.

Таблиця 5.1. **Вибір джерела тактового сигналу таймерів/лічильників Tn**

CSn2	CSn1	CSn0	Джерело тактового сигналу	
			T3 у моделях ATmega162x	**Інші**
0	0	0	Таймер/лічильник зупинено	Таймер/лічильник зупинено
0	0	1	$f_{clkI/0}$	$f_{clkI/0}$
0	1	0	$f_{clkI/0}/8$	$f_{clkI/0}/8$
0	1	1	$f_{clkI/0}/64$	$f_{clkI/0}/64$
1	0	0	$f_{clkI/0}/256$	$f_{clkI/0}/256$
1	0	1	$f_{clkI/0}/1024$	$f_{clkI/0}/1024$
1	1	0	$f_{clkI/0}/16$	Вивід Tn, лічба виконується за спадаючим фронтом
1	1	1	$f_{clkI/0}/32$	Вивід Tn, лічба виконується за наростаючим фронтом

Примітка. n = 1, 3, 4 або 5.

5.4.4.2. Режим Normal

Режим Normal є найпростішим режимом роботи таймерів/лічильників. У ньому лічильний РГ функціонує як звичайний лічильник, що підсумовує.

З кожним імпульсом тактового сигналу f_{clkTn} здійснюється інкремент лічильного регістра. Під час переходу через значення $FFFF виникає переповнення, і лічба продовжується зі значення $0000. У тому ж такті сигналу f_{clkTn}, в якому скидається в нуль регістр TCNTn, встановлюється в одиницю прапорець переривання за переповненням TOVn. Блоки порівняння обох таймерів можуть використовуватися як для генерації переривань, так і для формування сигналів на виходах OCnA/OCnB/OCnC.

Поведінка виходів OCnA/OCnB/OCnC кожного з блоків порівняння таймерів/лічильників Tn визначається станом розрядів COMnx1:COMnx0 регістрів TCCRnA, як показано в табл. 5.3.

Стан виходу будь-якого блока порівняння також може бути змінено примусово, записом одиниці у розряд FOCnA/FOCnB/FOCnC регістра TCCRnC або регістра TCCRnA. Зауважимо, що переривання при цьому не генерується.

Таблиця 5.2. **Режими роботи 16-розрядних таймерів/лічильників**

Номер режиму	WGMn3	WGMn2	WGMn1	WGMn0	Режим роботи таймера/лічильника Tn	Модуль лічби (TOP)	Оновлення регістрів TOVn	Момент встановлення прапорця TOVn
0	0	0	0	0	Normal	$FFFF	Негайно	$FFFF
1	0	0	0	1	Phase correct PWM, 8-розрядний	$00FF	За TOP	$0000
2	0	0	1	0	Phase correct PWM, 9-розрядний	$01FF	За TOP	$0000
3	0	0	1	1	Phase correct PWM, 10-розрядний	$03FF	За TOP	$0000
4	0	1	0	0	CTC (скидання за збігом)	OCRnA	Негайно	$FFFF
5	0	1	0	1	Fast PWM, 8-розрядний	$00FF	За TOP	За TOP
6	0	1	1	0	Fast PWM, 9-розрядний	$01FF	За TOP	За TOP
7	0	1	1	1	Fast PWM, 10-розрядний	$03FF	За TOP	За TOP
8	1	0	0	0	Phase and Frequency Correct PWM	ICRn	$0000	$0000
9	1	0	0	1	Phase and Frequency Correct PWM	OCRnA	$0000	$0000
10	1	0	1	0	Phase correct PWM	ICRn	За TOP	$0000
11	1	0	1	1	Phase correct PWM	OCRnA	За TOP	$0000
12	1	1	0	0	CTC (скидання за збігом)	ICRn	Негайно	$FFFF
13	1	1	0	1	Зарезервовано	_	_	_
14	1	1	1	0	Fast PWM	ICRn	За TOP	За TOP
15	1	1	1	1	Fast PWM	OCRnA	За TOP	За TOP

Примітка. n = 1, 3, 4 або 5.

Таблиця 5.3. **Керування виводом OCnA/OCnB/OCnC у режимі Normal**

COMnx1	COMnx0	Опис
0	0	Таймер/лічильник Tn відключено від виводу OCnx
0	1	Стан виводу змінюється на протилежний
1	0	Вивід скидається в нуль
1	1	Вивід встановлюється в одиницю

Примітка. n = 1, 3, 4 або 5; x = A, B або C.

5.4.4.3. Режим «Скидання за збігом»

Лічильний регістр у режимі «Скидання за збігом» (англ. Chop on Timer Coincidence) функціонує як звичайний лічильник, що підсумовує, інкремент якого здійснюється кожним імпульсом тактового сигналу f_{clkTn}.

Однак максимально можливе значення лічильного регістра (модуль лічби – TOP) та роздільна здатність лічильника в режимі «Скидання за збігом» визначається або регістром порівняння OCRnA, або регістром захоплення ICRn (див. табл. 5.2). Після досягнення максимального значення лічба продовжується зі значення $0000. У тому ж такті сигналу f_{clkTn}, в якому скидається в нуль лічильний регістр, встановлюється прапорець переривання TOVn.

Часові діаграми для цього режиму роботи таймера/лічильника наведено на рис. 5.5.

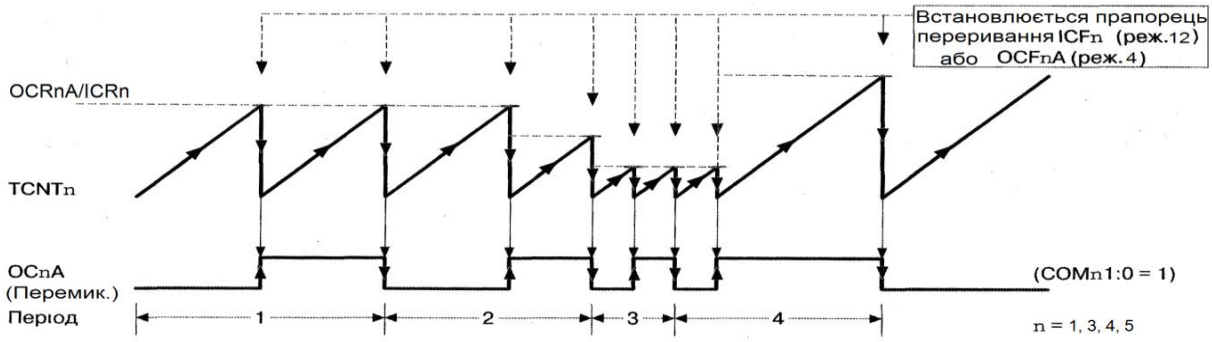


Рис. 5.5. Часові діаграми для режиму СТС

У разі досягнення лічильником максимального значення встановлюється прапорець:

- OCFnA, якщо модуль лічби визначається регістром порівняння OCRnA, режим 4;
- ICFnA, якщо модуль лічби визначається регістром захоплення ICRnA, режим 12.

Одночасно із встановленням прапорця може змінюватися стан виводу OCnx МК. Поведінка виводу визначається станом розрядів COMnx1:COMnx0 регістрів TCCRnA (табл. 5.4).

Для генерації сигналу заданої частоти необхідно записати в розряди COMnx1:COMnx0 значення «01» (перемикання стану виводу).

Таблиця 5.4. **Керування виводом OCnA у режимі CTC**

COMnx1	COMnx0	Опис
0	0	Таймер/лічильник *Tn* відключено від виводу OCnx
0	1	Стан виводу змінюється на протилежний
1	0	Вивід скидається у нуль
1	1	Вивід встановлюється в одиницю

Примітка. n = 1, 3, 4 або 5; x = A, B або C.

Частота сигналу, тривалість імпульсу та період сигналу, який генерується, визначаються виразами:

$$f_{OCnx} = f_{clkI/0}/2 \cdot N(1 + Y), \quad (5.1)$$

$$t_{імп} = N \cdot (1 + Y) \cdot T_{clkI/0}, \quad (5.2)$$

$$T_{OCnx} = 2\, t_{імп} = 2 \cdot N \cdot (1 + Y) \cdot T_{clkI/0}, \quad (5.3)$$

де N – коефіцієнт ділення попереднього дільника; Y – модуль лічби, обумовлений значенням, що знаходиться у регістрі OCRnA або ICRnA; $f_{clkI/0}$, $T_{clkI/0}$ – відповідно частота та період генератора тактових імпульсів підсистеми введення/виведення.

Як і в режимі Normal, стан виводу OCnx за необхідності може бути змінено примусово записом одиниці у розряд FOCnx регістра TCCRnC або регістра TCCRnA. Переривання при цьому не генерується і скидання лічильного регістра не виконується.

5.4.4.4. Режим «Швидкодіючий ШІМ»

Режим «Швидкодіючий ШІМ» (англ. Fast PWM) дозволяє генерувати високочастотний сигнал із широтно-імпульсною модуляцією. Цей режим практично повністю ідентичний однойменному режиму 8-розрядних таймерів/лічильників [3; 4]. Відмінність полягає тільки в тому, що максимальне значення, до якого буде рахувати лічильник, може програмно змінюватися (табл. 5.2).

Лічильний регістр у цьому режимі функціонує як лічильник, що підсумовує, інкремент якого здійснюється кожним імпульсом тактового сигналу f_{clkTn}. Стан лічильника змінюється від $0000 до максимального значення, після чого лічильний регістр скидається і цикл повторюється. Максимальне значення лічильника (роздільна здатність ШІМ-сигналу) є або фіксованим значенням, або визначається вмістом окремих розрядів керуючих регістрів таймерів/лічильників (табл. 5.2).

Роздільна здатність R визначається виразом:

$$R = \log(TOP + 1)/\log 2, \quad (5.4)$$

де TOP – модуль лічби.

Роздільну здатність модулятора у режимі Fast PWM наведено у табл. 5.5.

Таблиця 5.5. **Роздільна здатність модулятора у режимі Fast PWM**

Номер режиму	WGMn3	WGMn2 (CTC1)*	WGMn1 (PWM11)*	WGMn0 (PWM10)*	Роздільна здатність	Модуль лічби (TOP)
5	0	1	0	1	8 розрядів	$00FF
6	0	1	1	0	9 розрядів	$01FF
7	0	1	1	1	10 розрядів	$03FF
14**	1	1	1	0	змінна (2...16)	ICRn ($0003...$FFFF)
15**	1	1	1	1	змінна (2...16)	OCRnA ($0003...$FFFF)
* У моделях ATmega161x/163x/323x. ** У моделях ATmega161x/163x/323x відсутні.						

Примітка. n = 1, 3, 4 або 5.

У разі досягнення лічильником максимального значення встановлюється прапорець переривання TOVn. Одночасно з ним встановлюється прапорець ICFn (режим 14) або OCFn (режим 15). У разі рівності вмісту лічильного регістра та якого-небудь регістра порівняння встановлюється відповідний прапорець переривання OCFnA/OCFnB/OCFnC. Одночасно змінюється стан виходу блока порівняння OCnA/OCnB/OCnC. Поведінка цих виходів визначається вмістом розрядів COMnx1:COMxn0 регістра TCCRnA (табл. 5.6).

Таблиця 5.6. **Поведінка виводу OCnA/OCnB/OCnC у режимі Fast PWM**

COMnx1	COMnx0	Опис
0	0	Таймер/лічильник Tn відключено від виводу OCnx
0	1	WGMn3 = 0: таймер/лічильник Tn відключено від виводу OCnA/OCnB/OCnC*; WGMn3 = 1: стан виводу OCnA змінюється на протилежний
1	0	Скидається в нуль при рівності лічильного регістра і відповідного регістра порівняння. Встановлюється в одиницю у разі досягнення лічильником максимального значення (неінвертований ШІМ-сигнал)
1	1	Встановлюється в одиницю при рівності лічильного регістра і відповідного регістра порівняння. Скидається в нуль у разі досягнення лічильником максимального значення (інвертований ШІМ-сигнал)
* Також для моделей ATmega161x/163x/323x.		

Примітка. n = 1, 3, 4 або 5; x = A, B або C.

Часові діаграми для випадку, коли модуль лічби визначається вмістом регістра ICRn, показано на рис. 5.6.

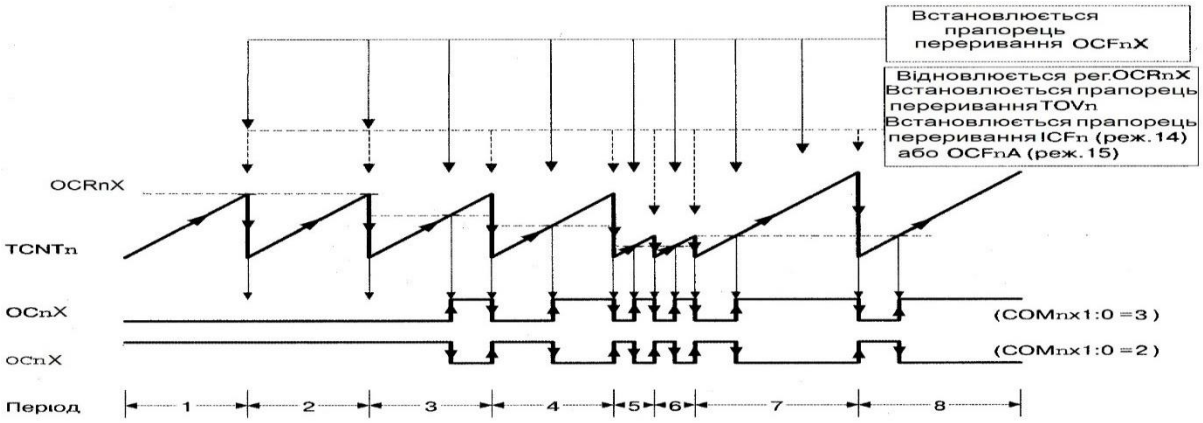


Рис. 5.6. Формування ШІМ-сигналу в режимі Fast PWM

Під час роботи з фіксованим значенням модуля лічильника для задання модуля лічби рекомендовано використовувати регістр захоплення ICRn, при цьому регістр OCRnA може бути використано для формування ШІМ-сигналу. Якщо у процесі формування ШІМ-сигналу його частота змінюється дуже часто, для задання модуля лічби лічильника рекомендовано використовувати регістр порівняння OCRnA. В цьому випадку завдяки буферизації запису в регістр порівняння виключається можливість появи несиметричних імпульсів сигналу на виході модулятора [3; 4].

Якщо вміст регістра порівняння дорівнює модулю лічби, то вихід відповідного блока порівняння переключиться у стійкий стан, обумовлений станом розрядів COMnx1:COMnx0 (див. табл. 5.6).

Частота сигналу, тривалість імпульсу та період сигналу, які генеруються, визначаються виразами:

$$f_{OCnx} = f_{clkI/0}/N*(TOP+1), \quad (5.5)$$

$$t_{імп} = T_{clkI/0} *N*(1 + OCRnx), \text{ неінвертований ШІМ-сигнал} \quad (5.5)$$

$$t_{імп} = T_{clkI/0} - (T_{clkI/0} *N*(1 + OCRnx)), \text{ інвертований ШІМ-сигнал} \quad (5.6)$$

$$T_{OCnx}=T_{clki/0}*N*(TOP+1), \quad (5.7)$$

де N – коефіцієнт ділення попереднього дільника; TOP – модуль лічби; OCRnx – вміст регістра OCRnx; $f_{clkI/0}$, $T_{clkI/0}$ – відповідно частота та період генератора тактових імпульсів підсистеми введення/виведення; n = 1, 3, 4 або 5; x = A, B або C.

За необхідності блок порівняння «А» у цьому режимі може також використовуватися для генерації сигналу «Меандр». Для цього необхідно записати до розрядів COMnA1:COMnA0 значення «01» та встановити WGMn3 = 1, що задає перемикання стану виходу OCnA у разі виникнення

події «Збіг».

5.4.4.5. Режим «ШІМ з корекцією фази»

Режим «ШІМ з корекцією фази» (англ. Phase Corect PWM) як і режим Fast PWM призначено для генерації сигналів із широтно-імпульсною модуляцією. Однак у цьому режимі лічильний регістр функціонує як реверсивний лічильник, стан якого спочатку змінюється від $0000 до максимального значення, а потім назад до $0000. Відповідно максимальна частота сигналу у цьому режимі в два рази менше максимальної частоти сигналу у режимі Fast PWM.

Максимальне значення лічильника (роздільна здатність модулятора ШІМ-сигналу) є або фіксованим значенням, або визначається вмістом визначених розрядів керуючих регістрів таймера/лічильника (табл. 5.7).

Таблиця 5.7. **Роздільна здатність модулятора в режимі Phase Correct PWM**

Номер режиму	WGMn3	WGMn2 (CTC1)*	WGMn1 (PWM11)*	WGMn0 (PWM10)*	Роздільна здатність	Модуль лічби (TOP)
1	0	0	0	1	8 розрядів	$00FF
2	0	0	1	0	9 розрядів	$01FF
3	0	0	1	1	10 розрядів	$03FF
10**	1	0	1	0	Змінна (2…16)	ICRn ($0003...$FFFF)
11**	1	0	1	1	Змінна (2...16)	OCRnA ($0003...$FFFF)
* В моделях ATmega161x/163x/323x. ** В моделях ATmega161x/163x/323x ці режими відсутні.						

Примітка. n = 1, 3, 4 або 5.

Роздільна здатність R визначається виразом R = log(TOP + 1)/log2, де TOP – модуль лічби.

Як і в режимі Fast PWM, під час роботи з якими-небудь фіксованими значеннями модуля лічби, що змінюються інколи, для його задання рекомендовано використовувати регістр захоплення, при цьому регістр OCRnA може використовуватися для формування ШІМ-сигналу.

Якщо ж у процесі формування ШІМ-сигналу його частота змінюється дуже часто, для задання модуля лічби рекомендовано використовувати регістр порівняння OCRnA [3].

У разі досягнення лічильником максимального значення відбувається зміна напрямку лічби, але лічильник залишається в цьому стані протягом одного періоду сигналу f_{clkTn}. У цьому ж такті відбувається відновлення вмісту

регістра порівняння.

Якщо модуль лічби визначається регістром ICRn (режим 10) або OCRnA (режим 11), одночасно з відновленням регістра порівняння встановлюється прапорець ICFn або OCFn відповідно. У разі досягнення лічильником мінімального значення ($0000) також відбувається зміна напрямку лічби та одночасно встановлюється прапорець переривання TOVn. При рівності вмісту лічильного регістра і якого-небудь регістра порівняння встановлюється відповідний прапорець OCFnA/OCFnB/OCFnC регістра TIFR. Одночасно змінюється стан виходу блока порівняння OCnA/OCnB/OCnC. Поведінка виходу визначається вмістом розрядів COMnx1:COMnx0 регістрів TCCRnA (табл. 5.8).

Часові діаграми для випадку, коли модуль лічби визначається вмістом регістра ICRn або OCRnA, показано на рис. 5.7.

Таблиця 5.8. **Поведінка виводу OCnA/OCnB/OCnC у режимі Phase Correct PWM**

COMnx1	COMnx0	Опис
0	0	Таймер/лічильник Tn відключено від виводу OCnA/OCnB/OCnC
0	1	WGMn3 = 0: таймер/лічильник Tn відключено від виводу OCnA/OCnB/OCnC*; WGMn3 = 1: стан виводу OCnA змінюється на протилежний
1	0	Скидається в «0» під час прямої лічби і встановлюється в «1» у разі зворотної лічби не інвертований ШІМ-сигнал)
1	1	Встановлюється в «1» під час прямої лічби і скидається в «0» у разі зворотної лічби (інвертований ШІМ-сигнал)
* Також для моделей ATmega161x/163x/323x.		

Примітка. n = 1, 3, 4 або 5; x= A, B або C.

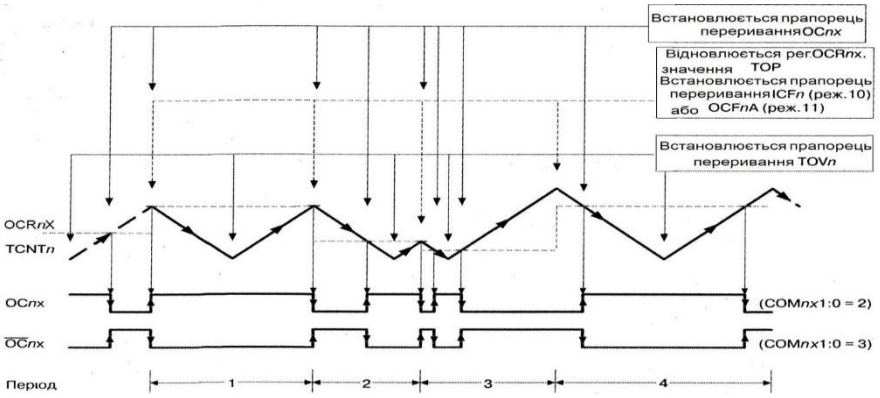


Рис. 5.7. Формування ШІМ-сигналу в режимі Phase Correct PWM

У разі зміни модуля лічби під час роботи таймера/лічильника на виході блоків порівняння можуть з’явитися імпульси, несиметричні відносно середини періоду вихідного сигналу.

Оскільки відновлення вмісту регістра порівняння відбувається в момент досягнення лічильником максимального значення, період ШІМ-сигналу дорівнює часу між цими моментами, при цьому час прямої лічби визначається попереднім значенням модуля лічби, а час зворотної лічби – новим значенням. Якщо ці значення різні, то час прямої і зворотної лічби також відрізняються. Результатом цього і є несиметричні імпульси на виході блоків порівняння, як показано на рис. 5.7 (3-й період сигналу).

У разі частої зміни модуля лічби під час роботи таймера/лічильника рекомендовано використовувати режим Phase and Frequency Correct PWM, який розглянуто нижче. Якщо ж використовується постійне значення модуля лічби, між цими двома режимами немає ніякої різниці.

Якщо значення, що перебуває у регістрі порівняння, дорівнює $0000 або модулю лічби (TOP), то під час наступного збігу стану лічильника і вмісту регістра порівняння вихід схеми порівняння переключиться в стійкий стан відповідно до табл. 5.9.

Таблиця 5.9. **Стійкі стани виходу схеми порівняння**

COMnx1	COMnx0	Регістр OCRnx	Стан виводу OCnx
1	0	$0000	0
1	0	TOP	1
1	1	$0000	1
1	1	TOP	0

Примітка. n = 1, 3, 4 або 5; x = A, B або C.

Частота, тривалість імпульсу та період сигналу, що генерується, визначаються виразами:

$$f_{OCnx} = f_{clkI/0}/2*N*(TOP+1), \quad (5.8)$$

$$t_{імп}= (T_{clkI/0} *2*N*(1 + OCRnx)), \text{ неінвертований ШІМ-сигнал} \quad (5.9)$$

$$t_{імп}= T_{clkI/0} - (T_{clkI/0} *2*N*(1 + OCRnx)), \text{ інвертований ШІМ-сигнал} \quad (5.9)$$

$$T_{OCnx}=T_{clki/0}*2*N*(TOP+1), \quad (5.10)$$

де *N* – коефіцієнт ділення попереднього дільника; TOP – модуль лічби; OCRnx – вміст регістра OCRnx; $f_{clkI/0}$, $T_{clkI/0}$ – відповідно частота та період генератора тактових імпульсів підсистеми введення/виведення; n = 1, 3, 4 або 5; x = A, B або C.

5.4.4.6. Режим «ШІМ з корекцією фази та частоти»

Режим «ШІМ з корекцією фази та частоти» (англ. Phase and Frequency Correct PWM) дуже схожий на режим Phase Corect PWM. Єдина принципова різниця між ними – момент відновлення вмісту регістра порівняння (рис. 5.8).

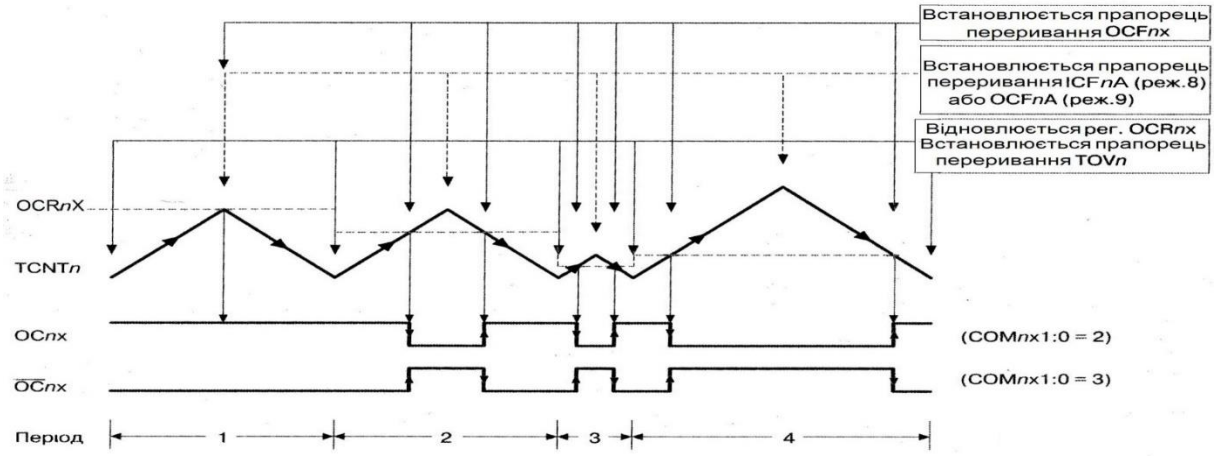


Рис. 5.8. Формування ШІМ-сигналу в режимі Phase and Frequency Correct PWM

Відновлення вмісту регістра порівняння відбувається в момент досягнення лічильником мінімального значення, тому кожен період вихідного сигналу є повністю симетричним. Час прямої лічби завжди дорівнює часу зворотної лічби. Вихідні імпульси симетричні відносно періоду і, відповідно, частота сигналу, що генерується, залишається сталою.

Максимальне значення лічильника (роздільна здатність ШІМ-сигналу) у цьому режимі може визначатися тільки регістрами ICRn або OCRnA таймера/лічильника Tn, як показано у табл. 5.2.

Роздільна здатність визначається виразом R = log(TOP + 1)/log2, де TOP – модуль лічби (табл. 5.10).

Таблиця 5.10. **Роздільна здатність модулятора в режимі Phase and Frequency Correct PWM**

Номер режиму	WGMn3	WGMn2	WGMn1	WGMn0	Роздільна здатність	Модуль лічби (TOP)
8	1	0	1	0	Змінна (2...16)	ICRnA ($0003...$FFFF)
9	1	0	1	1	Змінна (2...16)	OCRnA ($0003...$FFFF)

Примітка. n = 1, 3, 4 або 5.

Як і в інших режимах, під час роботи з фіксованими значеннями модуля лічби, що змінюються рідко, для його задання рекомендовано використовувати регістр захоплення ICRn, при цьому регістр OCRnA може використовуватися для формування ШІМ-сигналу. Якщо ж у процесі формування ШІМ-сигналу його

частота змінюється дуже часто, для модуля лічби рекомендовано використовувати регістр порівняння OCRnA.

У разі досягнення лічильником максимального значення відбувається зміна напрямку лічби, але лічильник залишається в цьому стані протягом одного періоду сигналу f_{clkTn}. У цьому ж такті встановлюється прапорець ICFn або OCFnA (залежить від того, який з регістрів використовується для задання модуля лічби). У разі досягнення лічильником мінімального значення ($0000) напрямок лічби знову змінюється, при цьому встановлюється прапорець переривання TOVn відповідного регістра і відбувається відновлення вмісту регістра порівняння.

Під час рівності вмісту лічильного регістра і якого-небудь регістра порівняння встановлюється відповідний прапорець OCFnA/OCFnB/OCFnC. Одночасно змінюється стан виходу блока порівняння OCnA/OCnB/OCnC. Поведінка виходу визначається вмістом розрядів COMnx1:COMnx0 регістрів TCCRnA (табл. 5.11).

Таблиця 5.11. **Поведінка виводу OCnA/OCnB/OCnC в режимі Phase and Frequency Correct PWM**

COMnx1	COMnx0	Опис
0	0	Таймер/лічильник Tn відключено від виводу OCnA/OCnB/OCnC
0	1	WGMn3 = 0: таймер/лічильник Tn відключено від виводу OCnA/OCnB/OCnC; WGMn3 = 1: стан виводу OCnA змінюється на протилежний
1	0	Скидається в «0» під час прямої лічби і встановлюється в «1» у разі зворотної лічби не інвертований ШІМ-сигнал)
1	1	Встановлюється в «1» під час прямої лічби і скидається в «0» у разі зворотної лічби (інвертований ШІМ-сигнал)

Примітка. n = 1, 3, 4 або 5; x= A, B або C.

Частота, тривалість імпульсу та період сигналу, що генерується, визначаються виразами:

$$f_{OCnx} = f_{clkI/0}/2*N*(TOP+1), \quad (5.11)$$

$$t_{імп}= (T_{clkI/0}*2*N*(1 + OCRnx)), \text{ неінвертований ШІМ-сигнал} \quad (5.12)$$

$$t_{імп}= T_{clkI/0} - (T_{clkI/0}*2*N*(1 + OCRnx)), \text{ інвертований ШІМ-сигнал} \quad (5.13)$$

$$T_{OCnx} = T_{clkI/0}*2*N*(TOP+1), \quad (5.14)$$

де N – коефіцієнт ділення попереднього дільника; TOP – модуль лічби; OCRnx – вміст регістра OCRnx; $f_{clkI/0}$, $T_{clkI/0}$ – відповідно частота та період генератора тактових імпульсів підсистеми введення/виведення; n = 1, 3, 4 або 5; x = A, B або C.

Якщо значення, що перебуває в регістрі порівняння, дорівнює $0000 або модулю лічби (TOP), то під час наступного збігу стану лічильника і вмісту регістра порівняння вихід схеми порівняння переключиться у стійкий стан відповідно до табл. 5.11.

5.4.5. Вартовий таймер

Усі AVR-мікроконтролери мають вартовий таймер, основна функція якого – захист пристрою від збоїв. Завдяки вартовому таймеру можна перервати виконання програми, що зациклилася або вийти з інших непередбачених ситуацій, що перешкоджають нормальному виконанню програми.

У МК сімейства трапляються вартові таймери у двох виконаннях: стандартному та розширеному [3; 4]. Структурну схему вартового таймера у стандартному режимі наведено на рис. 5.9.

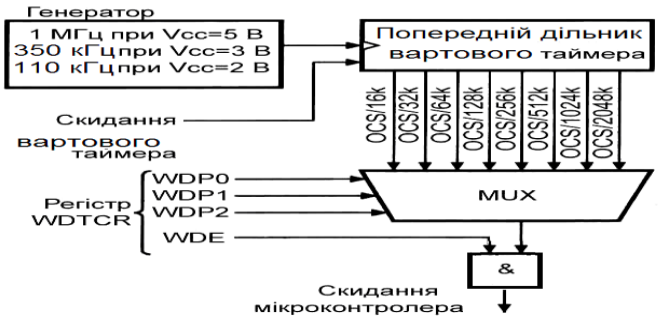


Рис. 5.9. Структурна схема вартового таймера

Вартовий таймер має незалежний тактовий генератор і працює навіть у сплячому режимі Power Down. Частота цього генератора залежить від напруги живлення пристрою, температури та технологічного розкиду [3; 4].

5.5. Моделювання модуля таймера AVR-мікроконтролера, що керує двигуном постійного струму

5.5.1. Опис моделі

Для керування двигуном постійного струму може використовуватись таймер AVR-мікроконтролера, який формує ШІМ-сигнал.

На рис. 5.10 наведено робочу модель пристрою керування двигуном постійного струму у пакеті PROTEUS 8.6.

З лівого боку моделі наведено кнопки керування: K1, K2, K3, K4, на які через резистори R1, R2, R3, R4 відповідно, подається напруга живлення.

У правому нижньому куті зображено двигун постійного струму (DC Motor), а трохи лівіше – мікросхему L293D, через яку МК керує двигуном за допомогою ШІМ-сигналу. В якості МК обрано мікросхему ATmega128, яка знаходиться у правому куті.

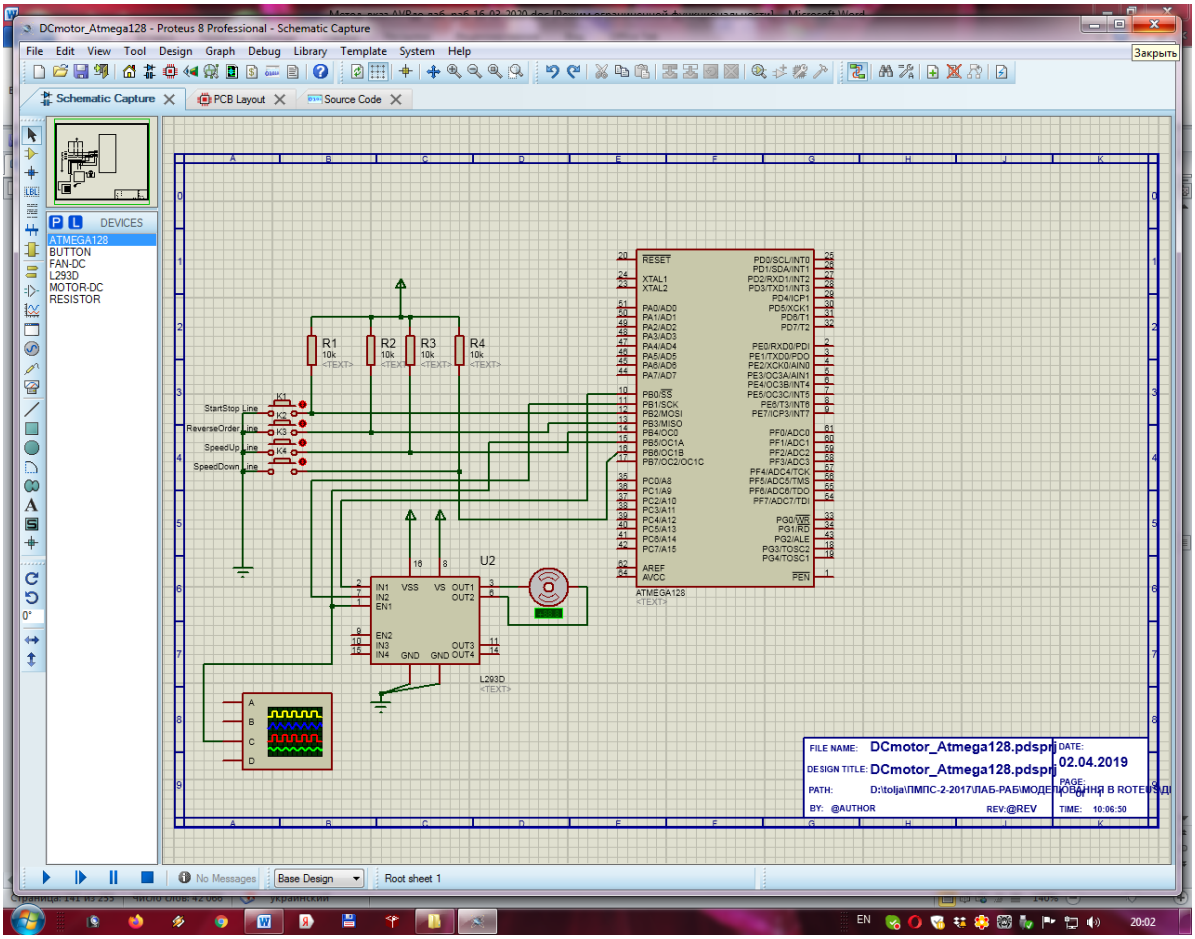


Рис. 5.10. Схема моделі пристрою керування двигуном постійного струму

Зовнішні резистори використовуються через те, що на вхідних лініях МК внутрішні підтягуючі резистори програмно не підключені.

У вихідному стані кнопки є нормально розімкненими. В цьому випадку на відповідні лінії порту В МК (PB) подаються високі рівні напруги, тобто логічні одиниці. Коли кнопки замикаються, на входи МК подаються низькі рівні напруги, тобто логічні нулі. Перемикання сигналу з логічного нуля у логічну одиницю від обраної кнопки сприймається МК як відповідний сигнал керування.

На рис. 5.11 наведено збільшений вигляд кнопок керування.

Кнопка K1 (Start Stop Line) відповідає за вмикання та вимикання двигуна. Тобто коли двигун вимкнено, то вона його ввімкне. Коли двигун увімкнений та обертається, то вона його вимкне. Кнопку K1 підключено до 12-го виводу МК – PB2, який має бути запрограмований як вхід.

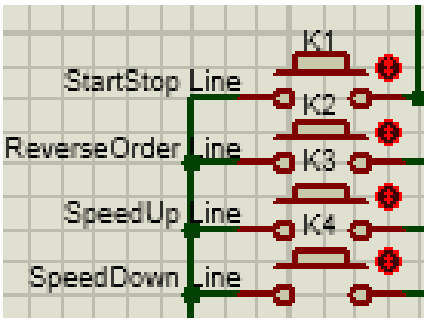


Рис. 5.11. Порядок розміщення елементів керування

Кнопка K2 (Reverse Order Line) відповідає за зміну напрямку обертання двигуна. Одне натискання – одна зміна напрямку. Потрібно мати на увазі, що двигун – пристрій інерційний, і він не може зупинитися миттєво і одразу почати обертатися в інший бік. Після зміни напрямку програмно, у нього витрачається певний період часу, щоб фізично зупинитись, та почати обертатись в інший бік. Кнопку K2 підключено до 13-го виводу МК – PB3, який має бути запрограмований як вхід.

Кнопка K3 (Speed Up Line) відповідає за збільшення швидкості обертання двигуна за допомогою поступового зменшення шпаруватості та збільшення постійної еквівалентної напруги імпульсного ШІМ-сигналу до максимуму, який дорівнює амплітуді імпульсу – значенню напруги живлення на виводі VS мікросхеми L293D (рис. 5.12).

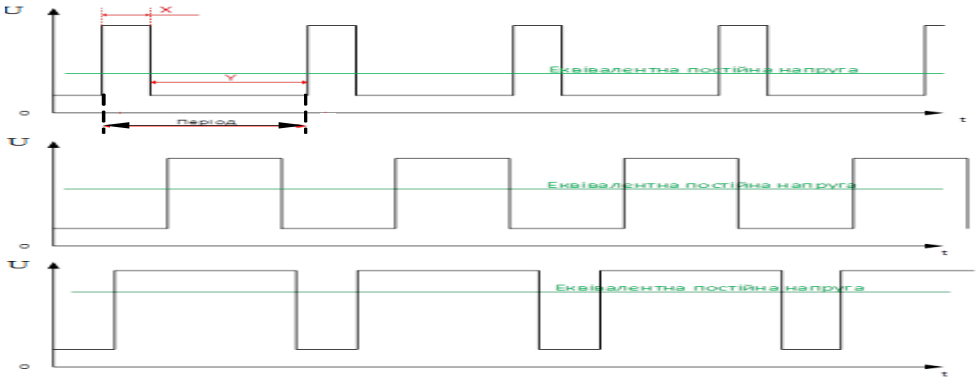


Рис. 5.12. Еквівалентна постійна напруга ШІМ-сигналу

Оскільки двигун інерційний, то швидкість обертання збільшується не стрибкоподібно, а поступово, і потребує певного проміжку часу, щоб вийти на новий програмно встановлений рівень.

Кнопку K3 підключено до 14-го виводу МК – PB4, який має бути запрограмований як вхід.

Кнопка K4 (Speed Down Line) відповідає за зменшення швидкості обертання двигуна, для чого вона поступово збільшує шпаруватість та зменшує значення постійної еквівалентної напруги імпульсного ШІМ-сигналу.

Через інерційність двигуна швидкість обертання зменшується не стрибками, а поступово і вимагає для свого зменшення певного проміжку часу. Кнопку K4 підключено до 16-го виводу МК – PB6, який має бути запрограмований як вхід.

На рис. 5.13 наведено підключення до МК мікросхеми L293D.

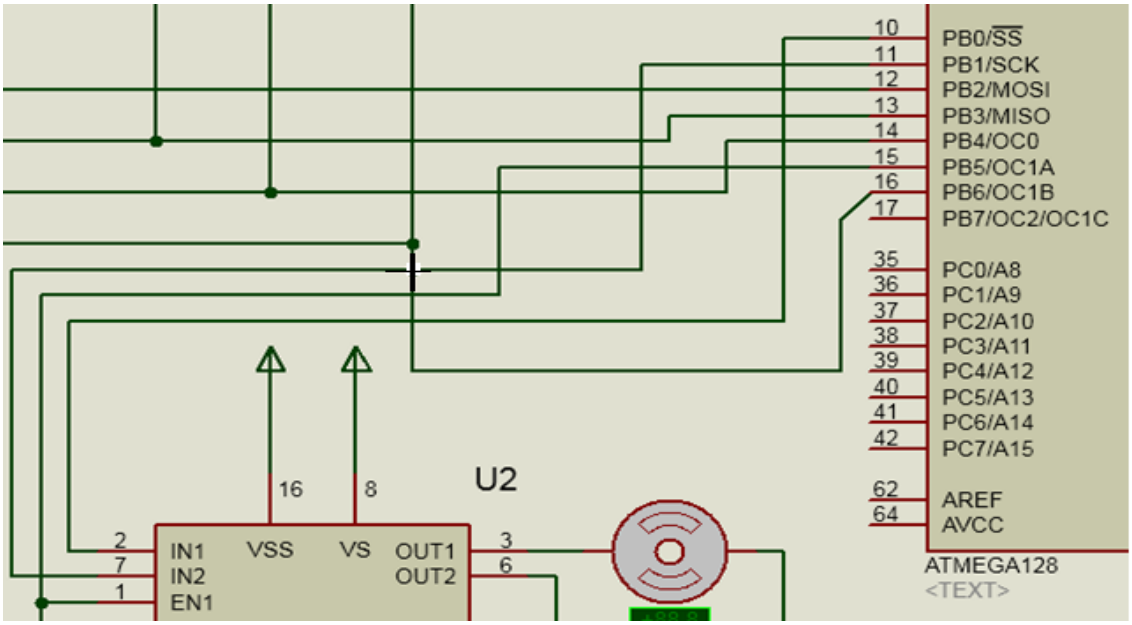


Рис. 5.13. Підключення до МК мікросхеми L293D

Виводи 10 та 11 МК (PB0 та PB1) запрограмовані як виходи і підключені, відповідно, до входів IN1 та IN2 мікросхеми L293D (U2). Залежно від комбінації сигналів керування на цих входах, двигун буде обертатись у відповідному напрямку.

15-й вивід МК – PB5 запрограмовано як вихід і підключено до входу EN1 мікросхеми L293D. З цього виходу МК знімається ШІМ-сигнал. Вхід EN1 відповідає за роботу першої мостової схеми у мікросхемі (див. п. 5.5.4). Коли на ньому високий рівень, на перший міст подається живлення, яке підведено до виводу VS мікросхеми. Коли на вході EN1 низький рівень, на перший міст напруга живлення не подається. Завдяки сигналу на цьому вході можна змінювати швидкість обертання двигуна.

Два входи мікросхеми L293D – GND заземлено. На вхід VSS подається напруга живлення самої мікросхеми, а на вхід VS – напруга живлення двигуна.

Підключення двигуна до мікросхеми L293D зображено на рис. 5.14.

Двигун підключено до *3* та *6* виводів мікросхеми L293D, а саме OUT1 та OUT2. На ці виходи, залежно від того, чи ввімкнене живлення першої мостової схеми, що керується входом EN1, та залежно від стану ключів у першій мостовій

схемі, які керуються входами IN1 та IN2, або подається напруга живлення, яку подано на вхід VS, або напругу не подано.

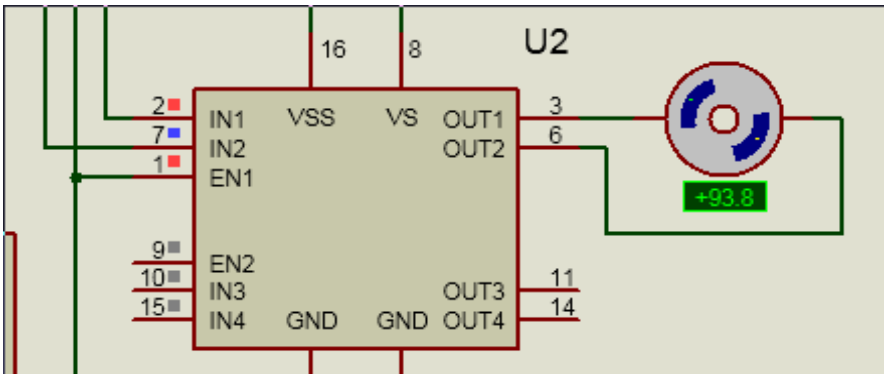


Рис. 5.14. Підключення двигуна постійного струму до мікросхеми L293D

У випадку, коли на один з виводів – OUT1 чи OUT2 подається напруга живлення, а на інший не подається, виникне різниця потенціалів. Це викличе проходження струму обмоткою збудження двигуна, внаслідок чого він почне обертатися.

На рис. 5.15…5.19 наведено деякі фрагменти, які демонструють роботу моделі.

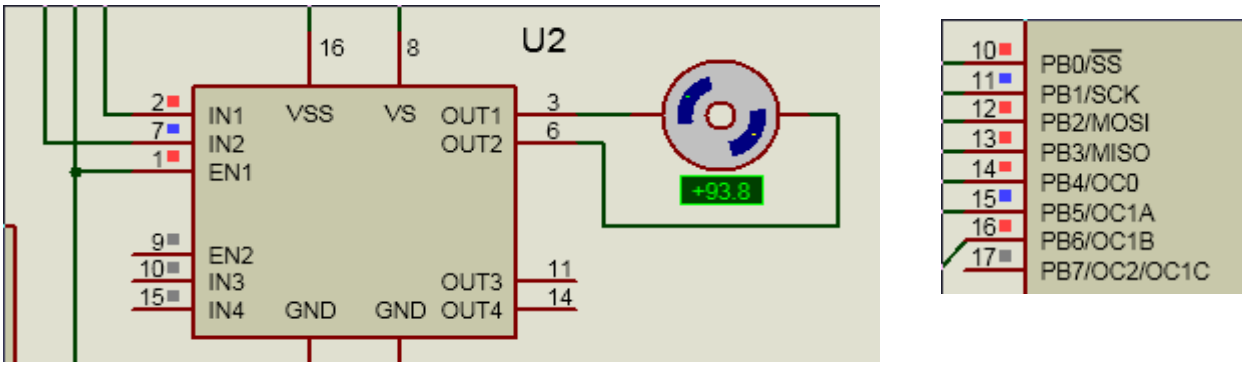


Рис. 5.15. Стан моделі після натискання кнопки «старт/стоп»

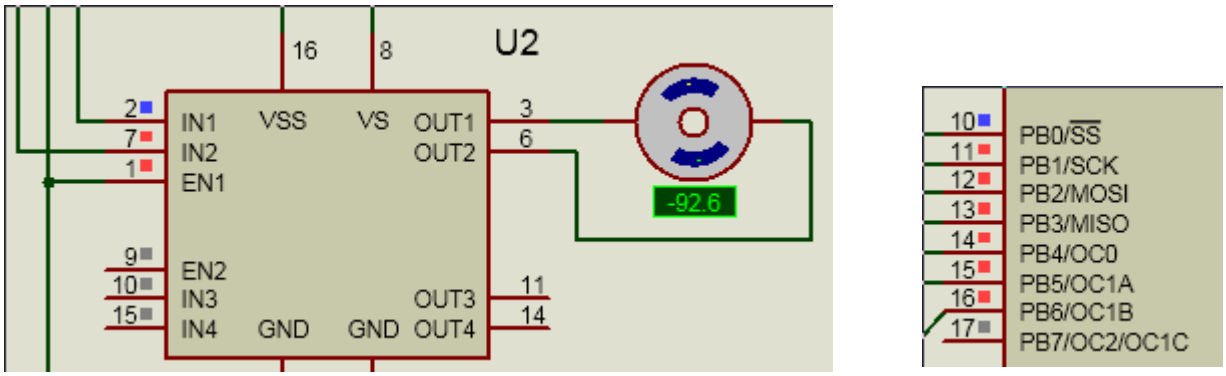


Рис. 5.16. Стан моделі після активації режиму реверсу

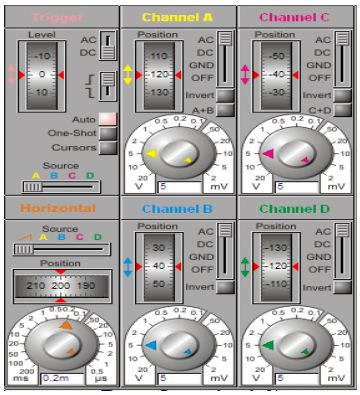


Рис. 5.17. Налаштування осцилографа

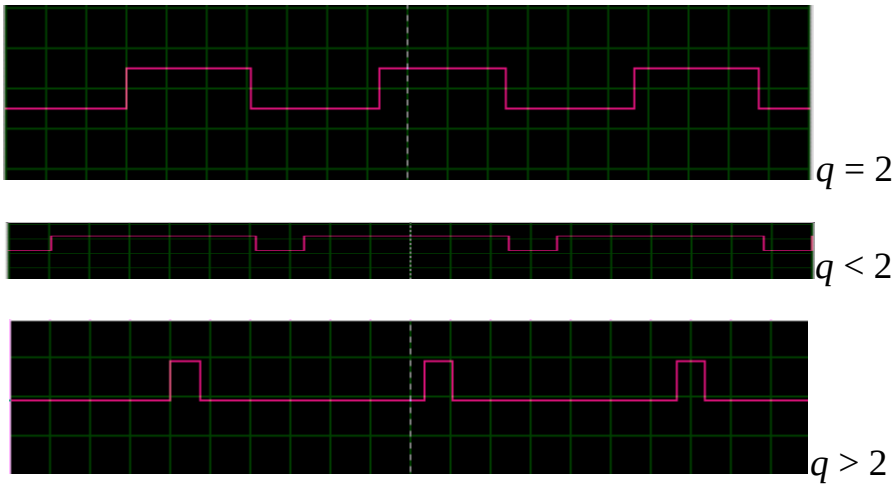


Рис. 5.18. ШІМ-сигнал, що подається на двигун (шпаруватість $q = 2$, $q < 2$, $q > 2$)

5.5.2. Мікроконтролер

Для контролю та виконання всіх функцій, які покладено на модель, було обрано AVR-мікроконтролер ATmega 128. Він є малопотужним 8-розрядним КМОН-мікроконтролером, який має розширену RISC-архітектуру [3]. Швидкодія МК досягає 1 мільйона операцій у секунду, оскільки більшість команд виконується за один машинний такт.

ATmega 128 має 64 виводи типу вхід–вихід. У МК вбудовано Flash-пам’ять програм, яку можна програмувати та перепрограмувати.

На рис. 5.20 наведено зовнішній вигляд та позначення виводів МК. Його основні характеристики наведено у [3].

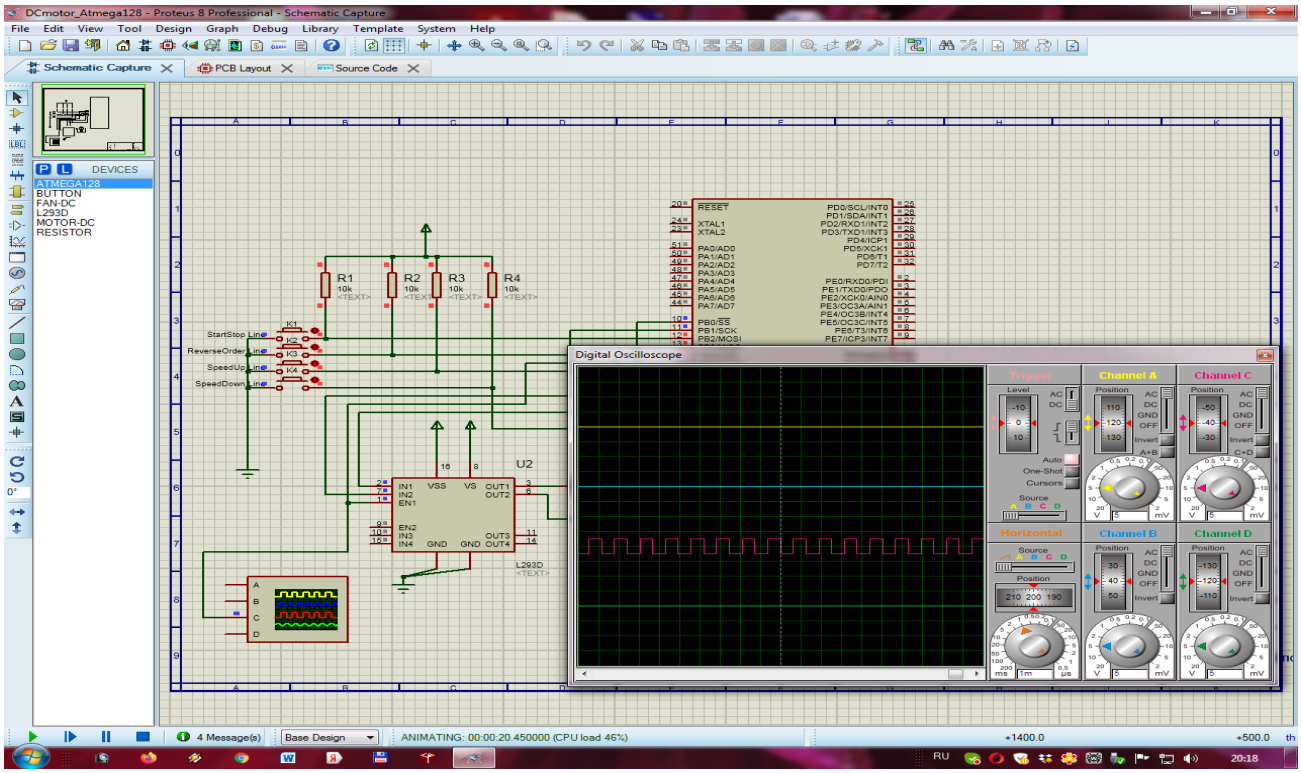


Рис. 5.19. Стан моделі після запуску сценарію

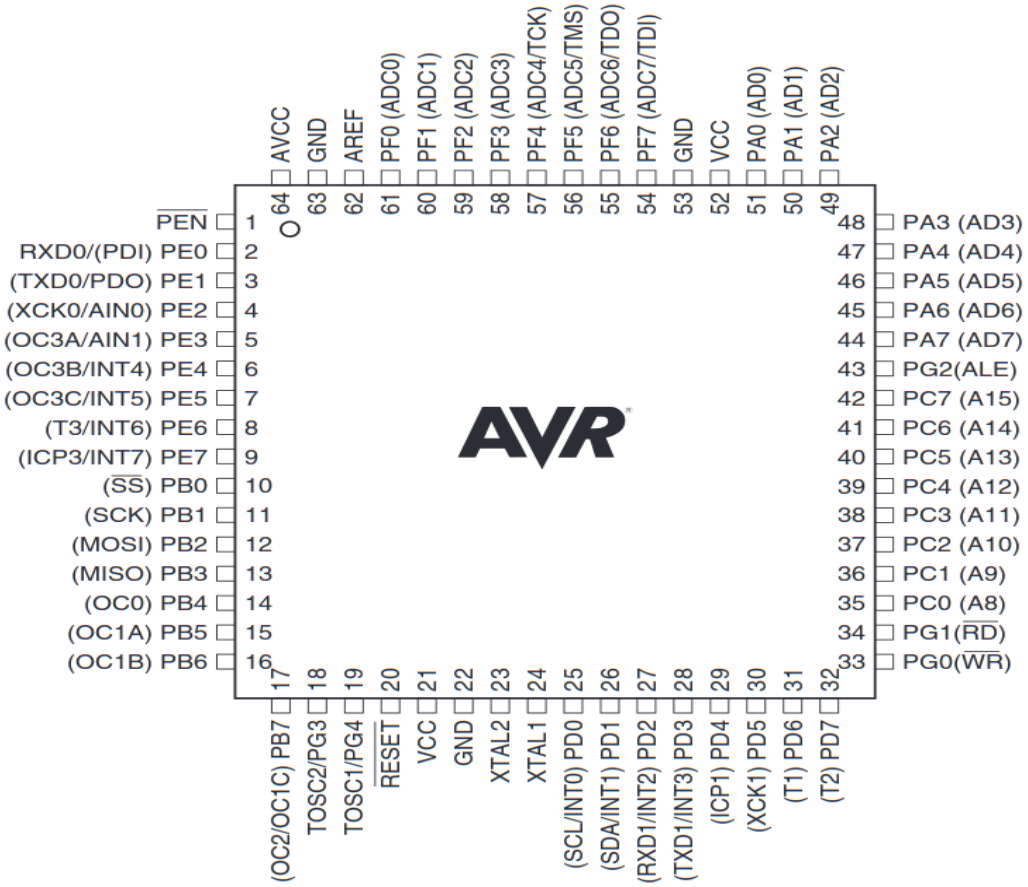


Рис. 5.20. Зовнішній вигляд та позначення виводів МК ATmega128

5.5.3. Модуль таймера

Для керування двигуном постійного струму в моделі використовується модуль таймера МК, який працює в режимі Phase and Frequency Correct PWM.

Регулювання швидкості обертання двигуна забезпечується зміною шпаруватості керуючих імпульсів, що у свою чергу викликає зміну постійної складової імпульсного сигналу. Чим менше шпаруватість, а відповідно більше величина відношення «тривалість імпульсу/період», тим більша напруга надходить на двигун, який буде обертатись із більшою швидкістю.

5.5.4. Драйвер двигуна

Для керування двигуном постійного струму між ним та МК використано мостову схему – драйвер ШІМ, яку зображено на рис. 5.21.

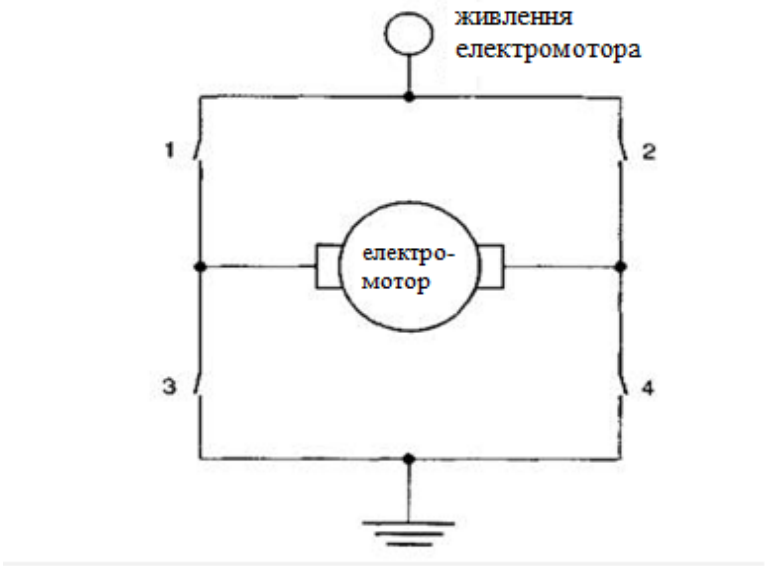


Рис. 5.21. Мостова схема для підключення двигуна до МК

Подібну схему мають драйвери двигунів, які використовуються для перетворення керуючих сигналів малої потужності від МК у сигнали, потужність яких достатня для керування двигунами.

Є декілька схем драйверів, які відрізняються як потужністю, так і елементною базою, на якій вони виконані. В моделі застосовано драйвер, який виконано у вигляді готової до роботи мікросхеми – L293D [2; 3].

Мікросхема має два канали для керування двома електродвигунами досить великої потужності. В мікросхемі є дві пари входів для сигналів керування і дві пари виходів для підключення електродвигунів (рис. 5.22).

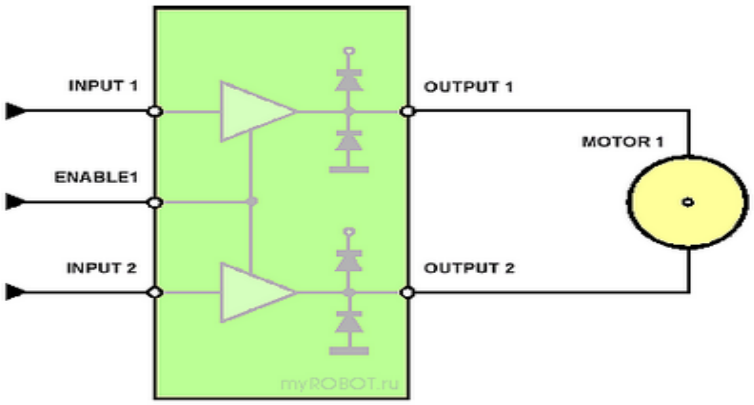


Рис. 5.22. Спрощений вигляд першого каналу мікросхеми L293D

Крім того, у L293D є два входи для вмикання кожного з драйверів.

Ці входи використовуються для керування швидкістю обертання електродвигунів за допомогою широтно-імпульсної модуляції сигналів. Окрім цього, L293D забезпечує розділене живлення для мікросхеми і для керованих нею двигунів, що дозволяє підключати електродвигуни з більшою напругою живлення, ніж у мікросхеми. Останнє корисне також для зменшення завад, які можуть викликатися стрибками напруги під час роботи двигунів.

Обидва канали, що входять у склад мікросхеми, мають ідентичні принципи роботи, тому розглянемо лише один з них.

До виходів OUTPUT1 та OUTPUT2 підключається двигун постійного струму (MOTOR1).

На вхід ENABLE1 подається імпульсний ШІМ-сигнал, який формується МК.

Змінюючи шпаруватість цього сигналу змінюємо постійну складову імпульсного сигналу, що в свою чергу змінює швидкість обертання двигуна.

Якщо у цьому разі на входи INPUT1 та INPUT2 не подаються відповідні керуючі сигнали, то двигун обертатися не буде.

Якщо вхід INPUT1 з'єднати з додатним полюсом джерела живлення, а вхід INPUT2 – з від'ємним, то двигун почне обертатися за годинниковою стрілкою.

Якщо з'єднати вхід INPUT1 з від'ємним полюсом джерела струму, а вхід INPUT2 – з додатним, то двигун почне обертатися в інший бік.

Якщо подати сигнали одного рівня одразу на два керуючих входи INPUT1 та INPUT2, тобто під'єднати обидва входи до додатного полюсу джерела живлення або до від'ємного, то двигун обертатися не буде.

Якщо прибрати керуючий сигнал зі входу ENABLE1, то за будь-яких варіантів наявності сигналів на входах INPUT1 та INPUT2 двигун також обертатися не буде.

Зовнішній вигляд, нумерацію та позначення виводів мікросхеми наведено у [3].

Інший спосіб керування двигунами постійного струму основано на використанні мостових схем типу L298N [2]. Це двоканальний пристрій, який працює від транзисторно-транзисторної логіки з переходами Шоткі рівнів з напругою до 46 В та струмом до 2 А на кожен канал.

5.5.5. Захисні діоди

Як видно з наведеного вище рис. 5.22, схема кожного драйвера містить чотири захисні діоди. Ці діоди призначено для захисту транзисторних ключів у драйвері від додатних та від'ємних паразитних імпульсів достатньо високої амплітуди, які з'являються на обмотках двигуна під час їх комутації.

Додатні імпульси виникають у разі запирання (вимикання) ключів, а від'ємні – у разі включення. Механізм виникнення цих імпульсів описано у [2; 3].

5.5.6. Програмування таймера мовою Асемблера

Нижче наведено приклад розрахунку та програмування таймера МК мовою Асемблера.

Вхідні дані

Треба розрахувати та запрограмувати таймер на основі таких вхідних даних:

- тип МК: AT mega 128;
- номер таймера: T/C 1;
- на виході PB5 МК треба сформувати інвертований ШІМ-сигнал;
- частота ШІМ-сигналу: f_{PB5} = 25 кГц;
- частота тактового сигналу підсистеми введення/виведення МК: $f_{CLKI/0}$ = 16 МГц;
- режим роботи таймера: Phase and Frequency Correct PWM (PFCPWM);
- шпаруватість ШІМ-сигналу: Q_{PB5} = 2.

Завдання

Розрахувати:

- коефіцієнт ділення переддільника: $К_{діл}$ = N;
- модуль лічби: TOP;
- період ШІМ-сигналу: T_{PB5};
- тривалість імпульсу ШІМ-сигналу: $t_{імпPB5}$.

Написати мовою Асемблера фрагмент програми, який забезпечує формування ШІМ-сигналу згідно з вхідними даними.

Рішення завдання

Формування таймером ШІМ-сигналу у режимі Phase and Frequency Correct PWM наведено вище на рис. 5.8.

Згідно з наведеною табл. 5.2 обираємо восьмий режим роботи таймера. Величина частоти ШІМ-сигналу f_{PB5} визначається виразом (5.11). Тоді значення модуля лічби TOP визначається вмістом регістра ICR1.

За f_{PB5} 25 кГц та $К_{діл}$ = N = 8 шукаємо значення ICR1 = TOP:

$$TOP+1 = ICR1+1 = \frac{f_{CLK\,I/O}}{2 \cdot N \cdot f_{PB5}} = \frac{16 \cdot 10^6}{2 \cdot 8 \cdot 25 \cdot 10^3} = 40.$$

Тоді ICR1 = 39 =00100111B = $0027. (5.14)

Період ШІМ-сигналу:

$$T_{PB5} = \frac{1}{f_{PB5}} = \frac{1}{25000} = 40 \text{ мкс}.$$

Потрібна тривалість імпульсу під час шпаруватості ШІМ-сигналу Q_{PB5} = 2:

$$t_{імпPB5} = \frac{T_{PB5}}{Q} = \frac{40 \text{ мкс}}{2} = 20 \text{ мкс}.$$

Згідно з рис. 5.8

$$t_{імпPB5} = T_{PB5} - 2\,\Delta\,t. \quad (5.15)$$

Із виразу (5.15) $\Delta\,t = \frac{T_{PB5} - t_{імпPB5}}{2} = \frac{40-20}{2} = 10$ мкс.

В свою чергу, згідно з роботою таймера в режимі Phase and Frequency Correct PWM

$$\Delta\,t = T_{CLKI/O} * N * OCR1A, \quad (5.16)$$

де OCR1A – регістр порівняння каналу А таймера 1.

Із виразу (5.16)

$$OCR1A = \frac{\Delta\,t}{T_{CLK\,I/O} \cdot N} = \frac{10 \cdot 10^6}{\frac{1}{16} \cdot 10^{-6} \cdot 8} = 20 = 00010100B = \$0014.$$

Тоді шпаруватість

$$Q = \frac{T_{PB5}}{T_{PB5} - 2\,\Delta\,t} = \frac{T_{PB5}}{T_{PB5} - 2\,T_{CLKI/O} * N * OCR1A} = \frac{1}{1 - \frac{2\,T_{CLKI/O} * N * OCR1A}{T_{PB5}}} =$$

$$= \frac{1}{1 - 2 * \frac{1}{f_{CLKI/O}} * N * OCR1A * f_{PB5}}. \quad (5.17)$$

Згідно з розрахунком, який виконано вище, за OCR1A = 20 шпаруватість Q повинна дорівнювати 2. Розрахуємо шпаруватість за виразом (5.17).

$$Q = \frac{1}{1 - 2 \cdot \frac{1}{f_{CLKI/O}} \cdot N \cdot OCR1A \cdot f_{PB5}} = \frac{1}{1 - 2 \cdot \frac{1}{16 \cdot \cdot 10^6} \cdot 8 \cdot 20 \cdot 25 \cdot 10^3} = 2.$$

Отримано значення Q = 2. Це означає, що проведені вище розрахунки виконано правильно.

Обчислимо значення Q у разі збільшення OCR1A на 5 та у разі зменшення на 5

$$Q_{(OCR1A = 25)} = \frac{1}{1-2\cdot\frac{1}{16\cdot 10^6}\cdot 8\cdot 25\cdot 25\cdot 10^3} = 2{,}667.$$

$$Q_{(OCR1A = 15)} = \frac{1}{1-2\cdot\frac{1}{16\cdot 10^6}\cdot 8\cdot 15\cdot 25\cdot 10^3} = 1{,}6.$$

Розробка окремих фрагментів програми мовою Асемблера

Зупинка таймера

Для зупинки таймера треба згідно з форматом регістра TCCR1B [3], адреса якого дорівнює $004E, записати в нього наведене нижче керуюче слово КС1 = 00000000B = $00:

7 р.	6 р.	5 р.	4 р.	3 р.	2 р.	1 р.	0 р.	КС1
ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS00	
0	0	0	0	0	0	0	0 ($00 = КС1)	

Тоді програма має такий вигляд:

LDI R18, $00; R18← КС1 = $00;
LDI R27, $00; R27 ← $00
LDI R26, $4E; R26 ← $4E } X(R27, R26) ← $004E;
ST X, R18; TCCR1B ← R18 = $00, зупинка таймера.

Завантаження регістра TCCR1A

Для задачі програмування режиму роботи 8 та формування інвертованого ШІМ-сигналу згідно табл. 5.2, 5.11 необхідно встановити біти регістра TCCR1A:COM1A1 = COM1A0 = 1, WGM11 = 0; WGM10 = 0. Інші біти цього регістра у нашому прикладі не використовуються, тому запишемо у них нулі. Тоді згідно з форматом регістра TCCR1A [3] керувальне слово КС2 = 11000000B = $C0:

7 р.	6 р.	5 р.	4 р.	3 р.	2 р.	1 р.	0 р.	КС2
COM1A1	COM1A0	COB1B1	COM1BO	COM1C1	COM1C0	WGM11	WGM10	
1	1	0	0	0	0	0	0 ($C0 = КС2)	

Адреса регістра TCCR1A = $004F [3].

Тоді програма має такий вигляд:

LDI R17, $C0; R17← КС2 = $C0;
LDI R29, $00; R27 ← $00
LDI R28, $4F; $26 ← $4F } Y(R29, R28) ← $004F;
ST Y, R17; TCCR5A←R17=$C0.

Програмування лінії PB5 на виведення

Для програмування лінії PB5 на виведення необхідно встановити п’ятий розряд регістра DDRB в одиницю [3]. Адреса цього регістра – $0037.

Тоді програма має вигляд:

```
LDI R31, $00;                  } Z(R31, R30) ← $0037
LDI R30,  $37;  R30 ← $37 ;
LD R19,  Z;  R19← DDRB
ORI R19, $20; R19←00100000B + R19; R19.5←1, інші біти без змін;
ST Z, R19;DDRB←R19; DDRB. 5 ←1.
```

Завантаження регістра ICR1

16-розрядний регістр ICR1 має адресу – $0047 (старший байт):$0046 (молодший байт) [3]. В ICR1 треба завантажити – 39 = 00100111B = $0027 (див. вище).

Тоді програма має вигляд:

```
LDI R23, $00; R23← $00;
LDI R27,  $00; R27 ← $00;
LDI R26, $47; R26 ← $47     } X(R27, R26)   ← $0047;
ST X, R23; СБ ICR1←R23 = $00 .
LDI R24, $28; R24← $27
LDI R29,  $00;  R29 ← $00
LDI R28,  $46;  R28 ← $46   } Y(R29, R28)   ← $0046;
ST Y, R24; МБ ICR1←R24 = $27.
```

Завантаження регістра OCR1A

16-розрядний регістр OCR1A має адресу – $004B (старший байт): $004A (молодший байт) [3]. В OCR1A треба завантажити – 20 = 00010100B = $0014 (див. вище).

Тоді програма має такий вигляд:

```
LDI R24, $00; R24← $00;
LDI R31, $00;  R31 ← $00;
LDI R30,  $4B;  R30 ← $4B   } Z(R31, R30) ← $004B;
ST Z, R24; СБ OCR1A←R24 = $00.
LDI R25, $14; R25 ← $14;
LDI R31,   $00;   R31 ← $00;
LDI R30,   $4A;  R30 ← $4A  } Z(R31, R30) ← $004A;
ST Z, R25; МБ OCR1A ← R25 = $14.
```

Програмування $К_{діл}$ = N = 8, режиму роботи номер 8 та запуск таймера

Вище під час завантаження регістра TCCR1A в нього було записано WGM11 = 0 та WGN10 = 0, що разом з двома бітами WGM12 = 0 та WGM13 = 1 регістра TCCRB програмують режим роботи № 8 (табл. 5.2).

Тоді для програмування $К_{діл}$ = 8, режиму роботи № 8 та запуску таймера в регістр TCCRB треба завантажити керувальне слово КС3 = $12:

7 р.	6 р.	5 р.	4 р.	3 р.	2 р.	1 р.	0 р.
ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10
0	0	0	1	0	0	1	0(КС3 = 12)

(режим 8) ($К_{діл}$ = 8)

Адреса регістра TCCR1B дорівнює $004E [3].

Тоді програма має такий вигляд:

```
LDI R17, $12; R17← $12;
LDI R29, $00; R29← $00;
LDI R28, $4E; R28← $4E;
ST Y, R17; TCCR1B ←R17 = $12.
```

5.5.7. Схема алгоритму роботи

Для моделювання пристрою керування двигуном постійного струму у пакеті PROTEUS 8.6 (рис. 5.10) розроблено схему алгоритму роботи (рис. 5.23) та робочу програму мовою С, яку наведено нижче.

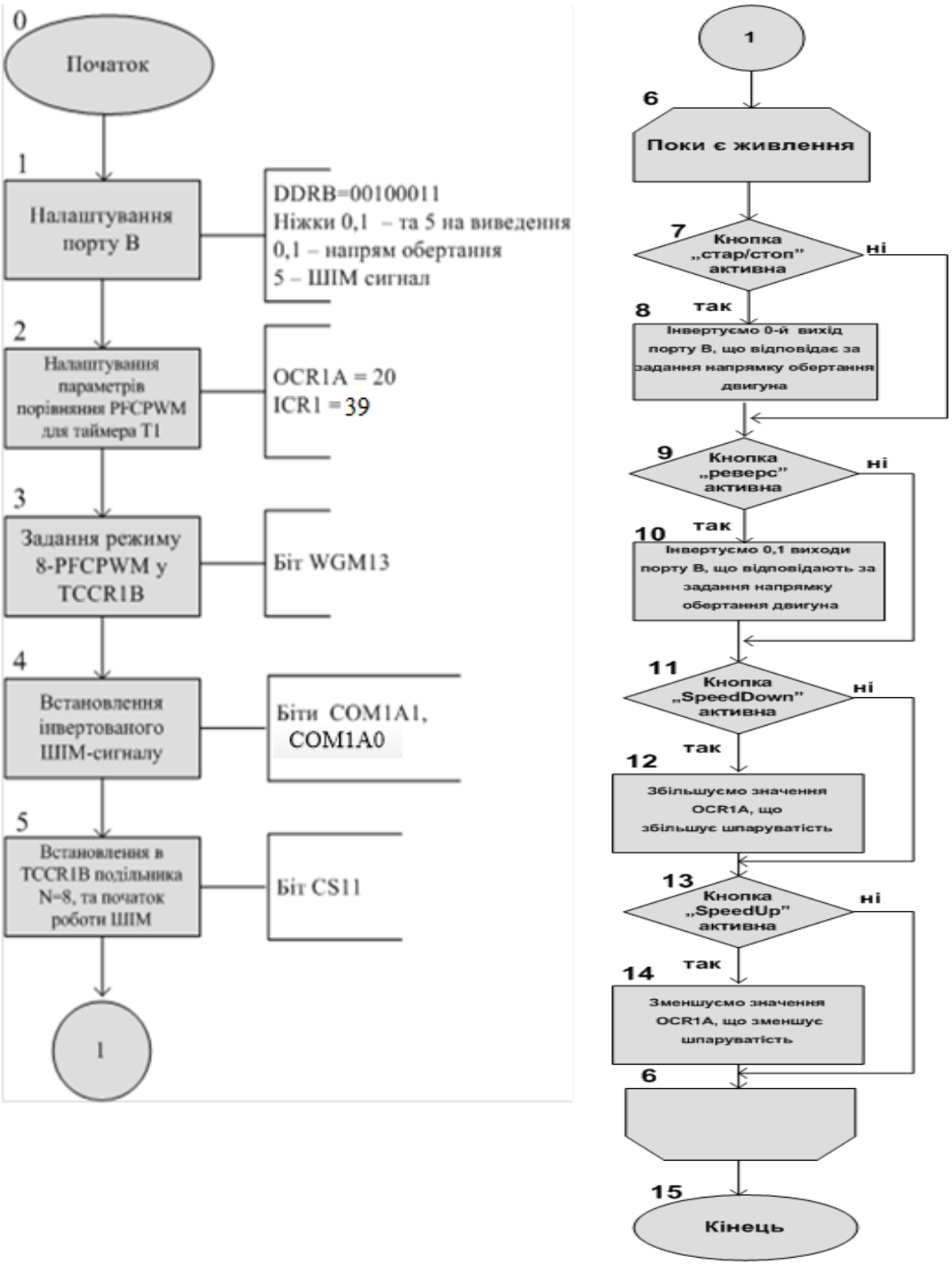


Рисунок 5.23 – Схема алгоритму роботи

5.5.8. Робоча програма мовою С

Запишемо робочу програму мовою С:

```
#include<inttypes.h>
#include<avr/io.h>
#include<avr/interrupt.h>
#include<avr/sleep.h>
#include<util/delay.h>
#define F_CPU 16000000L
int main(void) // 0
{
// 1: Init port B
DDRB = 0b00100011;
PORTB= 0b00000000;
// 2: Init 16-bit timer 1 values
OCR1A = 20;
ICR1 = 40;
int delay = 1;
// Init Phase and frequency correct PWM
TCCR1B = _BV(WGM13);  // 3
TCCR1A = 0;
TCCR1A |= _BV(COM1A0);// 4
TCCR1A |= _BV(COM1A1);
TCCR1B |= _BV(CS11);  // 5
char flag_start_stop=0,flag_reverse=0,flag_speedup=0,flag_speeddown=0;
while(1) // 6
{
// 7: Start– Stop button action
if(!(PINB&4))
{ flag_start_stop=1; _delay_ms(10); }
if(( flag_start_stop==1 )&&(PINB&4))
{PORTB^=1; flag_start_stop=0; }   // 8
// 9: Reverse button action
if(!(PINB&8))
{ flag_reverse=1; _delay_ms(10); }
if(( flag_reverse==1 )&&(PINB&8))
{PORTB^=3; flag_reverse=0; }  // 10
// 11: Speed –  button action
if(!(PINB&64))
{ flag_speeddown=1; _delay_ms(10); }
if(( flag_speeddown==1 )&&(PINB&64))
{ if (OCR1A!=40) OCR1A+=delay; flag_speeddown=0; } // 12
// 13: Speed + button action
if(!(PINB&16))
```

```
{ flag_speedup=1; _delay_ms(10); }
if(( flag_speedup==1 )&&(PINB&16))
{
if (OCR1A!=0)OCR1A– =delay; // 14
flag_speedup=0;
}
} // 6: End of loop
} // 15: End of program
```

5.6. Моделювання модуля таймера в якості годинника реального часу

5.6.1. Опис моделі годинника реального часу

Робочу модель годинника реального часу у пакеті PROTEUS 8.6 показано на рис. 5.24. Основним компонентом моделі є МК ATmega128 (U2, який виконує всі обчислення та керує зовнішніми пристроями.

Як задаючий елемент для відліку часу використовується годинниковий кварцовий резонатор із частотою 32768 Гц (X1). Кварц підключено до виводів TOSC1, TOSC2 МК. Сигнал з цих виводів у МК проходить блок генератора, передподільника частоти та надходить у 8-розрядний таймер/лічильник T0, де використовується як опорна частота. Крім годинникового кварцу в схемі є основний кварцовий резонатор (X2), з частотою 4 МГц. Сигнал з цього кварцу використовується для тактування ядра МК та його периферії.

Для індикації значення годин, хвилин і секунд застосовано семисегментний світлодіодний індикатор на 6 розрядів, що має спільний катод. Індикатор, якщо рахувати зліва, відображає час по 2 розряди: 1 і 2 – години, 3 і 4 – хвилини, 5 і 6 – секунди, між якими ставиться крапка, наприклад: «12.45.15».

Для управління індикатором потрібно подавати керуючі сигнали на лінії включення сегментів (A…G, DP) і обирати активний розряду за допомогою ліній 1…6. Оскільки виходи МК мають недостатню здатність навантаження за струмом, перед індикатором використовуються дві додаткові мікросхеми: TC4468 (U2, U3), які виконують функцію буфера-защіпки з трьома станами.

Крім цих мікросхем для керування індикатором використовується також мікросхеми TC4426 (U4…U6), які виконують функцію буфера з відкритим стоком, який інвертує. Цей буфер здатний віддавати струм у навантаження до 1,5 А, коли його вихід перебуває у стані логічного нуля. Наявність інвертора дозволяє вибирати потрібний розряд на індикаторі за допомогою логічної одиниці на відповідному виході МК.

Для обмеження струму через сегменти світлодіодного індикатора використовуються резистори R1…R8.

Для виведення цифр у програмі застосовано метод динамічної індикації.

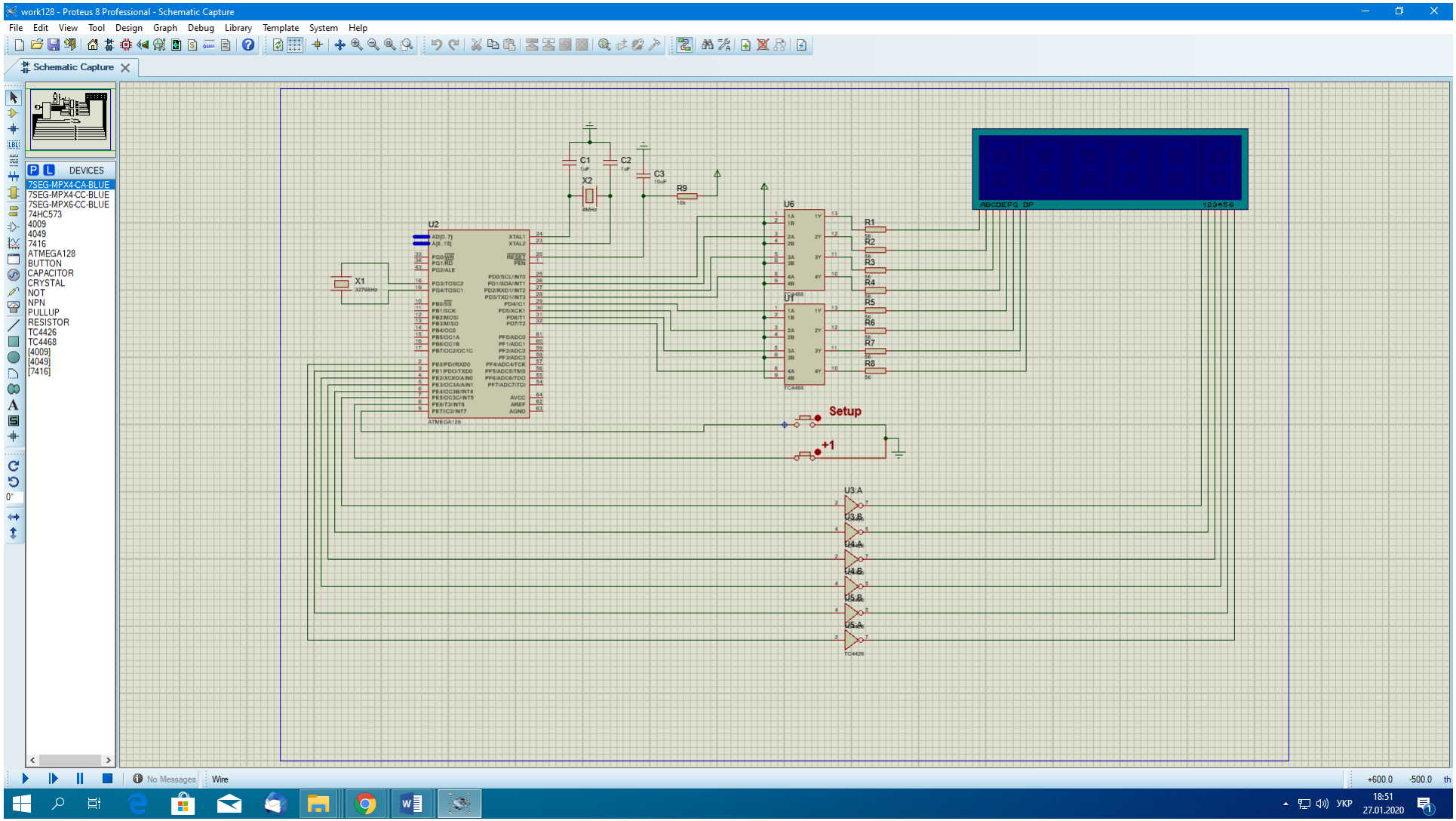


Рис. 5.24. Робоча модель годинника реального часу

Кожна цифра виводиться окремо одна за одною. В певний момент часу на індикаторі горить лише одна цифра, але завдяки високій швидкості виведення для ока цей процес непомітний, і все виглядає так, ніби всі цифри горять одночасно.

Для виведення бітового образу числа на відповідний розряд індикатора використовується порт D, а для вибору активного розряду індикатора – шість молодших розрядів порту E МК. Старші два біти порту E використовуються як входи зовнішніх переривань INT6 і INT7. До цих входів підключено кнопки: «+1» і «Setup».

Кнопка «Setup» виконує циклічне переключення режимів (робочий режим -> налаштування годин -> налаштування хвилин -> налаштування секунд -> робочий режим).

Кнопка «+1» працює тільки в режимах налаштування і виконує інкремент значення часу (залежно від режиму налаштування – годин, хвилин або секунд). Під час натискання та утримання цієї кнопки відбувається автоматичний інкремент відповідного значення.

Для автоматичного формування під час увімкнення живлення сигналу RESET на однойменному вході МК застосовано RC-ланцюжок (C3, R9).

Роботу годинника засновано на використанні RTC-блока 8-розрядного таймера T0 МК. Цей блок у разі відповідного налаштування викликає переривання з періодом, який дорівнює 1 секунді. На відміну від основного кварцу МК, блок RTC використовує додатковий кварц, який має частоту 32768 Гц. Це дозволяє під час ділення 32768/128/256 отримати рівно 1 с.

Мікросхема TC4468

Ця мікросхема є 4-канальним буфером-защіпкою з третім станом [1; 6]. Завдяки наявності цього стану і високої здатності навантаження виходів, ця мікросхема використання може використовуватися як шинний формувач.

Буфер-защіпка керується за допомогою входу 1B/2B/3B/4B (активація засувки). Коли на цьому вході присутня логічна одиниця, значення на виходах Y будуть повторювати значення з входів A. Якщо на вході 1B/2B/3B/4B логічний нуль, то виходи будуть зберігати своє значення доти, поки на цьому вході знову не з’явиться логічна одиниця.

Мікросхема TC4426

Ця мікросхема містить два інвертуючих буфери з виходами типу «відкритий сток» [1; 6]. У пристрої є три мікросхеми, які виконують функцію шести незалежних інверторів. Кожен буфер здатний віддавати струм у навантаження до 1,5 А, коли його вихід перебуває у стані логічного нуля.

Наявність інвертора дозволяє вибирати потрібний розряд на індикаторі за допомогою логічної одиниці на відповідному виході МК.

5.6.2. Схема алгоритму роботи пристрою

Схему алгоритму роботи наведено на рис. 5.25.

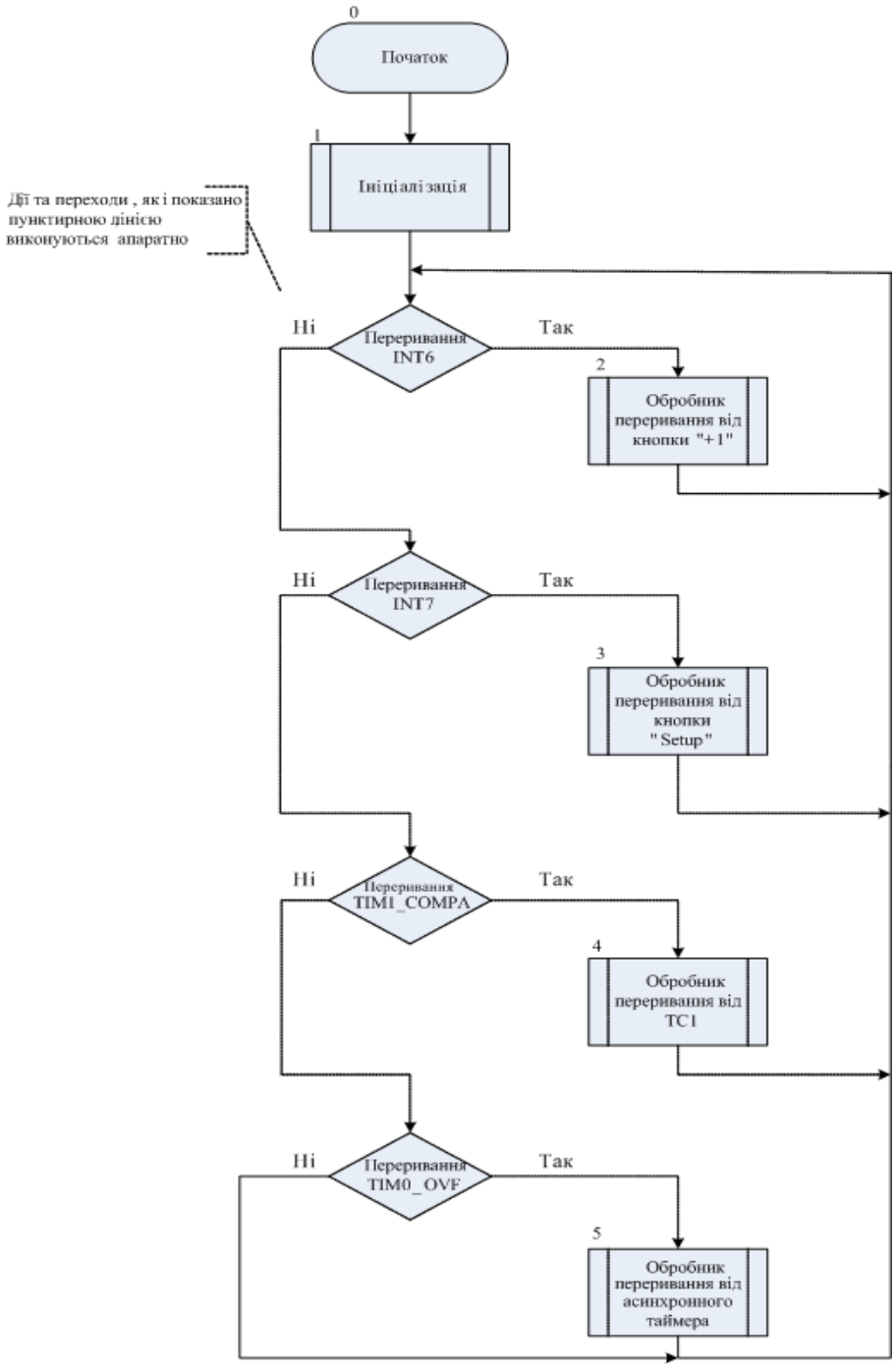


Рис. 5.25. Схема алгоритму роботи

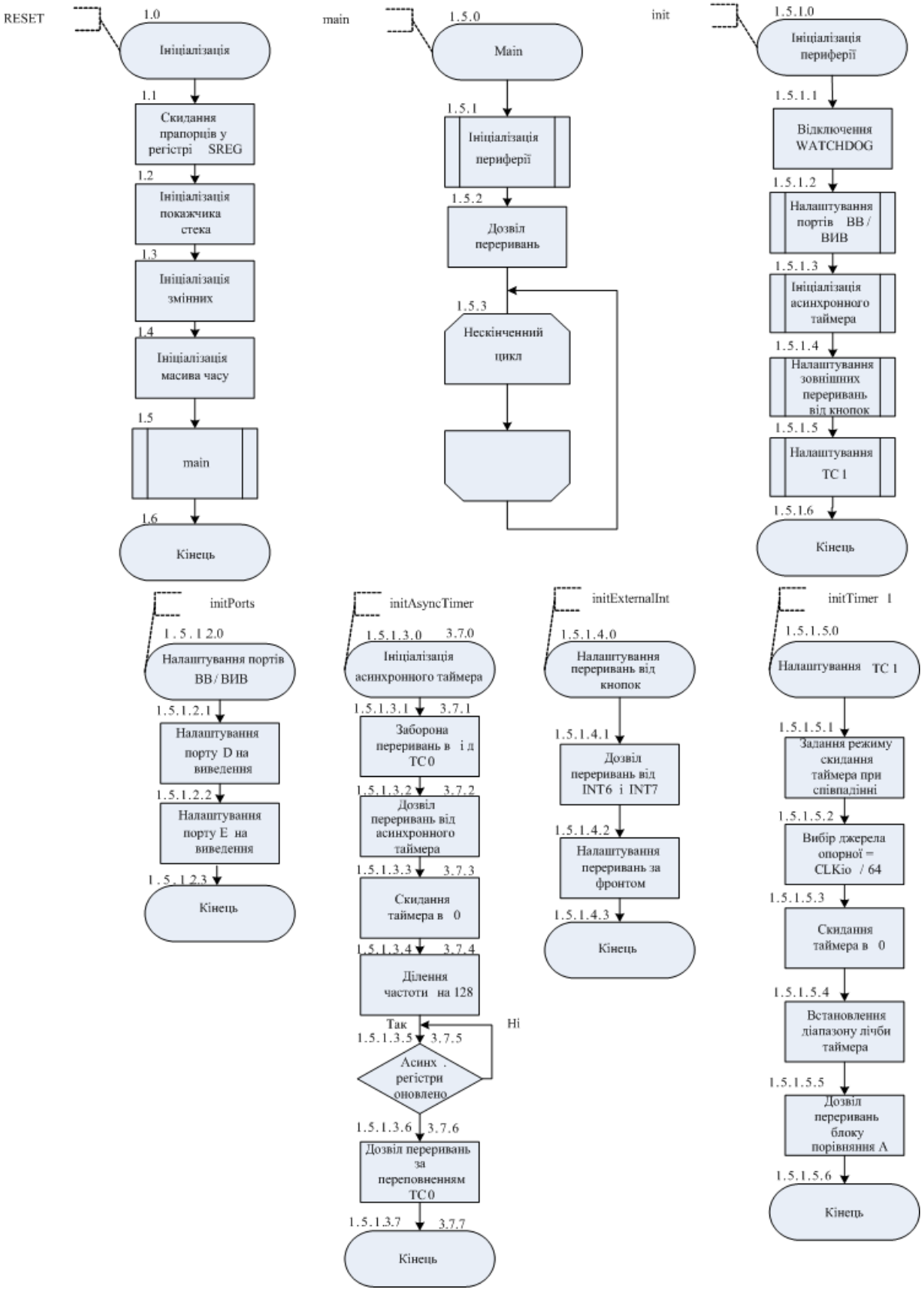


Рис. 5.25. Продовження (див. також с. 260)

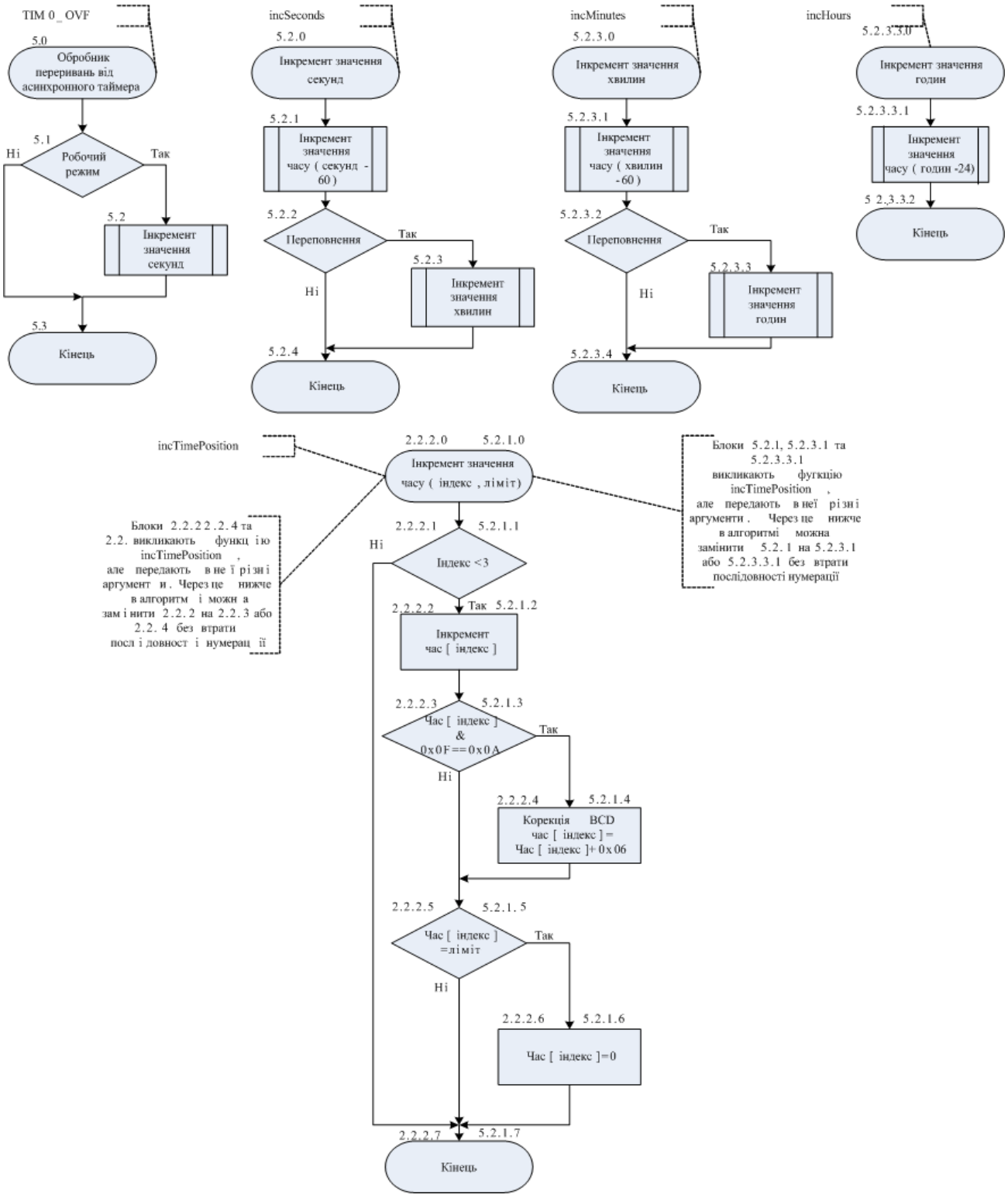


Рис. 5.25. Продовження (див. також с. 260, 261)

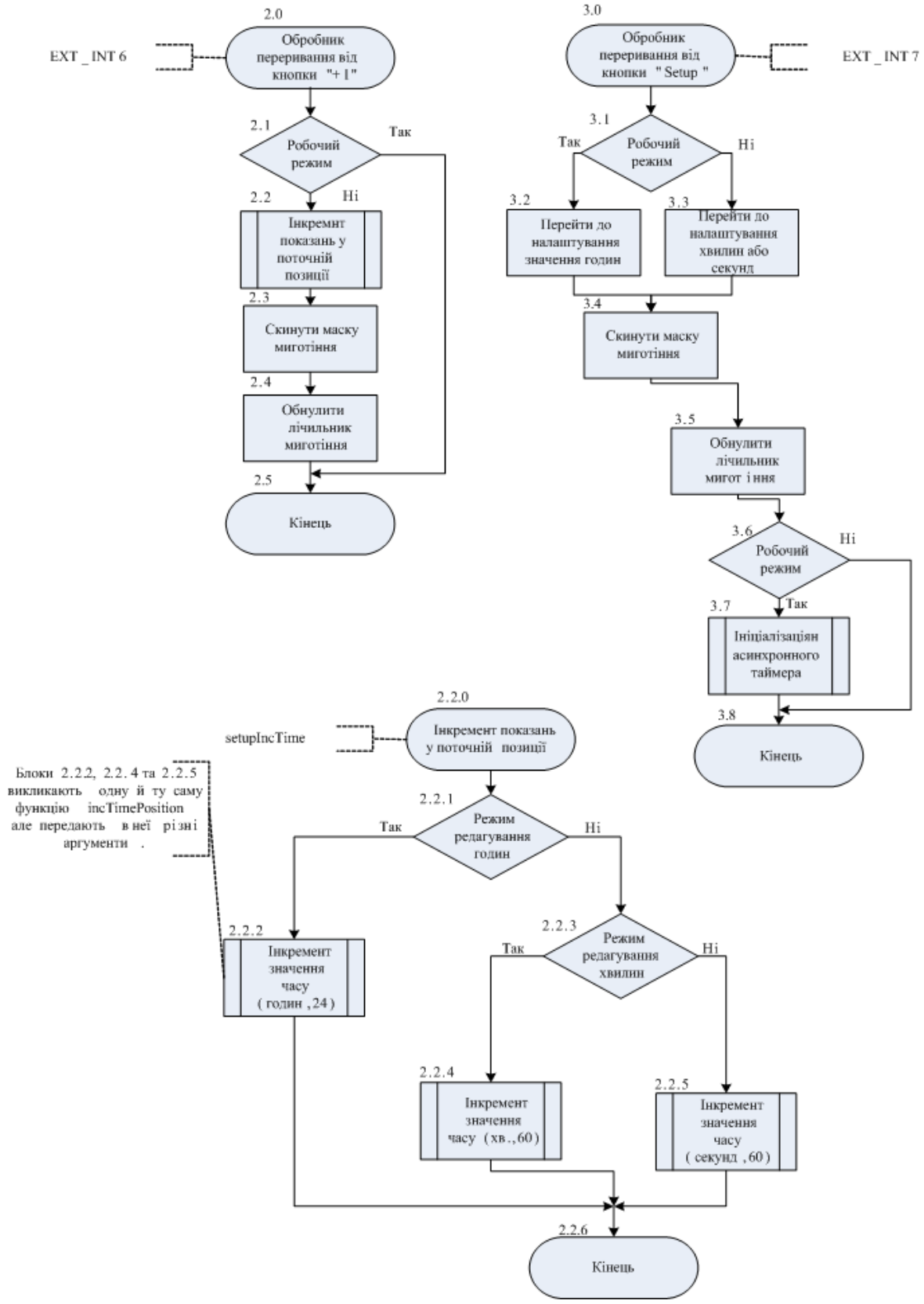


Рис. 5.25. Продовження (див. також с. 260…262)

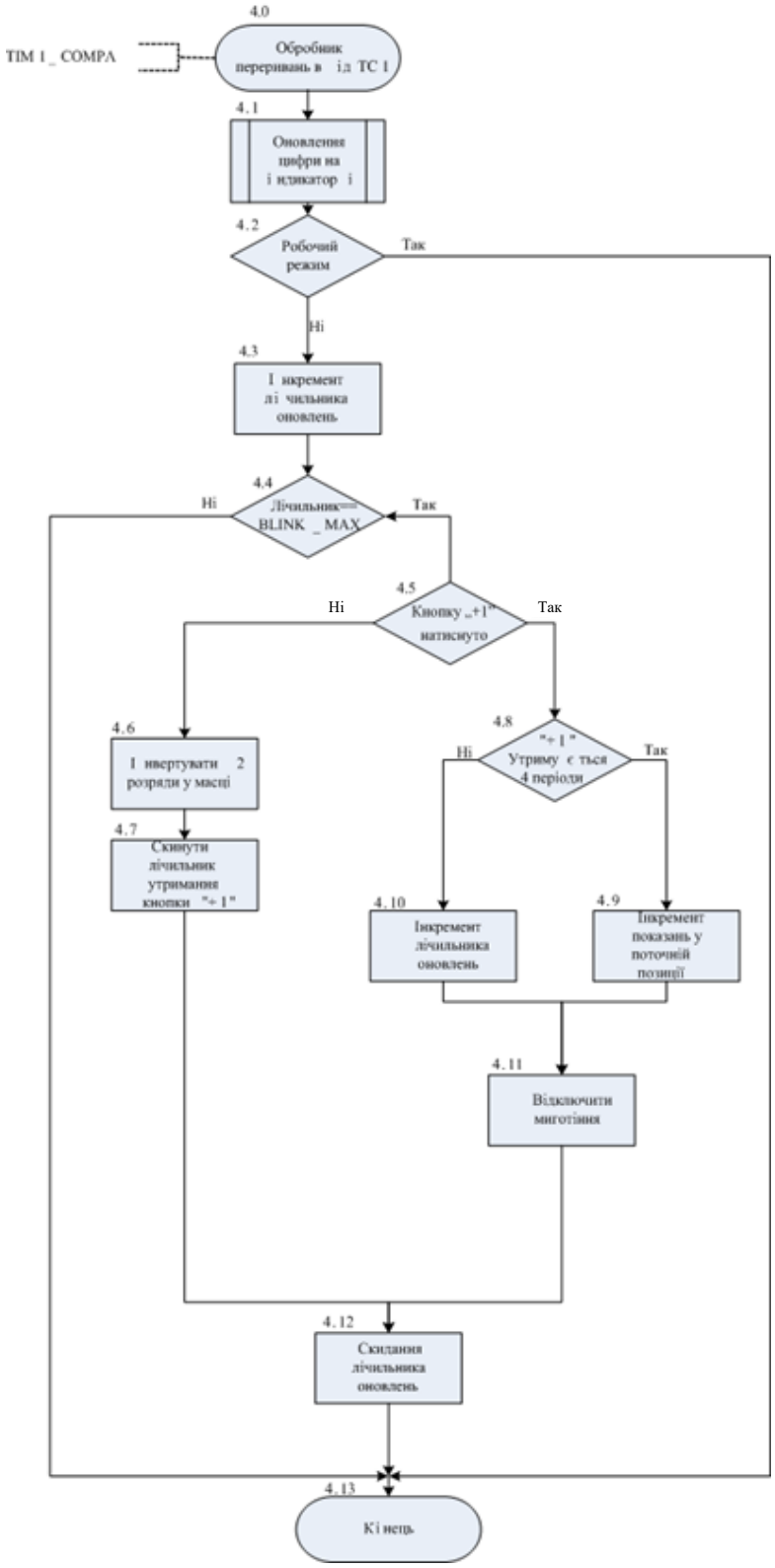


Рис. 5.25. Продовження (див. також с. 260…263)

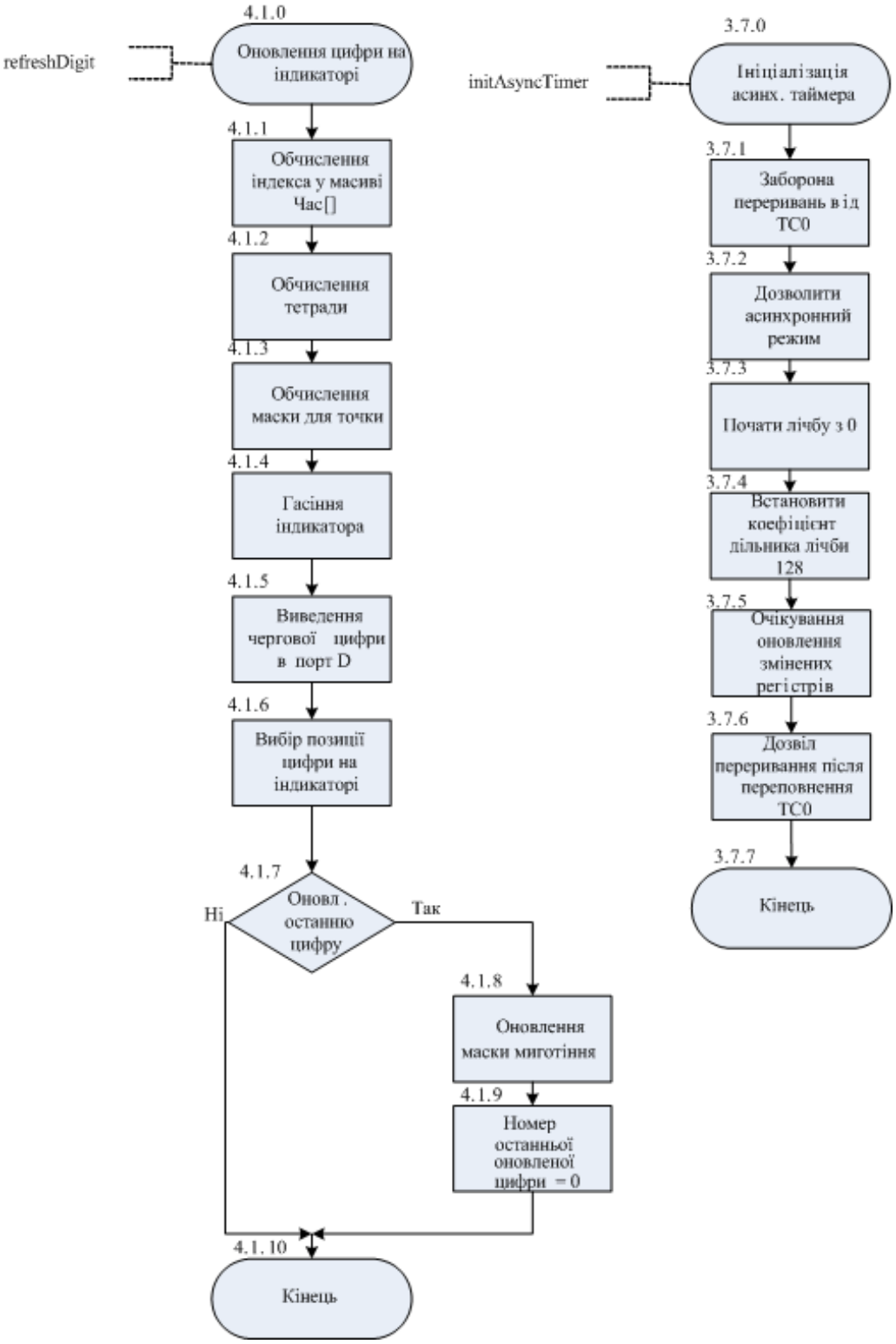


Рис. 5.25. Закінчення (див. також с. 260…264)

5.6.3. Робоча програма мовою С

Робочу програму мовою С та опис її роботи наведено у [3].

5.7. Застосування мікросхеми програмованого таймера для формування інтервалів часу

5.7.1. Загальна характеристика програмованого таймера

Для реалізації генераторів тактових імпульсів необхідної частоти, формування часових затримок, імпульсів визначеної тривалості, стробувальних імпульсів, а також підрахунку зовнішніх подій може використовуватися, наприклад, програмований таймер і8253 [2].

На рис. 5.26*а* показано умовне позначення на електричних схемах, а на рис. 5.26*б* зображено структурну схему програмованого таймера, що містить три однакових канали, кожен з яких має 16-бітний лічильник, що віднімає, і схему керування.

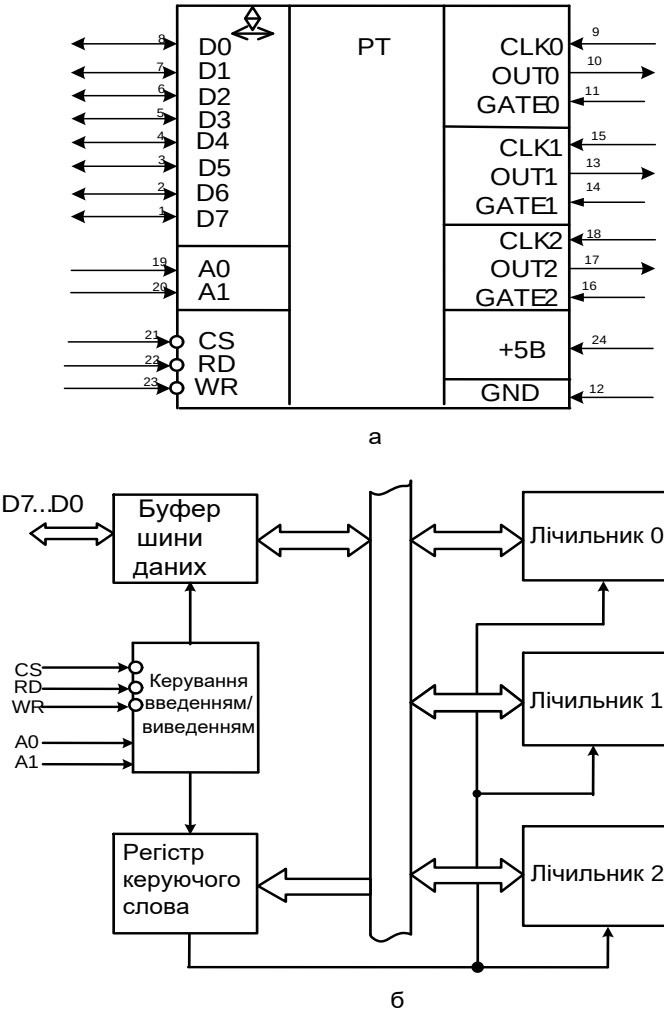


Рис. 5.26. Програмований таймер і8253:
а – умовне позначення на електричних схемах; *б* – структурна схема

Канали програмуються і працюють незалежно один від одного.

Лічильник можна запрограмувати для роботи у двійковому чи двійково-десятковому кодах. Діапазон підрахунку складає 65 536 для двійкового і 10 000 для двійково-десяткового коду.

Кожен лічильник має два керувальних входи CLK і GATE та один вихід OUT. На вхід CLK подаються синхроімпульси від зовнішнього генератора тактових імпульсів, зрізом яких виконується декремент вмісту лічильника. Крім цього, на вхід CLK можуть подаватися імпульси зовнішніх подій, якщо таймер запрограмований у відповідний режим.

Для керування кожним каналом можна використовувати зовнішній сигнал GATE, який подається на однойменний вхід. Кожен канал програмованого таймера, окрім 16-розрядного лічильника містить 16-розрядний буферний регістр, в який записується початкове значення під час програмування таймера.

Вплив керувального сигналу на вході GATE на роботу таймера описано в табл. 5.12.

Таблиця 5.12. **Вплив сигналу GATE на роботу таймера**

Режим	Низький рівень	Додатний фронт (фронт)	Високий рівень сигналу
0	Забороняє підрахунок (вентиль)	-	Дозволяє підрахунок (вентиль)
1	-	Ініціює підрахунок. У наступному такті синхронізації встановлює *OUT=0*	-
2	Забороняє підрахунок. Встановлює *OUT =1*	Ініціює підрахунок	Дозволяє підрахунок
3	Те саме	“ “	“ “
4	Забороняє підрахунок (вентиль)	-	Те саме (вентиль)
5	-	Ініціює підрахунок	-

Під підрахунком розуміється декремент лічильника зрізом кожного імпульсу на вході CLK. З виходу OUT знімається вихідний сигнал відповідного таймерного каналу. У кожен з лічильників можна завантажити початкове значення із системної шини, а його поточне значення прочитати, припиняючи чи не припиняючи підрахунок.

Зв’язок таймера із системною шиною МПС здійснюється через 8-розрядний двонапрямний буфер шини даних, як показано на рис. 5.27.

Схема керування введенням/виведенням (рис. 5.26) здійснює керування роботою таймера на основі обробки вхідних керувальних сигналів на входах: RD – читання, WR – запис, CS – вибір кристала, A1 і A0 – значення молодших бітів на системній адресній шині.

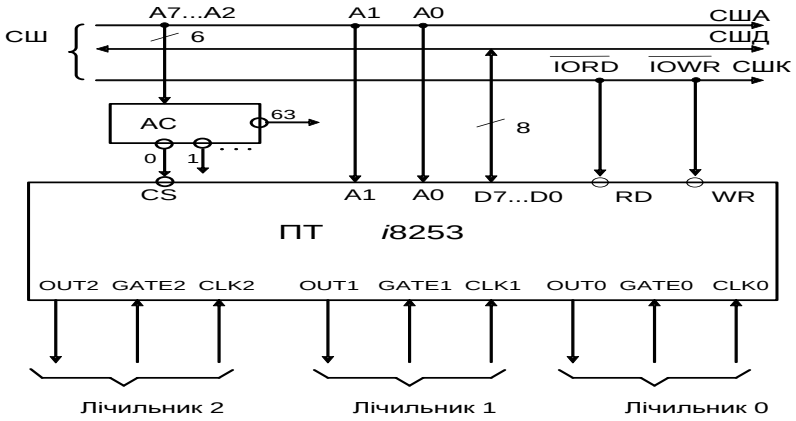


Рис. 5.27. Схема підключення таймера до системної шини МПС

Функції, які виконуються схемою керування залежно від комбінації сигналів на її входах, описано у табл. 5.13.

Таблиця 5.13. **Вплив керувальних сигналів на роботу таймера**

Сигнал на вході					Функція
CS	WR	RD	A1	A0	Дії, які виконує таймер
0	0	1	0	0	Завантаження лічильника 0 з шини даних
0	0	1	0	1	Завантаження лічильника 1 з шини даних
0	0	1	1	0	Завантаження лічильника 2 з шини даних
0	0	1	1	1	Завантаження регістра керуючого слова із шини даних
0	1	0	0	0	Зчитування лічильника 0 на шину даних
0	1	0	0	1	Зчитування лічильника *1* на шину даних
0	1	0	1	0	Зчитування лічильника *2* на шину даних
0	1	0	1	1	Виводи D7…D0 знаходяться у високоімпедансному стані
0	1	1	X	X	
1	X	X	X	X	

Примітка. Літерою «X» у таблиці позначено довільний стан сигналу: 0/1.

Програмований таймер може працювати в одному із шести режимів:

– переривання за закінченням підрахунку (програмована затримка), режим 0;
– програмований одновібратор, режим 1;
– програмований генератор прямокутних імпульсів зі шпаруватістю Q ≠ 2, режим 2;
– програмований генератор прямокутних імпульсів зі шпаруватістю Q = 2 (меандр), режим 3;
– лічильник зовнішніх подій (формувач одиничного імпульсу-строба) із програмним запуском, режим 4;
– лічильник зовнішніх подій (формувач одиничного імпульсу-строба) з апаратним запуском, режим 5.

Роботу таймера у різних режимах описано у [2].

5.7.2. Програмування таймера

Для функціонування таймера у визначеному режимі необхідно завантажити керувальне слово в однойменний регістр керувального слова, формат якого показано на рис. 5.28.

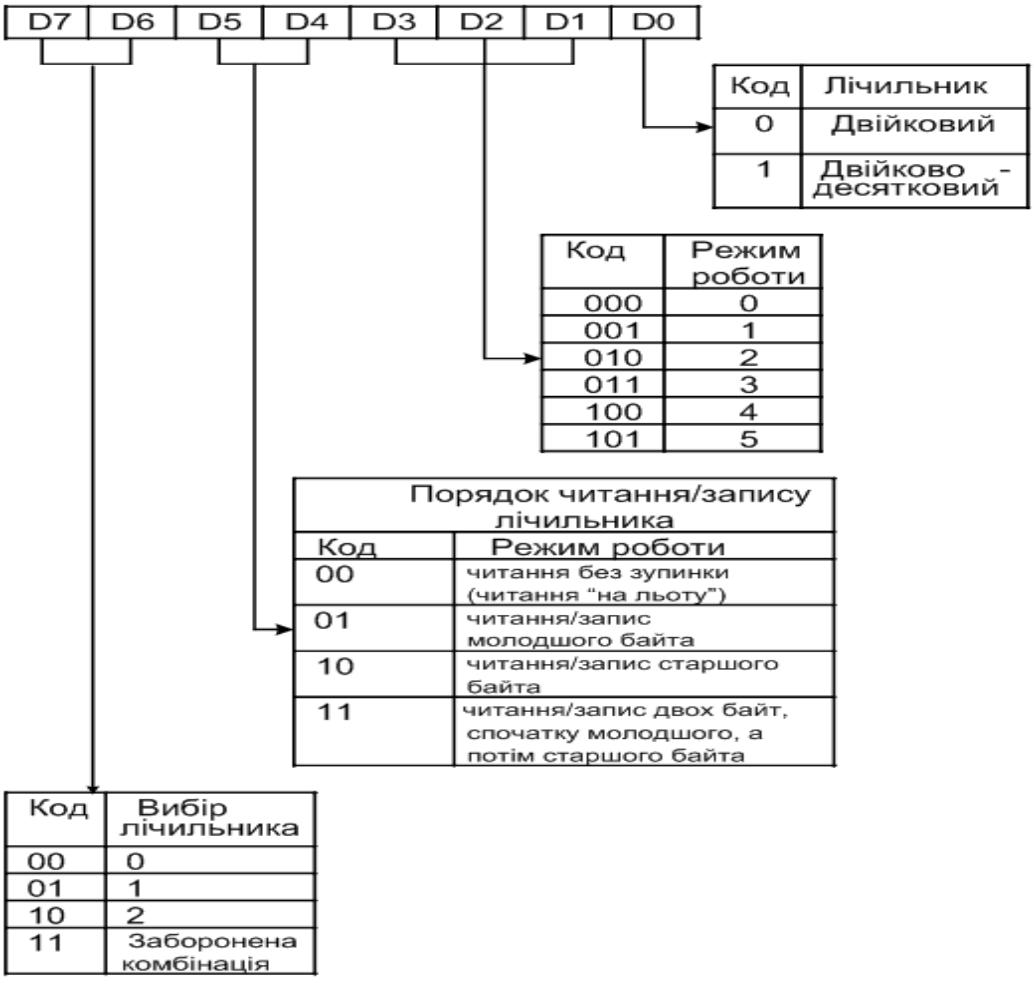


Рис. 5.28. Формат регістра керувального слова

Після завантаження керувального слова до регістра керувального слова вміст регістра і лічильників таймерів обнуляється. Керувальне слово програмує режим роботи лічильників, послідовність завантаження і зчитування інформації, вид коду, що використовується (двійковий чи двійково-десятковий). Вміст регістра керувального слова не може бути прочитаний.

Кожен канал (лічильник) програмованого таймера програмується у довільному порядку з урахуванням одного обмеження: керувальне слово записується в програмований канал першим.

Обраний лічильник обов'язково має бути завантажено тією кількістю байтів, що було запрограмовано в керувальному слові. Завантаження лічильників обов'язково має виконуватися відразу після запису керувального слова. *Після запису керувального слова у лічильнику встановлюється число початкового встановлення, що дорівнює нулю.*

Мікросхема і8253 може адресуватися або як пристрій введення/виведення, або як комірка пам'яті.

Нижче розглянуто приклад програмування таймера, в якого робочий канал – лічильник 1; адресація програмованого таймера відбувається розрядом А3 системної шини адреси МПС; режим роботи – 1; код, що використовується – двійковий; порядок читання/запис значень лічильника – запис/читання 2-х байт (спочатку молодший, потім старший байт):

```
MVI A, 01110010B; формування в акумуляторі керувального слова;
OUT 11110111B; завантаження керувального слова до регіста керувального
слова програмованого таймера: A0 = A1 = 1 – вибір регіста керувального слова,
A3 = CS = 0 – вибір програмованого таймера;
MVI A, < D7..D0 > B; формування в акумуляторі молодшого байта
                    ; числа, що завантажується у лічильник;
OUT 11110101B    ; завантаження в лічильник 1 молодшого байта
                ; числа: A1 = 0, A0 = 1– вибір лічильника 1,
              ; A3 = CS = 0 – вибір програмованого таймера);
MVI A, < D15..D8 > B; формування в акумуляторі старшого байта числа
                     ; що завантажується в лічильник;
OUT 11110101B     ; завантаження в лічильник 1 старшого байта
                ; числа: A1 = 0, A0 = 1 – вибір лічильника 1;
                ; A3 = CS = 0 – вибір програмованого таймера.
```

У цьому прикладі використано команди МП і8080 [2].

Вміст будь-якого лічильника можна прочитати програмним способом, що необхідно у випадках, коли на основі аналізування прочитаного значення приймається рішення про подальший обчислювальний процес.

Зчитування вмісту лічильника може виконуватись двома способами: *з зупинкою і без зупинки лічильника (читання «на льоту»).* У першому випадку роботу лічильника можна припинити подачею на керувальний вхід GATE логічного нуля (режими 0, 2, 3, 4) чи за допомогою зовнішньої логічної схеми, що припиняє подачу імпульсів на вхід CLK лічильника. Операцію читання вмісту лічильника обраного каналу необхідно обов'язково виконувати до кінця, тобто якщо запрограмовано читання/запис двох байтів, то ці два байти треба прочитати. У цьому разі спочатку читається молодший байт, а потім старший.

Режим читання «на льоту» дозволяє зчитувати вміст лічильника без зупинки підрахунку. Для цього керувальним словом необхідно відповідно запрограмувати таймер, встановивши D4 = D5 = 0.

У цьому разі в розрядах D7 і D6 керувального слова встановлюється номер лічильника (таймерного каналу), а значення D3...D0 – несуттєві.

Після запису керувального слова вміст лічильника запам'ятовується у спеціальному буферному регістрі лічильника, а потім командою читання зчитується. Операцію читання треба завершити відповідно до запрограмованої кількості байт. Запис керувального слова режиму читання не змінює режим роботи лічильника.

Програма читання «на льоту» вмісту лічильника 1, що адресується розрядом A3, має такий вигляд:

```
MVI A, 01000000B ; формування в акумуляторі керувального слова читання «на льоту»;
OUT 11110111B    ; запис керувального слова до керувального слова режиму таймера;
IN 11110101B     ; читання молодшого байта лічильника 1 в акумулятор;
STAX rp1         ; M(rp.1) ← A;
IN 11110101B     ; читання старшого байта лічильника 1 в акумулятор.
```

У цьому прикладі також використано команди МП i8080 [2].

На рис. 5.29 наведено схему, що пояснює використання таймера як формувача тактової частоти для послідовного інтерфейсу УСАПП i8251.

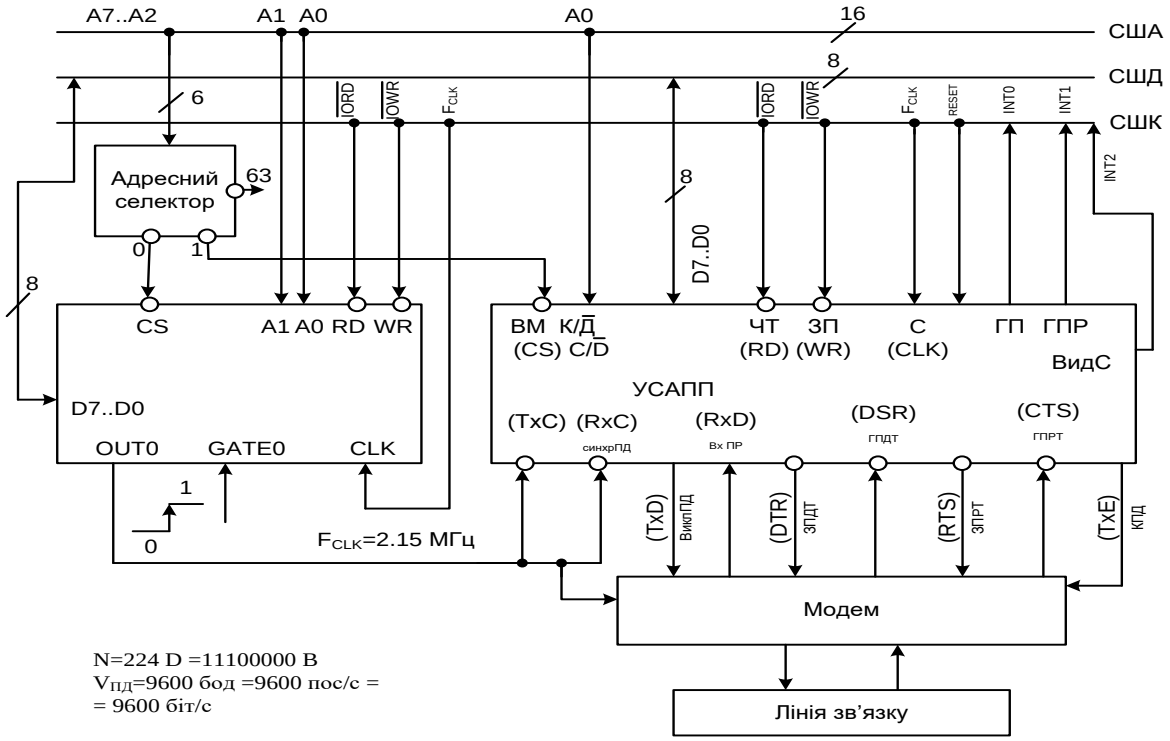


Рис. 5.29. Використання таймера і8253 у схемі сполучення МПС з модемом

Контрольні запитання та завдання

1. Назвіть способи формування інтервалів часу в МПС.
2. Який цифровий електронний пристрій складає основу програмованого таймера?
3. Чим відрізняється режим таймера від режиму підрахунку зовнішніх подій?
4. Для чого може використовуватись таймер/лічильник?
5. Опишіть призначення виводів таймерів/лічильників.
6. Назвіть регістри, які можуть використовуватись для дозволу/заборони переривань від таймерів/лічильників, та опишіть їх формати.
7. Наведіть структуру блока попереднього дільника таймера/лічильника без асинхронного режиму та опишіть її роботу.
8. Наведіть структуру блока попереднього дільника таймера/лічильника з асинхронним режимом та опишіть її роботу.
9. Назвіть регістри, які можуть здійснювати скидання попередніх дільників, або їхню зупинку, та опишіть їх формати.
10. Наведіть структури, назви та формати керуючих регістрів 8-розрядних таймерів/лічильників. Опишіть їх роботу.
11. Опишіть режими роботи 16-розрядних таймерів/лічильників.

12. Для рішення яких завдань краще використання режиму Phase Correct PWM ніж режиму Fast PWM?
13. Для чого можуть використовуватись 16-розрядні таймери/лічильники?
14. Під час виникнення яких подій таймери/лічильники T1, T3, T4, T5 можуть генерувати переривання?
15. За допомогою яких регістрів виконується керування таймером/лічильником?
16. Які сигнали можуть використовуватись в якості тактового сигналу f_{clkn} для таймерів/лічильників T1, T3, T4 і T5?
17. Як здійснюється вибір джерела тактового сигналу, а також запуск і зупинка таймерів/лічильників?
18. Завдяки чому в режимі Phase and Frequency Correct PWM кожен період сигналу є повністю симетричним?
19. Поясніть, що таке ШІМ-сигнал?
20. Як обчислюється шпаруватість ШІМ-сигналу?
21. Як впливає значення шпаруватості на швидкість обертання двигуна?
22. Наведіть та поясніть модель пристрою керування двигуном постійного струму.
23. Наведіть та поясніть схему алгоритму роботи моделі.
24. Наведіть та поясніть робочі програми мовою Асемблера та С, що керує моделлю пристрою керування двигуном.
25. Наведіть та поясніть робочу модель годинника реального часу.
26. Наведіть та поясніть схему алгоритму роботи та робочу програму керування моделлю годинника реального часу.
27. Опишіть структуру та призначення виводів програмованого таймера i8253.
28. Як таймер i8253 підключається до системної шини?
29. Опишіть особливості програмування таймера i8253.
30. Наведіть схему, що пояснює використання таймера як формувача тактової частоти для послідовного інтерфейсу УСАПП i8251, та поясніть її роботу.

6. АРХІТЕКТУРА МОДУЛЯ ВВЕДЕННЯ/ВИВЕДЕННЯ

6.1. Особливості архітектури модуля введення/виведення

6.1.1. Призначення та місце модуля введення/виведення в мікропроцесорній системі

На технічні характеристики МПС вагомо впливають засоби обміну інформацією між обчислювальним ядром і різноманітними периферійними пристроями. Ці засоби створюють підсистему (модуль) введення/виведення інформації, яка містить у собі програмні та апаратні засоби.

Під час вивчення модульної структури МПС було зазначено, що одним із трьох основних модулів МПС є модуль введення/виведення (рис. 1.15), що забезпечує взаємозв'язок МП із зовнішніми пристроями.

У ньому можуть міститися, наприклад, паралельний програмований інтерфейс і послідовний програмований універсальний синхронно/асинхронний приймач/передавач (інтерфейс) (УСАПП). Ці інтерфейси необхідні для організації паралельного та послідовного обміну інформацією між зовнішніми пристроями і МП.

Функціональні можливості МП досить обмежені, оскільки вони зазвичай не містять пам'ять достатнього об'єму і порти введення/виведення для зв'язку із зовнішніми пристроями. Тому МПС на основі МП містять модулі пам'яті та пристроїв введення/виведення.

Окремі МК, наприклад, сімейства МК51 та AVR, мають внутрішню (резидентну) пам'ять і порти введення/виведення, кількості яких може бути недостатньо, тому використовують зовнішні пристрої обміну та пам'ять.

Під час розгляду зв'язків між окремими елементами МПС зазвичай використовують поняття інтерфейс, що є границею між декількома пристроями, наприклад, між МП та зовнішнішнім пристроєм.

Під інтерфейсом розуміють сукупність уніфікованих технічних і програмних засобів, необхідних для підключення пристроїв до системи чи однієї системи до іншої. Серед властивостей інтерфейсу можна зазначити розв'язування задач синхронізації, вибору напрямку передачі даних, а також приведення у відповідність рівнів та форм сигналів.

Основною частиною інтерфейсу є технічні (апаратні) засоби, що забезпечують зв'язок МП/МК та зовнішнішнім пристроєм. Ці засоби називають модулем введення/виведення. Основу цього модуля складають порти введення/виведення інформації, які виконано на основі регістрів. Окрім наявності необхідних апаратних засобів модуль введення/виведення є програмованим, тобто має відповідні регістри для запису керувальних слів і регістри, які відображають стан інтерфейсу.

Сучасні МП/МК мають команди обміну даними з необхідною периферією. Введення (читання, прийом) відповідає потоку даних від зовнішнього пристрою у МП/МК, а виведення (запис, передача) – потоку даних із МП/МК до зовнішнього пристрою.

6.1.2. Внутрішня та зовнішня системи введення/виведення мікропроцесора/мікроконтролера

У мікропроцесорній техніці окрім поняття «інтерфейс» зазвичай використовують поняття «система введення/виведення», до якої належать, насамперед, порти введення/виведення та їхня програмна підтримка [2; 3].

У загальному випадку, система введення/виведення МП або МК може складатися з таких компонентів:

– внутрішньої, власної системи введення/виведення;
– зовнішньої системи введення/виведення.

МПС, які виконано на основі МП, наприклад i8080, i8086, мають тільки зовнішню систему введення/виведення, а системи, які побудовано із застосуванням МК, наприклад, сімейства МК51, AVR, включають внутрішню систему введення/виведення, що може розширюватися застосуванням зовнішньої системи введення/виведення.

До внутрішньої, власної системи введення/виведення МК сімейства МК51, наприклад, AT89C51 належать чотири паралельні порти введення/виведення та один послідовний порт [2].

МК AVR мають значну кількість паралельних та послідовних портів [3; 4].

Основу зовнішньої системи введення/виведення можуть складати:

– паралельний програмований інтерфейс, наприклад i8255;
– послідовний програмований універсальний асинхронний приймач/передавач, наприклад i8251, TL16C550;
– розширювач кількості портів (ліній) введення/виведення, наприклад i8243 і т. ін.

6.1.3. Способи обміну даними між зовнішніми пристроями і мікропроцесорною системою

Залежно від особливостей програмування обміну даними в МПС використовуються два способи:

– під керуванням програми (програмно-керований);
– за перериваннями.

Програмно-керований обмін виконується з ініціативи МП і під його керуванням.

МП постійно опитує стан зовнішнього пристрою за допомогою читання відповідного прапорця готовності. Якщо зовнішній пристрій готовий до обміну,

тобто встановлено у логічну одиницю прапорець готовності, виконується операція обміну: введення чи виведення даних.

Якщо зовнішній пристрій не готовий до обміну, то МП очікує готовності, постійно виконуючи відповідні команди програми і періодично аналізуючи готовність зовнішнього пристрою. Такий спосіб не завжди є ефективним, особливо у випадку, коли треба довго чекати та програма нічого не робить окрім постійної перевірки готовності зовнішнього пристрою до обміну.

Більш ефективним є обмін за перериванням, коли у разі необхідності обміну, викликається підпрограма, яка керує обміном. Після закінчення обміну, передається керування перерваній програмі.

У [2] розглянуто ці види обміну на прикладі використання мікросхеми програмовуваного паралельного інтерфейсу i8255.

Залежно від виду є два способи обміну:

– паралельний, коли одночасно передаються всі або декілька бітів слова даних;
– послідовний, коли біти слова даних пересилаються по черзі, починаючи, наприклад, з його молодшого розряду.

У разі паралельного обміну зовнішній пристрій зв’язується з МП лініями зв’язку, довжина яких обмежена і становить кілька метрів.

У разі послідовного обміну даними та використанні, наприклад, інтерфейсів: CAN, RS-485; модемів і т. ін., довжину ліній зв’язку можна суттєво збільшити.

Крім цього, бажання використовувати для дистанційного обміну інформацією між зовнішнім пристроєм і МПС наявні канали зв’язку, зумовили широке поширення послідовного обміну даними між зовнішнім пристроєм і МПС чи між декількома МПС.

Обмін інформацією всередині МП або МК здійснюється у паралельній формі. Якщо використовується послідовний обмін даними, необхідно: у разі передачі даних від МПС до зовнішнього пристрою – перетворити дані з паралельної форми в послідовну, а у разі прийому інформації від зовнішнього пристрою та введенні її у МПС – перетворити з послідовної форми в паралельну.

Процес перетворення даних з паралельної форми в послідовну показано на рис. 6.1.

Для перетворення даних з паралельної форми в послідовну, інформація завантажується у регістр зсуву. Вміст регістра зсуву під впливом тактових імпульсів від генератора тактових імпульсів послідовно зсувається на один розряд, наприклад, праворуч.

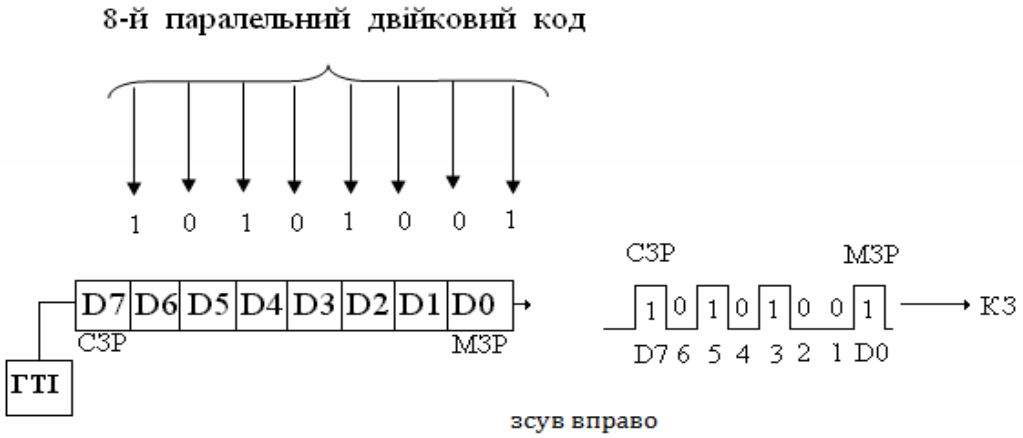


Рис. 6.1. Перетворення даних з паралельної форми в послідовну

Дані на виході такого регістра будуть мати послідовну форму. Переважно під час послідовної передачі в канал зв'язку першим передається молодший біт слова даних (молодший значущий розряд), останнім – старший біт (старший значущий розряд).

Для зворотного перетворення даних з послідовної форми в паралельну необхідно виконати дії, зворотні стосовно описаного. Дані, що надходять з каналу зв'язку в послідовній формі, вводяться біт за бітом у регістр зсуву. Після заповнення регістра зсуву, інформація з нього в паралельній формі передається в МП.

Подібні перетворення виконуються у МК у разі обміну даними через його послідовний порт (рис. 6.2).

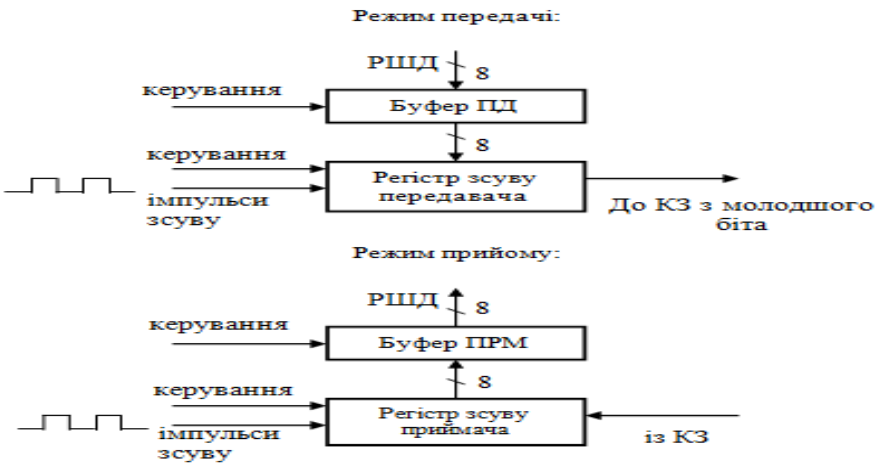


Рис. 6.2. Спрощена структура послідовного порту МК

6.1.4. Адресація пристроїв введення/виведення

У МПС на основі МП адресація пристрою введення/виведення виконується одним із двох способів:

– командами введення (IN) чи виведення (OUT);

– командами звернення до зовнішнього пристрою як комірок пам'яті.

У другому випадку МП розглядає порти введення/виведення як звичайні комірки пам’яті. Заздалегідь з’ясовується діапазон пам’яті, адреси якого привласнюються портам. Це дозволяє використовувати всі операції над вмістом комірок пам’яті для роботи з портами введення/виведення. Наприклад, з’являється можливість виконувати з портами арифметичні чи логічні операції. Недоліками цього способу є те, що, по-перше, у низці випадків може збільшитися час доступу до порту, а, по-друге, частина адресного простору пам’яті передається портам.

У 16-розрядному МП, наприклад, i8086 є два види команд введення/виведення:

– з прямою адресацією портів;

– з непрямою адресацією портів.

У першому випадку команда містить пряму 8-розрядну адресу порту, що дозволяє адресувати $2^8 = 256$ 8-розрядних портів чи 128 16-розрядних.

У другому випадку, адреса порту міститься в 16-розрядному регістрі DX, що дає можливість адресувати $2^{16} = 65\,536$ 8-розрядних портів чи 32 768 16-розрядних.

У МПС на основі МК, наприклад, AVR, порти, за допомогою яких відбувається обмін інформацією із зовнішнім пристрєм, містяться у МК. Для звернення до портів використовують команди з прямою адресацією.

6.1.5. Підключення пристрою введення/виведення до системної шини та зовнішніх пристроїв

Це питання розглянемо на прикладі обміну даними в МПС з використанням «інтелоподібного мікропроцесора», наприклад, i8086 та програмованого паралельного інтерфейсу – i8255, спрощену структуру якого наведено на рис. 6.3.

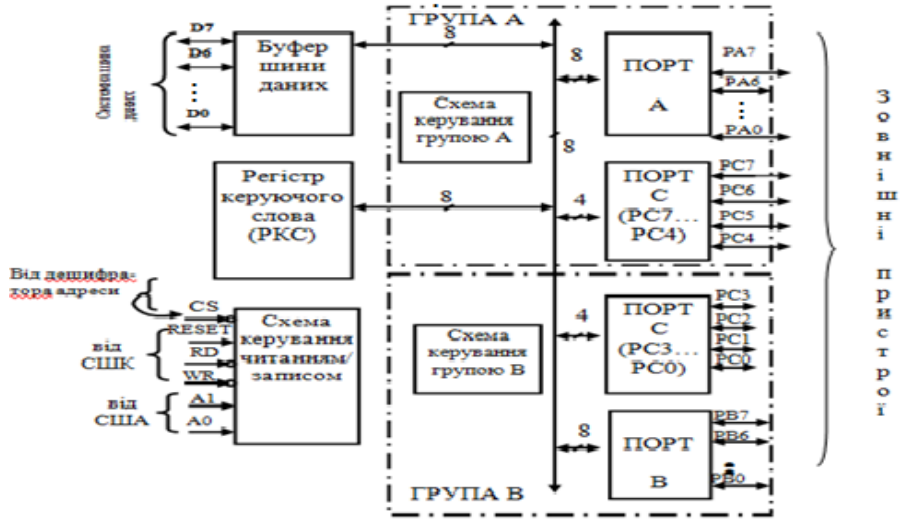


Рис. 6.3. Спрощена структурна схема програмованого паралельного інтерфейсу i8255

Інтерфейс має 2 виводи для подачі живлення, 8 – для зв'язку із системною шиною даних, 6 – для подачі керувальних сигналів і 24 виводи, що можуть окремо програмуватися по дві групи з дванадцяти виводів і використовуватися в трьох режимах роботи: 0, 1 і 2.

У режимі 0 кожну групу з дванадцяти виводів може бути запрограмовано на просте (нестробоване) введення чи виведення.

У режимі 1 вісім виводів кожної групи програмуються на стробоване введення чи виведення, а інші використовуються для керування програмним обміном чи обміном за перериванням.

У режимі 2 мікросхема переважно використовується як двонапрямний 8-розрядний канал обміну даними через порт *A*, який керується сигналами п'яти виводів порту *C*.

Режим обміну задається програмно пересиланням у регістр керувального слова інтерфейсу керувального слова відповідного формату.

Стан інтерфейсу також контролюється програмою за допомогою зчитування слова стану (порт *C*).

Програмований паралельний інтерфейс складається з портів введення/виведення: *A*, *B* і *C*, схем керування групами ліній *A* і *B*, буфера шини даних, регістра керувального слова і схеми керування читанням/записом. Порти *A*, *B* і *C* призначено для прийому і збереження інформації, що пересилається між МП і зовнішнім пристроєм.

Останні підключаються до програмованого паралельного інтерфейсу за допомогою виводів портів *A* (PA7...PA0), *B* (PB7...PB0) і *C* (PC7...PC0). Порт *A* містить вхідний і вихідний 8-розрядні регістри з формувачами та може використовуватися для введення чи виведення інформації у всіх трьох режимах. Порт *B* складається з 8-розрядного регістра введення/виведення з формувачами та може працювати на введення чи виведення даних у режимах 0 і 1. Порт *C* має два 4-розрядних регістри з формувачами. Ці регістри можуть застосовуватися для введення чи виведення інформації в режимі 0. Під час роботи портів *A* і *B* у режимах 1 чи 2 більша частина виводів порту *C* використовуються для прийому і видачі керувальних сигналів обміну, а регістри порту *C* виконують функцію регістрів стану. Особливістю порту *C* є можливість встановлення/скидання його окремих розрядів.

Буфер шини даних є тристабільним, двонапрямним і призначений для підключення внутрішньої шини даних програмованого паралельного інтерфейсу до системної шини МПС. Обмін інформацією відбувається через 8-розрядний буфер шини даних за командами МП. Час передачі/прийому визначається тривалістю сигналів «Запис» – $\overline{WR}$ та «Читання» – $\overline{RD}$. МП передає інтерфейсу дані для зовнішнього пристрою чи керувальні слова для програмування

паралельного програмованого інтерфейсу, а приймає інформацію від зовнішніх пристроїв чи слова стану інтерфейсу. Схеми керування групами шин *A* і *B* відповідно до вмісту керувального слова, яке записано у регістр керувального слова, задають режим роботи портів паралельного програмованого інтерфейсу.

Схема керування читанням/записом керує всіма внутрішніми та зовнішніми пересиланнями інформації. Для цього використовуються сигнали: $\overline{CS}$ – вибір кристала паралельного програмованого інтерфейсу, що звичайно надходить від селектора (дешифратора) адреси; $\overline{RD}$ – читання, що дозволяє передачу на системну шину даних інформацію від зовнішнього пристрою чи слово стану паралельного програмованого інтерфейсу (режим введення); $\overline{WR}$ – запис, керує передачею даних чи керувального слова із системної шини даних (режим виведення); A1, A0 – значення молодших розрядів системної адресної шини, що необхідні для вибору одного з портів паралельного програмованого інтерфейсу чи регістра керувального слова; RESET – скидання, що обнуляє усі внутрішні регістри інтерфейсу і перемикає його порти в режим 0 для введення інформації.

Вплив зовнішніх керувальних сигналів на роботу паралельного програмованого інтерфейсу пояснює табл. 6.1.

Таблиця 6.1. **Напрямок і вид обміну даними між МП і зовнішнім пристроєм**

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A1	A0	Напрям та вид обміну
					МП→ЗВПР (виведення, запис, передача)
0	1	0	0	0	Запис даних до порту A
0	1	0	0	1	Запис даних до порту B
0	1	0	1	0	Запис даних до порту C
0	1	0	1	1	Запис даних до РКС
					ЗВПР → МП (введення, читання, прийом)
0	0	1	0	0	Читання порту A
0	0	1	0	1	Читання порту B
0	0	1	1	0	Читання порту C
1	x	x	x	x	Схему відключено (високоімпедансний стан)
0	0	1	1	1	Заборонений стан

Якщо CS = 1 чи RD = WR = 1, то незалежно від інших сигналів паралельний програмований інтерфейс переходить у третій стан. Забороненою є комбінація A0 = A1 = 1, RD = 0, WR = 1, CS = 0 (читання регістра керувального слова заборонено).

Керувальні сигнали для системної шини виробляє МП. Вибір режиму роботи інтерфейсу здійснюється записом керувального слова до регістра керувального слова у будь-якому місці програми.

Це дозволяє обслуговувати одним інтерфейсом різні периферійні пристрої у визначеному порядку. Режими роботи каналів *A* і *B* встановлюються незалежно та автономно, а каналу *C* – залежно від режимів роботи каналів *A* і *B*.

Під час кожної зміни режиму роботи кожного з каналів регістри портів скидаються в нуль. У разі встановлення режимів 1 і 2 обнуляється регістр стану, функцію якого виконує порт *C*. Тому потрібно робити початкове встановлення розрядів цього регістра відповідно до режиму роботи каналів *A* і *B*.

На рис. 6.4 наведено підключення паралельного програмованого інтерфейсу до системної шини та зовнішніх пристроїв.

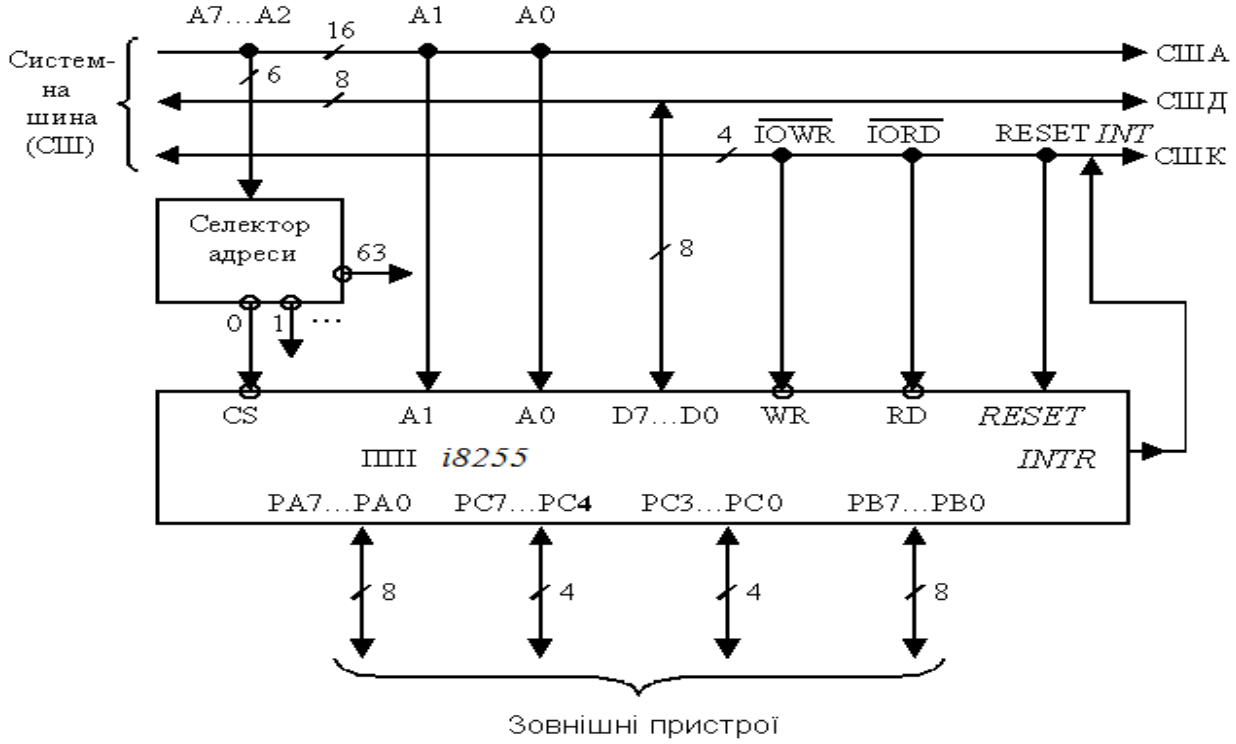


Рис. 6.4. Підключення ППІ до системної шини та зовнішніх пристроїв

6.1.6. Програмування модуля введення/виведення

Більшість систем введення/виведення є програмовані, тобто в системі команд МП і МК є команди для керування обміном.

У МПС на основі МП до процесу програмування належить передача керувальних слів у регістри керувального слова і читання слова стану інтерфейсу.

Керувальні слова можуть задавати інтерфейсу:

– режим роботи, в якому буде використовуватися інтерфейс;
– швидкість обміну (для послідовного інтерфейсу);
– формати даних, якими обмінюються;

– порти, через які буде вестися обмін;

– напрямок обміну і т. ін.

Розглянемо особливості програмування обміну даними між МП та зовнішніми пристроями на прикладі програмування ППІ типу i8255.

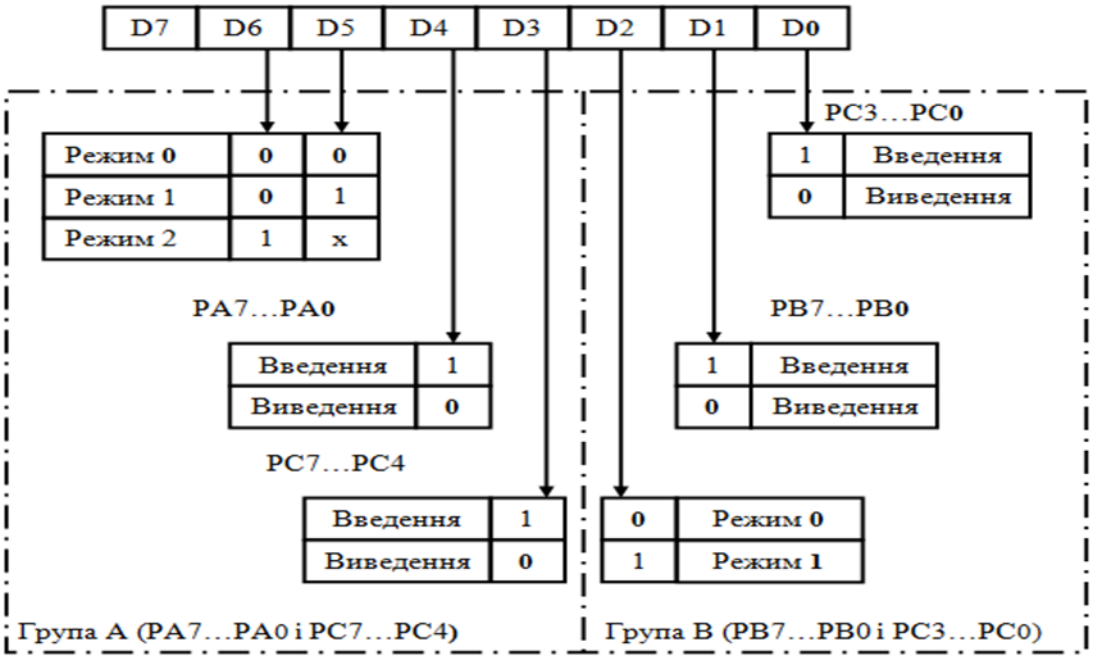
Для програмування ППІ застосовують одне з двох керувальних слів:

– керувальне слово режиму;

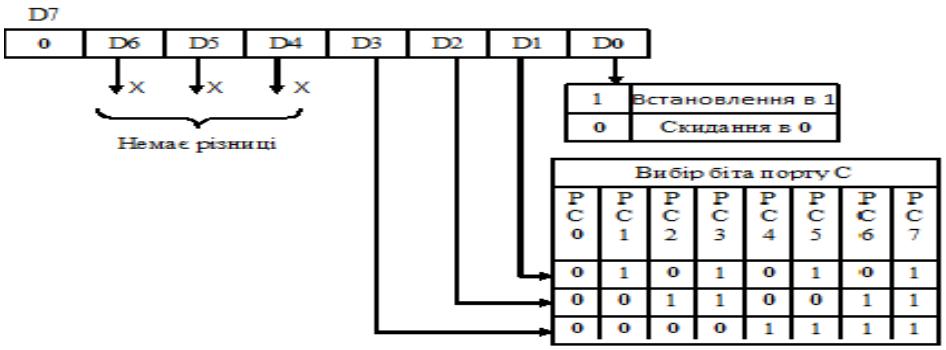
– керувальне слово встановлення/скидання бітів порту *C*.

Стан інтерфейсу у разі його роботи в режимах 1, 2 відображає регістр порту *C*.

Формати керувальних слів ППІ типу i8255: керувальне слово режиму і керувальне слово встановлення/скидання показано на рис. 6.5.



а



б

Рис. 6.5. Формати керувальних слів: *а* – керувальне слово режиму; *б* – керувальне слово встановлення/скидання бітів порту *C*

Наявність у керувальному слові режиму D7 = 1 вказує ППІ про налаштування режиму його роботи. Для запису керувального слова до регістра керувального слова використовують команду виведення. Для МП типу i8086 керувальне слово попередньо записується до акумулятора, а потім виводиться у відповідний інтерфейс із зазначенням адреси регістра керувального слова (A1 = A0 = 1).

6.1.7. Режими роботи програмованого інтерфейсу

Загальні відомості про режими роботи

Паралельний програмований інтерфейс типу i8255 може працювати в одному з трьох режимів: 0, 1 або 2.

Режим 0 – просте нестробоване введення/виведення

У цьому режимі виконується просте (нестробоване) введення чи виведення інформації через кожен з трьох каналів (портів). Сигнали керування (підтвердження) про встановлення зв'язку із зовнішнім пристроєм не вимагаються. Інформація, що вводиться, у буфері порту не запам'ятовується, а виведена – зберігається у вихідних регістрах портів до запису нової інформації чи до зміни режиму роботи. Особливості програмування ППІ в цьому режимі описано у [2].

Режим 1 – стробоване введення/виведення

У цьому режимі дані передаються через канали *A* і *B*, а частина ліній каналу *C* здійснює керування обміном. Кожен з каналів *A* і *B* може використовуватися для введення чи виведення 8-розрядних даних із запам'ятовуванням переданої інформації в регістрах портів. Цей режим надає такі можливості:

– запрограмувати порти *A* і *B* на стробоване введення/виведення інформації. У цьому разі кожен з портів може працювати лише в *одному напрямку*: на введення чи на виведення;
– запрограмувати один з портів *A* чи *B* на стробоване введення/виведення у режимі 1, а інші 13 ліній (вісім порту *B* чи *A* та п'ять порту *C*) – на роботу в режимі 0;
– запрограмувати порти *A* і *B* на стробоване введення/виведення у режимі 1, а 2 лінії порту *C*, що залишилися не зайняті керувальними сигналами, використовувати для обміну в режимі 0.

Обмін інформацією між МП і зовнішнім пристроєм може здійснюватися програмно чи за перериванням. Особливості програмування ППІ в режимі 1 описано у [2].

Режим 2 – двонапрямне стробоване введення/виведення

У цьому режимі забезпечується обмін інформацією однією 8-розрядною *двонапрямною шиною* введення/виведення, в якості якої може працювати тільки порт *A*. Для керування обміном використовуються п’ять ліній порту *C*, що виконують функції, аналогічні режиму 1. Вхідні дані фіксуються у вхідному, а вихідні – у вихідному регістрах порту *A*. Як тригер дозволу переривання під час введення інформації використовується тригер INTE2, встановлення та опитування якого робиться через розряд С4 порту *C*. Аналогічним тригером у разі виведення інформації є тригер INTE1, зв’язок з яким здійснюється через розряд С6 порту *C*. Призначення інших виводів регістра *C* таке саме, як у режимі 1. Сигнал «Запит переривання *A*» (INTRA) використовується для переривання роботи МП як у разі введення, так і у разі виведення даних. Особливості програмування ППІ в режимі 2 описано у [2].

Більш детально використання інтерфейсу типу i8255 розглянуто у [2].

У МПС на основі МК, наприклад, сімейства AVR, є значна кількість паралельних та послідовних портів. Для їх програмування використовуються відповідні команди та керувальні регістри.

6.2. Архітектура паралельних портів введення/виведення AVR-мікроконтролерів

6.2.1. Загальна характеристика паралельних портів

МК сімейства AVR мають модулі введення/виведення, апаратну частину яких виконано на основі портів введення/виведення (регістрів). Кожен порт має певну кількість виводів, через які МК може приймати або передавати цифрові сигнали [3; 4].

Задання напрямку передачі даних через будь-який контакт введення/виведення виконується програмно. На входах портів знаходяться тригери Шмітта. Якщо лінії сконфігуровані як вхідні, то є можливість підключення внутрішнього підтягуючого резистора з опором 35...120 кОм між входом і шиною живлення V_{CC}. Цей вивід може служити джерелом струму, якщо між виводом із задіяним внутрішнім підтягуючим резистором і спільною шиною підключити навантаження.

У портах введення/виведення МК сімейства реалізовано функцію «читання/модифікація/запис». Завдяки цьому, використовуючи команди SBI та CBI, можна виконувати операції над будь-яким виводом порту, не впливаючи на інші виводи. Це стосується зміни режиму роботи контакту введення/виведення, зміни вихідного значення і зміни стану внутрішнього підтягуючого резистора (для входів).

МК з кожної групи моделей сімейства мають різну кількість портів і,

відповідно, контактів введення/виведення [3; 4].

В усіх МК сімейства переважна більшість контактів введення/виведення мають додаткові (альтернативні функції) та використовуються відповідними периферійними пристроями МК.

6.2.2. Звернення до паралельних портів введення/виведення

Звернення до портів виконується через регістри введення/виведення. За деякими винятками, під кожен порт в адресному просторі введення/виведення зарезервовано по 3 адреси [3]. За цими адресами розміщуються три регістри: регістр даних порту PORTx, регістр напрямку даних DDRx і регістр виводів порту PINx. Дійсні назви регістрів отримуються підстановкою назви порту замість символу «x». Відповідно, регістри порту A називаються PORTA, DDRA, PINA, порту B – PORTB, DDRB, PINB і т. ін.

Регістри PINx, де x = A/B/D/ і т. ін., фізично не є регістрами. За цими адресами здійснюється доступ до значень сигналів на виводах порту.

Під час виведення у порт відбувається запис у відповідний регістр даних порту PORTx. Під час введення виконується читання регістра виводів порту PINx (сигналів, які присутні на виводах порту). Під час читання регістра PORTx у МК вводяться дані з регістра-защіпки порту. Під час знаходження МК у стані скидання виводи всіх портів перебувають у третьому (високоімпедансному) стані – Hi-Z [3].

Адреси регістрів, що належать до портів введення/виведення, наведено в [3].

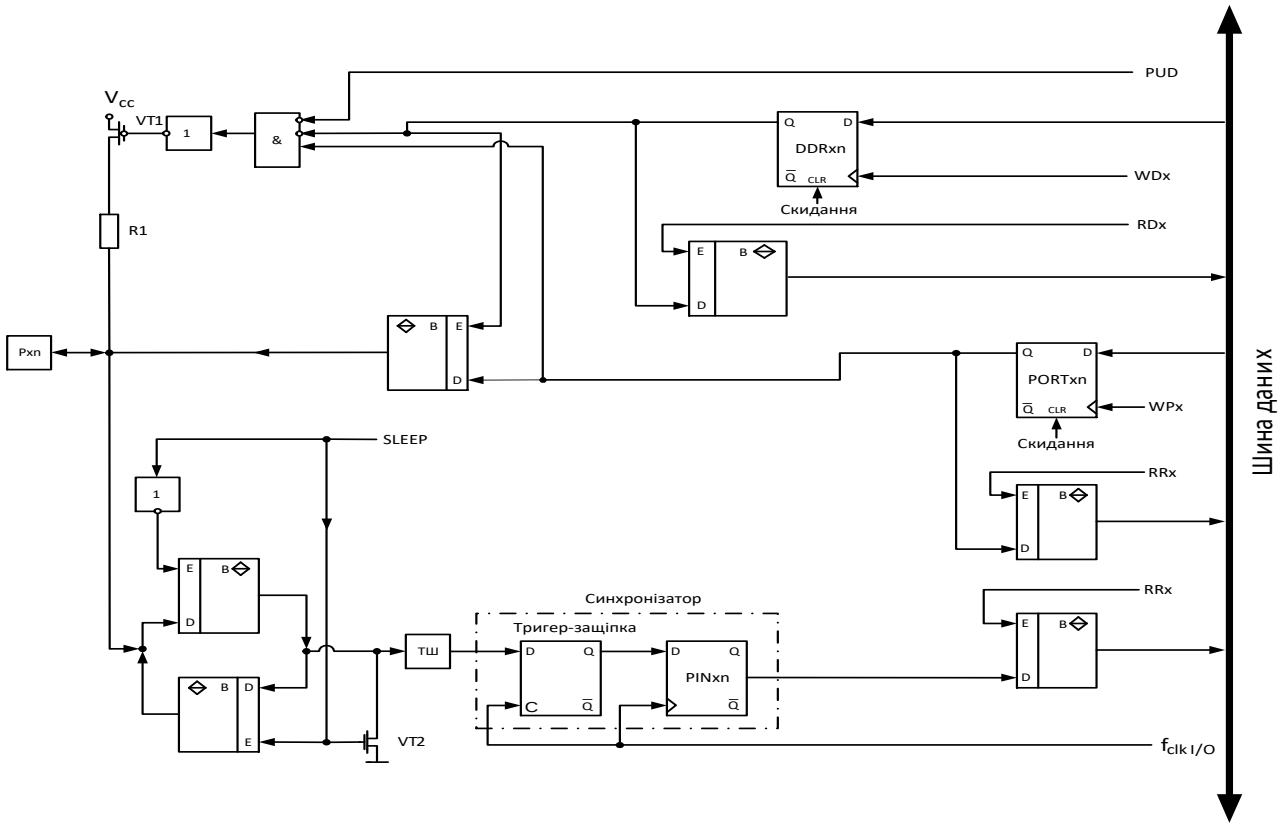
6.2.3. Структура паралельних портів введення/виведення

Спрощену структурну схему одного з каналів порту введення/виведення Pxn під час його роботи як цифрового входу/виходу загального призначення наведено на рис. 6.6.

Кожному каналу порту відповідають розряди трьох регістрів введення/виведення: PORTxn (регістр PORTx), DDRxn (регістр DDRx) і PINxn (регістр PINx). Дійсні назви розрядів регістрів отримуються підстановкою назви порту замість символу «x» і номера розряду замість символу «n».

Порядковий номер виводу порту відповідає порядковому номеру розряду регістрів цього порту.

Якщо розрядність порту менше восьми, у регістрах порту використовується відповідна кількість молодших розрядів. Незадіяні старші розряди регістрів доступні тільки для читання і завжди містять нуль.



PUD – відключення резистора, який підтягує; SLEEP – керування «сплячим» режимом; $f_{clk\ I/O}$ – тактовий сигнал підсистеми введення/виведення; WDx – запис регістра DDRx; RDx – читання регістра DDRx; WPx – запис регістра PORTx; RRx – читання регістра PORTx; RPx – читання виводів порту Px.

Примітка. Сигнали WPx, WDx, RRx, RPx, RDx – спільні для всіх виводів одного порту; сигнали $f_{clk\ I/O}$, SLEEP та PUD – спільні для всіх портів МК.

Рис. 6.6. Структурна схема контакту введення/виведення

У наведену структуру контакту окрім цифрових тригерів та логічних елементів входять буфери – В, вихідні сигнали яких мають три стани: логічна одиниця; логічний нуль та третій (високоімпедансний) стан. Використання останніх пояснюється тим, що виходи цих буферів підключено до спільних шин. Щоб уникнути спотворення сигналів на шині, тільки один з буферів у конкретний момент часу під впливом керуючого сигналу «Е» буде активним. Виходи інших (не обраних) буферів на час передачі переводяться у третій стан.

Розряд DDRxn регістра DDRx визначає напрямок передачі даних через контакт введення/виведення. Якщо цей розряд встановлено в одиницю, то *n*-й вивід порту є виходом, якщо ж скинуто в нуль – входом.

Розряд PORTxn регістра PORTx виконує подвійну функцію. Якщо вивід функціонує як вихід (DDRxn = 1), цей розряд визначає стан виводу порту. Якщо розряд встановлено в одиницю, на виводі встановлюється напруга ВИСОКОГО рівня. Якщо розряд скинуто в нуль, на виводі встановлюється напруга НИЗЬКОГО рівня.

Якщо ж вивід функціонує як вхід (DDRxn = 0), розряд PORTxn визначає стан внутрішнього підтягуючого резистора для цього виводу. Під час встановлення розряду PORTxn в одиницю підтягуючий резистор підключається між виводом МК і живленням, за умови, що біт PUD = 0.

Керування підтягуючими резисторами здійснюється на двох рівнях. Загальне керування для всіх виводів портів здійснюється розрядом PUD регістра спеціальних функцій (англ. SFIOR) або регістра керування МК MCUCR [3; 4]. Формати цих регістрів наведено на рис. 6.7.

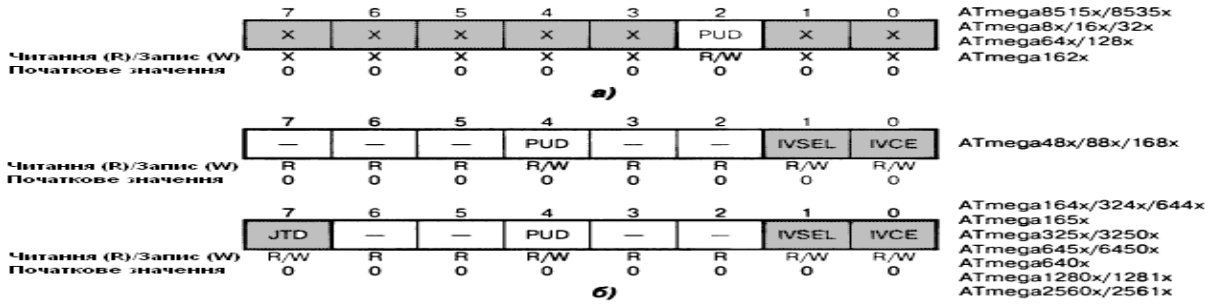


Рис. 6.7. Регістри керування підтяжкою SFIOR (*а*) та MCUCR (*б*)

Якщо розряд PUD у початковому стані скинуто в нуль, стан підтягуючих резисторів для кожного входу порту буде визначатися станом розрядів PORTxn, за умови, що відповідний розряд DDRxn скинуто в нуль. Якщо ж розряд PUD встановлено в одиницю, підтягуючі резистори відключаються від усіх виводів портів МК.

6.2.4. Конфігурування виводів паралельних портів введення/виведення

Різні можливості конфігурування виводів портів введення/виведення наведено у табл. 6.2.

Таблиця 6.2. **Конфігурації виводів портів**

DDRxn	PORTxn	PUD*	Функція виводу	Резистор	Примітки
0	0	X	Вхід	Відключений	Третій стан (Hi-Z)**
0	1	0	Вхід	Підключений	Під час підключення навантаження між виводом і спільним проводом вивід є джерелом струму
0	1	1	Вхід	Відключений	Третій стан (Hi-Z) для виходу
1	0	X	Вихід	Відключений	Вихід скинуто в нуль
1	1	X	Вихід	Відключений	Вихід встановлено в одиницю
* Відсутній у моделях ATmega161x. ** Стан виводів порту в разі скидання.					

Стан виводу МК (незалежно від розряду DDRxn) може бути отримано за допомогою читання розряду PINxn регістра PINx, при цьому варто пам'ятати, що між дійсною зміною сигналу на виводі та зміною розряду PINxn є затримка. Ця затримка вноситься вузлом синхронізації, що складається, як показано на рис. 6.8, з розряду тригера PINxn і додаткового тригера-защіпки.

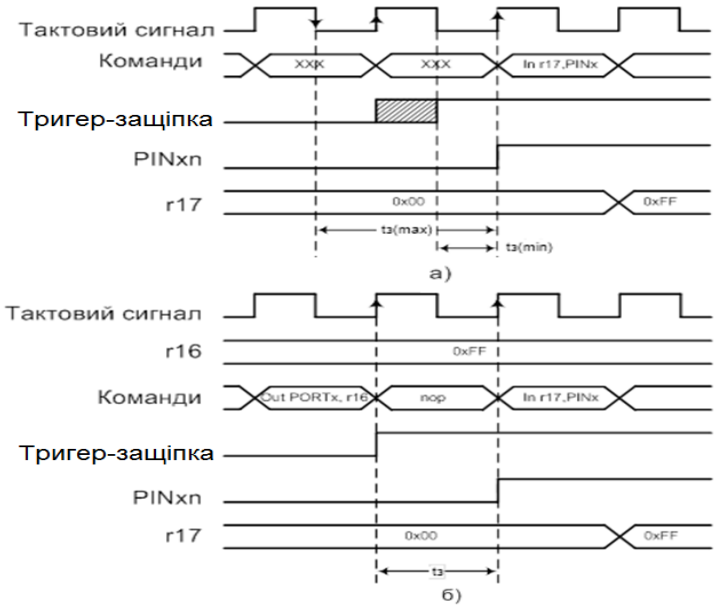


Рис. 6.8. Синхронізація під час читання стану виводу:
а – під час зчитування стану розряду PINxn; *б* – під час повторного зчитування

Значення сигналу на виводі МК фіксується тригером-защіпкою за високим рівнем тактового сигналу і переписується потім у розряд PINxn за наростаючим фронтом тактового сигналу ($f_{CLKI/O}$). Відповідно, величина затримки може становити від 0,5 до 1,5 періоду системного тактового сигналу, як показано на рис. 6.8*а*.

З цієї ж причини між операціями зміни, виведення і повторного зчитування станів виводу необхідно вставляти команду NOP. Оскільки команда OUT записує інформацію в тригер PORTxn за додатним фронтом тактового сигналу, затримка в цьому випадку дорівнює одному періоду тактового сигналу (рис. 6.8*б*).

6.2.5. Приклад конфігурування одного з паралельних портів мікроконтролера

У наведеному нижче прикладі виходи 0 і 1 порту B встановлюються в одиницю, виходи 2 і 3 – скидаються в нуль. Виводи 4...7 порту конфігуруються як входи, при цьому до виводів 6 і 7 підключаються підтягуючі резистори.

Приклад на мові Асемблера

```
…
ldi r16, (1<<PB7) | (1<<PB6) | (1<<PB1) | (1<<PB0)
ldi rl7, (1<<DDRB3) | (1<<DDRB2) | (1<<DDRB1) | (1<<DDRB0)
```

```
out P0RTB, rl6          ; Задання стану виходів і
                                ; підтягуючих резисторів
out DDRB, rl7;          ; Задання режимів роботи виводів
nop                ; Порожня операція (затримка на один такт)
in r16, PINB          ; Читання стану виводів порту
```

Приклад на мові С

```
unsigned char i;
…
/* Задання стану виходів і підтягуючих резисторів */
/* Задання режимів роботи виводів */
PORTB = (1<<PB7) | (1<<PB6) | (1<<PB1) | (1<<PB0);
DDRB = (1<< DDRB3) | (1<< DDRB2) | (1<< DDRB1) | (1<<DDRB0);
  _NOP(); /* Функція (порожня операція – затримка на один такт) */
rl6 = PINB;/* Читання стану виводів порту */
…
```

Переважна більшість контактів введення/виведення МК сімейства мають додаткові функції і можуть використовуватися різними периферійними пристроями МК. У цьому випадку користувач повинен або самостійно задавати конфігурацію виводу, або вивід конфігурується автоматично під час включення відповідного периферійного пристрою [3; 4].

6.3. Архітектура послідовного інтерфейсу AVR-мікроконтролерів

6.3.1. Загальна характеристика інтерфейсу

Більшість МК сімейства AVR мають модуль послідовного програмованого універсального асинхронного приймача/передавача (УАПП), або універсального синхронно/асинхронного приймача/передавача (УСАПП). У деяких моделях міститься по декілька таких модулів [3; 4].

В іноземній літературі модуль УАПП позначається як UART (Universal Asynchronous Receiver and Transmitter) , а модуль УСАПП – як USART (Universal Synchronous and Asynchronous Receiver and Transmitter).

Усі AVR-мікроконтролери сімейства Mega мають модулі УСАПП, які під час роботи в асинхронному режимі сумісні з модулями УАПП як за розміщенням розрядів керуючих регістрів, так і за функціонуванням. У деяких моделях модулі УСАПП можуть використовуватись в якості ведучого шини SPI [3; 4].

Усі модулі УАПП/УСАПП підтримують дуплексний обмін за послідовним каналом, при цьому швидкість передачі даних може змінюватись у доволі широких межах. У модулях УАПП пакет даних може бути 8- чи 9-розрядним, а в

модулях УСАПП його довжина може складати від 5-ти до 9-ти розрядів. Ще однією особливістю модулів УСАПП порівняно з УАПП є наявність схем формування та контролю парності.

Модулі УАПП/УСАПП можуть виявляти такі не типові ситуації:

- переповнення;
- помилку кадрування;
- некоректний старт-біт.

У модулях є також корисна функція – фільтрація завад.

Для програмування модулів є три переривання, запит на генерацію яких формується під час виникнення таких подій:

- передачу завершено;
- регістр даних передавача порожній;
- прийом завершено.

Виводи МК, що використовуються модулями УАПП/УСАПП, є лініями портів введення/виведення загального призначення. Як приклад, виводи деяких МК, що використовуються модулями УСАПП сімейства Mega, та їх функції наведено у [3; 4].

6.3.2. Опис структури модулів послідовного програмованого універсального асинхронного приймача/передавача та програмованого універсального синхронно/асинхронного приймача/передавача

Спрощену структурну схему модуля УАПП/УСАПП наведено на рис. 6.9.

Елементи, які виділені на рисунку темним кольором, є тільки у модулів УСАПП.

Модуль має три частини: блок тактування, блок передавача та блок приймача. Блок тактування включає схему синхронізації, яка використовується під час роботи у синхронному режимі, і контролер швидкості передачі. У модулях УАПП блок тактування складається тільки з контролера швидкості передачі.

Блок передавача має однорівневий буфер UDR, регістр зсуву, схему формування біта парності (тільки УСАПП) і схему керування.

Блок приймача включає схеми відновлення тактового сигналу і даних; схему контролю парності (тільки УСАПП); буфер UDR: дворівневий в УСАПП та однорівневий в УАПП; регістр зсуву, а також схему керування.

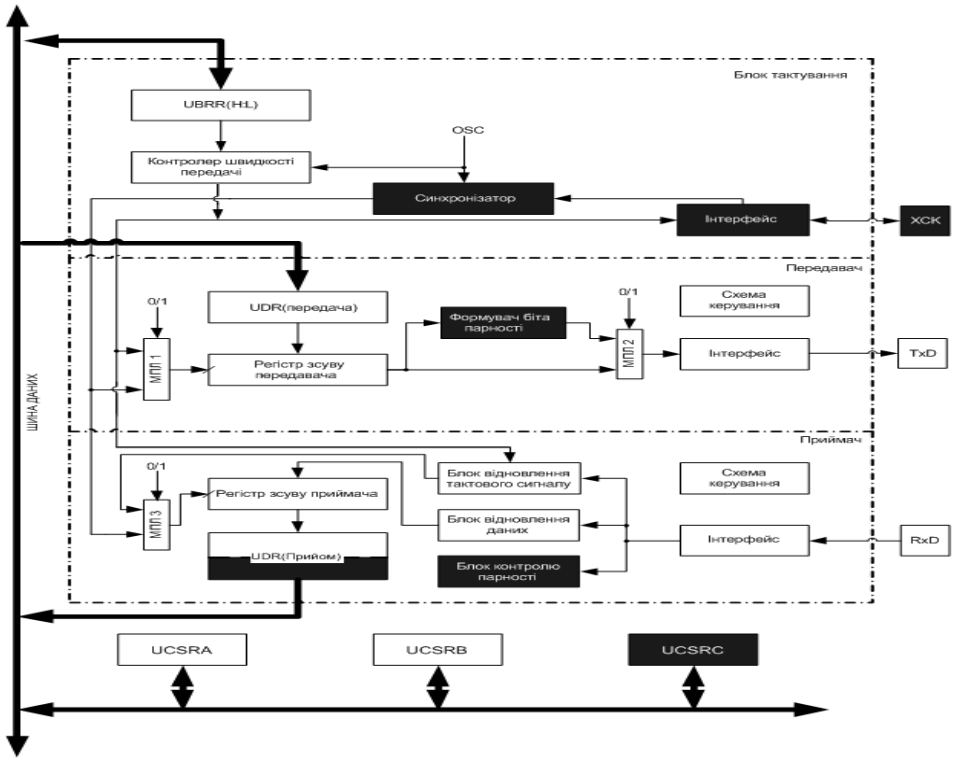


Рис. 6.9. Спрощена структурна схема модуля UART/USART

Буферні регістри приймача і передавача розміщуються за однаковою адресою простору регістрів введення/виведення і позначаються як регістр даних UDR (Universal Data Register – UDRn, де n = 0, 1, 2 або 3). У цих регістрах зберігаються молодші 8 розрядів даних, які приймаються чи передаються. Під час читання UDR виконується звернення до буферного регістра приймача, а у разі запису – до буферного регістра передавача. Розміщення регістрів даних для деяких моделей МК наведено у [3; 4].

Дворівневий буфер приймача у модулях УСАПП є FIFO-буфером (First In First Out – першим зайшов, першим вийшов). Зміна стану буфера відбувається у разі будь-якого звернення до регістра UDR. Тому не варто використовувати регістр UDR як операнди команд типу «читання/модифікація/запис» (SBI та CBI) та команди перевірки SBIC та SBIS, які також змінюють стан буфера приймача.

Для керування модулями УСАПП використовуються три регістр: UCSRA (UCSRnA), UCSRB (UCSRnB) та UCSRC (UCSRnC), де n = 0, 1, 2 або 3. Адреси цих регістрів, їх формати та опис окремих розрядів наведено у [3; 4].

6.3.3. Швидкість прийому/передачі даних

На рис. 6.10 як приклад наведено структурну схему блока синхронізації модуля УСАПП МК Mega 128.

На схемі використовуються такі сигнали:

- txclk – синхронізація передавача (внутрішній сигнал);
- rxclk – синхронізація приймача (внутрішній сигнал);
- xcki – вхідний сигнал від виводу XCK (внутрішній сигнал), який використовується для синхронної підлеглої (веденої) роботи;
- xcko – вихідний сигнал синхронізації до виводу XCK (внутрішній сигнал), який використовується у ведучому синхронному режимі;
- f_{CLK} – частота системного тактового генератора.

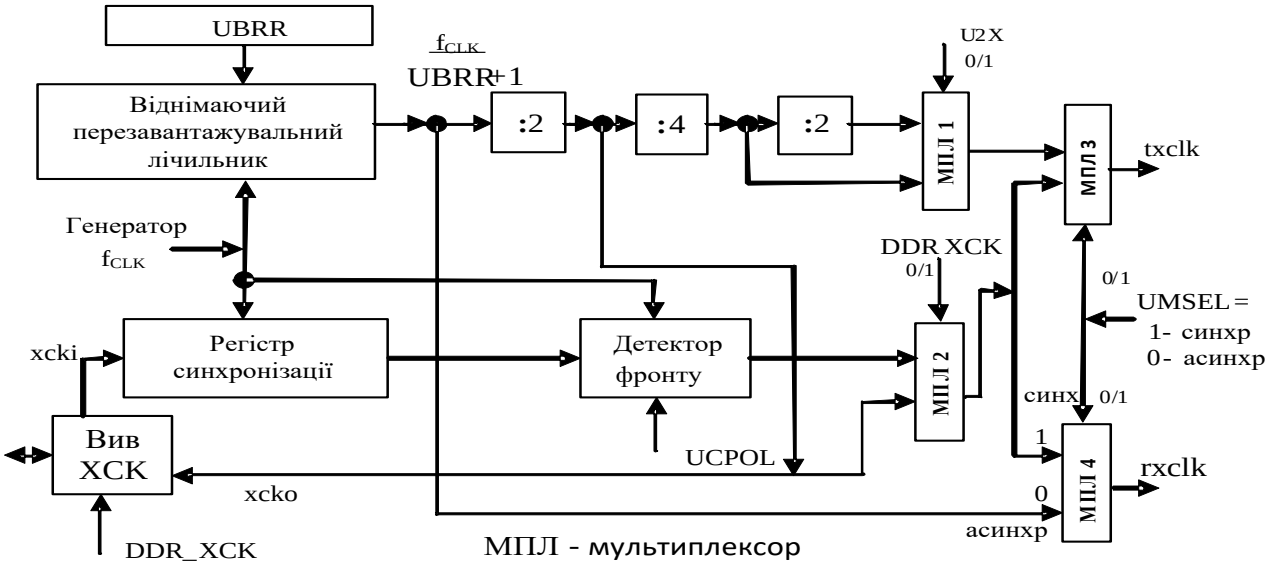


Рис. 6.10. Структурна схема блока синхронізації модуля УСАПП

В асинхронному режимі, а також у синхронному режимі під час роботи у якості ведучого, швидкість прийому та передачі даних задається контролером швидкості передачі, що функціонує як дільник системного тактового сигналу з програмованим коефіцієнтом ділення. Коефіцієнт визначається вмістом регістра контролера UBRR (UBRRn). Після дільника у блок приймача сигнал надходить одразу, а у блок передавача – через додатковий дільник, коефіцієнт ділення якого: 2, 8 чи 16 залежить від режиму роботи модуля УАПП/УСАПП.

УСАПП підтримує чотири режими роботи синхронізації:

- нормальна асинхронна;
- асинхронна з подвоєнням швидкості;
- ведуча синхронна;
- та підлегла (ведена) синхронна.

Для програмування асинхронного чи синхронного режиму роботи призначено біт UMSEL у регістрі керування та статусу UCSRC. Якщо UMSEL = 0, то модуль буде працювати в асинхронному режимі, якщо UMSEL = 1, то у синхронному.

В асинхронному режимі можливе подвоєння швидкості передачі. Для цього призначено біт U2X регістра UCSRA [3; 4].

У разі використання синхронного режиму біт DDR_XCK у регістрі напрямку даних для виводу XCK задає, чи буде синхронізація внутрішньою: DDR_XCK = 1 (ведучий режим) чи зовнішньою: DDR_XCK = 0 (ведений режим). Вивід XCK активний тільки у разі використання синхронного режиму.

Внутрішня синхронізація використовується для асинхронного та ведучого синхронного режимів роботи. Зовнішня синхронізація використовується у синхронному підлеглому (веденому) режимі роботи.

Регістр UBRR є 12-розрядним та фізично розміщений у двох регістрах введення/виведення: UBRRH (UBRRnH) та UBRRL (UBRRnL). Адреси і назви цих регістрів для деяких моделей мікроконтролерів МК наведено у [3; 4].

Регістр UBRRH (UBRRnH у низці моделей, наприклад, ATmega8x, ATmega8515x, ATmega16x, ATmega162x, ATmega32x та ATmega323x) має однакову адресу з регістром керування UCSRC (UCSRnC). Тому у разі звернення за цими адресами, необхідно виконати низку додаткових дій для вибору одного з цих регістрів.

Під час запису один з цих двох регістрів визначається станом старшого розряду числа, яке записано у регістр UCSRC (UCSRnC) – URSEL (URSELn). Якщо цей розряд скинуто у нуль, змінюється вміст регістра UBRRH (UBRRnH). Якщо старший розряд числа встановлено в одиницю, змінюється вміст регістра керування UCSRC (UCSRnC).

Для вибору одного з двох розглянутих вище регістрів під час читання використовується відповідна часова послідовність. Під час першого звернення за вказаною адресою повертається значення регістра UBRRH (UBRRHn). У разі повторного звернення за цією адресою у наступному машинному циклі повертається значення регістра UCSRC (UCSRnC).

Під час роботі роботи в асинхронному режимі швидкість обміну визначається не тільки вмістом регістра UBRR, але й станом розряду U2X (U2Xn) регістра UCSRA (UCSRnA) [3].

Якщо цей розряд встановлено в одиницю, коефіцієнт ділення попереднього дільника зменшується у два рази, а швидкість обміну відповідно подвоюється. Під час роботи у синхронному режимі цей розряд має бути скинуто.

Швидкість обміну в модулях УАПП/УСАПП визначається такими формулами:

– звичайний асинхронний режим (U2Xn = 0):
  BAUD = fclк/(16(UBRR + 1)); (6.1)
– пришвидшений асинхронний режим (U2Xn = 1):
  BAUD = fclк/(8(UBRR + 1)); (6.2)
– синхронний ведучий режим:
  BAUD = fclк/(2(UBRR + 1)), (6.3)

де BAUD – швидкість передачі у бодах; fclк – тактова частота МК; UBRR – вміст регістра контролера швидкості передачі: 0...4095.

ЯК приклад у [3] наведено значення регістра UBRR, що дозволяють отримати стандартні для асинхронного режиму швидкості передачі під час використання деяких резонаторів, а також значення похибок, що отримуються, відносно стандартних швидкостей.

Рекомендовано використовувати значення регістра UBRR, за яких отримувана швидкість передачі відрізняється від необхідного значення менше ніж на 0,5 %.

6.3.4. Формат кадру

Сукупність одного слова (байта) даних і додаткової інформації під час передачі називається кадром (рис. 6.11).

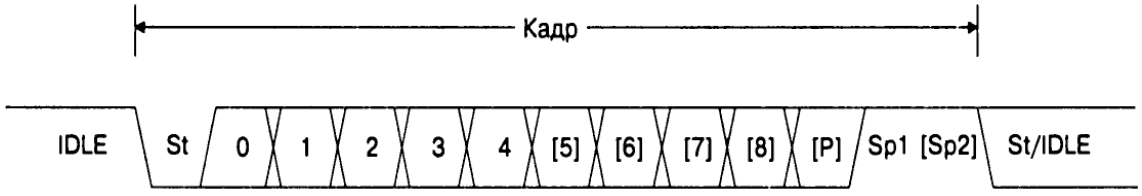


Рис. 6.11. Формат кадру

На рис. 6.11 використано такі скорочення:

- St – старт-біт, завжди має низький рівень;
- 0…8 – номер біта даних;
- P – біт паритету: парність або непарність;
- Sp1, Sp2 – стоп-біти, завжди мають високий рівень;
- IDLE – стан очікування, в якому призупинено обмін лінією RxD чи TxD.

У стані очікування на лінії має бути високий рівень.

До початку передачі в лінію передається високий рівень цифрового сигналу, що свідчить про справність лінії зв’язку.

Кадр починається із нульового старт-біта, який використовується схемою синхронізації приймача для підлаштування фази синхрочастоти приймача під час отримуваня посилки (біта). Далі передаються біти слова даних, починаючи з *молодшого* розряду. Після останнього старшого розряду слова даних передаються один або два стоп-біти. Якщо запрограмовано формування біта парності, то він розміщується між старшим розрядом слова даних і першим стоп-бітом.

Формат кадру визначається кількома розрядами регістрів UCSRB (UCSRnB) та UCSRC (UCSRnC) [3; 4]. Розмір слова даних у модулях USART визначається відповідно до табл. 6.3.

У модулях УАПП слово даних може бути тільки 8- або 9-розрядним, що визначається станом прапорця CHR9 (CHR9n) регістра UCSRB (UCSRnB). Якщо цей прапорець скинуто в нуль, розмір слова дорівнює 8 розрядам, якщо встановлено в одиницю – 9 розрядам.

Таблиця 6.3. **Визначення розміру слова даних у модулях УСАПП**

UCSZ2 (UCSZn2)	UCSZ1 (UCSZnl)	UCZ0 (UCSZn0)	Розмір слова даних
0	0	0	5 розрядів
0	0	1	6 розрядів
0	1	0	7 розрядів
0	1	1	8 розрядів
1	0	0	Зарезервовано
1	0	1	Зарезервовано
1	1	0	Зарезервовано
1	1	1	9 розрядів

Вибір кількості стоп-бітів у модулях УСАПП здійснюється за допомогою розряду USBS (USBSn) регістра UCSRC (UCSRnC). Якщо цей розряд скинуто в нуль, блок передавача у кінці посилки формує один стоп-біт. Якщо розряд встановлено в одиницю, блок передавача формує два стоп-біти. Слід зазначити, що приймачем другий стоп-біт ігнорується, і, відповідно, помилки кадрування виявляються тільки для першого стоп-біта.

Функціонування схеми контролю парності модулів УСАПП визначають розряди UPM1:UPM0 (UPMn1:UPMn0) регістра UCSRC (UCSRnC) відповідно до табл. 6.4.

Таблиця 6.4. **Керування контролем парності**

UPM1 (UPMnl)	UPM0 (UPMn0)	Режим роботи
0	0	Контроль відключено
0	1	Зарезервовано
1	0	Контроль включено, перевірка на парність (even parity)
1	1	Контроль включено, перевірка на непарність (odd parity)

Примітка. n = 0, 1, 2 чи 3.

Значення біта парності отримується за допомогою виконання операції «виключне АБО» над усіма розрядами слова даних, яке передається.

Якщо використовується перевірка на парність (even parity), то отриманий результат інвертується:

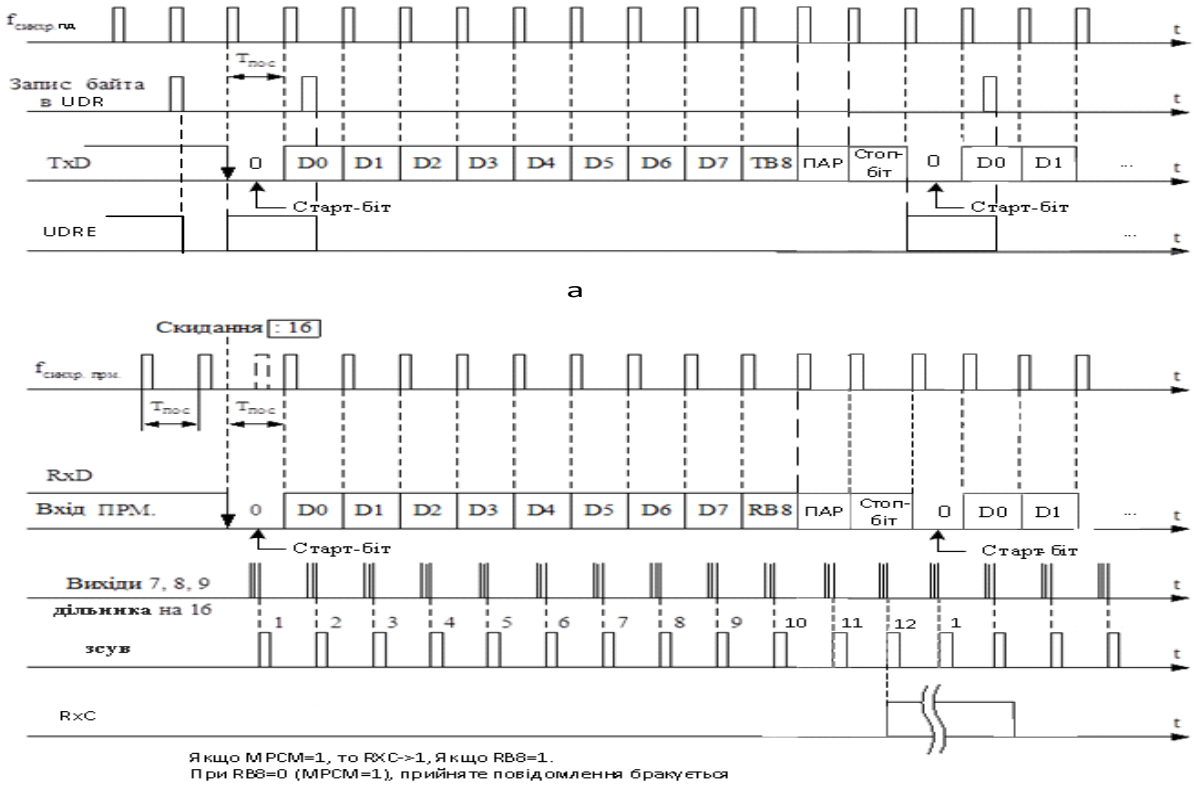
$$P_{EVEN} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1,$$
$$P_{ODD} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0. \quad (6.4)$$

Якщо контроль парності включено, біт парності вставляється передавачем між старшим розрядом даних, які передаються, і першим стоп-бітом.

6.3.5. Передача даних модулем

Для дозволу роботи передавача необхідно встановити в одиницю розряд TXEN (TXENn) регістра UCSRB (UCSRnB). Після цього вивід TXD (TXDn) підключається до передавача модуля УАПП/УСАПП і починає функціонувати, як вихід, незалежно від значень у регістрах керування напрямком обміну портом. Якщо використовується синхронний режим роботи, треба програмно перевизначити функціонування виводу ХСК (ХСКn) на введення або виведення.

В асинхронному режимі передача ініціюється записом даних, які передаються, у буферний регістр передавача – UDRn (рис. 6.12*а*).



а

б

Рис. 6.12. Часові діаграми роботи інтерфейсу в асинхронному режимі: *а* – передача; *б* – прийом даних

З регістра UDRn дані пересилаються у регістр зсуву передавача. Якщо використовуються 9-розрядний формат даних, значення розряду TXB8 (TXB8n) регістра UCSRB (UCSRnB) копіюється у 9-й розряд регістра зсуву. 9-й розряд даних має бути завантажений у розряд TXB8 (TXB8n) до запису байта даних у регістр даних.

Можливі два варіанти реакції на запис даних у регістр UDR:

– якщо запис у регістр здійснюється в той момент, коли передавач знаходиться у стані очікування (попередні дані вже передано), то у цьому випадку дані пересилаються у регістр зсуву одразу ж після запису в регістр UDR;
– якщо запис у регістр UDR здійснюється під час передачі, то в цьому випадку дані пересилаються у регістр зсуву після передачі останнього стоп-біта поточного кадру.

Після пересилання слова даних у регістр зсуву, прапорець UDRE (UDREn) регістра UCSRA (UCSRnA) встановлюється в одиницю, що означає готовність передавача до отримання нового байта даних. У цьому стані прапорець залишається до наступного запису в буфер. Одночасно з пересиланням у регістрі зсуву формується службова інформація – старт-біт, можливий біт парності (тільки в USART), а також один або два стоп-біти.

Після завантаження регістра зсуву його вміст починає зсуватися вправо і поступати на вивід TXD (TXDn) у порядку, який було розглянуто у п. 6.3.4. Швидкість зсуву визначається налаштуванням контролера тактових сигналів.

Під час роботи в синхронному режимі зміна стану виводу TXD (TXDn) відбувається за одним з фронтів сигналу XCK (XCKn). Якщо розряд UCPOL (UCPOLn) РГ UCSRC (UCSRnC) скинуто в нуль, зміна стану виводу відбувається за наростаючим фронтом сигналу XCK (XCKn), якщо ж встановлений в одиницю – за спадаючим фронтом сигналу, як показано на рис. 6.13.

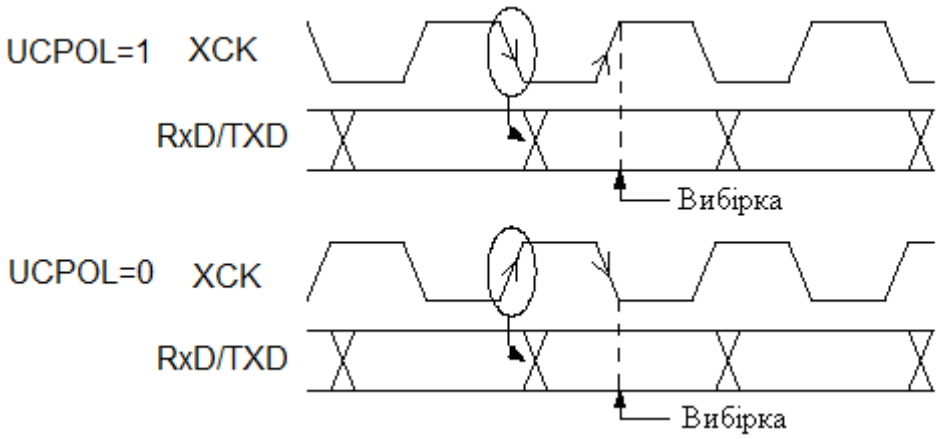


Рис. 6.13. Часові діаграми для синхронного режиму роботи модуля УСАПП

Якщо під час передачі попереднього слова у регістр UDR було записано нове слово даних, то після передачі останнього стоп-біта попереднього слова в регістр зсуву пересилається нове слово. Якщо ж до моменту закінчення передачі кадру нового запису виконано не було, то встановлюється прапорець переривання TXC (TXCn) регістра UCSRA (UCSRnA) – «Передачу завершено». Скидання прапорця здійснюється апаратно при вході у підпрограму обробки переривання або програмно, записом в цей розряд одиниці.

Відключення передавача здійснюється скиданням розряду TXEN (TXENn) регістра UCSRB (UCSRnB). Якщо у момент виконання цієї команди здійснювалася передача, скидання розряду відбудеться тільки після завершення поточної та відкладеної передач, тобто після очищення буферного регістра та регістра зсуву передавача. У разі вимкненого передавача вивід TXD (TXDn) може використовуватися як лінія введення/виведення загального призначення.

6.3.6. Прийом даних модулем

Для дозволу робота приймача треба встановити розряд RXEN (RXENn) регістра UCSRB (UCSRnB) [3]. Після цього вивід RXD (RXDn) підключається до приймача УАПП/УСАПП і починає функціонувати як вхід, незалежно від значень регістра керування портом. Якщо використовується синхронний режим роботи, треба програмно перевизначити функціонування виводу ХСК (ХСКn).

Прийом даних починається після виявлення приймачем коректного старт-біта (рис. 6.12*б*). Кожен розряд кадру зчитується з частотою, яка визначається контролером швидкості передачі або тактовим сигналом ХСК (ХСКn). Отримані розряди даних послідовно розміщуються у регістр зсуву приймача до виявлення першого стоп-біта кадру. Після цього вміст регістра зсуву пересилається у буфер приймача, з якого прийняте значення може бути зчитано. При використанні 9-розрядних слів даних значення старшого восьмого розряду, оскільки нумерація розрядів ведеться від нуля, може бути визначене за станом прапорця RXB8 (RXB8n) регістра UCSRB (UCSRnB). У модулях USART вміст цього розряду має бути зчитаний до звернення до регістра даних. Це пов’язано з тим, що прапорець RXB8 (RXB8n) відображає значення старшого розряду слова даних кадру, який знаходиться на верхньому рівні буфера приймача, стан якого під час читання регістра даних зміниться.

Якщо було запрограмовано контроль парності (тільки УСАПП), то схема контролю парності обчислює біт парності для всіх розрядів прийнятого слова даних і порівнює його з прийнятим бітом парності. Результат перевірки запам’ятовується у буфері приймача разом з прийнятим словом даних і стоп-бітами. Наявність або відсутність помилки контролю парності може бути потім визначено за станом прапорця UPE (UPEn) регістра UCSRA (UCSRnA).

Цей прапорець встановлюється в одиницю якщо наступне слово, яке може бути прочитане з буфера, має помилку контролю парності. Під час вимкненого контролю парності прапорець UPE (UPEn) завжди читається як нуль.

У регістрі UCSRA(UCSRnA) блока приймача модулів УАПП/УСАПП є ще два прапорці, які показують стан обміну: прапорець помилки кадрування FE (FEn) і прапорець переповнення DOR (DORn)/OR (Orn) [3]. Прапорець FE (Fen) встановлюється в одиницю, якщо значення першого стоп-біта прийнятого кадру не відповідає потрібному, тобто дорівнює нулю.

Прапорці DOR (DORn) в УСАПП та OR (ORn) в УАПП відображають втрату даних через переповнення буфера приймача. В УАПП прапорець встановлюється в одиницю, якщо до моменту закінчення прийому кадру (заповнення регістра зсуву приймача) дані попереднього кадру не були зчитані з регістра даних. В УСАПП прапорець встановлюється в одиницю у разі прийому старт-біта нового кадру при заповнених буфері та регістрі зсуву приймача. Встановлений прапорець DOR (DORn)/OR (ORn) означає, що між минулим байтом, який зчитано з регістра UDR, та байтом, зчитаним у цей момент, відбулася втрата одного або декількох кадрів.

Обробка прапорців у модулях УАПП/УСАПП дещо відрізняється.

У модулях УАПП прапорець помилки кадрування FE (FEn) має бути прочитаний перед зверненням до регістра даних, а прапорець переповнення OR (ORn) – після звернення до цього регістра.

У модулях УСАПП усі прапорці помилок буферизуються разом із словом даних, тобто відповідні розряди регістра UCSRA (UCSRnA) належать до кадру, слово даних якого буде прочитано під час наступного звернення до регістра даних UDR (UDRn). Тому стан цих прапорців треба зчитати перед зверненням до регістра даних.

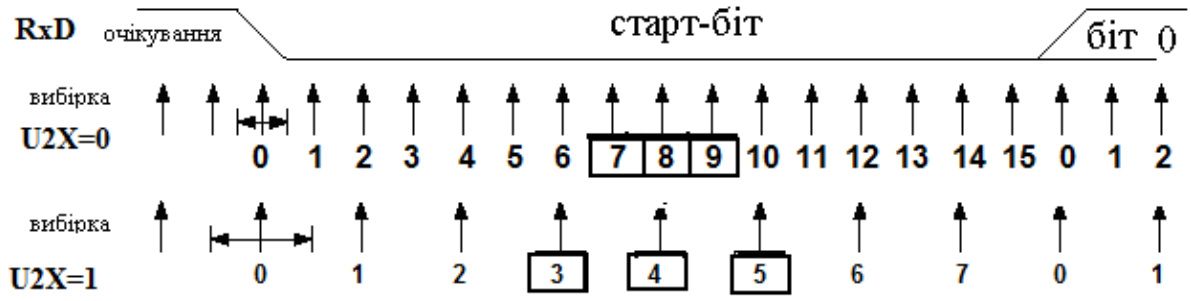
Для індикації стану приймача у модулях УАПП/УСАПП використовується прапорець переривання «Прийом завершено» RXC (RXCn) регістра UCSRA (UCSRnA). Цей прапорець встановлюється в одиницю за наявності в буфері приймача непрочитаних даних. У модулях УАПП цей прапорець скидається після читання регістра даних, а у модулях УСАПП – під час звільнення буфера (після зчитування всіх даних, які знаходяться у ньому).

Вимкнення приймача здійснюється скиданням розряду RXEN (RXENn) регістра UCSRB (UCSRnB). На відміну від передавача, приймач вимикається відразу ж після скидання розряду, тобто кадр, який приймається у цей момент, втрачається. У разі вимкнення приймача очищується його буфер, тобто втрачаються також усі непрочитані дані. При вимкненому приймачі вивід RXD (RXDn) може використовуватися як лінія введення/виведення загального призначення.

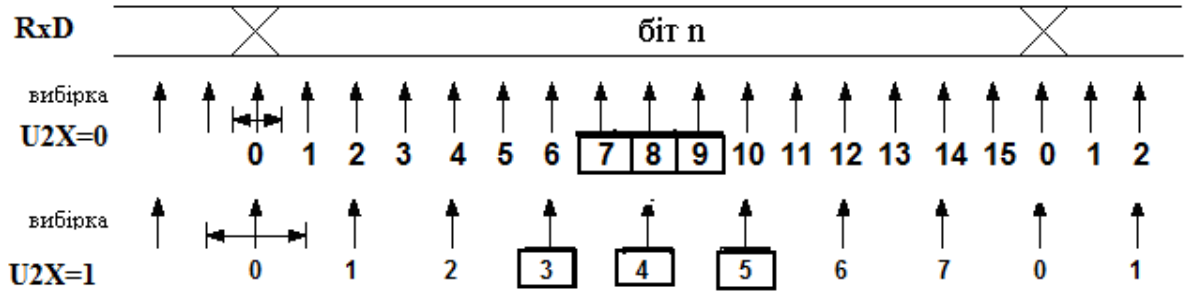
Прийом всіх розрядів кадру здійснюється по-різному, залежно від режиму роботи модуля.

Під час роботи модуля УСАПП у синхронному режимі стан виводу RXD (RXDn) читається за одним з фронтів сигналу XCK (XCKn). Якщо розряд UCPOL (UCPOLn) регістра UCSRC (UCSRnC) скинуто в нуль, зчитування стану виводу RXD відбувається за спадаючим фронтом сигналу XCK (XCKn), якщо ж встановлено в одиницю – за наростаючим фронтом сигналу. Інакше кажучи, зчитування даних з виводу RXD (RXDn) і їх видача на вивід TXD (TXDn) відбуваються за протилежними фронтами (рис. 6.13).

Під час прийому в асинхронному режимі роботи використовуються схеми відновлення тактового сигналу і даних. Схема відновлення тактового сигналу призначена для синхронізації внутрішнього тактового сигналу, який формується контролером швидкості передачі, і кадрів, які надходять на вивід RXD (RXDn) МК. Схема відновлення даних здійснює читання та фільтрацію кожного розряду кадру, що приймається. Вона опитує вхід приймача з метою визначення старт-біта кадру (рис. 6.14*а*).



а



б

Рис. 6.14. Розпізнавання розрядів кадру: *а* – старт-біт; *б* – інші розряди

Частота опитування залежить від стану розряду U2X (U2Xn) регістра UCSRA (UCSRnA). У звичайному режимі (за U2Xn = 0) частота опитування у 16

разів перевищує швидкість передачі даних, а у прискореному режимі (за U2X = 1) – у 8 разів. Горизонтальні стрілки на рис. 6.14 ілюструють можливий відхід синхронізації у процесі опитування. Більш високу розсинхронізацію у часі має прискорений обмін (біт U2X = 1).

Виявлення у процесі очікування зміни сигналу на виводі RXD (RXDn) з одиниці на нуль інтерпретується як можлива поява переднього фронту старт-біта. Коли виявлено цю зміну, скидається лічильник-дільник на 16 у ланцюзі формування сигналу «Синхр. RXD» (рис. 6.12*б*; рис. 6.14*а*). У результаті цього відбувається суміщення моментів переповнення цього лічильника-дільника з межами зміни на вході RXD бітів кадру, що приймається.

Шістнадцять станів лічильника-дільника ділять час, протягом якого кожен біт посилки, що приймається, присутній на вході RXD, на 16 фаз (вибірок), з 0-ї по 15-у для кожного біта. У нормальному режимі перевіряється значення 7-ї, 8-ї і 9-ї вибірок вхідного сигналу, а в прискореному режимі – 3-ї, 4-ї і 5-ї вибірок (рис. 6.11*а*). Якщо значення хоча б двох вибірок із вказаних дорівнює одиниці, старт-біт вважається помилковим (завада), а приймач переходить до очікування наступної зміни вхідного сигналу з одиниці на нуль.

Коли це відбувається, вважається, що виявлено старт-біт нової послідовності даних, з яким синхронізується внутрішній тактовий сигнал приймача. Після цього починає знову працювати схема відновлення даних.

Рішення про значення кожного прийнятого розряду також ухвалюється за результатами 7-ї, 8-ї і 9-ї (3-ї, 4-ї і 5-ї) вибірок вхідного сигналу (рис. 6.14*б*). Станом розряду вважається логічне значення, яке було отримане щонайменше у двох з трьох вибірок. Процес розпізнавання повторюється для всіх розрядів кадру, що приймається, включаючи перший стоп-біт.

Таким чином, старт-біт нового кадру може передаватись відразу ж після останньої вибірки, яка використовується для визначення значення першого стоп-біта.

6.3.7. Обмін даними через інтерфейс у мікроконтролерній мережі

У мікроконтролерних мережах відбувається обмін інформацією між одним ведучим та одним з декількох ведених МК. Кожен ведений МК-мережі має свою унікальну адресу. Якщо ведучий МК хоче що-небудь передати, то він посилає адресний байт, який визначає, до якого з ведених МК він збирається звернутися. Коли один з ведених МК розпізнав свою адресу, він переходить у режим прийому даних і відповідно приймає подальші байти як дані. Решта ведених МК ігнорують байти даних, що приймаються, до посилки ведучим нового адресного байта. Програмування режиму фільтрації кадрів даних, що приймаються і не містять

адреси, здійснюється встановленням в одиницю розряду MPCM (MPCMn) регістра UCSRA (UCSRnA) [3].

У МК, який виконує роль ведучого, треба встановити режим передачі 9-розрядних даних. Під час передачі адресного байта старший – 8-й розряд, за умови, що перший (молодший) розряд у разі двійкового кодування має номер – нуль, повинен встановлюватися в одиницю, а у разі передачі байтів даних він повинен скидатися в нуль.

У ведених МК механізм прийому залежить від режиму роботи приймача. Під час прийому 9-розрядних даних ідентифікація вмісту кадру здійснюється за старшим дев'ятим розрядом слова даних. Якщо вказаний розряд встановлено в одиницю, то кадр містить адресу, якщо скинуто в нуль – кадр містить дані.

Для програмування обміну даними у мікроконтролерній мережі виконуються такі дії:

1. У всіх ведених МК встановлюється в одиницю розряд MPCM (MPCMn) регістра UCSRA (UCSRnA).
2. Усі МК програмуються в режим передачі 9-розрядних кадрів.
3. Ведучий МК посилає адресний кадр з TB8 = 1, а всі ведені МК його приймають. У кожному з ведених МК встановлюється прапорець RXC (RXCn) регістра UCSRA (UCSRnA).
4. Кожен ведений МК читає вміст регістра даних (адресний кадр) та порівнює його із своєю мережевою адресою. МК, адреса якого збіглася з адресою, посланою ведучим, скидає в нуль розряд MPCM (MPCMn).
5. Адресований МК починає приймати кадри, що містять дані, оскільки їх дев'ятий розряд дорівнює нулю. У решті не адресованих ведених МК розряд MPCM (MPCMn) залишається встановленим в одиницю, через це кадри даних з бітом TB8 = 0 будуть ними ігноруватися.
6. Після прийому останнього байта даних адресований МК встановлює в одиницю розряд MPCM (MPCMn) і знову чекає прихід кадру з адресою. Процес повторюється з пункту 3.

6.3.8. Розрахунок швидкості передачі інформації, тривалості одного біта та часу передачі одного байта

Нижче наведено приклад розрахунку швидкості передачі інформації, тривалості одного біта та часу передачі одного байта для асинхронного режиму роботи (біт U2Xn = 0).

У цьому випадку швидкість передачі даних (обміну)

$$V_{ПД} = \frac{f_{CLK}}{16 \cdot (UBRR + 1)}, \qquad (6.5)$$

де f_{CLK} – частота системного тактового генератора; UBRR – вміст регістра контролера швидкості передачі, що змінюється програмно (UBRR = 0...4095).

Наприклад, якщо f_{CLK} = 16 МГц, а UBRR = 24, тоді

$$V_{ПД} = \frac{16000000}{16 \cdot 25} = 40000 \ \ посилок/секунду = 40000 \ \ біт/секунду = 40 Кбіт/с.$$

Відповідно тривалість одного біта (однієї посилки)

$$t_{ПОС} = \frac{1}{V_{ПД}} = \frac{1000000}{40000} = 25 \ \ мксек.$$

Наприклад, у лінію зв'язку передаються наступні 12 бітів: старт-біт, 8 інформаційних біт, 9-й службовий біт ТВ8, додатковий перевірочний біт на парність і один стоп-біт.

Тривалість цих 12 посилок однакова і за $V_{ПД}$ = 40 Кбіт/с дорівнює $t_{ПОС}$ = 25 мксек.

Оскільки передавач містить буферний регістр даних UDR, то в нього програмно завантажується черговий байт для передачі поки з регістра зсуву видаються посилки попереднього байта. Після передачі стоп-біта новий байт з регістра UDR пересилається в регістр зсуву та ініціюється новий цикл передачі.

Таким чином, час передачі одного байта дорівнює 12 · $t_{посилок}$, що у разі тривалості $t_{ПОС}$ = 25 мксек становить $t_Б$ = 25 · 12 = 300 мксек.

6.3.9. Моделювання модуля у пакеті PROTEUS

Моделювання модуля УСАПП у пакеті PROTEUS 8.6 описано у [3]. Наведено схему алгоритму роботи моделі та робочу програму мовою С. Отримані результати моделювання підтвердили працездатність цього алгоритму та робочої програми.

Контрольні запитання та завдання

1. Поясніть призначення та місце пристрою введення/виведення у МПС.
1. Дайте визначення поняттю «інтерфейс».
2. Що являє собою послідовний обмін даними між зовнішніми пристроями і МПС?
3. Чим відрізняються паралельний обмін від послідовного?
4. Які перетворення даних відбуваються у послідовному інтерфейсі?
5. Поясніть особливості обміну під керуванням програми та за перериванням.
6. Як адресуються зовнішні пристрої в МПС?
7. Назвіть та поясніть команди введення/виведення у МП і8086.

8. Поясніть структуру ППІ і8255.
9. Як інтерфейс і8255 підключається до системної шини?
10. Поясніть особливості програмування обміну даними у МПС.
11. Поясніть формати керувальних слів інтерфейсу і8255.
12. Назвіть та поясніть режими роботи ППІ і8255.
13. Які керувальні сигнали використовуються для організації програмно керованого стробованого введення та виведення даних ППІ і8255 у режимі 1?
14. Які способи адресації пристроїв введення/виведення є у МПС на основі і8086?
15. Які види команд введення/виведення є у 16-розрядному МП?
16. Як відбувається звернення до паралельних портів AVR-мікроконтролерів?
17. Наведіть та опишіть спрощену структурну схему одного з каналів паралельного порту введення/виведення AVR-мікроконтролерів?
18. Опишіть особливість використання буферів В у структурі порту.
19. Скільки регістрів є для програмування паралельних портів AVR-мікроконтролерів та як вони використовуються?
20. Як здійснюється керування підтягуючими резисторами портів.
21. Наведіть приклад конфігурування виводів портів введення/виведення.
22. Опишіть особливості читання стану виводів AVR-мікроконтролера.
23. Що таке УСАПП та УАПП?
24. З яких трьох частин складається структура модуля УСАПП/УАПП? Які елементи містить кожна з них?
25. Які регістри використовуються для керування модулями УАПП та УСАПП?
26. Поясніть структурну схему блока синхронізації модуля УСАПП для МК Mega 128.
27. Які режими синхронізації підтримує УСАПП?
28. Як визначається швидкість обміну в модулі УСАПП?
29. Як визначається розмір слова даних у модулях УСАПП?
30. Як працює схема контролю парності?
31. Як проходить передача даних у модулях УСАПП?
32. Як проходить прийом даних у модулях УСАПП?
33. Опишіть послідовність дій для здійснення обміну даними у мікроконтролерній мережі.
34. Як визначити швидкість передачі даних для асинхронного звичайного режиму?

35. Як у моделі підключено кнопки, які задають байт для передачі?
36. Як у моделі підключено світлодіоди, які відображають передану в моделі інформацію?
37. Як у моделі визначається тривалість переданого біта?
38. Опишіть формат та призначення розрядів регістра UCSRC.
39. Як у програмі відбувається налаштування порту А на передачу?
40. Як у програмі визначається час завершення передачі та прийому чергового байта?
41. Як працюють схеми відновлення тактового сигналу і даних під час асинхронного прийому?

7. ЗВ'ЯЗОК МІКРОПРОЦЕСОРА ТА МІКРОКОНТРОЛЕРА З АНАЛОГОВИМ ОБ'ЄКТОМ КЕРУВАННЯ ТА МОДЕМОМ

7.1. Особливості введення/виведення аналогової інформації в мікропроцесорній системі

Для зв'язку МП/МК з аналоговим об'єктом керування використовується модуль АЦП/ЦАП [1–3].

Під час проектування модуля АЦП/ЦАП потрібно вирішувати такі задачі.

На апаратному рівні:

– вибір розрядності за заданою похибкою дискретизації;
– вибір величини дискретизації за часом за теоремою Котельникова;
– визначення необхідності застосування і, якщо це необхідно, то вибір мікросхеми пристрою вибірки-зберігання;
– визначення необхідного часу перетворення;
– якщо МПС проектується на основі МП, то вибір мікросхем АЦП і ЦАП, що забезпечують потрібну похибку, швидкодію і споживану потужність;
– вибір схем включення, що забезпечують необхідний діапазон зміни вхідних і вихідних напруг;
– якщо МПС проектується на основі МК, то вибір МК, який має відповідний модуль АЦП та ЦАП;
– розробка принципової схеми.

На програмному рівні. Якщо МПС проектується на основі МП, то:

– формування імпульсу вибірки для пристрою вибірки-зберігання;
– формування сигналу запуску АЦП («СТАРТ»);
– перевірка готовності даних на виході АЦП (аналіз виходу «READY» або обмін за перериванням);
– після визначення готовності АЦП введення даних у МП;
– формування сигналу «СКИДАННЯ» для АЦП;
– після завершення обробки даних, отриманих від АЦП, виведення керувальних впливів у порт, до якого під'єднано ЦАП. Зазвичай порт виведення має буферний регістр, який зберігає байт до наступного виводу. ЦАП перетворює цей байт в аналогову напругу, яка подається на аналоговий виконавчий пристрій.

Якщо МПС проектується на основі МК, то:

– виконання ініціалізації відповідного модуля;
– запуск процес перетворення;

– за допомогою аналізу прапорця або за перериванням дізнатись про закінчення перетворення;
– відповідним чином обробити результат перетворення.

7.2. Застосування аналого-цифрового перетворювача і пристрою вибірки-зберігання під час введення аналогової інформації у мікропроцесор/мікроконтролер

7.2.1. Аналого-цифровий перетворювач

Аналого-цифровий перетворювач – це пристрої, що перетворюють вхідні аналогові сигнали у відповідні їм цифрові сигнали, які придатні для роботи з МК/МП й іншими цифровими пристроями. АЦП широко застосовуються у пристроях дискретної автоматики, цифрових системах керування для перетворення аналогових сигналів від датчиків у цифрову форму, в системах відображення інформації для цифрової індикації, в системах передачі даних і багатьох інших галузях техніки.

Різні за фізичною природою сигнали, що знімаються з датчиків і характеризують контрольований процес, спочатку перетворюються в електричний сигнал, а потім вже за допомогою перетворювачів «напруга-код» – у цифровий. На вході АЦП зазвичай є напруга, яка постійна чи повільно змінюється, а з виходу знімаються дані зазвичай у паралельному двійковому коді.

Методи побудови АЦП, які орієнтовані на використання у МПС, поділяються на послідовні, паралельні і послідовно-паралельні [1].

Різним методам побудови АЦП відповідають пристрої, що розрізняються за точністю, швидкодією, завадостійкістю, складністю реалізації і т. ін.

Одним з найбільш поширених є метод послідовного наближення, який зазвичай застосовується в АЦП, які орієнтовано на використання у МПС, наприклад, AD571 [1]. На рис. 7.1 наведено спрощену структурну схему АЦП послідовного наближення.

АЦП містить регістр послідовного наближення, ЦАП, аналоговий компаратор і генератор тактових імпульсів. Аналоговий компаратор включає власне аналоговий компаратор на мікросхемі операційного підсилювача, схему формування рівнів, що перетворює різнополярні імпульси, які знімаються з виходів власне аналогового компаратора, у цифровий сигнал, та інвертор [1].

Після надходження імпульсу «ПУСК» на виході старшого (N-1)-го розряду регістра послідовного наближення з’являється напруга логічної одиниці, а на інших його виходах – логічні нулі. На виході ЦАП формується

напруга $U_{цап} \approx 0{,}5 \cdot U_{вх.max}$, що на входах аналогового компаратора порівнюється з вхідною аналоговою напругою $U_{вх}$.

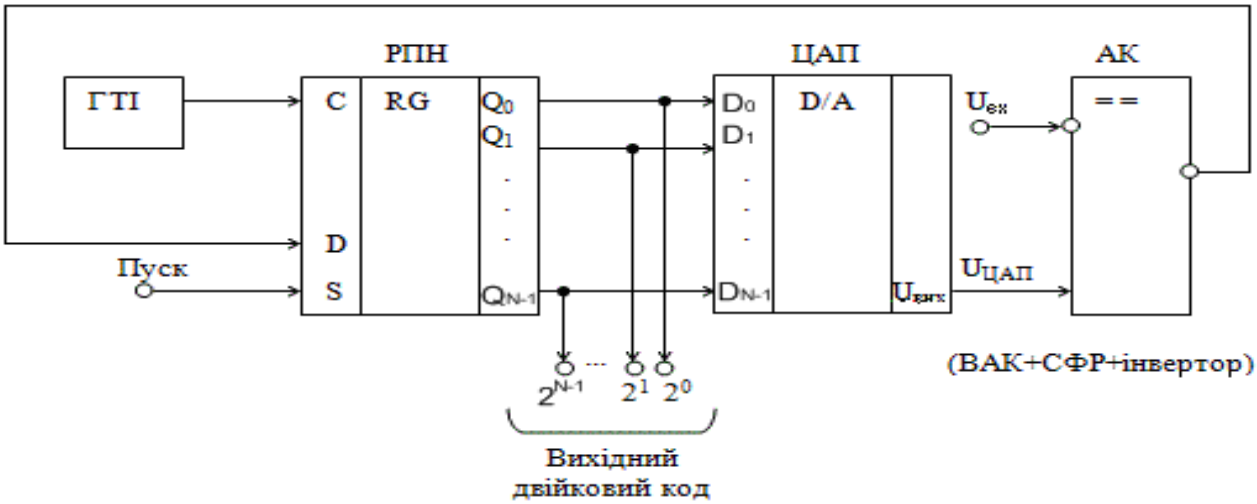


Рис. 7.1. Спрощена структурна схема АЦП послідовного наближення

Якщо вхідна напруга $U_{вх}$ більше напруги, що знімається з виходу ЦАП, то на виході власне аналогового компаратора з'являється від'ємний імпульс. Схема формування рівнів перетворює його в нульовий цифровий сигнал. З виходу інвертора аналогового компаратора знімається логічна одиниця, що подається на вхід D регістра послідовного наближення. Під час надходження на вхід C регістра послідовного наближення імпульсу від генератора тактових імпульсів зберігається логічна одиниця у старшому (N-1)-ому розряді і з'являється одиниця у (N-2) розряді. Якщо під час першого порівняння $U_{вх} < U_{цап}$, то з виходу аналогового компаратора знімається логічний нуль. Імпульсом на синхровході вміст старшого (N-1)-ого розряду регістра послідовного наближення обнуляється, а в (N-2)-й записується логічна одиниця. Якщо після наступного порівняння на виходах двох старших розрядів регістра послідовного наближення містяться дві одиниці (під час першого порівняння $U_{вх} > U_{цап}$), то вихідний сигнал ЦАП: $U_{цап} \approx (0{,}5 + 0{,}25) \cdot U_{вх.max}$. На компараторі $U_{вх}$ знову порівнюється з цією напругою і т. д. Так формуються всі розряди на виході регістра послідовного наближення до наймолодшого. Після виконання останнього $N_р$-го порівняння, де $N_р$ – число розрядів вихідного коду АЦП, цикл формування вихідного коду закінчується. Стан виходів регістра послідовного наближення відповідає цифровому двійковому еквіваленту вхідної аналогової напруги. Якщо, наприклад, $U_{вх} = U_{вх.max}$, то комбінація вихідного двійкового коду дорівнює 111...11 (всі одиниці). Час перетворення в АЦП, який розглянуто вище, визначається кількістю розрядів вихідного двійкового коду $N_р$ і тактовою частотою $f_{ГТІ} = 1/T_{ГТІ}$: $t_{прт} \approx N_р \cdot T_{ГТІ}$. АЦП послідовного наближення мають досить

високу швидкодію та відносно просту структуру, тому широко застосовуються у МПС.

АЦП у МК, наприклад, AVR, побудовано за принципом послідовного наближення.

Роботу паралельно-послідовного та паралельного АЦП розглянуто в [1].

7.2.2. Пристрій вибірки-зберігання

Під час аналого-цифрового перетворення сигналів, які швидко змінюються, можуть виникати динамічні похибки, що визначаються, по-перше, частотою і часом перетворення, а по-друге, похибкою, що виникає через невідповідність рівня вхідного сигналу, що перетворюється, його цифровому значенню. Ця похибка називається апертурною похибкою АЦП та виникає тоді, якщо зміна вхідного сигналу під час перетворення еквівалентна більш ніж одиниці молодшого значущого розряду [2; 10]. У цьому випадку під час вхідного сигналу, що швидко змінюється у часі, створюється невизначеність у тому, яким насправді було миттєве значення вхідного сигналу в момент часової вибірки. Час між моментом фіксації миттєвого значення вхідного сигналу t_i (моментом відліку) і моментом одержання його цифрового еквівалента називається апертурним часом – t_a.

Якщо прийняти, що для Np-розрядного АЦП апертурна похибка не повинна перевищувати кроку квантування за рівнем $\Delta U_{вх}$, то між частотою сигналу f, апертурним часом ta та апертурною похибкою є співвідношення [1]:

$$\Delta U_{вх} = \frac{U_m}{2^{Np} - 1} \geq Um * 2\pi f * ta. \tag{7.1}$$

Розділивши ліву і праву частини нерівності (7.1) на U_m отримаємо:

$$\frac{1}{2^{Np} - 1} \geq 2\pi f * ta. \tag{7.2}$$

Наприклад, якщо Np = 8, а час перетворення АЦП $t_{пер}$ = 7,5 мкс, то частота вхідного сигналу не повинна перевищувати 83 Гц. У цьому випадку апертурна похибка не перевищує одиниці молодшого значущого розряду двійкового коду на виході АЦП.

Для зменшення апертурної похибки АЦП звичайно використовуються пристрій вибірки-збкрігання, що включаються між входом АЦП і виходом джерела аналогового сигналу. Приклад схеми пристрою вибірки-зберігання наведено у пп. 1.3.2.3. У цій схемі в момент часової вибірки запам’ятовується миттєве значення вхідного сигналу. Таким чином, на вхід АЦП подається постійний рівень, який може змінитися тільки після наступної часової вибірки.

Усі модулі АЦП у МК мають зазвичай пристрій вибірки-збкрігання (підрозд. 7.3).

7.2.3. Вибір та розрахунок аналого-цифрового перетворювача

В АЦП здійснюється квантування (дискретизація) за рівнем і часом (рис. 7.2) [1].

Під час дискретизації (квантування) за рівнем діапазон зміни аналогової напруги на вході АЦП розділяється на низку рівнів (дискретних значень) $N_Д$ з кроком квантування за рівнем ΔU, що становить

$$\Delta U = \frac{U_{BX\,MAX} - U_{BX\,MIN}}{N_Д - 1}. \tag{7.3}$$

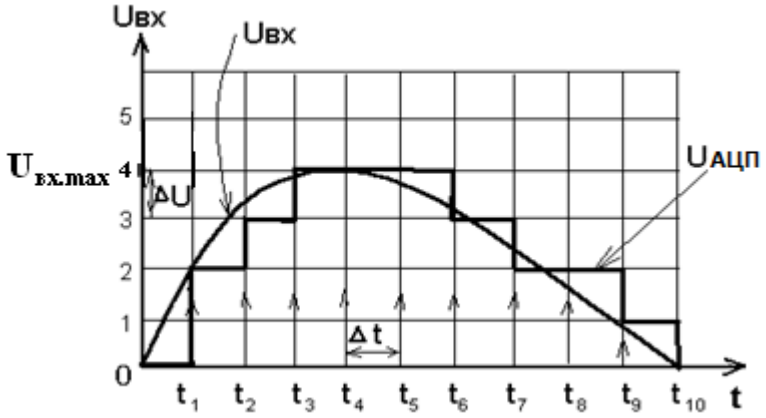


Рис. 7.2. Квантування (дискретизація) аналогової величини за рівнем і часом

Вхідна аналогова напруга $U_{вх}$ перетворюється у дискретну величину, яка визначається у фіксовані моменти часу найближчим до безперервної (аналогової) величини рівнем квантування.

Кожному дискретному значенню (рівню) відповідає комбінація двійкового коду з кількістю розрядів – $N_Р$.

Величина $N_Р$ залежить від числа дискретних значень $N_Д$ на виході АЦП, включаючи нульове.

Вибір $N_Р$ робиться відповідно до співвідношення:

$$2^{Np} \geq N_д. \tag{7.4}$$

Число дискретних значень (рівнів квантування) залежить від похибки квантування за рівнем.

Абсолютна похибка квантування за рівнем

$$\delta_{АБС} \leq \frac{\Delta U}{2}, \tag{7.5}$$

де ΔU – величина кроку квантування за рівнем, що визначається з виразу (7.3).

Із співвідношення (7.5) випливає, що максимальна абсолютна похибка дорівнює половині кроку квантування за рівнем.

Відповідно відносна похибка квантування за рівнем

$$\delta_{\text{ВІД}} \le \frac{\delta_{\text{АБС}} \cdot 100\%}{U_{\text{ВХ.MAX}} - U_{\text{ВХ.MIN}}} = \frac{\Delta U \cdot 100\%}{\{2 \cdot (U_{\text{ВХ.MAX}} - U_{\text{ВХ.MIN}})\}} =$$

$$= \frac{(U_{\text{ВХ.MAX}} - U_{\text{ВХ.MIN}}) \cdot 100\%}{\{(N_{\text{Д}} - 1) \cdot 2 \cdot (U_{\text{ВХ.MAX}} - U_{\text{ВХ.MIN}})\}} = \frac{50}{N_{Д} - 1}[\%]. \quad (7.6)$$

У наведеній формулі з $N_{\text{Д}}$ віднімається одиниця, тому що одним з дискретних значень є нульове. Звідси необхідне число дискретних значень, що відображає нашу безперервну функцію із заданою точністю, визначається як

$$N_{\text{Д}} \ge \frac{50}{\delta_{ВІД}} + 1. \quad (7.7)$$

Наприклад, якщо $\delta_{\text{від}} \le 0{,}2$ %, тоді $N_{\text{Д}}$ має бути не менше 251. Приймаючи $N_{\text{Д}} = 256$, визначаємо, що кількість розрядів $N_{\text{р}}$ у цьому випадку має бути 8 ($2^8 = 256$). Якщо вхідна величина змінюється, наприклад, у діапазоні від 0 до 2,55 В, то величина кроку квантування за рівнем у разі $N_{\text{д}} = 256$ становить $\Delta U = 10$ мВ. Відповідно $\delta_{\text{абс.}} \le 5$ мВ, а $\delta_{\text{від.}} \le 50/255 < 0{,}2$ %.

Під час проектування АЦП важливе значення має вибір величини кроку квантування за часом $\Delta t = T$. Значення T визначає необхідну швидкодію АЦП і каналу обробки інформації.

За теоремою Котельникова значення $\Delta t = T$ повинне задовольняти вираз

$$\Delta t = T \le \frac{1}{2 f_{\max}}, \quad (7.8)$$

де f_{max} – частота вищої гармоніки спектра вхідного аналогового сигналу АЦП [1].

Фізично цей вираз варто трактувати в такий спосіб: на один період максимальної гармоніки спектра вхідного аналогового сигналу під час переходу від аналогової до дискретної величини необхідно взяти не менше двох відліків.

Якщо на вхід АЦП, наприклад, надходить аналоговий сигнал з виходу фільтра нижніх частот, то в якості f_{max} можна використовувати частоту зрізу цього фільтра.

У МПС можуть використовуватися різні типи АЦП. В [1] розглянуто декілька з них.

7.3. Особливості архітектури модуля аналого-цифрового перетворювача у AVR-мікроконтролерах

7.3.1. Функціональна схема модуля

До більшості моделей AVR-мікроконтролерів належить модуль 10-розрядного АЦП послідовного наближення. Цей модуль може мати різну кількість входів аналогового мультиплексора. На рис. 7.3 наведено

функціональну схему модуля АЦП, аналоговий мультиплексор якого має 16 аналогових входів.

У деяких моделях МК елементи і пов’язані з ними сигнали, які виділено на рис. 7.3 сірим кольором, відсутні, а неінвертуючий вхід компаратора з пристроєм вибірки-зберігання підключено безпосередньо до виходу мультиплексора (показано пунктирною лінією).

У більшості моделей входи АЦП можуть об’єднуватися попарно для формування каналів з диференціальним входом.

Для деяких каналів є можливість 10- та 200-кратного попереднього підсилення вхідного сигналу.

Як джерело опорної напруги для АЦП може використовуватись напруга живлення МК та внутрішнє або зовнішнє джерело опорної напруги.

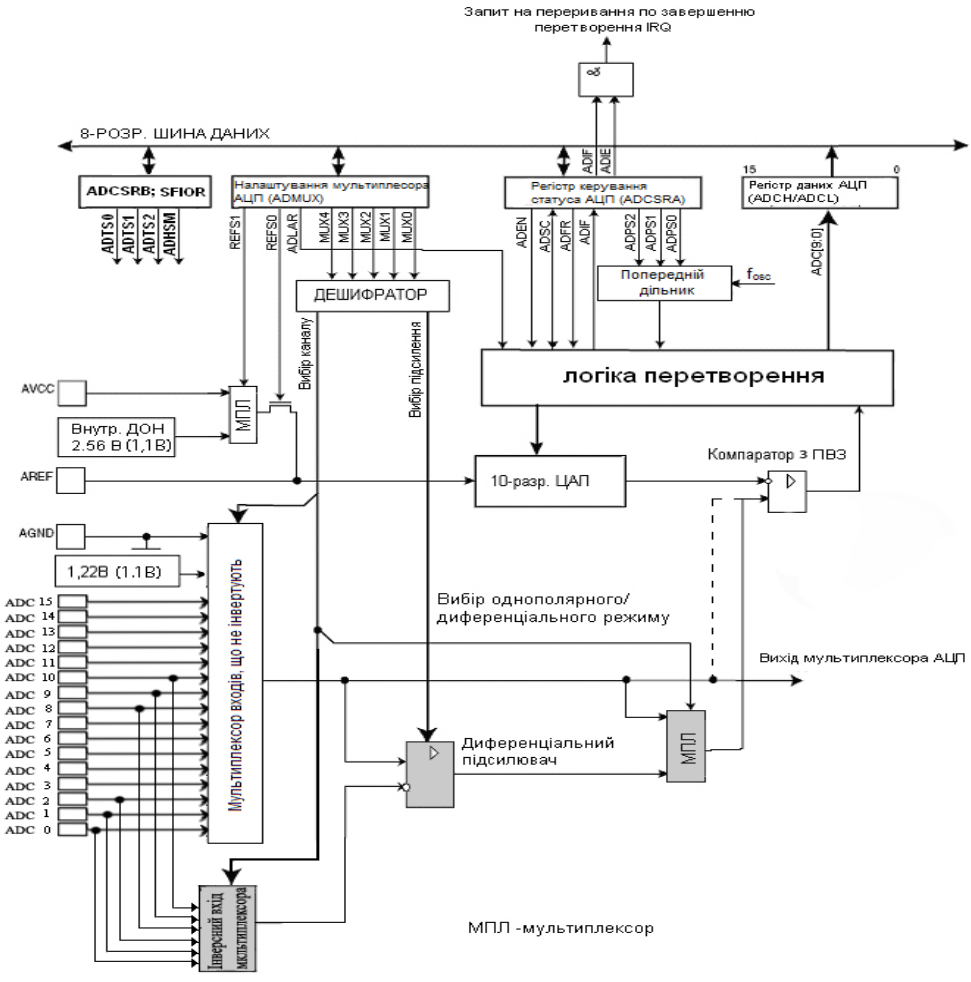


Рис. 7.3. Функціональна схема модуля АЦП

Модуль АЦП може функціонувати у двох режимах:

– режим одиночного перетворення, коли запуск кожного перетворення ініціюється користувачем;

– режим безперервного перетворення, коли запуск перетворень виконується безперервно через певні інтервали часу.

Вхідний аналоговий сигнал зберігається у пристрої вибірки-зберігання, що під час перетворення підтримує напругу на вході безпосередньо АЦП на постійному рівні.

АЦП працює за принципом послідовного наближення, який розглянуто у підрозд. 7.1.

У кінці перетворення в регістрі даних АЦП (ADCH та ADCL) буде 10-розрядний двійковий код, який еквівалентний вхідній напрузі.

АЦП може обробляти вхідні сигнали у двох режимах: однополярному та диференціальному. Під час однополярного режиму вхідний сигнал надходить на один із входів ADC0…ADC15. Під час диференціального режиму використовуються шість входів мультиплексора (ADC0, ADC1, ADC2, ADC8, ADC9, ADC10). Різниця сигналів між цими входами підсилюється за допомогою диференціального підсилювача і через мультиплексор надходить на вхід компаратора.

7.3.2. Програмування модуля

У табл. 7.1 наведено адреси регістрів, що використовуються для керування модулем АЦП деяких моделей сімейства Mega.

Таблиця 7.1. **Адреси регістрів керування модулем АЦП**

Регістр	**Адреса**	**ATmega8535x**	**ATmega8x**	**aTmega16x\32x**	**ATmega163x**	**ATmega323x**	**ATmega48x\88x\168x**	**ATmega64x**	**ATmega164x\324x\644x**	**ATmega165x**	**ATmega325x/3250x, ATmega645x/6450x**	**ATmega640x, ATmegal280x/1281x, ATmega2560x/2561x**	**ATmega128x**	**Опис**
ADCSR	$06($26)		♦		♦	♦								Регістр керування і стану
ADCSRA	$06 ($26)	♦		♦				♦					♦	Регістр керування і стану А
	($7A)						♦		♦	♦	♦	♦		
ADCSRB	($8E)							♦						Регістр керування і стану В
	($7B)						♦		♦	♦	♦	♦		
ADMUX	$07 ($27)	♦	♦	♦	♦	♦		♦					♦	Регістр керування мультиплек сором
	(7C)						♦		♦	♦	♦	♦		
SFIOR	$30 ($50)	♦	♦	♦										Регістр спеціальних функцій
	$20 ($40)												♦	

Формати регістрів ADCSRA (ADCSR) і ADMUX наведено на рис. 7.4 та 7.5, а опис функцій їх розрядів – у табл. 7.2 і 7.3 відповідно.

	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	ADIF	ADIÉ	ADPS2	ADPS1	ADPSO	ATmega8x aTmega128x
Зчитування (R) / Запис (W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIÉ	ADPS2	ADPS1	ADPSO	Інші моделі
Зчитування (R) / Запис (W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

ATmega 8x – ADCSR
Інші моделі – ADCSRA

Рис. 7.4. Формат регістра ADCSRA (ADCSR)

	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	ADIF	ADIÉ	ADPS2	ADPS1	ADPSO	ATmega8x aTmega128x
Зчитування (R) / Запис (W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ATmega8x aTmega48x/88x/168x
Зчитування (R) / Запис (W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	Інші моделі
Зчитування (R) / Запис (W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

Рис. 7.5. Формат регістра ADMUX

Таблиця 7.2. **Опис розрядів регістра ADCSRA (ADCSR*)**

Розряд	**Назва**	**Опис**
7	ADEN	Дозвіл АЦП (1 – увімкнено, 0 – вимкнено)
6	ADSC	Запуск перетворення (1 – почати перетворення)
5	ADATE (ADFR*)	Вибір режиму роботи АЦП (0 – одиночне, 1 – безперервне перетворення)
4	ADIF	Прапорець завершення АЦП
3	ADIE	Дозвіл переривання від завершення АЦП
2...0	ADPS2:ADPS0	Вибір частоти перетворення (табл. 7.4)
* У моделях ATmega8x, ATmega128x.		

Формат регістрів ADCSRB і SFIOR наведено рис. 7.6 і 7.7 відповідно (невикористовувані розряди регістра SFIOR позначено як «–»).

Таблиця 7.3. **Опис розрядів регістра ADMUX**

Розряд	Назва	Опис	Модель
7...6	REFS1:REFS0	Вибір джерела опорної напруги	Усі моделі
5	ADLAR	Ліве вирівнювання результату (табл. 7.7)	Усі моделі
4	–	Зарезервовано	ATmega8x
	MUX4	Вибір вхідних каналів (табл. 7.6)	Усі моделі крім ATmega8x
3...0	MUX3...MUX0	Вибір вхідних каналів і частоти перетворення (табл. 7.6)	Усі моделі

	7	6	5	4	3	2	1	0	
	–	–	–	–	–	ADTS2	ADTS1	ADTSO	ATmega64x
Зчитування (R) / Запис (W)	R	R	R	R	R	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	
	7	6	5	4	3	2	1	0	ATmega48x/88x/168x ATmega164x/324x/644x ATmega165x/325x/3250x ATmega645x/6450x ATmega1281x/2561x
	–	ACME	–	–	–	ADTS2	ADTS1	ADTSO	
Зчитування (R) / Запис (W)	R	R/W	R	R	R	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	
	7	6	5	4	3	2	1	0	
	–	ACME	–	–	MUX5	ADTS2	ADTS1	ADTSO	ATmega640x/1280x/2560x
Зчитування (R) / Запис (W)	R	R/W	R	R	R/W	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

Рис. 7.6. Формат регістра ADCSRB

	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTSO	–	X	X	X	X	ATmega8535x ATmega16x/32x
Зчитування (R) / Запис (W)	R	R	R	R	R	R/W	R/W	R/W	
Початкове значення	0	0	0	0	0	0	0	0	

Рис. 7.7. Формат регістра SFIOR

Для дозволу робити АЦП необхідно записати одиницю у розряд ADEN регістра ADCSRA (ADCSR), а для заборони – відповідно нуль. Якщо АЦП буде вимкнено під час циклу перетворення, то перетворення не буде завершено (в регістрі даних АЦП залишиться результат попереднього перетворення).

У більшості моделей модуль АЦП може використовуватись як мультиплексор аналогових сигналів для модуля аналогового компаратора. Для цього необхідно встановити розряд ACME (Analog Comparator Multiplexer Enable) в одиницю. Інверсний вхід аналогового компаратора може бути з'єднано, або з входом AIN1 модуля, або з будь-яким із входів АЦП залежно від значення розряду ADEN (одиниця – вхід AIN1, нуль – один із входів АЦП).

У деяких моделях, наприклад, ATmega8x та ATmega128x режим роботи АЦП визначається станом розряду ADFR (табл. 7.2). Якщо його встановлено в одиницю, АЦП працює в режимі безперервного перетворення. У цьому режимі запуск кожного наступного перетворення здійснюється автоматично після закінчення поточного перетворення.

Якщо розряд ADFR скинуто в нуль, АЦП працює в режимі одиночного перетворення та запуск кожного перетворення здійснюється командою користувача.

У більшості моделей, окрім ATmega8x та ATmega128x, запуск АЦП можливий перериванням від деяких периферійних пристроїв, наявних у МК. Для вибору режиму роботи в цих моделях використовується розряд ADATE регістра ADCSRA і розряди ADTS2...0 регістра SFIOR або ADCSRB.

Якщо розряд ADATE скинуто в нуль, АЦП працює в режимі одиночного перетворення. Якщо розряд ADATE встановлено в одиницю, функціонування АЦП визначається вмістом розрядів ADTS2...0, які визначають джерело сигналу запуску АЦП [3].

Запуск перетворення за перериванням здійснюється у разі встановлення в одиницю прапорця обраного переривання, розряд ADSC регістра ADCSRA при цьому апаратно встановлюється в одиницю. Запуск перетворення у цих режимах також може бути здійснено встановленням в одиницю розряду ADSC регістра ADCSRA.

Для модуля можуть використовуватись різні джерела опорної напруги. Вибір конкретного джерела опорної напруги здійснюється за допомогою розрядів REFS1:REFS0 регістра ADMUX [3]. Якщо REFS1 = REFS0 = 1, то використовується внутрішнє джерело опорної напруги напругою 2,56 В.

Канал однополярного або диференціального аналогового введення та каскад диференціального підсилення обираються за допомогою програмування розрядів MUXn (n = 0, 1…5) у РГ ADMUX [3]. Як однополярний аналоговий вхід АЦП залежно від моделі МК може бути обрано один із входів ADC0...ADC15,

внутрішнє джерело опорної напруги 1,22 (1,1) В, або 0 В (GND). У режимі диференціального введення є можливість вибору входів диференціального підсилювача, що інвертують або не інвертують.

Якщо обрано диференціальний режим введення, то підсилювач буде помножувати різницю напруг між обраною парою входів на заданий коефіцієнт підсилення. Якщо обрано однополярний режим введення, то каскад підсилення не використовується.

Робота АЦП дозволяється за допомогою встановлення в одиницю розряду ADEN у регістрі ADCSRA. Вибір опорного джерела та каналу перетворення неможливо виконати до встановлення ADEN. Якщо ADEN = 0, то АЦП не споживає струм, тому під час переходу в режими сну рекомендовано попередньо відключити АЦП.

АЦП генерує 10-розрядний результат, що міститься в парі регістрів даних АЦП: ADCH і ADCL. У початковому стані результат перетворення розміщується в молодших десяти розрядах 16-розрядного слова (вирівнювання вправо), але може бути розміщено у старших десяти розрядах (вирівнювання вліво) за допомогою встановлення в одиницю розряду ADLAR у регістрі ADMUX (табл. 7.3).

Коли досить точності 8-розрядного значення, то рекомендовано подання результату перетворення з вирівнюванням вліво. В цьому випадку необхідно читати тільки регістр ADCH. В іншому ж випадку необхідно першим читати вміст регістра ADCL, а потім ADCH, що гарантує, що обидва байти є результатом того самого перетворення. Як тільки виконано читання ADCL блокується доступ до регістрів даних з боку АЦП. Це означає, що якщо зчитано ADCL і перетворення завершується перед читанням регістра ADCH, то жоден з регістрів не може модифікуватися та результат перетворення губиться. Після читання ADCH доступ до регістрів ADCH і ADCL знову дозволяється.

АЦП генерує власний запит на переривання за завершенням перетворення. Якщо між читанням регістрів ADCH і ADCL доступ до даних для АЦП заборонено, то переривання виникне, навіть якщо результат перетворення буде загублено.

Одиночне перетворення запускається за допомогою запису одиниці у розряд запуску перетворення АЦП – ADSC. Цей розряд залишається в одиничному стані в процесі перетворення та скидається у нульовий стан за завершенням перетворення. Якщо в процесі перетворення перемикається канал аналогового введення, то перш ніж це зробити, АЦП завершить поточне перетворення.

У режимі автоматичного перезапуску АЦП безупинно оцифровує аналоговий сигнал і оновлює регістр даних АЦП. Цей режим задається за допомогою запису одиниці в розряд ADFR (ADATE) регістра ADCSR (ADCSRA).

Перше перетворення ініціюється за допомогою запису одиниці у розряд ADSC регістра ADCSR (ADCSRA). У цьому режимі АЦП виконує послідовні перетворення, незалежно від того скидається прапорець переривання АЦП – ADIF, чи ні.

7.3.3. Формування тактового сигналу

Для формування тактового сигналу модуль АЦП має попередній дільник, який формує похідні частоти відносно частоти синхронізації МК (рис. 7.8).

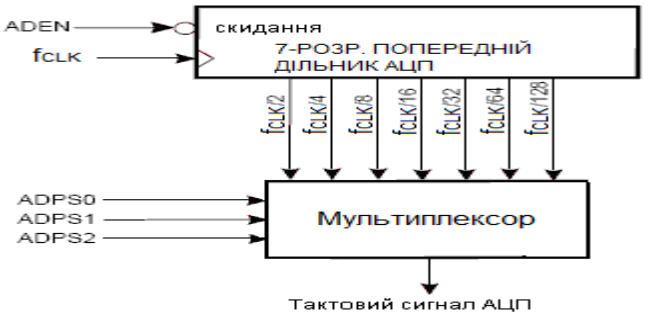


Рис. 7.8. Схема попереднього дільника АЦП

Коефіцієнт ділення встановлюється за допомогою розрядів ADPS2; ADPS1 та ADPS0 у регістрі ADCSRA (табл. 7.4).

Таблиця 7.4. **Встановлення коефіцієнта ділення попереднього дільника АЦП**

ADPS2	ADPS1	ADPS0	Коефіцієнт ділення
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

З моменту включення АЦП попередній дільник починає лічбу та працює доки розряд ADEN = 1.

7.3.4. Часові діаграми роботи

Часові діаграми роботи АЦП у різних режимах наведено на рис. 7.9…7.11.

Перше одиночне однополярне перетворення ініціюється встановленням в одиницю розряду ADSC у регістрі ADCSRA. Починається перетворення з наступного наростаючого фронту тактового (синхро) сигналу АЦП (рис. 7.9).

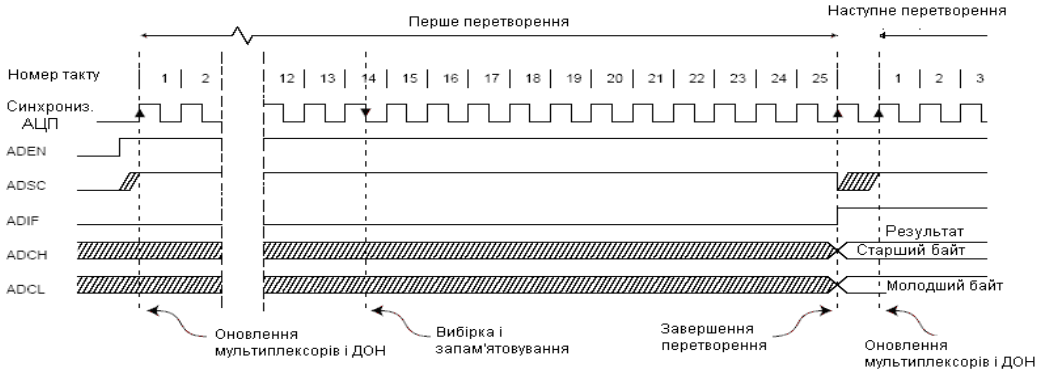


Рис. 7.9. Часові діаграми роботи АЦП при першому одиночному перетворенні в режимі одиночного перетворення

Нове одиночне перетворення може бути запущено відразу ж після скидання розряду ADSC (до збереження результату поточного перетворення). Однак реально цикл перетворення почнеться не раніше ніж через один такт після закінчення поточного перетворення.

За рахунок необхідності ініціалізації (встановлення біта ADEN у регістрі ADCSRA) перше одиночне перетворення після включення АЦП модуля виконується за 25 тактів синхронізації. Після початку першого одиночного перетворення на вибірку-зберігання витрачається 13,5 тактів.

Наступне одиночне перетворення вимагає 13 тактів синхронізації АЦП (рис. 7.10).

Рис. 7.10. Часові діаграми роботи АЦП у режимі наступного одиночного перетворення

Після його початку на вибірку-зберігання витрачається 1,5 такти синхронізації. За завершенням аналого-цифрового перетворення результат зберігається у регістрах даних АЦП і встановлюється прапорець ADIF. У режимі одиночного перетворення одночасно скидається розряд ADSC. Розряд ADSC може бути знову встановлено програмно і нове перетворення буде ініційовано першим наростаючим фронтом тактового сигналу АЦП.

Якщо використовується запуск за перериванням, то цикл перетворення починається за першим наростаючим фронтом тактового сигналу після встановлення прапорця обраного переривання (рис. 7.11).

Рис. 7.11. Часові діаграми роботи АЦП у режимі запуску за перериванням

У разі встановлення цього прапорця здійснюється скидання попереднього дільника модуля АЦП, що забезпечує фіксовану затримку між генерацією запиту на переривання та початком циклу перетворення.

Перетворення запускається під час встановлення відповідного прапорця, тобто навіть тоді, коли перетворення встановленням прапорця ADSCA (ADSC) не ініціалізувалося.

У режимі безперервного перетворення (автоматичного перезапуску) нове перетворення починається одразу після завершення попереднього перетворення (рис. 7.12), при цьому біт ADSC залишається у високому стані.

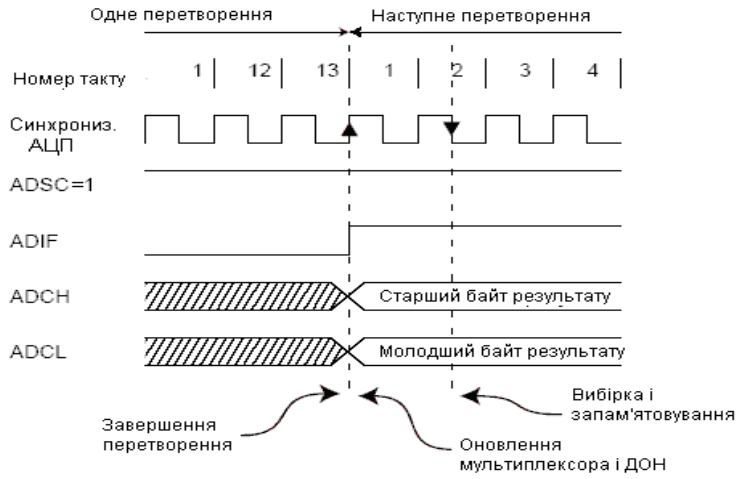


Рис. 7.12. Часові діаграми роботи АЦП у режимі автоматичного перезапуску для однополярного перетворення

Час перетворення для різних режимів подано в табл. 7.5.

Таблиця 7.5. **Час перетворення АЦП**

Тип перетворення	**Тривалість вибірки-зберігання (у тактах з моменту початку перетворення)**	**Час перетворення (у тактах)**
Перше перетворення	14,5	25
Нормальне однополярне перетворення	1,5	13
Нормальне диференціальне перетворення	1,5/2,5	13/14

Прапорець ADIF, як і прапорці інших переривань, скидається апаратно під час запуску підпрограми обробки переривання від АЦП або програмно, записом у нього одиниці. Дозвіл переривання здійснюється встановленням в одиницю розряду ADIE регістра ADCSR при встановленому прапорці I регістра SREG.

Використання диференціальних вхідних каналів має деякі особливості, які описано у [3].

7.3.5. Керування вхідним мультиплексором

Виводи МК, які підключено до входу АЦП, визначаються станом розрядів MUX3...MUX0 або MUX4...MUX0 регістрів ADMUX та ADCSRB відповідно до [3]. Частина МК Mega мають шістнадцять однополярних входів. У цьому випадку для керування вхідним мультиплексором використовуються шість керуючих розрядів: MUX5...MUX0.

Для каналів з диференціальним входом зазначені розряди визначають також коефіцієнт попереднього підсилення вхідного сигналу [3].

У табл. 7.6 наведено керування вхідним мультиплексором деяких моделей AVR-мікроконтролерів.

Під час перемикання вхідних каналів необхідно врахувати деякі рекомендації з перемикання та особливості підключення джерела опорної напруги, які наведено у [3].

Таблиця 7.6. **Керування вхідним мультиплексором у моделях ATmega16x/164x/32x/8535x/64x/128x/164x/165x/325x/3250x/645x/6450x/ 1281x/2561x**

MUX4…MUX0	Однополярний вхід	Диференціальний вхід		Попереднє підсилення
		(додатний)	(від'ємний)	
00000	ADC0	Не застосовується		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000[1]	Не застосовується	ADC0	ADC0	10x
01001[1]		ADC1	ADC0	10x
01010[1]		ADC0	ADC0	200x
01011[1]		ADC1	ADC0	200x
01100[1]		ADC2	ADC2	10x
01101[1]		ADC3	ADC2	10x
01110[1]		ADC2	ADC2	200x
01111[1]		ADC3	ADC2	200x
10000[1]		ADC0	ADC1	1x
10001[1]		ADC1	ADC1	1x
10010[1]		ADC2	ADC1	1x
10011[1]		ADC3	ADC1	1x
10100[1]		ADC4	ADC1	1x
10101[1]		ADC5	ADC1	1x
10110[1]		ADC6	ADC1	1x
10111[1]		ADC7	ADC1	1x
11000[1]		ADC0	ADC2	1x
11001[1]		ADC1	ADC2	1x
11010[1]		ADC2	ADC2	1x
11011[1]		ADC3	ADC2	1x
11100[1]	Не застосовується	ADC4	ADC2	1x
11101[1]		ADC5	ADC2	1x
11110	1,22 В (1,1 В[1])	Не застосовується		
11111	0 В (GND)			
[1] У моделях ATmega165x/325x/3250x/645x/6450x/1251x/2561x				

7.3.6. Збереження результату перетворення

Після завершення перетворення (під час встановлення в одиницю прапорця ADIF регістра ADCSR) його результат зберігається в регістрі даних АЦП. Оскільки АЦП має 10 розрядів, цей регістр фізично розміщено у двох регістрах введення/виведення ADCH:ADCL, доступних тільки для читання. Ці регістри розміщено за адресами $05:$04 і у разі включення МК містять значення «$0000».У початковому стані результат перетворення вирівнюється вправо (старші 6 розрядів регістра ADCH – не є значущими). Однак він може вирівнюватися також вліво (молодші 6 розрядів регістра ADCL будуть незначущими).

Для керування вирівнюванням результату перетворення призначено розряд ADLAR регістра ADMUX. Якщо цей розряд встановлено в одиницю, результат перетворення вирівнюється за лівою границею 16-розрядного слова, якщо скинутий в нуль – за правою границею.

Під час використання диференціального режиму перетворення результат подається в коді двійкового доповнення до двох (у додатковому коді).

У табл. 7.7 наведено приклади вирівнювання результату вліво та вправо.

Звернення до регістів ADCH і ADCL для отримання результату перетворення повинно виконуватися в певній послідовності: спочатку необхідно прочитати регістр ADCL, а потім ADCH.

Таблиця 7.7. **Вирівнювання результату АЦП**

ADLAR	**Розряд**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**ADCH**
0		–	–	–	–	–	–	ADC9	ADC8	
		ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	**ADCL**
	Розряд	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
	R/$\overline{W}$	R	R	R	R	R	R	R	R	**ADCH**
		R	R	R	R	R	R	R	R	**ADCL**
	Поч. зн.	0	0	0	0	0	0	0	0	**ADCH**
		0	0	0	0	0	0	0	0	**ADCL**
1	**Розряд**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	
		ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	**ADCH**
		ADC1	ADC0	–	–	–	–	–	–	**ADCL**
	Розряд	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
	R/$\overline{W}$	R	R	R	R	R	R	R	R	**ADCH**
		R	R	R	R	R	R	R	R	**ADCL**
	Поч. зн.	0	0	0	0	0	0	0	0	**ADCH**
		0	0	0	0	0	0	0	0	**ADCL**

Ця вимога пов’язана з тим, що після звернення до регістра ADCL процесор блокує доступ до регістрів даних з боку АЦП доти, поки не буде прочитано регістр ADCH. Завдяки цьому можна бути впевненим, що під час читання регістрів ADCH, ADCL у них будуть перебувати складові того самого результату. Відповідно, якщо чергове перетворення завершиться до звернення до регістра ADCH, результат перетворення буде загублено.

Інакше, якщо результат перетворення вирівнюється вліво й досить точності 8-розрядного значення, для отримання результату можна прочитати тільки вміст регістра ADCH.

7.3.7. Результат перетворення

Для каналів з однополярним входом результат перетворення визначається виразом

$$ADC = \frac{1023 \cdot U_{IN}}{U_{REF}}, \qquad (7.9)$$

де U_{IN} – значення вхідної напруги в мілівольтах, а U_{REF} = $U_{ДОН}$ – величина джерела опорної напруги у мілівольтах.

Коефіцієнт передачі АЦП визначається формулою

$$К_{ПЕР} = 1023/U_{REF}[МЗР/мВ]. \quad (7.10)$$

На рис. 7.13 зображено функцію перетворення АЦП в однополярному режимі. Код 0x000 відповідає рівню аналогової землі, а 0x3FF – рівню джерела опорної напруги мінус один крок квантування за напругою.

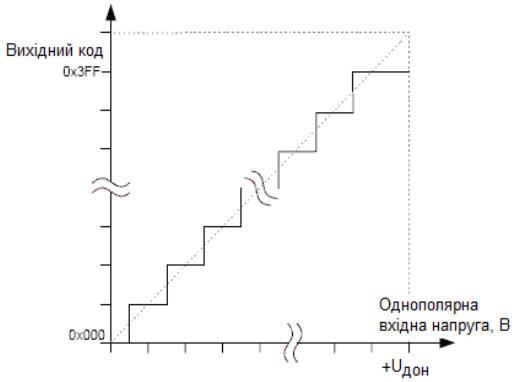


Рис. 7.13. Функція перетворення АЦП у разі зміни однополярного сигналу

Зв’язок між вхідним сигналом та вихідними кодами для однополярного режиму відображає табл. 7.8.

Таблиця 7.8. **Зв’язок між вхідним сигналом і вихідним кодами**

$U_{АЦПn}$*	**Зчитаний код**	**Відповідне десяткове значення**
$U_{АЦПm}$ + $U_{ДОН}$	0x3FF	1023
$U_{АЦПm}$ + 0,999 $U_{ДОН}$	0x3FF	1023
$U_{АЦПm}$ + 0,998 $U_{ДОН}$	0x3FE	1022
…	...	...
$U_{АЦПm}$ + 0,001 $U_{ДОН}$	0x001	1
$U_{АЦПm}$	0x000	0

* $U_{АЦПm}$ – вхідна напруга, яка дорівнює нулю, $U_{АЦПn}$ – поточне значення вхідної напруги.

Приклад 7.1. Нехай ADMUX = 0x00…0x07 (будь-який однополярний вхід). Напруга на цьому вході становить 1000 мВ; $U_{ДОН}$ = 2,56 В. Тоді код на виході АЦП = 1023 · 1000 / 2560 = 400 = 0x190.

Для каналів з диференціальним входом результат перетворення визначається виразом

$$ADC = 512 \cdot К_{П} \cdot \frac{U_{POS} - U_{NEG}}{U_{REF}}, \quad (7.11)$$

де U_{P0S} – величина напруги на додатному (неінвертуючому) вході; U_{NEG} – величина напруги на від'ємному (інвертуючому) вході, а $К_п$ – коефіцієнт підсилення.

У цьому випадку результат перетворення подано в додатковому коді, а його значення перебуває в діапазоні: $200 (–512)...$1FF (+511).

Приклад 7.2. Нехай ADMUX = 0xED (пара входів ADC3, ADC2, $К_п$ = 10, $U_{дон}$ = 2,56 В, результат з лівостороннім вирівнюванням). Напруга на вході ADC3 = 300 мВ, а на вході ADC2 = 500 мВ.

Тоді

$$КОД_{АЦП} = 512 \cdot 10 \cdot \frac{300-500}{2560} = -400 = 0x270.$$

З урахуванням обраного формату розміщення результату (лівосторонній) ADCL = 0x00, а ADCH = 0x9C. Якщо ж обрано правосторонній формат (ADLAR = 0), то ADCL = 0x70, ADCH = 0x02.

За завершенням перетворення (ADIF = 1) результат може бути зчитано з пари регістрів результату перетворення АЦП (ADCL, ADCH). У разі необхідності визначити полярність результату треба опитати старший розряд результату перетворення (ADC9 в ADCH). Якщо цей розряд дорівнює одиниці, то результат від'ємний, якщо ж нуль, то додатний.

У [3; 4] наведено рекомендації, що дозволяють підвищити точність перетворення та найбільшою мірою використати можливості АЦП.

7.3.8. Моделювання модуля аналого-цифрового перетворювача та цифрового вольтметра

Моделювання модуля АЦП та цифрового вольтметра у пакеті PROTEUS 8.6 описано у [3], де наведено схеми алгоритмів роботи моделей та робочі програми мовою С. Отримані результати моделювання підтвердили працездатність цих алгоритмів та робочих програм.

7.4. Застосування модуля цифро-аналогового перетворювача під час виведення цифрової інформації з мікропроцесорних систем

7.4.1. Загальні відомості про модуль цифро-аналогового перетворювача

Цифро-аналогові перетворювачі призначені для перетворення цифрових сигналів в аналогові. Вони служать для сполучення цифрових і аналогових пристроїв та широко використовуються для керування аналоговими пристроями за допомогою МК у таких галузях техніки, як системи керування технологічними процесами: виконавчі пристрої програмованих верстатів, роботів і т. ін.; дискретна автоматика; вимірювальна автоматика тощо. Серед

різних схемних виконань ЦАП застосування знаходять перетворювачі з резисторною матрицею R-2R із підсумовуванням струмів або підсумовування напруг, наприклад, AD7520 та MAX506 [2; 3].

7.4.2. Мікросхема цифро-аналогового перетворювача AD7520

Мікросхема ЦАП типу AD7520 знаходить застосування в різній апаратурі завдяки малій споживаній потужності, досить високій швидкодії, невеликим габаритам і т. ін [1; 2].

Мікросхему призначено для перетворення 10-розрядного паралельного двійкового коду на цифрових входах у струм на аналоговому виході, який пропорційний значенням коду та опорної напруги. Вона виконана за КМОН-технологією з використанням резисторної матриці R-2R та підсумовуванням струмів.

Вираз для визначення напруги на виході має вигляд:

$$U_{ВИХ} = -\frac{U_{ОП}}{2^{10}} \sum_{i=0}^{9} a_i \cdot 2^i, \tag{7.12}$$

де a_i – значення i-го розряду вхідного двійкового коду: 0/1; $U_{ОП}$ – значення опорної напруги.

Максимальне значення вихідної напруги:

$$U_{ВИХ\max} = -U_{ОП}\left(1 - 2^{-10}\right). \tag{7.13}$$

Коефіцієнт передачі:

$$K_{ЦАП} = -\frac{U_{ОП}}{2^{10}}\left[\frac{\text{мВ}}{МЗР}\right]. \tag{7.14}$$

Коли $U_{оп}$ = –10,24 В = –10240 мВ,

$$U_{ВИХ\max} = \frac{10{,}24 \cdot 1023}{1024} = 10{,}23\ В,$$

$$K_{ЦАП} = \frac{10240}{1024} = 10\left[\frac{мВ}{МЗР}\right].$$

7.4.3. Мікросхема цифро-аналогового перетворювача MAX506

У [1; 2] розглянуто ЦАП фірми «MAXIM» – MAX506, яка є чотириканальним 8-бітовим ЦАП, а також виконує функції шинного формувача, регістрів і схеми узгодження рівнів.

Функціональну схему мікросхеми наведено у [2]. Вона виконана за КМОН-технологією з використанням резисторної матриці R-2R та підсумовуванням напруг.

Вихідна напруга, максимальна вихідна напруга та коефіцієнт передачі визначаються виразами

$$U_{\text{вих}} = \frac{\text{U}_{\text{ОП}}}{2^8} \cdot N_B \quad , \qquad (7.15)$$

$$U_{\text{вих.max}} = \frac{\text{U}_{\text{ОП}}}{2^8} \sum_{i=0}^{7} 2^i , \qquad (7.16)$$

$$K_{\text{ЦАП}} = \frac{U_{\text{ОП}}}{2^8} \left[\frac{\text{В}}{\text{МЗР}} \right], \qquad (7.17)$$

де $N_B = \sum_{i=0}^{7} a_i \cdot 2^i$ – десятковий еквівалент значення вхідного двійкового коду.

Якщо, наприклад, $U_{\text{ОП}}$ = 5,12 В, тоді $K_{\text{ЦАП}}$ = 20 mB/МЗР.

7.5. Особливості архітектури модуля цифро-аналогового перетворювача в AVR-мікроконтролерах

7.5.1. Загальна характеристика модуля

Частина AVR-мікроконтролерів, наприклад, XMega мають модуль 12-розрядного ЦАП. Його виконано за КМОН-технологією з використанням резисторної матриці R-2R та підсумовуванням напруг.

Основні характеристики модуля наведено у [3; 11].

7.5.2. Розрахунок та функціональна схема модуля

Вихідна напруга та коефіцієнт передачі каналу ЦАП обчислюються за наступними виразами:

$$U_{\text{цап}} = \frac{CHnDATA}{4096} \cdot U_{ref}, \qquad (7.18)$$

$$K_{\text{цап}} = U_{ref}/4096 \text{ [мВ/МЗР]}, \qquad (7.19)$$

де $CHnDATA$ – десятковий еквівалент числа, яке зберігається у регістрі даних, n = 0/1 – номер регістра, U_{ref} – опорна напруга.

На рис. 7.14 зображено функціональну схему модуля.

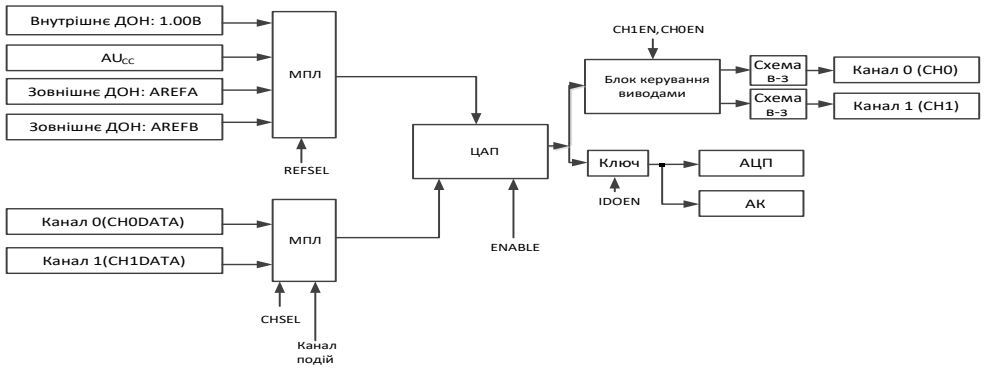


Рис. 7.14. Функціональна схема модуля ЦАП

Схема функціонує таким чином. Вхідні дані для перетворення записуються у вхідні регістри даних CH0DATA (в одноканальному режимі роботи) або CH0DATA та CH1DATA (у двоканальному режимі роботи).

Після початку перетворення дані з регістрів крізь мультиплексор потрапляють безпосередньо на схему ЦАП, де за допомогою джерела опорної напруги U_{ref}, обраного відповідними бітами регістра керування, виконується перетворення. Регістри керування задають режим живлення, вибір каналу, джерело запуску, дані калібрування тощо. Аналоговий сигнал, отриманий під час перетворення, залежно від обраного режиму роботи подається через блок керування виводами в канал 0 та канал 1, та на вхід аналогового компаратора та АЦП, залежно від значення розряду IDOEN регістра CTRLA.

7.5.3. Джерела опорної напруги

Для ЦАП у МК XMega можуть використовуватися такі джерела опорної напруги:

- внутрішнє джерело опорної напруги зі значенням 1,00 В;
- зовнішнє джерело AU_{cc};
- зовнішня напруга, що подається на вивід AREF порту A та/або B.

7.5.4. Вихідні канали цифро-аналогового перетворювача

В якості виходу ЦАП можуть бути або один аналоговий вихід (канал 0), або два окремих виходи зі схемами вибірки-зберігання (в-з). Виходи схеми вибірки-зберігання можуть працювати повністю незалежно, дозволяючи генерувати два аналогових сигнали, що розрізняються як за амплітудою, так і за частотою.

Для кожного з виходів схем вибірки-зберігання передбачено окремі регістри даних та регістри керування. Вихідна напруга ЦАП може бути подана на вхід аналогового компаратора або АЦП (рис. 7.14).

7.5.5. Режими роботи цифро-аналогового перетворювача

7.5.5.1. Одноканальний режим роботи

Модуль ЦАП може працювати в одноканальному або двоканальному режимі роботи.

В одноканальному режимі роботи регістр даних CH0DATA через мультиплексор з’єднується з входом ЦАП (рис. 7.15).

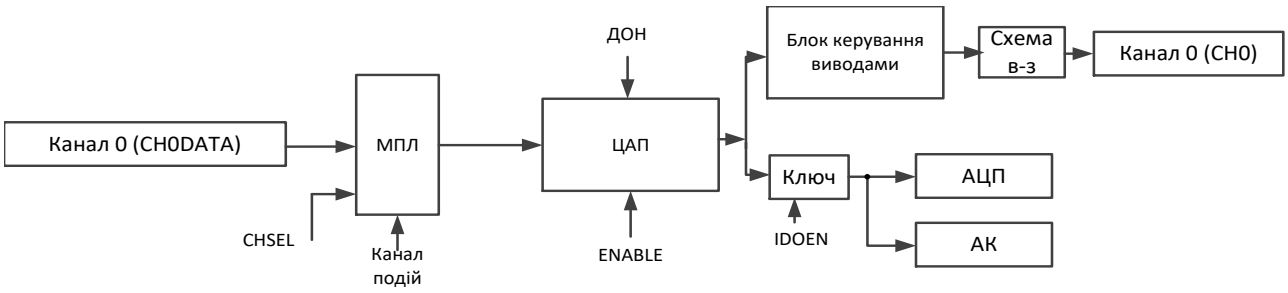


Рис. 7.15. Функціональна схема роботи ЦАП в одноканальному режимі

У [3] наведено послідовність програмування ЦАП для одноканального режиму роботи.

7.5.5.2. Двоканальний режим роботи

У двоканальному режимі роботи ЦАП по черзі перетворює дані з CH0DATA та CH1DATA (рис. 7.16).

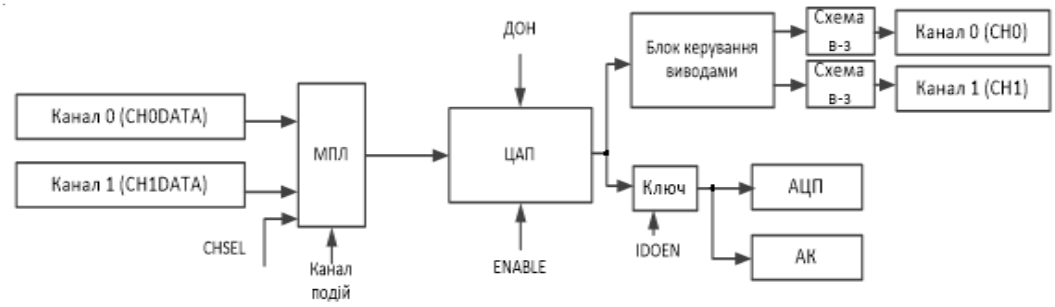


Рис. 7.16. Функціональна схема роботи ЦАП у двоканальному режимі

Блоки вибірки-зберігання використовуються для збереження значень між перетвореннями.

Для коректної видачі вихідного значення на два виходи ЦАП повинен регулярно оновлювати канали [3; 11].

У [3] наведено послідовність програмування ЦАП для двоканального режиму роботи.

7.5.6. Тактування модуля

Модуль ЦАП тактується сигналом синхронізації від зовнішнього тактового генератора. Інтервал перетворень і частота оновлення в двоканальному режимі задаються кратними періоду сигналу синхронізації.

7.5.7. Обмеження часових характеристик

Для коректної роботи модуля ЦАП необхідно дотримуватися ряду часових обмежень, які задаються кратними періоду сигналу синхронізації. Недотримання обмежень може погіршити точність перетворення.

Час оновлення ЦАП – це інтервал часу між оновленнями каналів у двоканальному режимі. Величина цього інтервалу не повинна перевищувати 30 мкс. Цей параметр впливає на те, щоб забезпечувати підтримку стабільного

вихідного сигналу на два виходи у двоканальному режимі роботи. Ця необхідність виникає тому, що схема буде втрачати амплітуду сигналу з часом, як конденсатор втрачає напругу через під'єднаний до нього резистор. Більш висока частота оновлення спричиняє більш високе енергоспоживання.

Інтервал вибірки та перетворення ЦАП – це проміжок часу від моменту початку перетворення в каналі до запуску нового перетворення.

Цей проміжок не має бути менше 1 мкс в одноканальному режимі й 1,5 мкс у двоканальному режимі (режимі вибірки-зберігання). Фактично це час, необхідний для перетворення та зберігання значення для аналогового виходу. Це аналогічно процесу зарядки конденсатора. Якщо часовий інтервал завеликий, можна втратити інформацію з високою швидкістю наростання вхідної напруги. Якщо інтервал вибірки та перетворення менший ніж інтервал оновлення, то канали будуть оновлені у визначений час, навіть якщо зайві (повторні) перетворення було виконано між інтервалами, що викликано ручним оновленням даних у регістрі.

У [3; 11] наведено часові діаграми оновлення та запитів перетворення каналів 0 та 1, якщо ЦАП запрограмовано на роботу у двоканальному режимі.

7.5.8. Режим енергозбереження

У разі необхідності зниження споживаного струму модулем ЦАП під час перетворень можна перевести в економічний режим роботи. У цьому режимі між виконанням перетворень модуль переходить у відключений стан. Робота в цьому режимі супроводжується збільшенням часу перетворення під час запуску нового перетворення.

7.5.9. Система подій

Система подій МК XMega − це набір функцій, який дозволяє периферійним модулям взаємодіяти один з одним без втручання центрального процесора. Деякі периферійні модулі можуть генерувати події, часто за тими самими умовами, що і переривання. Ці події проходять через систему маршрутизації подій до споживачів подій, де споживачами можуть бути ініційовані певні дії. Центральний процесор не бере участі в цьому процесі, за винятком етапу налаштування. Наприклад, можна ініціювати захоплення входу таймера/лічильника під час натискання користувачем на кнопку, або почати аналого-цифрове перетворення у разі переповнення таймера/лічильника.

7.5.10. Програмування модуля

7.5.10.1. Регістри даних

На рис. 7.17…7.20 наведено опис двох регістрів: CHnDATAH і CHnDATAL (де n = 0/1) − відповідно старша і молодша частини 12-розрядного

значення CHnDATA, яке перетворюється в аналоговий сигнал. За замовчуванням 12-розрядів поділяються на 8 розрядів у CHnDATAL і 4 CHnDATAH з позиції молодшого значущого розряду − вирівнювання вправо. Для деяких програм краще використовувати лівоспрямовані розряди. Вибрати дані з вирівнюванням вліво можна, встановлюючи розряд LEFTADJ у регістрі CTRLC.

	розряд		7	6	5	4	3	2	1	0
Вирівнювання вправо		+0x18	CHDATA [7:0]							
Вирівнювання вліво		+0x18	CHDATA [3:0]				-	-	-	-
Вирівнювання вправо	Читання/Запис		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Вирівнювання вліво	Читання/Запис		R/W	R/W	R/W	R/W	R	R	R	R
Вирівнювання вправо	Початкове значення		0	0	0	0	0	0	0	0
Вирівнювання вліво	Початкове значення		0	0	0	0	0	0	0	0

Рис. 7.17. CH0DATAL − молодший байт регістра даних 0

	розряд		7	6	5	4	3	2	1	0
Вирівнювання вправо		+0x19	-	-	-	-	CHDATA [11:8]			
Вирівнювання вліво		+0x19	CHDATA [11:4]							
Вирівнювання вправо	Читання/Запис		R	R	R	R	R/W	R/W	R/W	R/W
Вирівнювання вліво	Читання/Запис		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Вирівнювання вправо	Початкове значення		0	0	0	0	0	0	0	0
Вирівнювання вліво	Початкове значення		0	0	0	0	0	0	0	0

Рис. 7.18. CH0DATAH − старший байт регістра даних 0

	розряд		7	6	5	4	3	2	1	0
Вирівнювання вправо		+0x1A	CHDATA [7:0]							
Вирівнювання вліво		+0x1A	CHDATA [3:0]				-	-	-	-
Вирівнювання вправо	Читання/Запис		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Вирівнювання вліво	Читання/Запис		R/W	R/W	R/W	R/W	R	R	R	R
Вирівнювання вправо	Початкове значення		0	0	0	0	0	0	0	0
Вирівнювання вліво	Початкове значення		0	0	0	0	0	0	0	0

Рис. 7.19. CH1DATAL − молодший байт регістра даних 1

	розряд		7	6	5	4	3	2	1	0
Вирівнювання вправо		+0x1B	-	-	-	-	CHDATA [11:8]			
Вирівнювання вліво		+0x1B	CHDATA [11:4]							
Вирівнювання вправо	Читання/Запис		R	R	R	R	R/W	R/W	R/W	R/W
Вирівнювання вліво	Читання/Запис		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Вирівнювання вправо	Початкове значення		0	0	0	0	0	0	0	0
Вирівнювання вліво	Початкове значення		0	0	0	0	0	0	0	0

Рис. 7.20. CH1DATAH − старший байт регістра даних 1

7.5.10.2. Регістри керування

CTRLA − регістр керування A (рис. 7.21)

Bit	7	6	5	4	3	2	1	0	
+0x00	-	-	-	IDOEN	CH1EN	CH0EN	.	ENABLE	CTRLA
Read/Write	R	R	R	R/W	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рис. 7.21. Формат регістра CTRLA

Bits 7:5 – зарезервовано;

Bit 4 − IDOEN: дозвіл виведення в АЦП та аналоговий компаратор. Встановлення цього розряду в одиницю направляє внутрішній вихід ЦАП до АЦП та аналоговий компаратор;

Bit 3 − CH1EN: дозвіл виведення каналу 1. Встановлення цього розряду в одиницю виводить результат перетворення на вихід мікросхеми, інакше канал доступний тільки для внутрішнього використання.

Bit 2 − CH0EN: дозвіл виведення каналу 0. Встановлення цього розряду в одиницю виводить результат перетворення на вихід мікросхеми, інакше канал доступний тільки для внутрішнього використання;

Bit 1 – зарезервовано;

Bit 0– ENABLE: дозвіл ЦАП. Встановлення цього розряду в одиницю дозволяє функціонування ЦАП.

CTRLB − регістр керування B (рис. 7.22)

Bit	7	6	5	4	3	2	1	0	
+0x01	-	CHSEL[1:0]		-	-	-	CH1TRIG	CH0TRIG	CTRLB
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рис. 7.22. Формат регістра CTRLB

Bit 7 – зарезервовано;

Bits 6:5 – CHSEL [1:0]: вибір режиму роботи ЦАП (табл. 7.9). Ці розряди програмують режими роботи ЦАП: одно- або двоканальний;

Bits 4:2 – зарезервовано;

Bit 1 – CH1TRIG: режим автоматичного перетворення каналу 1. Якщо розряд встановлено в одиницю, то подія, яку встановлено у регістрі EVCTRL, ініціює перетворення за умови, що дані у регістрі CH1DATA ще не перетворено;

Bit 0 – CH0TRIG: режим автоматичного перетворення каналу 0. Якщо розряд встановлено в одиницю, то подія, яку встановлено у регістрі EVCTRL, ініціює перетворення за умови, що дані у регістрі CH0DATA ще не перетворено.

Таблиця 7.9. **Вибір режиму роботи ЦАП**

CHSEL [1:0]	Режим роботи
00	Одноканальний (працює тільки канал 0)
01	Зарезервовано
10	Двоканальний (вибірка-зберігання для каналів 0 та 1)
11	Зарезервовано

CTRLC – регістр керування С (рис. 7.23)

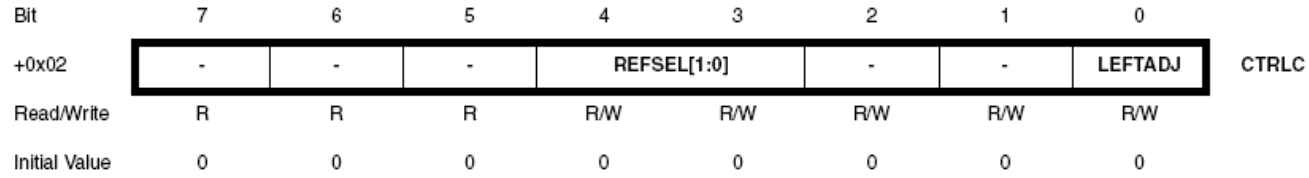


Рис. 7.23. Формат регістра CTRLC

Bits 7:5 – зарезервовано;

Bits 4:3 − REFSEL [1:0]: вибір джерела опорної напруги. Джерело опорної напруги і, відповідно, діапазон перетворення ЦАП вибирається згідно з табл. 7.10;

Bit 2:1 – зарезервовано;

Bit 0 – LEFTADJ: лівоспрямовані дані; якщо розряд встановлено в одиницю, то регістри CH0DATA і CH1DATA – лівоспрямовані.

Таблиця 7.10. **Вибір джерела опорної напруги ЦАП**

REFSEL [1:0]	Джерело опорної напруги
00	Вбудоване джерело 1,00 В
01	AU_{cc}
10	Вивід AREF порту А
11	Вивід AREF порту В

EVCTRL – регістр керування подіями (рис. 7.24)

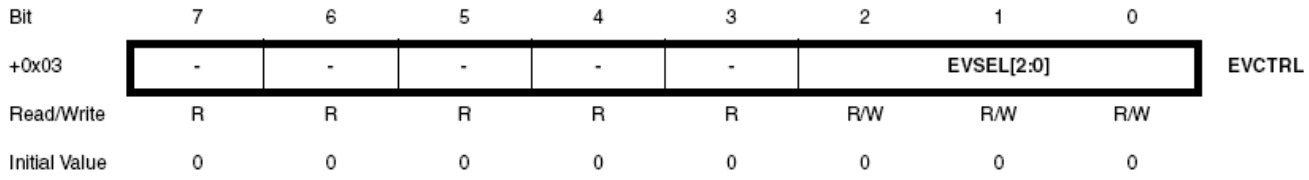


Рис. 7.24. Формат регістр EVCTRL

Bits 7:3 – зарезервовано;

Bits 2:0 – обирають маршрут подій (табл. 7.11).

Таблиця 7.11. **Вибір маршруту подій ЦАП**

EVSEL [2:0]	Маршрут подій
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

TIMCTRL – регістр керування часовими інтервалами (рис. 7.25)

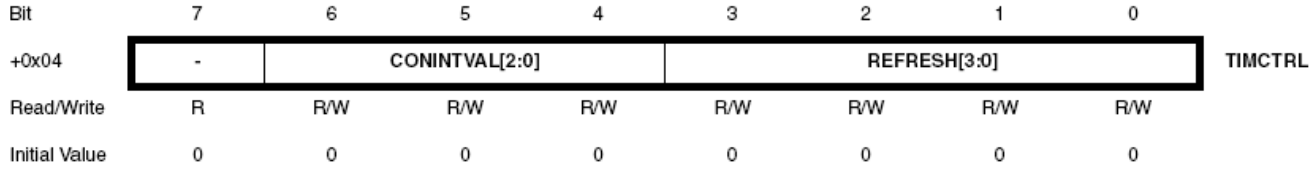


Рис. 7.25. Формат регістра TIMCTRL

Bit 7 – зарезервовано;

Bits 6:4 – CONINTVAL [2:0]: інтервал між двома завершеними перетвореннями. Інтервал має бути встановлено відповідно периферійному тактовому генератору, щоб переконатися, що нові перетворення не почнуться доти, поки результат попереднього перетворення не запишеться. Інтервал між перетвореннями ЦАП не має бути менше ніж 1 мкс в одноканальному режимі, і не менше ніж 1,5 мкс у двоканальному режимі (табл. 7.12);

Bits 3:0 – REFRESH [3:0]: керування часом оновлення результатів. Ці розряди контролюють часовий інтервал між оновленням результату в двоканальному режимі. Інтервал має бути встановлено відповідно з периферійним тактовим генератором, щоб уникнути втрати точності конвертованого значення (табл. 7.13).

Таблиця 7.12. **Доступні установки інтервалу між перетвореннями**

CONINTVAL [2:0]	Кількість тактів між перетвореннями в одноканальному режимі	Кількість тактів між перетвореннями у двоканальному режимі
0	1	1
1	2	3
10	4	6
11	8	12
100	16	24
101	32	48
110	64	96
111	128	192

Таблиця 7.13. **Частота оновлення результатів**

REFRESH [3:0]	Кількість тактів між оновленням результату
0	16
1	32
10	64
11	128
100	256
101	512
110	1024
111	2048
1000	4096
1001	8192
1010	16384
1011	32768
1100	65536
1101	Зарезервовано
1110	Зарезервовано
1111	Оновлення вимкнено

STATUS – регістр статусу ЦАП (рис. 7.26)

Bit	7	6	5	4	3	2	1	0	
+0x05	-	-	-	-	-	-	CH1DRE	CH0DRE	STATUS
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рис. 7.26. Формат регістра STATUS

Bits 7:2 – зарезервовано;

Bit 1 – CH1DRE: регістр даних для каналу 1 порожній. Якщо розряд скинуто у нуль, запис у регістр даних може призвести до втрати результату перетворення. Цей розряд може використовуватись для запитів DMA [3; 11];

Bit 0 – CH0DRE: регістр даних для каналу 0 порожній. Якщо розряд скинуто у нуль, запис у регістр даних може призвести до втрати результату перетворення. Цей розряд може використовуватись для запитів DMA.

Регістри калібрування

Для досягнення високої точності перетворення, треба калібрувати коефіцієнти передачі і зміщення модуля ЦАП. Калібрувальні значення для корегування коефіцієнта передачі і зміщення є 7-розрядними.

Найкращі результати досягаються під час калібрування в тих самих умовах, в яких планується використовувати ЦАП, тобто за одних і тих самих U_{REF}, вихідному каналі, часі перетворення й інтервалі оновлення.

З урахуванням похибок, теоретичну передатну функцію ЦАП можна записати таким чином:

$$U_{\text{цап}} = gain \cdot U_{ref} \cdot \frac{CHnDATA}{4096} + offset, \qquad (7.20)$$

де *gain* – коефіцієнт передачі; *offset* – коефіцієнт зсуву.

В ідеального ЦАП коефіцієнт передачі дорівнює 1, а коефіцієнт зміщення дорівнює 0. Калібрування відбувається за допомогою регістрів GAINCAL та OFFSETCAL (рис. 7.27, 7.28) і не залежить від вибору режиму.

GAINCAL – регістр калібрування підсилення ЦАП (рис. 7.27)

Bit	7	6	5	4	3	2	1	0	
+0x08	-	GAINCAL[6:0]							GAINCAL
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рис. 7.27. Формат регістра GAINCAL

Bit 7 – зарезервовано;

Bits 6:0 – GAINCAL [6:0] – корегування коефіцієнта передачі ЦАП. Ці розряди використовуються, щоб компенсувати помилку підсилення ЦАП.

OFFSETCAL – регістр калібрування зсуву ЦАП (рис. 7.28)

Bit	7	6	5	4	3	2	1	0	
+0x09	-	OFFSETCAL[6:0]							OFFSETCAL
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рис. 7.28. Формат регістра OFFSETCAL

Bit 7 – зарезервовано;

Bits 6:0 – OFFSETCAL [6:0] – корегування коефіцієнта зсуву. Ці розряди використовуються щоб компенсувати помилку зсуву ЦАП.

7.6. Зв’язок мікропроцесорів/мікроконтролерів з модемом

7.6.1. Загальна характеристика обміну інформацією між МП/МК

Обмін інформацією між МП/МК і модемом може здійснюватися через інтерфейс RS-232 [2; 3].

На рис. 7.29 зображено структурну схему сполучення МП/МК з модемом за допомогою інтерфейсу RS-232, яка включає:

– УАПП;
– пристрій перетворення рівнів;
– роз’єм RS-232;
– буферний регістр адреси;
– шинний формувач.

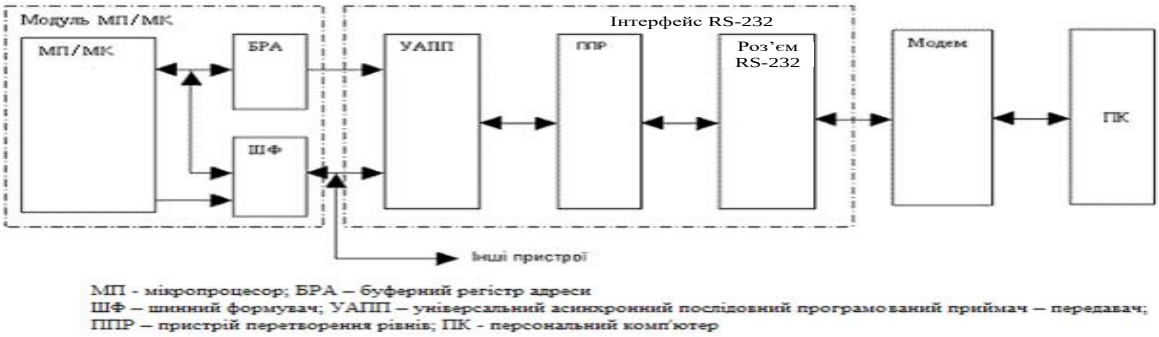


Рис. 7.29. Структурна схема сполучення МП/МК з модемом за допомогою інтерфейсу RS-232

7.6.2. Універсальний асинхронний приймач/передавач

Універсальний асинхронний приймач/передавач перетворює дані з паралельного формату в послідовний під час передачі (виведення) з МП/МК і з послідовного формату в паралельний під час прийому (введення) у МП/МК. В якості УАПП може використовуватись, наприклад, мікросхема TL16550. У МПС з використанням МК окремий модуль УАПП необхідний у випадку, коли МК не

має вбудованого модуля УАПП. Обмін інформацією у МПС з використанням УАПП ведеться в асинхронному послідовному старт-стопному режимі.

Формат даних, які передаються у канал зв'язку в послідовному форматі, зображено на рис. 7.30.

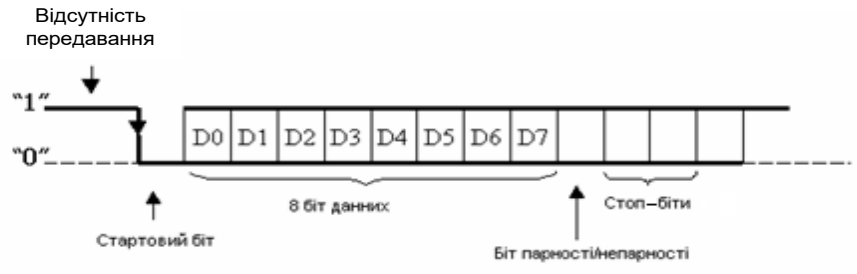


Рис. 7.30. Формат даних інтерфейсу RS-232

Власне дані (5, 6, 7 чи 8 біт) супроводжуються стартовим нульовим бітом, бітом парності/непарності (якщо такий контроль програмно передбачено) і стоповим одиничним сигналом, що включає 1; 1,5 чи 2 стоп-біти. Одержавши стартовий біт, приймач вибирає з лінії біти даних через визначені інтервали часу. Дуже важливо, щоб тактові частоти приймача і передавача були однаковими та стабільними. Швидкість передачі за RS-232 може вибиратися з ряду: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 біт/с (бод).

7.6.3. Пристрій перетворення рівнів

Усі сигнали RS-232 передаються/приймаються спеціально обраними рівнями, що забезпечують високу завадостійкість зв'язку (рис. 7.31).

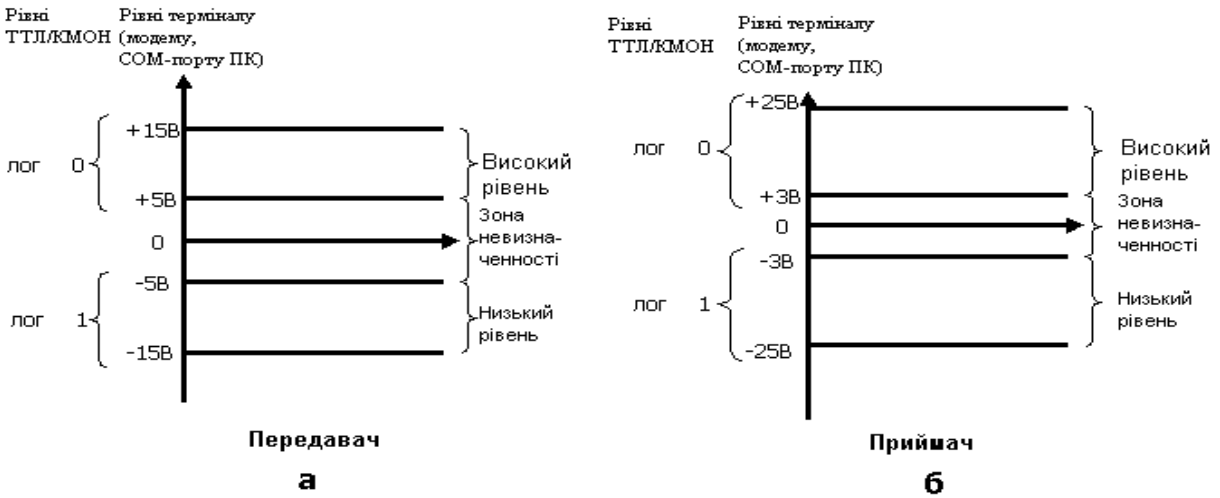


Рис. 7.31. Рівні сигналів RS-232 на передавальній (*а*) та на приймальній (*б*) кінцях лінії зв'язку

Дані передаються/приймаються в інверсному вигляді: цифровій логічній одиниці відповідає низький рівень, а логічному нулю – високий рівень.

Як видно з рис. 7.31 під час передачі логічного нуля на виході інтерфейсу формується високий рівень напруги в діапазоні: +5 В…+15 В, під час передачі логічної одиниці – низький рівень напруги в діапазоні: –5 В…–15 В.

Під час прийому на вхід інтерфейсу надходить високий рівень напруги в діапазоні: +3 В…+25 В, що несе інформацію про логічний нуль, чи низький рівень напруги в діапазоні: –3 В…–25 В, що відображає логічну одиницю.

Таким чином, для узгодження транзисторно-транзисторна логіка з переходами Шоткі/комплементарний метал-оксид-напівпровідник рівнів сигналів, що діють у МПС, з рівнями сигналів послідовного інтерфейсу, що передаються у лінію зв'язку або приймаються з лінії зв'язку використовують пристрій перетворення рівнів.

Різні варіанти схемної реалізації пристрою перетворення рівнів розглянуто у [10], одним із яких є застосування мікросхеми MAX232A фірми MAXIM.

7.6.4. Роз'єм RS-232

Для зв'язку інтерфейсу RS-232 із зовнішнім терміналом (модемом) або ПК можуть використовуватися, наприклад, 9-контактні роз'єми (рис. 7.32).

Роз'єм RS-232

	Контакт	Ланцюг	
→	1	DCD (виявлення несучої)	
→	2	RxD (вхід приймача)	
←	3	TxD (вихід передавача)	
←	4	DTR (запит передавача терміналу)	
→	5	SG (сигнальна земля)	
→	6	DSR (готовність передавача терміналу)	
←	7	RTS (запит приймача терміналу)	
→	8	CTS (готовність приймача терміналу)	
→	9	RI (індикатор виклику)	

Модем (термінал, ПК)

ППР, УАПП, МП

Рис. 7.32. 9-контактний роз'єм RS-232

Призначення контактів роз'єма таке:

– SG – сигнальне заземлення, нульовий провід;
– TxD – дані, що передаються МП/МК у послідовному коді (від'ємна логіка);
– RxD – дані, що приймаються МП/МК у послідовному коді (від'ємна логіка);
– DCD – виявлення несучої даних (детектування сигналу, що приймається МП/МК);
– DTR – запит передавача термінала (модему);
– DSR – готовність передавача термінала (модему);
– RTS – запит приймача термінала (модему);
– CTS – готовність приймача термінала (модему);
– RI – індикатор виклику. Вказує на прийом модемом сигналу виклику від телефонної мережі.

На рис. 7.29 було наведено структурну схему пристрою сполучення МП/МК і модему за допомогою інтерфейсу RS-232. Функціональні схеми двох модулів цієї схеми, характеристику окремих регістрів мікросхеми УАПП TL16550 та рекомендації по їх програмуванню наведено у [2].

Контрольні запитання та завдання

1. Опишіть особливості введення/виведення аналогової інформації у МПС.
2. Опишіть особливості програмування АЦП AD571.
3. Поясніть принцип роботи АЦП послідовного наближення.
4. Обґрунтуйте необхідність застосування пристрою вибірки-зберігання у МПС.
5. Як розрахувати абсолютну та відносну похибки АЦП від квантування за рівнем?
6. З яких міркувань обирається число розрядів АЦП?
7. Як обирається величина кроку квантування за часом?
8. Як змінюється коефіцієнт передачі та максимальне значення вхідної напруги АЦП AD571 залежно від кількості та номерів виходів АЦП?
9. Як розрахувати основні параметри АЦП на основі мікросхем AD571 та MAX 154?
10. Як розрахувати основні параметри ЦАП на основі мікросхем AD7520 та MAX 506?
11. До яких моделей AVR-мікроконтролерів належить модуль АЦП? Назвіть його розрядність.
12. Назвіть основні параметри АЦП.
13. Назвіть основні джерела опорної напруги для АЦП.
14. Назвіть режими роботи АЦП.

15. Яку роль у функціональній схемі модуля АЦП виконують: дешифратор, мультиплексор та пристрій вибірки та зберігання?

16. В якому регістрі зберігається результат перетворення?

17. Який регістр відповідає за налаштування мультиплексора АЦП?

18. Наведіть та поясніть формати регістрів стану і керування АЦП.

19. За якими перериваннями може відбуватися запуск АЦП?

20. Яку функцію виконує попередній дільник?

21. Як формується тактовий сигнал АЦП?

22. Поясніть часові діаграми роботи АЦП.

23. Назвіть способи вирівнювання результату АЦП.

24. Яке вирівнювання слід використовувати якщо досить точності 8-розрядного значення?

25. Яким буде результат перетворення для каналів з однополярним входом?

26. Яким буде результат перетворення для каналів з диференціальним входом?

27. Якою формулою визначається коефіцієнт передачі АЦП?

28. Поясніть функцію перетворення АЦП під час зміни однополярного сигналу.

29. Назвіть методи підвищення точності перетворення АЦП.

30. Опишіть відмінності в програмах під час моделювання модуля АЦП та цифрового вольтметра.

31. Опишіть особливості програмування мовою С виведення на дисплей результату роботи модуля АЦП та цифрового вольтметра.

32. Як розраховується відносна похибка АЦП від квантування за рівнем?

33. Чим відрізняється виведення результату моделювання цифрового вольтметра від моделювання модуля АЦП?

34. Поясніть призначення УАПП.

35. Назвіть та поясніть призначення пристрою перетворення рівнів.

36. Наведіть та поясніть структурну схему сполучення МП/МК та модему за допомогою інтерфейсу RS-232.

37. Назвіть та опишіть регістри, які використовуються для програмування УАПП.

38. Як розраховується та програмується швидкість обміну даними за допомогою УАПП ?

39. Опишіть відмінності мікросхем УАПП TL16550 та TL16450.

8. МІКРОКОНТРОЛЕРНІ МЕРЕЖІ

8.1. Мережа на основі інтерфейсу I²C (TWI)

8.1.1. Види мікроконтролерних мереж

Під час проектування мікроконтролерних мереж використовуються такі інтерфейси: I²C (TWI); SPI; CAN; RS-485; RS-232; 1-WIRE і т. ін. [3]. Нижче розглянуто інтерфейси AVR-мікроконтролерів, які можуть використовуватись під час проектування мікроконтролерних мереж.

8.1.2. Особливості архітектури інтерфейсу I²C

Інтерфейс I²C (InterIC, або IIC) є двопровідним послідовним синхронним інтерфейсом, який розроблено фірмою «Philips Corporation», і призначено для зв'язку між інтегральними мікросхемами або модулями. Є ціла група I²C-сумісних пристроїв для різних додатків: ЦАП та АЦП, мікросхеми пам'яті, давачі, МК, що містять модуль I²C і т. ін. [2; 3]. Інтерфейс I²C в AVR-мікроконтролерах має назву TWI (Two Wire (Serial) Interface).

Шина інтерфейсу I²C складається із двох ліній:

– двонаправленої лінії даних (англ. SDA);
– лінії тактових синхроімпульсів (англ. SCL).

На рис. 8.1 наведено структурну схему типової мікроконтролерної мережі, що використовує для обміну даними інтерфейс (шину) I²C.

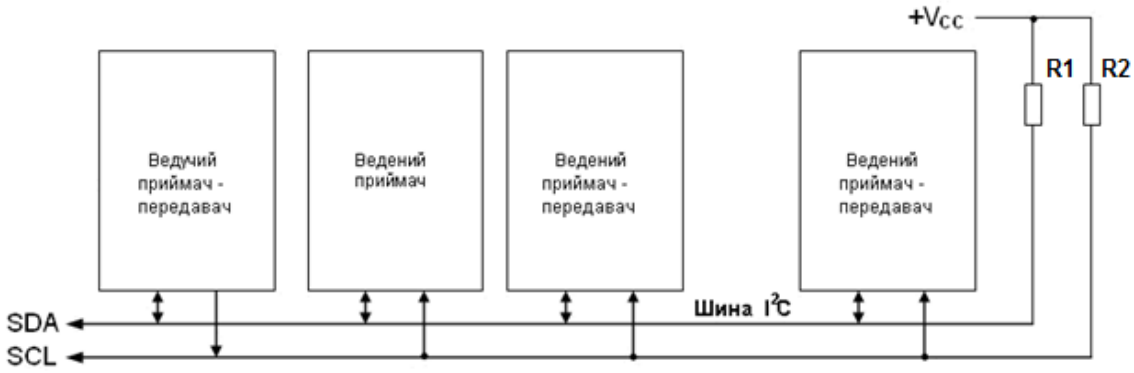


Рис. 8.1. Структурна схема мережі з інтерфейсом I²C

Лінії SDA і SCL шини з'єднані з додатним полюсом напруги живлення (+*V*cc) через підтягуючі резистори: *R*1 та *R*2.

Передавач генерує і передає повідомлення, а приймач його приймає.

Один із двох пристроїв, які беруть участь у обміні, є ведучим (Master), а інший – веденим (Slave). Ведучий пристрій керує роботою шини та формує тактові сигнали (синхросигнали) SCL.

Кожний пристрій, який використовує для обміну інтерфейс I^2C, має свою адресу. Коли ведучий пристрій бажає ініціювати обмін даними, він передає на лінію SDA адресу пристрою, з яким буде виконуватися обмін (передача/прийом). Усі ведені пристрої слідкують за адресою, яка виставляється на шину, і порівнюють її із власною мережевою адресою. Після адреси ведучий передає біт напрямку $R/\overline{W}$, який визначає, чи буде ведучий читати дані від веденого ($R/\overline{W}=1$), чи буде передавати дані веденому ($R/\overline{W}=0$). Ведений приймач після одержання адреси або даних видає на шину SDA біт підтвердження (логічний нуль). Інтерфейс I^2C може використовувати два формати адреси:

– 7-бітну адресу;
– 10-бітну адресу.

В AVR-мікроконтролерах сімейства Mega використовується 7-бітна адреса.

На рис. 8.2 наведено формат адреси, на якому використано такі позначення: S – умова «СТАРТ»; $R/\overline{W}$ – біт «*читання* / $\overline{\textit{запис}}$»; $\overline{ACK}$ – біт підтвердження.

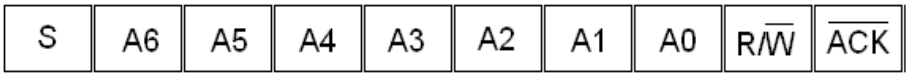
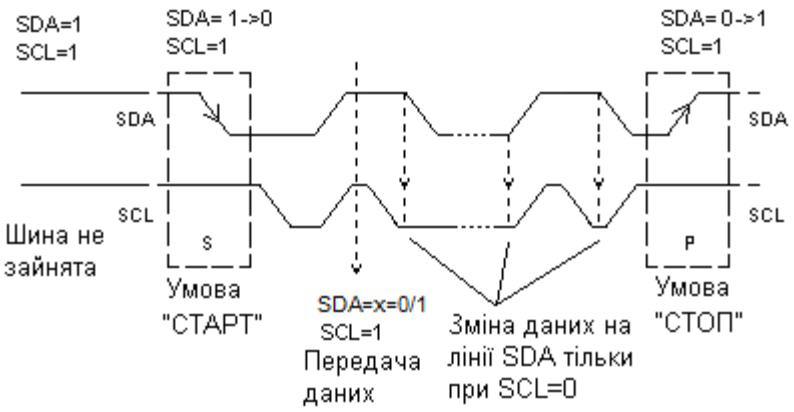


Рис. 8.2. Формат 7-бітної адреси

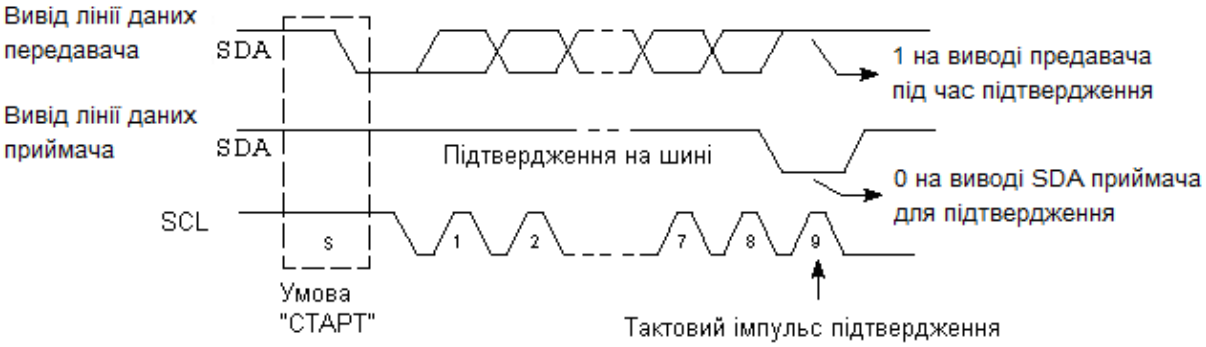
На рис. 8.3 наведено часові діаграми, які відображають стани на шині (рис. 8.3*а*), а також пояснюють формування сигналу підтвердження (рис. 8.3*б*).

Наведені діаграми відображають такі коректні стани сигналів на шині під час обміну даними:

– шина не зайнята: на обох лініях одиниці (SDA = SCL = 1);
– початок обміну даними (умова «СТАРТ» – S): зміна сигналу на лінії даних (англ. SDA) з одиниці в нуль при одиничному значенні сигналу на лінії синхронізації (англ. SCL);
– припинення передачі (умова «СТОП» – P): зміна сигналу на лінії даних з нуля в одиницю при одиничному значенні сигналу на лінії синхронізації;
– коректність даних: щоб не сформувати невірну умову «СТАРТ» або «СТОП» стан лінії даних не повинен змінюватися при одиничному значенні сигналу на лінії синхронізації. Дані можна змінювати, якщо на лінії синхронізації є низький рівень сигналу (логічний нуль). На один біт інформації на лінії SDA міститься один тактовий імпульс на лінії SCL. Кожний цикл обміну даними починається умовою «СТАРТ» і закінчується умовою «СТОП». Кількість інформаційних бітів даних, переданих між цими станами, необмежена.



а



б

Рис. 8.3. Часові діаграми: *а* – передача даних; *б* – формування сигналу підтвердження

Дані передаються побайтово. Приймач підтверджує одержання чергового байта, посилаючи біт підтвердження (логічний нуль) після прийому кожного байта даних або адреси. Активний передавач (ведений або ведучий) після передачі чергового байта формує на лінії SDA сигнал високого рівня. Ведучий пристрій (приймач або передавач) формує на лінії SCL тактовий імпульс, а приймач (ведучий або ведений) видає на SDA сигнал підтвердження низького рівня. Приймач, що генерує біт підтвердження, підключає лінію SDA до низького рівня й утримує її в цьому стані доти, поки тактовий імпульс лінії SCL не переключиться у стан низького рівня. Для припинення обміну приймач повинен залишити останній прийнятий байт без підтвердження, що автоматично викликає формування активним передавачем умови «СТОП».

Для інтерфейсу I^2C можливі чотири режими (типи) обміну даними:

1. *Ведучий передавач*: на вихід SDA передавача виводяться дані, а на лінію SCL – синхроімпульси. Перший переданий байт містить адресу веденого приймача (7 біт) і біт напряму обміну даними $R/\overline{W} = 0$, що свідчить про те, що буде проводитися запис (передача). Дані передаються послідовно по 8 біт. Після

передачі чергового байта (адреси або даних), ведучий передавач очікує від веденого приймача біт підтвердження $\overline{ACK}$. Для задання початку і кінця сеансу обміну даними ведучий передавач формує умови «СТАРТ» і «СТОП».

2. *Ведучий приймач*: на початку сеансу обміну ведучий приймач передає на лінію SDA адресу веденого передавача (7 біт) і біт напряму обміну $R/\overline{W}=1$, що свідчить про те, що ведучий буде здійснювати прийом. Ведучий приймач формує імпульси синхронізації, які передаються лінією SCL. Після прийому адреси ведений передавач виставляє на лінію SDA сигнал підтвердження $\overline{ACK}$, а потім передає дані. Дані від веденого передавача передаються послідовно по 8 біт лінією SDA. Після прийому чергового байта ведучий приймач виставляє на лінію SDA сигнал підтвердження $\overline{ACK}$. Умови «СТАРТ» і «СТОП» формуються ведучим пристроєм для вказання початку і кінця сеансу обміну послідовними даними.

3. *Ведений приймач*: ведучий передавач видає на лінію SDA адресу веденого приймача й біт напрямку $R/\overline{W}=0$, що свідчить про те, що буде виконуватися запис (передача). На лінії SCL ведучий передавач видає синхроімпульси. Після одержання адреси ведений приймач передає сигнал підтвердження $\overline{ACK}$, після чого ведучий передавач послідовно передає дані на лінію SDA. Ведений приймач після прийому чергового байта даних передає сигнал підтвердження $\overline{ACK}$, що надходить до ведучого передавача лінією SDA. Умови «СТАРТ» і «СТОП» формуються ведучим передавачем.

4. *Ведений передавач*: перший байт (адреса) на шині SDA приймається і обробляється веденим передавачем так само, як і в режимі веденого приймача, при цьому біт напрямку $R/\overline{W}=1$, що свідчить про те, що ведучий буде здійснювати прийом. Дані послідовно передаються лінією SDA від веденого передавача тоді, як синхроімпульси передаються лінією SCL від ведучого приймача. Після передачі кожного байта ведений передавач аналізує наявність на лінії SDA біта підтвердження $\overline{ACK}$, який передає ведучий приймач. Умови «СТАРТ» і «СТОП» формує ведучий приймач.

У підпорядкованому (веденому) режимі апаратні засоби інтерфейсу I^2C здійснюють пошук своєї власної підпорядкованої адреси або адреси загального виклику. Якщо детектується одна із цих адрес, відбувається запит на переривання і виконуються відповідні дії. Якщо модуль I^2C хоче захопити шину й стати ведучим, то він чекає, поки шина звільниться (SDA = SCL = 1). Можливе функціонування в якості веденого при цьому не переривається.

Два і більше пристрої можуть спробувати стати ведучими і одночасно згенерувати умову «СТАРТ». У цьому випадку здійснюється арбітраж шини в моменти, коли шина SCL перебуває у високому стані.

Якщо один ведучий передає на лінію даних низький рівень, а інший – високий, то останній відключається від лінії, тому що стан шини SDA (низький) не відповідає високому стану внутрішньої шини даних пристрою, який бажає стати ведучим.

Якщо арбітраж шини загублено у головному (ведучому) режимі, то відповідний пристрій I^2C переключається у підпорядкований режим і може розпізнавати свою власну підпорядковану адресу. На рис. 8.4 наведені часові діаграми, які відображають обмін даними шиною I^2C.

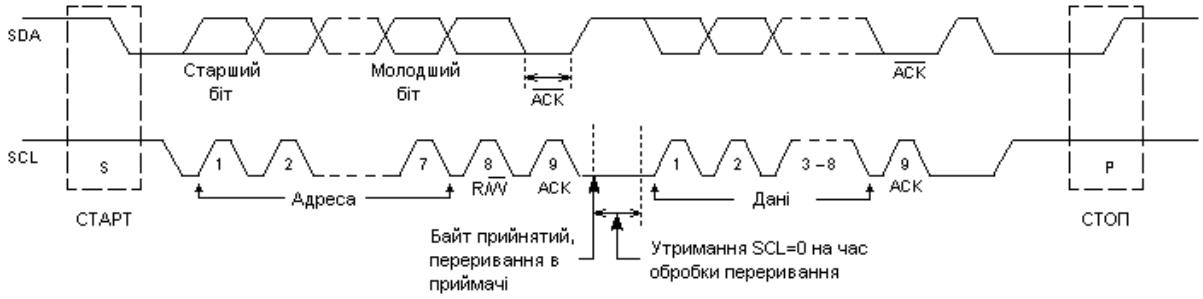


Рис. 8.4. Часові діаграми обміну даними шиною I^2C

8.1.3. Модуль I^2C мікроконтролерів AVR

8.1.3.1. Загальна характеристика модуля

Фактично всі МК AVR сімейства Mega мають модуль двопровідного синхронного послідовного інтерфейсу TWI (Two Wire (Serial) Interface). У деяких моделях функцію TWI може виконувати модуль USI [3].

Інтерфейс TWI є повним аналогом базової версії інтерфейсу I^2C фірми «Philips» (п. 8.1.2). Він дозволяє об’єднати разом до 128 різних пристроїв за допомогою двонаправленої шини, яка складається із двох ліній: лінії SCL і лінії SDA. Додатково для реалізації шини використовуються два підтягуючі резистори: *R*1, *R*2, по одному на кожну лінію (рис. 8.5).

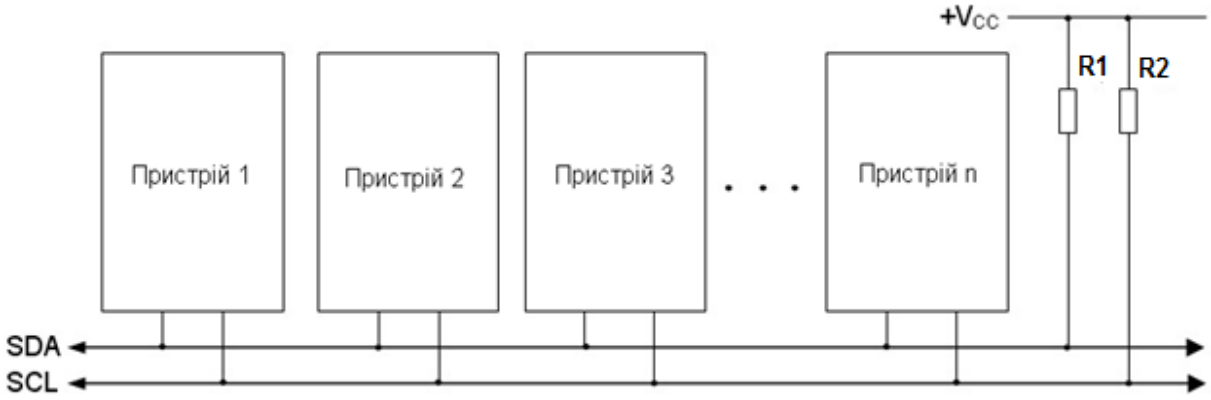


Рис. 8.5. Використання підтягуючих резисторів у мережі TWI

Шинні формувачі на вихідних лініях всіх TWI-сумісних пристроїв виконуються за схемою з відкритим стоком, що дозволяє реалізувати функцію «монтажне АБО/І» [1].

Відповідно, НИЗЬКИЙ рівень на лінії встановлюється тоді, коли один або більше пристроїв виставляють на лінію сигнал логічного нуля (функція АБО для нулів), а ВИСОКИЙ рівень на лінії встановлюється тоді, коли всі пристрої, які підключені до неї, встановлюють свої виходи в одиничний або третій стан (функція І для одиниць).

Шина TWI повністю статична і швидкість обміну може бути досить низькою. Максимально допустима кількість мікросхем, які можна під'єднати до однієї шини, обмежується максимальною ємністю шини: 400 пФ.

Швидкість передачі даних для основного режиму роботи дорівнює: 100 Кбіт/с, а для режиму із заниженою швидкістю – 10 Кбіт/с.

Протокол інтерфейсу TWI дозволяє підключати до шини кілька ведучих пристроїв (режим Multi-Master), при цьому можуть виникати різні проблеми, однією з яких є розбіжність частот тактових сигналів, які генеруються різними ведучими. Задача синхронізації вирішується завдяки приєднанню всіх пристроїв до лінії SCL за схемою «монтажне АБО/І». Всі ведучі контролюють рівень, що присутній на лінії SCL, і визначають момент початку відліку імпульсу або паузи тактового сигналу за відповідною зміною сигналу.

Іншою задачею, яку доводиться вирішувати під час підключення до шини декількох ведучих пристроїв, є задача розподілу пріоритетів, у випадку, якщо два і більше ведучих одночасно намагаються почати передачу. У разі виникнення такої ситуації передачу може здійснити тільки один ведучий, інші ж повинні переключитися в режим веденого, причому передані дані під час розподілу пріоритетів не повинні спотворюватися.

Для розв'язання описаної задачі всі ведучі пристрої після видачі даних на лінію SDA контролюють її стан. Якщо стан лінії відрізняється від того, в який її переводив ведучий, він втрачає пріоритет. Процес розподілу пріоритетів триває доти, поки на шині не залишиться тільки один ведучий.

Під час передачі шиною TWI даних, разом з ними передається певна службова інформація. Сукупність даних і відповідної службової інформації називається пакетом. Розрізняють адресні пакети і пакети даних.

8.1.3.2. Формат адресного пакета

Усі адресні пакети, які передаються шиною TWI, мають довжину 9 біт. Пакет включає 7-розрядну адресу (першим передається старший розряд), керуючий біт $R/\overline{W}$ і біт підтвердження $\overline{ACK}$ (рис. 8.6).

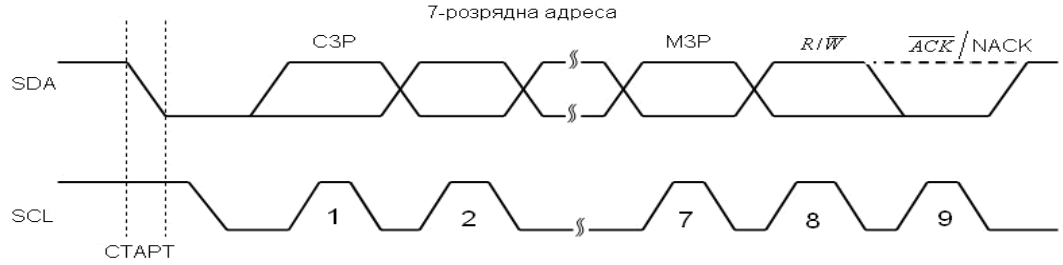


Рис. 8.6. Формат адресного пакета

Нульова адреса зарезервована для реалізації загальних викликів. Загальний виклик використовується тоді, коли ведучий хоче передати повідомлення всім веденим, які підключено до шини. Під час прийому адресного пакета з нульовою адресою і скинутим керуючим бітом (запит на передачу) всі ведені пристрої, які підключено до шини, повинні сформувати підтвердження. Винятки становлять лише ті пристрої, у яких розпізнавання загального виклику з якихось причин заборонено. Відповідно, наступні пакети даних, що посилаються ведучим, будуть отримані всіма веденими пристроями, які розпізнали адресу загального виклику.

Загальний виклик із встановленим керуючим розрядом (запит на читання) не має сенсу, оскільки різні пристрої не можуть одночасно здійснювати передачу даних шиною.

8.1.3.3. Формат пакета даних

Усі пакети даних, які передаються шиною TWI, теж мають довжину 9 біт (рис. 8.7).

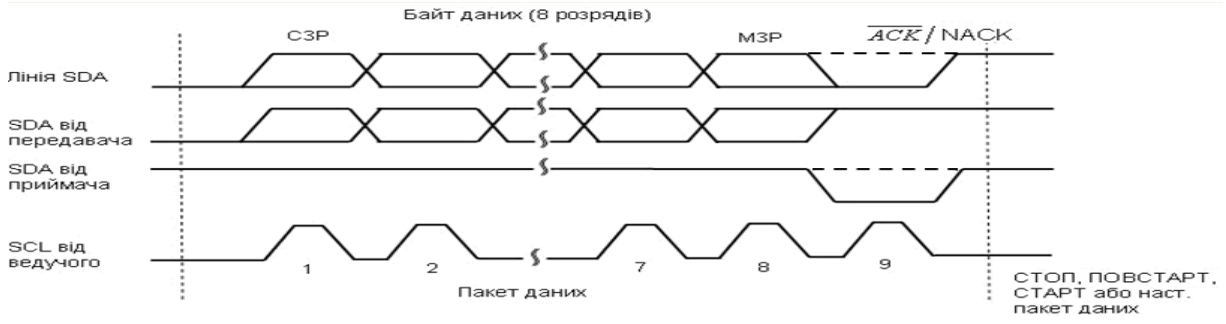


Рис. 8.7. Формат пакета даних

Пакет складається з байта даних (першим передається старший розряд) і біта підтвердження $\overline{ACK}$.

На практиці кожен цикл обміну шиною TWI складається з таких етапів (рис. 8.4):

– формування стану «СТАРТ»;

– передача адресного пакета $SLA + R/\overline{W}$;
– передача одного або декількох пакетів даних;
– формування стану «СТОП».

Швидкість обміну шиною TWI задається ведучим, оскільки саме він генерує тактові імпульси. Однак, якщо ведений не може приймати дані з такою швидкістю або йому просто потрібен час на обробку даних між прийомом пакетів, він може збільшити паузу між тактовими імпульсами, утримуючи на лінії SCL сигнал НИЗЬКОГО рівня (рис. 8.4). На тривалість тактових імпульсів це не впливає.

8.1.3.4. Опис структури модуля TWI

Загальна характеристика модуля

Структурну схему модуля TWI наведено на рис. 8.8.

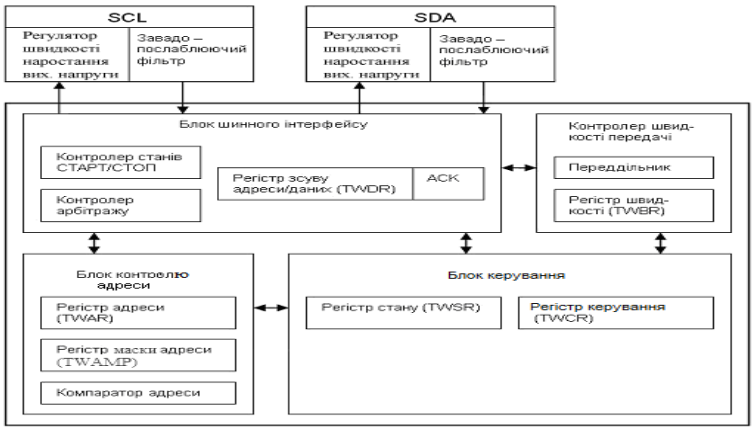


Рис. 8.8. Структурна схема модуля TWI

Для керування модулем використовуються п’ять (у нових моделях – шість) регістрів введення/виведення [3]. Назви і призначення цих регістрів, а також їх положення в адресному просторі регістрів введення/виведення деяких моделей сімейства Mega наведено в табл. 8.1.

Для підключення модуля до однойменних ліній шини використовуються виводи: SCL і SDA.

Обидва виводи є лініями портів введення/виведення МК (табл. 8.2).

Під час використання зазначених виводів модулем TWI, до них, як і у разі звичайного їхнього використання, можна підключити внутрішні підтягуючі резистори. Це дозволить у низці випадків обійтися без зовнішніх підтягуючих резисторів, що необхідно згідно зі специфікацією інтерфейсу TWI.

Контролер швидкості передачі

Синхроімпульси, які видаються ведучим МК на лінію SCL, формуються контролером швидкості передачі (Bit Rate Generator).

Таблиця 8.1. **РГ керування модулем TWI**

Регістр	**ATmega8535x**	**ATmega8x/16x/32x**	**ATmega64x/128x**	**ATmega48x/88x/168x**	**ATmega164x/324x/644x**	**ATmega640x ATmega1280x/1281x ATmega2560x/2561x**	**Призначення**
TWBR	$00 ($20)	$00 ($20)	($70)	($B8)	($B8)	($B8)	Регістр швидкості передачі
TWSR	$01 ($21)	$01 ($21)	($71)	($B9)	($B9)	($B9)	Регістр стану
TWAR	$02 ($22)	$02 ($22)	($72)	($BA)	($BA)	($BA)	Регістр адреси
TWDR	$03 ($23)	$03 ($23)	($73)	($BB)	($BB)	($BB)	Регістр даних
TWCR	$36 ($56)	$36 ($56)	($74)	($BC)	($BC)	($BC)	Регістр керування
TWAMR	–	–	–	($BD)	($BD)	($BD)	Регістр маски адреси

Таблиця 8.2. **Виводи деяких МК, що використовуються модулем TWI**

Назва	**ATmega8x**	**ATmega16x/32x**	**ATmega163x**	**ATmega48x/88x/168x**	**ATmega323x/324x ATmega464x/644x**	**ATmega64x/128x/640x ATmega1280x/1281x ATmega2560x/2561x**	**Призначення**
SDA	PC4	PC1	PC1	PC4	PC1	PD1	Лінія даних
SCL	PC5	PC0	PC0	PC5	PC0	PD0	Лінія тактового сигналу

Керування швидкістю здійснюється за допомогою регістра TWBR. Частота сформованого сигналу SCL (під час роботи пристрою в режимі ведучого) визначається виразом

$$f_{sck} = f_{clk} /(16 + 2TWBR) . \qquad (8.1)$$

У нових моделях, наприклад сімейства Mega, для керування швидкістю разом із регістром TWBR використовуються два молодших розряди (TWPS1:TWPS0) регістра TWSR. Для цих моделей частота сформованого сигналу SCL визначається виразом

$$f_{scl} = f_{clk}/(16 + 2TWBR \cdot 4^{TWPS}), \qquad (8.2)$$

де f_{scl} – частота сигналу SCL (визначає швидкість передачі); f_{clk} – тактова частота МК; TWBR – значення, яке записано в регістрі TWBR; TWPS – десятковий еквівалент розрядів TWPS1:TWPS0 регістра TWSR. Значення 4^{TWPS} обчислюється згідно з табл. 8.3.

Таблиця 8.3. **Обчислення значення 4^{TWPS}**

TWPS1	TWPS0	Значення 4^{TWPS}
0	0	1
0	1	4
1	0	16
1	1	64

Блок контролю адреси

Блок контролю адреси (Address Match Unit) перевіряє прийняту адресу на відповідність значенню, яке перебуває у старших семи розрядах регістра адреси TWAR. Він також перевіряє наявність загальних викликів, якщо дозволено їхнє розпізнавання. У разі виявлення коректної адреси інформація про це передається блоку керування. Контроль адресних пакетів здійснюється блоком навіть у разі знаходження МК у «сплячому» режимі, що дозволяє перевести МК у робочий режим у випадку адресації його ведучим пристроєм.

Формат регістра TWAR показано на рис. 8.9, а опис його розрядів наведено в табл. 8.4.

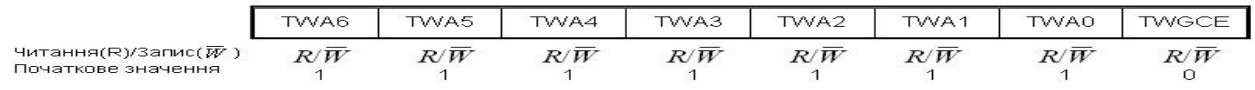


Рис. 8.9. Формат регістра TWAR

Таблиця 8.4. **Опис розрядів регістра адреси TWAR**

Розряд	Назва	Опис
7...1	TWA6...TWA0	**Адреса пристрою** У цих розрядах утримується адреса, на яку пристрій буде відзиватися під час роботи в режимі веденого. Під час роботи пристрою в режимі ведучого вміст цих розрядів не має значення
0	TWGCE	**Дозвіл розпізнавання загальних викликів** Якщо цей розряд встановлено в одиницю, пристрій буде озиватися на загальні виклики (пакети з адресою $00) так само, як і на виклики з адресою, що перебуває в розрядах TWA6...0 регістра. Якщо розряд скинуто в нуль, розпізнавання загальних викликів заборонено

Блок керування

Керування модулем TWI здійснює блок керування (Control Unit) відповідно до значень регістра керування TWCR та інформації, яка надходить до нього від інших блоків модуля. У разі настання певних подій, зазначених нижче, блок керування формує в регістрі стану TWSR код статусу, що відповідає події, і встановлює в регістрі TWCR прапорець запиту на переривання TWINT.

До моменту скидання цього прапорця на лінії SCL утримується НИЗЬКИЙ рівень, припиняючи тим самим передачу даних шиною.

Формування запиту на переривання здійснюється у разі виникнення таких подій:

- закінчення формування стану «СТАРТ/ПОВСТАРТ»;
- закінчення передачі адресного пакета $(SLA+R/\overline{W})$;
- закінчення передачі байта даних;
- втрата пристроєм пріоритету;
- адресація пристрою або наявність загального виклику;
- закінчення прийому байта даних;
- виникнення помилок на шині, обумовлених неприпустимими умовами формування станів «СТАРТ/СТОП».

Формат регістра TWCR показано на рис. 8.10, а опис його розрядів наведено в табл. 8.5.

	7	6	5	4	3	2	1	0
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
Читання(R)/Запис($\overline{W}$)	$R/\overline{W}$	$R/\overline{W}$	$R/\overline{W}$	$R/\overline{W}$	R	$R/\overline{W}$	R	$R/\overline{W}$
Початкове значення	1	1	1	1	1	1	0	0

Рис. 8.10. Формат регістра TWCR

Формат регістра TWSR показано на рис. 8.11, а опис його розрядів наведено в табл. 8.6.

	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	–	–	–	ATmega163x ATmega323x
Читання(R)/Запис($\overline{W}$)	R	R	R	R	R	R	R	R	
Початкове значення	1	1	1	1	1	0	0	0	
	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	Інші моделі
Читання(R)/Запис($\overline{W}$)	R	R	R	R	R	R	R	R	
Початкове значення	1	1	1	1	1	0	0	0	

Рис. 8.11. Формат регістра TWSR

Таблиця 8.5. **Опис розрядів регістра керування TWCR**

Розряд	Назва	Опис
7	TWINT	**Прапорець переривання від модуля TWI** Цей прапорець встановлюється апаратно після виконання чергової операції, коли модуль очікує відгук з боку програми. Якщо встановлено прапорці: I регістра SREG і TWIE регістра TWCR, генерується переривання і здійснюється виклик відповідного оброблювача. Доки прапорець TWINT встановлено, на лінії SCL утримується сигнал НИЗЬКОГО рівня. Скидання прапорця може бути здійснено *тільки записом у нього логічної одиниці*
6	TWEA	**Дозвіл біта підтвердження** Розряд TWEA керує формуванням біта підтвердження. Якщо цей розряд встановлено в одиницю, то пристрій формує сигнал підтвердження, коли це необхідно. У разі скидання розряду в нуль пристрій віртуально відключається від шини TWI, тобто біт підтвердження не формується
5	TWSTA	**Прапорець стану «СТАРТ»** Під час записування в розряд TWSTA логічної одиниці модуль перевіряє стан шини TWI та, якщо шина вільна, формує стан «СТАРТ». Якщо шина зайнята, модуль TWI очікує появи стану «СТОП», і тільки після цього формує стан «СТАРТ». Прапорець скидається апаратно по закінченні формування стану «СТАРТ»
4	TWSTO	**Прапорець стану «СТОП»** Встановлення прапорця TWSTO в одиницю в режимі ведучого приводить до формування на шині стану «СТОП». Прапорець скидається апаратно по закінченні формування стану «СТОП». Встановлення прапорця TWSTO в режимі веденого може використовуватися для виходу з помилкових ситуацій. Після запису в цей розряд одиниці модуль TWI повертається в режим неадресованого веденого, а виводи SCL і SDA для передачі переводяться у третій стан. Стан «СТОП» не формується
3	TWWC	**Прапорець конфлікту запису** Прапорець встановлюється в одиницю у разі спроби запису в регістр TWDR, коли прапорець переривання TWINT скинуто. Прапорець скидається під час записування в регістр TWDR, коли прапорець переривання TWINT встановлено
2	TWEN	**Дозвіл роботи модуля TWI** Розряд TWEN керує функціонуванням модуля TWI. Під час записування в цей розряд одиниці модуль TWI включається і бере на себе керування контактами введення/виведення МК, які відповідають виводам SCL і SDA. У разі скидання розряду TWEN в нуль модуль TWI вимикається
1	–	Зарезервовано, читається як нуль
0	TWIE	**Дозвіл переривання від модуля TWI** Якщо в цей розряд записано одиницю і прапорець I регістра SREG також встановлено в одиницю, то переривання від модуля TWI дозволено

Таблиця 8.6. **Опис розрядів регістра стану TWSR**

Розряд	Назва	Опис
7...3	TWS7...TWS3	**Стан модуля TWI** Значення, яке утримується в цих розрядах, відображає стан вузлів модуля і шини TWI. Можливі коди статусу буде описано під час подальшого розгляду модуля TWI. Розряди TWS7...3 доступні тільки для читання
2	–	Зарезервовано, читається як «0»
1, 0	TWPS1, TWPS0	**Коефіцієнт ділення попереднього дільника контролера швидкості передачі** Стан цих розрядів визначає коефіцієнт ділення попереднього дільника контролера швидкості передачі, який керує частотою сигналу SCL, що генерується (див. вище опис контролера швидкості передачі)

8.1.3.5. Взаємодія прикладної програми з модулем TWI

Взаємодія прикладної програми з модулем TWI ґрунтується на використанні переривання від модуля, яке виникає після кожної події, що відбулася на шині (прийом байта, формування станів «СТАРТ/СТОП» і т. ін.). Відповідно, під час передачі даних шиною програма може виконувати інші задачі. Якщо переривання від модуля TWI з яких-небудь причин використовувати не можна, його можна заборонити. У цьому випадку програма повинна буде постійно слідкувати за станом прапорця TWINT для реагування на події, які відбуваються на шині.

Встановлення прапорця TWINT регістра TWCR означає, що модуль TWI закінчив виконання чергової операції й очікує реакції програми, при цьому у старших п’яти розрядах регістра TWSR формується відповідне значення, яке характеризує поточний стан шини TWI. Відповідно, програма повинна проаналізувати це значення і задати подальше поводження модуля TWI, змінюючи вміст регістрів TWCR та TWDR.

На рис. 8.12 показано взаємодію прикладної програми з модулем TWI на прикладі передачі одного байта даних від ведучого до веденого.

Передача даних відбувається у такій послідовності.

1. Першою операцією під час передачі даних шиною TWI є формування стану «СТАРТ». Для цього треба записати відповідне значення в регістр TWCR. Формування стану «СТАРТ» почнеться відразу ж після скидання прапорця TWINT. Для скидання прапорця TWINT розряд TWCR, який відповідає цьому прапорцю, треба встановити в одиницю. Після успішного формування стану «СТАРТ» встановлюється прапорець TWINT. Число, яке перебуває у регістрі стану TWSR, відображає результат успішного виконання цієї операції.

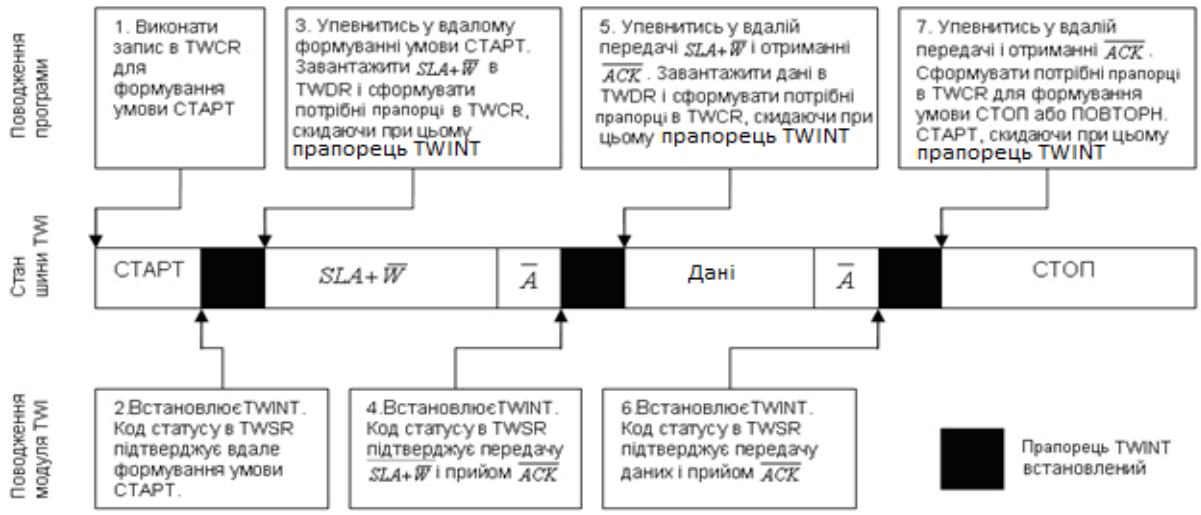


Рис. 8.12. Приклад взаємодії програми з модулем TWI

2. Далі треба перевірити вміст регістра TWSR та впевнитися в успішному формуванні стану «СТАРТ». Якщо код статусу відповідає очікуваному, завантажити в регістр TWDR вміст пакета $SLA+\overline{W}$ і сформувати в регістрі TWCR команду на передачу адресного пакета (не забуваючи скинути при цьому прапорець TWINT). Передача пакета почнеться відразу ж після скидання прапорця.

3. По закінченні передачі адресного пакета встановлюється прапорець TWINT. Код статусу, що перебуває в регістрі TWSR, свідчить про успішну передачу пакета, а також про одержання підтвердження від веденого пристрою.

4. Далі треба перевірити вміст регістра TWSR та впевнитися в успішній передачі адресного пакета та одержанні підтвердження. Якщо код статусу відповідає очікуваному, завантажити дані в регістр TWDR, а потім сформувати в регістрі TWCR команду на передачу пакета даних (не забуваючи скинути при цьому прапорець TWINT). Передача пакета даних почнеться відразу ж після скидання прапорця.

5. По закінченні передачі пакета даних встановлюється прапорець TWINT. Код статусу, що перебуває в регістрі TWSR, свідчить про успішну передачу пакета, а також про одержання підтвердження від веденого пристрою.

6. Далі треба перевірити вміст регістра TWSR та впевнитися в успішній передачі пакета даних та одержанні підтвердження. Якщо код статусу відповідає очікуваному, слід записати в регістр TWCR значення, відповідно до якого модуль TWI сформує на шині стан «СТОП» або «ПОВСТАРТ» (не забуваючи при цьому скинути прапорець TWINT). Формування стану «СТОП» або «ПОВСТАРТ» почнеться відразу ж після скидання прапорця TWINT.

Таким чином, взаємодії прикладної програми з модулем TWI складається із таких трьох частин.

1. Після завершення модулем виконання чергової операції, він встановлює у регістрі TWCR прапорець TWINT, записує у регістр TWSR код статусу й очікує

реакції програми. Доки прапорець встановлено в одиницю, на лінії SCL утримується НИЗЬКИЙ рівень.

2. Після встановлення прапорця TWINT, програма повинна занести в регістр TWDR значення, яке відповідає наступному етапу обміну.

3. Після оновлення вмісту регістра TWDR, програма повинна сформувати в регістрі TWCR команду для виконання наступного етапу обміну. У разі завантаження в регістр нового значення необхідно скинути прапорець TWINT записом у нього одиниці. Після скидання прапорця модуль почне виконання операції, обумовленої вмістом регістра TWCR.

8.1.3.6. Програмування інтерфейсу TWI

Режими роботи модуля TWI

Модуль TWI може працювати в таких режимах:

- ведучий передавач (Master Transmitter);
- ведучий приймач (Master Receiver);
- ведений передавач (Slave Transmitter);
- ведений приймач (Slave Receiver).

Вибір конкретного режиму залежить від роботи програми і, відповідно, дій, які вона виконує.

Нижче буде описано конкретні режими роботи модуля TWI з наведенням можливих значень кодів статусу.

Під час опису дій програми у відповідь на отримання відповідного коду статусу, розглядаються тільки чотири прапорці регістра TWCR із семи: TWSTA (STA), TWSTO (STO), TWINT і TWEA. Інші три прапорці мають такі значення: TWWC = x (0/1), що залежить від відсутності/наявності помилки запису під час передачі; TWEN = 1 (модуль TWI – включений); TWIE = x (0/1) – забороняє/дозволяє переривання у разі встановлення прапорця TWINT.

Є два коди стану загального призначення ($F8 і $00), які не пов’язані з певним режимом роботи [3].

Режим «Ведучий передавач»

У режимі «Ведучий передавач» (Master Transmitter) здійснюється передача даних від ведучого пристрою до веденого. Для перемикання пристрою в режим «Ведучий» необхідно записати у регістр TWCR керуюче слово, яке переводить модуль TWI у стан «СТАРТ». Формат адресного пакета, який потім передається, визначає, в якому з режимів буде працювати ведучий. Під час передачі пакета $SLA+\overline{W}$ модуль переходить у режим «Ведучий передавач», а під час передачі пакета $SLA+R$ – у режим «Ведучий приймач».

Керуючі слова, які треба записувати в регістр TWCR для програмування цього режиму, коди статусу, схему алгоритму роботи модуля в цьому режимі та його програмну реалізацію наведено у [3].

На рис. 8.13 наведено часові діаграми роботи та різні стани модуля TWI у режимі «Ведучий передавач».

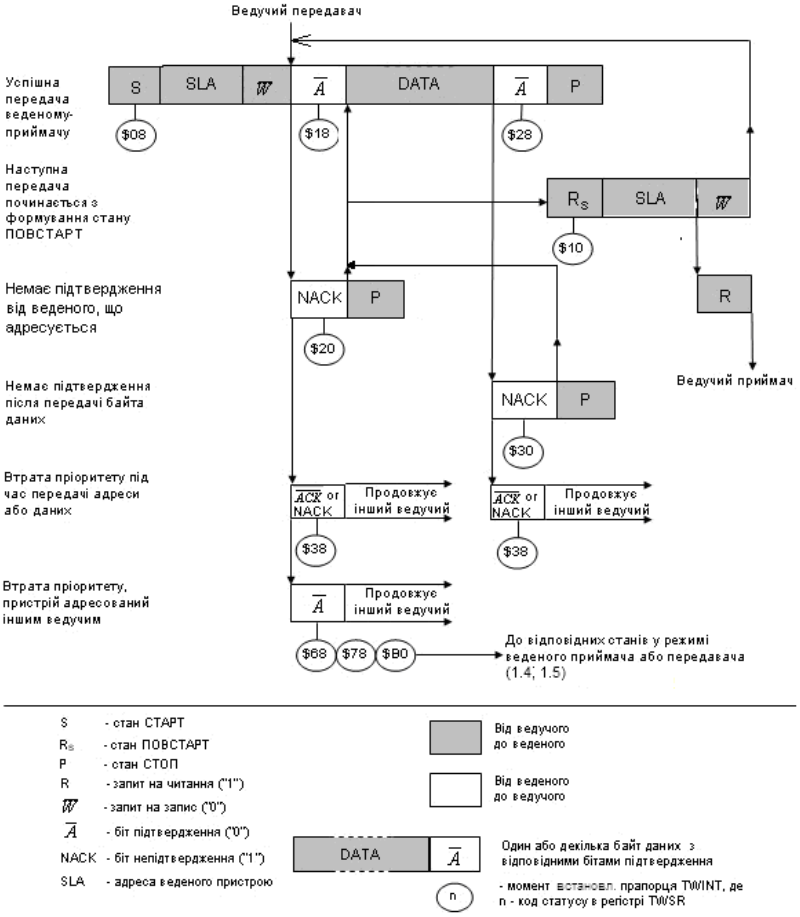


Рис. 8.13. Часові діаграми роботи та стани модуля TWI в режимі «Ведучий передавач»

Режим «Ведучий приймач»

У режимі «Ведучий приймач» ведучий здійснює прийом даних від веденого пристрою. Для переключення пристрою в режим «Ведучий» необхідно записати в регістр команд команду для формування сигналу «СТАРТ».

Формат адресного пакета, який передається потім, визначає, в якому з режимів буде працювати ведучий. Під час передачі пакета $SLA+\overline{W}$ модуль переходить у режим «Ведучий передавач», а під час передачі пакета $SLA+R$ – переходить у режим «Ведучий приймач».

Керуючі слова, які треба записувати в регістр TWCR для програмування цього режиму, коди статусу, схему алгоритму роботи модуля в цьому режимі та його програмну реалізацію наведено в [3].

На рис. 8.14 наведено часові діаграми роботи та різні стани модуля TWI у режимі «Ведучий приймач».

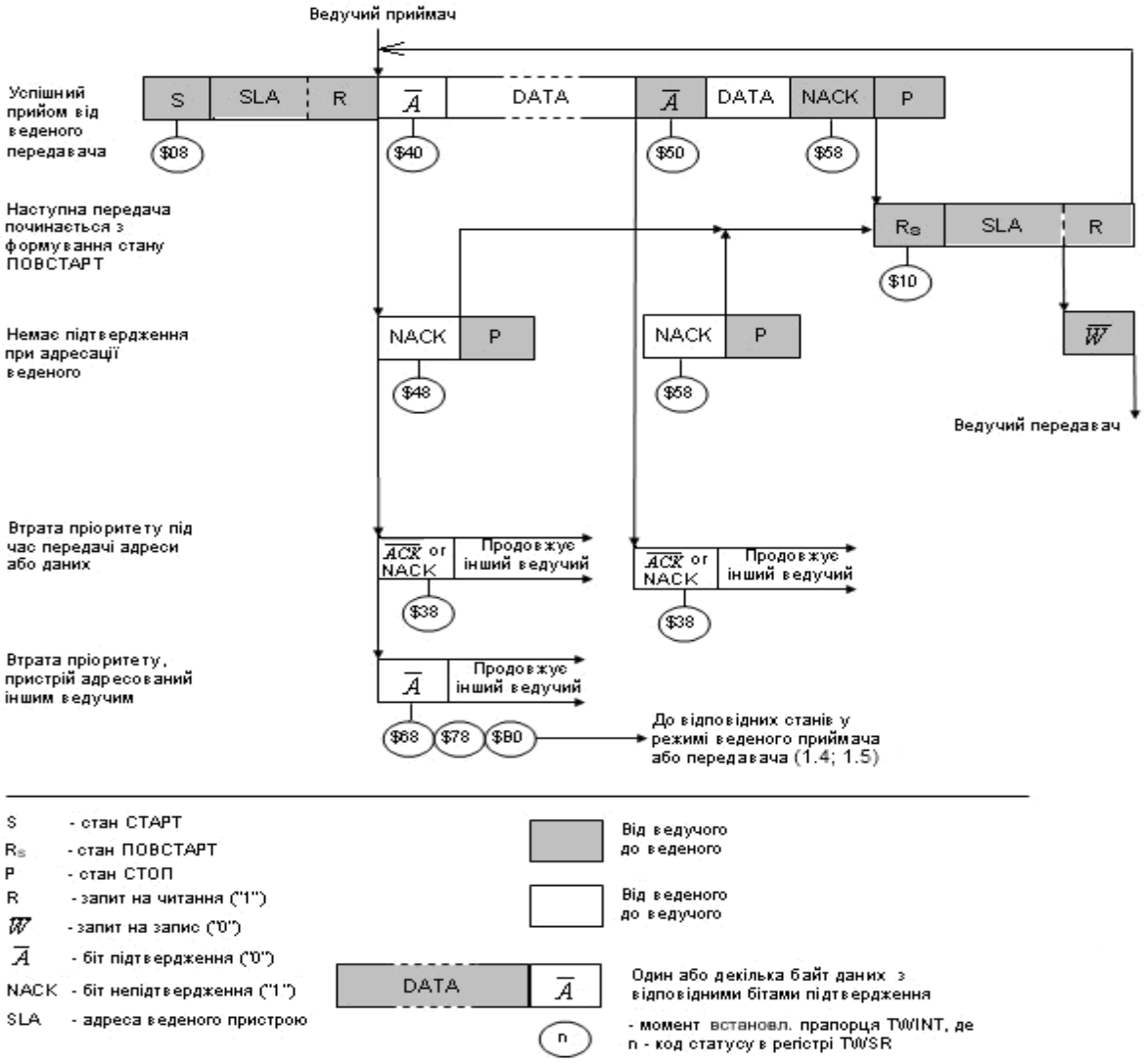


Рис. 8.14. Часові діаграми роботи та стани модуля TWI у режимі «Ведучий приймач»

Режим «Ведений приймач»

У режимі «Ведений приймач» ведений пристрій здійснює прийом даних від ведучого. Перед тим як переключити модуль у режим «Ведений приймач», слід внести у старші розряди регістра TWAR адресу пристрою і відповідно до логіки роботи програми встановити або скинути молодший розряд цього регістра (TWGCE).

Керуючі слова, які треба записувати в регістр TWCR для програмування цього режиму, коди статусу, схему алгоритму роботи модуля в цьому режимі та його програмну реалізацію наведено в [3].

На рис. 8.15 наведено часові діаграми роботи та різні стани модуля TWI у режимі «Ведений приймач».

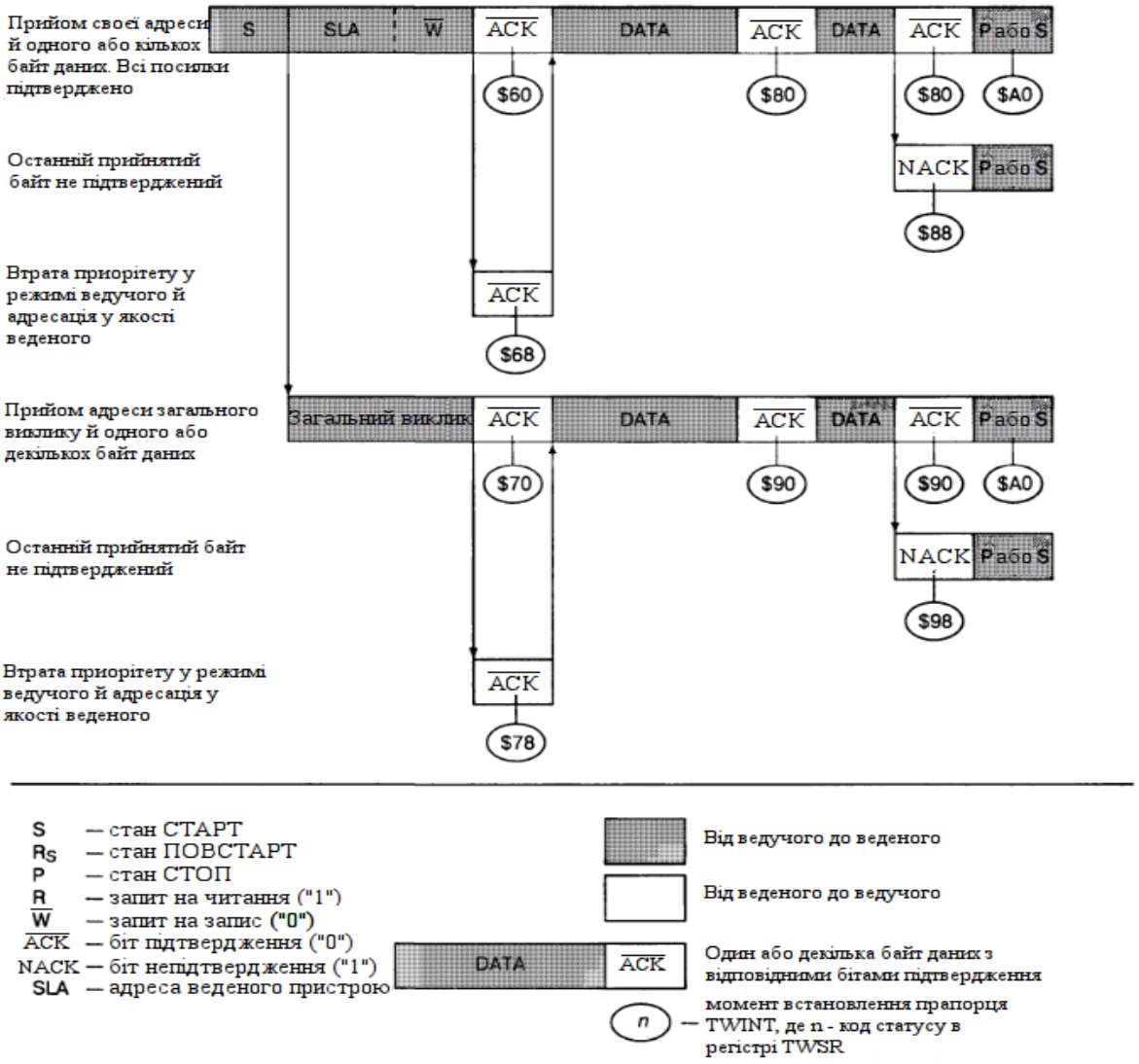


Рис. 8.15. Часові діаграми роботи та стани модуля TWI в режимі «Ведений приймач»

Режим «Ведений передавач»

У режимі «Ведений передавач» ведений пристрій здійснює передачу даних ведучому, який у цьому випадку є приймачем. Перед тим як переключити модуль у цей режим, слід занести у старші розряди регістра TWAR адресу пристрою і відповідно до логіки роботи програми встановити або скинути молодший розряд регістра (TWGCE).

Керуючі слова, які треба записувати в регістр TWCR для програмування цього режиму, коди статусу, схему алгоритму роботи модуля в цьому режимі та його програмну реалізацію наведено в [3].

На рис. 8.16 наведено часові діаграми роботи та різні стани модуля TWI у режимі «Ведений передавач».

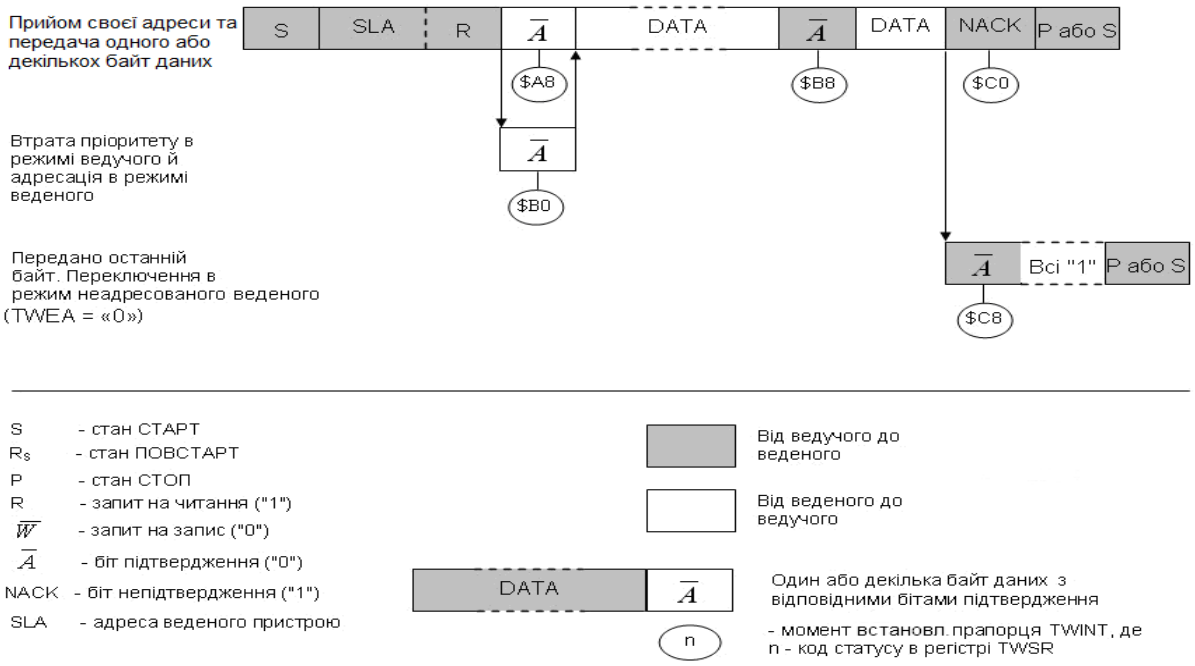


Рис. 8.16. Часові діаграми роботи та стани модуля TWI в режимі «Ведений передавач»

Комбінування різних режимів

На практиці під час виконання чергової операції шиною TWI пристрою доводиться використовувати кілька режимів, переключаючись між ними за необхідності. Як приклад нижче розглянуто операцію читання даних із зовнішньої пам’яті даних типу EEPROM (рис. 8.17).

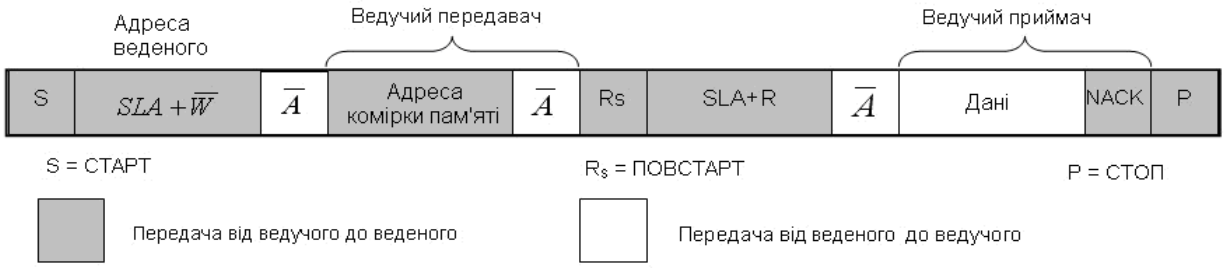


Рис. 8.17. Приклад звернення до зовнішньої пам’яті даних типу EEPROM

Під час виконання операції відбувається передача інформації як від ведучого до веденого, так і навпаки.

Після ініціювання обміну шиною ведучий повинен перебувати в режимі «Ведучий передавач», щоб повідомити веденому адресу комірки пам’яті, за якою він має намір здійснити читання. Для виконання наступного етапу операції (читання даних) ведучий повинен переключитися в режим «Ведучий приймач», при цьому він повинен зберігати контроль над шиною під час виконання всіх етапів операції. Для цього використовується стан «ПОВСТАРТ». Ведучий

формує цей стан між передачею адреси і прийомом даних, як показано на рис. 8.17.

8.1.3.7. Арбітраж

У випадку присутності на шині декількох ведучих можлива ситуація, за якої декілька ведучих одночасно почнуть процес обміну. Для таких випадків специфікацією TWI передбачено процес розподілу пріоритетів (арбітраж), у результаті виконання якого на шині залишається тільки один ведучий пристрій. Принципи цього процесу було розглянуто в п. 8.1.2, однак його виконання може розвиватися за різними сценаріями, які описано у [3].

Перелік деяких AVR-мікроконтролерів та пристроїв, які підтримують інтерфейс TWI, наведено у [3].

8.1.4. Моделювання модуля TWI (I^2C)

Моделювання модуля TWI (I^2C) МК ATmega16 у пакеті PROTEUS 8.6 описано в [3], де наведено схему алгоритму роботи моделі та робочу програми мовою С. Отримані результати моделювання підтвердили працездатність цього алгоритму та робочої програми.

Контрольні запитання та завдання

1. Якою фірмою було розроблено інтерфейс I^2C та для чого він призначений?
2. Чим відрізняється інтерфейс I^2C від інтерфейсу TWI?
3. З яких двох ліній складається шина інтерфейсу I^2C (TWI) та як вони мають бути підключені в мережі?
4. Поясніть призначення функції «монтажне АБО/І» та як вона використовується в I^2C (TWI)-мережі.
5. Наведіть та поясніть структурну схему типової мережі, яка використовує для обміну даними шину I^2C (TWI).
6. Опишіть роботу пристроїв, які використовують для обміну інтерфейс I^2C (TWI).
7. Опишіть чотири можливі режими обміну даними інтерфейсом I^2C (TWI).
8. Поясніть призначення станів «СТАРТ» і «СТОП» шини TWI. Яким чином вони формуються?
9. У чому полягає задача розподілу пріоритетів під час підключення до шини TWI декількох ведучих пристроїв? Як вона вирішується?
10. Опишіть формат адресного пакету та пакету даних.
11. Наведіть та поясніть структурну схему типового модуля TWI.

12. Що задає контролер швидкості передачі?
13. Які два вузли містять блок шинного інтерфейсу? Які їх функції?
14. Що перевіряє блок контролю адреси?
15. У разі виникнення яких подій блок керування здійснює формування запиту на переривання?
16. Назвіть та опишіть використання регістрів керування TWI-модулем.
17. На чому ґрунтується взаємодія прикладної програми з модулем TWI?
18. З яких трьох частин складається будь-який етап взаємодії прикладної програми з модулем TWI? Поясніть цю взаємодію.
19. Назвіть деякі пристрої, які підтримують інтерфейс TWI.
20. В яких режимах може працювати модуль TWI, реалізований у МК сімейства Mega?
21. Опишіть роботу модуля TWI у режимі «Ведучий передавач».
22. Опишіть роботу модуля TWI у режимі «Ведучий приймач».
23. Опишіть роботу модуля TWI у режимі «Ведений приймач».
24. Опишіть роботу модуля TWI у режимі«Ведений передавач».
25. Які можливості надає використання стану «ПОВСТАРТ»?
26. За якими сценаріями може розвиватися процес арбітражу?
27. З якою швидкістю може виконуватися обмін інформацією в I^2C (TWI) мережі?
28. На яку відстань може передаватися інформація в I^2C (TWI) мережі?
29. Скільки ведучих пристроїв може одночасно бути в I^2C (TWI) мережі?
30. Опишіть схему моделювання інтерфейсу TWI у пакеті Proteus 8.6.
31. Опишіть схему алгоритму роботи моделі.
32. Опишіть схему алгоритму запису в EEPROM-пам’ять.
33. Опишіть схему алгоритму читання EEPROM-пам’яті.
34. Поясніть окремі етапи обміну шиною TWI (I^2C), які отримано у вікні I^2C Debug після запуску моделювання.
35. Поясніть часові діаграми обміну між МК та EEPROM-пам’яттю, отриманих на екрані осцилографа після запуску моделювання.

8.2. Мережа на основі інтерфейсу SPI

8.2.1. Загальна характеристика інтерфейсу

У МПС крім задачі передачі неперервного потоку інформації, достатньо часто необхідно передавати окремі цифрові пакети чи команди керування. Для передачі такого виду інформації призначено синхронний послідовний периферійний інтерфейс (англ. Serial Peripheral Interface – SPI) [3].

Шину інтерфейсу SPI було розроблено компанією Motorola для МК серії 68000. Завдяки простоті та популярності цієї шини, багато інших виробників вже багато років використовують цей стандарт.

Усі МК сімейства Mega та XMega мають інтерфейс SPI.

За його допомогою може здійснюватися обмін даними на відстані до трьох метрів зі швидкістю до 2 Мбіт/с між МК або МК і різними периферійними пристроями, такими, як цифрові потенціометри, ЦАП/АЦП, FLASH-ПЗП і т. ін.

Крім того, через інтерфейс SPI може бути здійснено програмування мікроконтролера (режим послідовного програмування).

У цьому розділі передусім буде розглянуто використання інтерфейсу SPI в якості досить високошвидкісного каналу зв’язку.

Під час обміну даними інтерфейсом SPI в мікроконтролерній мережі AVR-мікроконтролер може працювати як ведучий (режим «Master»), або як ведений (режим «Slave»), при цьому користувач може задавати швидкість передачі (сім програмованих значень) і формат передачі (від молодшого розряду до старшого або навпаки).

Додатково під час надходження даних за допомогою інтерфейсу SPI можна виводити МК з режиму зниженого енергоспоживання Idle.

8.2.2. Характеристика модуля SPI мікроконтролерів AVR

8.2.2.1. Опис структурної схеми модуля

Структурну схему модуля SPI наведено на рис. 8.18.

Модуль використовує чотири виводи МК, які є лініями портів введення/виведення загального призначення. Як приклад, у табл. 8.7 наведено виводи, які використовуються модулем SPI деяких МК сімейства Mega.

Режим роботи зазначених виводів (напрям передачі даних) при включеному модулі SPI перевизначається згідно з табл. 8.8.

Напрямок обміну даними: передача (вихід) або прийом (вхід) визначається або програмуванням відповідних розрядів регістра DDRB, або задається схемою керування модулем залежно від режиму роботи «Master»/«Slave», при цьому зберігається можливість керування внутрішніми підтягуючими резисторами виводів, що працюють як входи.

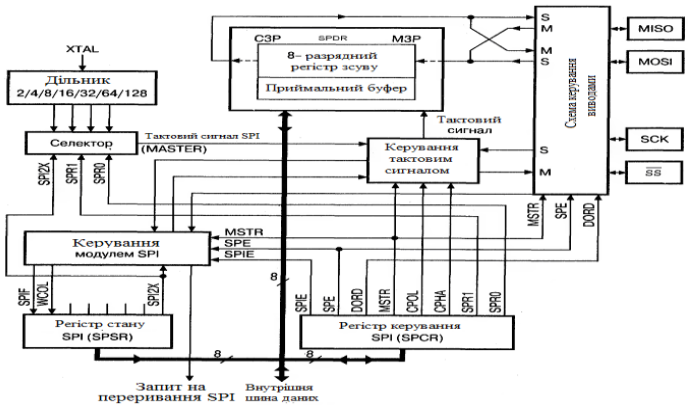


Рис. 8.18. Структурна схема модуля SPI

Таблиця 8.7. **Виводи, що використовуються модулем SPI**

Вивід	**ATmega8x /48x/88x/168x**	**ATmega8515x/8535x**	**ATmega16x/32x**	**ATmega161x/323x**	**ATmega162x/163x**	**ATmega164x/324x/644x**	**Atmega165x/64x/128x**	**ATmega325x/3250x/645x/6450x**	**ATmega640x/1280x/1281x/2560x/2561x**	**Призначення**
SCK	PB5	PB7	PB7	PB7	PB7	PB7	PB1	PB1	PB1	Вихід (master)/вхід (slave) тактовий сигнал
MISO	PB4	PB6	PB6	PB6	PB6	PB6	PB3	PB3	PB3	Вхід (master)/вихід (slave) даних
MOSI	PB3	PB5	PB5	PB5	PB5	PB5	PB2	PB2	PB2	Вихід (master)/вхід (slave) даних
$\overline{SS}$	PB2	PB4	PB4	PB4	PB4	PB4	PB0	PB0	PB0	Вибір веденого пристрою

Таблиця 8.8. **Перепризначення режиму роботи виводів модуля SPI**

Вивід	**Режим «Master»**	**Режим «Slave»**
MOSI	Визначається користувачем як вихід	Вхід
MISO	Вхід	Визначається користувачем як вихід
SCK	Визначається користувачем як вихід	Вхід
$\overline{SS}$	Визначається користувачем як вхід або вихід	Вхід

Вивід SCK МК у режимі Master працює як вихід, тому відповідна лінія порту введення/виведення загального призначення має бути запрограмована на виведення.

8.2.2.2. Програмування модуля

Для програмування модуля SPI призначено регістр керування SPCR, який у моделях ATmega8515x/8535x/162x/8x/16x/32x/64x/128x розміщено за адресою $0D ($2D), а в інших моделях – за адресою $2C ($4C). Формат цього регістра наведено на рис. 8.19, а опис його розрядів – в табл. 8.9.

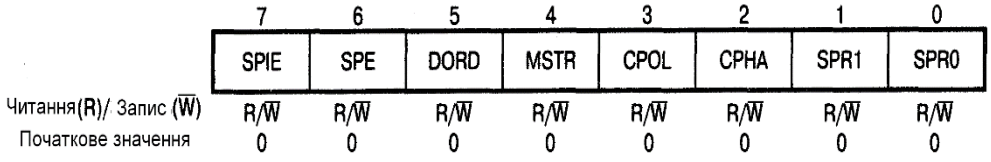


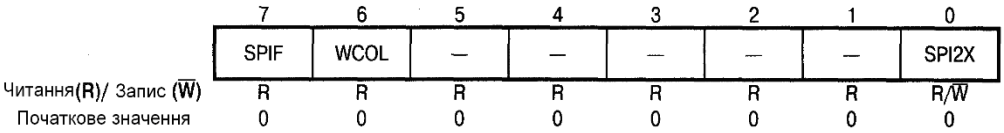
Рис. 8.19. Формат регістра SPCR

Таблиця 8.9. **Призначення розрядів регістра SPCR**

Розряд	Назва	Опис
7	SPIE	Дозвіл переривання від SPI, якщо SPIE = 1
6	SPE	Ввімкнення/вимкнення модуля SPI: SPE = 1 – ввімкнений; SPE = 0 – вимкнений
5	DORD	Порядок передачі даних: DORD = 1 – першим передається МЗР, DORD = 0 – першим передається СЗР
4	MSTR	Вибір режиму роботи («Master»/«Slave»): MSTR = 1 – «Master», MSTR = 0 – «Slave»
3	CPOL	Визначає полярність тактового сигналу (табл. 8.11)
2	CPHA	Фаза тактового сигналу – визначає момент зчитування сигналу, табл. 8.11
1, 0	SPR1:SPR0	Програмування швидкості передачі (табл. 8.12)

Контроль стану модуля, а також додаткове керування швидкістю обміну здійснюється за допомогою регістра стану SPSR, який розміщено за адресою $0E ($2E) в моделях ATmega8515x/8535x/162x/8x/16x/32x/64x/128x, та за адресою $2D ($4D) – в інших моделях. Розряди з сьомого по перший цього регістра доступні тільки для читання, а нульовий розряд – як для читання, так і для запису. Формат цього регістра наведено на рис. 8.20, а опис його розрядів – у табл. 8.10.

.



	7	6	5	4	3	2	1	0
	SPIF	WCOL	–	–	–	–	–	SPI2X
Читання(R)/ Запис ($\overline{W}$)	R	R	R	R	R	R	R	R/$\overline{W}$
Початкове значення	0	0	0	0	0	0	0	0

Рис. 8.20. Формат регістра SPSR

Таблиця 8.10. **Опис розрядів регістра SPSR**

Розряд	Назва	Опис
7	SPIF	**Прапорець «Кінець передачі»** Даний прапорець встановлюється в одиницю по закінченні передачі/прийому чергового байта. Якщо прапорець SPIE регістра SPCR також встановлено в одиницю, то генерується переривання від модуля SPI. Прапорець SPIF також встановлюється в одиницю під час переведення МК з режиму «Master» у режим «Slave» за допомогою виводу $\overline{SS}$, див. пп. 8.2.2.7. Прапорець скидається апаратно під час старту підпрограми обробки переривання, або після читання регістра стану SPI з наступним зверненням до регістра даних SPDR
6	WCOL	**Прапорець конфлікту запису** Цей прапорець встановлюється в одиницю при спробі запису у регістр даних SPDR під час передачі чергового байта. Прапорець скидається апаратно після читання регістра стану SPI з наступним зверненням до регістра даних SPDR
5...1	–	Зарезервовано, читаються як «0»
0	SPI2X	**Подвоєння швидкості обміну** У разі встановлення цього розряду в одиницю та роботі МК у режимі «Master» подвоюється частота сигналу SCK

Дані, що передаються, записуються в регістр даних SPDR, а дані, які приймаються, зчитуються з цього регістра. Регістр розміщено за адресою $0F ($2F) у моделях ATmega8515x/8535x/162x/8x/16x/32x/64x/128x, а в інших моделях – за адресою $2E ($4E). Запис у цей регістр ініціює початок передачі, а під час його читання зчитується вміст приймаючого буфера регістра зсуву, тобто регістр даних під час читання служить буфером між регістровим файлом МК і регістром зсуву модуля SPI. Під час передачі додаткового буфера немає, а функцію регістра даних виконує регістр зсуву.

8.2.2.3. Обмін даними між двома мікроконтролерами

З’єднання двох МК (ведучий–ведений) інтерфейсом SPI показано на рис. 8.21. Вивід SCK ведучого МК є виходом тактового сигналу, а веденого МК – входом.

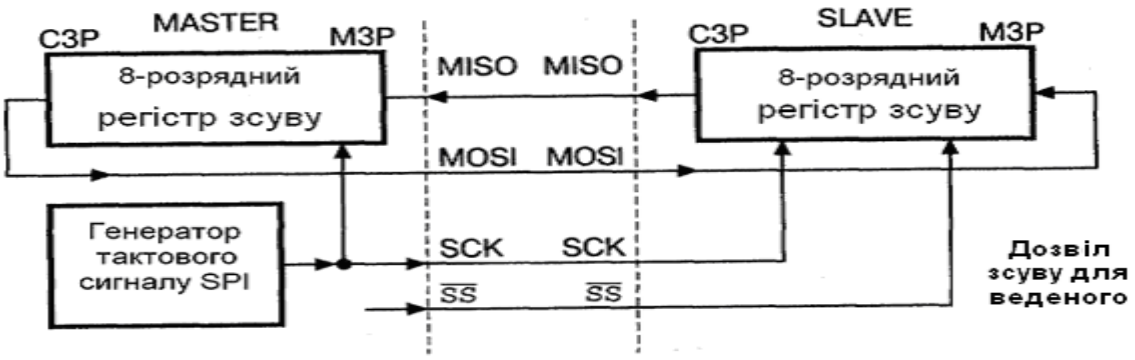


Рис. 8.21. З’єднання МК інтерфейсом SPI

Перед виконанням обміну необхідно дозволити роботу модуля SPI. Для цього потрібно встановити в одиницю розряд SPE регістра SPCR. Режим роботи визначається станом розряду MSTR цього регістра: якщо розряд встановлено в одиницю, МК працює в режимі «Master», якщо скинуто в нуль – у режимі «Slave».

Передача даних здійснюється таким чином. У разі запису в регістр даних SPI ведучого МК запускається генератор тактового сигналу модуля SPI. Після цього дані починають порозрядно видаватися на вивід MOSI ведучого і, відповідно, надходити на вивід MOSI веденого МК. Порядок передачі розрядів даних визначається станом розряду DORD регістра SPCR. Якщо розряд встановлено в одиницю, першим передається молодший розряд байта, якщо скинуто в нуль – старший розряд. Після видачі останнього розряду поточного байта генератор тактового сигналу зупиняється з одночасним встановленням в одиницю прапорця «Кінець передачі» – SPIF. Якщо переривання від модуля SPI дозволено (прапорець SPIE регістра SPCR встановлено в одиницю), генерується запит на переривання.

Можлива наступна передача залежить від режиму роботи (пп. 8.2.2.5).

Одночасно з передачею даних від ведучого до веденого відбувається передача у зворотному напрямку за умови, що на вході $\overline{SS}$ веденого присутня напруга низького рівня. Таким чином, у кожному циклі зсуву відбувається обмін даними між двома пристроями. Наприкінці кожного циклу прапорець SPIF встановлюється в одиницю як у ведучому МК, так і у веденому. Прийняті байти зберігаються в приймальних буферах для подальшого використання.

У модулі реалізовано одинарну буферизацію під час передачі і подвійну під час прийому. Це означає, що готовий для передачі байт даних не може бути записано у регістр даних SPI до закінчення попереднього циклу обміну.

При спробі змінити вміст регістра даних під час передачі встановлюється в одиницю прапорець WCOL регістра SPSR. Цей прапорець скидається після читання регістра SPSR з наступним зверненням до регістра даних SPDR.

Під час прийому даних прийнятий байт треба прочитати з регістра даних до того, як у регістр зсуву надійде останній розряд наступного байта. В іншому випадку попередній байт буде загублено.

8.2.2.4. Структура SPI-мережі мікроконтролерів

Структуру SPI-мережі МК наведено на рис. 8.22.

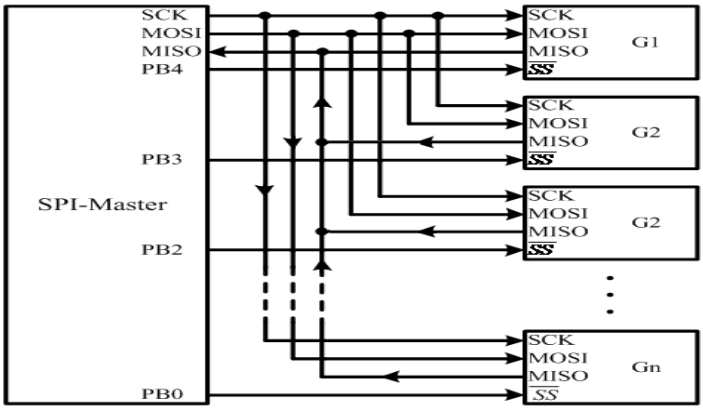


Рис. 8.22. Структура SPI-мережі МК

До інтерфейсу ведучого SPI можна одночасно підключити декілька ведених периферійних пристроїв, але активним буде тільки той, у якого на вхід $\overline{SS}$ через відповідну лінію порту ведучого пристрою буде подано рівень логічного нуля. Виводи MISO не задіяних ведених блоків знаходяться у високоомному стані і не впливають на процес передачі даних.

На рис. 8.22 блокам G1, G2…Gn відповідають ведені інтерфейси SPI, в яких дані можуть передаватися в обох напрямках. Один з ведених пристроїв з точки зору ведучого пристрою, може бути, наприклад, тільки блоком передачі.

В якості такого веденого можна навести, наприклад, АЦП з інтерфейсом SPI. Іншим веденим може бути, наприклад, ЦАП, який буде використовуватись як блок прийому.

8.2.2.5. Режими передачі даних SPI-інтерфейсом

Є чотири режими передачі даних SPI-інтерфейсом: 0…3 (рис. 8.23, 8.24).

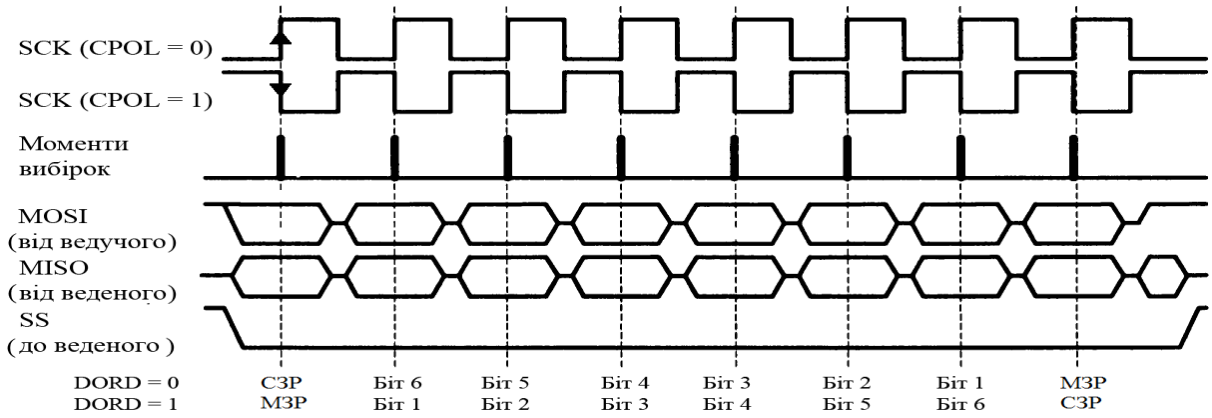


Рис. 8.23. Передача даних за CPHA = «0» (режими 0, 1)

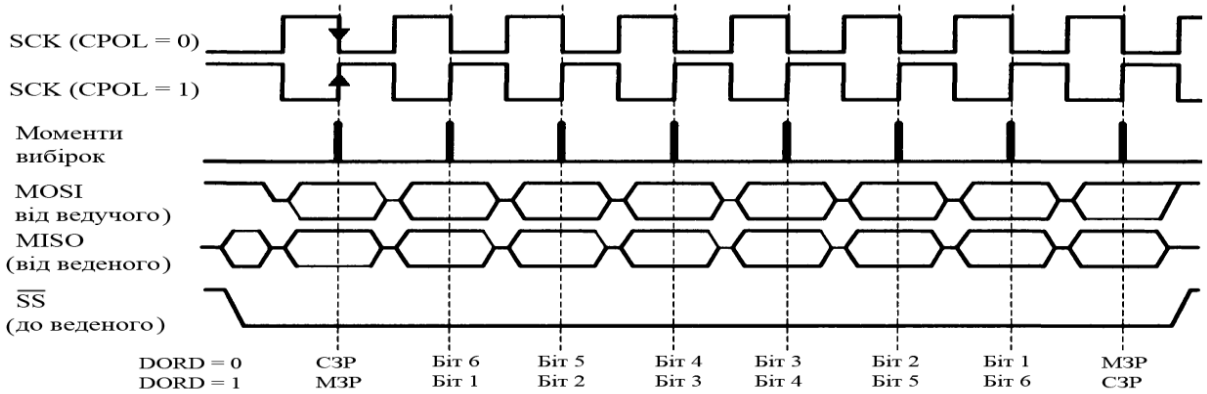


Рис. 8.24. Передача даних за CPHA = «1» (режими 2, 3)

Ці режими розрізняються відповідністю між фазою тактового сигналу SCK (визначає момент зчитування сигналу), його полярністю та даними, які передаються. Є чотири комбінації розрядів CPHA і CPOL регістра SPCR (табл. 8.11), які визначають відповідний режим роботи модуля.

В режимах 0, 1 (CPHA = 0) початок обміну визначається встановленням сигналу на виводі $\overline{SS}$ веденого в активний стан – логічний нуль, при цьому ведений виставляє на лінію MISO старший розряд даних, якщо першим передається старший значущий розряд, або молодший – якщо першим передається молодший значущий розряд (рис. 8.21).

Ведучий видає на лінію MOSI старший значущий розряд/молодший значущий розряд на $0{,}5 \cdot T_{SCK}$ раніше, ніж на лінії SCK з'являється перший імпульс, T_{SCK} – період синхроімпульсів.

Перед входами 8-розрядних регістрів зсуву ведучого та веденого МК (рис. 8.21) знаходяться два синхронних тригери (буфери), які на рисунку не показано. В ці буфери першим перепадом сигналу на лінії SCK записується значення сигналу, яке присутнє на їх інформаційному вході. Цей момент на рис. 8.23 відмічено на лінії «Моменти вибірок».

Таблиця 8.11. **Задання режиму передачі даних**

Розряд	Опис
CPOL	**Полярність тактового сигналу** «0» – генеруються імпульси додатної полярності, у разі відсутності імпульсів на виводі присутній низький рівень; «1» – генеруються імпульси від'ємної полярності, у разі відсутності імпульсів на виводі присутній високий рівень (рис. 8.23, 8.24)
CPHA	**Фаза тактового сигналу** «0» – обробка даних виконується за переднім фронтом імпульсів сигналу SCK (для CPOL = «0» – за наростаючим фронтом, а для CPOL = «1» – за спадаючим фронтом) (рис. 8.23); «1» – обробка даних виконується за заднім фронтом імпульсів сигналу SCK (для CPOL = «0» – за спадаючим фронтом, а для CPOL = «1» – за наростаючим фронтом) (рис. 8.24)

Другим перепадом імпульсів на лінії SCK відбувається зсув інформації вліво відповідно до рис. 8.21, при цьому стан буферів переписується в молодші розряди регістрів зсуву, а черговий вихідний біт з регістрів зсуву виставляється на лінії MOSI/MISO та з затримкою записується у буфери.

Вказаний зсув у часі між моментом видачі чергового розряду в лінію зв'язку між МК і моментом фіксації цього біта в буфері дозволяє компенсувати часові затримки під час передачі сигналів між МК.

Далі аналогічно здійснюється обмін між ведучим та веденим всіма наступними бітами. З кожним непарним перепадом сигналу SCK відбувається фіксація чергового біта в буфері, а з кожним парним – зсув інформації вліво.

У режимах 0, 1 після обміну байтом треба подати на вхід $\overline{SS}$ веденого МК напругу високого рівня та перевести останній у стан очікування. Після цього, щоб почати передавати наступний байт, треба подати на вхід $\overline{SS}$ веденого МК напругу низького рівня (рис. 8.25*а*).

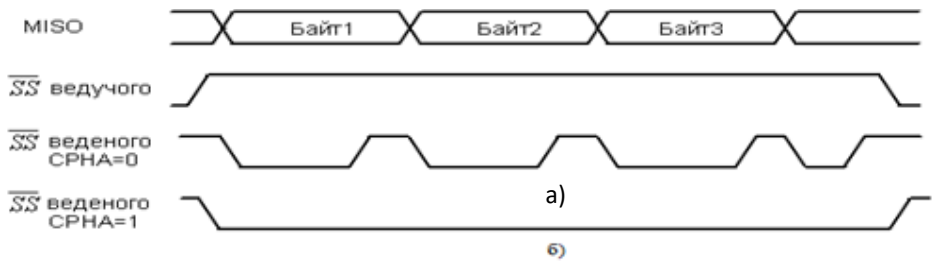


Рис. 8.25. Вплив сигналу $\overline{SS}$ для веденого на початок обміну даними:
а – режими 0, 1; *б* – режими 2, 3

У режимах 2, 3 сигнал вибору веденого може залишатися в активному стані – логічний нуль протягом передачі декількох байт інформації (рис. 8.25*б*). Це трохи спрощує логіку програмування модуля SPI. Під час передачі даних від ведучого до веденого та у зворотному напрямку всі непарні перепади SCK викликають видачу чергового біта посилки з регістра зсуву передавача на лінію. Кожний парний перепад використовується для запису цього біта в буфер перед регістром зсуву.

Таким чином, під час передачі біти даних висуваються з виходів регістрів зсуву від'ємними або додатними фронтами сигналу SCK, а фіксуються в буферах на входах регістрів додатними або від'ємними фронтами синхросигналу SCK, які зсунуто відносно моментів зміни бітів на виході регістрів зсуву на половину періоду (рис. 8.23, 8.24). Це гарантує достатній час на встановлення даних на входах відповідних регістрів після висування.

8.2.2.6. Програмування швидкості передачі даних

Частота тактового сигналу SCK і, відповідно, швидкість передачі даних інтерфейсом визначаються станом розрядів SPR1:SPR0 регістра SPCR і розряду SPI2X регістра SPSR МК, який працює в режимі «Master», тому що саме він є джерелом тактового сигналу (табл. 8.12).

Для пристрою, який перебуває в режимі «Slave», стан цих розрядів байдужий. Нормальне функціонування МК у режимі «Slave» гарантується на частотах, менших або рівних $f_{CLK}/4$ [3; 4].

Таблиця 8.12. **Задання частоти тактового сигналу SCK**

SPI2X	SPR1	SPR0	Частота сигналу SCK
0	0	0	$f_{CLK}/4$
0	0	1	$f_{CLK}/16$
0	1	0	$f_{CLK}/64$
0	1	1	$f_{CLK}/128$
1	0	0	$f_{CLK}/2$
1	0	1	$f_{CLK}/8$
1	1	0	$f_{CLK}/32$
1	1	1	$f_{CLK}/64$

Примітка. f_{CLK} – тактова частота МК.

8.2.2.7. Використання виводу $\overline{SS}$

Цей вивід призначено для вибору активного веденого пристрою та у режимі «Slave» завжди є входом (рис. 8.25). У разі подачі на нього напруги

низького рівня модуль SPI активується і вивід MISO перемикається в режим виведення даних (якщо це задано користувачем, див. вище табл. 8.8). Інші виводи модуля SPI в цьому режимі є входами.

Коли на вивід $\overline{SS}$ веденого подається напруга високого рівня, відбувається скидання модуля SPI. Відповідно, якщо зміна стану цього виводу відбудеться під час обміну даними, прийом і передача негайно припиняться, а переданий і прийнятий байти будуть загублено.

Якщо МК перебуває в режимі «Master» (розряд MSTR регістра SPCR встановлено в одиницю), напрямок передачі даних через вивід $\overline{SS}$ визначається користувачем. Якщо вивід сконфігуровано як вихід, він використовується для керування виводом $\overline{SS}$ МК, що працює в режимі «Slave».

Якщо в режимі «Master» вивід сконфігуровано як вхід, то для забезпечення нормальної роботи модуля SPI на нього треба подати напругу високого рівня. Подача на цей вхід напруги низького рівня від якої-небудь зовнішньої схеми буде сприйнято модулем SPI як вибір цього МК в якості веденого, і відповідно, початок передачі йому даних.

Щоб уникнути конфлікту на шині модуль SPI у таких випадках виконує такі дії:

1. Прапорець MSTR регістра SPCR скидається, і МК перемикається в режим «Slave». Як наслідок, виводи MOSI і SCK починають функціонувати як входи (табл. 8.8).
2. Встановлюється прапорець SPIF регістра SPSR, генеруючи запит на переривання від SPI. Якщо переривання від SPI дозволено і прапорець I регістра SREG встановлено в одиницю, відбувається запуск підпрограми обробки переривання.

Таким чином, якщо ведучий МК використовує передачу даних, керовану перериванням, та існує ймовірність подачі на вхід $\overline{SS}$ напруги низького рівня, у підпрограмі обробки переривання від SPI обов’язково повинна здійснюватися перевірка стану прапорця MSTR. У разі виявлення скидання цього прапорця його треба знову програмно встановити в одиницю для зворотного переведення МК у режим «Master».

8.2.2.8. Використання інтерфейсу SPI для програмування пам’яті

У МПС з використанням МК сімейств Mega, XMega, а також деяких МК сімейства Tiny для програмування пам’яті може використовуватись послідовний інтерфейс SPI. Зазвичай цей режим використовується для програмування (перепрограмування) МК безпосередньо у пристрої (рис. 8.26) [3; 4].

Зауваження (для ATmega) - якщо у якості тактового використовується внутрішній RC-генератор, вихід XTAL1 залишають не підключеним

Рис. 8.26. Включення МК у режимі програмування послідовним каналом

Як видно з рис. 8.26, для підключення програматора до пристрою використовуються три лінії інтерфейсу: SCK (тактовий сигнал), MOSI (вхід даних) і MISO (вихід даних). Відповідність між лініями інтерфейсу і контактами портів введення/виведення деяких AVR-мікроконтролерів наведено в табл. 8.13.

Таблиця 8.13. **Виводи, що використовуються під час програмування послідовним каналом**

Назва лінії інтерфейсу	**ATmega8515x/8535x**	**ATmega8x**	**ATmega16x/32x**	**ATmega64x/128x**	**ATmega48x/88x/168x**	**ATmega162x**	**ATmega164x/324x/644x**	**ATmega165x ATmega325x/3250x ATmega645x/6450x**	**ATmega640x/1280x/2560x**	**ATmega1281xATmega2561x**	**Призначення виводів**
SCK	PB7	PB5	PB7	PB1	PB5	PB7	PB7	PB1	PB1	PB1	Вхід тактового сигналу
MISO (PDO)	PB6	PB4	PB6	PE1	PB4	PB6	PB6	PB3	PE1	PB3	Вихід даних
MOSI (PDI)	PB5	PB3	PB5	PE0	PB3	PB5	PB5	PB2	PE0	PB2	Вхід даних

В окремих МК, наприклад, ATmega64x та ATmega128x виводи, які використовуються для програмування (табл. 8.13), не збігаються з виводами, призначеними для штатної роботи інтерфейсу SPI.

Часові діаграми сигналів під час програмування МК у розглянутому режимі зображено на рис. 8.27, а значення параметрів сигналів наведено в [3; 4].

Як і в робочому режимі, під час програмування послідовним каналом МК потрібно джерело тактового сигналу. В якості такого може використовуватися кожне із можливих для МК джерел, при цьому повинна виконуватися така умова: тривалість імпульсів як низького рівня, так і високого рівня сигналу SCK має

бути більше двох (за $f_{CLK} < 12$ МГц) або трьох (за $f_{CLK} \geq 12$ МГц) періодів тактового сигналу МК [3; 4].

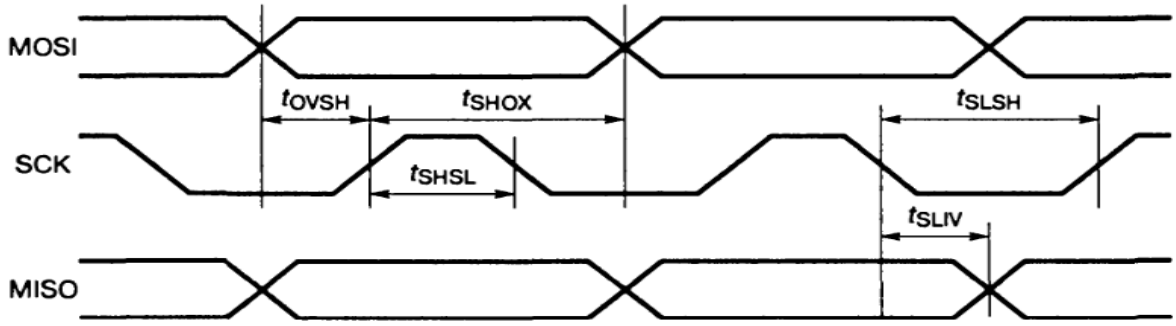


Рис. 8.27. Часові діаграми сигналів під час програмування послідовним каналом

Програмування здійснюється за допомогою посилки 4-байтних команд на вивід MOSI МК. Результат виконання команд читання знімається з виводу MISO МК. Передача команд і видача результатів їхнього виконання здійснюється від старшого розряду до молодшого, при цьому запис вхідних даних виконується за наростаючим фронтом сигналу SCK, а зсув вихідних даних – за спадаючим (рис. 8.28).

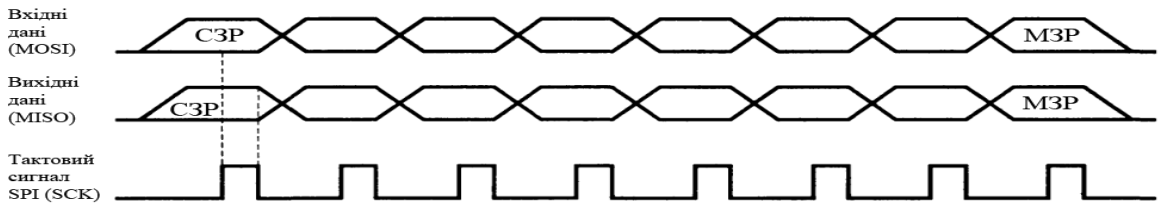


Рис. 8.28. Передача даних під час програмування послідовним каналом

Більш детально це питання розглянуто у [3; 4].

8.2.2.9. Периферійні пристрої з SPI-інтерфейсом

Для спрощення вибору елементної бази під час проектування електронних схем у [3] наведено деякі мікросхеми: АЦП; ЦАП; давачі й т. ін., що містять інтерфейс SPI.

8.2.2.10. Універсальний послідовний інтерфейс USI

Деякі моделі, наприклад ATmega165x, ATmega325x/3250/645x/6450x мають модуль універсального послідовного інтерфейсу (Universal Serial Interface – USI) [3; 4]. Цей модуль є «напівфабрикатом», який надає базові апаратні ресурси, що необхідні для здійснення обміну послідовним каналом. Використовуючи цей модуль, можна досягти суттєво більшої швидкості передачі та отримати більш компактний код, ніж при за суто програмної реалізації різноманітних протоколів обміну.

8.2.3. Моделювання модуля SPI

Моделювання модуля SPI мікроконтролера ATmega8 у пакеті PROTEUS 8.6 описано у [3], де наведено схему алгоритму роботи моделі та робочу програми мовою С. Отримані результати моделювання підтвердили працездатність цього алгоритму та робочої програми.

Контрольні запитання та завдання

1. Який режим обміну використовується в модулі SPI: асинхронний чи синхронний? Відповідь пояснити.
2. На яку відстань і з якою швидкістю можна здійснювати обмін даними через інтерфейс SPI?
3. Поясніть структурну схему модуля SPI.
4. В яких режимах може працювати МК під час обміну даними інтерфейсом SPI?
5. Скільки та які виводи МК використовує модуль SPI?
6. Як виконується перепризначення режиму роботи виводів модуля SPI?
7. Поясніть схему з’єднання двох МК інтерфейсом SPI.
8. Які пристрої знаходяться перед входами 8-розрядних регістрів зсуву ведучого та веденого МК? Яку функцію вони виконують?
9. Який регістр призначено для керування модулем SPI?
10. Який регістр використовується для контролю стану модуля, а також для додаткового керування швидкістю обміну?
11. Як задати режим роботи МК у режимі «Master»?
12. В який спосіб здійснюється передача даних у модулі SPI?
13. Як визначити кінець передачі байта?
14. За якої умови під час передачі даних від ведучого до веденого можлива передача у зворотному напрямку?
15. Що означає те, що у модулі реалізовано одинарну буферизацію під час передачі та подвійну під час прийому?
16. Поясніть структуру SPI-мережі МК.
17. Як обирається потрібний ведений МК у SPI-мережі?
18. Поясніть вплив сигналу $\overline{SS}$ на початок обміну даними в різних режимах роботи модуля.
19. Чи можна до інтерфейсу підключати декілька периферійних пристроїв?
20. Назвіть кількість режимів передачі даних та принцип їх роботи.
21. Що є джерелом тактового сигналу під час роботи модуля? Як формується цей сигнал?
22. З яких етапів складається обмін у режимі SPI?
23. Поясніть схему включення МК у режимі програмування послідовним каналом під час використання інтерфейсу SPI.

8.3. Мережа на основі CAN-інтерфейсу

8.3.1. Особливості архітектури CAN-мережі

8.3.1.1. Загальні відомості про CAN-мережі

CAN (англ. Controller Area Network – мережа контролерів) є асинхронним послідовним протоколом високошвидкісної та високонадійної передачі даних у широкомовному режимі в мультимайстерному середовищі. CAN-протокол було розроблено фірмою “Bosch” для систем керування вузлами автомобіля. Положення стандарту, які закріплено у використовуваній на сьогоднішній день специфікації 2.0 A/B фірми “Bosch” та міжнародному стандарті ISO 11898, відповідають двом початковим рівням (фізичному та канальному) семирівневої моделі взаємодії відкритих систем ISO/OSI [3; 7].

Основними перевагами цього стандарту є:

- зрілість стандарту. Протокол вже багато років використовується в мікроконтролерних мережах. На сучасному ринку представлено тисячі різних CAN-виробів;
- апаратна підтримка протоколу;
- високонадійний алгоритм передачі та обробки помилок, механізм, що дозволяє від’єднувати несправний віддалений вузол і тим самим не допускати блокування всієї мережі;
- хороша підтримка систем реального часу та систем, що керовані подіями.

Controller Area Network має асинхронну послідовну структуру шини з одним логічним сегментом мережі. CAN-мережа може складатися з двох або більше вузлів, з можливістю під’єднання та від’єднання вузлів від шини без переналагоджування інших пристроїв. Логіка шини працює за механізмом «монтажне І», в якому «рецесивний» біт відповідає логічній одиниці, а «домінантний» – логічному нулю. Якщо жоден вузол не формує нульовий біт, шина перебуває в одиничному стані. Поява нульового біта від будь-якого вузла створює нульовий стан шини.

У CAN-мережі інформація передається парою скручених дротів (кручена пара). Лінії шини називаються CANH та CANL. Через диференціальний характер лінії зв’язку CAN-шина малочутлива до електромагнітних завад. Для підвищення надійності у високошвидкісних режимах роботи застосовують екранування шини.

Двійкова інформація кодується NRZ-кодом. Код NRZ (Non Return to Zero – без повернення до нуля) – це найпростіший двійковий код, що передається звичайним цифровим сигналом. Логічному нулю відповідає низький рівень напруги в кабелі, логічній одиниці – високий рівень напруги (або навпаки, що не

принципово). Протягом бітового інтервалу (bit time, BT), тобто часу передачі одного біта ніяких змін рівня сигналу в кабелі не відбувається.

До переваг NRZ-коду належить його досить проста реалізація (вихідний сигнал не треба спеціально кодувати на передавальному кінці та декодувати на приймальному кінці). Порівняно з іншими кодами потрібна мінімальна пропускна здатність лінії зв'язку.

Для синхронізації даних всіма вузлами використовується додатковий наповнюючий біт (виконується біт-стаффінг). Під час послідовної передачі п'яти біт однакової полярності, передавач вставляє один додатковий біт протилежної полярності. Приймач також перевіряє полярність та знищує додаткові біти (виконує дебіт-стаффінг).

Структуру типової CAN-мережі у МПС керування показано на рис. 8.29.

Кожне повідомлення, яке передається мережею, має оригінальний 11-бітний або 29-бітний ідентифікатор.

У разі групового доступу до шини використовується неруйнівний арбітраж з опитуванням стану шини. Перед початком передачі даних вузол перевіряє стан шини (відсутність активності на шині). Коли починається передача повідомлення, вузол стає керуючим для шини, а всі інші вузли переходять у режим прийому. Кожен вузол видає підтвердження прийому, перевіряє ідентифікатор повідомлення, обробляє або знищує прийняті дані. Якщо два або більше вузли починають передачу одночасно, порозрядний арбітраж дозволяє уникнути конфлікту на шині. Кожен вузол видає на шину свій ідентифікатор і контролює її стан.

Якщо вузол посилає «рецесивний» біт, а читає «домінантний», значить арбітраж втрачено і вузол перемикається в стан прийому. Таким чином, вузол з більш високим пріоритетом виграє арбітраж без необхідності повторювати передачу повідомлення. Всі інші вузли будуть намагатися передати повідомлення після того, як шина звільниться.

Даний механізм не дозволяє передавати одночасно повідомлення з однаковим ідентифікатором, оскільки можуть виникати помилки. Оригінальна специфікація – CAN 2.0A визначає довжину ідентифікатора 11 біт. Такі повідомленнями називаються стандартними, при цьому можливо використовувати лише 2048 різних ідентифікаторів, з яких 16 ідентифікаторів з найменшим пріоритетом: 2032…2047 – зарезервовано. CAN-модулі версії 2.0A можуть тільки передавати або приймати дані.

Рис. 8 . 29 Мікропроцесорна система керування

У версії CAN 2.0B довжина ідентифікатора може бути 11 або 29 біт (536 мільйонів варіантів).

Версія 2.0B дозволяє активним вузлам додатково робити віддалений стандартний або розширений запит та отримувати дані від потрібного пристрою, використовуючи відповідно стандартне або розширене повідомлення.

CAN-модуль апаратно реалізує шостий рівень та PLS-, MDI- та PMA-підрівні еталонної ISO/OSI-моделі взаємозв'язку відкритих систем OSI (Open System Interconnection) [3; 12].

Шостий (канальний) рівень – має LCC- та MAC-підрівні. LCC-підрівень забезпечує: фільтрацію повідомлень; повідомлення про перевантаження; керування пошуком помилок. MAC-підрівень забезпечує: отримання повідомлень; виконання арбітражу; перевірку помилок; передачу повідомлень та сигналів помилки; видачу висновку про несправність.

PLS-підрівень забезпечує: синхронізацію біт; кодування біт; загальну синхронізацію. PMA-підрівень є мікросхемою CAN-трансивера, а MDI-підрівень – роз'єм для під'єднання до шини.

8.3.1.2. Основні характеристики CAN-протоколу

До основних характеристик CAN-протоколу належить [3; 7]:

- пріоритетність повідомлень;
- гарантований час відгуку;
- гнучкість конфігурації;
- груповий прийом із синхронізацією часу;
- система несуперечності даних;
- робота у мультимайстерному режимі;
- виявлення помилок та сигналізація про їх наявність;
- автоматична ретрансляція пошкоджених повідомлень, як тільки шина знову стане вільною;
- можливість відрізняти нерегулярні помилки та постійні відмови вузлів, а також автономне вимикання дефектних вузлів.

Інформація на шині подається у вигляді фіксованих повідомлень різної, але обмеженої довжини. Коли шина вільна, будь-який підключений до неї вузол може почати передавати нове повідомлення.

У CAN-мережах немає ніякої інформації щодо конфігурації системи (наприклад, адреса вузла). Вузли можуть бути додані до CAN-мережі без зміни програмного забезпечення та апаратних засобів будь-якого вузла або прикладного рівня.

Зміст повідомлення визначається *ідентифікатором*. Ідентифікатор не вказує адресата повідомлення, але описує значення даних так, щоб всі вузли в мережі мали здатність фільтрацією вирішити, чи повинні вони реагувати на це повідомлення чи ні. Будь-яка кількість вузлів може отримувати та одночасно реагувати на одне і теж повідомлення. В середині CAN-мережі гарантується, що повідомлення одночасно буде прийнято або всіма вузлами або жодним вузлом.

Швидкість CAN може бути різною в різних системах. Однак у конкретній мережі швидкість передачі інформації має бути фіксованою. Ідентифікатор визначає статичний пріоритет повідомлення протягом доступу до шини.

Посилаючи кадр віддаленого запиту даних, вузол може виконувати запит даних у іншого вузла, які той повинен передати у вигляді відповідного кадру даних. Кадр віддаленого запиту даних та кадр даних мають однаковий ідентифікатор.

Коли шина вільна, будь-який вузол може починати передавати повідомлення. Вузлу з повідомленням більш високого пріоритету, буде передане право на доступ до шини.

Якщо два або більше вузлів починають одночасно передавати повідомлення, конфлікт доступу до шини вирішується порозрядним арбітражем з використанням ідентифікатора. Якщо кадр даних і кадр віддаленого запиту даних з однаковим ідентифікатором починають передаватися одночасно, то кадр даних має перевагу над кадром віддаленого запиту даних. Протягом арбітражу кожен передавач порівнює рівень переданого біта з рівнем, що спостерігається на шині. Якщо ці рівні однакові, вузол може продовжувати передачу. Коли передано одиничний рівень, а спостерігається нульовий рівень, це означає, що вузол втратив право арбітражу і не повинен посилати більше жодного біта.

Для досягнення безпеки передачі даних у кожному CAN-вузлі є потужні засоби самоконтролю, виявлення та повідомлення про помилки.

Для виявлення помилок застосовуються такі засоби:

- поточний контроль (передавачі порівнюють рівні бітів, що передаються, з рівнями, виявленими на шині);
- контроль правильності прийнятих повідомлень із використанням надлишкового циклічного коду;
- контроль формату переданого кадру.

Механізми виявлення помилок дозволяють виявляти [3; 7]:

- всі помилки глобального характеру;
- всі локальні помилки передавача;

– до 5 випадкових помилок у повідомленні;
– послідовну групу помилок завдовжки до 15 біт;
– будь-які помилки непарності.

Помилки передач будуть виявлятися з досить високою ймовірністю за допомогою вбудованого механізму виявлення помилок CAN-протоколу.

Пошкоджені кадри позначаються кожним вузлом, що виявив помилку. Такі кадри перериваються та автоматично передаються заново.

CAN-вузли здатні відрізнити короткі неполадки від постійних відмов. Дефектні вузли від'єднуються від шини CAN-модуля.

Послідовна лінія зв'язку CAN є шиною, до якої теоретично може бути під'єднано будь-яку кількість вузлів. Фактично загальна кількість вузлів буде обмежено часовою затримкою та (або) електричним навантаженням на лінії шини.

Шина складається з каналу обміну, що передає біти. Із цих даних може бути отримана інформація про пересинхронізацію. В якості каналу обміну можуть бути два диференціальних дроти, оптичне скловолокно і т. ін.

Шина може мати одне із двох логічних значень: «домінантне» або «рецесивне». Вважається, що «домінантний» рівень еквівалентний нульовому рівню, а «рецесивний» рівень еквівалентний одиничному рівню.

У разі одночасної передачі «домінантного» та «рецесивного» бітів значення, що виникає на шині буде «домінантне».

Усі приймачі перевіряють несуперечність отриманого повідомлення, підтверджують несуперечність повідомлення та позначають суперечливі повідомлення.

Для зменшення кількості енергії, яка споживається від джерела живлення, CAN-вузол може бути переведений у режим «сну» без внутрішньої активності та з відключеними формувачами шини. Вихід із цього режиму відбувається у разі появи будь-якої активності на шині або внутрішньому стані системи. Під час пробудження здійснюється внутрішнє перезавантаження. MAC-підрівень буде чекати стабілізації генератора системи, і потім буде очікувати самосинхронізації із сигналами на шині (перевіряючи одинадцять послідовних одиничних біт), поки вихідні формувачі знову не встановляться у стан «під'єднано до шини».

Вимоги до синхронізації дозволяють використовувати керамічні резонатори в системах зі швидкостями передачі до 125 Кбіт/с. Для більш високої швидкості CAN-шини потрібен кварцовий резонатор.

8.3.2. Структура повідомлень CAN-мережі

8.3.2.1. Загальна характеристика повідомлень

Повідомлення, що передаються CAN-шиною, називаються кадрами (фреймами). Є чотири типи кадрів:

1. Data Frame – кадр даних.
2. Remote Frame – кадр запиту даних.
3. Error Frame – кадр помилки.
4. Overload Frame – кадр перевантаження.

Є два формати даних (повідомлень), які передаються, що відрізняються довжиною поля ідентифікатора:

- кадри з 11-розрядним ідентифікатором, які називаються стандартними кадрами;
- кадри, що містять 29-розрядні ідентифікатори та називаються розширеними кадрами.

Кадр даних передає дані від передавача до приймача. Кадр віддаленого запиту даних передається вузлом, щоб запросити передачу кадру даних від іншого вузла з тим же самим ідентифікатором. Кадр помилки передається будь-яким вузлом у разі виявлення помилки на шині. Кадр перевантаження використовується, щоб забезпечити додаткову затримку між попереднім і наступним кадром даних або кадром віддаленого запиту даних. Кадри даних і кадри віддалених запитів даних можуть використовуватися у стандартному та розширеному форматах. Вони відокремлюються від попередніх кадрів міжкадровим простором.

8.3.2.2. Кадр даних

Загальна характеристика кадру даних

Кадр даних (Data Frame) складається із семи різних полів: «початок кадру» (start of frame); «поле арбітражу» (arbitration field); «поле керування» (control field); «поле даних» (data field); «поле CRC» (CRC field); «поле підтвердження» (ACK-field) та «кінець кадру» (end of frame). Поле даних може мати нульову довжину. На рис. 8.30, 8.31 наведено структури, відповідно стандартного та розширеного повідомлень.

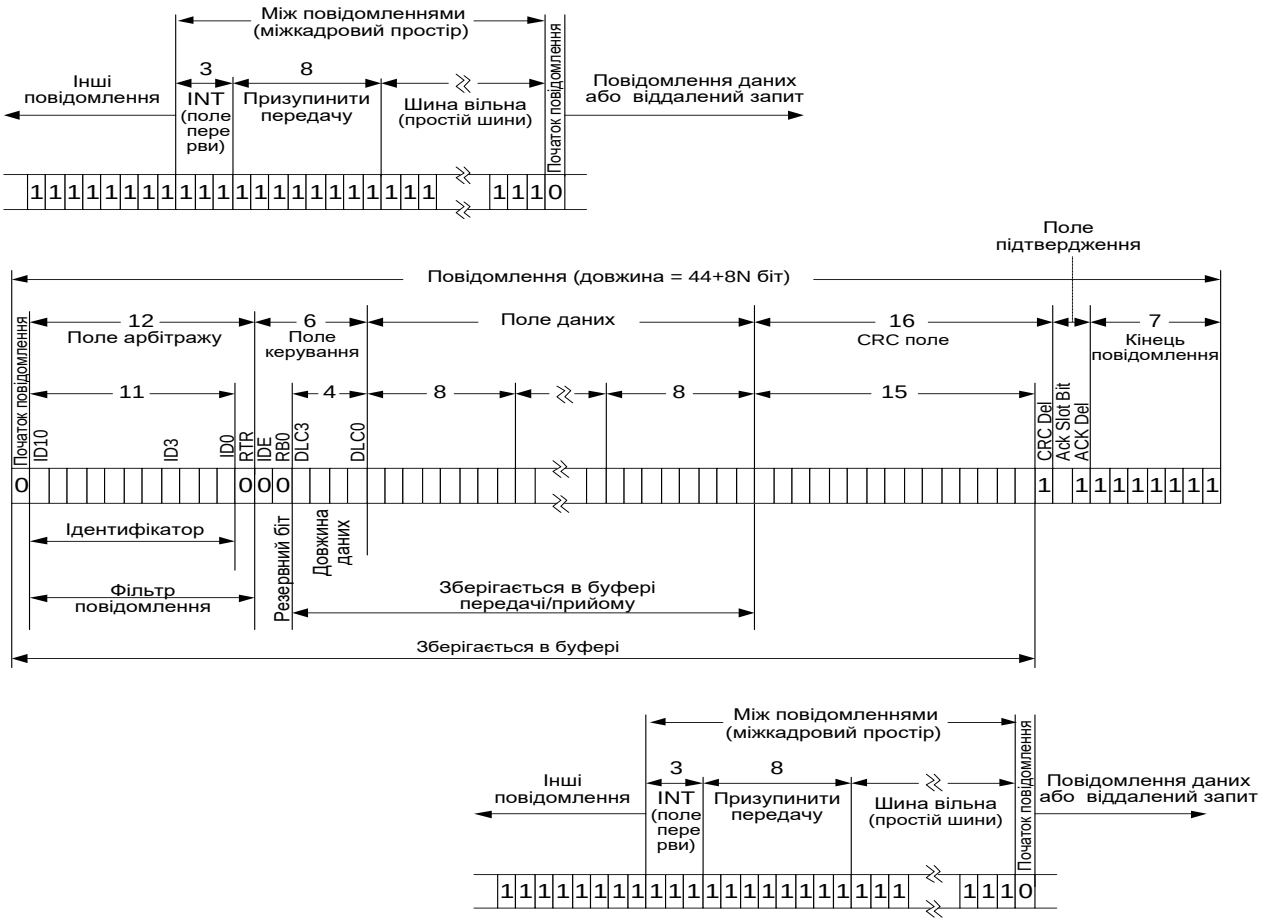


Рис. 8.30. Стандартне повідомлення

Розглянемо окремі поля цих повідомлень.

Ідентифікатор стандартного повідомлення

Довжина ідентифікатора (Identifier) стандартного повідомлення – 11 біт і відповідає BASE ID у розширеному форматі (рис. 8.31). Ці біти передаються в порядку ID10...ID0. Молодший біт – ID0. Сім старших бітів (ID10…ID4) не мають бути одиничними бітами. У стандартному кадрі ідентифікатор супроводжується RTR-бітом, який дорівнює нулю.

Ідентифікатор розширеного повідомлення

На відміну від стандартного ідентифікатора, розширений ідентифікатор (рис. 8.31) складається з 29 біт. Його формат містить дві секції:

1. Base ID – 11 біт.
2. Extended ID – 18 біт.

Base ID

Base ID складається з 11 біт. Ця секція передається в порядку від ID28 до ID18.

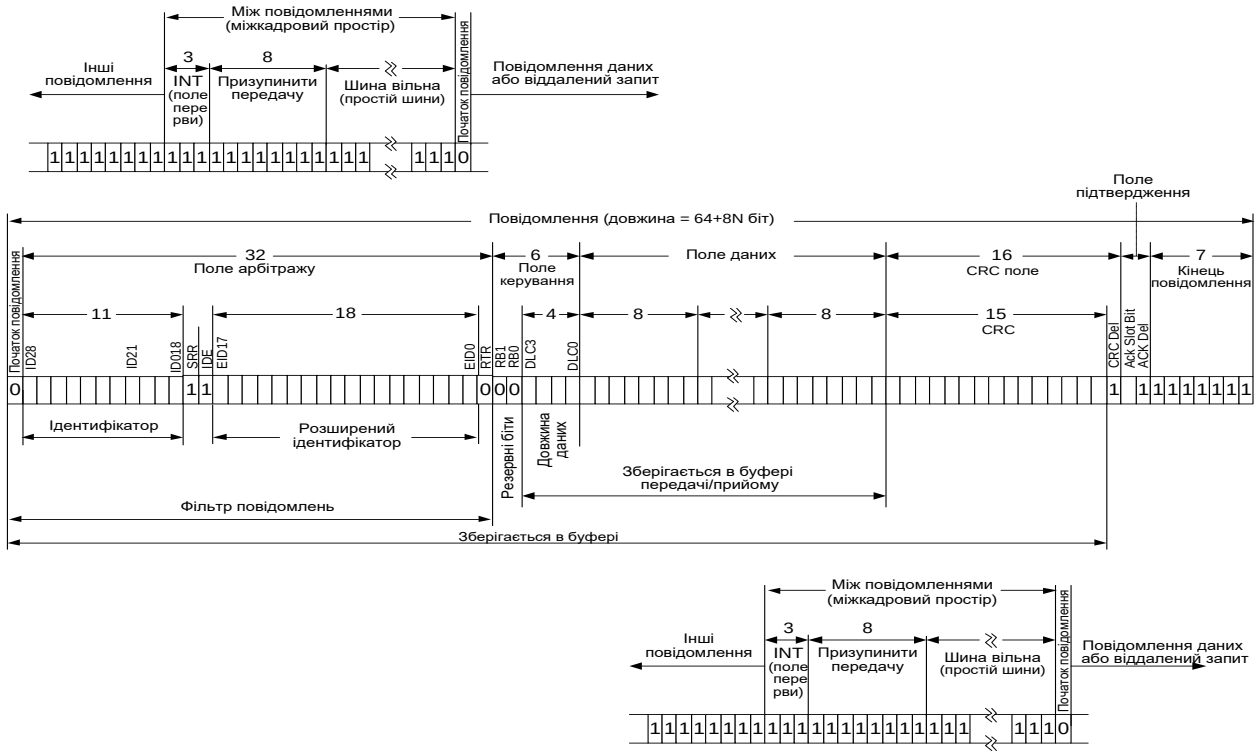


Рис. 8.31. Розширене повідомлення

Це еквівалентно формату стандартного ідентифікатора. Base ID визначає базовий пріоритет розширеного кадру.

Extended ID

Extended ID складається з 18 біт. Ця секція передається в порядку від ID17 до ID0. У розширеному кадрі ідентифікатор супроводжується RTR-бітом, який дорівнює нулю.

Біт віддаленого запиту RTR (стандартний і розширений формат)

У кадрах даних біт віддаленого запиту – RTR передається нульовим рівнем. У кадрі віддаленого запиту даних RTR-біт є одиничним. У розширеному кадрі спочатку передається Base ID, а наступними бітами передаються SRR та IDE. Extended ID передається після IDE-біта.

Біт SRR (розширений формат)

У розширених кадрах біт SRR дорівнює одиниці та замінює RTR-біт стандартного кадру. У разі одночасної передачі стандартного кадру та розширеного кадру, Base ID якого збігається з ідентифікатором стандартного кадру, стандартний кадр має перевагу над розширеним кадром.

Біт IDE (розширений формат)

Біт розширеного ідентифікатора IDE належить:

– полю арбітражу для розширеного формату;

– полю керування для стандартного формату.

IDE-біт у стандартному форматі передається нульовим рівнем, а у розширеному форматі – одиничним рівнем.

Поле керування (стандартний і розширений формат)

Поле керування (Control field) складається із шести біт. Формат поля керування відрізняється для стандартного та розширеного формату. Кадри в стандартному форматі включають: код довжини даних DLC, біт IDE, що передається нульовим рівнем (див. вище), та зарезервований біт RB0. Кадри поля керування в розширеному форматі включають код довжини даних і два зарезервованих біти RB1 та RB0. Зарезервовані біти повинні мати нульовий рівень.

Код довжини даних

Кількість байт у полі даних стандартного та розширеного форматів визначається кодом довжини даних – Data length code. Цей код має довжину 4 біти та передається всередині поля керування (табл. 8.14).

Таблиця 8.14. **Кодування довжини даних**

Кількість байт даних	**Код довжини даних**			
	DLC3	**DLC2**	**DLC1**	**DLC0**
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d	d	d
d – «домінантний» r – «рецесивний»				

Допустиме кількість байт даних, що передаються: 0; 1; ... 8. Інші величини використовуватися не можуть.

Поле даних

Поле даних (Data field) стандартного та розширеного форматів складається з даних, які будуть передані всередині кадру даних. Воно може містити від 0 до 8 байт, кожен з яких містить 8 біт, які передаються, починаючи зі старшого значущого розряду.

Поле CRC

Поле CRC стандартного та розширеного форматів (CRC field) містить послідовність CRC та CRC-роздільник.

Послідовність CRC

Під час обчислення послідовності контролю кадру (CRC Sequence) береться початковий поліном, коефіцієнти якого задано послідовністю біт, які складаються з полів: «початок кадру», «поле арбітражу», «керуюче поле», «поле даних» (якщо воно є) і для 15 молодших коефіцієнтів – нулів. Цей поліном ділиться на поліном:

$$X^{15} + X^{14} + X^{10} + X^{8} + X^{7} + X^{4} + X^{3} + 1.$$

Залишок від цього поліноміального ділення і є послідовність CRC, яка передається шиною.

Приклад розрахунку поля CRC

Нижче наведено приклад розрахунку п’ятнадцяти бітів поля CRC для кадру даних, в якому передається повідомлення з ідентифікатором – 00000000001 з одним байтом даних, – 00001111 (рис. 8.32).

Процес поліноміального ділення для обчислення поля CRC показано на рис. 8.33, а повідомлення з розрахованим полем CRC – на рис. 8.34.

Початок повідомлення	Ідентифікатор	RTR	IDE	RB0	Довжина данних	Поле даних	CRC поле	CRC Del	ACK Slot Bit	ACK Del	Кінець кадру
0	0 0 0 0 0 0 0 0 0 0 1	0	0	0	0 0 0 1	0 0 0 0 1 1 1 1	x x x x x x x x x x x x x x x	1	1		1 1 1 1 1 1 1

Рис. 8.32. Повідомлення, яке потрібно передати

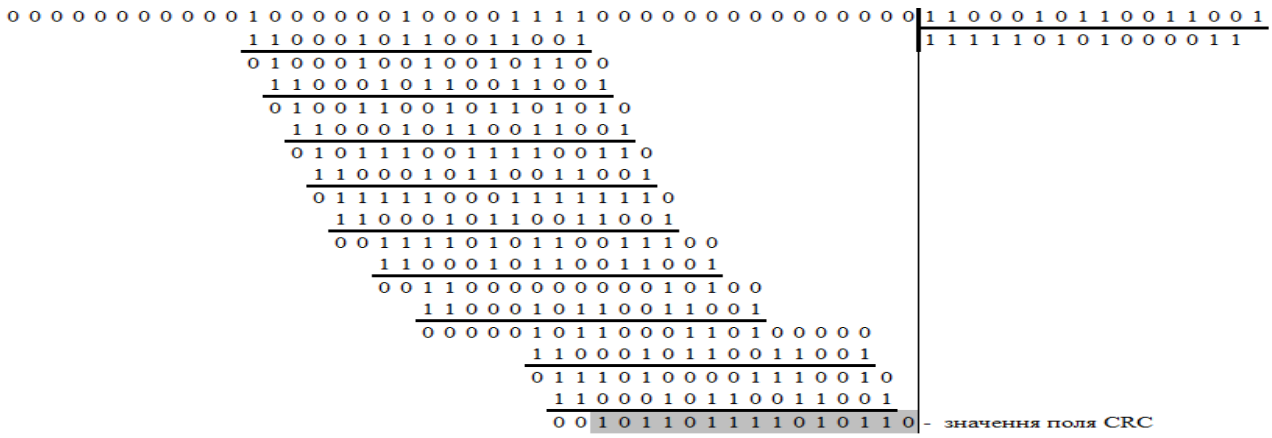


Рис. 8.33. Поліноміальне ділення

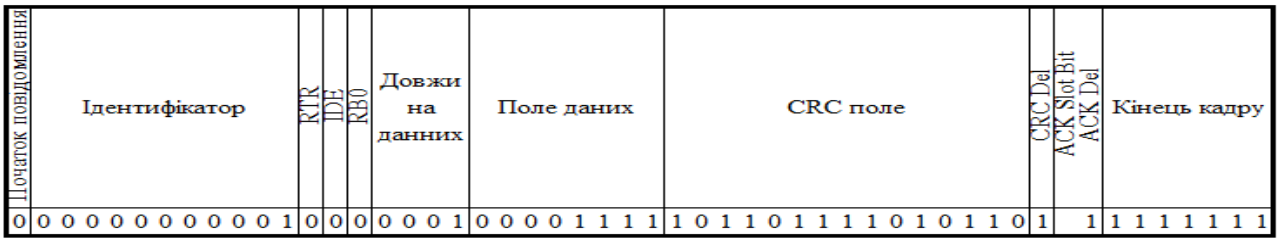


Рис. 8.34. Повідомлення з розрахованим полем CRC

Роздільник CRC (стандартний та розширений формати)

Послідовність CRC супроводжується роздільником CRC (CRC-Delimiter), що складається з одного одиничного біта.

Циклічні CRC-коди названо так тому, що в них частина комбінацій коду або всі комбінації можуть бути отримані за допомогою циклічного зсуву однієї або декількох комбінацій коду. Циклічний зсув здійснюється справа наліво, причому крайній лівий символ щоразу переноситься в кінець комбінації. Практично всі циклічні коди належать до систематичних кодів тим, що у них контрольні та інформаційні розряди розміщено на чітко визначених місцях. Крім того, циклічні коди належать до кількості блочних кодів. Кожен блок кодується самостійно. Ідея побудови циклічних кодів ґрунтується на використанні многочленів, що не приводяться в полі двійкових чисел. Многочлени, що не приводяться – це такі многочлени, які діляться без залишку тільки на себе і на одиницю.

Поле підтвердження (стандартний та розширений формати)

Поле підтвердження (ACK-field) має довжину два біти й містить: перший біт – «Область підтвердження» і другий біт – роздільник підтвердження. В поле підтвердження передавальний вузол посилає два біти з одиничним рівнем. Приймач, що отримав повідомлення правильно, сповіщає про це передавачу, посилаючи біт з нульовим рівнем протягом прийому біта «область підтвердження».

Біт «Область підтвердження»

Усі вузли, що отримали повідомлення з відповідною послідовністю CRC, сповіщають про це під час прийому біта «Область підтвердження» (ACK-Slot Bit) за допомогою заміни біта з одиничним рівнем на біт з нульовим рівнем.

Роздільник підтвердження

Роздільник підтвердження (ACK-delimiter) – другий біт поля підтвердження, який має одиничний рівень.

Кінець кадру (стандартний та розширений формати)

Кожен кадр даних і кадр віддаленого запиту даних закінчується послідовністю прапорців, яка складається із семи одиничних біт (Endof Frame).

8.3.2.3. Кадр віддаленого запиту даних

Вузол, який виконує прийом даних, може запросити передачу відповідних даних від потрібних вузлів, посилаючи кадр віддаленого запиту даних (Remote Frame) (рис. 8.35).

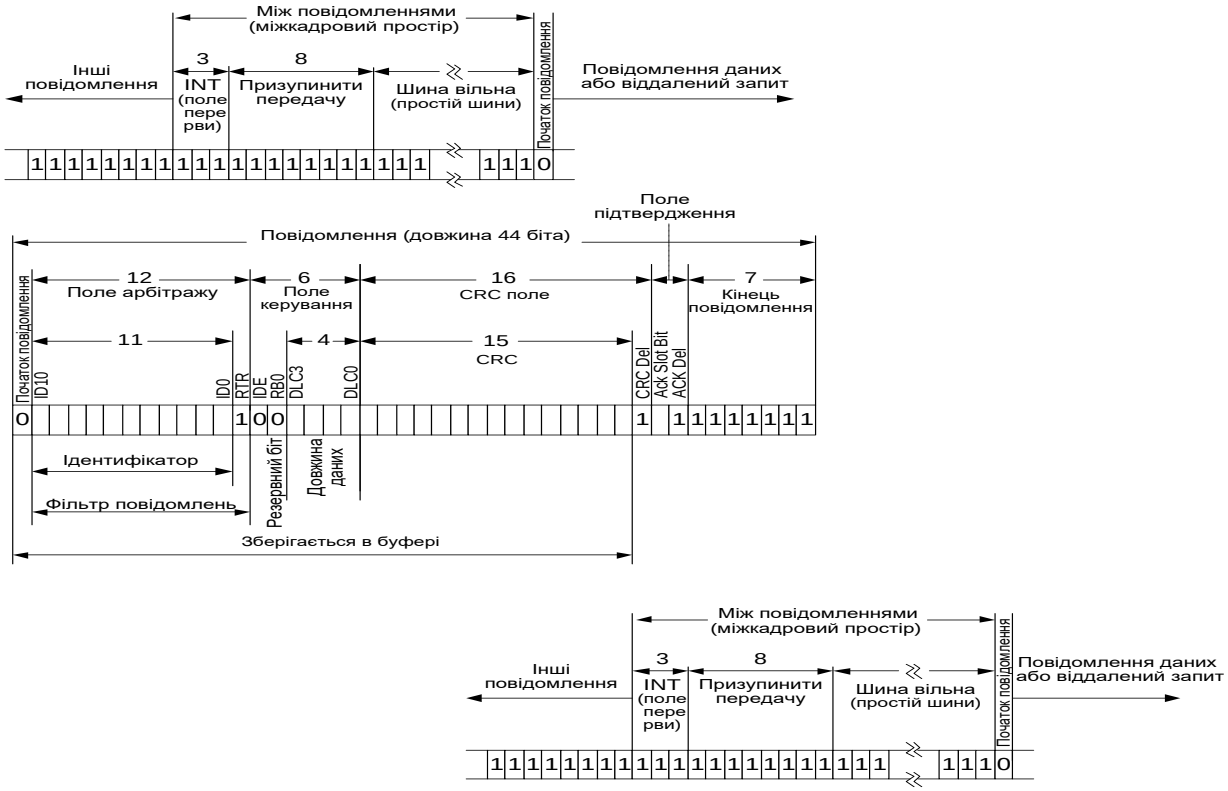


Рис. 8.35. Кадр віддаленого стандартного запиту даних

Кадр віддаленого запиту даних є у стандартному та розширеному форматах. В обох випадках він складається із шести бітових полів: «початок кадру (повідомлення)» (Start of Frame), «поле арбітражу» (Arbitration-field), «поле керування» (Control-field), «поле CRC» (CRC-field), «поле підтвердження» (ACK-field), «кінець кадру» (End of Frame). На відміну від кадру даних, RTR-біт кадру віддаленого запиту даних – одиничний. У цьому кадрі поле даних відсутнє. Значення коду довжини даних відповідає коду довжини даних кадру даних, який запитується. RTR-біт вказує, чи є переданий кадр кадром даних чи кадром віддаленого запиту.

8.3.2.4. Кадр помилки

Кадр (повідомлення) помилки (Error frame) складається із двох різних полів. Перше поле є суперпозицією прапорців помилки, отриманих від вузла, який передає кадр помилки, та інших вузлів. Наступне поле – роздільник помилки (Error Delimiter). Формат кадру помилки та опис його окремих полів наведено у [3; 7].

8.3.2.5. Кадр перевантаження

Кадр перевантаження (Overload Frame) містить два бітових поля: прапорець перевантаження та роздільник перевантаження [3; 7].

Є три види перевантаження, які приводять до передачі прапорця перевантаження:

- внутрішній стан приймача, який потребує затримки наступного кадру даних або кадру віддаленого запиту даних;
- виявлення «домінантного» біта під час передачі першого та другого бітів поля перерви;
- якщо вузол виявляє «домінантний» біт на восьмому біті (останньому біті) роздільника помилки або роздільника перевантаження, це спричиняє передачу кадру перевантаження (а не кадру помилки). Лічильники помилок не будуть збільшені.

У табл. 8.15 показано вплив бітів RTR, SRR та IDE на тип повідомлення, що передається.

Таблиця 8.15. **Вплив бітів RTR, SRR та IDE на тип повідомлення, що передається**

№ з/п	Значення біта			Тип повідомлення
	RTR у стандартному повідомленні (SRR у розширеному повідомленні)	IDE	RTR у розширеному повідомленні	
1	0	0	–	Стандартне повідомлення
2	1	1	0	Розширене повідомлення
3	1	0	–	Віддалений стандартний запит
4	1	1	1	Віддалений розширений запит

8.3.3. CAN-модуль AVR-мікроконтролера

8.3.3.1. Загальні відомості

У сімействі AVR є, наприклад, МК AT90CAN128, який має CAN-модуль. Він апаратно підтримує протоколи CAN-2.0A та CAN-2.0B.

Нижче наведено основні характеристики цього МК:

- стандартний і розширений типи повідомлень;
- довжина даних у повідомленні від 0 до 8 байт;
- програмована швидкість передачі інформації до 1 Мбіт/с;
- підтримка віддаленого запиту даних;
- 15 повних об’єктів повідомлень з окремими ідентифікаторами і масками;
- можливість указання пріоритету та аварійного припинення передачі;
- можливість пробудження з SLEEP-режиму;
- гнучка система переривань від CAN-модуля;
- низьке енергоспоживання в режимі SLEEP.

8.3.3.2. Структура CAN-модуля

Структуру CAN-модуля МК AT90CAN128 наведено на рис. 8.36.

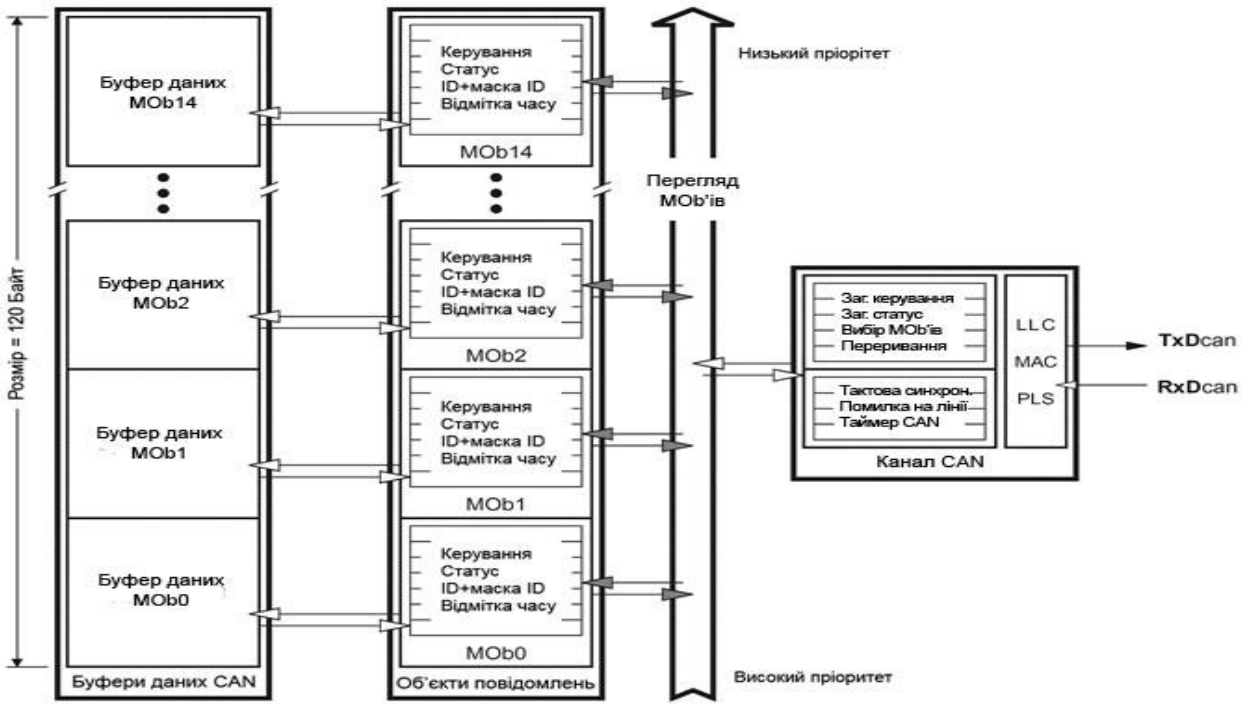


Рис. 8.36. Структура CAN-модуля МК AT90CAN128

МК забезпечує повну апаратну підтримку фільтрації повідомлень та керування ними. Для кожного повідомлення, яке має бути передано або

отримано, CAN-модуль містить об’єкт повідомлення – MOb (Message Objects).

Останній було розроблено для опису CAN-кадру, як об’єкта. MOb є дескриптором CAN-кадру та містить всю інформацію для роботи з ним.

8.3.3.3. Організація керуючих регістрів

Склад та організацію керуючих регістрів CAN-модуля наведено на рис. 8.37.

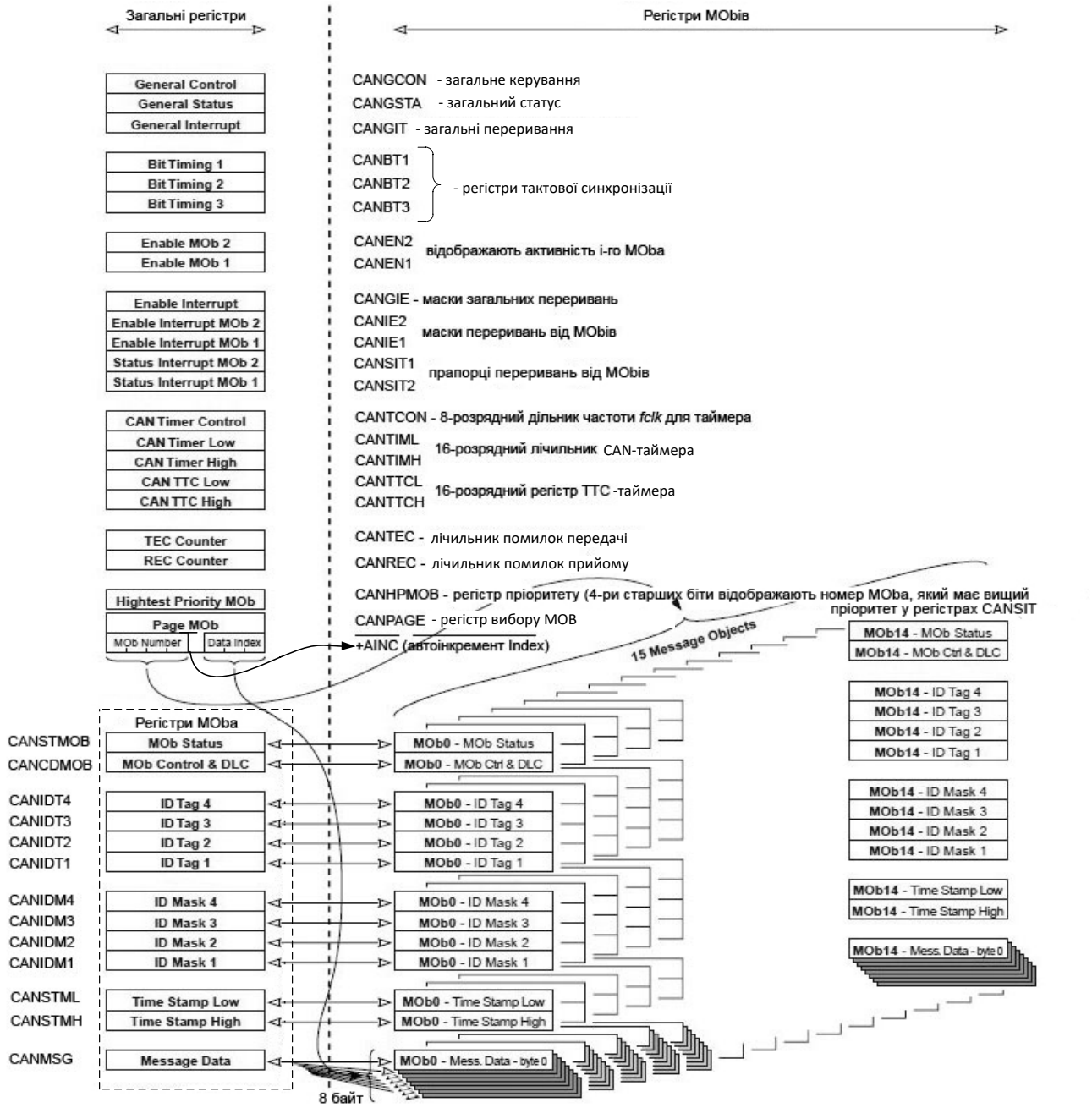


Рис. 8.37. Організація керуючих регістрів CAN-модуля

Ці регістри поділяються на дві групи: загальні регістри та регістри MOb [3].

Для програмування CAN-модуля в цілому використовують загальні регістри, а для програмування об’єктів повідомлень – регістри MOb.

8.3.3.4. Режими роботи CAN-модуля

Загальна характеристика режимів роботи

Кожен MOb має свої власні біти, щоб керувати поточним режимом. Після перезавантаження МК жоден з MOb не має вибраного режиму роботи.

Перед активацією периферійних пристроїв, під’єднаних до CAN-модуля, кожен MOb має бути налаштований відповідним чином на поточний режим (виняток лише для дезактивованого режиму – біти CONMOB0; CONMOB1 регістра CANCDMOB дорівнюють нулю) [3].

Налаштування об’єктів повідомлень виконується згідно з табл. 8.16.

Таблиця 8.16. **Налаштування об’єктів повідомлень**

Конфігурація MOb		**Відповідь**	**RTR Tag**	**Поточний режим**
CONMOB0	**CONMOB1**	**Біт RPLV регістра CANCDMOB**		
0	0	x	x	MOb дезактивований
0	1	x	0	Передача кадру даних
		x	1	Передача кадру віддаленого запиту
1	0	x	0	Прийом кадру даних
		0	1	Прийом кадру віддаленого запиту
		1		Прийом кадру віддаленого запиту та автоматична відповідь
1	1	x	x	Прийом кадрів в буфер (прийом мультикадрів)

CAN-модуль визначає, які повідомлення повинні передаватися, а які прийматися. Коли ідентифікатор отриманого повідомлення збігається з одним із запрограмованих ідентифікаторів, дозволених до прийому повідомлень, об’єкт повідомлення зберігається і програма інформується перериванням. Також на повідомлення віддаленого запиту можуть автоматично надсилатися відповідні дані.

Дезактивований режим

У цьому режимі MOb залишається «вільним» (табл. 8.16).

Передача кадру даних або віддаленого запиту (Tx-конфігурація)

Робота CAN-модуля в цьому режимі відбувається у такій послідовності:

1. Перед відправкою повідомлення або кадру віддаленого запиту наведені нижче біти керуючих регістрів MOb мають бути ініціалізовані [3]:

- IDT (регістри CANIDT1…CANIDT4) – 11-бітний або 29-бітний ідентифікатор;
- IDE (регістр CANCDMOB) – біт ознаки розширеного ідентифікатора;
- RTRTAG (регістр CANIDT4) – біт ознаки віддаленого запиту на передачу;
- DLC (регістр CANCDMOB) – розмір поля даних;
- RBnTAG (регістр CANIDT4) – зарезервовані поля;
- MSG (регістри CANMSG, CANPAGE) – поле даних.

2. Після встановлення Tx-конфігурації (табл. 8.16) та ініціалізації регістрів MOb готовий відіслати кадр даних або віддаленого запиту.

3. Далі CAN-модуль переглядає (сканує) всі MOb у Tx-конфігурації. Можлива ситуація, коли декілька MOb очікують передачі:

- в наслідок втрати арбітражу шини або невдалої попередньої передачі;
- ініціювання передачі декількох наступних MOb до завершення передачі поточного;
- активування CAN-модуля за попереднім налаштуванням на передачу декількох MOb.

У цьому випадку CAN-модуль знаходить MOb, що має найбільший пріоритет і намагається надіслати його.

4. Коли передачу завершено, встановлюється біт TXOK регістра CANSTMOB. У цьому випадку можливе переривання, якщо воно не замасковано.

5. Усі параметри і дані залишаються доступними в MOb до нової ініціалізації.

Схему алгоритму підготовки MOb до передачі повідомлення чи віддаленого запиту наведено у [3], а керуючу програму мовою С – у [7].

Прийом кадру даних або віддаленого запиту (Rx-конфігурація)

Робота CAN-модуля в цьому режимі відбувається у такій послідовності:

1. Перед прийомом повідомлення наведені нижче поля керуючих регістрів MOb мають бути ініціалізовані:

- IDT – 11-бітний або 29-бітний ідентифікатор (регістри CANIDT1…CANIDT4);
- IDMSK – маска IDT (регістри CANIDM1…CANIDM4);
- IDE – біт ознаки розширеного ідентифікатора (регістр CANCDMOB);
- IDEMSK – маска розширеного ідентифікатора IDE (регістр CANIDM4);
- RTRTAG – біт ознаки віддаленого запиту на передачу (регістр CANIDT4);
- RTRMSK – біт маски RTRTAG (регістр CANIDM4);
- DLC – розмір поля даних (регістр CANCDMOB);
- RBnTAG – резервні поля (регістр CANIDT4).

2. Після встановлення Rx-конфігурації (табл. 8.16) та ініціалізації регістрів MOb готовий прийняти кадр даних або віддаленого запиту.
3. Коли з CAN-мережі отримано ідентифікатор кадру, CAN-модуль переглядає всі MOb намагаючись знайти відповідний MOb з найбільшим пріоритетом.
4. IDT-, IDE-, та DLC-поля знайденого відповідного MOb оновлюються значеннями із прийнятого кадру.
5. Як тільки прийом закінчено, байти даних отриманого повідомлення зберігаються (не для кадру віддаленого запиту) в буфері даних відповідного MOb (рис. 8.36) і встановлюється біт RXOK регістра CANSTMOB. У цьому випадку можливе переривання, якщо воно не замасковано.
6. Усі параметри і дані залишаються доступними в MOb до наступної ініціалізації.

Схему алгоритму підготовки MOb до прийому повідомлення чи віддаленого запиту наведено у [3], а керуючу програму мовою С – у [7].

Автоматична відповідь

Робота CAN-модуля в цьому режимі відбувається у такій послідовності:

1. Кадр даних у відповідь на кадр віддаленого запиту може бути автоматично відправлено після прийому очікуваного кадру віддаленого запиту, якщо біт RPLV регістра CANCDMOB встановлено в одиницю (табл. 8.16).

2. Для цього режиму наведені нижче поля керуючих регістрів MOb мають бути ініціалізовані:

– IDT (регістри CANIDT1…CANIDT4) – 11-бітний або 29-бітний ідентифікатор;
– IDE (регістр CANCDMOB) – біт ознаки розширеного ідентифікатора;
– RTRTAG (регістр CANIDT4) – біт ознаки віддаленого запиту на передачу;
– DLC (регістр CANCDMOB) – розмір поля даних;
– RBnTAG (регістр CANIDT4) – резервні поля;
– MSG (регістри CANMSG, CANPAGE) – поле даних.

3. MOb, обраний запитом, готовий відразу відповісти без додаткових налаштувань, оскільки не треба заповнювати його CAN-буфер даних з кадру запиту. IDT-, IDE-, DLC- та деякі інші поля прийнятого кадру віддаленого запиту використовуються для автоматичної відповіді.

4. Коли передачу відповіді завершено, встановлюється біт TXOK регістра CANSTMOB, при цьому можливе переривання, якщо воно не замасковано.

5. Усі параметри і дані залишаються доступними в MOb до наступної ініціалізації.

Схему алгоритму роботи CAN-модуля у цьому режимі наведено у [3], а керуючу програму мовою С – у [7].

Прийом мультикадрів

Цей режим використовується для прийому в буфер відповідного набору кадрів – отримання мультикадрів. Керувати вхідними кадрами такого типу дозволяє наявність пріоритету між MOb. У цьому режимі налаштовується лише один набір MOb, включаючи непослідовні MOb. Коли всі MOb набору отримають належні кадри буде встановлено біт завершення прийому мультикадрів у буфер – BXOK регістра CANGIT, при цьому можливе переривання, якщо воно не замасковано.

Робота CAN-модуля в цьому режимі відбувається у наступній послідовності:

1. MOb мають бути ініціалізовані як MOb у Rx-конфігурації.

2. MOb будуть готові отримувати кадри даних, коли встановлено їх відповідну конфігурацію (табл. 8.16).

3. Коли ідентифікатор кадру отримано з CAN-мережі, CAN-модуль перебирає всі MOb, намагаючись знайти відповідний MOb з найбільшим пріоритетом.

4. IDT-, IDE- та DLC-поля відповідного MOb оновлюються значеннями з прийнятого кадру.

5. Як тільки прийом закінчено, байти даних отриманого повідомлення зберігаються (не для кадру віддаленого запиту) в буфері даних відповідного MOb і встановлюється біт RXOK регістра CANSTMOB. У цьому випадку можливе переривання, якщо воно не замасковано.

6. Коли прийом для останнього MOb з набору завершено, встановлюється біт завершення прийому кадрів у буфер BXOK (регістр CANGIT). У цьому випадку можливе переривання, якщо воно не замасковано.

7. Біт BXOK може бути очищено тоді, коли біти CONMOB0, 1 регістра CANCDMOB для всіх MOb набору було перезавантажено.

8. Усі параметри і дані залишаються доступними в MOb до наступної ініціалізації.

Схему алгоритму роботи CAN-модуля у цьому режимі наведено у [3], а керуючу програму мовою С – у [7].

8.3.3.5. Структура блока фільтрації

Одним з важливих вузлів CAN-модуля є блок фільтрації повідомлень (рис. 8.38).

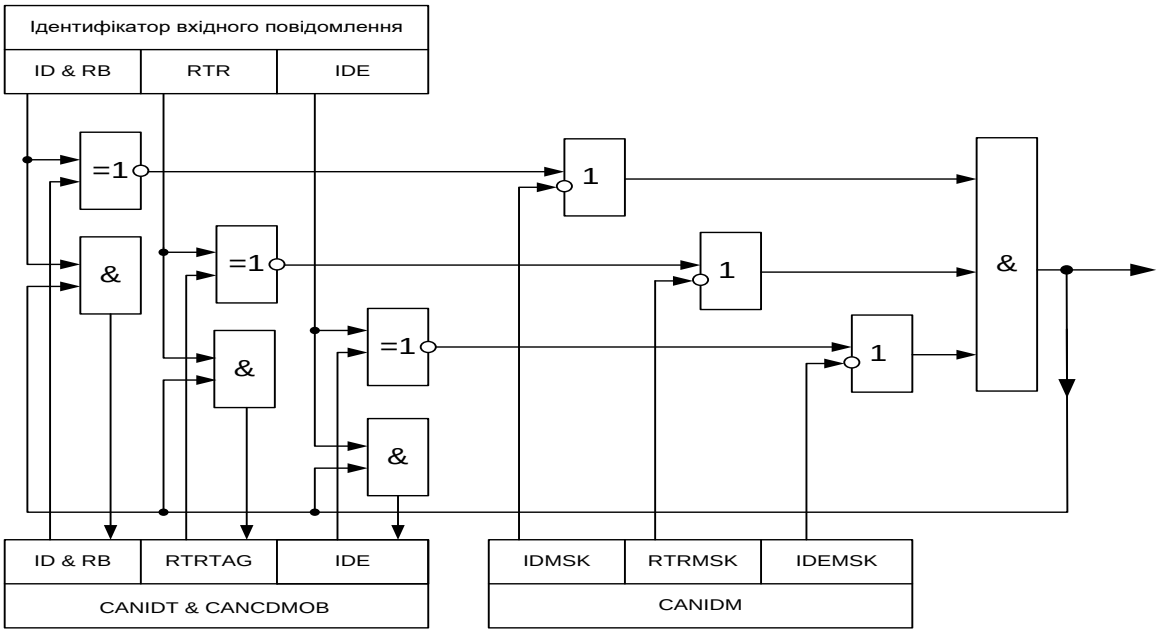


Рис. 8.38. Структура блока фільтрації повідомлень

У цьому блоці ідентифікатор вхідного повідомлення побітно підсумовується за модулем два з регістом фільтру, який складається з 13/32 бітів регістрів CANIDT1, CANIDT2, CANIDT3, CANIDT4 (залежно від формату кадру – версія 2.0 A/B) і біта IDE регістра CANCDMOB. Після цього результат інвертується і на нього накладається маска з регістрів масок ідентифікатора CANIDM1, CANIDM2, CANIDM3, CANIDM4. Якщо після

цих операцій отримуємо послідовність одиничних біт, то повідомлення відповідає вимогам фільтрації та записується у відповідний об’єкт повідомлення (MOb).

Маска застосовується для того, щоб вказати, які біти фільтра будуть використовуватися для перевірки ідентифікатора вхідного повідомлення. Якщо біт маски дорівнює нулю, то відповідний біт ідентифікатора повідомлення буде прийнято незалежно від значення біта фільтра.

Нижче наведено два приклади роботи блока фільтрації для прийому стандартних повідомлень.

Приклад 8.1. Повна фільтрація

Прийом тільки ID = 0x317:
ID MSK = 111 1111 1111 b;
ID TAG = 011 0001 0111 b.

Приклад 8.2. Часткова фільтрація

Прийом ID від 0x310 до 0x317:
ID MSK = 111 1111 1000 b;
ID TAG = 011 0001 0xxx b.

8.3.3.6. Структура переривань від CAN-модуля

CAN-модуль підтримує кілька різних джерел переривань. Підпрограма обробки CAN-переривання від різних джерел має адресу 0x0024, а обробки переривання від переповнення CAN-таймера – адресу 0x0026. Усі переривання визначаються індивідуальними бітами дозволу. Щоб дозволити переривання, відповідний біт дозволу треба встановити в одиницю разом з бітом глобального дозволу переривань у регістрі статусу.

У CAN-модулі МК AT90CAN128 можлива обробка переривань як для всіх повідомлень, так і для кожного конкретного об’єкта повідомлень (MOb).

Структуру переривань зображено на рис. 8.39.

Існують такі переривання:

- успішне отримання повідомлення;
- успішне відправлення повідомлення;
- виявлено помилку;
- заповнення буфера зберігання кадрів;
- встановлення стану «відключення від шини»;
- переповнення лічильника CAN-таймера.

На виході логічного елемента І, який розміщено у правій частині рисунку, формується «Запит на переривання» від різних джерел. Цей запит має адресу в пам’яті програм – 0x0024.

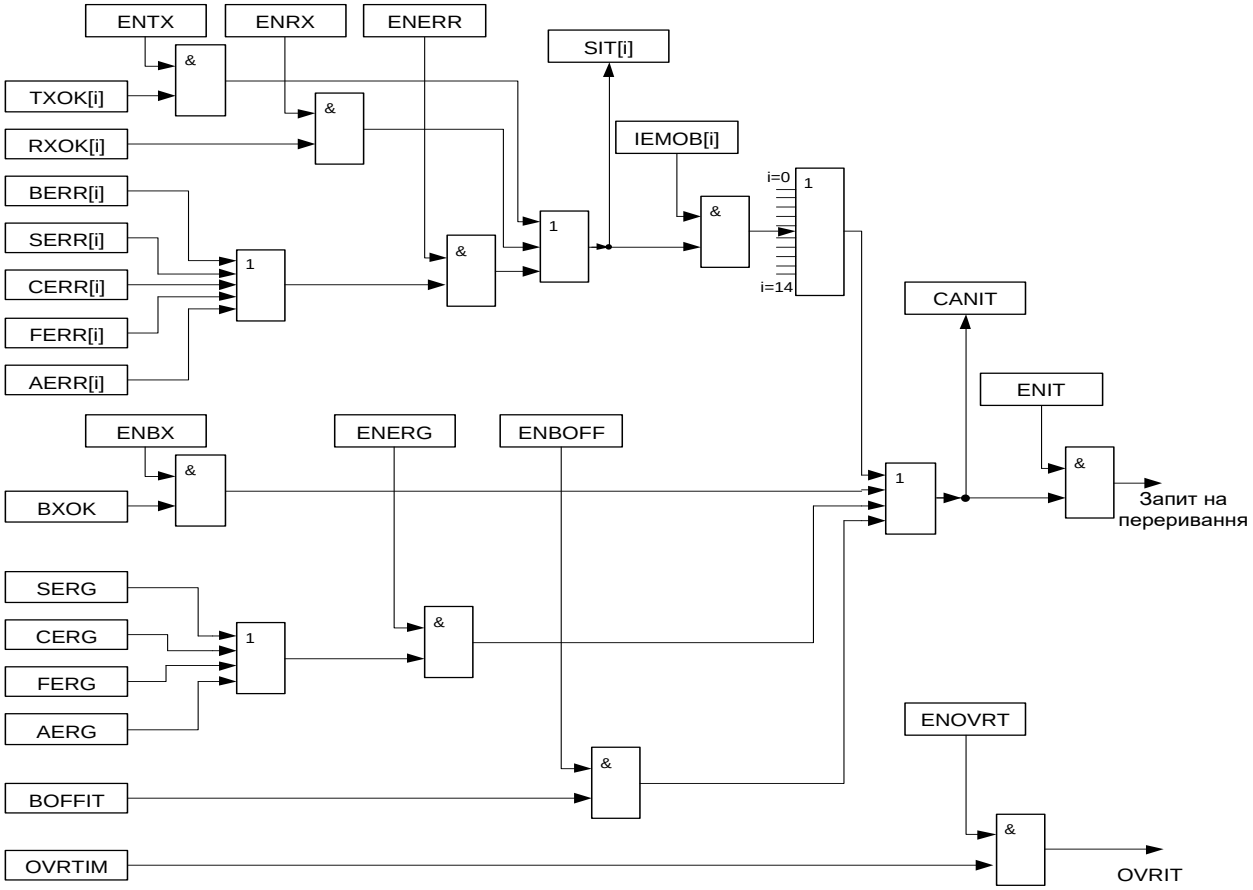


Рис. 8.39. Структура переривань МК AT90CAN128

Наведемо пояснення умовних позначень, які використано на рис. 8.39.

Регістр статусу об'єкта повідомлення – CANSTMOB:

- CANSTMOB.6 (TXOK) – успішна передача;
- CANSTMOB.5 (RXOK) – успішний прийом;
- CANSTMOB.4 (BERR) – помилка біта;
- CANSTMOB.3 (SERR) – помилка наповнення;
- CANSTMOB.2 (CERR) – помилка CRC;
- CANSTMOB.1 (FERR) – помилка форми;
- CANSTMOB.0 (AERR) – помилка підтвердження.

Загальний регістр переривань – CANGIT:

- CANGIT.7 (CANIT) – загальний прапорець переривань (відображає наявність будь-якого переривання крім OVRTIM);
- CANGIT.6 (BOFFIT) – прапорець переривання відключення від шини;
- CANGIT.5 (OVRTIM) – прапорець переповнення лічильника CAN-таймера;

– CANGIT.4 (BXOK) – прапорець отримання відповідного набору кадрів у буфер;
– CANGIT.3 (SERG) – прапорець помилки наповнення;
– CANGIT.2 (CERG) – прапорець CRC-помилки;
– CANGIT.1 (FERG) – прапорець помилки форми;
– CANGIT.0 (AERG) – прапорець помилки підтвердження.

Загальний регістр дозволу переривань – CANGIE:

– CANGIE.7 (ENIT) – дозвіл усіх переривань (крім переривання від переповнення CAN-таймера);
– CANGIE.6 (ENBOFF) – дозвіл переривання від'єднання від шини;
– CANGIE.5 (ENRX) – дозвіл переривання прийому;
– CANGIE4 (ENTX) – дозвіл переривання передачі;
– CANGIE.3 (ENERR) – дозвіл переривань від помилки MOb;
– CANGIE.2 (ENBX) – дозвіл переривання від отримання відповідного набору кадрів у буфер (BXOK);
– CANGIE.1 (ENERG) – дозвіл переривань від загальних помилок;
– CANGIE.0 (ENOVRT) – дозвіл переривання від переповнення лічильника CAN-таймера.

Регістри дозволу переривань від MOb – CANiE1 і CANIE2:

Відповідні біти (IEMOB[i]), які встановлено в одиницю, показують, для яких MOb дозволено переривання. Регістри мають 15 біт, що відповідає кількості об'єктів повідомлення.

Під час виникнення загального переривання встановлюється в одиницю відповідний прапорець (біт) у регістрі CANGIT. Якщо також в одиницю встановлено «дозволяючий біт» у регістрі CANGIE, який відповідає прапорцю загального переривання, то встановлюється в одиницю загальний прапорець переривань регістра CANGIT – CANGIT.7 (CANIT).

Під час виникнення переривання від одного з MOb встановлюється в одиницю відповідний біт у регістрі CANSTMOB. Встановлення в одиницю прапорця переривань CANIT у цьому випадку відбудеться тільки у разі встановленого «дозволяючого біта» для цього переривання в регістрі CANGIE (CANGIE.3; CANGIE.4; CANGIE.5) та встановленого біта, що дозволяє переривання від цього MOb, в у регістрах CANIE1 або CANIE2.

Для правильного визначення переривань від MOb та загальних переривань спочатку мають бути очищені відповідні біти регістрів CANSTMOB та CANGIT.

8.3.3.7. Структура блока CAN-таймера

CAN-модуль має блок таймера, структуру якого наведено на рис. 8.40.

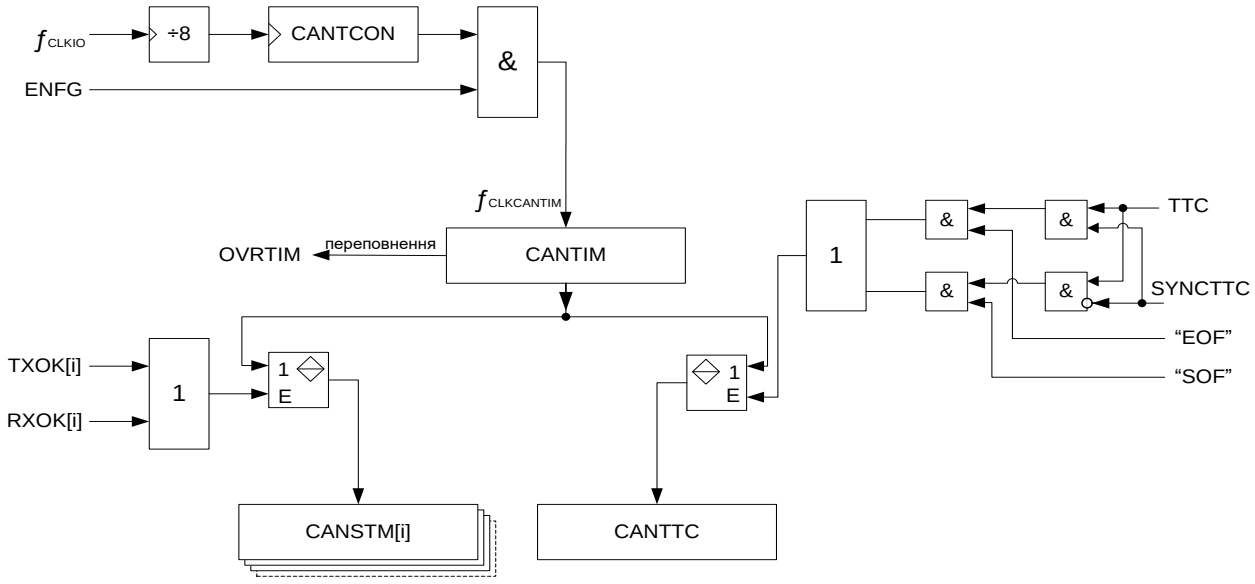


Рис. 8.40. Структура блока CAN-таймера

Основним елементом блока є 16-бітний лічильник – CANTIM. Під час встановлення в одиницю біта ENA/$\overline{\text{STB}}$ регістра CANGCON через деякий час CAN-модуль переходить в активний стан, що відображає встановлення в одиницю біта ENFG регістра CANGSTA. Після цього на вхід лічильника CANTIM починає надходити послідовність імпульсів тактової частоти від підсистеми введення/виведення МК. Ця частота ділиться апаратно на вісім, а потім проходить через 8-розрядний програмований дільник – CANTCON. У результаті роботи цих дільників період частоти на вході CANTIM встановлюється відповідно до такого виразу:

$$T_{CLKCANTIM} = T_{CLKIO} \cdot 8 \cdot \left(CANTCON[7:0]+1\right), \qquad (8.3)$$

де T_{CLKIO} – період тактової частоти підсистеми введення/виведення МК [3].

Під впливом цієї частоти CANTIM починає рахувати зі значення 0x0000. Коли відбувається його переповнення і перехід зі значення 0xFFFF на 0x0000, генерується переривання – OVRTIM = 1. Підпрограма обробки цього переривання має адресу 0x0026.

Кожен MOb має 16-розрядний регістр часових відміток – CANSTM, в якому за подіями: RXOK – успішний прийом або TXOK – успішна передача (регістр CANSTMOB) відбувається збереження поточного значення CANTIM.

CAN-мережі можлива взаємодія за стандартом TTCAN (Time-Triggered CAN – CAN, керована часом). Цей стандарт дозволяє

підвищити ефективність використання пропускної здатності шини та організувати взаємодію вузлів у мережі таким чином, що зникає невизначеність у часі передачі повідомлень, яка є у CAN. Наприклад, повідомлення з низьким пріоритетом може бути надіслано не з першого разу через втрату арбітражу вузлом, що намагається його передати і завчасно невідомо, чи це трапиться. В режимі TTCAN кожен вузол передає повідомлення у певний виділений для нього часовий інтервал, інші вузли в цей час не намагатимуться передавати дані. Це дозволяє уникнути колізій і втрати арбітражу. Також у цьому режимі передавач не намагається автоматично повторно передати повідомлення у разі виникнення помилки, оскільки тоді він буде використовувати не свій часовий інтервал, що у цьому режимі неприпустимо.

Режим TTC програмується встановленням в одиницю біта TTC регістра CANGCON. У цьому режимі кадр надсилається один раз, навіть якщо відбулася помилка.

Для TTC передбачено два режими синхронізації:

- синхронізація за початком кадру (SYNCTTC = 0, SOF = 1);
- синхронізація за кінцем кадру (SYNCTTC = 1, EOF = 1).

Якщо встановлено режим TTC (TTC = 1), а біт SYNCTTC = 0, то застосовується синхронізація за початком кадру – SOF (Start of Frame)) і у разі виявлення відповідного поля, яке складається з одного нульового біта, значення з регістра таймера CANTIM записується в регістр CANTTC.

Якщо встановлено режим TTC (TTC = 1), а біт SYNCTTC = 1, то застосовується синхронізація за кінцем кадру – EOF (End of Frame)) і у разі виявлення останнього одиничного біта поля «кінець кадру», яке складається з 7 одиничних біт, значення з регістра таймера CANTIM записується в регістр CANTTC.

Для того, щоб МК міг брати участь у роботі CAN-шини, де використовується режим TTCAN, необхідно додатково програмно реалізовувати відповідні алгоритми взаємодії [12].

8.3.3.8. Обробка помилок

Є п’ять різних типів помилок, які можуть виникати одночасно.

Помилка біта

Вузол, що передає дані на шину, одночасно її контролює. Помилка біта виникає, якщо значення біта на шині відрізняється від значення, що передається. Виняток становить поле арбітражу, поле підтвердження та передачі додаткового шостого біта протилежної полярності, після передачі п’яти біт однакової полярності.

Передавач, що посилає прапорець пасивної помилки і виявляє «домінантний» біт, не вважає, що це помилка біта.

Помилка заповнення

Помилка заповнення виникає, якщо під час прийому між початком повідомлення та роздільником CRC виявлено підряд 6 біт однакової полярності.

Помилка CRC

Послідовність CRC обчислюється передавачем і передається в кадрі повідомлення, а також обчислюється під час прийому повідомлення. Помилка CRC виникає, якщо значення, що обчислено приймачем, не збігається із прийнятим від передавача.

Помилка форми

Помилка форми виявляється, якщо бітове поле фіксованого формату (кінець повідомлення, роздільники CRC і підтвердження) містять один або більше заборонених бітів. «Домінантний» біт протягом останнього біта «кінець кадру» не розглядається як помилка форми.

Помилка підтвердження

Помилка підтвердження виявляється передавачем щоразу, коли він не виявляє «домінантний» біт в області підтвердження, що свідчить про те, що жоден з вузлів не одержав повідомлення правильно, при цьому повідомлення передається повторно.

Лічильники помилок

У CAN-модулі для підрахунку помилок використовуються 8-розрядні лічильники помилок: CANTEC – лічильник помилок передачі та CANREC – лічильник помилок прийому.

Залежно від кількості помилок CAN-модуль може перебувати в одному з трьох станів:

- активна помилка;
- пасивна помилка;
- відключення від шини (рис. 8.41).

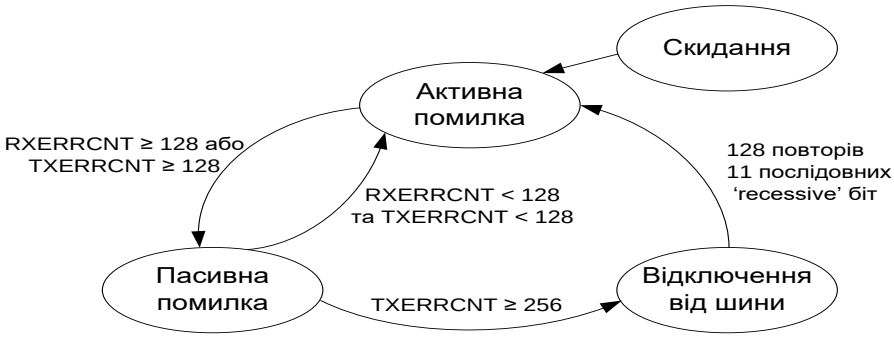


Рис. 8.41. Стан лічильників помилок

Під час обміну CAN-мережею стан лічильників змінюється згідно з правилами, які описано в [3; 7].

8.3.3.9. Бітова синхронізація

Номінальна швидкість передачі та номінальна тривалість біта

Номінальна швидкість передачі інформації визначається кількістю бітів у секунду, які передаються ідеальним передавачем за відсутності пересинхронізації [3].

Номінальний час передачі біта становить 1/номінальну швидкість передачі інформації в бітах у секунду.

Номінальний час передачі біта розділяють на кілька ділянок, що не перекриваються (рис. 8.42):

- сегмент синхронізації (SYNC_SEG);
- сегмент часу розповсюдження (PROP_SEG);
- фазовий сегмент 1 (PHASE_SEG1);
- фазовий сегмент 2 (PHASE_SEG2).

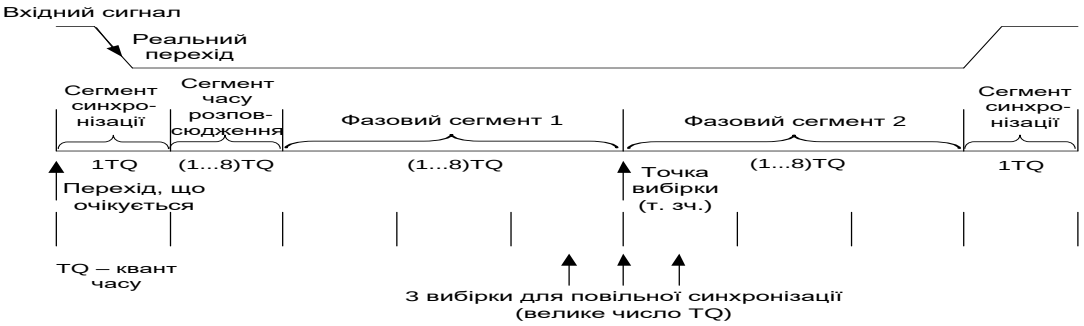


Рис. 8.42. Номінальний час біта

Кожен сегмент складається із цілого числа відрізків часу, які називаються квантами часу – TQ.

Номінальний час передачі біта коливається від 8 TQ до 25 TQ (включно). За швидкості передачі 1 Мбіт/с мінімальний номінальний час передачі біта дорівнює 1 мкс.

Тривалість одного кванта часу – T_{SCL} (T_{TQ}) визначається тактовою частотою підсистеми введення/виведення МК і значенням коефіцієнта ділення цієї частоти, що може змінюватися програмно:

$$T_{SCL} = (BRP[5:0] + 1) \cdot T_{CLKI/O}, \quad (8.4)$$

де $T_{CLKI/O}$ – період тактової частоти підсистеми введення/виведення МК; BRP[5:0] – десятковий еквівалент шести біт регістра CANBT1.

Сегмент синхронізації SYNC_SEG

Сегмент синхронізації SYNC_SEG має довжину один квант часу – TQ та використовується для синхронізації різних вузлів на шині. Початок передачі визначає від'ємний фронт вхідного сигналу, що повинен перебувати в межах сегмента синхронізації.

Сегмент часу розповсюдження

Сегмент часу розповсюдження використовується, щоб компенсувати час фізичного запізнювання в мережі. Цей час дорівнює подвоєній сумі часу розповсюдження сигналу на лінії шини, затримки вхідного компаратора і затримки вихідного формувача CAN-вузла. Значення цього сегмента може програмуватися від 1 TQ до 8 TQ.

Фазові сегменти 1, 2

Ці сегменти використовуються, щоб компенсувати помилки фази (фазові зсуви) під час прийому. Фазовий сегмент 1 може бути подовжено, а фазовий 2 – вкорочено пересинхронізацією (ресинхронізацією) [3; 7].

Фазові сегменти призначено для оптимального розміщення точки вибірки отриманого біта у межах часу біта, що передається. Точка вибірки розміщується між фазовим сегментом 1 і фазовим сегментом 2 (рис. 8.42).

Фазовий сегмент 1 визначає точку вибірки в межах біта, що передається. Фазовий сегмент 2 забезпечує затримку до початку наступного біта. Тривалість обох цих сегментів може програмуватися від 1 TQ до 8 TQ.

Точка вибірки

Точка вибірки (точка зчитування), (Sample point) – це момент часу, за якого рівень на шині читається та інтерпретується, як значення відповідного біта. Вона розміщена наприкінці фазового сегмента 1 (див. рис. 8.42).

Якщо біт має значну кількість TQ, то можна задати багаторазовий вибірковий контроль стану шини. В цьому випадку CAN-модуль з періодичністю TQ/2 робить вибірку три рази для кожного прийнятого біта. Правильним вважається значення, яке отримано за мажоритарним принципом – два або три рази прийнято нуль або одиницю. Увімкнення

багаторазової вибірки здійснюється встановленням спеціального програмованого біта SMP регістра CANBT3.

Синхронізація

Для компенсації зсуву фази між частотами генераторів різних вузлів шини та під час зміни ведучого при арбітражі, кожен CAN-модуль повинен синхронізуватися до переходу рівня вхідного сигналу з одиниці на нуль (рис. 8.42). Коли перехід виявлено, то схема синхронізації порівнює розміщення переходу з переходом, що очікується під час прийому, та виконує налаштування значень фазових сегментів 1 та 2.

Є два механізми синхронізації:

- апаратна синхронізація;
- синхронізація з відновленням тактових інтервалів (пересинхронізація/ресинхронізація).

Апаратна синхронізація

Апаратна синхронізація виконується під час переходу від «рецесивного» до «домінантного» біта протягом холостого стану шини, що вказує на початок повідомлення. Під час апаратної (жорсткої) синхронізації тактові інтервали (тривалість сегментів) не змінюються протягом усього повідомлення.

Після апаратної синхронізації внутрішній бітовий час кожного вузла перезапускається з сегмента синхронізації SYNC_SEG (рис. 8.42).

Синхронізація з відновленням тактових інтервалів

Синхронізація з відновленням тактових інтервалів (пересинхронізація/ресинхронізація) призначена для зменшення фазових спотворень під час прийому та виконується автоматичним подовженням фазового сегмента 1 або скороченням фазового сегмента 2. Максимальне значення зміни фазових сегментів коливається в межах від 1 TQ до 4 TQ. Синхронізація виконується під час переходів від «рецесивного» до «домінантного» біта протягом прийому кадру. Фіксоване значення максимальної кількості послідовних бітів однакової полярності («біт-стаффінг») гарантує своєчасне відновлення синхронізації. В межах бітового інтервалу допускається тільки один тип пересинхронізації. Фронт сигналу буде використовуватися для пересинхронізації тільки тоді, якщо значення, яке виявлено під час попередньої точки зчитування (попереднє значення на шині), відрізняється від значення на шині відразу після фронту.

Фазове спотворення (помилка фази) «e» визначається в квантах TQ як різниця часу точки зчитування бітового інтервалу приймача, протягом якого

з’явився перехід вхідного сигналу з одиниці в нуль, та часу появи цього переходу – $t_{\downarrow}$ (рис. 8.42).

Значення та знак «e» визначається в такий спосіб:

- e = 0, якщо фронт вхідного сигналу (перехід з одиниці в нуль) перебуває в межах сегмента синхронізації SYNC_SEG (ресинхронізація не проводиться);
- e > 0, якщо фронт вхідного сигналу (перехід з одиниці в нуль) не перебуває в межах SYNC_SEG та знаходиться перед точкою зчитування бітового інтервалу приймача, протягом якого з’явився перехід вхідного сигналу з одиниці в нуль;
- e < 0, якщо фронт сигналу (перехід з одиниці в нуль) не перебуває в межах SYNC_SEG та знаходиться після точки зчитування бітового інтервалу приймача, протягом якого з’явився перехід вхідного сигналу з одиниці в нуль.

Ефект від пересинхронізації такий самий як і у випадку апаратної синхронізації, коли величина *помилки фази «e»* менше або дорівнює значенню ширини періоду пересинхронізації, що програмується.

Коли величина *помилки фази «e»* більша, ніж ширина переходу пересинхронізації, що програмується, тоді:

- якщо *помилка фази «e»* додатна, то фазовий сегмент 1 подовжується на ширину періоду пересинхронізації, що програмується;
- якщо *помилка фази «e»* від’ємна, то фазовий сегмент 2 скорочується на ширину періоду пересинхронізації, що програмується.

На рис. 8.43, 8.44 наведено два приклади пересинхронізації.

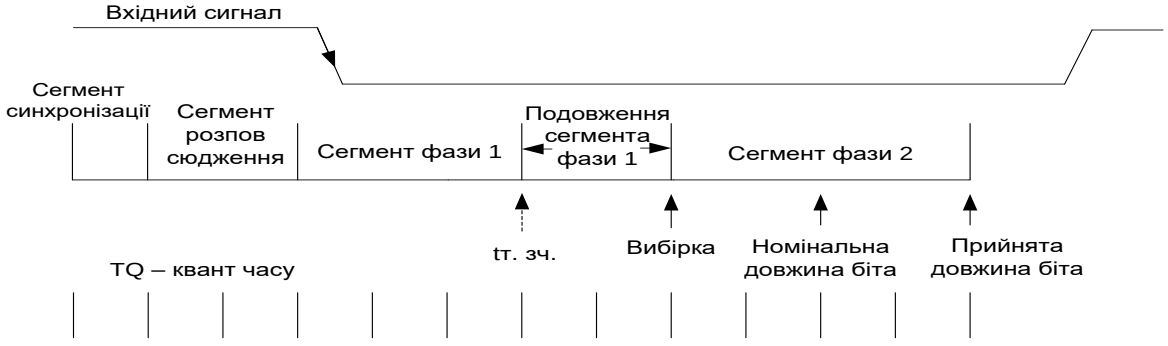


Рис. 8.43. Синхронізація з відновленням тактових інтервалів завдяки автоматичному подовженню фазового сегмента 1

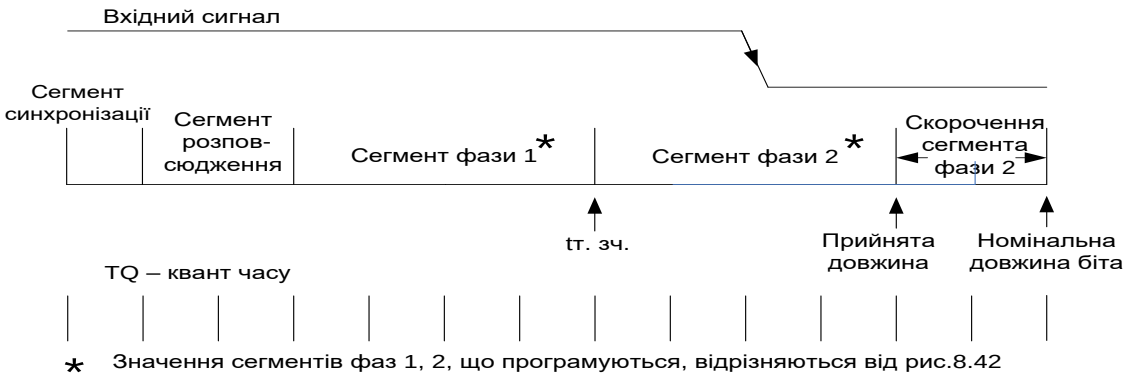


Рис. 8.44. Синхронізація з відновленням тактових інтервалів завдяки автоматичному скороченню фазового сегмента 2

Приклад програмування швидкості обміну інформацією CAN-мережею

Зміст завдання

Написати мовою Асемблера фрагмент програми, який забезпечує потрібну швидкість обміну інформацією CAN-мережею для AVR-мікроконтролера AT90CAN128, розрахувати тривалість одного біта $t_{\text{біта}}$ та тривалість одного кванта часу $T_{SCL} = T_{TQ}$.

Вихідні дані:

- тактова частота підсистеми введення/виведення МК: $f_{CLKIO} = 10$ МГц;
- довжина періоду ресинхронізації: $t_{\text{п.р.}} = T_{SJW} = 2T_{SCL}$;
- тривалість сегменту часу розповсюдження: $t_{\text{с.р.}} = T_{PRS} = 6T_{SCL}$;
- довжина фазового сегмента 2: $T_{PHS2} = 5T_{SCL}$;
- довжина фазового сегмента 1: $T_{PHS1} = 4T_{SCL}$;
- використовувати одноразову вибірку під час прийому кожного біта;
- швидкість обміну $V_{\text{пд}} = 10$ Кбіт/с.

Розв’язання завдання

Розрахуємо тривалість одного біта: $t_{\text{біта}} = \frac{1}{V_{\text{пд}}} = \frac{1}{10000} = 100$ мкс.

Згідно з рис. 8.42 та вихідними даними, номінальний час одного біта має таку кількість квантів часу:

$$n_{\text{кв.ч}} = 1TQ + 6TQ + 5TQ + 4TQ = 16TQ = 16T_{SCL}.$$

Тоді потрібна тривалість одного кванта часу:

$$T_{SCL} = \frac{\text{t}_{\text{біта}}}{\text{n}_{\text{кв.ч}}} = \frac{100}{16} = 6{,}25 \text{ мкс}.$$

Відповідно до формули (8.4) розраховуємо потрібне значення:

$$\text{BRP}[5:0] = \frac{T_{SCL}}{T_{CLKIO}} - 1 = \frac{6{,}25}{0{,}1} - 1 = 62{,}5 - 1 = 61{,}5,$$

де $T_{CLKIO} = \frac{1}{f_{CLKIO}} = \frac{1}{10*10^6\ \text{Гц}} = 0{,}1$ мкс.

Приймаємо BRP[5: 0] = 61 = 111101b.

Нижче наведено керуюче слово КС1 (8 біт), яке потрібно завантажити у регістр CANBT1 для отримання розрахованого вище значення BRP[5: 0]:

7р	6р	5р	4р	3р	2р	1р	0р
–	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	–
0	1	1	1	1	0	1	0

Для програмування потрібних значень $t_{п.р.} = T_{SJM}$; $t_{с.р.} = T_{PRS}$; $t_{фс2} = T_{PHS2}$ та $t_{фс1} = T_{PHS1}$ використовуються регістри CANBT2 та CANBT3 [3; 7], в які треба завантажити керуючі слова КС2 та КС3, які наведено нижче.

КС2:

7р	6р	5р	4р	3р	2р	1р	0р
–	SJW1	SJW0	–	PRS2	PRS1	PRS0	–
0	0	1	0	1	0	1	0

КС3:

7р	6р	5р	4р	3р	2р	1р	0р
–	PHS22	PHS21	PHS20	PHS12	PHS11	PHS10	SMP
0	1	0	0	0	1	1	0

Під час формування КС2 та КС3 треба враховувати, що потрібні значення T_{SJM}, T_{PRS}, T_{PHS2} та T_{PHS1} зв'язані з відповідними розрядами регістрів CANBT2 та CANBT3 виразами, які наведено нижче [3]:

$$T_{SJW} = T_{SCL}(\text{SJW}[1:0] + 1) = \text{t}_{\text{п.р.}};$$

$$T_{PRS} = T_{SCL}(\text{PRS}[2:0] + 1) = t_{с.р.};$$

$$T_{PHS2} = T_{SCL}(\text{PHS2}[2:0] + 1) = \text{t}_{\text{фс2}};$$

$$T_{PHS1} = T_{SCL}(\text{PHS1}[2:0] + 1) = \text{t}_{\text{фс1}}.$$

Біт SMP має значення нуль, тому що в завданні було вказано, що треба використовувати одноразову вибірку під час прийому кожного біта. Якщо в завданні буде вказано, що значення кожного біта під час прийому зчитується тричі, один раз в точці вибірки і по одному разу на відстані 1/2 тривалості T_{SCL}, тоді біт SMP треба встановити у одиницю.

Згідно з [3] регістри CANBT1, CANBT2 та CANBT3 мають відповідно такі адреси у просторі адрес регістрів введення/виведення МК AT90CAN128: $00E2; $00E3 та $00E4.

Нижче наведено фрагменти програми мовою Асемблера, які програмують потрібну швидкість обміну CAN-мережею.

Програмування регістра CANBT1:

```
LDI  R18, $7A;      R18 <- KC1=01111010b=$7A;
LDI  R27, $00;      R27 <- старший байт адреси  CANBT1;
LDI  R26, $E2;      R26 <- молодший байт адреси CANBT1;
ST  X, R18;         CANBT1 <- R18=$7A.
```

Програмування регістра CANBT2:

```
LDI  R17, $2A;      R17 <- KC2=00001000b=$2A;
LDI  R27,$00;       R27 <- старший байт адреси  CANBT2;
LDI  R26, $E3;      R26 <- молодший байт адреси  CANBT2;
ST  X, R17;         CANBT2 <- R17=$2A.
```

Програмування регістра CANBT3:

```
LDI  R16, $46;      R16 <- KC3=01001000b=$46;
LDI  R27, $00;      R27 <- старший байт адреси  CANBT3;
LDI  R26, $E4;      R26 <- молодший байт адреси  CANBT3;
ST  X, R16;         CANBT3 <- R16=$46.
```

8.3.3.10. Фізичний рівень CAN-протоколу

Загальні відомості про фізичний рівень

Фізичний рівень (Physical Layer) CAN-протоколу визначає опір кабелю, рівень електричних сигналів у мережі і т. ін. Існує кілька фізичних рівнів CAN-протоколу: ISO 11898, ISO 11519, SAEJ2411 і т. ін. [3; 7]. Серед останніх часто використовується фізичний рівень, який описано у стандарті ISO 11898. Цей рівень в якості середовища передачі визначає двопровідну диференціальну лінію з резисторами-термінаторами на кінцях зі значенням 120 Ом. Допускається коливання цього значення в межах від 108 Ом до 132 Ом.

Швидкість передачі CAN-мережею залежить від довжини кабелю. Це пов’язано з кінцевою швидкістю світла та механізмом побітового арбітражу. Під час останнього всі вузли мережі повинні отримувати поточний переданий біт практично одночасно, тобто сигнал в мережі повинен встигнути поширитися по всьому кабелю за один квант часу.

Залежність між швидкістю передачі та максимальною довжиною кабелю відображено у табл. 8.17.

Таблиця 8.17. **Залежність швидкості передачі даних від довжини шини**

Швидкість передачі	Час передачі біта	Довжина лінії зв’язку
1 Мбіт/с	1 мкс	30 м
800 Кбіт/с	1,25 мкс	50 м
500 Кбіт/с	2 мкс	100 м
250 Кбіт/с	4 мкс	250 м
125 Кбіт/с	8 мкс	500 м
62,5 Кбіт/с	20 мкс	1000 м
20 Кбіт/с	50 мкс	2500 м
10 Кбіт/с	100 мкс	5000 м

За довжини кабелю 30 метрів швидкість передачі даних CAN-мережею буде максимальною та дорівнює 1 Мбіт/с.

У разі необхідності передавати дані на більші відстані рекомендовано використовувати вузли-ретранслятори.

Трансивери для CAN-мережі

Для організації дротового з’єднання на CAN-шині найбільш широке поширення отримали два типи приймачів/передавачів (трансиверів): «High Speed» (ISO 11898-2) та «Fault Tolerant» [3; 13–15].

Трансивери, які виконано відповідно до стандарту «High Speed» (ISO 11898-2), найбільш прості, дешеві та надають можливість передавати дані зі швидкістю до 1 Мбіт/с. «Fault Tolerant» приймачі/передавачі не дуже чутливі до ушкоджень на шині та дозволяють побудувати високонадійну мережу, що мало споживає, зі швидкостями передачі даних не вище 125 Кбіт/с.

Організація фізичного середовища передачі даних повинна задовольняти обов'язкові вимоги, які вимагають способи арбітражу та синхронізації для CAN-інтерфейсу. Фізичне середовище передачі даних повинно мати можливість перебувати у двох станах – «рецесивному» та «домінантному». CAN-шина перебуває в «рецесивному» стані, якщо жоден вузол не передає «домінантний» біт, і в «домінантному» стані, якщо хоча б один вузол встановив «домінантний» біт. Інакше кажучи, на шині не повинен відбуватися конфлікт, якщо одночасно кілька пристроїв виставили різні логічні рівні. На шині в цьому випадку повинен встановлюватися «домінантний» стан.

Максимальна кількість вузлів, підключених до шини, фактично визначається здатністю навантаження застосованих трансиверів. Наприклад, під час використання мікросхеми PCA82C250 від компанії NXP (цю мікросхему було розроблено також компанією Philips Semiconductors) воно дорівнює 110. Цей трансивер належить до стандарту «High Speed» (ISO 11898-2) та широко використовується у CAN-мережах.

Прикладом такого трансивера є мікросхема PCA82C250. Її з'єднання із CAN-шиною показано на рис. 8.45.

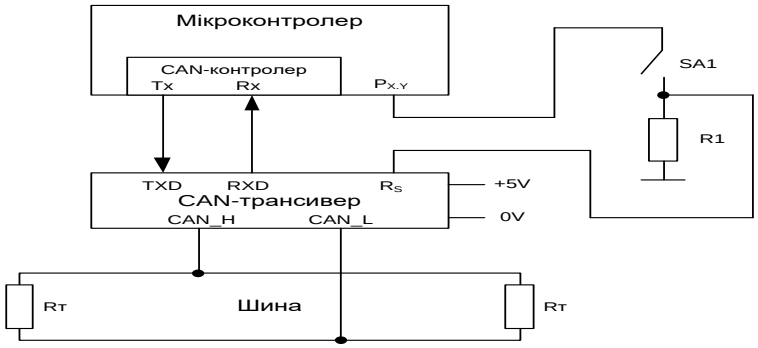


Рис. 8.45. З'єднання із шиною «High Speed»-трансивера

Напруга живлення цього трансивера становить 5 В. Рівні напруг, що відповідають «рецесивному» й «домінантному» для стандарту ISO 11898-2 показано на рис. 8.46.

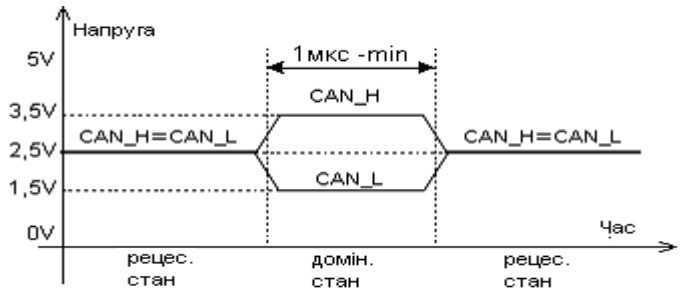


Рис. 8.46. Фізичні рівні «High Speed»-трансиверів

Логічному нулю відповідає високий рівень сигналу (приблизно 3,5 В) на лінії CAN-H і низький рівень (приблизно 1,5 В) на лінії CAN-L, тобто між лініями CAN-H і CAN-L присутня різниця потенціалів приблизно 2 В. Логічній одиниці відповідають рівні напруги (приблизно 2,5 В) на обох лініях, тобто між лініями CAN-H і CAN-L присутня різниця потенціалів приблизно 0 В.

Є три режими роботи трансивера, які налаштовуються ззовні через вивід Rs (рис. 8.45):

1. Високошвидкісний режим – High Speed.
2. Режим регулювання швидкості наростання сигналу, що передається в CAN-шину – Slope-Control.
3. Режим очікування – Standby.

Високошвидкісний режим реалізується підключенням виводу Rs до V_{SS} – 0 В. У цьому режимі вихідні драйвери мають швидкий час наростання і спаду, що забезпечує швидкість передачі до 1 Мбіт/с.

Якщо треба зменшити випромінювані драйвером електромагнітні завади, PCA82C250 можна встановити в режим регулювання швидкості наростання сигналу, що передається в CAN-шину. Для цього треба підключити резистор R1 від виводу Rs на спільний мінус [7]. Зменшення швидкості наростання вихідної напруги призводить до зменшення швидкості передачі даних CAN при заданій довжині шини, або до скорочення довжини шини при заданій швидкості передачі даних.

Режим очікування (режим сну – Sleep) встановлюється під'єднанням виводу Rs до V_{DD} – +5 В. На рис. 8.45 для цього використано перемикач SA1 та лінія $P_{X.Y}$ одного із вільних портів введення/виведення МК. У режимі сну передавач від'єднано, а приймач працює в режимі зниженого енергоспоживання. Вивід RxD, який приймає, як і раніше функціонує, але на нижчій швидкості.

Режим очікування можна використовувати для встановлення пристрою в режим низького енергоспоживання і від'єднання передавача у випадку, якщо CAN-модуль несправний і видає на шину непередбачувані дані.

Інший тип трансивера – «Fault Tolerant» орієнтовано на застосування в легкових автомобілях, де не потрібна висока швидкість передачі даних, але

велике значення мають такі якості шини, як висока надійність і мінімальне енергоспоживання. Трансивер «Fault Tolerant» може бути виконано, наприклад, на мікросхемі Motorola MC33388 [3; 15].

Роз’єми для CAN-мережі

Міжнародна організація CiA (The CAN in Automation international users and manufacturers group) рекомендує у своєму стандарті CiA DS-102 для з’єднання окремих вузлів з CAN-шиною використовувати наведену на рис. 8.47 розводку виводів 9-контактного роз’єму D-Sub.

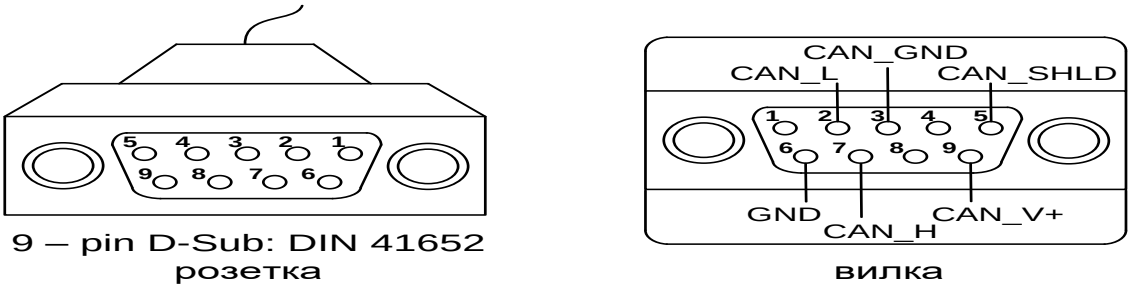


Рис. 8.47. CAN-роз’єм згідно зі стандартом CiA DS-102

Цю угоду використовують деякі протоколи більш високих рівнів CAN (CANopen, Smart Distributed System).

Призначення виводів CAN-роз’єму згідно зі стандартом CiADS-102 наведено у табл. 8.18.

Таблиця 8.18. **Призначення виводів CAN-роз’єму згідно зі стандартом CiA DS-102**

Контакт	Ланцюг	Примітка
1	–	–
2	CAN_L	Низький рівень – домінантний стан
3	CAN_GND	Земля
4	–	–
5	(CAN_SHLD)	Екран (необов’язковий)
6	GND	Земля (необов’язковий)
7	CAN_H	Високий рівень – рецесивний стан
8	–	–
9	(CAN_V+)	Живлення (необов’язковий)

Принципова схема вузла CAN-мережі

Принципову схему вузла CAN-мережі наведено у [3]. Основним елементом схеми є МК AT90CAN128, який містить CAN-модуль, що підтримує протокол CAN 2.0A/B.

Контрольні запитання та завдання

1. Дайте характеристику CAN-протоколу. Назвіть його основні переваги.
2. Опишіть структуру CAN-мережі.
3. Який завадостійкий код використовується у CAN-мережі? Дайте характеристику цьому коду.
4. Опишіть схему мікропроцесорної системи керування із використанням CAN-мережі.
5. Опишіть еталонну модель ISO/OSI CAN-шини.
6. Поясніть основні характеристики CAN-модуля.
7. Перерахуйте типи помилок, що можуть бути виявлені під час передачі CAN-мережею. Наведіть механізми виявлення помилок.
8. Назвіть рівні сигналів під час передачі CAN-шиною.
9. Дайте визначення поняттю «фрейм». Типи фреймів. Формати фреймів.
10. Опишіть структури стандартного та розширеного повідомлень.
11. Як визначається номінальний час передачі біта?
12. Для чого потрібна синхронізація? Назвіть та опишіть види синхронізації.
13. Наведіть основні характеристики CAN-модуля МК AT90CAN128.
14. Опишіть структуру CAN-модуля МК AT90CAN128.
15. Дайте характеристику режимів роботи CAN-модуля.
16. Опишіть послідовність програмування режиму передачі кадру (повідомлення) або віддаленого запиту (Tx-конфігурація).
17. Опишіть послідовність програмування режиму прийому кадру даних або віддаленого запиту (Rx-конфігурація).
18. Опишіть програмування режиму прийому мультикадрів у буфер.
19. Опишіть структуру блока фільтрації повідомлень.
20. Перерахуйте переривання від CAN-модуля. Опишіть структуру переривань.
21. Опишіть структуру блока таймера CAN-модуля.
22. Опишіть організацію регістрів CAN-модуля.
23. Дайте характеристику фізичному рівню (Physical Layer) CAN-протоколу.
24. Вкажіть максимальну швидкість передачі даних CAN-мережею. Від чого вона залежить?
25. Опишіть призначення виводів CAN-роз’єму.
26. Наведіть та поясніть схему з’єднання «High Speed» трансиверів із шиною.
27. Опишіть принципову схему вузла CAN-мережі.
28. Опишіть алгоритми роботи вузла CAN-мережі в різних режимах.

8.4. Мережа RS-485

8.4.1. Особливості архітектури мережі RS-485

8.4.1.1. Загальна характеристика мережі

У мережах RS-485 використовується стандарт передачі даних по двопровідному напівдуплексному багатоточковому послідовному каналу зв'язку – RS-485/EIA-485 (RS485 – англ. Recommended Standard 485, EIA-485 – англ. Electronic Industries Alliance-485).

Стандарт RS-485 розроблений спільно двома асоціаціями: Асоціацією електронної промисловості (EIA – Electronics Industries Association) та Асоціацією промисловості засобів зв'язку (TIA – Telecommunications Industry Association). Раніше EIA маркувала всі свої стандарти префіксом «RS» (англ. Recommended Standard – рекомендований стандарт). Багато інженерів продовжують використовувати це позначення, проте EIA/TIA офіційно замінив «RS» на «EIA/TIA» з метою полегшити ідентифікацію походження своїх стандартів. Нині різні розширення стандарту RS-485 охоплюють широке розмаїття програм. Цей стандарт став основою для створення цілого сімейства обчислювальних мереж, які широко використовуються в промисловій автоматизації.

У стандарті RS-485 для передачі і прийому даних часто використовується єдина вита пара проводів. Передача даних здійснюється за допомогою диференціальних сигналів. Різниця напруг між проводами однієї полярності означає логічну одиницю, різниця іншої полярності – нуль.

Стандарт RS-485 це назва популярного інтерфейсу, що використовується в промислових АСК ТП для з'єднання контролерів та іншого обладнання. Головна відмінність RS-485 від також широко розповсюдженого RS-232 – можливість об'єднання декількох пристроїв у мережу.

Стандартом RS-485 описується використаний в інтерфейсі спосіб передачі диференціальних сигналів у мікроконтролерній мережі і визначає характеристики лінійних формувачів і приймачів. Таким чином, інтерфейс RS-485 передає і приймає лише послідовності логічних рівнів, не піклуючись про логічні подробиці передачі даних у вигляді поділу на байти, видачі старт-стопних сигналів, корекції помилок, керування напрямком і черговістю передачі даних. Усе це має бути реалізовано окремо, але саме тому на RS-485 можна виконати безліч інтерфейсних протоколів – від найпростіших типу «точка–точка» до складних інтелектуальних інтерфейсів розподілених систем.

Для передачі сигналів формувач передавача генерує дві комплементарні напруги (логічно протилежних: високий рівень на виході A, низький рівень на виході B, або високий рівень на B, низький рівень на A). Усі інші передавачі у цей час, щоб уникнути конфлікту і спотворення сигналу перебувають у третьому (високоімпедансному) стані [3; 6].

На рис. 8.48 показано, як стандарт EIA-RS-485 визначає напруги U0a, U0b і U0.

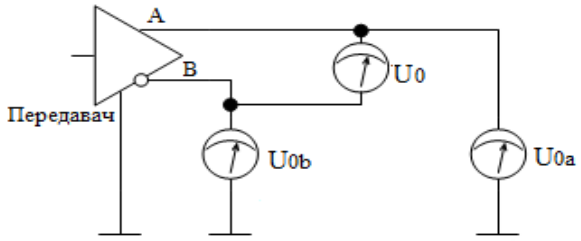
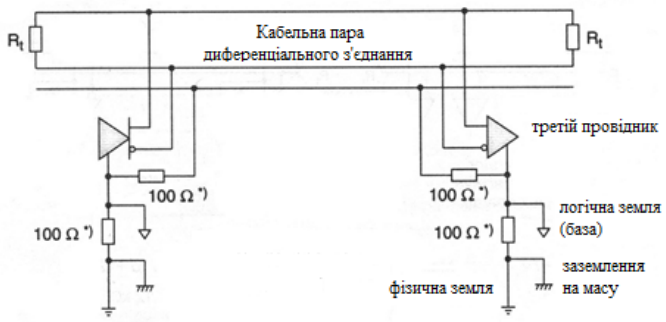


Рис. 8.48. Вимірювання напруг U0a, U0b і U0

Коли рівень напруги U0a – низький, тоді U0b – високий, а коли U0a – високий, тоді U0b – низький. Незважаючи на те, що на схемах включення RS-485 вказується двопровідна вита пара, RS-485 не є струмовою петлею з взаємно-протилежними струмами. Є інтерфейс «current loop» (струмова петля), в якому логічну 1 подано імпульсом струму в петлі, а логічний 0 – відсутністю струму. Фактично це дві струмові петлі, два електричні контури. Формувачі і приймачі мережі RS-485 повинні використовувати спільну землю, тому термін «двопровідна лінія» непридатний до RS-485.

У загальному випадку, необхідно використовувати трипровідну лінію (рис. 8.49).



Примітка. *) – резистори, включені послідовно з лініями повернення сигналу, обмежують циркулюючі струми.

Рис. 8.49. Схема підключення інтерфейсу RS-485/ EIA-485 до трипровідної лінії

Третім проводом може бути високоякісне реальне заземлення з низьким власним опором. Іноді рекомендують логічну «землю» заземлювати на реальну землю (на шасі) через резистор у 100 Ом для обмеження струму.

Правильна робота передавача і приймача вимагає наявності шляху повернення сигналу між підключеними до ланцюга землями обладнання в кожній точці підключення.

Приймачі мережі розроблені так, щоб реагувати на різницю напруг U_0 = A – B, більшу ніж 200 Мв (напруги вимірюються щодо спільного потенціалу (землі)). За досить довгих лініях зв'язку та наявності струму через третій провід (за різних значень падіння напруги в локальних заземлюючих ланцюгах) можлива поява значного постійного потенціалу, який додається до напруги в сигнальних проводах A і B. Саме тому приймачі RS-485 працюють у діапазоні напруг: –7…+12 В, а не 0…+5 В (діапазон вихідної напруги передавача). Таким чином допускається наявність неузгодженості локальних потенціалів землі до 7 В.

Загальні рекомендації щодо використання RS-485 наведено у [3; 6].

8.4.1.2. Кількість вузлів мережі

Стандартом EIA RS-485 визначається приймач з питомим навантаженням (Unit-Load) на передавач в один UL, при цьому передавач повинен формувати 32 UL, звідки випливає, що максимальна кількість вузлів у мережі не може перевищувати 32/1 = 32. Однак уже є приймачі з питомим навантаженням в 1/4 UL і навіть 1/8 UL, що робить можливим створення мережі з 256 вузлів. Під час використання повторювачів (ретрансляторів) можна з'єднувати разом складні мережі, об'єднуючи фактично необмежену кількість вузлів, але одночасно зі зростанням діаметра топології мережі збільшуються і затримки, а швидкість передачі даних може стати неприйнятно низькою.

8.4.1.3. Швидкість та дальність передачі даних

Мережа RS-485 забезпечує передачу даних зі швидкістю до 10 Мбіт/с. Максимальна дальність залежить від швидкості. Наприклад, при швидкості 10 Мбіт/с максимальна довжина лінії – 120 м, а при швидкості 100 Кбіт/с – 1200 м.

8.4.1.4. Протоколи та роз'єми для передачі

Стандарт не нормує формат інформаційних кадрів і протокол обміну. Найбільш часто для передачі байтів даних використовуються протоколи, що і в інтерфейсі RS-232: стартовий біт, біти даних, біт паритету (якщо потрібно), стоповий біт. Тобто у мікроконтролерній мережі фізичний інтерфейс RS-485 використовується разом з інтерфейсом RS-232, який програмується.

Протоколи обміну в більшості систем працюють за принципом – «ведучий–ведений». Один пристрій на магістралі є ведучим (master) й ініціює

обмін посилкою запитів веденим пристроям (slave), які розрізняються логічними адресами. Одним з популярних протоколів є протокол Modbus RTU.

Тип з'єднувачів і розпаювання також не обумовлюються стандартом. Трапляються з'єднувач DB9, клемні з'єднувачі т. ін.

8.4.1.5. Підключення інтерфейсів до мережі

Підключення інтерфейсів RS-485 до мікроконтролерної мережі відбувається за схемою, яку наведено на рис. 8.50.

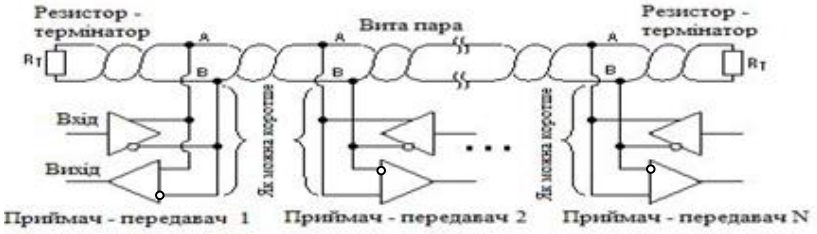


Рис. 8.50. Схема підключення інтерфейсів RS-485 до мікроконтролерної мережі

Під час підключення слід правильно приєднати сигнальні ланцюги, що зазвичай звуться A і B.

8.4.1.6. Узгодження «відкритого» кінця кабелю

Для узгодження «відкритого» кінця кабелю з рештою лінії використовують резистори-термінатори, які забезпечують усунення відбиття сигналу.

Номінальний опір резисторів відповідає хвильовому опору кабелю, і для кабелів на основі витої пари зазвичай становить: 100…120 Ом. Наприклад, широко поширений кабель UTP-5, використовуваний для прокладки Ethernet, має імпеданс 100 Ом. Для іншого типу кабелю може знадобитися інший номінал.

Резистори можуть бути запаяні на контакти кабельних роз'ємів наприкінці лінії. Іноді резистори бувають вмонтовані в самому пристрої і для підключення резистора потрібно встановити перемичку. У цьому випадку у разі від'єднання пристрою лінія розузгоджується, і для нормальної роботи решти системи потрібне підключення узгоджуючої заглушки.

8.4.1.7. Рівні сигналів у мережі

Інтерфейс RS-485 використовує балансну (диференційну) схему передачі сигналу. Це означає, що рівні напруг на сигнальних ланцюгах A і B змінюються в протифазі, як показано на рис. 8.51.

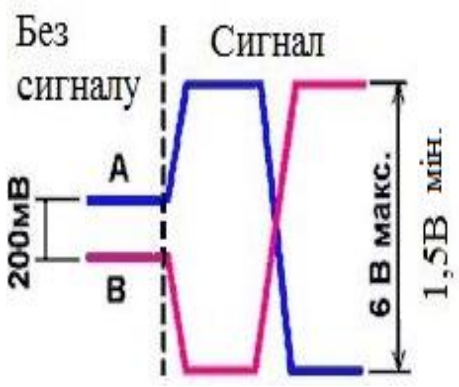


Рис. 8.51. Рівні сигналів у мережі

Передавач повинен забезпечувати мінімальний рівень сигналу 1,5 В за максимального навантаження – 32 стандартних входи і 2 резистора-термінатора, і не більше 6 В на холостому ході. Рівні напруг вимірюють диференціально, один сигнальний провід відносно іншого.

Пороговий діапазон прийнятого сигналу RS-485: ±200 мВ.

8.4.1.8. Зсув на сигнальних ланцюгах

За відсутності сигналу в мережі на сигнальних ланцюгах є невеликий зсув. Цей зсув призначений для захисту приймачів від помилкових спрацьовувань.

Рекомендовано створювати зсув трохи більше 200 мВ (зона недостовірності вхідного сигналу відповідно до стандарту), при цьому ланцюг А «підтягують» до додатного полюса джерела, а ланцюг В – до «спільного» (рис. 8.52).

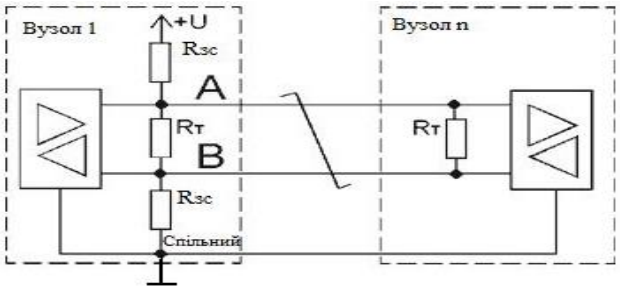


Рис. 8.52. Схема реалізації ланцюга зсуву на сигнальних ланцюгах

Номінали резисторів розраховують, виходячи з необхідного зсуву і напруги джерела живлення.

Величини опорів для резисторів захисного зсуву $R_{зс}$ розраховуються за дільником (рис. 8.53).

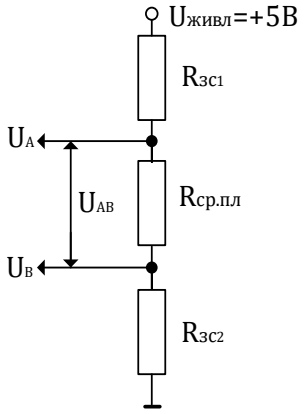


Рис. 8.53. Еквівалентна схема захисного зсуву

Необхідно забезпечити $U_{AB} > 200$ мВ. Напруга живлення: +5 В.

Опір середнього плеча $R_{ср.пл}$ дорівнює еквівалентному значенню паралельно включених двох резисторів-термінаторів $R_т$, які дорівнюють 120 Ом, і еквівалентного опору, наприклад, п’яти приймачів RS-485, кожен з яких має $R_{вх} = 12$ кОм. Тоді

$$R_{ср.пл} = \frac{R_{вх}}{5} \left\| \frac{R_{т}}{2} = \frac{12000}{5} \right\| \frac{120}{2} = 2400 \| 60 = \frac{2400 \cdot 60}{2400 + 60} = 59 Ом.$$

Приймемо $R_{зс1} = R_{зс2} = 560$ Ом, тоді напруга

$$U_A = \frac{U_{пит}(R_{ср.пл} + R_{зс2})}{R_{зс1} + R_{ср.пл} + R_{зс2}} = \frac{5 \cdot (59 + 560)}{560 + 59 + 560} = \frac{5 \cdot 619}{1179} = 2{,}63 В.$$

Напруга $U_в = \frac{U_{пит} \cdot R_{зс2}}{R_{зс1} + R_{ср.пл} + R_{зс2}} = \frac{5 \cdot 560}{560 + 59 + 560} = \frac{5 \cdot 560}{1179} = 2{,}38 В.$

Різниця напруг $U_{AB} = U_A - U_B = 2{,}63$ В $- 2{,}38$ В $= 0{,}25$ В $= 250$ мВ, тобто це відповідає умові $U_{AB} > 200$ мВ. Отже інтерфейс RS-485 подає на приймач інтерфейсу RS-232 логічну одиницю.

Якщо на лінії знаходиться багато приймачів, то номінал $R_{зс}$ має бути менше. В довгих лініях передачі необхідно також враховувати опір крученої пари, який може «з’їдати» частину зміщуючої різниці потенціалів для віддалених від місця підтяжки пристроїв. Для довгої лінії краще ставити два комплекти підтягуючих резисторів в обидва віддалені кінці поруч із термінаторами.

У разі наявності зміщення потенціал ланцюга А на холостому ході додатний відносно ланцюга В. Це може служити орієнтиром під час нового пристрою до кабелю за немаркованими проводами та може сповістити приймачу інтерфейсу RS-232 про наявність одиничного логічного рівня, після якого в мережі під час початку передачі приймач отримує нульовий стартовий біт (пп. 8.4.2.3).

8.4.1.9. Реалізація інтерфейсу RS-485

Нижче розглянуто практичну реалізацію інтерфейсу RS-485 на основі недорогих, економічних мікросхем фірми MAXIM: MAX481, MAX483, MAX485, MAX487…MAX491 і MAX1487. Ці мікросхеми є малопотужними приймачами/передавачами (трансиверами) для взаємодії за протоколами RS-485 і RS-422 (табл. 8.19).

Таблиця 8.19. **Параметри RS-485 мікросхем MAXIM**

Найменування	Дуплекс/ напівдуплекс	Швидкість передачі (Мбіт/с)	Обмеження наростання вихідної напруги	Режим мікро-спожи-вання	Дозвіл приймача/ передавача	Струм спокою (мкА)	Кількість пристроїв на шині	Кількість виводів
MAX481	Напів-дуплекс	2.5	Ні	Так	Так	300	32	8
MAX483	Напів-дуплекс	0.25	Так	Так	Так	120	32	8
MAX485	Напів-дуплекс	2.5	Ні	Ні	Так	300	32	8
MAX487	Напів-дуплекс	0.25	Так	Так	Так	120	128	8
MAX488	Дуплекс	0.25	Так	Ні	Ні	120	32	8
MAX489	Дуплекс	0.25	Так	Ні	Так	120	32	14
MAX490	Дуплекс	2.5	Ні	Ні	Ні	300	32	8
MAX491	Дуплекс	2.5	Ні	Ні	Так	300	32	14
MAX1487	Напів-дуплекс	2.5	Ні	Ні	Так	230	128	8

Кожна мікросхема включає в себе один вихідний формувач і один приймач. Конструктивно вона оформлюється в корпусах типу DIP (англ. Dual-in-line package), SO, μMAX. Мікросхеми: MAX483, MAX487, MAX488 та MAX489 містять вихідні формувачі з обмеженням швидкості наростання вихідної напруги. Це зменшує електромагнітне випромінювання і відбиття сигналу, викликані неправильним термінуванням крученої пари, тим самим дозволяючи передачу даних без помилок на швидкостях до 250 Кбіт/с. Вихідні формувачі мікросхем MAX481, MAX485, MAX490, MAX491 і MAX1487 не обмежують швидкість наростання вихідної напруги і дозволяють передавати дані на швидкості до 2,5 Мбіт/с.

Розглянуті приймачі/передавачі споживають від 120 мкА до 300 мкА у разі відсутності навантаження або наявності повного навантаження, але з відключеними вихідними формувачами. Додатково, MAX481, MAX483 і MAX487 мають режим зниженого споживання енергії, в якому вони споживають лише 0,1 мкА. Всі мікросхеми живляться від однополярної напруги: +5 В. Приймачі розглянутих мікросхем працюють у діапазоні: –7...+12 В.

Мікросхеми MAX487 і MAX1487 мають вчетверо більше навантаження на лінію, що дозволяє підключити до 128 приймачів на спільну шину. Мікросхеми MAX488…MAX491 забезпечують повнодуплексну передачу даних, а мікросхеми MAX481, MAX483, MAX485, MAX487 та MAX1487 розроблено для напівдуплексного режиму (рис. 8.54).

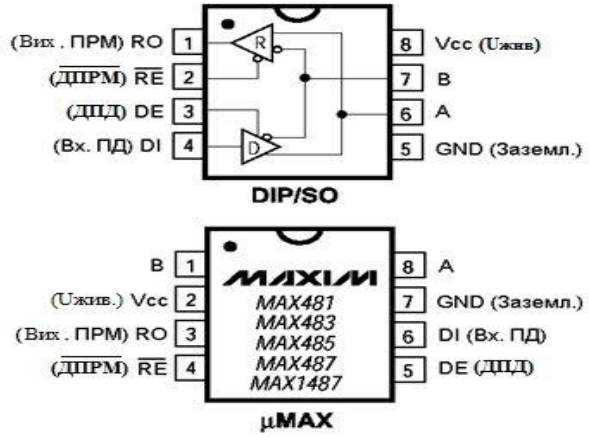


Рис. 8.54. Призначення виводів напівдуплексних мікросхем RS-485 фірми MAXIM

Розглянуті мікросхеми можуть працювати не тільки в мережі (рис. 8.50), а й у простому з'єднанні типу «точка–точка» (рис. 8.55).

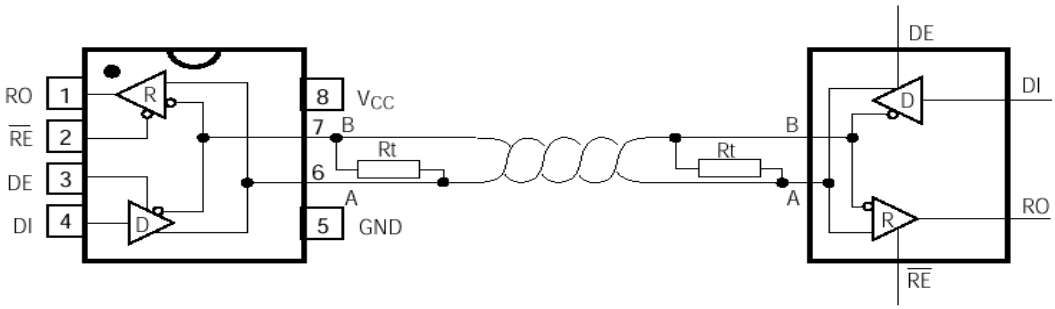


Рис. 8.55. Приклад простого з'єднання мікросхем типу точка-точка (напівдуплекс)

Нижче наведено приклад типового використання мікросхем RS-485 у напівдуплексній (рис. 8.56) та в дуплексній мережах (рис. 8.57).

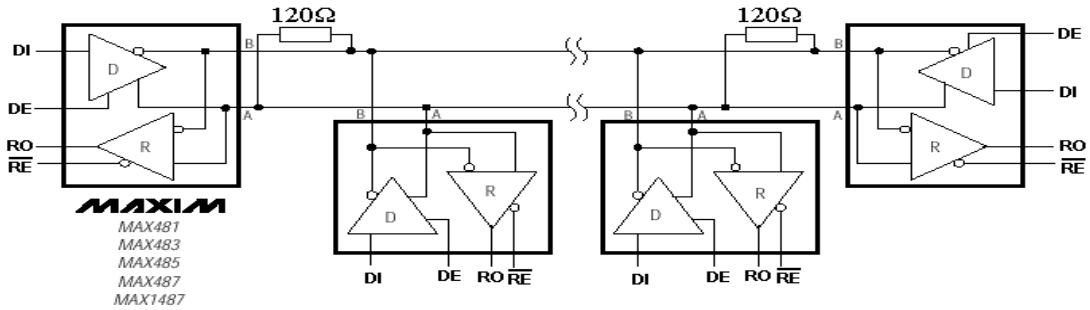
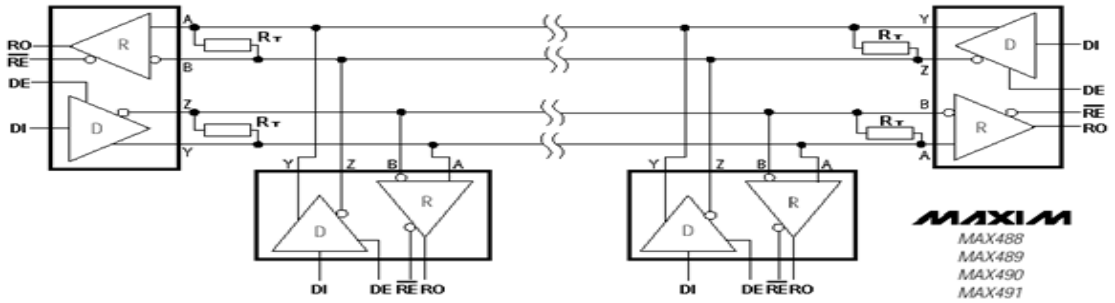


*Рис. 8.56.*Типове використання мікросхем RS-485 в напівдуплексній мережі



Примітка. $\overline{RE}$ і DE тільки для MAX489/MAX491. R_T – термінатор: 120 Ω.

Рис. 8.57. Типове використання мікросхем RS-485 у дуплексній мережі (мікросхеми MAX488…MAX491)

На рис. 8.58 наведено приклад використання мікросхеми MAX485 спільно з МК AT89C51.

На рис. 8.59 зображено функціональну схему мікроконтролерної мережі RS-485 в МПС з розподіленим керуванням.

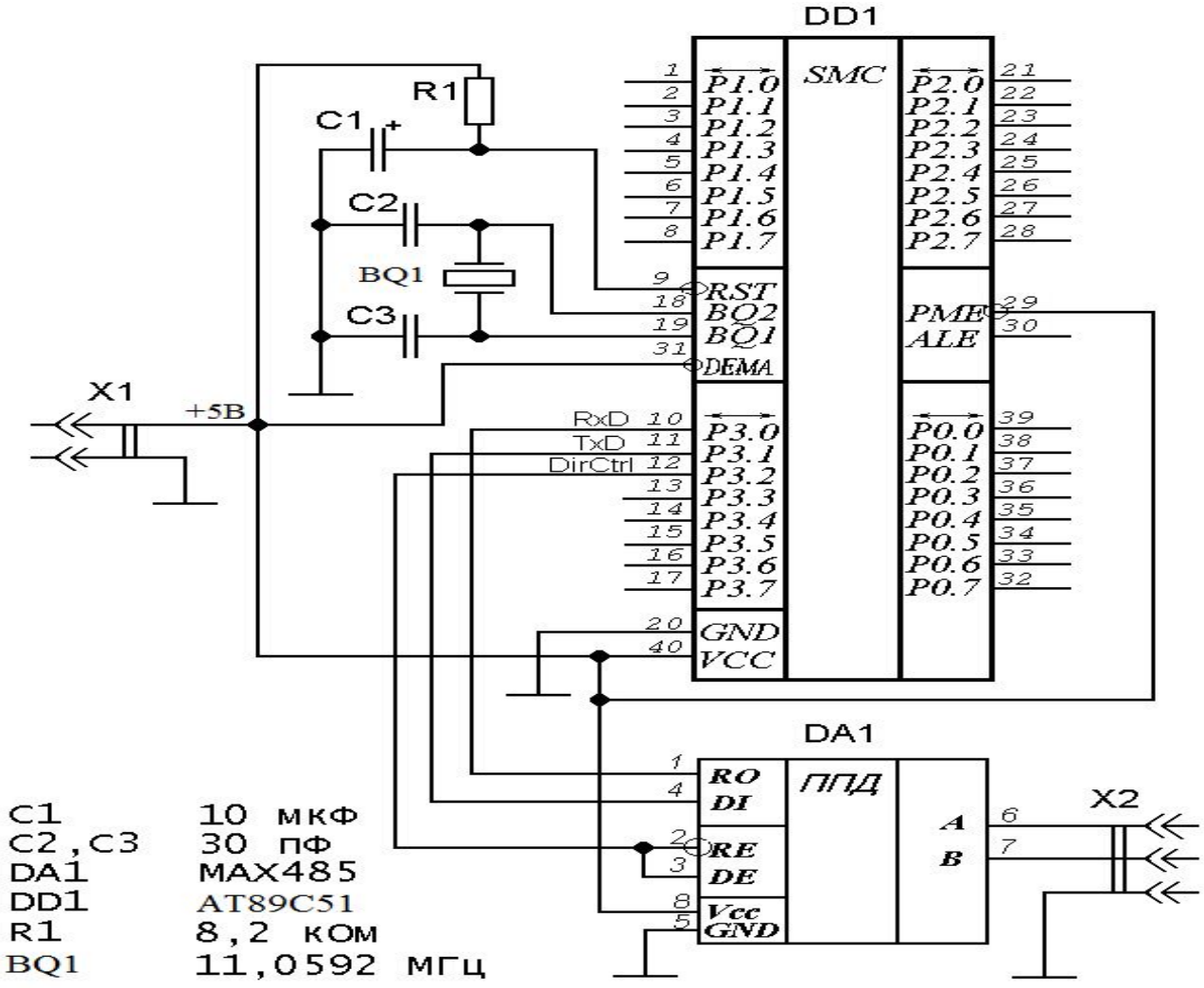


Рис. 8.58. Принципова схема підключення МК AT89C51 до мікросхеми RS-485

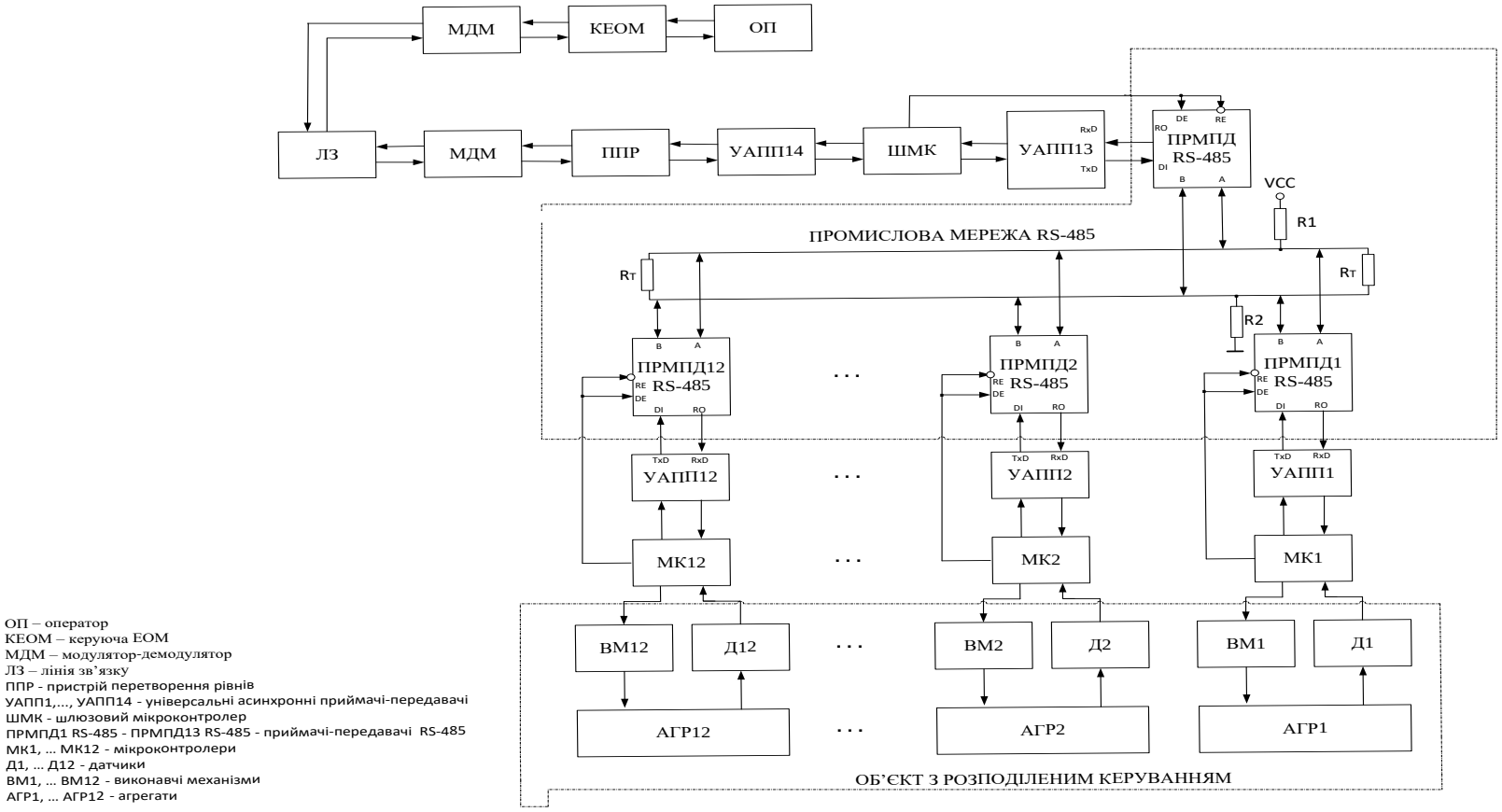


Рис. 8.59. Функціональна схема промислової мережі RS-485 для мікропроцесорної системи з розподіленим керуванням

8.4.2. Сполучення мікропроцесора/мікроконтролера з модемом/комп'ютером за допомогою інтерфейсу RS-232

8.4.2.1. Загальна характеристика

Розглянутий інтерфейс RS-485 є фізичним інтерфейсом, який забезпечує обмін даними у розглянутих вище мережах. Цей інтерфейс часто використовується разом з програмованим (логічним) інтерфейсом RS-232 для зв'язку з комп'ютером або МК.

RS-232 (англ. Recommended Standard 232), інша назва EIA232 [16] – стандарт фізичного рівня для асинхронного інтерфейсу (UART). Пристрій, який підтримує цей стандарт, широко відомий як послідовний порт персональних комп'ютерів. Історично стандарт мав широке поширення у телекомунікаційному устаткуванні. Нині він використовується для підключення до комп'ютерів широкого спектра обладнання, невибагливого до швидкості обміну, особливо у разі значного віддалення його від комп'ютера та відхилення умов застосування від стандартних. У комп'ютерах, зайнятих офісними та розважальними програмами, практично витіснений інтерфейсом USB.

RS-232 забезпечує передачу даних та деяких спеціальних сигналів між терміналом (англ. Data Terminal Equipment, DTE) та комунікаційним пристроєм (англ. Data Communications Equipment, DCE) на відстань до 15 метрів на максимальній швидкості – 115200 бод. Оскільки цей інтерфейс відомий не лише простотою програмування, але й невибагливістю, у реальних умовах ця відстань збільшується у багато разів із приблизно пропорційним зниженням швидкості.

Для передачі даних за інтерфейсом RS-232 використовується код NRZ, який є не таким, що самосинхронізується, тому для синхронізації використовуються стартовий та стоповий біти, що дозволяють виділити бітову послідовність та синхронізувати приймач з передавачем.

Спочатку інтерфейс створювався для підключення телефонних модемів до комп'ютерів. У зв'язку з такою спеціалізацією має рудименти у вигляді окремої лінії RING («дзвінок»). Поступово телефонні модеми перейшли на інші інтерфейси (USB), але роз'єм для RS-232 був на всіх персональних комп'ютерах, і багато виробників обладнання використовували його для підключення свого обладнання (наприклад, комп'ютерної миші).

Нині найчастіше використовується у промисловому та вузькоспеціальному устаткуванні та вбудованих системах. Крім того, RS-232 є на деяких телевізорах та ресиверах, зокрема супутникових, де призначений

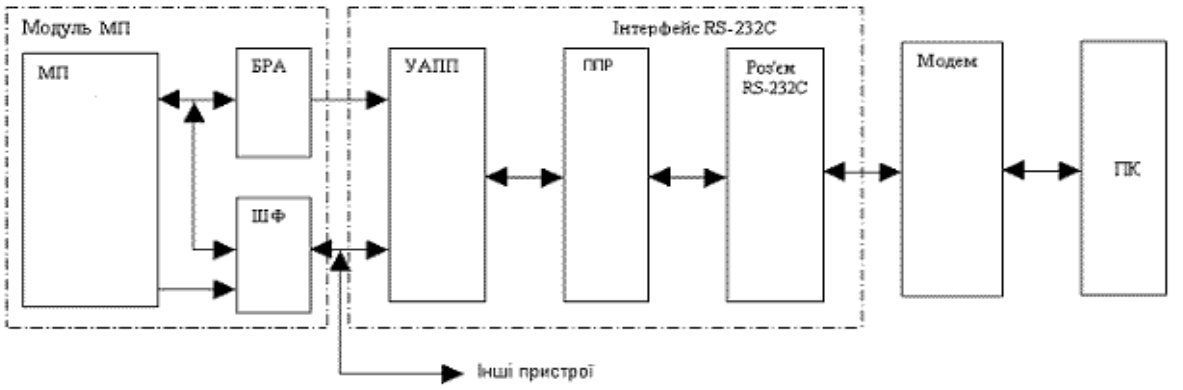
зокрема для оновлення вбудованого програмного забезпечення через комп’ютер.

Цей стандарт часто використовується для взаємодії МК різних архітектур, що мають інтерфейс UART/USART, з іншими цифровими пристроями та периферією.

RS-232 – дротовий дуплексний інтерфейс. Метод передачі даних аналогічний до асинхронного послідовного інтерфейсу UART. Інформація передається по дротах двійковим сигналом із двома рівнями напруги (код NRZ). Логічному «0» відповідає додатна напруга (від +5 до +15 для передавача), а логічній «1» – від’ємна (від –5 до –15 для передавача). Для електричного узгодження ліній RS-232 та стандартної цифрової логіки UART випускається велика номенклатура мікросхем драйверів, наприклад MAX232.

На рис. 8.60, наприклад, зображено структурну схему сполучення МП з модемом/комп’ютером через інтерфейс RS-232, яка включає:

– УАПП;
– пристрій перетворення рівнів;
– роз’єм RS-232.



МП - мікропроцесор; БРА – буферний регістр адреси
ШФ – шинний формувач; УАПП – універсальний асинхронний послідовний програмовуваний приймач – передавач;
ППР – пристрій перетворення рівнів; ПК - персональний комп'ютер

Рис. 8.60. Структурна схема сполучення МП з модемом/комп’ютером за допомогою інтерфейсу RS-232

Крім інтерфейсу RS-232 схема містить:

– буферний регістр адреси;
– шинний формувач.

8.4.2.2. Послідовний програмований універсальний асинхронний приймач/передавач

УАПП перетворює дані з паралельного формату в послідовний під час передачі (виведення) з МП і з послідовного формату в паралельний під час прийому (введення) у МП. На (рис. 8.61) наведено умовне позначення на

електричних схемах однієї з мікросхем, яка виконує функцію УАПП (англ. UART).

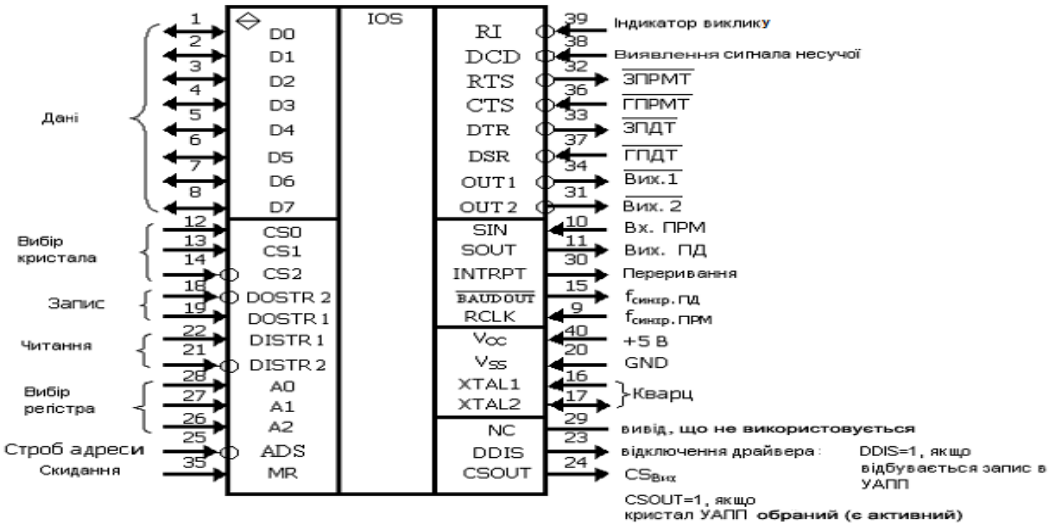


Рис. 8.61. Умовне позначення IMC TL16C450/550 (УАПП)

8.4.2.3. Формат даних послідовного програмованого універсального асинхронного приймача/передавача

Формат даних, які передаються в КЗ у послідовному форматі подано на рис. 8.62.

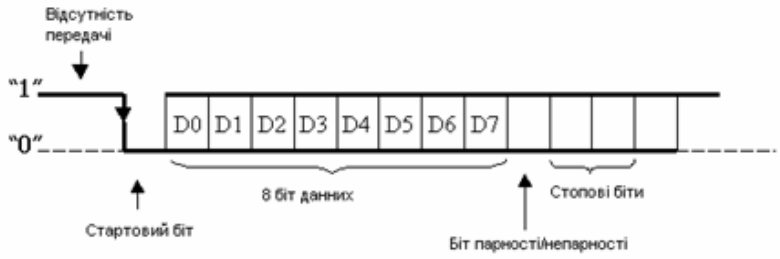


Рис. 8.62. Формат даних інтерфейсу RS-232

Власне дані (5, 6, 7 чи 8 біт) супроводжуються стартовим бітом, бітом парності/непарності (якщо такий контроль програмно передбачений), і стоповим одиничним сигналом, що включає 1; 1,5 чи 2 стоп-біта. Одержавши стартовий біт, приймач вибирає з лінії біти даних через визначені інтервали часу. Дуже важливо, щоб тактові частоти приймача і передавача були однаковими та стабільними (допустима розбіжність – не більше 10 %).

Швидкість передачі за RS-232 може вибиратися з ряду: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 біт/с (бод).

8.4.2.4. Пристрій перетворення рівнів

Усі сигнали RS-232 передаються/приймаються спеціально обраними рівнями, що забезпечують високу завадостійкість зв'язку (рис. 8.63).

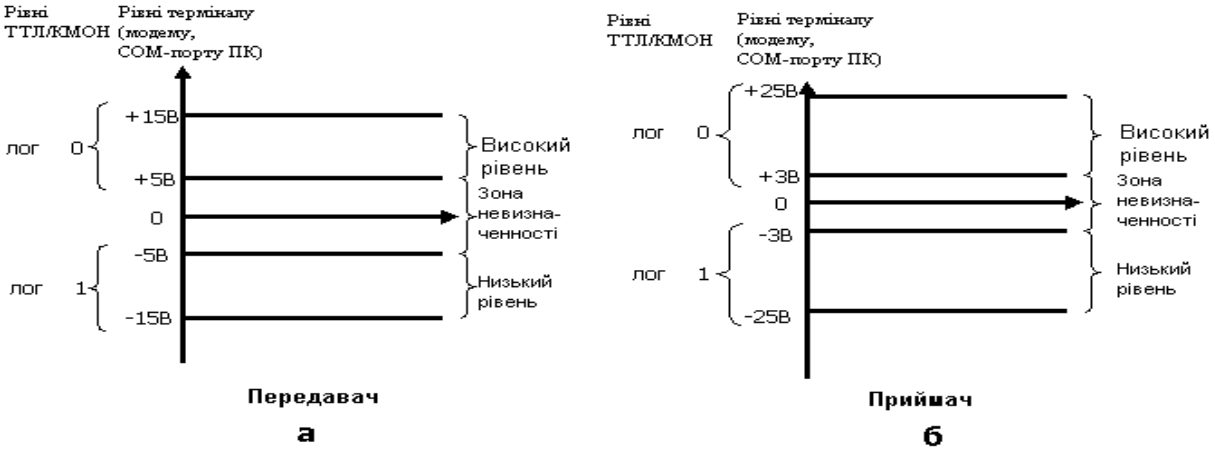


Рис. 8.63. Рівні сигналів RS-232C на передаючій та приймальній кінцях лінії зв'язку

Слід зазначити, що дані передаються/приймаються в інверсному вигляді: логічній одиниці відповідає низький рівень, а логічному нулю – високий рівень.

Як видно з рис. 8.63, під час передачі логічного нуля на виході інтерфейсу формується високий рівень напруги в діапазоні: +5 В…+15 В, під час передачі логічної одиниці – низький рівень напруги в діапазоні: –5 В…–15 В.

Під час прийому на вхід інтерфейсу надходить високий рівень напруги в діапазоні: +3 В…+25 В, що несе інформацію про логічний нуль, чи низький рівень напруги в діапазоні: –3 В…–25 В, що відображає логічну одиницю.

Таким чином, для узгодження ТТЛ/КМОН рівнів сигналів, що діють у МПС, з рівнями сигналів послідовного інтерфейсу, які передаються у лінію зв'язку або приймаються з лінії зв'язку, використовують пристрої перетворення рівнів.

Різні варіанти схемної реалізації пристроїв перетворення рівнів розглянуті в [10], одним із яких є застосування мікросхеми фірми MAXIM: MAX232A. Ця мікросхема (рис. 8.64) вимагає одне ДЖ +5 В та низку додаткових елементів – конденсаторів: C1, C2, …, C5, що не є надмірною ціною за переваги її застосування.

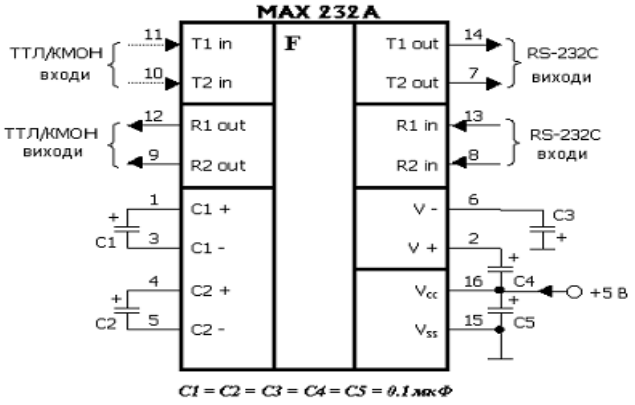


Рис. 8.64. Позначення і особливості підключення мікросхеми MAX232A

8.4.2.5. Роз'єм RS-232

Для зв'язку інтерфейсу RS-232 із зовнішнім терміналом (модемом) або ПК може використовуватися 25- чи 9-контактні роз'єми (рис. 8.65).

Контакт	Ланцюг
1	DCD (виявлення несучої)
2	RxD (вхід приймача)
3	TxD (вихід передавача)
4	DTR (запит передавача термінала)
5	SG (сигнальна земля)
6	DSR (готовність передавача термінала)
7	RTS (запит приймача термінала)
8	CTS (готовність приймача термінала)
9	RI (індикатор виклику)

Рис. 8.65. 9-контактний роз'єм RS-232

Опишемо призначення основних контактів роз'єму:

- SG – сигнальне заземлення, нульовий провід;
- TxD – дані, що передаються МП у послідовному коді (від'ємна логіка);
- RxD – дані, що приймаються МП у послідовному коді (від'ємна логіка);
- DCD – виявлення несучої даних (детектування сигналу, що приймається МП від модему);
- DTR – запит передавача терміналу (модему або ПК);
- DSR – готовність передавача терміналу (модему або ПК);
- RTS – запит приймача терміналу (модему або ПК);
- CTS – готовність приймача терміналу (модему або ПК);

– RI – індикатор виклику (свідчить про прийом модемом сигналу виклику від телефонної мережі).

Якщо у пристрої використовується МП (рис. 8.60), який не містить послідовного порту (УАПП), то у структурі інтерфейсу використовуються додатково буферний регістр адреси та шинний формувач.

8.4.2.6. Буферний регістр адреси

Буферний регістр адреси призначений для демультиплексування суміщеної шини адреси/даних МП. Для запам'ятовування адреси (у нашому випадку окремих регістрів УАПП) застосовують буферний регістр адреси, у якості якого може бути, наприклад, використана мікросхема SN74ALS374 (рис. 8.66).

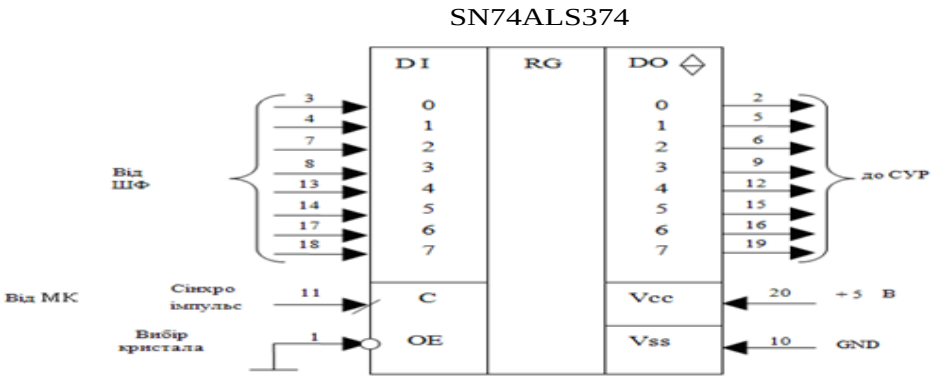


Рис. 8.66. Схема включення регістра

Ця мікросхема є 8-розрядним паралельним регістром з можливістю переведення виходів у третій (високоімпедансний, відключений) стан (OE = 1). В активному режимі на вході OE має бути логічний нуль. Для запису даних у буферний регістр адреси необхідно подати динамічний синхросигнал (перепад з 0 в 1) на вхід C (CLOCK). Після цього під час активного сигналу (логічний нуль) на вході OE (дозвіл виведення) на вихід буферного регістра адреси видається інформація, що відповідає даним на його входах у момент приходу синхросигнала.

Шинний формувач

З метою підвищення здатності навантаження виводів МП/МК, а також організації двостороннього обміну інформацією між МП/МК і системною шиною даних застосовують шинні формувачі.

Як шинний формувач може бути використана, наприклад, мікросхема SN54ALS245 (рис. 8.67), яка забезпечує двосторонній обмін інформацією за 8 лініями і здатна віддати в навантаження струм 30 мА.

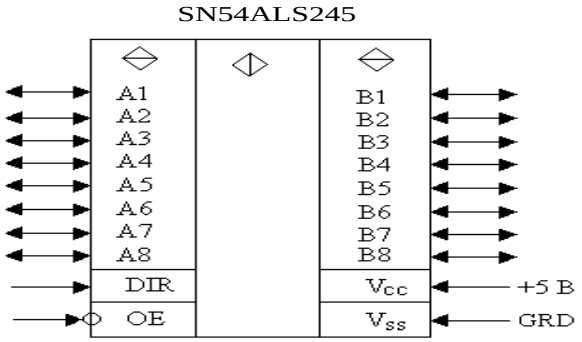


Рис. 8.67. Шинний формувач SN54ALS245

Напрямок обміну інформацією залежить від значення керуючого сигналу на вході DIR. Якщо DIR = 1, то дані передаються від A до B, а якщо DIR = 0, то – від B до A, при цьому на вході OE має бути присутнім активний сигнал – логічний нуль. Якщо OE = 1, то виходи шинного формувача переводяться у високоімпедансний (відключений) стан.

8.4.2.7. Мікропроцесор

В якості МП під час реалізації вищенаведеної структури інтерфейсу може бути використано, наприклад, один з «інтелоподібних» МП, які розглянуто в підрозд. 1.4.

У [3] розглянуто структуру мікросхеми УАПП 16450/16550, її підключення до МПС та програмування.

8.4.2.8. Реалізація інтерфейсу RS-232 для сполучення з мікроконтролером

Під час використання інтерфейсу RS-232 разом з сучасним МК, наприклад, сімейства AVR, з наведеної структури (див. рис. 8.60), залишаться пристрій перетворення рівнів та роз’єм.

Роль УАПП буде виконувати модуль UART/USART мікроконтролера.

Контрольні запитання та завдання

1. Дайте загальну характеристику інтерфейсу RS-485/EIA-485.
2. Наведіть та опишіть схему підключення інтерфейсу RS-485/EIA-485 до трипровідної лінії.
3. Яку кількість вузлів можна підключити в мережу RS-485/EIA-485?
4. Якою може бути швидкість обміну та відстань у мережі RS-485/EIA-485?
5. Наведіть та поясніть схему підключення інтерфейсів RS-485 до мікроконтролерної мережі.
6. Назвіть загальні рекомендації щодо використання інтерфейсу RS-485.

7. Наведіть та поясніть схему реалізації ланцюга зсуву на сигнальних ланцюгах інтерфейсу RS-485.
8. Назвіть параметри інтерфейсу RS-485 для мікросхем фірми MAXIM.
9. Опишіть призначення виводів напівдуплексних мікросхем RS-485 фірми MAXIM.
10. Наведіть та опишіть схему типу точка-точка (напівдуплекс) з'єднання двох мікросхем RS-485.
11. Наведіть та опишіть схеми типового використання мікросхем RS-485 у напівдуплексній та дуплексній мережах.
12. Наведіть та опишіть схему підключення МК AT89C51 до мікросхеми RS-485.
13. Дайте загальну характеристику інтерфейсу RS-232.
14. Наведіть та опишіть формат даних інтерфейсу RS-232.
15. Опишіть призначення та наведіть приклад реалізації пристрою перетворення рівнів.
16. Опишіть рівні сигналів RS-232 на передаючій та приймальній кінцях лінії зв'язку.
17. Опишіть позначення та особливості підключення мікросхеми MAX232.
18. Наведіть та опишіть 9-контактний роз'єм RS-232.

8.5. Мережа 1-WIRE

8.5.1. Особливості архітектури мережі 1-WIRE

Мережі 1-WIRE проектуються на основі однойменного інтерфейсу. Однопровідний інтерфейс 1-WIRE, розроблений фірмою Dallas Semiconductor, був рекомендований розробниками для застосування в багатьох сферах, однією з яких є система автоматизації (технологія 1-WIRE-мереж).

1-WIRE-мережі застосовуються для обслуговування високотехнологічних галузей машинобудування; хімічної промисловості; виготовлення електроніки для унікального експериментального і наукового обладнання і т. ін. Випускаються різноманітні зонди для вимірювання високих і низьких температур, датчики вологості, тиску і кислотності з особливими функціями, спеціальні оптичні сенсори, плати збору інформації, пристрої сполучення з різним аналітичним обладнанням та багато іншого.

Іншим прикладом, що наочно демонструє на практиці можливості технології шини 1-WIRE, є проект побудови повністю автоматичних метеорологічних станцій (1-WIRE Weather Station (1-WWS)).

Було створено кілька експериментальних систем 1-WWS, побудованих на основі персонального комп'ютера з адаптером DS9097U, керуючого комплексом з трьох термометрів DS18S20 для контролю температури, мікросхеми DS2438 для обслуговування датчика вологості повітря, компоненти DS2423 для визначення швидкості вітру і 16-ти електронних міток DS2401, що визначають його напрямок. Натомість окремі 1-WWS-системи комплектувалися додатковими однопровідними рішеннями, які забезпечували контроль сигналів від датчиків: барометричного тиску, розрядів блискавки, кількості опадів на поверхні, сонячної активності, вологості ґрунту і т. ін. Дані з усіх сенсорів, що реєструються кожною з подібних систем, надходили до персонального комп'ютера і через Інтернет транслювалися в режимі реального часу на центральний операторський пульт. Там виконувались прийом, обробка та архівація результатів про погоду всього регіону, які було отримано завдяки аналізу інформації від кількох територіально розподілених станцій.

До одного з найбільш затребуваних типових застосувань 1-WIRE належить маркування та безпека картриджів для принтерів і копіювальних апаратів, а також будь-яких електронних виробів, що вимагають цього. Однією з причин затребуваності є вбудований у чіпи механізм шифрування SHA.

Однопровідні 1-WIRE-чіпи DS2401, DS2411, DS2431, DS2432, DS28E01-100, DS2433, DS250x та інші від Dallas Semiconductor надзвичайно широко поширені для реалізації захисту CRUM (Customer Replaceable Unit Monitor – змінний користувачем блок моніторингу), що міститься у більшості копіювальних і розмножувальних пристроїв, при цьому вони часто виконують відразу кілька функцій, використовуючи мінімум контактів для обміну інформацією (тільки шини DATA і RET). Наприклад, виконують роль ідентифікатора устаткування, аутентифікатора легальності доступу, лічильника копій (або витрати тонера) і т. ін. Однопровідні 1-WIRE-чіпи сьогодні використовуються в картриджах величезної кількості моделей копіювальних апаратів від таких фірм-виробників, як: Xerox, Sharp, Fujitsu, Minolta і т. ін., причому багато компаній, що виробляють копіювальне обладнання, закуповують ці мікросхеми в промислових масштабах саме через вбудований SHA-механізм. Таке рішення є нині найбільш оптимальним у співвідношенні ефективність/ціна порівняно з будь-якими іншими варіантами апаратного захисту.

Досить перспективною галуззю, в якій повною мірою використовуються переваги технології 1-WIRE-мереж і якій особливо багато уваги приділяє компанія Dallas Semiconductor, є менеджмент автономних хімічних джерел струму – акумуляторних батарей. Під менеджментом тут

розуміється передусім сувора і повна ідентифікація джерел енергії, збереження в пам'яті вбудованого в батарею електронного пристрою особливостей її виготовлення та індивідуальних технічних характеристик, найбільш повний моніторинг їх основних експлуатаційних параметрів протягом усього терміну використання, а також формування коректного керуючого впливу, пов'язаного з відновленням заряду акумулятора, який обслуговується. Від правильного менеджменту і знання історії експлуатації батареї багато в чому залежить вибір алгоритму її повторного заряду, що безпосередньо пов'язано з ефективністю використання і терміном служби багатьох типів акумуляторів. Для цього кожна з батарей багатоелементних енергетичних конструкцій (особливо для мобільних або бездротових пристроїв і джерел безперебійного живлення) забезпечується індивідуальним однопровідним 1-WIRE-компонентом, перетворюючись по суті в інтелектуальний системний елемент автономного живлення, при цьому сповідується комплексний підхід до проблеми менеджменту енергетичних елементів, коли 1-WIRE-рішення дозволяють організувати недорогу багатоточкову шину комплексного обслуговування пристроїв менеджменту і керування зарядом, що дає можливість супроводжувати як окремі автономні джерела енергії, так і цілі батареї, складені з багатьох одиничних енергетичних складових.

Нині Dallas Semiconductor постачає широку номенклатуру різних за функціональним призначенням однопровідних компонентів для реалізації найрізноманітніших мережевих задач. Є багато конкретних прикладів застосування 1-WIRE-інтерфейсу для автоматизації у найрізноманітніших галузях і все більше розробників цікавляться цією технологією.

Як середовище для передачі інформації по однопровідній лінії найчастіше використовується звичайний телефонний кабель і, як наслідок, швидкість обміну в цьому випадку невелика. Однак, якщо ретельно проаналізувати об'єкти, які потребують автоматизації, то для більшості з них гранична швидкість обслуговування в 16,4 Кбіт/с є достатньою.

Інші переваги 1-WIRE-технології, такі як:

- просте вирішення адресування;
- нескладний протокол;
- проста структура лінії зв'язку;
- проста зміна конфігурації мережі;
- дешевизна всієї технології в цілому.

Такі переваги показують раціональність та високу ефективність цього інструмента під час вирішення задач комплексної автоматизації у різних областях діяльності.

8.5.2. Основні характеристики інтерфейсу та мережі 1-WIRE

Мережа 1-WIRE є мережею з централізованим керуванням та шинною топологією. Для здійснення зв'язку використовується одна лінія даних (DATA) та один зворотний провід. Таким чином, для реалізації середовища обміну цієї мережі можуть бути застосовані доступні кабелі, які мають неекрановану виту пару будь-якої категорії, а також, наприклад, звичайний телефонний провід. Такі кабелі під час їх прокладання не потребують наявності спеціального обладнання, а обмеження максимальної довжини однопровідної лінії регламентовано розробниками на рівні 300 м.

Основою архітектури 1-WIRE-мереж є топологія спільної шини, коли кожен з пристроїв підключено безпосередньо до єдиної магістралі без каскадних з'єднань або розгалужень, при цьому як базова використовується структура мережі з одним ведучим та багатьма веденими.

Конфігурація будь-якої 1-WIRE-мережі може довільно змінюватися в процесі її роботи, при цьому не виникає перешкод подальшої експлуатації та працездатності всієї системи в цілому, якщо у разі цих змін дотримуються основні принципи організації однопровідної шини. Ця можливість досягається завдяки наявності в протоколі 1-WIRE-інтерфейсу спеціальної команди пошуку ведених пристроїв, яка дозволяє швидко визначити нових учасників інформаційного обміну. Стандартна швидкість виконання такої команди дозволяє обробити приблизно 75 вузлів мережі за секунду.

Завдяки наявності у будь-якому пристрої, який має 1-WIRE-інтерфейс, унікальної індивідуальної адреси (відсутність збігу адрес для пристроїв, які виробляє Dallas Semiconductor, гарантується самою фірмою-виробником) така мережа має практично необмежений адресний простір, при цьому кожен з однопровідних приладів одразу готовий до використання у 1-WIRE-мережі без будь-яких додаткових апаратно-програмних модифікацій.

Однопровідні компоненти є самотактуючими напівпровідниковими пристроями, в основі обміну інформацією між якими лежить керування зміною тривалості часових інтервалів імпульсних сигналів в однопровідному середовищі та їх вимірювання. Передача сигналів для 1-WIRE-інтерфейсу асинхронна та напівдуплексна, а вся інформація, що циркулює в мережі, сприймається абонентами або як команди, або як дані. Команди мережі генеруються ведучим та забезпечують різні варіанти пошуку та адресації ведених пристроїв, визначають активність на лінії окремих компонентів, керують обміном даними в мережі і т. ін.

Стандартну швидкість роботи 1-WIRE-мережі, яка складає 16,4 Кбіт/с, було обрано, по-перше, з урахуванням забезпечення максимальної надійності

передачі даних на великі відстані, та, по-друге, з урахуванням швидкодії найбільш широко розповсюджених типів МК, які переважно повинні використовуватися під час реалізації ведучих пристроїв однопровідної шини. Це значення швидкості обміну може бути зменшено до будь-якого можливого значення завдяки введенню примусової затримки між передачею в лінію окремих бітів даних (розтягуванню часових слотів протоколу). Швидкість обміну також може бути збільшено завдяки переходу на спеціальний прискорений режим обміну (швидкість Overdrive – до 125 Кбіт/с), який допускається для окремих типів однопровідних 1-WIRE-компонентів на невеликій по відстані якісній лінії, яка не перевантажена іншими пристроями лінії зв'язку.

Під час реалізації однопровідного інтерфейсу використовуються стандартні КМОН/ТТЛШ логічні рівні сигналів, а живлення більшості однопровідних компонентів може здійснюватися від зовнішнього джерела з робочою напругою в діапазоні від 2,8 до 6,0 В.

Альтернативою застосуванню зовнішнього живлення є механізм «паразитного живлення», дія якого полягає у використанні кожним з ведених компонентів 1-WIRE-лінії електричної енергії імпульсів, які передаються по шині даних. Ця енергія акумулюється спеціальною ємністю, яку вбудовано в пристрій. Окрім того, окремі компоненти однопровідних мереж можуть використовувати режим живлення по шині даних, коли енергія до приймача надходить безпосередньо від ведучого пристрою по лінії зв'язку, при цьому обмін інформацією в мережі примусово припиняється.

8.5.3. Фізична реалізація інтерфейсу 1-WIRE

Фізична реалізація інтерфейсу 1-WIRE достатньо проста. На рис. 8.68 показано спрощену схему апаратної реалізації інтерфейсу 1-WIRE.

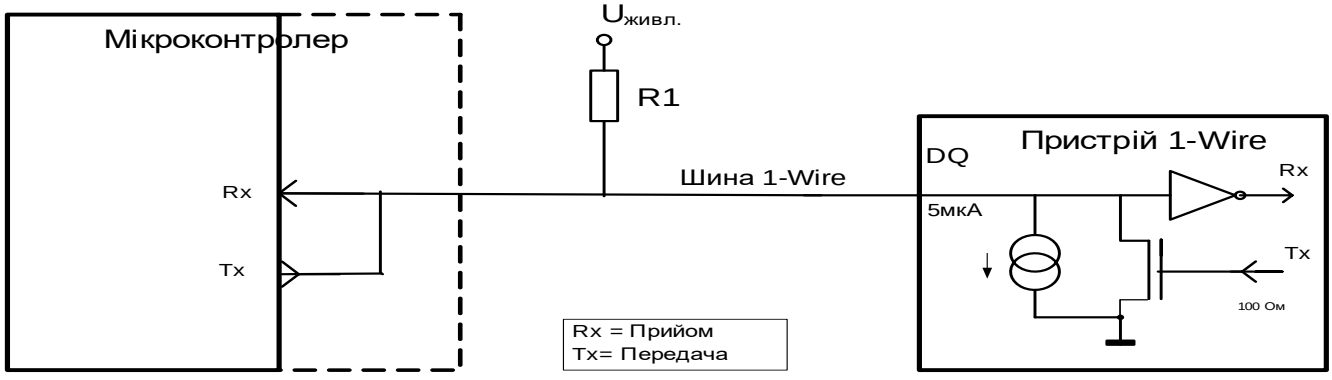


Рис. 8.68. Схема апаратної реалізації інтерфейсу 1-WIRE

Вивід DQ 1-WIRE-пристрою є входом КМОН-логічного елемента, який може бути зашунтований польовим транзистором. Опір каналу цього транзистора у відкритому стані – близько 100 Ом. Коли транзистор закрито – є невеликий струм витоку (приблизно 5 мкА) на спільний провід.

Шина 1-WIRE має бути підтягнута окремим резистором R1 до напруги живлення пристроїв. Опір цього резистора 4,7 кОм, однак це значення рекомендовано лише для достатньо коротких ліній. Якщо шина 1-WIRE використовується для підключення віддалених на велику відстань пристроїв, то опір цього резистора слід зменшити. Мінімально допустиме його значення – 300 Ом, а максимальне – близько 20…30 кОм. Дані величини орієнтовні та завжди уточнюються за характеристиками конкретного пристрою 1-WIRE.

Підключення шини 1-WIRE до МК на рис. 8.68 показано умовно в двох варіантах: з використанням або двох окремих виводів МК (один у якості входу, інший – виходу), так й одного, який працює як на вхід, так і на вихід. Розділення цих способів показано за допомогою пунктирної лінії, яка умовно позначає межу корпуса МК. З деяким припущенням можна уявити собі логічну будову шини 1-WIRE, як відоме з'єднання виводів мікросхем з відкритим стоком за схемою «монтажне АБО» для логічних сигналів низького рівня [1]. Очевидно, що передача будь-якої інформації можлива тільки видачею низького рівня в лінію – замиканням її на спільний провід, а у високий логічний рівень лінія повернеться сама завдяки наявності зовнішнього резистора, який його підтягує. Залежно від способу прокладання, сполучення з веденими пристроями і матеріалів, що використовуються під час прокладання, розрізняють три основних варіанти організації 1-WIRE-мереж, кожен з яких передбачає використання особливої технології та аксесуарів під час реалізації лінії [8].

Обмін інформацією ведеться тайм-слотами: один тайм слот служить для обміну одним бітом інформації; дані передаються побайтно, біт за бітом, починаючи з молодшого біта. Достовірність переданих/прийнятих даних (перевірка відсутності спотворень) гарантується за допомогою підрахунку циклічної контрольної суми [8].

1-WIRE-мережі мають свою нішу для застосування під час побудови систем автоматизації. Безглуздо всерйоз використовувати їх для передачі великих масивів інформації, під час побудови, наприклад, систем відеоспостереження або швидкісного обміну, пов'язаних з обслуговуванням швидких процесів, або ж порівнювати можливості однопровідних мереж з такими потужними мережевими промисловими інтерфейсами, як ProfiBus, FeldBus, LonWorks, Industrial Ethernet і т. ін.

Основні обмеження для застосування систем, побудованих на основі 1-WIRE-мереж в галузі автоматизації, описано у [6].

8.5.4. Структура мережі 1-WIRE

На рис. 8.69 наведено структурну схему 1-WIRE-мережі у системі вимірювання температури.

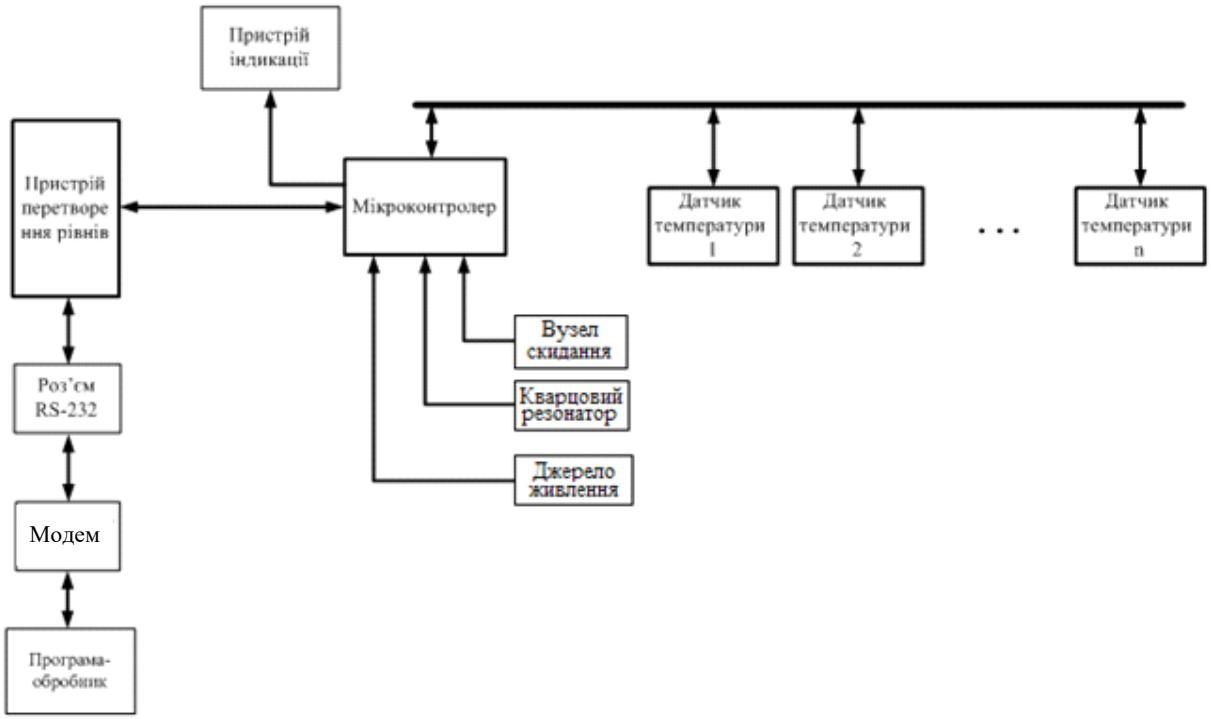


Рис. 8.69. Структурна схема 1-WIRE-мережі у системі вимірювання температури

Система призначена для вимірювання температури в приміщенні, де необхідно вимірювати температуру в декількох точках. Система є набором датчиків температури, які можуть бути розміщено на відстані до 300 метрів.

Система виконує індикацію температури в самому приміщенні (виведення на LCD-дисплей) і передачу даних через модем на віддалений пункт обробки та керування.

У цій системі як ведучий у мережі 1-WIRE використовується, наприклад, МК ATmega8535 – 8-розрядний КМОН-мікроконтролер, заснований на розширеній AVR-RISC-архітектурі.

Завдяки виконанню більшості інструкцій за один машинний цикл ATmega8535 досягає продуктивності 1 млн операцій за секунду, що дозволяє проектувальникам систем оптимізувати співвідношення енергоспоживання і швидкодії [3].

8.5.5. Загальна характеристика датчика температури

В якості ведених пристроїв 1-WIRE-мережі (рис. 8.69) можуть бути датчики температури, наприклад, DS18B20 з програмованим значенням розрядності вихідного коду від 9 до 12 біт, яку можна зберігати у EEPROM-пам’яті приладу [3; 6; 8]. Датчик DS18B20 обмінюється даними по 1-WIRE-шині і при цьому може бути як єдиним пристроєм на лінії, так і працювати в групі. Всіма процесами на шині керує центральний МК.

Діапазон вимірювань від –55 ° С до +125 °С. У діапазоні від –10 °С до +85 °С точність 0,5 °С.

Датчик DS18B20 може живитися напругою лінії даних («паразитне живлення») за відсутності зовнішнього джерела напруги.

На рис. 8.70 показано розміщення контактів датчика.

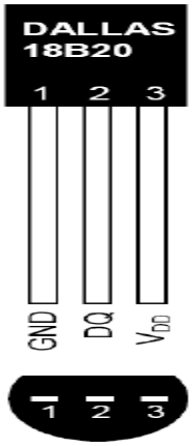


Рис. 8.70. Датчик температури DS18B20

Призначення виводів датчика DS18B20 наведено в табл. 8.20.

Таблиця 8.20. **Призначення виводів датчика DS18B20**

№ з/п	Назва виводу	Опис
1	GND	Спільний
2	DQ	Вивід введення/виведення даних (Input/Output pin). Цією лінією подається живлення в режимі роботи з «паразитним живленням»
3	VDD	Вивід живлення. Для режиму роботи із «паразитним живленням» VDD необхідно з’єднати зі спільним проводом

Кожен датчик DS18B20 має унікальний 64-бітовий послідовний двійковий код адреси, який дозволяє спілкуватися з багатьма датчиками DS18B20, які встановлено на одній шині. Такий принцип дозволяє використовувати один МК, щоб контролювати багато датчиків DS18B20, розміщених на деякій відстані. Програми, які можуть отримати вигоду з цієї особливості, мають системи контролю температури в будівлях, устаткуванні чи машинах.

На рис. 8.71 наведено структурну схему датчика DS18B20.

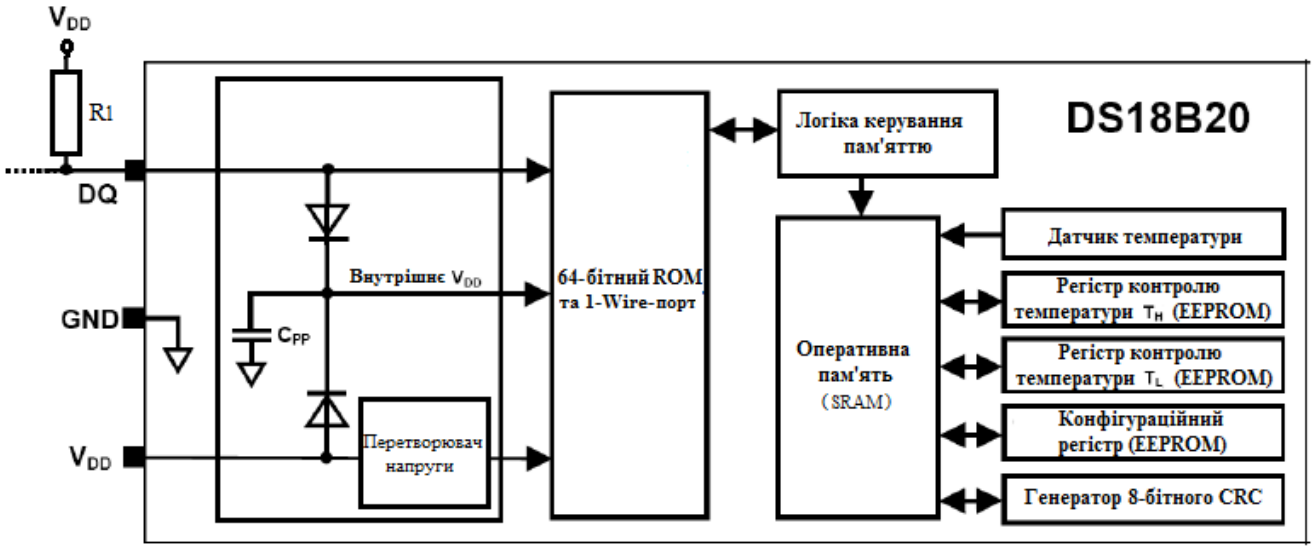


Рис. 8.71. Структурна схема датчика DS18B20

Датчик має 64-бітовий постійний запам’ятовуючий пристрій (англ. ROM), який зберігає унікальний двійковий код приладу. Оперативна пам’ять (SRAM) містить двобайтовий регістр температури, який зберігає значення температури по закінченню процесу її вимірювання.

У пам’яті EEPROM є два однобайтових регістра контролю температури: TH і TL та конфігураційний регістр. Цей регістр дозволяє користувачеві встановлювати розрядність цифрового перетворювача температури: 9, 10, 11 або 12 біт, що впливає на час конвертації температури. Оскільки регістри TH, TL і регістр конфігурації є комірками енергонезалежної пам’яті даних EEPROM, то вони зберігають дані, коли прилад вимкнено.

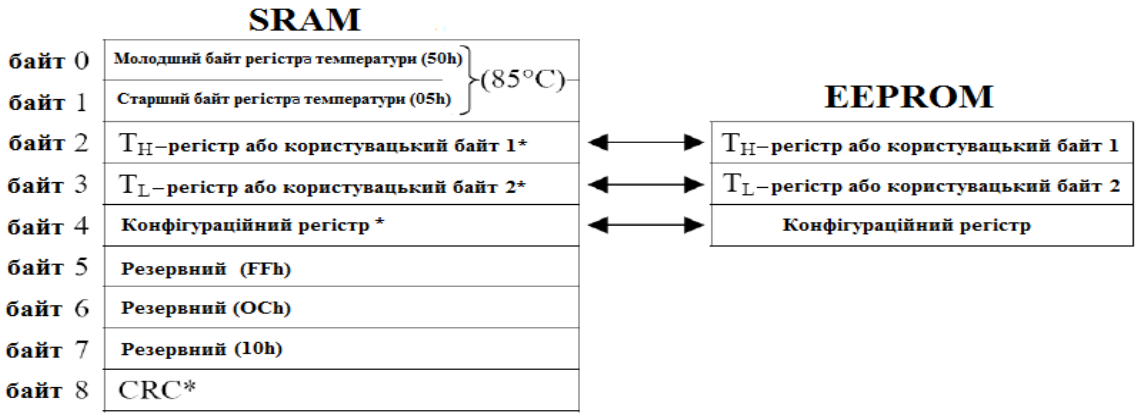
Датчик DS18B20 використовує 1-WIRE-протокол. МК (пристрій керування) ідентифікує і звертається до датчиків температури, використовуючи 64-бітовий код приладу. Оскільки кожен прилад має унікальний код, кількість приладів, до яких можна звернутися на одній шині, фактично необмежено.

Інша особливість датчика DS18B20 – здатність працювати без зовнішнього живлення. Ця можливість надається через резистор, що підтягує. Високий рівень сигналу шини заряджає внутрішній конденсатор (Cpp), який живить прилад, коли на шині низький рівень. Цей метод називається «паразитне живлення», при цьому максимальна температура, яка вимірюється, становить +100 °C. Для розширення діапазону температур до +125 °C необхідно використовувати зовнішнє живлення.

8.5.6. Організація пам'яті датчика температури

8.5.6.1. Загальна характеристика пам'яті датчика

Організацію пам'яті датчика DS18B20 показано на рис. 8.72.



Примітка. * – Стан після включення живлення залежить від значень, збережених в EEPROM

Рис. 8.72. Організація пам'яті датчика DS18B20

Пам'ять складається з оперативної енергозалежної пам'яті SRAM і енергонезалежної пам'яті EEPROM. Перші два байти пам'яті SRAM – це регістр температури, наступні – регістри стану «Аварія», які відображають верхню та нижню межу вимірювання (TH і TL) і регістр конфігурації. Якщо стан «Аварія» не використовується, то регістри TH і TL можуть бути комірками оперативної пам'яті SRAM.

8.5.6.2. Регістр температури

Молодший та старший байти регістра температури зберігають поточну вимірювану температуру. Під час подачі живлення початкове значення температури становить +85 градусів. Формат регістра подано на рис. 8.73, де S – знак температури (0 – додатна температура, 1 – від'ємна), біти від 11-го до 4-го – ціла частина значення температури, біти від 3-го до 0-го – дробова частина.

	біт 7	біт 6	біт 5	біт 4	біт 3	біт 2	біт 1	біт 0
Молодший байт	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	біт 15	**біт 14**	**біт 13**	**біт 12**	**біт 11**	**біт 10**	**біт 9**	**біт 8**
Старший байт	S	S	S	S	2^7	2^6	2^5	2^4

*Рис. 8.73.*Формат регістра температури

8.5.6.3. Формування стану «Аварія»

Після того, як датчик DS18B20 виконає температурне перетворення, поточне температурне значення порівнюється зі значеннями, які записано в регістрах TH і TL. На рис.8.74 наведено формат цих регістрів.

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

Рис.8.74. Формат регістрів TH та TL

Біт знака S вказує на додатне чи від'ємне значення температури: для додатних чисел S = 0, а для від'ємних – S = 1. Регістри TH і TL є комірками енергонезалежної пам'яті даних EEPROM. Таким чином, вони зберігають дані, коли пристрій знеструмлено. До TH і TL можна звернутися через другий та третій байти SRAM.

Для порівняння використовуються тільки біти 4…10 регістра температури (ціле значення температури).

Якщо вимірювана температура нижче або дорівнює TL або вище або дорівнює TH, формується стан «Аварія» і встановлюється прапорець «Аварія». Цей прапорець оновлюється після кожного наступного температурного перетворення. Якщо при цьому стан «Аварія» зникне, то прапорець «Аварія» буде скинуто.

Ведучий пристрій може перевірити стан «Аварія» всіх датчиків DS18B20 на шині, подаючи команду «Пошук Аварії» [ECh]. Будь-який датчик DS18B20 з встановленим прапорцем «Аварія» відповість на цю команду. Таким чином, ведучий пристрій точно може визначити, які датчики DS18B20 перебувають у стані «Аварія». Якщо змінюються значення регістрів TH або TL, то необхідно запустити нове температурне перетворення, щоб виконалась перевірка умови контролю температури, яку задано в регістрах TH або TL.

8.5.6.4. Конфігураційний регістр

Четвертий байт пам'яті SRAM – є конфігураційним регістром. У ньому задається температурна розрядність датчика DS18B20. Початкове значення регістра залежить від вмісту EEPROM-пам'яті. Формат

конфігураційного регістра подано на рис. 8.75, де R1, R0 – біти, що задають температурну розрядність термометра, від якої залежить максимальний час перетворення температури (табл. 8.21).

біт 7	біт 6	біт 5	біт 4	біт 3	біт 2	біт 1	біт 0
0	R1	R0	1	1	1	1	1

Рис. 8.75. Формат конфігураційного регістра

Таблиця 8.21. **Залежність розрядності та максимального часу перетворення від значень бітів R1, R0**

R1	R0	Розрядність	Максимальний час перетворення
0	0	9 біт	93,75 мс
0	1	10 біт	187,5 мс
1	0	11 біт	375 мс
1	1	12 біт	750 мс

8.5.6.5. Контрольна сума циклічного надлишкового коду

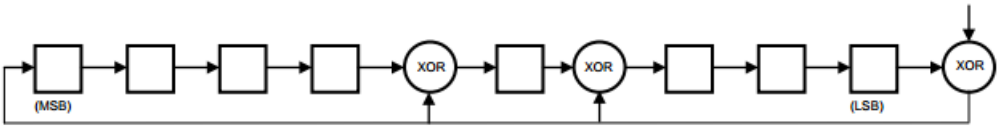
Контрольна сума (CRC) – це байт, який зберігається останнім (восьмим) у пам'яті SRAM. Він обчислюється за спеціальним алгоритмом на основі значення всіх семи попередніх байтів.

Алгоритм підрахунку такий, що якщо всі байти передані-прийняті без спотворень (а спотворення цілком можливі, якщо згадати характер апаратної реалізації інтерфейсу), прийнятий байт контрольної суми обов'язково збігається з розрахованим у МК або пристрої значенням. Тобто під час реалізації обміну інформацією потрібно у разі передачі та прийому байтів підраховувати їх контрольну суму за встановленим алгоритмом, а потім або передати отримане значення (якщо було введено передачу адреси/даних), або порівняти розраховане значення з прийнятим значенням CRC. Тільки у разі збігу обох CRC МК або пристрій вважають прийняті дані достовірними. В іншому випадку продовження обміну неможливе. Алгоритм підрахунку CRC має бути однаковим як для МК, так і для будь-якого пристрою.

Байт CRC обчислюється з використанням утворюючого многочлена

$$x^8 + x^5 + x^4 + 1.$$

Схема формування CRC має такий вигляд [3; 8]:



Наприклад, реальна 64-бітова адреса датчика DS18B20 є «2830C5B80000008E». Останній байт адреси (8E) – контрольна сума.

Фірма Dallas Semiconductor розраховує CRC для оберненої послідовності. Для даного прикладу маємо комбінацію: 8E 00 00 00 B8 C5 30 28. У двійковій системі комбінація виглядає так: 10001110000000000000000000000000101110001100010100110000001010000.

Дзеркально відобразимо цю комбінацію, щоб полегшити обчислення, інакше нам довелося б ділити комбінацію з кінця, зсуваючи біти, яких бракує, вліво (а не вправо, як під час звичайного ділення). Відкидаємо останній байт, записуємо адресу датчика у двійковій системі, дописуємо в кінець 8 нулів (кількість останніх відповідає ступеню утворюючого многочлена) та отримаємо:
0001010000001100101000110001110100000000000000000000000000000000.

Утворюючий многочлен можна подати як

$$2^8 + 2^5 + 2^4 + 1 = 305,$$

що у двійковому вигляді дорівнює 100110001.

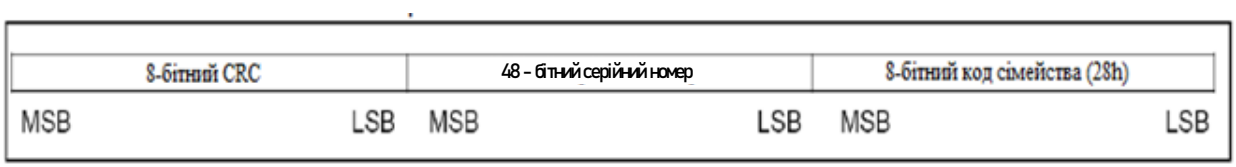
Виконуємо ділення адреси датчика (з вісьмома нулями в кінці) на утворюючий многочлен:

0001010000001100101000110001110100000000000000000000000000000000	100110001
…	
01110001	

Перевертаємо залишок від ділення 01110001 та отримаємо 10001110, що дорівнює 8E у 16-ковій системі числення (збігається з останнім контрольним байтом нашої адреси давача). Приклад розрахунку CRC для адреси давача «28 30 C5 B8 00 00 00 8E» наведено у [8].

8.5.7. Формат коду датчика

Кожен датчик DS18B20 містить унікальний 64-бітовий код, збережений у ROM. Формат коду подано на рис. 8.76.



Примітка. LSB – Least Significant Bit (молодший значущий розряд); MSB – Most Significant Bit (старший значущий розряд).

Рис. 8.76. Формат коду датчика

Молодші 8 біт коду містять код сімейства 1-WIRE. Для датчика DS18B20 це 28h. Наступні 48 біт містять унікальний серійний номер датчика. Старші 8 біт містять CRC-байт, який обчислено від перших 56 біт коду ROM. Саме завдяки унікальності коду забезпечується адресація пристроїв 1-WIRE.

8.5.8. Команди мікроконтролера

8.5.8.1. Загальна характеристика команд

Як було зазначено вище, кожен пристрій 1-WIRE має унікальний ідентифікаційний 64-бітовий номер, який було запрограмовано на етапі виробництва мікросхеми. Унікальний – це означає, що фірма-виробник гарантує, що не знайдеться двох мікросхем з однаковим ідентифікаційним номером (принаймні протягом кількох десятків років за наявних темпів виробництва).

Під час розгляду протоколу обміну будемо вважати, що на шині 1-WIRE є більше одного пристрою. У цьому випадку перед МК буде дві задачі: визначення кількості наявних пристроїв і вибір (адресація) одного конкретного з них для обміну даними.

Команди МК поділяються на дві групи:

- ROM-команди – команди для роботи з адресами пристроїв (пошук адреси, зчитування адреси, вибір адреси тощо);
- функціональні команди – команди, наявність яких вимагає виконання відповідних дій з боку кінцевого пристрою.

Послідовність операцій доступу до датчика DS18B20 така:

- ініціалізація;
- подача ROM-команди;
- подача функціональної команди DS18B20.

8.5.8.2. ROM-команди

У табл. 8.22 наведено деякі з важливих загальних ROM-команд.

Опис виконання цих команд наведено у [8].

8.5.8.3. Функціональні команди

Функціональні команди дозволяють МК читати і записувати в оперативну пам’ять датчика DS18B20, запускати температурне перетворення датчика, визначати його режим живлення.

Опис виконання цих команд наведено у [8].

Таблиця 8.22. **ROM-команди пристроїв 1-WIRE**

Команда	Значення байта	Опис
SEARCH ROM	0xF0	Пошук адрес – використовується під час універсального алгоритму визначення кількості та адрес підключених пристроїв
READ ROM	0x33	Зчитування адреси пристрою – використовується для визначення адреси єдиного пристрою на шині
MATCH ROM	0x55	Вибір адреси – використовується для звернення до конкретної адреси пристрою з багатьох підключених
SKIP ROM	0xCC	Ігнорування адреси – використовується для звернення до всіх або єдиного пристрою на шині, при цьому адреса пристрою ігнорується (можна звертатися до невідомого пристрою)

8.5.9. Схема алгоритму роботи системи

Схему алгоритму роботи системи наведено у [8]. До схеми алгоритму належать: алгоритм ініціалізації МК, що міститься у системі; алгоритм ініціалізації LCD; алгоритм ініціалізації УАПП; алгоритм роботи головної підпрограми, що виконується під час роботи системи; алгоритм пошуку пристроїв на шині; алгоритм пошуку адреси пристрою; алгоритм процедури отримання значень температури від датчиків; алгоритми перевірки CRC та розрахунку контрольної суми.

Також у [8] наведено програму, яка реалізує алгоритм роботи мовою С.

8.5.10. Моделювання мережі 1-WIRE

8.5.10.1 Схема моделі

Для моделювання мережі 1-WIRE використано пакет програмного забезпечення Proteus 8.6. Схему моделі наведено на рис. 8.77.

8.5.10.2. Інструкція користувача для роботи зі схемою

Для зміни температури на кожному з датчиків, використовуються кнопки під дисплеєм встановлення температури (рис. 8.78).

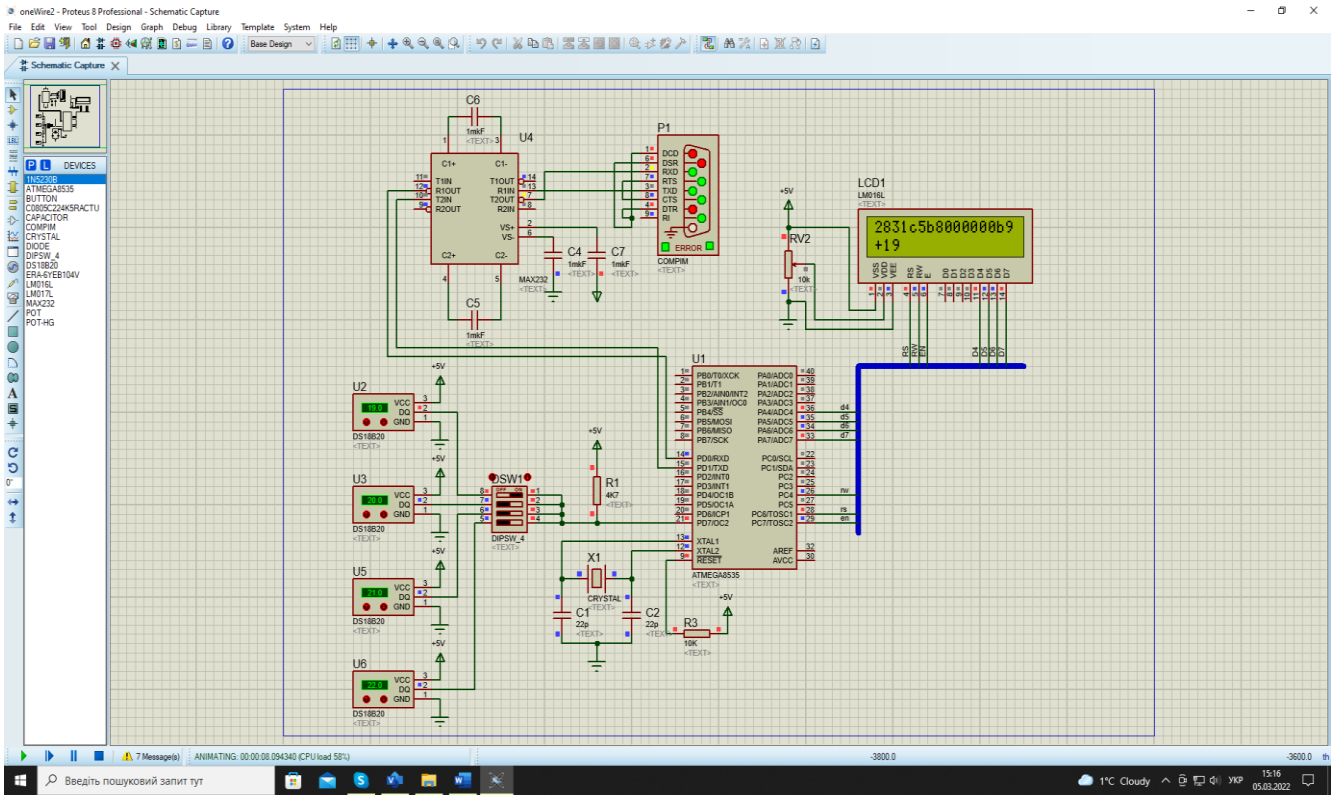


Рис. 8.77. Стан моделі під час підключення одного давача, який вимірює температуру 19 град

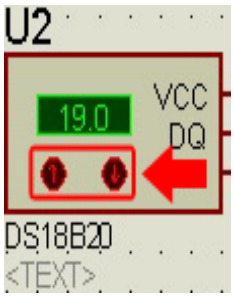


Рис. 8.78. Кнопки встановлення температури

Для режиму роботи із «паразитним живленням» необхідно VCC з’єднати зі спільним проводом (рис. 8.79).

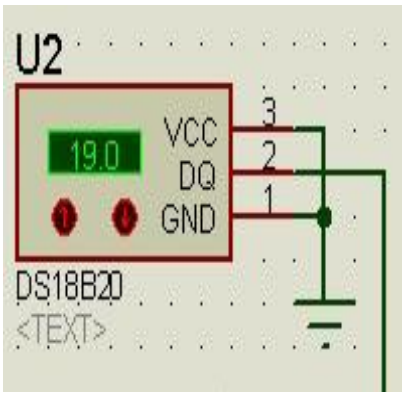


Рис. 8.79. Приклад датчика з «паразитним живленням»

Оскільки на схемі присутні кілька датчиків, то їх підключення або відключення здійснюється через «світч», який має 2 положення – «ON» та «OFF» відповідно. На рис. 8.80 наведено «світч», де всі 4 датчики підключено.

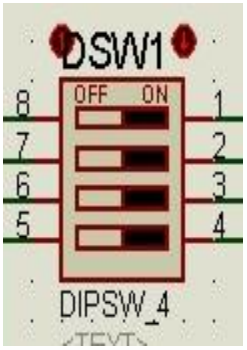


Рис. 8.80. Приклад «світча» з увімкнутими датчиками

8.5.10.3. Принцип роботи моделі

Датчики температури підключено до МК за допомогою опору R1. Кнопки імітують динамічне підключення датчиків до шини.

На старті роботи системи контролер опитує шину на наявність підключених датчиків. Якщо підключено хоча б один датчик, контролер починає опитувати датчики та виводити значення температури від кожного з них на LCD-дисплей. З виходу послідовного порту МК інформація подається на перетворювач рівнів MAX232, а далі надходить на роз'єм RS-232. Потім, наприклад, через модем інформація подається на віддалений пункт керування, де виконується подальша обробка значень температури програмою високого рівня.

Контрольні запитання та завдання

1. Дайте загальну характеристику інтерфейсу 1-WIRE.
2. Якою фірмою був розроблений інтерфейс 1-WIRE?
3. Які датчики мають інтерфейс 1-WIRE?
4. На яку відстань можуть передаватися дані в мережах 1-WIRE?
5. Скільки ведених пристроїв може бути в мережі 1-WIRE?
6. Назвіть основні застосування інтерфейсу 1-WIRE.
7. З якою швидкістю можуть передаватися повідомлення в мережі 1-WIRE?
8. Як ведеться передача сигналів у мережі 1-WIRE?
9. Наведіть та опишіть схему апаратної реалізації інтерфейсу 1-WIRE.
10. Опишіть, як підключається інтерфейс 1-WIRE в мережу?

11. Опишіть основні правила передачі даних у мережі 1-WIRE.
12. Опишіть обмеження застосування 1-WIRE-мереж.
13. Наведіть та опишіть структурну схему системи вимірювання температури з використанням давачів 1-WIRE.
14. Дайте загальну характеристику датчикам температури DS18B20.
15. Поясніть використання унікального 64-бітового послідовного двійкового коду адреси датчика DS18B20.
16. Наведіть та опишіть структурну схему датчика DS18B20.
17. Опишіть організацію пам’яті датчика DS18B20.
18. В якому діапазоні може вимірювати температуру DS18B20?
19. Як формується контрольна сума циклічного надлишкового коду?
20. Наведіть та поясніть формат коду датчика DS18B20.
21. На які групи поділяються команди датчика DS18B20?
22. Наведіть та опишіть загальні ROM-команди.
23. Опишіть послідовність пошуку пристроїв у мережі 1-WIRE.
24. Наведіть та опишіть функціональні команди.
25. Опишіть послідовність операцій для доступу до датчика DS18B20.
26. Наведіть та опишіть схему моделювання мережі 1-WIRE.

9. МОДУЛЬ АНАЛОГОВОГО КОМПАРАТОРА

9.1. Особливості архітектури аналогового компаратора

Компаратором називається електронний пристрій, який призначено для порівняння двох напруг. Залежно від форми подання порівнюваних сигналів компаратори поділяються на:

– аналогові компаратори;
– цифрові компаратори (ЦК) [1].

Нижче описано аналогові компаратори, які призначено для порівняння двох аналогових напруг, одна з яких виконує функцію еталонної – $U_{ЕТ}$, а інша порівнюється з еталонною – U_X .

Зазвичай аналоговий компаратор включає власне аналоговий компаратор і схему формування рівнів (рис. 9.1).

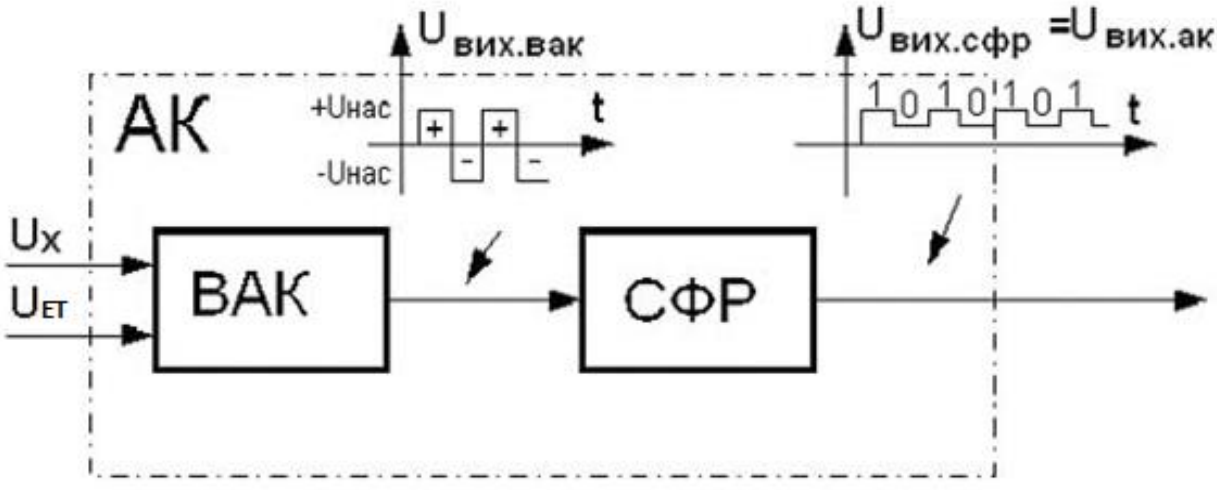


Рис. 9.1. Структура аналогового компаратора

Власне аналоговий компаратор може бути виконано, наприклад, на інтегральній мікросхемі операційного підсилювача [1]. Він виконує порівняння двох напруг: $U_{ЕТ}$ і U_X і, залежно від їх співвідношення, формує на виході дискретний сигнал, що приймає одне з двох значень: $+U_{НАС}$; $-U_{НАС}$.

Схема формування рівнів перетворює значення: $+U_{НАС}$; $-U_{НАС}$ у рівні цифрових сигналів ТТЛШ/КМОН-схем.

Значення $+U_{НАС}$ перетворюється в рівень логічної одиниці, а значення $-U_{НАС}$ – у рівень логічного нуля.

В електроніці застосовуються два варіанти схемного виконання аналогового компаратора:

– на інтегральній мікросхемі операційного підсилювача;
– на спеціалізованій мікросхемі аналогового компаратора.

9.2. Аналогові компаратори на інтегральній мікросхемі операційного підсилювача

Розглянемо схему аналогового компаратора, що виконує порівняння двох додатних напруг (рис. 9.2*а*). Одна напруга є еталонною: $(U_{ET} = const)$, а друга: (U_X) – повільно змінюється за законом, зображеним на рис. 9.2*в*.

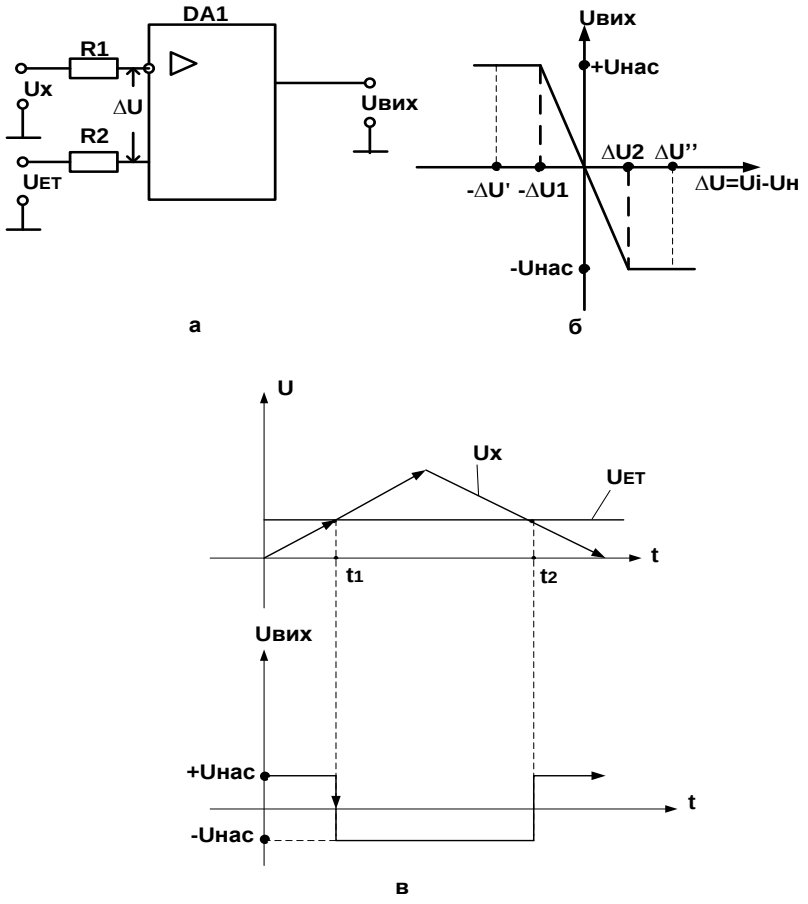


Рис. 9.2. Аналоговий компаратор для порівняння однополярних напруг:
а – схема; *б* – передатна характеристика інтегральнрї мікросхеми операційного підсилювача; *в* – часові діаграми

До моменту часу $t = t_1$ напруга $U_X < U_{ET}$. Потенціал входу інтегральної мікросхеми операційного підсилювача, що не інвертує (*H*), більш додатний, ніж потенціал входу, що інвертує (*I*).

У цьому випадку різниця напруг між входами *I* та *H*

$$\left|-\Delta U'\right| = \left|(U_I - U_H)\right| > \left|-\Delta U_1\right|. \qquad (9.1)$$

Відповідно до передатної характеристики інтегральної мікросхеми операційного підсилювача (рис. 9.2*б*) у цьому випадку $U_{ВИХ} = +U_{НАС}$. Після моменту часу $t = t_1$ вхідний сигнал $U_X \geq U_{ET}$.

З’являється різниця напруг між входами

$$\Delta U'' = (U_1 - U_H) > \Delta U_2. \qquad (9.2)$$

Відповідно до передатної характеристики інтегральної мікросхеми операційного підсилювача (рис. 9.2*б*) у цьому випадку $U_{\text{ВИХ}} = -U_{\text{НАС}}$.

На рис. 9.2*в* переключення аналоговрго компаратора відбувається миттєво, що є ідеальним випадком. Реально є невелика затримка між моментом досягнення рівності сигналів $U_{\text{Х}}$ та $U_{\text{ЕТ}}$ і моментом, коли вихідний сигнал $U_{\text{вих}}$ починає зменшуватися. Крім того, лінія, що відображає зменшення $U_{\text{вих}}$, проходить не перпендикулярно вісі часу, а під невеликим нахилом.

Після моменту часу $t = t_2$ напруга $U_{\text{ЕТ}}$ знову стає більшою ніж $U_{\text{Х}}$, вихідна напруга змінюється і приймає значення: $+U_{\text{НАС}}$.

9.3. Схема формування рівнів

Рівні напруг, що з’являються на виході імпульсних пристроїв, змінюються від мінімального (зазвичай дорівнює нулю) до максимального значення: $U_{\text{ВИХ.max}}$. З виходу імпульсних схем, виконаних на інтегральній мікросхемі операційного підсилювача, знімаються напруги, що набувають одне з двох значень: $+U_{\text{НАС}}$; $-U_{\text{НАС}}$. Наприклад, якщо напруги джерела живлення інтегральної мікросхеми операційного підсилювача дорівнюють $\pm 15B$, тоді $+U_{\text{НАС}} = +11B$, а $(-U_{НАС}) = -11B$.

Вихідні сигнали імпульсних пристроїв зазвичай обробляються цифровими ТТЛШ/КМОН-схемами, у яких діють тільки два значення (рівня) сигналів (напруг), які називаються:

– логічна одиниця (переважно високий рівень напруги);
– логічний нуль (зазвичай низький рівень напруги).

Наприклад, вихідна напруга низького рівня (логічний нуль) – приблизно 0,4...0,5 В, а вихідна напруга високого рівня (логічна одиниця) – від 2,4 В до напруги живлення $-E_{\text{ЖИВ}} = 5B$.

Виникає задача перетворення рівнів сигналів, що з’являються на виході імпульсних схем, у рівні цифрових сигналів, що діють у цифрових пристроях.

Цю задачу вирішують схеми формування рівнів або пристрої формування (перетворення) рівнів. Є різні варіанти схемної реалізації пристроїв формування рівнів [1]. Розглянемо один з них.

У цьому прикладі розглянуто схему формування рівнів, яку виконано на послідовному діодному ключі (рис. 9.3).

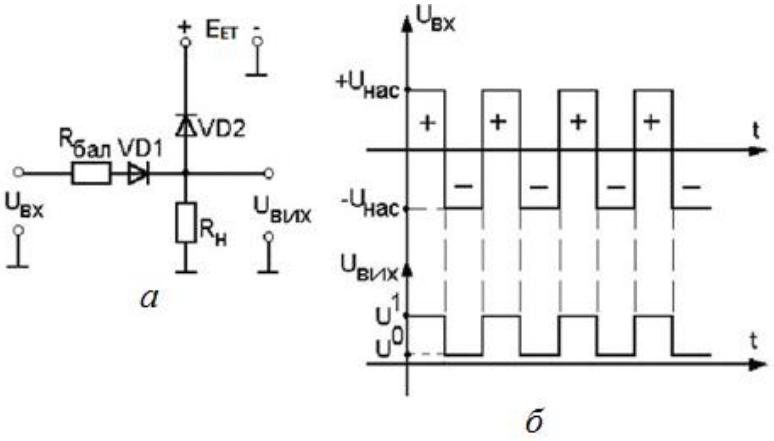


Рис. 9.3. Схема формування рівнів на послідовному діодному ключі:
а – схема; *б* – часові діаграми роботи

Пристрій перетворює:

– напругу $U_{\text{ВХ}} = +U_{\text{НАС}}$ у напругу $U_{\text{ВИХ}} = U^1 = U_{VD2.\text{ПР}} + E_{\text{ЕТ}}$;

– напругу $U_{\text{ВХ}} = -U_{\text{НАС}}$ у напругу $U_{\text{ВИХ}} = U^0 = I_{0.\text{VD2}} \cdot R_{\text{Н}}$,

де $I_{0.\text{VD2}}$ – зворотний струм насичення закритого діода VD2.

9.4. Аналоговий компаратор у AVR-мікроконтролерах

9.4.1. Загальні відомості про аналоговий компаратор

До МК сімейства AVR входить модуль аналогового компаратора [3]. Коли він увімкнений, компаратор дає можливість порівнювати значення напруг на двох відповідних виводах МК. Результатом порівняння є логічне значення нуль або одиниця, що може бути прочитано з програми. За результатом порівняння може бути згенероване переривання, а також відбутися захоплення стану таймера/лічильника Т1. Остання функція дозволяє вимірювати тривалості аналогових сигналів. Виводи, що використовуються компаратором, є контактами портів введення/виведення загального призначення (табл. 9.1).

Щоб вони могли використовуватися аналоговим компаратором, їх треба запрограмувати як входи (відповідний розряд регістра DDRx скинути у нуль). Крім того, записом нуля у відповідний розряд регістра PORTxn необхідно відключити внутрішні підтягуючі резистори.

У деяких МК є можливість відключення вхідних цифрових буферів у випадку, якщо контакти МК, що відповідають виводам AIN0 та AIN1, використовується тільки для введення аналогових сигналів [3]. При відключених цифрових буферах зменшується загальний струм споживання МК, а відповідні розряди регістра PINx завжди читаються як нуль.

Таблиця 9.1. **Виводи, що використовуються аналоговим компаратором**

Назва	**ATmega8x**	**ATmega8515x/8535x**	**ATmega16x/32x**	**ATmega64x/128x/165x**	**ATmega162x/163x**	**ATmega48x/88x/168x**	**ATmega164x/324x/644x**	**ATmega325x/3250x/645x /6450x**	**ATmega640x/1280x/1281x /2560x/2561x**	**Призначення**
AIN0	PD6	PB2	PB2	PE2	PB2	PD6	PB2	PE2	PE2	Неінвертуючий вхід
AIN1	PD7	PB3	PB3	PB3	PB3	PD7	PB3	PB3	PE3	Інвертуючий вхід

Відключення цифрових буферів на входах AIN0 та AIN1 здійснюється записом одиниці відповідно у розряди AIN0D та AIN1D регістра DIDR1, розміщеного за адресою ($7F). Формат цього регістра наведено на рис. 9.4.

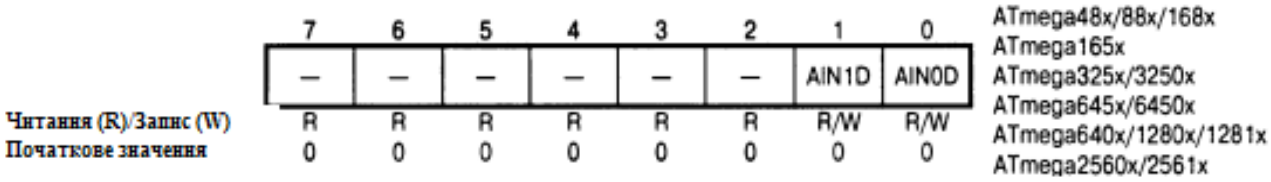


Рис. 9.4. Формат регістра DIDR1

9.4.2. Функціонування та програмування компаратора

Структурну схему аналогового компаратора наведено на рис. 9.5.

Керування компаратором і контроль його стану здійснюється за допомогою регістра ACSR, який залежно від моделі розміщено за адресою $08 ($28) або $30 ($50) (табл. 9.2). Формат цього регістра наведено на рис. 9.6, а призначення розрядів коротко описано в табл. 9.3.

Модуль АК AVR-мікроконтролера є звичайним компаратором (підрозд. 9.1).

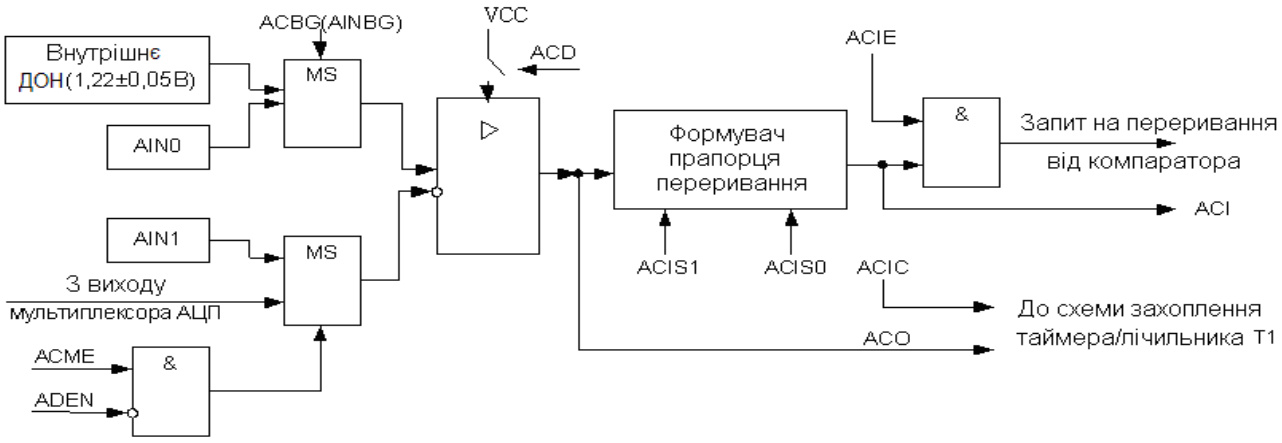


Рис. 9.5. Структурна схема аналогового компаратора

Таблиця 9.2. **Адреса регістра ACSR**

Модель	Адреса регістра ACSR
ATmega8515x/8535x	$08 ($28)
ATmega8x/16x/32x/64x/128x	$08 ($28)
ATmega48x/88x/168x	$30 ($50)
ATmega162x	$08 ($28)
ATmega164x/324x/644x	$30 ($50)
ATmega165x	
ATmega325x/3250x/645x/6450x	
ATmega640x/1280x/1281x/2560x/2561x	

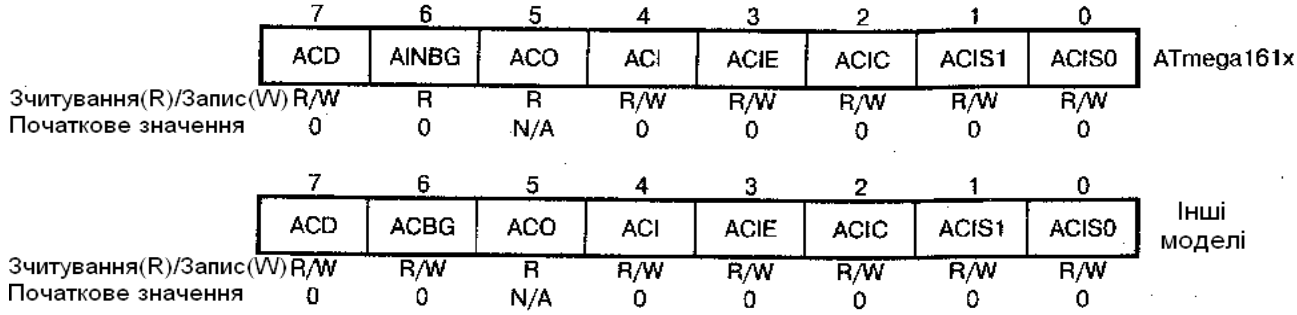


Рис. 9.6. Формат регістра ACSR

Якщо напруга на виводі AIN0 (неінвертуючий вхід) більш додатна (менш від'ємна) напруги на виводі AIN1 (інвертуючий вхід), то результат порівняння буде дорівнювати одиниці. У іншому випадку результат порівняння буде дорівнювати нулю. Цей результат (стан виходу компаратора) зберігається в розряді ACO регістра ACSR.

Таблиця 9.3. **Призначення розрядів регістра ACSR**

Розряд	Назва	Опис
7	ACD	Вимикання компаратора (нуль – увімкнено, одиниця – вимкнено)
6	ACBG (AINBG*)	Підключення до неінвертуючого входу компаратора внутрішнього джерела опорної напруги (нуль – не підключено, одиниця – підключено)
5	ACO	Результат порівняння (вихід компаратора)
4	ACI	Прапорець переривання від компаратора
4	ACIE	Дозвіл переривання від компаратора
2	ACIC	Підключення компаратора до схеми захоплення таймера/лічильника T1 (одиниця – підключено, нуль – не підключено)
1,0	ACIS1:ACIS0	Умова виникнення переривання від компаратора
*У моделях ATmega161x		

За вмикання і вимикання компаратора відповідає розряд ACD. Оскільки під час подачі напруги живлення всі розряди регістра ACSR скидаються в нуль, компаратор вмикається автоматично. Для вимикання компаратора розряд ACD слід встановити в одиницю. У разі зміни стану цього розряду переривання від компаратора слід заборонити.

Якщо стан виходу компаратора (розряд ACO) змінився заданим чином, встановлюється прапорець переривання ACIF регістра ACSR і генерується запит на переривання, якщо воно дозволено.

Як і для інших переривань, цей прапорець скидається апаратно під час запуску підпрограми обробки переривання або програмно, записом у нього одиниці. Для дозволу переривання необхідно встановити в одиницю розряд ACI регістра ACSR і прапорець I регістра SREG. Яка саме зміна стану виходу компаратора викликає переривання, визначається станом розрядів ACIS1:ACIS0 регістра ACSR відповідно до табл. 9.4.

Таблиця 9.4. **Умови генерації запиту на переривання від компаратора**

ACIS1	ACIS0	Умова
0	0	Будь-яка зміна стану виходу компаратора
0	1	Зарезервовано
1	0	Зміна стану виходу компаратора з одиниці на нуль
1	1	Зміна стану виходу компаратора з нуля на одиницю

Під час програмування цих розрядів переривання від компаратора має бути заборонено.

Крім генерації переривання, компаратор може керувати схемою захоплення таймера/лічильника T1. Для цього необхідно встановити в одиницю розряд ACIC регістра ACSR. У результаті вихід компаратора підключиться до схеми захоплення замість виводу ICP1 мікроконтролера.

Якщо розряд ACIC скинуто у нуль, компаратор відключено від блока захоплення таймера/лічильника T1. Компаратор може порівнювати сигнали, не тільки на виводах AIN0 і AIN1. Замість виводу AIN0 до неінвертуючого входу компаратора може бути підключено внутрішнє джерело опорної напруги величиною 1, 22 (±0,05) В. Для цього необхідно встановити в одиницю розряд ACBG (AINBG у моделях ATmega161x) регістра ACSR.

На інвертуючий вхід компаратора може надходити сигнал з виходу мультиплексора модуля АЦП. Для цього треба встановити в одиницю розряд ACME, який розміщено, залежно від моделі (табл. 9.5), або в регістрі спеціальних функцій SFIOR (третій розряд регістра), або в регістрі керування АЦП ADCSRB (шостий розряд) [3]. Модуль АЦП при цьому має бути вимкнено (розряд ADEN регістра ADCSRA треба скинути в нуль).

Таблиця 9.5. **Підключення мультиплексора АЦП до компаратора**

<table>
<tr><th>Модель</th><th>Регістр</th><th>Адреса</th></tr>
<tr><td>ATmega8515x/8535x</td><td>SFIOR</td><td>$30 ($50)</td></tr>
<tr><td>ATmega8x/16x/32x/64x/128x</td><td rowspan="2">SFIOR</td><td>$30 ($50)</td></tr>
<tr><td>ATmega48x/88x/168x</td><td>$20 ($40)</td></tr>
<tr><td>ATmega162x</td><td rowspan="6">ADCSRB</td><td rowspan="6">($7B)</td></tr>
<tr><td>ATmega164x/324x/644x</td></tr>
<tr><td>ATmega165x</td></tr>
<tr><td>ATmega325x/3250x/645x/6450x</td></tr>
<tr><td>ATmega640x/1280x/1281x/2560x/2561x</td></tr>
</table>

Основні параметри аналогового компаратора подано в [3].

Контрольні запитання та завдання

1. Який електронний пристрій називається компаратором?
2. На які види поділяються компаратори залежно від форми представлення порівнюваних сигналів?

3. Який електронний пристрій виконує функцію власне аналогового компаратора?
4. Поясніть роботу аналогового компаратора, який виконано на інтегральній мікросхемі операційного підсилювача.
5. Поясніть призначення та роботу схеми формування рівнів.
6. Наведіть та поясніть структуру модуля аналогового компаратора AVR-мікроконтролерів.
7. Наведіть та поясніть формати керуючих регістрів модуля аналогового компаратора.
8. Яким чином можна підключити мультиплексор АЦП до компаратора?
9. Від чого залежить стан виходу компаратора?
10. Де зберігається стан виходу компаратора?
11. Назвіть умови генерації запиту на переривання від компаратора.
12. Як мають бути сконфігуровані виводи, які використовуються аналоговим компаратором?
13. Які виводи МК використовуються компаратором?
14. Який розряд відповідає за вмикання та вимикання компаратора?
15. Як вихід компаратора підключити до схеми захоплення таймера/лічильника T1?
16. Наведіть та поясніть роботу схеми формування рівнів на послідовному діодному ключі.

10. СПЕЦІАЛЬНІ РЕЖИМИ РОБОТИ МІКРОКОНТРОЛЕРА

10.1. Тактування AVR-мікроконтролерів

10.1.1. Загальні відомості про тактування

Розглянемо виконання тактування AVR-мікроконтролерів на прикладі сімейства Mega [3; 4]. Тактові сигнали формує пристрій синхронізації, спрощену структуру якого наведено на рис. 10.1. Блок попереднього дільника і можливість підключення зовнішнього RC-ланцюга є не у всіх моделях.

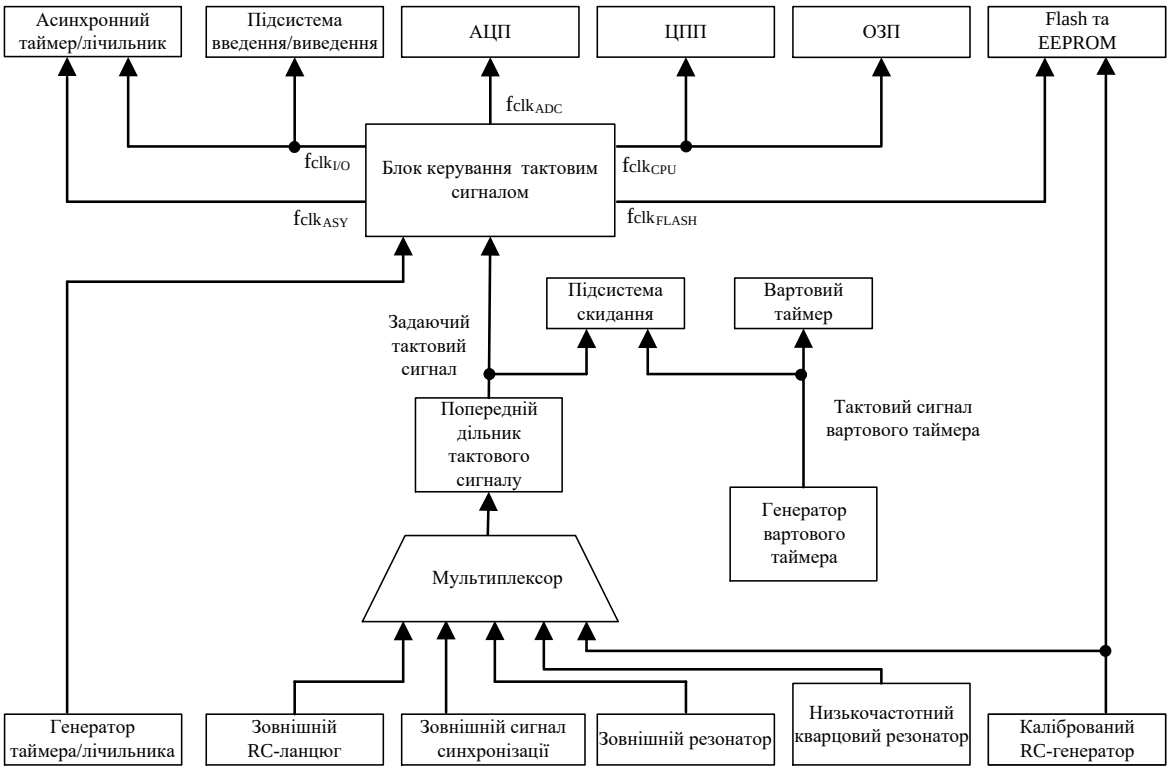


Рис. 10.1. Спрощена структура пристрою синхронізації

Для тактування можуть використовуватися різні джерела тактового сигналу. Передусім це вбудований генератор з під'єднуваним зовнішнім кварцовим/керамічним резонатором. Як тактовий може використовуватися найпростіший RC-генератор – як з внутрішнім (каліброваним), так і з зовнішнім RC-ланцюгом. Крім того, як тактовий може використовуватися сигнал від зовнішнього джерела.

Можливість використання того чи іншого джерела тактового сигналу залежить від моделі МК (табл. 10.1).

Таблиця 10.1. **Джерела тактового сигналу**

Джерело тактового сигналу	**ATmega8x/16x/32x/ 64x/128x**	**ATmega8515x**	**ATmega8535x**	**ATmega48x/88x/168x**	**ATmega162x**	**ATmega164x/324x/ 644x**	**ATmega165x**	**ATmega325x/3250x, ATmega645x/6450x**	**ATmega640x, ATmega1280x/1281x**
Кварцовий генератор	•	•	•	•	•	•	•	•	•
Генератор із зовнішнім RC-ланцюгом	•	•	•						
Внутрішній RC-генератор	•	•	•	•	•	•	•	•	•
Зовнішній сигнал синхронізації	•	•	•	•	•	•	•	•	•

На основі системного тактового сигналу формуються додаткові сигнали, що використовуються для тактування різних модулів і блоків МК (рис. 10.1):

– f_{clkCPU} – тактовий сигнал центрального процесора. Використовується для тактування блоків МК, що відповідають за роботу з ядром МК: регістровий файл, пам’ять даних і т. ін. У разі відключення цього сигналу центральний процесорний пристрій зупиняється і відповідно припиняються всі обчислення;

– $f_{clkI/O}$ – тактовий сигнал підсистеми введення/виведення. Використовується більшістю периферійних пристроїв, таких як таймери/лічильники та інтерфейсні модулі. Цей сигнал використовується також підсистемою зовнішніх переривань, проте низка зовнішніх переривань може генеруватися і за його відсутності;

– $f_{clkFLASH}$ – тактовий сигнал для Flash-пам’яті програм. Зазвичай цей сигнал активується та деактивується одночасно з тактовим сигналом центрального процесора f_{clkCPU};

– f_{clkASY} – тактовий сигнал асинхронного таймера/лічильника. Тактування здійснюється безпосередньо від зовнішнього кварцового резонатора, частотою 32768 Гц. Наявність цього сигналу дозволяє використовувати відповідний таймер/лічильник як годинник реального часу, навіть у разі знаходження МК у «Сплячому» режимі;

– f_{clkADC} – тактовий сигнал для модуля АЦП. Наявність цього тактового сигналу дозволяє здійснювати перетворення під час зупинених центральних процесорних пристрїв і підсистеми введення/виведення, при цьому значно зменшується рівень завад, що генерує МК, і відповідно збільшується точність перетворення.

Оскільки архітектура МК повністю статична, мінімально допустиму частоту нічим не обмежено (аж до покрокового режиму роботи), а максимальна робоча частота визначається конкретною моделлю МК.

Вибір режиму роботи тактового генератора здійснюється програмуванням конфігураційних комірок (FUSE Bits) CKSEL3...0. Необхідні значення для кожного режиму роботи наведено у [3; 4].

10.1.2. Генератор із зовнішнім резонатором

Резонатор підключається до виводів XTAL1 і XTAL2 мікроконтролерів МК, як показано на рис. 10.2.

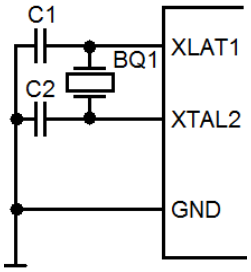


Рис. 10.2. Підключення кварцового чи керамічного резонатора

Ці виводи є відповідно входом і виходом інвертуючого підсилювача тактового генератора, який вбудовано у МК.

Конденсатори C1 і C2, які призначено для підвищення стабільності роботи генератора, підключаються між виводами резонатора та спільним проводом. Їх ємності залежать від типу резонатора. Для кварцових резонаторів ємності цих конденсаторів зазвичай знаходяться в межах 12...22 пФ, а для керамічних повинні вибиратися відповідно до рекомендацій виробників резонаторів. Конфігурування тактового генератора із зовнішнім резонатором у різних моделях розглянуто у [3; 4].

10.1.3. Низькочастотний кварцовий генератор

Цей режим призначено для використання низькочастотного кварцового резонатора – «годинникового кварцу», частотою 32768 Гц. Як і інші зовнішні резонатори, він підключається до виводів TOSC1 та TOSC2 МК.

У деяких моделях є внутрішні конденсатори, які можна підключити між виводами резонатора й спільним проводом [3; 4]. Ємність кожного з конденсаторів становить 10 пФ. В інших моделях необхідно використовувати зовнішні конденсатори.

10.1.4. Зовнішній сигнал синхронізації

Сигнал від зовнішнього джерела подається на вивід XTAL1, як показано на рис. 10.3.

Рис. 10.3. Підключення зовнішнього джерела тактового сигналу

Цей сигнал повинен задовольняти вимогам МК за частотою, шпаруватістю та рівнями напруги. Вивід XTAL2 залишають непідключеним.

У деяких моделях між виводом XTAL1 і спільним проводом можна включити внутрішній конденсатор ємністю 36 пФ [3; 4].

10.1.5. Генератор із зовнішнім RC-ланцюгом

Під час реалізації додатків, які не потребують високої часової точності, можна використовувати найпростіший RC-генератор, при цьому зовнішній RC-ланцюг підключається до виводу XTAL1, як показано на рис. 10.4.

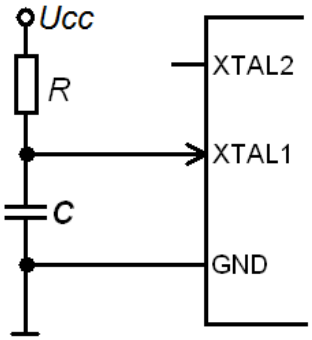


Рис. 10.4. Підключення зовнішнього RC-ланцюга

Ємність конденсатора ланцюга має бути не менше 22 пФ, а опір резистора рекомендовано вибирати з діапазону 3,3...100 кОм. Орієнтовно частоту сигналу генератора можна розрахувати за формулою [3; 4]:

$$f = \frac{1}{3 \cdot \mathrm{RC}}.$$

Зовнішній конденсатор у цих моделях можна відключити, задіявши внутрішній, ємністю 36 пФ.

Як і у випадку кварцового генератора, під час використання зовнішнього RC-ланцюжка тактовий генератор може працювати в чотирьох різних режимах, кожен з яких оптимізовано для певного діапазону частот.

10.1.6. Внутрішній калібрований RC-генератор

Використання вбудованого RC-генератора з внутрішнім часозадаючим RC-ланцюгом (внутрішнього RC-генератора) є найбільш економічним рішенням, оскільки при цьому не потрібні жодні зовнішні компоненти.

Номінальні частоти внутрішнього RC-генератора для деяких моделей наведено в табл. 10.2.

Таблиця 10.2. **Номінальні частоти внутрішнього RC-генератора**

Модель	**Частота [МГц]**	**Зауваження**
ATmega8515x/8535x	1,0; 2,0; 4.0; 8,0	За Ucc = 5,0 В, T = +25°C
ATmega8x/16x/32x/64x/128x	1,0; 2,0; 4,0; 8,0	
ATmega48x/88x/168x	8,0	За Ucc = 5,0 В, T = +25 °C
ATmegal62x	8,0	
ATmega 164x/324x/644x	8,0	
ATmega 165x/325x/3250x/645x/6450x	8,0	
ATmega640x/1280x/1281x/2560x/2561x	8,0	

Як видно з таблиці, в деяких моделях внутрішній RC-генератор може працювати на декількох фіксованих частотах. Робоча частота генератора цих моделей визначається вмістом конфігураційних комірок CKSEL3...0 [3; 4].

10.1.7. Керування тактовою частотою

У деяких моделях сімейства є можливість програмного зменшення частоти сигналу, що надходить від тактового генератора. Зрозуміло, що одночасно зі зменшенням тактової частоти зменшуються частоти сигналів f_{clkCPU}, $f_{clkI/0}$, $f_{clkFLASH}$, f_{c1kADC}, тобто сповільнюється робота всіх периферійних пристроїв МК. Якщо асинхронний таймер/лічильник працює в синхронному режимі, то відповідним чином змінюється і частота сигналу f_{clkASY}.

Для керування попереднім дільником тактового сигналу використовується один з регістрів введення/виведення.

Назва цього регістра, його адреса, та призначення окремих розрядів для різних моделей наведено в [3; 4].

10.2. Режими зниженого енергоспоживання AVR-мікроконтролерів

10.2.1. Загальні відомості про режими

AVR-мікроконтролери мають декілька режимів зниженого енергоспоживання, що називають SLEEP-режимом [3; 4]. Кожен з цих режимів дозволяє знизити енергоспоживання МК завдяки відключенню виконання окремих робочих функцій. Вхід у будь-який зі «сплячих» режимів виконується командою SLEEP. Під час виході виходу МК зі SLEEP-режиму виконання програми продовжується з місця зупинки.

Режими зниженого енергоживлення надають користувачеві широкі можливості щодо зменшення струму, який споживається кристалом. Під час переходу у SLEEP-режим виконання програми припиняється, а поновлюється у разі настання певних подій. У нових моделях, крім того, передбачено зниження енергоспоживання кристала відключенням тактових сигналів незадіяних периферійних модулів.

10.2.2. Керування режимами зниженого енергоспоживання

Мікроконтролери сімейства MEGA підтримують від 3 до 6 режимів зниженого енергоспоживання (табл. 10.3).

Таблиця 10.3. **Режими зниженого енергоспоживання**

Режими зниженого енергоспоживання	**ATmega8515x**	**ATmega8535x**	**ATmega8x**	**ATmega16x/32x/64x/128x**	**ATmega48x/88x/168x**	**ATmega162x**	**ATmega164x/324x/644x**	**ATmega165x, ATmega325x/3250x, ATmega645x/6450x**	**ATmega640x, ATmega1280x/1281x, ATmega2560x/2561x**
Idle	•	•	•	•	•	•	•	•	•
ADC Noise Reduction		•	•	•	•		•	•	•
Power Down	•	•	•	•	•	•	•	•	•
Power Save		•	•	•	•	•	•	•	•
Standby	•	•	•	•	•	•	•	•	•
Extended Standby		•		•		•	•		•

Режими відрізняються кількістю периферійних пристроїв МК, функціонуючих під час «сну», і відповідно ступенем зменшення енергоспоживання.

Залежно від моделі для керування «сплячим» режимом використовується різна кількість регістрів введення/виведення, які наведено в табл. 10.4.

Таблиця 10.4. **Регістри для керування «сплячим» режимом**

Назва	Опис	Адреса	Модель
MCUCR	Регістр керування МК	$35($55)	ATmega8x/16x/32x/64x/128x, ATmega8535x
MCUCR	Регістр керування МК	$35($55)	ATmega8515x, ATmegal62x
MCUCSR	Регістр керування і стану МК	$34($54)	
EMCUCR	Додатковий регістр керування МК	$36($56)	
SMCR	Регістр керування «сплячим» режимом	$33($53)	ATmega48x/88x/168x, ATmega 164x/324x/644x/l 65x, ATmega325x/3250x/645x/6450x, ATmega640x, ATmega1280x/1281x/2560x/2561x

Формати цих регістрів наведено у [3; 4].

Для керування «сплячим» режимом у МК сімейства, залежно від моделі, використовується два або чотири розряди керуючих регістрів. Призначення цих розрядів наведено в табл. 10.5.

Таблиця 10.5. **Призначення розрядів регістрів для керування «сплячим» режимом**

Розряд	Опис
SE	**Дозвіл переходу в режим зниженого енергоспоживання**. Встановлення цього розряду в одиницю дозволяє переведення МК у режим зниженого енергоспоживання. Перемикання здійснюється за командою SLEEP. Під час скинутого розряду SE виконання команди не робить жодних дій
SM2...SM0	**Вибір режиму зниженого енергоспоживання**. Стан цих розрядів визначає, в який режим перейде МК після виконання команди SLEEP (табл. 10.6).

Перехід у будь-який з режимів зниженого споживання здійснюється командою SLEEP, при цьому прапорець SE має бути встановлений в одиницю. Щоб уникнути ненавмисного перемикання МК у «сплячий» режим рекомендовано встановлювати цей прапорець безпосередньо перед виконанням команди SLEEP. Режим, в який перейде МК після виконання команди SLEEP, визначається станом розрядів SM2...SM0. Відповідність між вмістом цих розрядів і режимом зниженого енергоспоживання наведено в табл. 10.6.

Таблиця 10.6. **Вибір режиму зниженого енергоспоживання**

SM2	SM1	SM0	Режими
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power Down
0	1	1	Power Save
1	0	0	Зарезервовано
1	0	1	Зарезервовано
1	1	0	Standby[1)]
1	1	1	Extended Standby[*]
[*] Ці режими можна використовувати тільки під час роботи із зовнішнім резонатором			

Наявність того чи іншого режиму в конкретній моделі можна визначити за табл. 10.3. У разі відсутності в конкретної моделі МК будь-якого з режимів значення розрядів SM2...SM0, відповідні цьому режиму, є зарезервованими.

Вихід зі «сплячого» режиму може бути здійснено:

- у результаті переривання. Під час генерації переривання МК переходить у робочий режим, зупиняється на 4 такти, виконує підпрограму обробки переривання і відновлює виконання програми з інструкції, наступної за командою SLEEP. Вміст регістрів загального призначення, ОЗП і регістрів введення/виведення при цьому не змінюється;
- у результаті скидання. Після переходу МК у робочий режим керування передається за адресою вектора скидання.

Особливості окремих режимів зниженого енергоспоживання та рекомендації з їх застосування розглянуто у [3; 4].

10.3. Скидання AVR-мікроконтролерів

Переведення роботи МК у початковий стан виконується за допомогою «скидання». Останнє може бути викликано такими подіями:

- включення напруги живлення МК;
- подача сигналу НИЗЬКОГО рівня на вивід RESET (апаратне скидання);
- «тайм-аут» вартового таймера;
- падіння напруги живлення нижче заданої величини;
- скидання за інтерфейсом JTAG.

У разі настання будь-якої з перерахованих вище подій у всі регістри введення/виведення заносяться їх початкові значення, а в лічильник команд

завантажується значення адреси вектора скидання. За цією адресою має знаходитись одна з команд безумовного переходу RJMP або JMP. Значення адреси вектора скидання визначається станом конфігураційних комірок BOOTSZ1 і BOOTSZ0 [3; 4].

Структурну схему підсистеми скидання наведено на рис. 10.5.

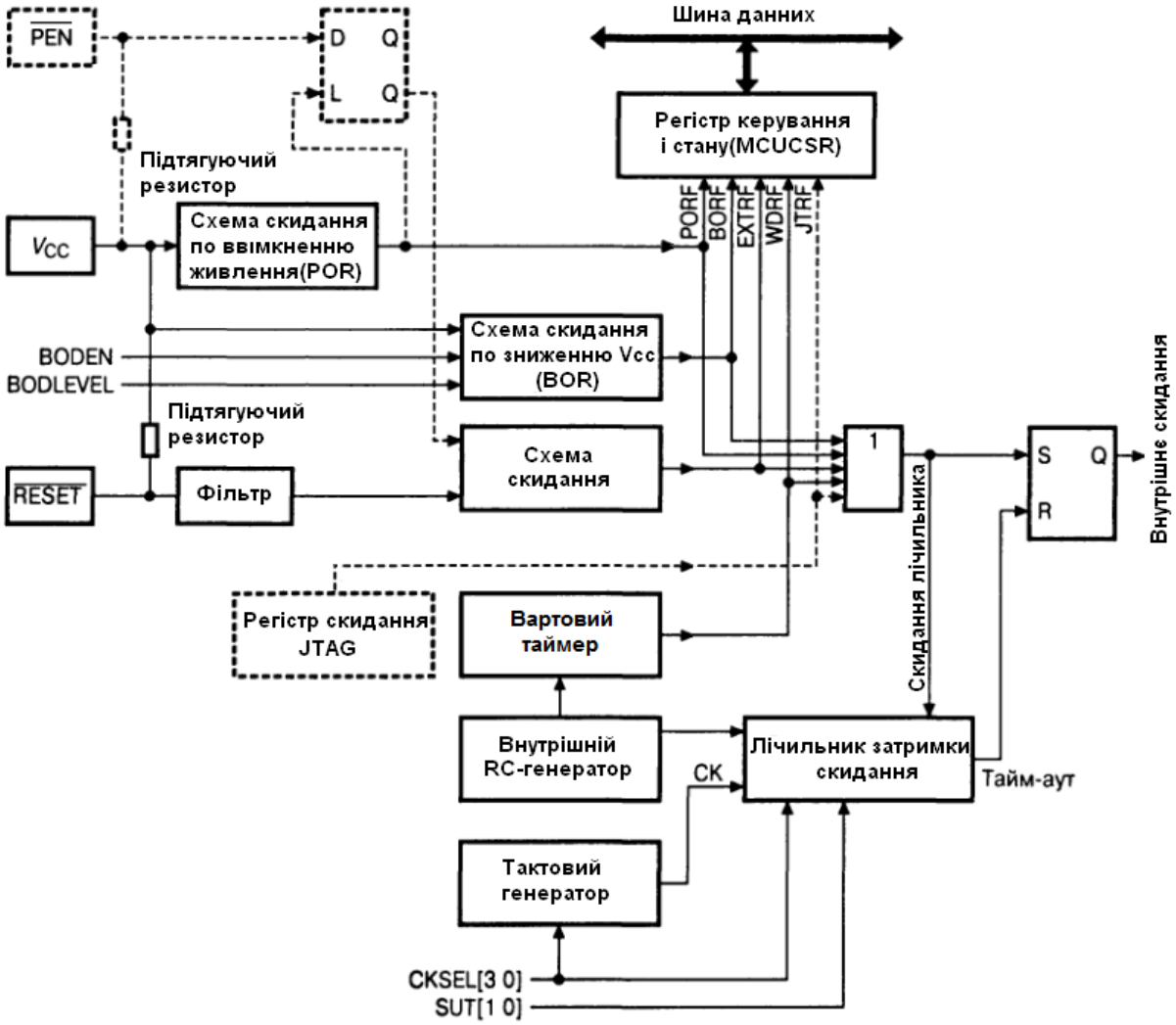


Рис. 10.5. Структурна схема підсистеми скидання

У деяких моделей відсутні елементи, які на рисунку виділено пунктиром.

Схема скидання працює таким чином. У разі настання події, що приводить до скидання МК, формується внутрішній сигнал скидання. Одночасно запускається таймер формування затримки скидання. Після закінчення певного проміжку часу внутрішній сигнал скидання знімається і починається виконання програми.

Усі МК сімейства дозволяють визначити подію, в результаті якої відбулося скидання пристрою. Залежно від моделі (табл. 10.7) для цього використовується або регістр керування і стану МК MCUCSR, який

розміщено за адресою $34 ($54), або регістр стану МК MCUSR, розміщений за тією самою адресою.

Таблиця 10.7. **Регістри для визначення джерела скидання**

Назва	Опис	Адреса	Модель
MCUCSR	Регістр керування і стану МК	$34 ($54)	ATmega8515x/8535x, ATmega8x/16x/32x/64x/128x, ATmegal62x
MCUSR	Регістр стану МК	$34 ($54)	ATmega48x/88x/168x, ATmegal64x/324x/644x, ATmegal65x/325x/3250x/645x/6450x, ATmega640x/1280x/1281x/2560x/2561x

Наведені в табл. 10.7 регістри містять набір прапорців, стан яких залежить від події, що викликала скидання пристрою. Формат регістрів MCUSR і MCUCSR, опис прапорців, які використовуються для визначення джерела скидання та опис окремих видів скидання наведено в [3; 4].

10.4. Самопрограмування AVR-мікроконтролерів

10.4.1. Загальна характеристика самопрограмування

Пам’ять МПС з початку виникнення останніх поділялася на:

- ПЗП, або ROM (англ. Read-Only Memory) – пам’ять тільки для читання;
- ОЗП, або RAM (англ. Random Access Memory) – пам’ять з довільним доступом.

У ПЗП зберігалися робочі програми, а в ОЗП – дані, які брали участь в операціях. ПЗП належить до енергонезалежної пам’яті, яка програмувалася на етапі створення МПС та в процесі роботи не змінювалася, тобто програма тільки читалася. ОЗП, яку виконано на тригерах (SRAM), належить до енергозалежної пам’яті [1]. Якщо зникає живлення такої пам’яті, то її зміст губиться.

У сучасних МПС в якості ОЗП крім SRAM використовується EEPROM-пам’ять.

Крім того, з’явилася можливість змінювати вміст ПЗП під час роботи МПС.

Сучасні AVR-мікроконтролери, наприклад, усі МК сімейства Mega та XMega мають можливість самопрограмування, тобто самостійної зміни

вмісту своєї пам’яті програм під час виконання програми [3]. Ця особливість дозволяє створювати на їхній основі дуже гнучкі системи, алгоритм роботи яких буде змінюватися самим МК залежно від відповідних внутрішніх умов або зовнішніх подій.

Для підтримки самопрограмування всю область пам’яті програм логічно розділено на дві секції – секцію прикладної програми (Application Section) і секцію завантажувача (Boot Loader Section) (рис. 10.6).

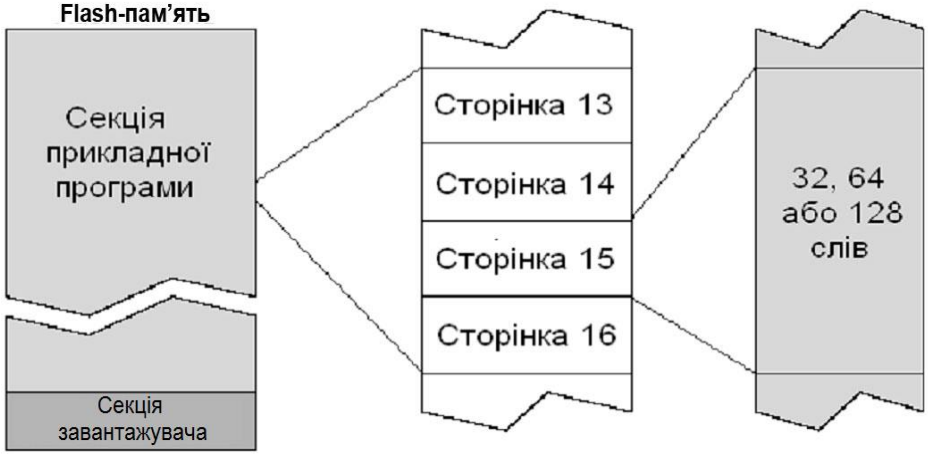


Рис. 10.6. Розділення пам’яті програм на дві секції

Програмування пам’яті програм здійснюється програмою-завантажувачем, яку розміщено в однойменній секції.

Розмір секції завантажувача, її розміщення в пам’яті програм та розмір секції прикладної програми у більшості МК визначається двома конфігураційними комірками BOOTSZ1:BOOTSZ0 [3; 4].

Для завантаження нового вмісту пам’яті програм, а також для вивантаження старого вмісту програма-завантажувач може використовувати будь-який інтерфейс передачі даних – USART/UART, SPI, TWI, наявний у конкретному МК.

Завантажувач може змінювати вміст обох секцій, що дозволяє модифікувати власний код та видаляти себе з пам’яті, якщо потреби в ньому не буде. Програму-завантажувача можна викликати з основної програми командами CALL/JMP, або перемістити вектор скидання на початок секції завантажувача.

У другому випадку запуск програми-завантажувача буде здійснюватися автоматично після кожного скидання МК. Положення вектора скидання визначається станом конфігураційної комірки BOOTRST. Якщо в ній зберігається одиниця, вектор скидання розміщується на початку пам’яті програм за адресою $0000. Для запрограмованої комірки, коли в ній

зберігається нуль, вектор скидання розміщується на початку секції завантажувача.

Пам’ять програм поділяють на дві області фіксованого розміру, які називають «читання під час запису» (Read-While-Write (RWW)) та «немає читання під час запису» (No Read-While-Write (NRWW)). У [3; 4] наведено розміри цих областей для деяких AVR-мікроконтролерів сімейства Mega та описано їх використання під час програмування МПС.

10.4.2. Керування процесом самопрограмування

Керування процесом програмування здійснюється командою SPM під час використання регістра введення/виведення SPMCR або SPMCSR. У всіх моделях, за винятком ATmega64x/128x, цей регістр розміщено за адресою $37 ($57). У МК ATmega64x/128x цей регістр розміщено у просторі додаткових регістрів введення/виведення за адресою $68. Формат цього регістра для різних моделей сімейства показано на рис. 10.7, а опис його розрядів наведено в табл. 10.8.

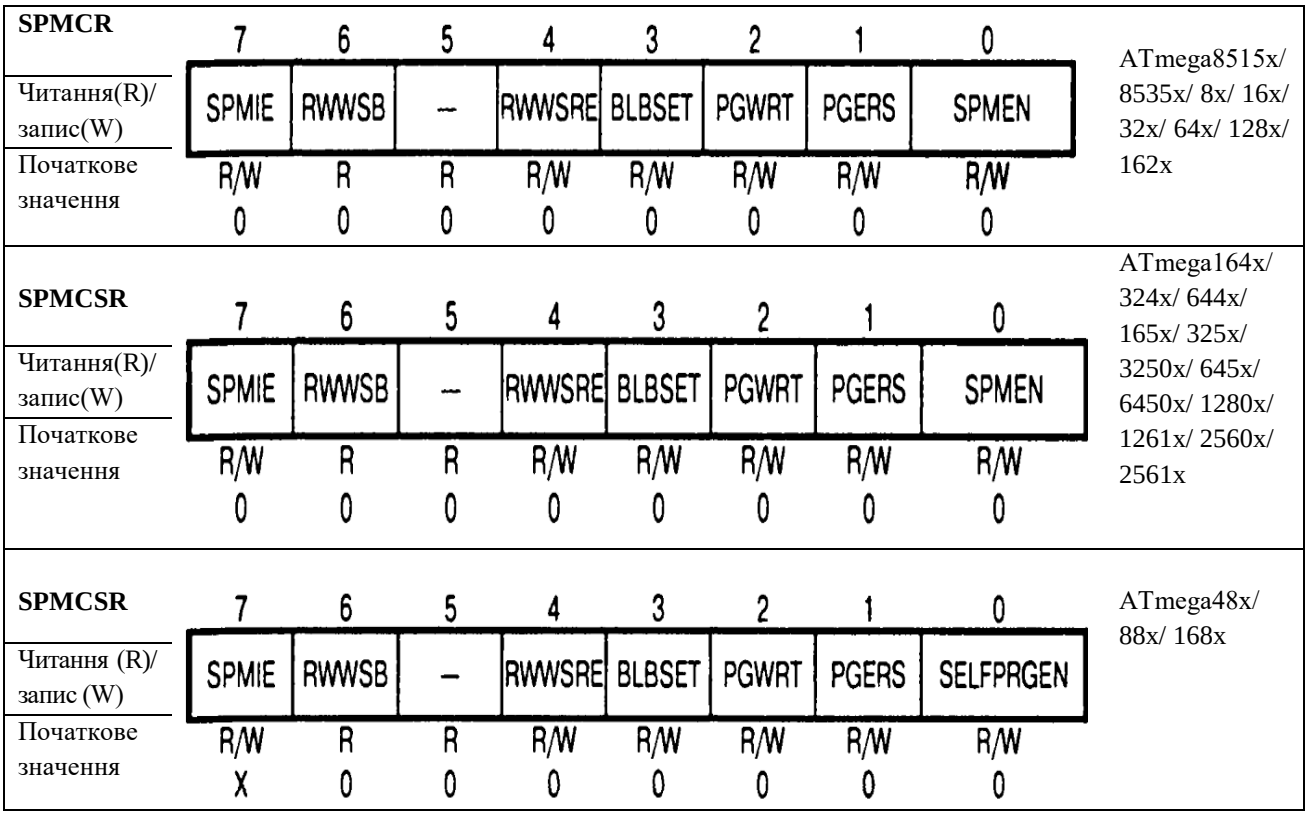


Рис. 10.7. Формат регістра SPMCR/SPMCSR

Таблиця 10.8. **Опис розрядів регістра SPMCR**

Розряд	Назва	Опис
7	SPMIE	**Дозвіл переривання SPM**. Якщо в цей розряд записано одиницю і також встановлено в одиницю прапорець I регістра SREG, то дозволяється переривання SPM. Переривання генерується увесь час, поки розряд SPMEN регістра скинуто в нуль
6	RWWSB	**Заборона доступу до області RWW**. Цей прапорець показує можливість звернення за адресами, розміщеними в області RWW. Якщо прапорець встановлено в одиницю, то доступ до області RWW заборонено. Якщо прапорець скинуто в нуль, то доступ дозволено. Встановлення цього прапорця здійснюється апаратно під час виконання операцій запису або очищення сторінки пам'яті. Прапорець можна скинути або програмно – записом одиниці у розряд RWWSRE по закінченні операції, або апаратно – під час запуску операції завантаження буфера сторінки
5	–	Зарезервовано, читається як нуль
4	RWWSRE	**Дозвіл читання області RWW**. Одночасне встановлення в одиницю цього розряду та розряду SPMEN дозволяє доступ до області RWW. Доступ до цієї області здійснюється командою SPM, запущеною протягом чотирьох машинних циклів після встановлення зазначених розрядів. Дозволити доступ до області RWW можна тільки після завершення операції програмування (після скидання прапорця SPMEN)
3	BLBSET	**Зміна комірок захисту завантажувача та прикладної програми**. У разі одночасного встановлення цього розряду та розряду SPMEN команда SPM, запущена протягом чотирьох машинних циклів, здійснить встановлення захисних комірок завантажувача та прикладної програми відповідно до вмісту регістра R0. Скидання розряду BLBSET здійснюється апаратно після встановлення комірок захисту або після закінчення зазначеного часу. За командою LPM, запущеною протягом трьох машинних циклів після встановлення зазначених розрядів, буде здійснено читання або комірок конфігурації, або комірок захисту (залежить від значення розряду Z0 регістра Z)
2	PGWRT	**Запис сторінки.** У разі одночасного встановлення цього розряду та розряду SPMEN команда SPM, запущена протягом чотирьох машинних циклів, запише сторінку пам'яті програм. Адресу сторінки необхідно завантажити у старші біти регістра-покажчика Z. Скидання розряду PGWRT здійснюється апаратно по закінченню запису сторінки або після закінчення відповідного часу. Під час запису в секцію NRWW центральний процесор зупиняється на час виконання операції*

Закінчення табл. 10.8

Розряд	Назва	Опис
1	PGERS	**Очищення сторінки**. У разі одночасного встановлення цього розряду та розряду SPMEN команда SPM, запущена протягом чотирьох машинних циклів, здійснить очищення сторінки пам'яті програм. Адресу сторінки попередньо необхідно завантажити у старші біти регістра-покажчика Z. Розряд PGERS скидається апаратно по закінченню очищення сторінки або після закінчення зазначеного часу. Під час запису в секцію NRWW центральний процесор зупиняється на час виконання операції*
0	SPMEN (SELFPRGEN)**	**Дозвіл виконання команди SPM**. Встановлення цього розряду дозволяє протягом чотирьох машинних циклів запуск команди SPM. Якщо розряд SPMEN встановлюється одночасно з одним з розрядів RWWSRE, BLBSET, PGWRT або PGERS, виконується операція, визначена цим розрядом Якщо встановлюється тільки розряд SPMEN, то вміст регістрів R1:R0 зберігається у тимчасовому буфері за адресою, що міститься в регістрі Z (молодший значущий розряд регістра ігнорується). Розряд SPMEN скидається апаратно після завершення операції або після закінчення зазначеного часу
* У моделях ATmega48x, процесор зупиняється у разі звернення до будь-якої КП програм		
** У моделях ATmega48x/88x/168x		

Запис значень у молодші п'ять розрядів регістра, відмінних від «10001», «01001», «00101», «00011» та «00001», ігнорується.

Для зміни вмісту пам'яті програм та комірок захисту завантажувача використовується команда SPM. Адреса області пам'яті попередньо завантажується в індексний регістр RAMPZ:Z, а дані, за необхідності, – у регістрову пару R1:R0.

Під час запису в EEPROM-пам'ять зміна вмісту регістра SPMCR неможлива. Тому, перед тим як записати будь-яке значення в регістр SPMCR, рекомендовано дочекатися закінчення запису в EEPROM-пам'ять – скидання прапорця EEWE регістра EECR.

Для адресації пам'яті програм під час використання команди SPM використовується індексний регістр Z, який є об'єднанням двох старших регістрів загального призначення R30 (молодший байт) і R31 (старший байт), а в моделях з об'ємом пам'яті вище 64 Кбайт – ще регістр введення/виведення RAMPZ. Оскільки пам'ять програм у МК сімейства Mega має сторінкову організацію (табл. 10.9), лічильник команд можна умовно розділити на дві частини.

Перша частина (молодші розряди) адресують комірки на сторінці, а друга частина визначає сторінку (рис. 10.8, табл. 10.10).

Рис. 10.8. Адресація пам’яті програм під час використання команди SPM

Таблиця 10.9. **Сторінкова організація пам’яті програм**

Характеристики пам’яті / Тип МК	Загальна ємність	Кількість сторінок	Кількість слів на сторінці
ATmega48x	$2^6 \cdot 2^5 = 2K$	$2^6 = 64$	$2^5 = 32$
ATmega8x ATmega8515x ATmega8535x ATmega88x	$2^7 \cdot 2^5 = 4K$	$2^7 = 128$	$2^5 = 32$
ATmega16x ATmega162x ATmega164x ATmega165x ATmega168x	$2^7 \cdot 2^6 = 8K$	$2^7 = 128$	$2^6 = 64$
ATmega32x ATmega324x ATmega325x ATmega3250x	$2^8 \cdot 2^6 = 16K$	$2^8 = 256$	$2^6 = 64$
ATmega64x ATmega640 ATmega645x ATmega6450x	$2^8 \cdot 2^7 = 32\ K$	$2^8 = 256$	$2^7 = 128$
ATmega128x ATmega1280x ATmega1281x	$2^9 \cdot 2^7 = 64K$	$2^9 = 512$	$2^7 = 128$
ATmega2560x ATmega2561x	$2^{10} \cdot 2^7 = 128\ K$	$2^{10} = 1024$	$2^7 = 128$

Таблиця 10.10. **Адресація пам’яті програм під час використання команди SPM**

Параметр	**ATmega48x**	**ATmega8515x/ 8535x/8x/88x**	**ATmega16x/ 162x/ 164x/165x/ 168x**	**ATmega32x/ 324x/ 325x/ 3250x**	**ATmega64x/ 640x/ 644x/ 645x/ 6450x**	**ATmega128x/ 1280x/ 1281x**	**ATmega2560x/ 2561x**
Розмір пам’яті програм, слів	2K	4K	8K	16K	32K	64K	128K
PCMSB	10	11	12	13	14	15	16
PCWORDMSB	4	4	5	5	6	6	6
ZPCMSB	11	12	13	14	15	16	17
ZWORDMSB	5	5	6	6	7	7	7
PCWORD	PC[4:0]	PC[4:0]	PC[5:0]	PC[5:0]	PC[6:0]	PC[6:0]	PC[6:0]
	Z5:Z1	Z5:Z1	Z6:Z1	Z6:Z1	Z7:Z1	Z7:Z1	Z7:Z1
PCPAGE	PC[10:5]	PC[11:5]	PC[12:6]	PC[13:6]	PC[14:7]	PC[15:7]	PC[16:7]
	Z11:Z6	Z12:Z6	Z13:Z7	Z14:Z7	Z15:Z8	Z16:Z8	Z17:Z8

Примітка. Біти Z17 та Z16 розміщуються у регістрі RAMPZ

Після запуску операції програмування вміст регістра Z фіксується і його можна використати для інших цілей. У командах LPM регістр Z може використовуватись для адресації окремих байтів пам’яті програм, при цьому старші розряди регістра Z, крім нульового, містять адресу потрібного слова пам’яті програм, а молодший біт – Z0 адресує молодший (Z0 = 0), або старший (Z0 = 1) байт обраного слова. Під час самопрограмування регістр Z буде використовуватись для адресації окремих слів у пам’яті програм, тому молодший біт регістра Z: Z0 = 0, а інші біти адресують слова.

Як приклад, у табл. 10.11 для МК Mega8515, який має 32 слова на сторінці, наведено у двійковому та десятковому кодах адреси окремих слів, та адреси молодших байтів.

Таблиця 10.11. **Адреси слів на сторінці та адреси молодшихбайтів**

Адреса слова на сторінці		**Адреса молодшого байта слова**	
Двійковий код	**Десятковий код**	**Двійковий код**	**Десятковий код**
00000	0	00000 0	0
00001	1	00001 0	2

Закінчення табл. 10.11

Адреса слова на сторінці		Адреса молодшого байта слова	
Двійковий код	**Десятковий код**	**Двійковий код**	**Десятковий код**
00010	2	00010 0	4
00011	3	00011 0	6
00100	4	00100 0	8
00101	5	00101 0	10
00110	6	00110 0	12
00111	7	00111 0	14
01000	8	01000 0	16
01001	9	01001 0	18
01010	10	01010 0	20
01011	11	01011 0	22
01100	12	01100 0	24
01101	13	01101 0	26
01110	14	01110 0	28
01111	15	01111 0	30
10000	16	10000 0	32
10001	17	10001 0	34
10010	18	10010 0	36
10011	19	10011 0	38
10100	20	10100 0	40
10101	21	10101 0	42
10110	22	10110 0	44
10111	23	10111 0	46
11000	24	11000 0	48
11001	25	11001 0	50
11010	26	11010 0	52
11011	27	11011 0	54
11100	28	11100 0	56
11101	29	11101 0	58
11110	30	11110 0	60
11111	31	11111 0	62

Як видно із цієї таблиці, адреси молодших байтів слів – парні.

10.4.3. Зміна вмісту пам’яті програм

10.4.3.1. Загальні відомості

Зміна вмісту пам’яті програм здійснюється в такій послідовності:

1. Заповнення тимчасового буфера сторінки новим вмістом.
2. Очищення попередньої сторінки.
3. Перенесення вмісту буфера у пам’яті програм.

Слід зазначити, що очищення сторінки може виконуватися як після заповнення буфера, так і перед його заповненням. Однак, якщо необхідно змінити тільки частину сторінки, наведений порядок дій є єдино можливим. У цьому випадку вміст комірок, які не потребують зміни, зберігається в буфері перед очищенням сторінки.

Для визначення моменту закінчення виконання SPM-операцій можна або опитувати стан прапорця SPMEN регістра SPMCR, чекаючи на його скидання, або скористатися перериванням «Готовність SPM».

Це переривання генерується увесь час, поки прапорець SPMEN скинуто. У цьому разі таблиця векторів переривань повинна міститися в секції завантажувача, а переривання необхідно дозволити встановленням прапорця SPMIE регістра SPMCR.

Для очищення сторінки пам’яті програм необхідно занести адресу сторінки в регістр Z (секція ZPAGE) (рис. 10.8), записати значення «x0000011» у регістр SPMCR і протягом чотирьох машинних циклів виконати команду SPM, вміст регістрів R1 та R0 при цьому ігнорується. В МК з кількістю сторінок 512 або 1024 для адресації використовуються додаткові молодші біти регістра RAMPZ: Z16 = RAMPZ0, Z17 = RAMPZ1 [3; 4].

Для занесення слова команди в буфер слід завантажити адресу комірки в регістр Z (секція ZWORD) (рис. 10.8), а слово команди – у регістри R1:R0. Після цього необхідно записати значення «x0000001» у регістр SPMCR і протягом чотирьох машинних циклів, виконати команду SPM.

Очищення буфера здійснюється автоматично по закінченню запису сторінки в пам’ять або вручну, записом одиниці у розряд RWWSRE регістра SPMCR. Зазначимо, що запис за тією самою адресою в буфері неможливий без його очищення.

Запис вмісту буфера до пам’яті програм здійснюється аналогічно. У регістр Z (секція ZPAGE) заноситься адреса сторінки, у регістр SPMCR записується значення «x0000101» і протягом чотирьох машинних циклів виконується команда SPM. Вміст регістрів R1 та R0 при цьому ігнорується.

Нижче наведено більш детальний описаних вище дій.

10.4.3.2. Заповнення тимчасового буфера сторінки новим вмістом

Перед записом нових даних у сторінку необхідно спочатку завантажити тимчасовий буфер сторінок. Тимчасовий буфер сторінок – це доступний тільки для запису окремий буфер, що не міститься в ОЗП, і який містить одну тимчасову сторінку. Слова в нього повинні вноситися послідовно. Перезапис буфера сторінок у Flash-пам’ять відбувається за одну операцію.

Для вибору адреси слова, що буде записано в буфер, використовується регістр Z. Молодший значущий розряд регістра Z ігнорується (містить нуль), оскільки запис двох байтів слова відбувається за одну операцію. Під час заповнення буфера сторінок ігноруються старші біти регістра Z, які не використовуються. Як приклад, структуру регістра Z для МК із 32-слівними (64-байтовими) сторінками показано на рис. 10.9.

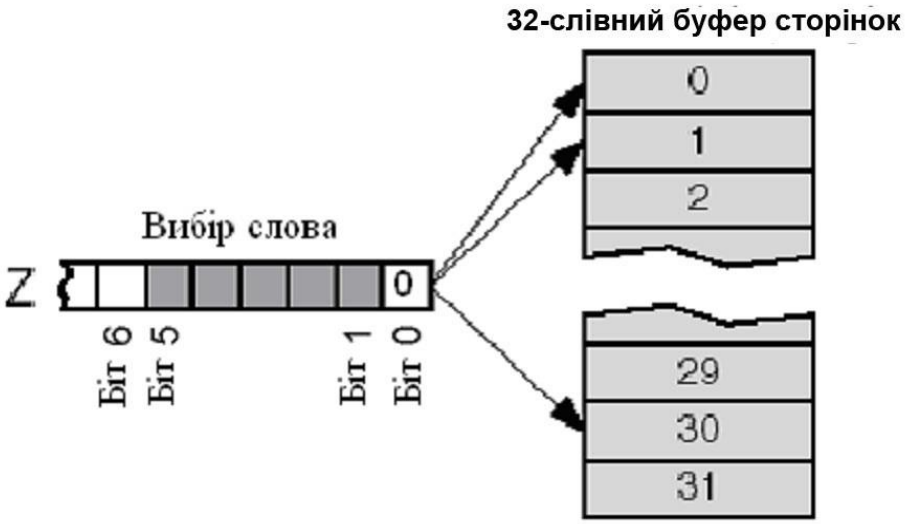


Рис. 10.9. Запис слова у буфер сторінок

МК, що мають більші розміри сторінки для вибору слів використовують більше біт. Наприклад, ATmega128 для адресації слів використовує молодші біти Z1...Z7, тому що одна сторінка містить $2^7 = 128$ слів ($2^8 = 256$ байт).

Щоб записати слово в буфер сторінок необхідно завантажити його в регістри R1:R0 та встановити біт SPMEN у регістрі SPMCR. Протягом чотирьох машинних циклів після цього необхідно виконати команду SPM.

10.4.3.3. Оновлення Flash-пам’яті

Flash-пам’ять оновлюється сторінками. Перед записом нових даних сторінку необхідно стерти.

Для вибору сторінки, що підлягає стиранню, використовується регістр Z (секція ZPAGE) (рис. 10.8). Він призначений для вказання номера сторінки, яка стирається. Молодші біти, що вибирають слово на сторінці, ігноруються. Наприклад, у МК, що має розмір сторінки 32 слова (64 байти), ігноруються шість молодших бітів регістра Z.

Для того щоб стерти сторінку необхідно встановити біти PGERS та SPMEN у регістрі SPMCR і протягом чотирьох машинних циклів виконати команду SPM.

Після завантаження даних у тимчасовий буфер сторінок його необхідно записати у Flash-пам’ять. Для цього необхідно записати номер сторінки, яка переноситься, в регістр Z.

Потім встановлюються біти PGWRT та SPMEN у регістрі SPMCR, і протягом чотирьох машинних циклів необхідно виконати команду SPM, вміст регістрів R1:R0 при цьому ігнорується.

Як приклад використання регістра Z для МК, сторінки якого містять 32 слова (64 байти), показано на рис. 10.10.

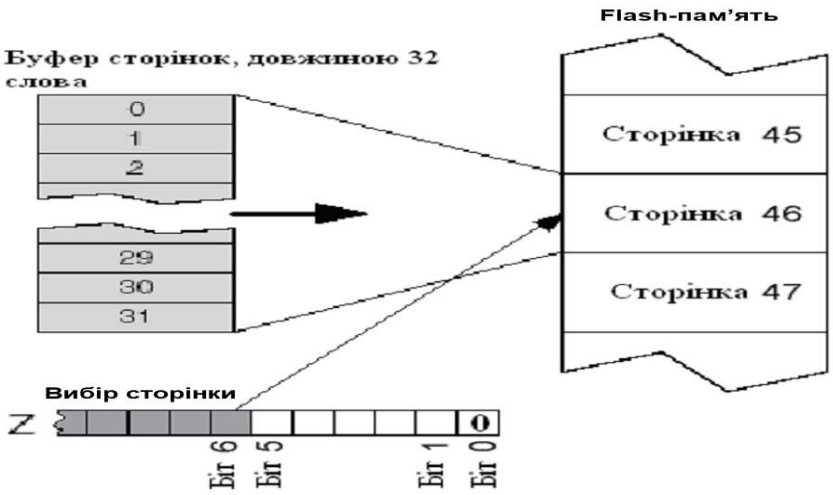


Рис. 10.10. Запис сторінки у Flash-пам’ять

Для визначення готовності МК до наступного оновлення необхідно опитувати біт SPMEN.

10.4.3.4. Зміна комірок захисту завантажувача та прикладної програми

Для захисту завантажувача і прикладної програми можуть використовуватися комірки регістра R0: BLB12:BLB11 та BLB02:BLB01 (рис. 10.11).

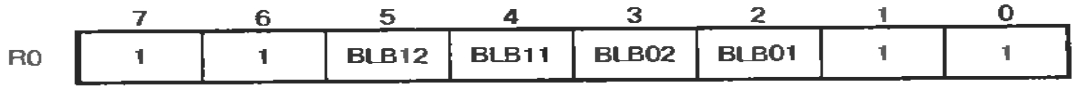


Рис. 10.11. Комірки захисту завантажувача та прикладної програми

Програмуванням цих комірок можна заборонити доступ із прикладної програми до програми завантажувача та забезпечити завантажувачу можливість доступу до секції прикладної програми.

Для програмування комірки треба скинути відповідний розряд цього регістра. Для цього необхідне значення завантажується в регістр R0, потім у регістр SPMCR записується значення «x0001001» і протягом чотирьох машинних циклів виконується команда SPM [3; 4]. Вміст регістра Z при цьому ігнорується, однак для сумісності з майбутніми пристроями рекомендовано записувати в нього значення $0001. Під час програмування комірок захисту можна звертатися до будь-якої ділянки пам'яті програм.

10.4.3.5. SPM-переривання

У всіх МК, що підтримують режим самопрограмування, за допомогою переривань можна контролювати процес оновлення Flash-пам'яті. Встановлення біта SPMIE у регістрі SPMCR дозволить формувати SPM-переривання, яке може використовуватися для відстеження закінчення режиму SPM.

10.4.3.6. Конфлікти EEPROM-пам'яті

Перед виконанням SPM-команди необхідно закінчити всі операції з EEPROM-пам'яттю. Запис/очищення Flash- і EEPROM-пам'яті не може відбуватися одночасно.

10.4.4. Типові процедури оновлення Flash-пам'яті

На рис. 10.12 показано дві стандартні схеми алгоритмів оновлення Flash-пам'яті.

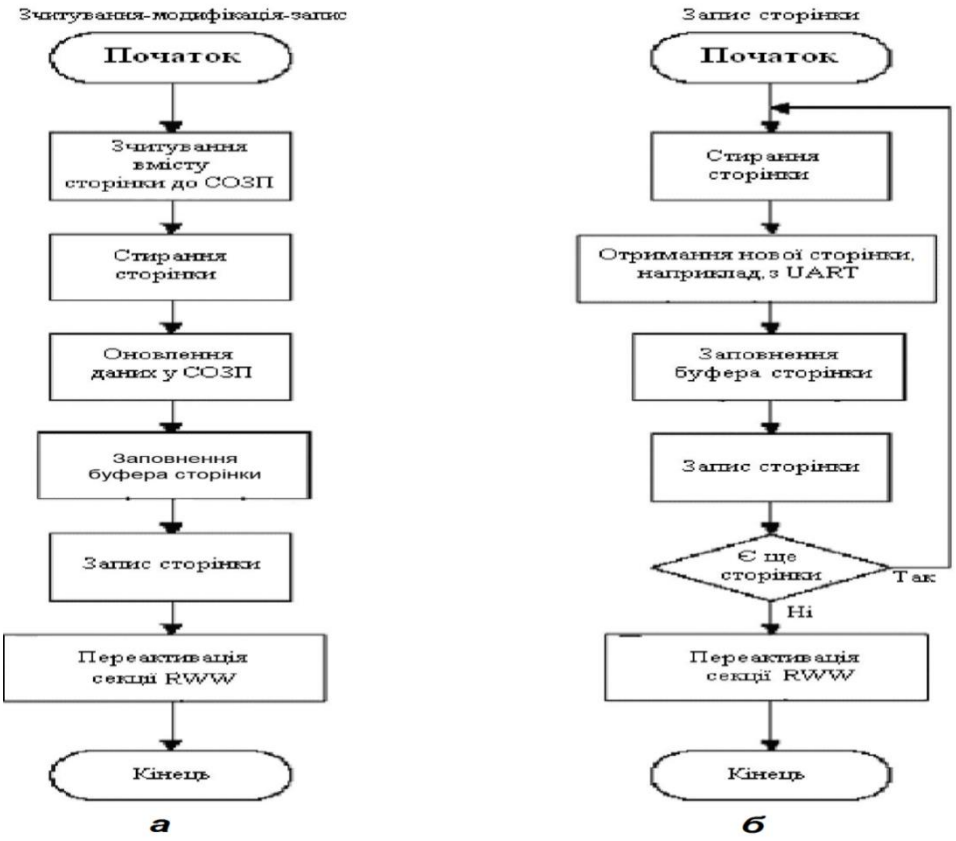
Схема ліворуч (рис. 10.12*а*) описує алгоритм «зчитування–модифікації–запису» невеликих частин Flash-пам'яті, наприклад, констант. Схема праворуч (рис. 10.12*б*) описує алгоритм запису сторінки без попереднього зчитування її вмісту, наприклад, виконується запис даних, що надійшли від UART.

10.4.5. Режими захисту Flash- та EEPROM-пам'яті

Вміст Flash- та EEPROM-пам'яті можна захистити від запису та читання за програмуванням комірок захисту (Lock Bits) LB1 та LB2 [3; 4].

10.4.6. Читання конфігураційних комірок і комірок захисту

Крім програмування МК, завантажувач може також зчитувати вміст конфігураційних комірок та комірок захисту [3; 4].



Примітка. Необхідні дії для переактивації секції RWW описано у [3; 4].

Рис. 10.12. Схеми алгоритмів оновлення Flash-пам’яті

Контрольні запитання та завдання

1. Як називається пристрій, який формує тактові сигнали для МК?
2. Які джерела тактового сигналу можуть використовуватись МК сімейства Mega?
3. Як здійснюється підключення кварцового резонатора?
4. Для чого призначено зовнішні конденсатори C1 і C2 у схемі підключення кварцового резонатора?
5. Як здійснюється підключення до генератора зовнішнього RC-ланцюга?
6. Як здійснюється підключення зовнішнього джерела тактового сигналу?
7. Як вирахувати необхідний опір резистора під час використання генератора з зовнішнім RC-ланцюгом?
8. Поясніть особливості використання внутрішнього каліброваного RC-генератора.
9. Від чого залежить можливість використання того чи іншого джерела тактового сигналу?
10. Які пристрої МК використовують тактовий сигнал $f_{clkI/O}$? Як він формується?

11. Чим обмежено мінімально допустиму тактову частоту МК?
12. За якою командою здійснюється перехід у «сплячий» режим?
13. Скільки режимів зниженого енергоспоживання підтримують МК сімейства Mega?
14. Чим відрізняються між собою режими зниженого енергоспоживання?
15. Як здійснюється вибір режиму зниженого енергоспоживання?
16. Які біти використовуються для керування «сплячим» режимом?
17. Завдяки чому можна знизити енергоспоживання МК?
18. Як може бути здійснено вихід зі «сплячого» режиму?
19. Які види скидання можна використовувати у МК AVR-сімейства?
20. Що має знаходитись за адресою вектора скидання?
21. Наведіть структуру схеми скидання та поясніть її роботу.
22. Яке значення завантажується в лічильник команд під час скидання?
23. Як можна визначити подію, в результаті якої відбулося скидання пристрою?
24. Поясніть, що таке самопрограмування МК та для чого воно використовується.
25. Як логічно розділено всю область пам’яті програм для підтримки самопрограмування?
26. Яким чином відбувається керування процесом самопрограмування МК?
27. Наведіть алгоритм зміни вмісту пам’яті програм.
28. Опишіть процес очищення сторінки пам’яті.
29. Поясніть, у чому полягає особливість використання переривань під час самопрограмування.
30. Як може виконуватись завантаження нового вмісту пам’яті програм у МК?
31. Чим визначається розмір секції завантажувача та розмір секції прикладної програми?
32. Опишіть призначення окремих розрядів регістра SPMCR.
33. Як відбувається заповнення тимчасового буфера сторінки новим вмістом?
34. Яким чином новий вміст пам’яті програм під час самопрограмування потрапляє в МК?
35. Поясніть призначення та використання SPM-переривання.

СПИСОК ЛІТЕРАТУРИ

Основна

1. Новацький А. О. Комп'ютерна електроніка [Електронний ресурс]: підруч. для студ. спец. 126 «Інформаційні системи та технології», спеціалізації «Інтегровані інформаційні системи» / А. О. Новацький.– Електронні текстові дані (1 файл: 80,9 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2018. – 468 с.
2. Новацький А. О. Мікропроцесорні та мікроконтролерні системи: підруч. У 2 ч. Ч. 1. Мікропроцесорні системи [Електронний ресурс]/ А. О. Новацький. – Електронні текстові дані (1 файл: 43,8 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2019. – 367 с.
3. Новацький А. О. Мікропроцесорні та мікроконтролерні системи: підруч. для студ. освітньої програми «Інтегровані інформаційні системи» за спец. 126 «Інформаційні системи та технології». У 2 ч. Ч.2. Проектування мікропроцесорних систем [Електронний ресурс] / А. О. Новацький. – Електронні текстові дані (1 файл: 20,3 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2020. – 460 с.
4. Евстифеев А. В. Микроконтроллеры AVR семейства Mega. Руководство пользователя / А. В. Евстифеев. – М. : Изд. дом «Додэка-XXI», 2007. – 592 с.
5. Мікропроцесорні та мікроконтролерні системи : лаб. практикум [Електронний ресурс] : навч. посіб. для студ. освітньої програми «Інтегровані інформаційні системи» спец. 126 «Інформаційні системи та технології» / Уклад. А. О. Новацький. – Електронні текстові дані (1 файл: 18,97 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 415 с.
6. Новацький А. О. Проектування вбудованих систем : лаб. практикум [Електронний ресурс] : навч. посіб. для студ. освітньої програми «Інтегровані інформаційні системи» спец. 126 «Інформаційні системи та технології» / А. О. Новацький, В. М. Шимкович. – Електронні текстові дані (1 файл: 34,22 Кбайт). – Київ : КПІ ім. Ігоря Сікорського, 2022.

Допоміжна

7. Новацький А. О. Проектування мікропроцесорних систем та мереж : навч. посіб. для студ. спец. 8.050201.01 «Комп'ютеризовані системи управління та автоматика» / А. О. Новацький. – Київ : НТУУ «КПІ», 2016.

8. Новацький А. О. Проектування та програмування мікропроцесорних систем і мереж: проектування мережі 1-WIRE : навч. посіб. для студ. спец. 7.05020101, 8.05020101 «Комп'ютеризовані системи управління та автоматика» / А. О. Новацький. – Київ : НТУУ «КПІ», 2014.
9. Сван Т. Освоение Turbo Assembler / Т. Сван. – 2-е изд. – Киев : СПб. : Диалектика, 1996.
10. Новиков Ю. В. Разработка устройств сопряжения для персонального компьютера типа IBM PC : практ. пособие / Ю. В. Новиков, О. А. Калашников, С. Э. Гуляев. – М. : ЭКОМ, 1997.
11. Официальное описание микроконтроллеров XMEGA [Электронный ресурс]. – [Режим доступа] : http://www.gaw.ru/html.cgi/txt/ic/Atmel/micros/avr_xmega/start.htm
12. Опис CAN-протоколу [Електронний ресурс]. – [Режим доступу] : http://www.itt-ltd.com/reference/ref_can.html
13. Схеми та пояснення роботи CAN-контролерів та трансиверів [Електронний ресурс]. – [Режим доступу] : http://atmel.com
14. CAN-трансивер [Електронний ресурс]. – [Режим доступу] : https://www.nxp.com/docs/en/data-sheet/PCA82C250.pdf
15. CAN-трансивер [Електронний ресурс]. – [Режим доступу] : https://www.nxp.com/docs/en/data-sheet/MC33388.pdf
16. RS-232 [Електронний ресурс]. – [Режим доступу] : https://uk.wikipedia.org/wiki/RS-232.
17. Васильев В. И. Электронные промышленные устройства : учеб. для студ. вузов / В. И. Васильев, Ю. М. Гусев, В. Н. Миронов. – М. : Высш. шк., 1988.

ПРЕДМЕТНИЙ ПОКАЖЧИК

Електронне навчальне видання

Новацький Анатолій Олександрович

ЕЛЕКТРОНІКА ТА МІКРОПРОЦЕСОРНА ТЕХНІКА

Частина 2

Мікропроцесорні системи

Підручник

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Свідоцтво про державну реєстрацію: серія ДК № 5354 від 25.05.2017 р.
просп. Перемоги, 37, м. Київ, 03056

Формат 60/84$^{1}/_{8}$. Гарнітура Times. Ум. друк. арк. 57,19. Обл.-вид. арк. 29,06.
Поз. 23-1-2-001. Зам. № 23-009.

КПІ ім. Ігоря Сікорського, Видавництво «Політехніка»,
вул. Політехнічна, 14, корп. 15
м. Київ, 03056
тел. (044) 204-81-78