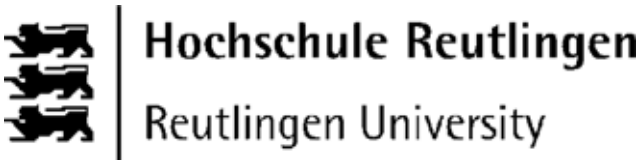
**МІНІСТЕРСТВО ОСВІТИ І НАУКИ,
МОЛОДІ ТА СПОРТУ УКРАЇНИ**

**ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«НАЦІОНАЛЬНИЙ ГІРНИЧИЙ УНІВЕРСИТЕТ»**



**В.В. Ткачов, Г. Грулер, Н. Нойбергер,
С.М. Проценко, М.В. Козарь**

МІКРОПРОЦЕСОРНА ТЕХНІКА

Навчальний посібник

Дніпропетровськ
НГУ
2012

УДК 004.31 (075.8)
ББК 32.973.26-04я73
М-59

Рекомендовано Міністерством освіти і науки, молоді та спорту України як навчальний посібник для студентів вищих навчальних закладів напряму підготовки «Автоматизація та комп'ютерно-інтегровані технології»
(лист №1/11-6422 від 08.05.2012 р.)

Рецензенти:
С.А. Положаєнко, д-р техн. наук, професор, завідувач кафедри «Комп'ютеризовані системи управління» (Одеський національний політехнічний університет);
В.Я. Копп, д-р техн. наук, професор, завідувач кафедри «Автоматизованих прикладних систем» (Севастопольський національний технічний університет), заслужений діяч науки і техніки України

Мікропроцесорна техніка [Текст]: навч. посібник/В.В. Ткачов, Г. Грулер,
М-59 Н. Нойбергер та ін. – Д.: Національний гірничий університет, 2012. –
188 с.

ISBN 978-966-350-359-2

Розглянуто питання організації архітектури мікроконтролерів, сигналів керування та системи команд однокристального мікроконтролера МК-51 (К1816ВЕ51), організації режимів преривань, тимчасових затримок, передачі інформації в послідовному форматі. Наведено приклади організації систем керування об'єктами з неперервними і дискретними характеристиками.

Посібник укладено відповідно до програми дисципліни «Мікропроцесорна техніка» для студентів, що навчаються за напрямом підготовки 050202 «Автоматизація та комп'ютерно-інтегровані технології», а також для студентів інших спеціальностей.

УДК 004.31 (075.8)
ББК 32.973.26-04я73

ISBN 978-966-350-359-2

ЗМІСТ

ПЕРЕДМОВА

Дисципліна «Мікропроцесорна техніка» належить до групи професійно орієнтованих і займає важливе місце у підготовці бакалаврів за напрямом 050202 «Автоматизація та комп'ютерно-інтегровані технології». Завдання предмету – формування знань з організації архітектури сучасних мікропроцесорів, мікроконтролерів, пам'яті та системи команд, паралельного й послідовного інтерфейсів, режимів тимчасових затримок і переривань, а також набуття навичок розробки алгоритмів і програм керування неперервними і дискретними об'єктами.

Зміст дисципліни вимагає від студентів знань з вищої математики, алгоритмічних мов й програмування, основ електротехніки і електроніки; забезпечує викладання основ збирання, обробки і передачі інформації, мікропроцесорних та програмних засобів систем керування, курсове та дипломне проектування.

Матеріал посібника відповідає лекційному курсу дисципліни, який впродовж кількох років викладався авторами у Національному гірничому університеті. Книга може бути використана для студентів заочно-дистанційної і екстернатної форми навчання. З цією метою під час викладення теоретичного матеріалу наводяться приклади розв'язання задач, наприкінці кожного розділу містяться контрольні питання, надається програма та методичні вказівки для лабораторного практикуму.

Посібник розроблено колективом викладачів Національного гірничого університету у співпраці з професором Ройтлінського університету (Німеччина) Герхардом Грулером і професором Еслінгенського університету (Німеччина) Ніколаусом Нойбергером.

Перший розділ написано доцентом М.В. Козарем. Другий, третій, четвертий і п'ятий – професором В.В. Ткачовим. Шостий, сьомий, восьмий, одинадцятий – доцентом С.М. Проценко. Дев'ятий – професором Г. Грулером. Десятий – професором Н. Нойбергером.

Враховуючи навчальне призначення, у посібнику не робиться бібліографічних посилань на запозичення з відомих джерел. Список літератури містить тільки вказівки на матеріали, що рекомендуються як додаткові для поглиблення знань та розширення навичок в галузі мікропроцесорної техніки.

Посібник створено для бакалаврів, що навчаються за напрямом 050202 «Автоматизація та комп'ютерно-інтегровані технології», а також для спеціалістів і магістрів, що продовжують навчання на основі означеного напрямка, зокрема 05020201 «Автоматизоване управління технологічними процесами».

Написання навчального посібника – досить клопітка робота, при виконанні якої неминучі помилки. Автори з вдячністю отримають зауваження та побажання, що сприятимуть подальшому покращенню змісту книги.

1. ЗАГАЛЬНІ ПОНЯТТЯ Й СТРУКТУРА СИСТЕМ КЕРУВАННЯ

Курс лекцій «Мікропроцесорна техніка» базується на дисциплінах «Теоретичні основи електротехніки», «Вимірювання електричних і не електричних величин», «Електроніка й мікросхемотехніка», «Програмування», «Елементи і пристрої систем керування».

З грецької мови слово «система» перекладається «складний», «багатокомпонентний», «взаємозалежний».

Під системою керування (СК) розуміється комплекс логічно взаємодіючих механічних, апаратних і програмних засобів, а також рішень, прийнятих людиною, що виконує завдання контролю й керування технологічним процесом або об'єктом.

Технологічний процес – це виконання послідовності дій і операцій перетворення вихідної сировини, матеріалу або енергії з метою одержання товарної продукції із заданими параметрами.

Керування – це сукупність впливів на керований об'єкт, спрямованих на підтримку його функціонування відповідно до обраних правил на підставі інформації, що надходить від об'єкта керування й зовнішнього середовища, у якому він функціонує. У випадку, коли керуючі впливи виконуються без участі людини, керування називається автоматичним. Якщо ж деякі функції під час вибору керуючих впливів здійснюються людиною, то керування зветься автоматизованим. За аналогією системи теж поділяються на автоматичного й автоматизованого керування. Узагальнена структурна схема такої СК зображена на рис. 1.1.

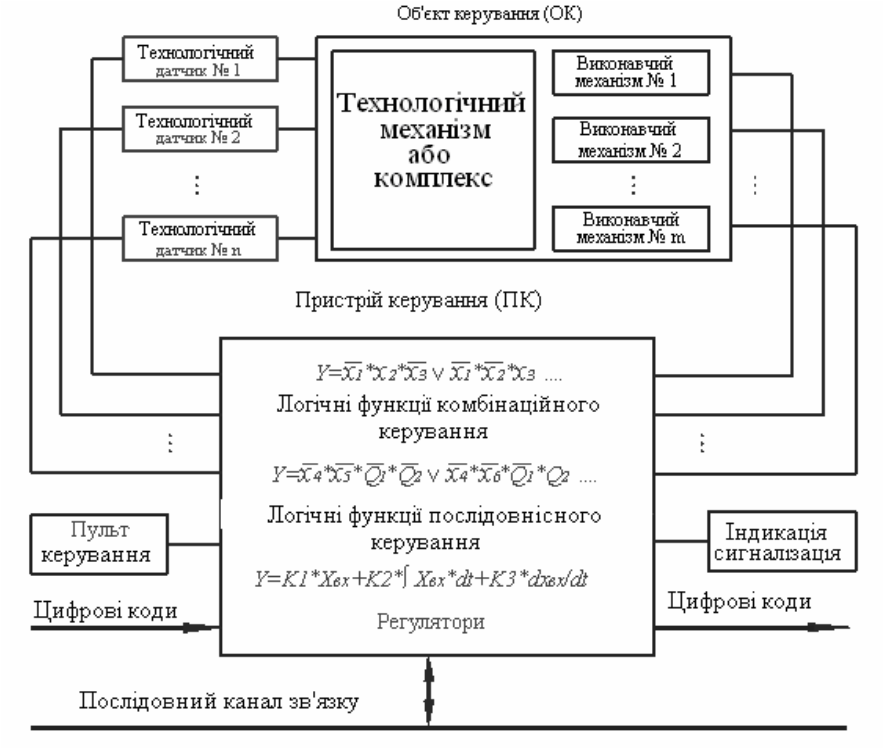


Рис. 1.1. Структурна схема системи керування

Наведена схема не є абсолютною з погляду «тільки так і ніяк інакше», але вона дає можливість найбільш повно уявити сукупність компонентів СК й розібратися з їхніми призначенням і взаємозв'язками.

Для СК джерелами інформації, як правило, є датчики, рішення, прийняті людиною, а також інші системи або підсистеми, з якими обумовлено спільне функціонування.

За допомогою датчиків виконується контроль параметрів технологічного процесу, зокрема переміщення або положення машин і механізмів, зусилля, температури, тиску, що розвиваються, й т.п. Тому датчики – це обладнання, де відбувається перетворення контрольованої фізичної величини в електричний сигнал. Вони поділяються на два основні типи – дискретні й аналогові, і узагальнено їх можна назвати технологічними.

У датчиках дискретного типу зміна вихідного сигналу відбувається східчасто – від мінімального значення до максимального, або навпаки. В аналогових датчиках вихідний сигнал змінюється безупинно, залежно від зміни контрольованого технологічного параметра, і в кожний момент часу може приймати величину, що перебуває між мінімальним і максимальним значеннями. До групи аналогових слід віднести також датчики, вихідними параметрами яких є зміна тривалості сигналу або частоти вихідних імпульсів. У кожному разі ці параметри пропорційні зміні контрольованої фізичної величини.

Інформація від людини передається в систему через органи пультів керування (кнопки, що задають елементи, маніпулятори), а також через комп'ютерні засоби шляхом вибору й впровадження у дію відповідних завдань, програм або режимів.

Інформація від підсистем, з якими обумовлено спільне функціонування, може надходити як у вигляді окремих сигналів (дискретних або аналогових), так і цифрових кодів, переданих по паралельних або послідовних каналах зв'язку.

Процес керування передбачає зняття інформації з об'єкта (вимірювання параметрів об'єкта або процесу), аналіз цієї інформації й вироблення керуючих впливів згідно з заданим законом керування.

Керований об'єкт, яким може бути технологічний механізм, лінія або комплекс, виділено в окремий прямокутник ОК. Під ним мається на увазі деяка сукупність виконавчих обладнань і механізмів, що приводять у дію технологічне встаткування. Це можуть бути електричні, пневматичні або гідравлічні приводи, нагрівальні елементи, маслостанції, компресори тощо. У цю частину входять також магнітні пускачі, контактори, силові перетворювачі, гідравлічні й пневматичні розподільники, через які енергія надходить до виконавчих приводів і на які безпосередньо спрямовані керуючі впливи від пристроїв керування (ПК).

На входи ПК можуть надходити аналогові й дискретні сигнали від технологічних датчиків і пультів керування, а також інформація у вигляді цифрових кодів у тих випадках, коли за умов перешкодозахищеності кодування знижує ймовірність її спотворення. Джерелом вхідної інформації є також і послідовний канал зв'язку, за допомогою якого може відбуватися

інформаційний обмін і взаємодія між суміжними підсистемами й верхніми рівнями ієрархії всієї структури підприємства.

По суті, пристрій керування є основним елементом СК, що сприймає вхідну інформацію, перетворює її за заданими законами й алгоритмами і виробляє керуючі впливи на ОК. Крім цього, ПК виконує функції надання інформації обслуговуючому персоналу у вигляді світлової й звукової сигналізації або графічної візуалізації, а також спільно взаємодіючим підсистемам і вищим за ієрархією рівням керування.

1.1. Дискретні або цифрові СК

Якщо припустити, що об'єкт керування відомий, тобто, визначені виконавчі механізми та засоби комутації енергії до них, обрані відповідні датчики, визначені зовнішні зв'язки, сформульовані необхідні вимоги до алгоритмів керування й параметрів регулювання, то залишається створити або розробити самий ПК. За своєю суттю ПК є пристроєм перетворення вхідної інформації у вихідну. Але оскільки вхідні і вихідні сигнали неоднакові, при розробці необхідно застосовувати різні елементи і пристрої як аналогові, так і дискретні.

Дискретні способи перетворення сигналів мають декілька безперечних переваг. Насамперед, це велика точність перетворення, висока перешкодозахищеність і загальна надійність як способів перетворення, так і пристроїв, що їх реалізують. Дискретне перетворення дає реальну можливість використовувати складні алгоритми обробки сигналів з точним цифровим відображенням вхідної й вихідної інформації. Самі пристрої мають відносно низьку вартість. Зазначені переваги такі, що у деяких випадках вигідніше змінити характер вхідних або вихідних сигналів для збереження однорідності функцій перетворення.

У системах керування, де вхідними й вихідними є дискретні сигнали, побудова самих пристроїв керування найчастіше зводиться до розробки рішень, заснованих на законах логічного керування, описуваних відповідними логічними функціями. До них відносяться пристрої, що реалізують комбінаційне або послідовнісне керування. У першому випадку значення функцій виходів залежать тільки від комбінацій вхідних дискретних сигналів, що розглядаються як аргументи або логічні змінні. У другому випадку вихідні функції формуються також, але з урахуванням стану елементів пам'яті. У класиці дискретної логіки це розглядається як комбінаційні автомати без пам'яті й послідовнісні автомати з пам'яттю.

Вирішуючи питання дискретизації при реалізації функцій неперервного регулювання, у технічних системах найчастіше використовують аналого-цифрові (АЦП) і цифро-аналогові (ЦАП) перетворювачі. Таким чином вхідні аналогові сигнали перетворюються в послідовності кодів (чисел), які піддаються математичній обробці, а вихідні аналогові сигнали формуються з послідовності кодів (чисел), одержуваних в результаті цієї обробки. За зазначеним принципом будуються контури неперервного регулювання, де

потрібно або підтримувати, або змінювати регульовану величину вихідного сигналу відповідно до обраного закону керування. Такі регулятори одержали назву цифрові, а системи – цифрового керування.

1.2. Основи обчислювальної техніки

1.2.1. Основи булевої алгебри

Для опису роботи дискретних і цифрових пристроїв використовується математичний апарат алгебри логіки або булевої алгебри. Основними поняттями булевої алгебри є логічна змінна й логічна функція.

Логічна змінна – це величина, яка може приймати тільки два можливі значення, одне з яких за твердженням вважається «справжнім» а друге – «помилковим». У булевій алгебрі справжнє значення змінної позначається символом «1» (логічна одиниця), неправильне – символом «0» (логічний нуль). Самі змінні частіше позначають символами $x_1, x_2, ..., x_n$. У силу визначення логічні змінні можна називати також двійковими.

Логічна – це функція двійкових змінних (аргументів), яка також може приймати тільки одне з двох значень, яке так само, за твердженням, може бути справжнім (що дорівнює «1») або помилковим (що дорівнює «0»). Значення деякої логічної функції n змінних задається для кожної комбінації двійкових змінних, кількість можливих різних наборів яких дорівнює 2^n. При цьому, оскільки сама функція на кожному наборі може приймати значення «0» або «1», загальне число можливих функцій від n змінних дорівнює $2 \cdot 2^n$. Позначається логічна функція звичайно символом y.

Для безлічі значень, які можуть приймати як аргументи, так і функції в булевій алгебрі визначаються відношення еквівалентності, позначуване символом рівності «=», і три операції: логічного додавання (диз'юнкції), логічного множення (кон’юнкції), логічного заперечення (інверсії), позначувані відповідно символами:

+ або $\vee$ – операція диз'юнкції,

$\cdot$ або $\wedge$ або & – операція кон’юнкції,

$\bar{*}$ – операція інверсії (* – символ аргументу або функції).

З огляду на постулати при виконанні перерахованих операцій відносини еквівалентності мають вигляд:

а)	б)	в)
$0 + 0 = 0$	$0 \cdot 0 = 0$	$\bar{0} = 1$
$0 + 1 = 1$	$0 \cdot 1 = 0$	$\bar{1} = 0$
$1 + 0 = 1$	$0 \cdot 1 = 0$	
$1 + 1 = 1$	$1 \cdot 1 = 1$	

(1.1)

На підставі постулатів (1) можна вивести наступні співвідношення (закони) алгебри логіки:

1. Закони одинарних елементів: універсальної множини – а); нульової множини – б); тавтології – в).

а)	б)	в)
$x + 1 = 1$	$x + 0 = x$	$x+x= x$
$x \cdot 1 = x$	$x \cdot 0 = 0$	$x \cdot x = x$

(1.2)

2. Закони заперечення: подвійного заперечення – а); додатковості – б), подвійності – в).

а)	б)	в)
$\bar{\bar{x}} = x$	$x + \bar{x} = 1$	$\overline{x_1 + x_2} = \bar{x}_1 \cdot \bar{x}_2$
	$x \cdot \bar{x} = 0$	$\overline{x_1 \cdot x_2} = \bar{x}_1 + \bar{x}_2$

(1.3)

3. Закони абсорбції або поглинання – а), склеювання – б).

а)	б)
$x_1 + x_1 \cdot x_2 = x_1$	$x_1 \cdot x_2 + x_1 \cdot \bar{x}_2 = x_1$
$x_1 \cdot (x_1 + x_2) = x_1$	$(x_1 + x_2) \cdot (x_1 + \bar{x}_2) = x_1$

(1.4)

Закони подвійності (3, в), які також мають назву де-Моргана, були узагальнені К. Шенноном на випадок довільного (n) числа аргументів.

Крім законів, зазначених вище, які не мають аналогів у звичайній алгебрі (алгебрі чисел), для алгебри логіки справедливі закони звичайної алгебри: комутативні або переставні, дистрибутивні або розподільні, асоціативні або сполучні.

Будь-яка логічна функція y для n двійкових змінних $x_1, x_2, ..., x_n$ може бути задана за допомогою таблиць. Вони одержали назву таблиць істинності, що містять 2^n рядків, у які записуються всі можливі двійкові набори значень аргументів, а також відповідне кожному із цих наборів значення функції.

У дискретній і цифровій схемотехніці широко використовуються функції заперечення, кон’юнкції, заперечення кон’юнкції, диз'юнкції, заперечення диз'юнкції й додавання за модулем два. Для реалізації цих функцій розроблені й випускаються інтегральні мікросхеми. Їх часто називають базовими логічними елементами «НІ», «І», «І – НІ», «АБО», «АБО – НІ», а пристрої, синтезовані на їхній основі, називають комбінаційними автоматами або пристроями комбінаційного типу.

1.2.2. Базові логічні елементи

Найпростішою є функція заперечення, значення якої завжди протилежно значенню аргументу. Аналітична форма запису цієї функції має вигляд:

$$y = \overline{x}. \tag{1.5}$$

Функція заперечення, а відповідно й логічна операція «НІ» є унарною, тобто має всього один аргумент. Таблиця істинності цієї функції, умовна позначка елемента «НІ» і його тимчасові діаграми наведені на рис. 1.2.

Інверсія мовою дискретної або цифрової техніки означає, що значення вихідного сигналу (y) завжди протилежно значенню вхідного (x).

Логічні функції кон'юнкції і диз'юнкції є бінарними, тому що являють собою результати дій більше ніж над однією змінною.

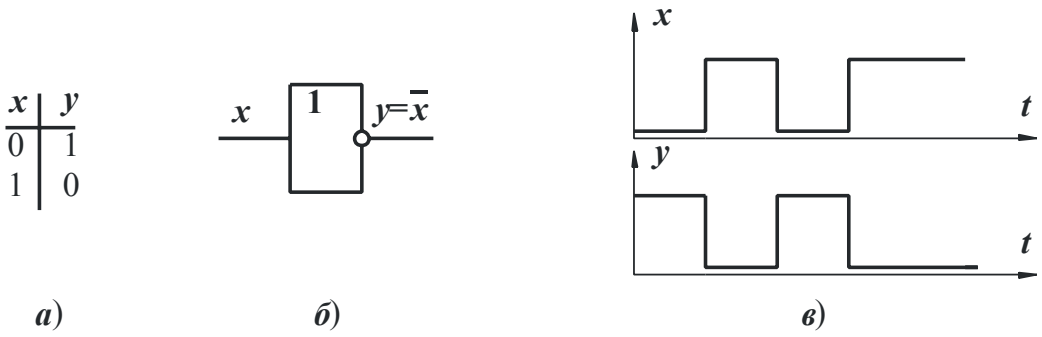


Рис. 1.2. Елемент «НІ»: а) таблиця істинності; б) умовне зображення; в) тимчасові діаграми

Аналітична форма запису функції кон’юнкції двох аргументів x_1 і x_2 має вигляд:

$$y = x_1 \cdot x_2 \text{ або } y = x_1 \wedge x_2 \text{ або } y = x_1 \,\&\, x_2. \tag{1.6}$$

Функція кон’юнкції дійсна або дорівнює 1 тоді й тільки тоді, коли всі її аргументи набувають істинного значення або дорівнюють 1.

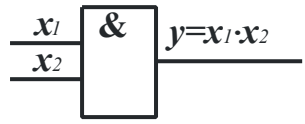
Логічний елемент, що реалізує функцію кон’юнкції, називають кон’юнктором або елементом «І». На рис. 1.3 наведені: (а) – таблиця істинності; (б) – умовне зображення; (в) – тимчасові діаграми двовходового кон’юнктора.

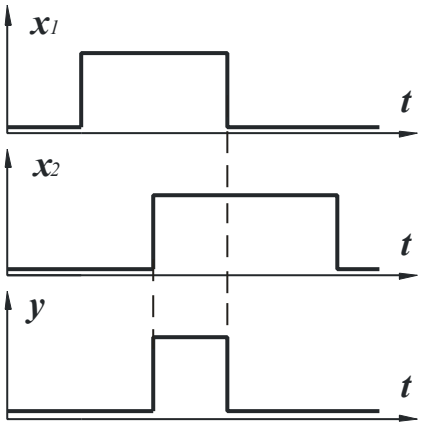
Елементи «І» часто використовують для керування потоком інформації. При цьому на один з його входів надходять сигнали, що несуть деяку інформацію, а на інший – керуючий сигнал: пропустити інформацію – 1, не пропустити – 0.

Функція заперечення кон’юнкції («І – НІ») одержується шляхом інвертування самої функції кон’юнкції. Запис такої функції має вигляд:

$$y = \overline{x_1 \cdot x_2}. \tag{1.7}$$

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



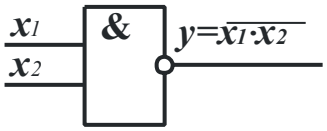


а) *б*) *в*)

Рис. 1.3. Елемент «І»: а) таблиця істинності, б) умовне позначення, в) тимчасові діаграми

Умовне зображення елемента «І – НІ» й таблиця істинності функції (1.7) наведені на рис. 1.4.

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0



а) *б*)

Рис. 1.4. Елемент «І – НІ»:
а) таблиця істинності, б) умовне зображення

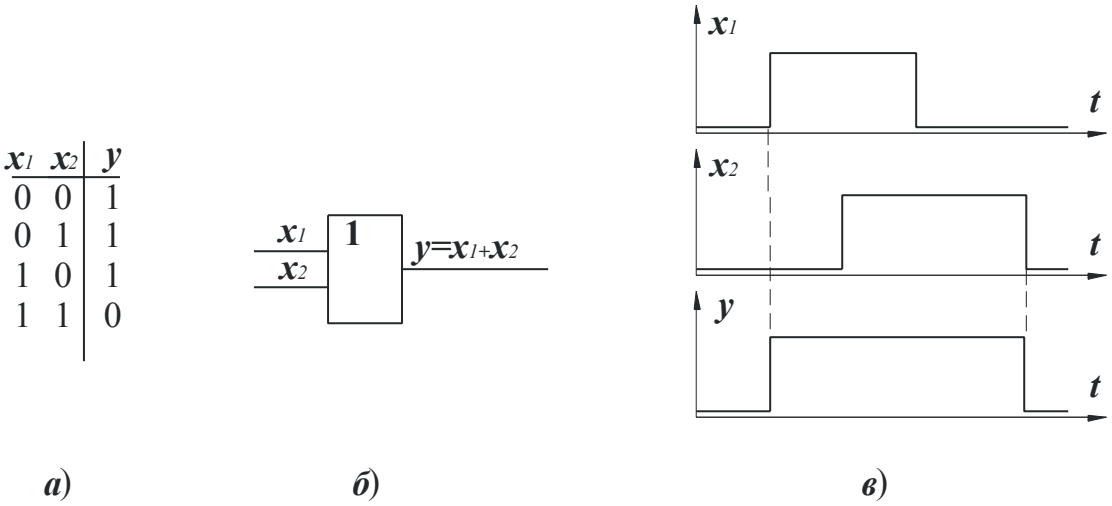
Аналітична форма запису функції диз'юнкції двох аргументів x_1 і x_2 має вигляд:

$$y = x_1 + x_2 \qquad \text{або} \qquad y = x_1 \vee x_2 . \tag{1.8}$$

Функція диз'юнкції істинна або дорівнює 1, якщо хоча б один з її аргументів набуває істинного значення або дорівнює 1. Така функція часто називається функцією логічного додавання. Логічний елемент, що реалізує функцію диз'юнкції, називають диз’юнктором або елементом «АБО». На рис. 1.5 для нього наведені: (а) – таблиця істинності; (б) – умовне зображення; (в) – тимчасові діаграми.

Функція заперечення диз'юнкції («АБО – НІ») одержується шляхом інвертування самої функції диз'юнкції і її запис має вигляд:

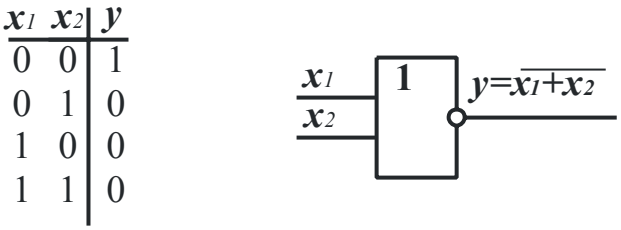
$$y = \overline{x_1 + x_2} . \tag{1.9}$$



а) *б*) *в*)

Рис. 1.5. Елемент «АБО»: а) таблиця істинності, б) умовне позначення, в) тимчасові діаграми

Умовне зображення елемента «АБО – НІ» й таблиця істинності функції (1.9) наведені на рис. 1.6.



а) *б*)

Рис. 1.6. Елемент «АБО – НІ»:
а) таблиця істинності; б) умовне зображення

Функція «додавання за модулем два» (М2) найчастіше використовується як функція двох аргументів. У цьому випадку її називають також функцією «нерівнозначності». Аналітична форма запису такої функції має вигляд:

$$y = x_1 \oplus x_2 \qquad (1.10)$$

Назва функції пов'язана з тим, що $x_1 \oplus x_2$ є арифметичною сумою двійкових чисел x_1 і x_2 в межах одного розряду: 0+0=0; 0+1=1; 1+0=1; 1+1=10. В останньому випадку виникаючий перенос у старший розряд ігнорується, а в розряді самих доданків одержуємо нуль. Логічні елементи такого типу широко застосовуються при синтезі підсумовуючих пристроїв.

Функція М2 має деяку властивість, яку корисно знати.

$$\begin{matrix} \text{а)} & \text{б)} & \text{в)} \\ \bar{x}_1 \oplus x_2 = x_1 \oplus \bar{x}_2 = \overline{x_1 \oplus x_2} \end{matrix} \qquad (1.11)$$

При інвертуванні одного з аргументів уся функція інвертується. У цьому випадку реалізується логічна функція «рівнозначності» $x_1 \equiv x_2$. Вона дорівнює одиниці, якщо $x_1 = x_2$. Очевидні також і співвідношення (12).

$$x \oplus 0 = x; \qquad x \oplus 1 = \bar{x} \qquad x \oplus x = 0 \qquad x \oplus \bar{x} = 1 \qquad (1.12)$$

1.2.3. Реалізація логічних елементів (ЛЕ)

Релейно-контактні логічні елементи

Логічну функцію, яка визначає залежність стану вихідних сигналів обладнання керування від вхідних, можна реалізувати на релейно-контактних логічних (РКЛ) елементах, що являють собою електромагнітні реле. Вхід такого елемента – обмотка, а вихід – стан контактів. Звичайно на схемах зображуються не обмотки, а тільки контакти реле. Логічна змінна x може бути зумовлена станом контакту, а також ухвалювати одне із двох значень. Значенню «істина» відповідає замкнений стан контакту, значенню «неправда» – розімкнений (рис. 1.7). Часто «істинний» і «хибний» стани відповідають термінам «високий» і «низький» рівень, «так» – «ні», «увімкнено» – «вимкнено», «одиниця (1)» – «нуль (0)».

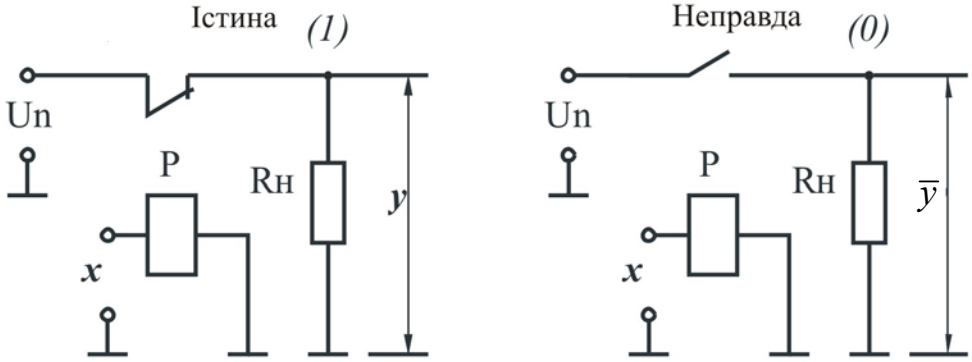


Рис. 1.7. Релейно-контактні логічні елементи

Реалізація функцій диз'юнкції, кон’юнкції, інверсії на релейно-контактних елементах показана на рис. 1.8. Функцію інверсії НІ можна реалізувати, використовуючи розмикальний контакт реле x (рис. 1.8, а). Функція кон’юнкції припускає послідовне увімкнення контактів і вона буде дорівнювати 1, якщо всі змінні одночасно на вході дорівнюють 1 (рис. 1.8, б). Оскільки функція диз'юнкції дорівнює 1, якщо хоча б одна змінна на вході дорівнює 1, то це рівносильно паралельному увімкненню контактів реле (рис. 1.8, в).

Приклад реалізації складної логічної функції на релейно-контактних елементах наведено на рис. 1.8, г.

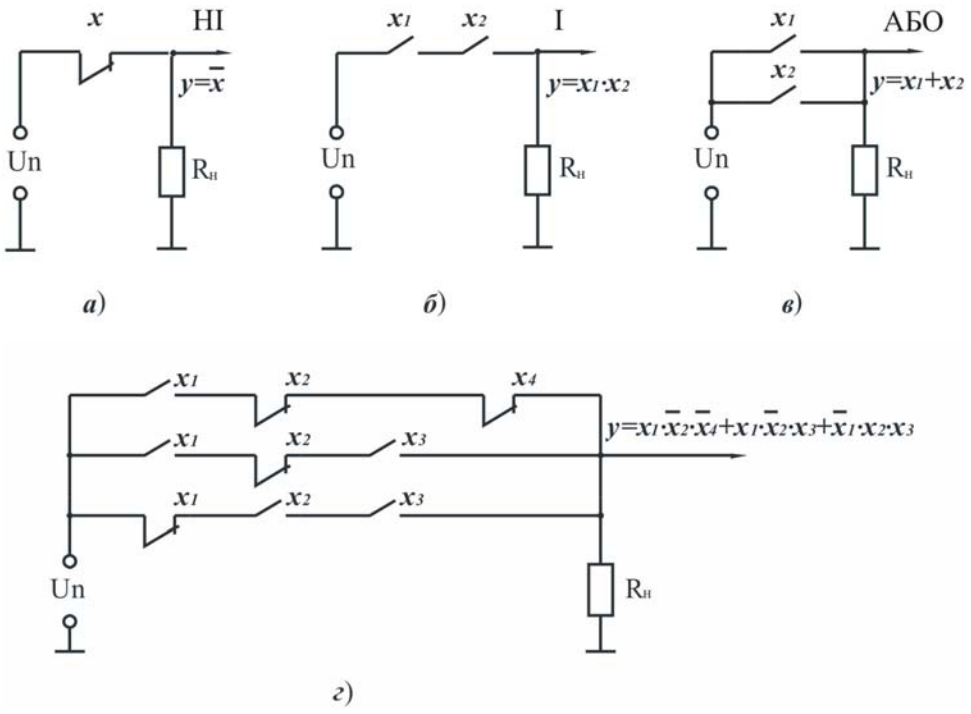


Рис. 1.8. Реалізація логічних функцій на РКЛ елементах

Діодні логічні елементи

Реалізувати основні логічні операції «І», «АБО», «НІ» можна на напівпровідникових діодах. У схемі «АБО» (рис. 1.9, а) не потрібно додаткового джерела енергії, тому що під час подачі позитивного потенціалу на один з виходів x через відповідний діод і резистор $R_н$ проходить струм, під дією якого на $R_н$ спадає напруга, що відповідає на виході логічній одиниці. Функція «І» (рис. 1.9, б) реалізується з використанням додаткового джерела зсуву $U_{зс}$. За відсутності вхідних сигналів $x_1, x_2, ..., x_n$ через резистор $R_б$, діоди $VD1, ..., VDn$ і вхідні опори джерел сигналу x проходить струм зсуву, і вся напруга виділяється на резисторі $R_б$, а на виході – низький потенціал (0). Під час подачі на вхід сигналів x діоди $VD1, ..., VDn$ замикаються, тому на виході буде високий потенціал (1). Якщо навіть на одному із входів відсутній вхідний сигнал, то на виході буде 0, що й відповідає функції «І».

Функція «НІ» реалізується з використанням двох джерел живлення (рис. 1.9, в). Якщо відсутній вхідний сигнал x, то діод $VD1$ відкритий і на виході буде високий потенціал (1) від Vn. Якщо на вхід поданий негативний потенціал, то діод $VD1$ закривається, струм через $R_н$ не проходить, і на виході потенціал буде дорівнювати 0. Мала навантажувальна здатність і велике згасання сигналів – основні недоліки діодної ЛЕ.

Транзисторні логічні елементи

Транзисторні ЛЕ здобули найбільшого поширення, оскільки мають кілька схемних рішень реалізації логічних функцій.

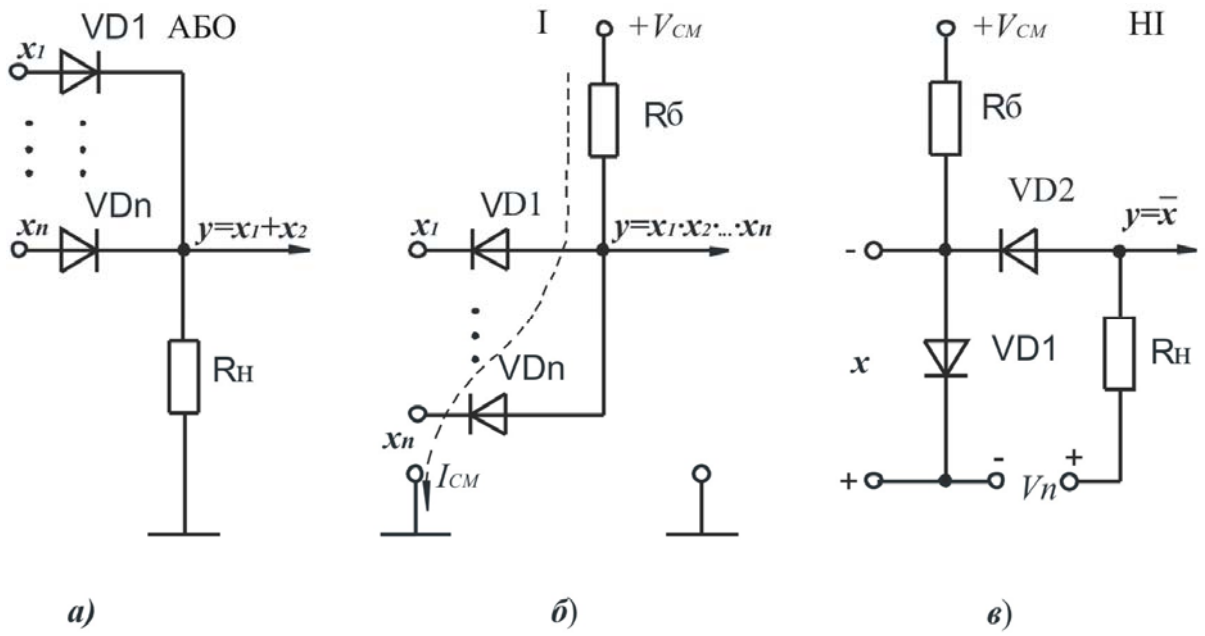


Рис. 1.9. Діодні логічні елементи

Реалізація основних логічних операцій з використанням транзисторів наведена на рис. 1.10. Операція «АБО» (рис. 1.10, а) реалізується, якщо відкрито один із транзисторів $VT1$ або $VT2$, тобто $y = x_1 \vee x_2$; операція «І» (рис. 1.10, б) реалізується, якщо відкриті транзистори $VT1$ й $VT2$, тобто $y = x_1 \cdot x_2$. Операція інверсії реалізується за допомогою звичайного ключа з навантаженням у ланцюзі колектора (рис. 1.10, в). За відсутності сигналу на вході $(x = 0)$ транзистор VT закритий, і на виході буде високий потенціал (1). За наявності сигналу на вході $(x = 1)$ транзистор відкривається, потенціал на виході дорівнює потенціалу «загальної крапки» (тобто «0»).

Використовуючи діод-транзисторні ЛЕ, можна реалізувати логічну функцію у функціонально повному базисі «І – НІ» (рис. 1.10, г), а застосовуючи резисторно-транзисторні ЛЕ, – у функціонально повному базисі «АБО – НІ» (рис. 1.10, д).

Транзисторно-транзисторні логічні елементи (ТТЛ) з'явилися внаслідок їх подальшого розвитку завдяки заміні діодів багатоеміттерним транзистором. Основна відмінність багатоеміттерного транзистора від звичайного полягає в тому, що він має кілька еміттерів, розташованих таким чином, що пряма взаємодія між ними виключена.

ТТЛ дозволяють реалізувати логічні функції у функціонально повному базисі «І – НІ» (рис. 1.10, е). Якщо на входи x_1 й x_2 подано високий потенціал $(x = 0)$, то проходить струм I_1, який відкриває транзистори $VT2$ й$VT4$, і на виході буде низький потенціал $(y = 0)$. Якщо навіть на один із входів x_1 або x_2 подано низький потенціал (тобто заземлено один із входів), то в цьому випадку стане проходити струм I_2, транзистори $VT2$ й $VT4$ будуть закриті, і на виході утворюється високий потенціал $(y = 1)$.

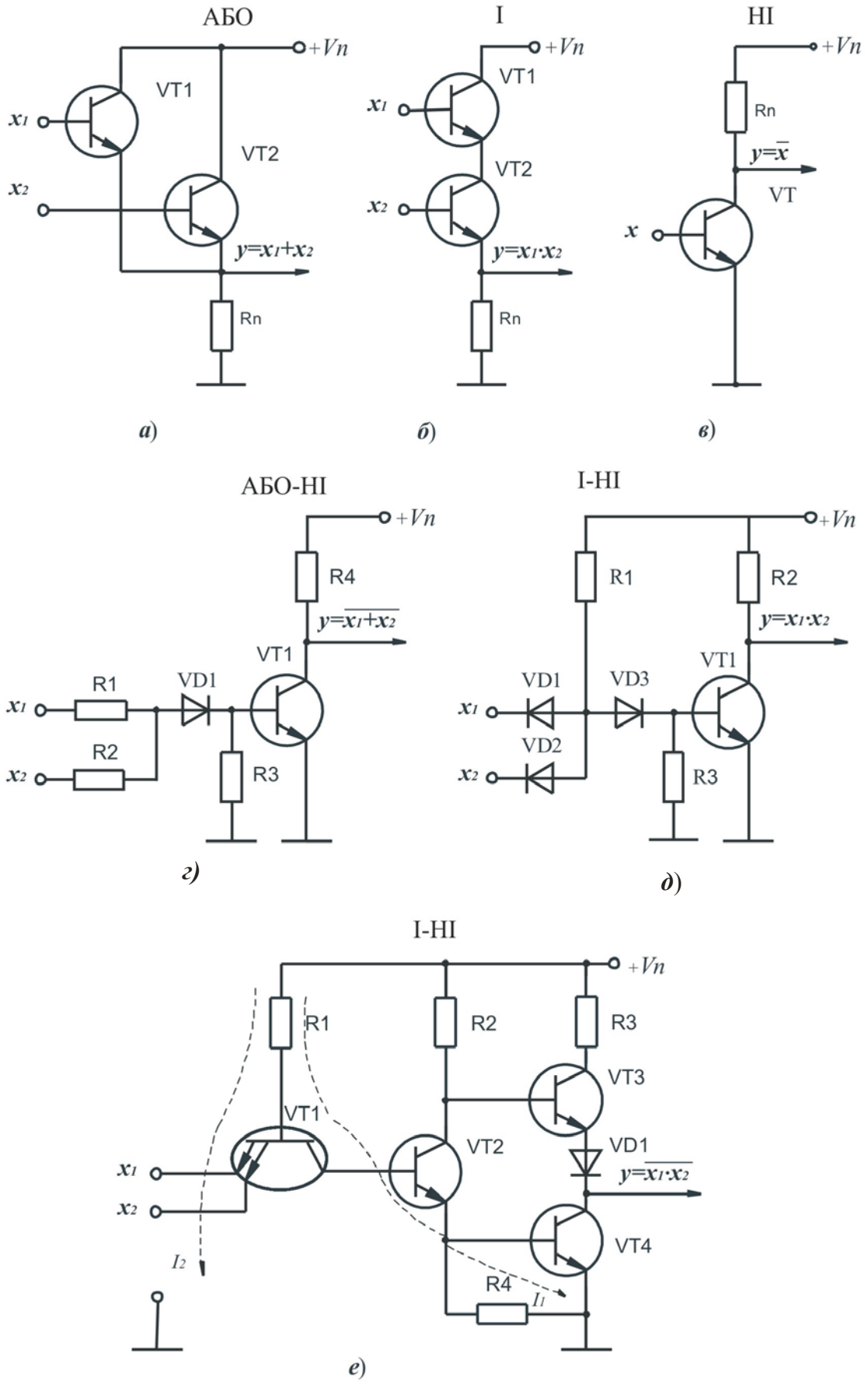


Рис. 1.10. Транзисторні логічні елементи

ТТЛ елементи застосовуються винятково в інтегральних схемах, що мають високу щільність упаковки електрично з'єднаних елементів і кристалів, коли вартість транзисторів не є великою, а простота проектованого пристрою – не головна вимога. Елемент, наведений на рис. 1.10, е, є базовим або типовим для ТТЛ і випускається серійно з різною кількістю входів – від двох до восьми.

1.2.4. Комбінаційні пристрої

Комбінаційні пристрої повністю описуються логічними функціями. У деяких випадках при реалізації пристроїв керування об'єктами їх виявляється цілком достатньо. Але це тільки ті випадки, коли функції керування залежать тільки від станів або комбінацій вхідних сигналів.

В обчислювальній і мікропроцесорній техніці до пристроїв комбінаційного типу належать дешифратори, мультиплексори, суматори, тригери й деякі інші. Побудову таких пристроїв буде розглянуто нижче.

Дешифратори, мультиплексори, демультиплексори

Дешифраторами називають цифрові пристрої комбінаційного типу, що мають кілька входів і виходів, у яких певним комбінаціям вхідних сигналів відповідають цілком певні комбінації вихідних сигналів. Іноді такі пристрої називають декодерами. Різноманітність дешифраторів дуже широка, але в мікропроцесорній техніці звичайно використовуються так звані лінійні дешифратори.

Лінійним дешифратором називається пристрій, що здійснює перетворення n-розрядного двійкового коду в m-розрядний унітарний код. Унітарний код може бути прямим, якщо одна «1» у деякому розряді m-розрядного двійкового коду й $m-1$ нулів, або зворотним, якщо один «0» і $m-1$ одиниць. Наприклад, запис унітарного коду для $m=4$ буде мати вигляд:

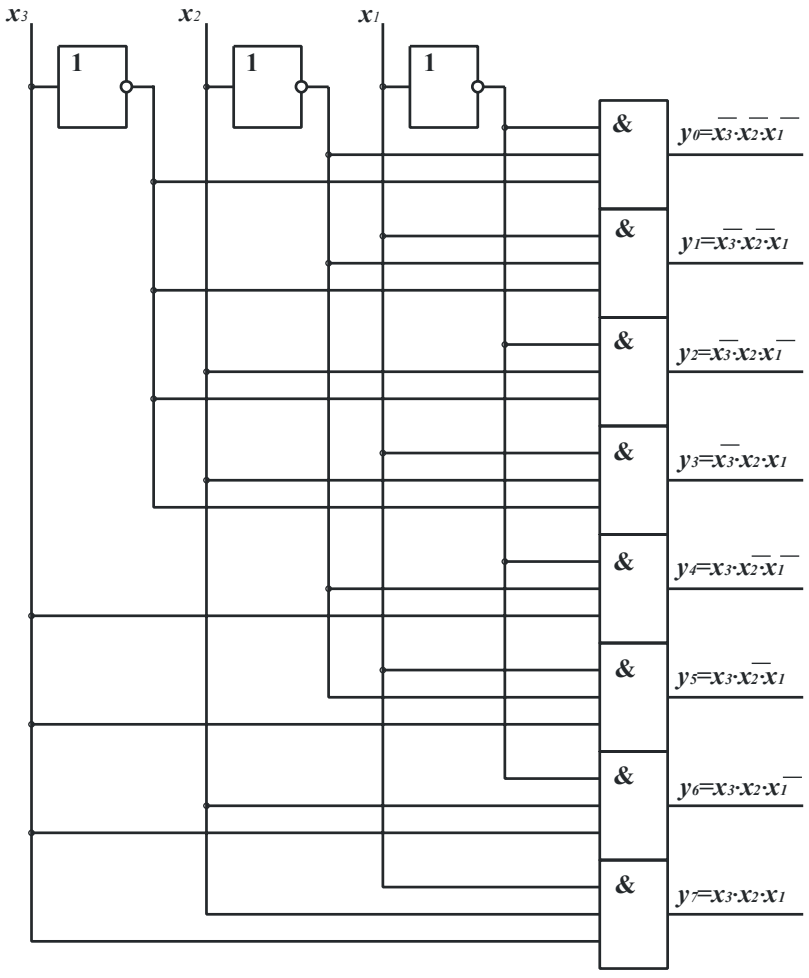
прямого – 1000, ... , 0010, 0001
зворотного – 0111, ... , 1101, 1110

Схема дешифратора має n входів, на які надходять відповідні розряди двійкового коду $x_n, x_{n-1}, ..., x_2, x_1$ й m виходів, на яких формуються розряди унітарного коду $y_{m-1}, ..., y_1, y_0$. Так, наприклад, дешифратор, що має $n=3$ входів, буде мати $m=2^n=8$ виходів. Таблиця істинності цього дешифратора наведена на рис. 1.11, його функціональна схема й умовне графічне позначення показані на рис. 1.12.

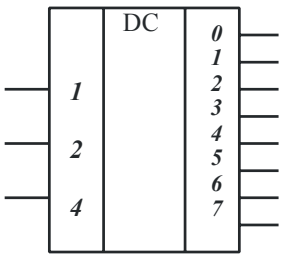
Функціональна схема дешифратора побудована відповідно до рівнянь $y_7 ... y_0$ і являє собою 8 кон'юнкторів з 3-я входами, кожний з яких реалізує одну з вихідних функцій. Якщо виходи дешифратора проінвертирувати, то вийде дешифратор зі зворотним унітарним кодом або, як його називають, дешифратор з інверсними виходами.

x_3	x_2	x_1	y_0	y_1	y_2	...	y_6	y_7
0	0	0	1	0	0	...	0	0
0	0	1	0	1	0	...	0	0
0	1	0	0	0	1	...	0	0
.....			.....					
1	1	0	0	0	0	...	1	0
1	1	1	0	0	0	...	0	1

Рис. 1.11. Таблиця істинності дешифратора для $n = 3$ входів



а)



б)

Рис. 1.12. Дешифратор: а) функціональна схема; б) умовне позначення

Варіант схемної реалізації мультиплексора «4-1» і його умовне графічне зображення наведені на рис. 1.13.

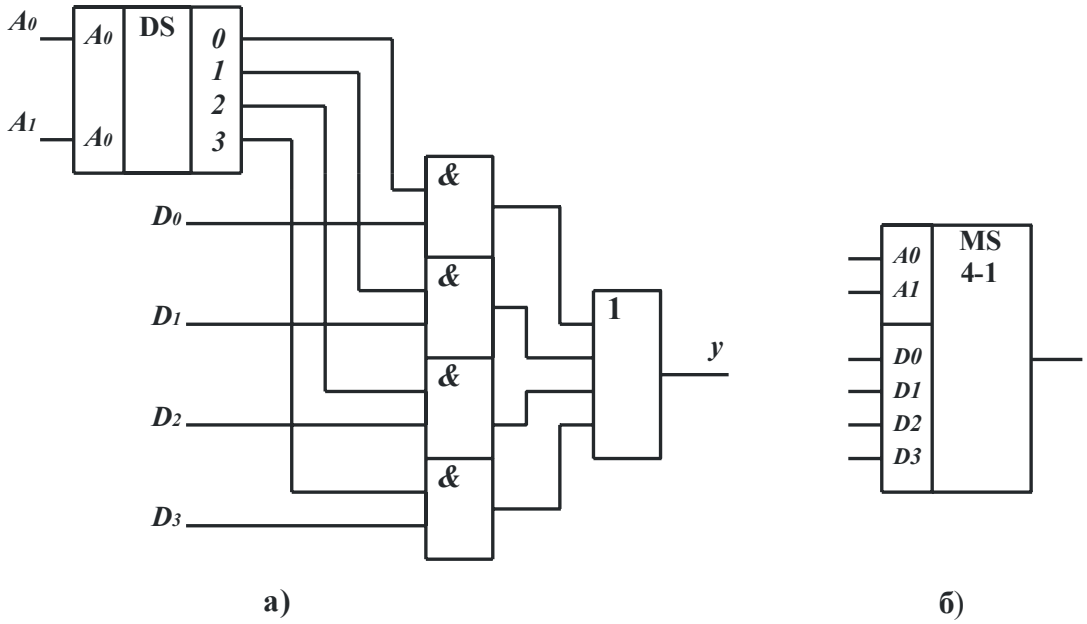


Рис. 1.13. Мультиплексора: а) функціональна схема;
б) умовне зображення

Мультиплексор побудований як сукупність дешифратора двійкового адресного коду й чотирьох двовходових елементів «І», виходи яких об'єднані елементом «АБО». Очевидно, що стан виходу буде залежати від адресної комбінації A_0, A_1 й стану відповідної їй входу $Д_i$.

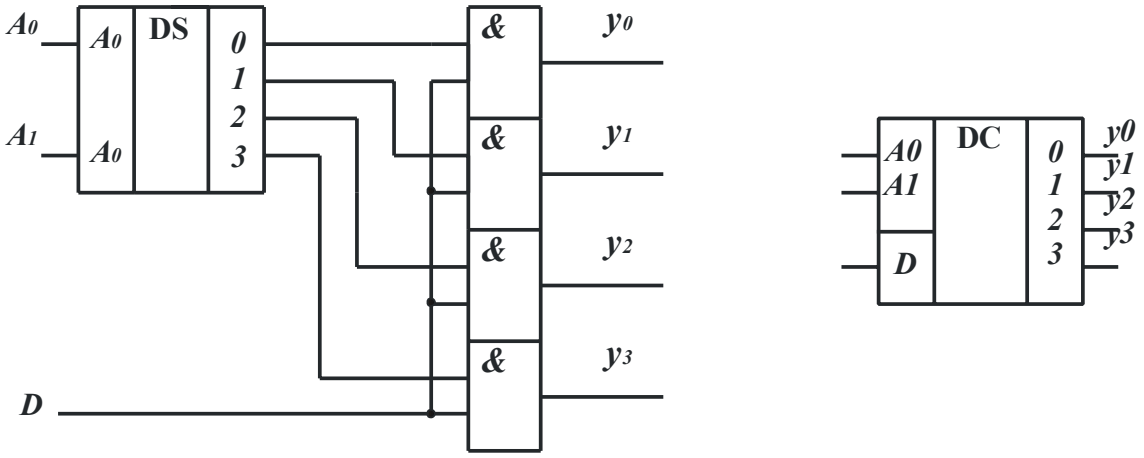
Демультиплексор – це схема, що виконує функцію, яка є зворотною для функції мультиплексора, тобто це комбінаційна схема, що має один інформаційний вхід $(Д)$, n інформаційних виходів $(y_0, y_1, ..., y_{n-1})$ і k керуючих (адресних) входів $(A_0, A_1, ..., A_{k-1})$. Двійковий код, що надходить на адресні входи, визначає один з n виходів, на який передається значення змінної з інформаційного входу $(Д)$.

Таблиця функціонування демультиплексора, що має $n = 4$ інформаційних виходів (y_0, y_1, y_2, y_3) і $k = 2$ адресних входів(A_0, A_1), зображена на рис. 1.14.

$Д$	A_0,	A_1	$в_0$	$в_1$	$в_2$	$в_3$
0	0	0	0	0	0	0
1	0	0	1	0	0	0
0	0	1	0	0	0	0
1	0	1	0	1	0	0
0	1	0	0	0	0	0
1	1	0	0	0	1	0
0	1	1	0	0	0	0
1	1	1	0	0	0	1

Рис. 1.14. Таблиця істинності демультиплексора для $n = 4$ виходів

Схема демультиплексора і його умовне зображення показані на рис. 1.15.



а) **б)**

Рис. 1.15. Демультиплексор: а) функціональна схема;
б) умовне зображення

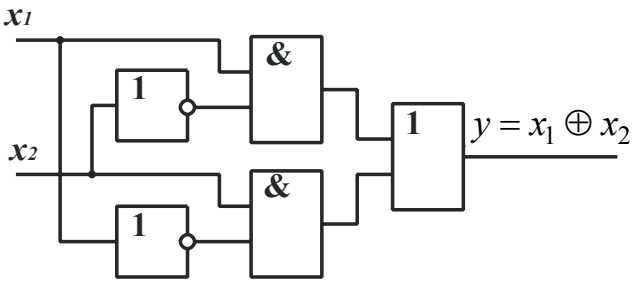
Схему реалізовано з використанням тривходових елементів «І» і двох інверторів. Передача сигналу із входу $(Д)$ на один з виходів буде відповідати адресній комбінації на входах A_0, A_1.

1.2.5. Арифметичні пристрої

Напівсуматор

Напівсуматор – це пристрій, що реалізує функцію «сума за модулем 2» (М2). Напівсуматори застосовуються при побудові арифметико-логічних пристроїв (АЛП) як елементи, що реалізують функцію нерівнозначності, і як складові частини повних арифметичних суматорів. Таблиця істинності функції, її функціональна схема й умовне зображення показані на рис. 1.16. Очевидно, що на виході напівсуматора буде 1 тільки при комбінаціях вхідних сигналів 01 і 10.

x_2	x_1	y
0	0	0
0	1	1
1	0	1
1	1	0



а) б) в)

Рис. 1.16. Функція М2: а) таблиця істинності;
б) функціональна схема; в) умовне зображення

Повні двійкові суматори

Правила виконання арифметичних операцій

Арифметичні належать до поширених операцій, що виконуються цифровими пристроями.

Правила виконання арифметичних операцій над двійковими числами є аналогічними до відповідних правил десяткової арифметики й зведені в табл. 1.1 і 1.2.

Таблиця 1.1

Правила й приклад виконання арифметичного додавання

Двійкове додавання			
Доданки k-го розряду	Сума k-го розряду	Перенос в $k+1$-й розряд	Приклад
0 + 0 =	0	0	1100 – перенос
0 + 1 =	1	0	1101 – 1-й доданок
1 + 0 =	1	0	+ 1100 – 2-й доданок
1 + 1 =	0	1	11001 – сума

Таблиця 1.2

Правила й приклад виконання арифметичного вирахування

Двійкове віднімання						
Зменшуване k-го розряду		Від'ємне k-го розряду		Різниця k-го розряду	Позика з $k+1$-го розряду	Приклад
0	–	0	=	0	0	010 – позика
0	–	1	=	1	1	1101 – зменшуване
1	–	0	=	1	0	– 1010 – від'ємний
1	–	1	=	0	0	0011 – різниця

Для виконання арифметичних операцій над двійковими числами зі знаком уводять додатковий (знаковий) розряд, який указує, чи є число позитивним або негативним. Якщо число позитивне, у знаковий розряд проставляється символ 0, якщо негативне, то символ 1. Наприклад, число (+5) з урахуванням знакового розряду (відділяється крапкою) запишеться як 0.101, а число (–3) як 1.011.

При додаванні чисел з однаковими знаками вони складаються, і сумі привласнюється код знака доданків, наприклад:

$$\begin{array}{r} +3 \\ +2 \\ \hline 5_{10} \end{array} \Rightarrow \begin{array}{r} 0.011 \\ +0.010 \\ \hline 0.101_2 \end{array} \qquad \begin{array}{r} -3 \\ -2 \\ \hline -5_{10} \end{array} \Rightarrow \begin{array}{r} 0.011 \\ +0.010 \\ \hline 0.101_2 \end{array}$$

Ускладнюється операція додавання чисел з різними знаками (алгебраїчне додавання), що рівносильно відніманню чисел. У цьому випадку необхідно визначити більше за модулем число, зробити віднімання й привласнити різниці знак більшого (за модулем) числа.

Для спрощення виконання цієї операції доданки зображуються у зворотному або додатковому кодах, оскільки відомо, що операція віднімання (алгебраїчного додавання) зводиться до операції простого арифметичного додавання двійкових чисел, зображених у зворотному або додатковому кодах. Позитивні числа в прямому, зворотному й додатковому кодах мають той самий вид, а негативні – різний.

Щоб зобразити негативне двійкове число у зворотному коді треба поставити в знаковий розряд 1, а у всіх інших розрядах прямого коду замінити одиниці нулями, а нулі – одиницями, тобто проінвертирувати число.

При записі негативного двійкового числа в додатковому коді, треба поставити 1 у знаковий розряд, а інші розряди одержати зі зворотного коду числа за допомогою додавання 1 до молодшого розряду.

Наведемо приклади запису двійкових чисел зі знаками в прямому, зворотному й додатковому кодах.

Число	Прямий код	Зворотний код	Додатковий код
+6	0.110	0.110	0.110
-5	1.101	1.010	1.011
-11	1.1011	1.0100	1.0101

Пояснимо процедуру віднімання чисел 5 і 3, 3 і 5. Послідовність і взаємозв'язок операцій наведено у табл. 1.3.

Таблиця 1.3

Правила віднімання чисел

Дія	Зворотний код	Додатковий код
$5-3=5+(-3)=2$	$x_1=0,101$ $x_2=1,100$ 0,101 +1,100 10,001 0,010 Перенос у молодший розряд. Сума позитивна	$x_1=0,101$ $x_2=1,101$ 0,101 +1,101 10,010 x 0,010 Одиниця переносу в молодший розряд ігнорується. Сума позитивна
$3-5=3+(-5)=-2$	$x_1=0,011$ $x_2=1,010$ 0,011 +1,010 1,101 Перенос у молодший розряд відсутній. Сума негативна й зображена у зворотному коді	$x_1=0,011$ $x_2=1,011$ 0,011 +1,011 1,110 Сума негативна й зображена в додатковому коді

З прикладів випливає, що при використанні зворотного коду в пристрої, який забезпечує підсумовування багаторозрядних двійкових чисел – двійковому суматорі, необхідно передбачити ланцюг циклічного переносу. У випадку використання додаткового коду цей ланцюг відсутній.

З наведеного вище можна зробити наступний висновок. Застосування простого математичного «трюку» (подання двійкових чисел у зворотному або додатковому коді) дозволяє пристосувати двійковий суматор для виконання як операцій додавання двійкових чисел, так і операцій їх віднімання.

Більше того, за допомогою двійкового суматора можна забезпечити також виконання й операцій множення й ділення двійкових чисел (тобто всіх чотирьох арифметичних дій), оскільки множення являє собою послідовне додавання, а ділення – послідовне віднімання.

Одноразрядний двійковий суматор

Підсумовування багаторозрядних двійкових чисел $A = a_n a_{n-1} ... a_0$ і $B = b_n b_{n-1} ... b_0$ проводиться шляхом їх порозрядного додавання з переносом між розрядами. Тому основним вузлом багаторозрядних суматорів є комбінаційний одноразрядний суматор, який виконує арифметичне додавання трьох одноразрядних чисел (цифр): цифри даного розряду першого доданка (a_i), цифри даного розряду другого доданка (b_i) й цифри (1 або 0) переносу із сусіднього молодшого розряду (p_i). У результаті додавання для кожного розряду виходять дві цифри – сума для цього розряду (S_i) й перенос у наступний старший розряд (p_{i+1}).

Таблиця істинності одноразрядного суматора наведена на рис. 1.17.

a_i	b_i	p_i	S_i	p_{i+1}
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

Рис. 1.17. Таблиця істинності одноразрядного суматора

Для синтезу схеми одноразрядного суматора необхідно записати логічні рівняння для функцій виходів S_i і P_{i+1} й виконати перетворення відповідно до правил алгебри логіки. Для цього необхідно перегрупувати вихідне рівняння за прямим і інверсним значенням аргумента p_i, винести їх за дужки й

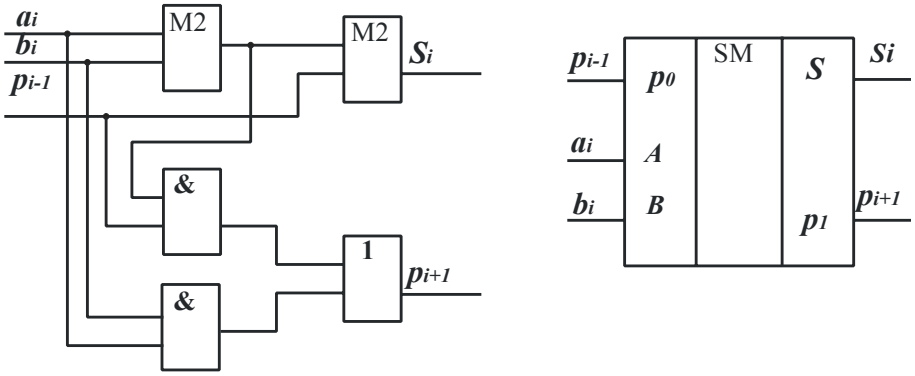
перетворити у функції нерівнозначності (М2). Це перетворення наведено у формулах 13 і 14.

$$S_i = a_i\bar{b}_i\bar{p}_i + \bar{a}_i b_i\bar{p}_i + \bar{a}_i\bar{b}_i p_i + a_i b_i p_i = (a_i\bar{b}_i + \bar{a}_i b_i)\bar{p}_i + (\overline{a_i \oplus b_i})p_i = \\ = (a_i \oplus b_i)\bar{p}_i + (\overline{a_i \oplus b_i})p_i = (a_i \oplus b_i) \oplus p_i \qquad (1.13)$$

Аналогічно перетворюється й рівняння для функції P_{i+1}

$$P_{i+1} = (a_i \oplus b_i)p_i + a_i b_i \qquad (1.14)$$

Умовна позначка й функціональна схема однорозрядного суматора, що побудована відповідно з отриманими виразами, наведені на рис. 1.18.



a) b)

Рис. 1.18. Однорозрядний суматор: а) функціональна схема;
б) умовна позначка

Багаторозрядний паралельний суматор може бути складений з однорозрядних суматорів, число яких дорівнює числу розрядів, що складаються шляхом з'єднання виходу, на якому формується сигнал переносу даного розряду із входом для сигналу переносу сусіднього старшого розряду. Такий спосіб організації переносу називається послідовним.

Багаторозрядний суматор

Приклад побудови чотирирозрядного паралельного суматора демонструє рис. 1.19. У суматорах цього типу перенос поширюється послідовно від розряду до розряду з формуванням суми в кожному розряді. Так, наприклад, при додаванні чисел 11…11 і 00…01 буде мати місце «пробіг» одиниці переносу через увесь суматор від самого молодшого до самого старшого розряду. Даний тип суматора найбільш простий з погляду схеми ланцюгів поширення переносу.

Нескладно побачити, що на базі суматора легко реалізувати й функцію віднімання чисел. Для цього, наприклад, від'ємник повинен бути в додатковому

коді. Останнє досягається інвертуванням розрядів, що віднімається, й подачею (додаванням) «1» до молодшого розряду. Це показано в лівій частині рис. 1.19.

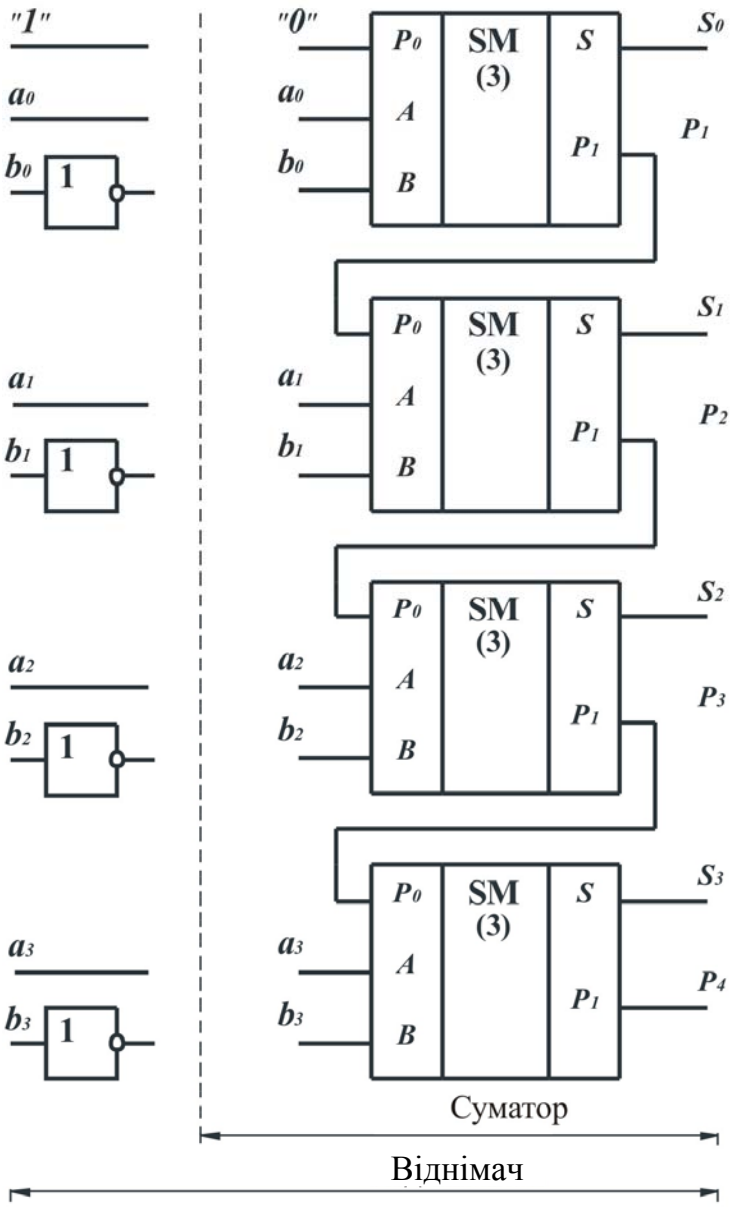


Рис. 1.19. Багаторозрядний суматор-віднімач

Арифметико-логічні пристрої

Вершиною розвитку мікросхемотехніки середнього ступеня інтеграції стали арифметико-логічні пристрої (АЛП), що об'єднали в собі можливість виконання даних логічних і арифметичних операцій. У порівнянні із приладами, що працюють на твердій логіці, АЛП являють собою пристрої більш високого класу. В АЛП перетворення виконуються над багаторозрядними двійковими операндами.

До основних виконуваних операцій відносяться:

– логічні – порозрядні диз'юнкція або кон’юнкція;

– арифметичні – додавання, віднімання, додавання за модулем 2;

– спеціальні – інверсія, інкремент або декремент, зрушення вліво або вправо одного з операндів.

Звичайно в АЛП є дві групи входів для операндів і група входів для вибору виду виконуваної операції. Крім цього, є вихід, що означає виникнення переносу в старшому розряді, й вхід, на який може надходити перенос із молодшого розряду. Це дає можливість збільшувати розрядність оброблюваних операндів шляхом каскадного включення декількох АЛП разом.

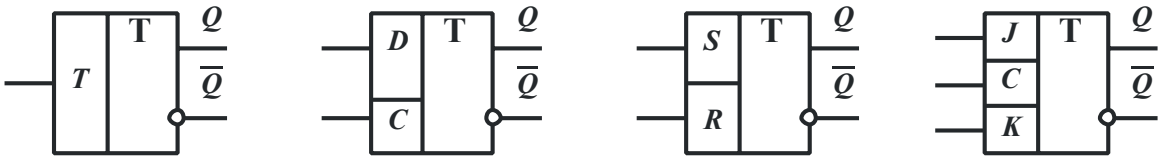
В архітектурі мікроконтролерів АЛП є базовим елементом, за допомогою якого виконується арифметична й логічна обробка даних.

1.2.6. Цифрові пристрої послідовністного типу

Тригери

Тригери – це елементи цифрової, дискретної техніки, що мають здатність перебувати в одному із двох стійких станів протягом необмеженого часу. Перший стан називається «одиничним», інший «нульовим». Тригери мають два виходи, один з яких є прямим Q, а другий інверсним $\overline{Q}$. Прийнято вважати, що стан тригера відповідає сигналу на його прямому виході.

За логікою функціонування розрізняють тригери типів $R-S$, D, T і $J-K$, що відповідає назві їх керуючих входів. Умовне зображення цих тригерів показано на рис. 1.20. За допомогою вхідних сигналів проводиться керування перемиканням або установкою тригера в необхідний стан. До моменту впливу на керуючий вхід тригер зберігає той стан, у котрий його було встановлено. Тимчасові інтервали, протягом яких ці стани зберігаються незмінними, називають тактами або кроками роботи.



а) *б*) *в*) *г*)

Рис. 1.20. Умовні зображення: а) T-тригер; б) D-тригер; в) $R-S$-тригер; г) $J-K$-тригер

Логіка функціонування тригерів залежить не тільки від значення вхідних змінних (керуючих сигналів), що надходять у поточному кроці, але й від стану самого тригера, в якому він перебував на попередньому кроці. Для опису їх роботи застосовуються логічні таблиці виходів і переходів, які часто сполучають. У таких таблицях стану тригери на попередньому й наступних кроках записуються як Q^k і Q^{k+1}. Сполучені таблиці переходів і виходів для T-тригера й D-тригера наведені на рис. 1.21. У цих тригерах є по одному керуючому входу, які позначаються відповідно символами T и D.

T^k	Q^k	Q^{k+1}
0	0	0
1	0	1
0	1	1
1	1	0

а)

C	D^k	Q^k	Q^{k+1}
1	0	0	0
1	1	0	1
1	0	1	0
1	1	1	1

б)

Рис. 1.21. Таблиці переходів і виходів: а) T-тригера; б) D-тригера

З таблиці а) можна побачити, що T-тригер перемикається в протилежний стан щоразу при подачі на його вхід сигналу рівня «1» незалежно від того, в якому стані він перебував. Такий тригер називається рахунковим або «тригер з рахунковим входом». Він використовується як дільник на 2.

Для D-тригера властиво те, що він є повторювачем вхідного сигналу. Щоб забезпечити функцію зберігання інформації, потрібен додатковий керуючий сигнал, за яким дозволяється запис у тригері. Такий сигнал називається синхронізуючим і позначається символом C. У проміжку між синхросигналами стан тригера не змінюється. Функціонування D-тригера будуть розглянуті нижче.

Таблиця переходів і виходів для тригерів $R-S$ і $J-K$ наведена на рис. 1.22. У цих тригерах є окремі входи для установки виходів у стан «1» відповідно S і J і для установки в «0» – входи R і K. Входи R і K називають також входами скидання тригера. За відсутності керуючих сигналів тригери зберігають свій стан.

$R^k\left(K^k\right)$	$S^k\left(J^k\right)$	Q^k	Q^{k+1}	
			$R-S$	$J-K$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	*	1
1	1	1	*	0

Рис. 1.22. Таблиці переходів і виходів $R-S$ і $J-K$-тригерів

Відмінності між $R-S$ і $J-K$-тригерами проявляються в їхніх реакціях на одночасну подачу сигналів рівня «1» на обидва керуючих входи. Для $R-S$-тригера така комбінація сигналів є забороненої, а $J-K$-тригер змінює свій стан на протилежний, перемикаючись як T-тригер.

Окремо слід сказати про деякі інші особливості $J-K$-тригерів. Вони були розроблені, щоб виключити ефект, що одержав назву «перегони тригерів», який проявлявся при паралельній, груповій роботі декількох тригерів. Індивідуальною особливістю мікросхем тригерів, навіть якщо вони того самого типу, є тривалість часу перемикання. Двох однакових немає! А це призводить до того, що при одночасному перемиканні декількох тригерів у якомусь проміжку з'являються неправильні комбінації вихідних сигналів, відбувається збій у роботі всього обладнання.

Функціонально $J-K$-тригери складаються із двох ступенів, тригера, що веде, і веденого. Ці тригери завжди такі, що синхронізуються. Перший ступінь перемикається за фронтом синхроімпульсу, а другий – за зрізом. Час перемикання веденого ступеню значно менше ніж у ведучого. І при груповій роботі декількох тригерів зміна вихідних сигналів відбувається дуже швидко, начебто одномоментно. Це дало можливість виключити появу непередбачених станів тригерів. У цифровій техніці $J-K$-тригери широко застосовуються при побудові лічильників.

За способом керування розрізняють тригери асинхронні та синхронні або такі, що синхронізуються. В асинхронних тригерах перехід до нового стану залежить тільки від стану вхідних керуючих сигналів. У синхронних тригерах додатково потрібно формувати сигнал синхронізації. Їх перемикання відбувається тільки за умови побудови такого сигналу.

За способом сприйняття синхросигналів тригери поділяються на статичні, керовані за рівнем і динамічні керовані за фронтом або зрізом синхроімпульсу. Умовні позначки входів синхронізації показані на рис. 1.23.

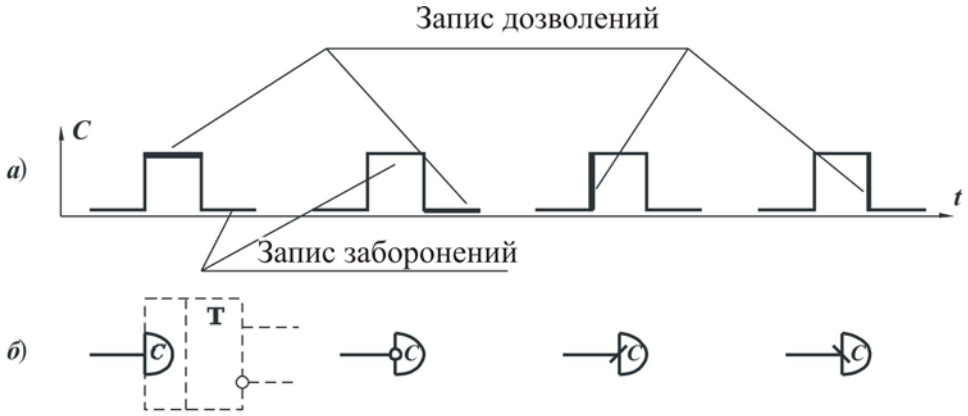


Рис. 1.23. Входи синхронізації: а) тимчасові діаграми;
б) умовні позначки

Найпростішим із тригерів є асинхронний $R-S$-тригер. Його схеми, що виконані на елементах «АБО – НІ» і «І – НІ» наведені на рис. 1.24.

Зіставлення схем а) і б) показує, що функціонування тригера не залежить від типу елементів, на яких вони реалізовані. Відмінністю є те, що тригер, виконаний на елементах «АБО – НІ», керується прямими вхідними сигналами R й S (рівень логічної 1), а на елементах «І – НІ» – інверсними сигналами $\overline{R}$ й

$\overline{S}$ (рівень логічного 0). Для варіанта а) на рис. 1.24 наведено тимчасову діаграму. Тригер встановлюється при $S=1$ й скидається при $R=1$.

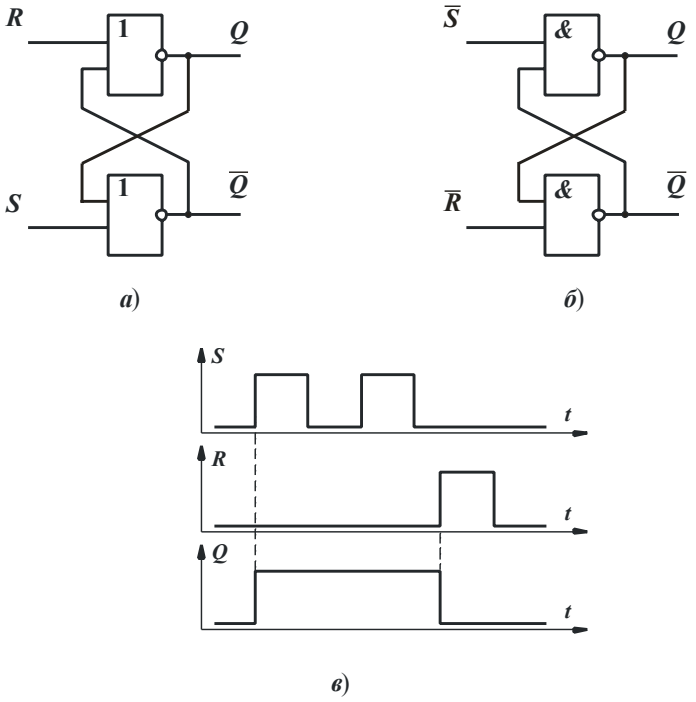


Рис. 1.24. Асинхронний $R-S$-тригер: а) у базисі «АБО – НІ»; б) у базисі «І – НІ»; в) тимчасові діаграми

У синхронному $R-S$-тригері необхідно створити умови, щоб керуючі сигнали R або S надходили на вхід тригера тільки в момент присутності синхроімпульсу. Це легко вирішується за допомогою елементів «2І – НІ». Приклад такої схеми й тимчасові діаграми її роботи наведено на рис. 1.25.

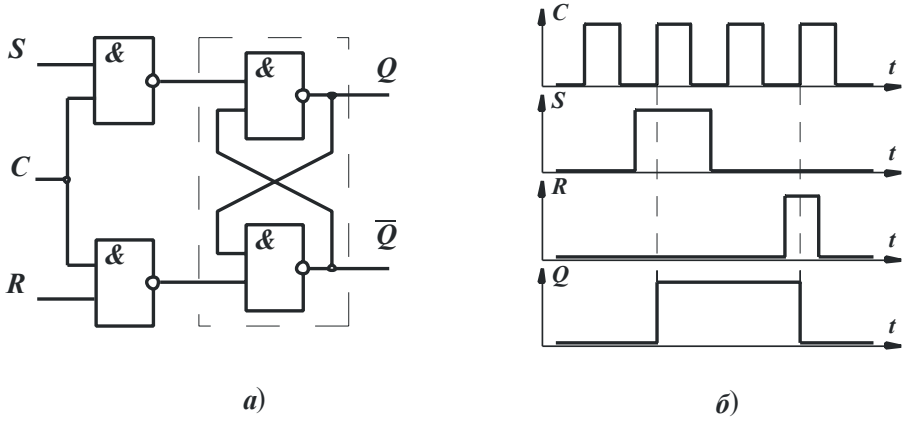


Рис. 1.25. Синхронний $R-S$-тригер: а) функціональна схема; б) тимчасові діаграми

На базі синхронного $R-S$-тригера легко одержати D-тригер. Для цього вводиться зворотний зв'язок (рис. 1.26). Проходження сигналу із входу D можливо тільки за наявності «1» на вході C. Умовне графічне зображення й тимчасові діаграми роботи D-тригера так само можна побачити на рис. 1.26.

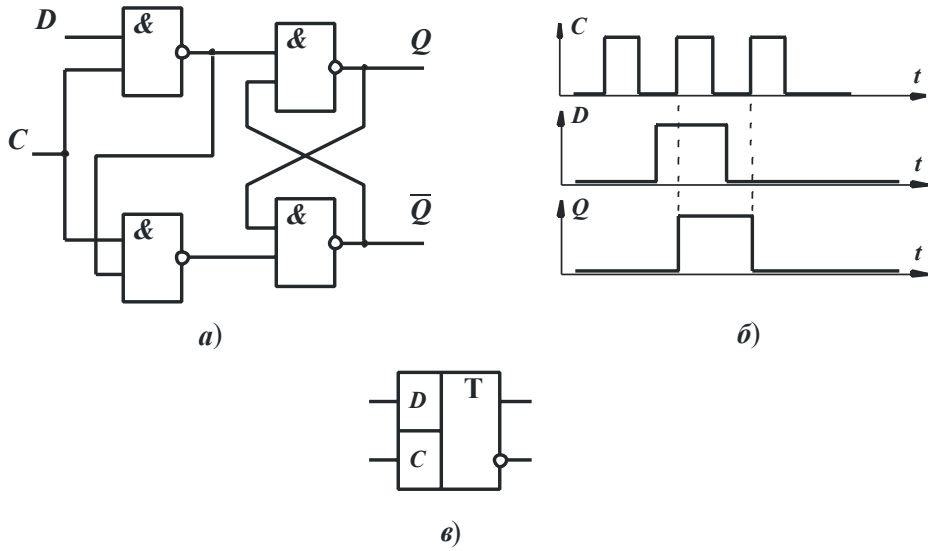


Рис. 1.26. D-тригер: а) функціональна схема; б) тимчасові діаграми; в) умовна позначка

У мікропроцесорних структурах D-тригер є базовим елементом при побудові регістрів, лічильників, а також оперативних запам'ятовувальних пристроїв. На рис. 1.27 зображено приклад реалізації T-тригера на базі D-тригера й тимчасові діаграми роботи.

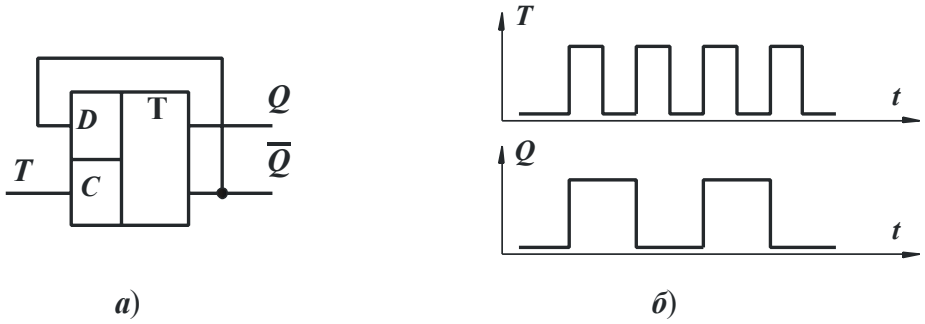


Рис. 1.27. T-тригер: а) функціональна схема; б) тимчасові діаграми

Регістри

Регістрами називаються цифрові пристрої, використовувані для запису й зберігання n-розрядного двійкового слова. Вони будуються на базі тригерів, число яких у схемі регістру звичайно відповідає числу розрядів двійкового слова. Особливістю роботи схеми регістру є те, що запис, зчитування або перетворення інформації відбувається одночасно у всіх його тригерах. У багаторозрядних оперативних запам'ятовувальних пристроях (ОЗП) регістри є базовими елементами. Їх називають комірками пам'яті. У мікроконтролерах розрядність таких гнізд звичайно становить 8 або 16 біт.

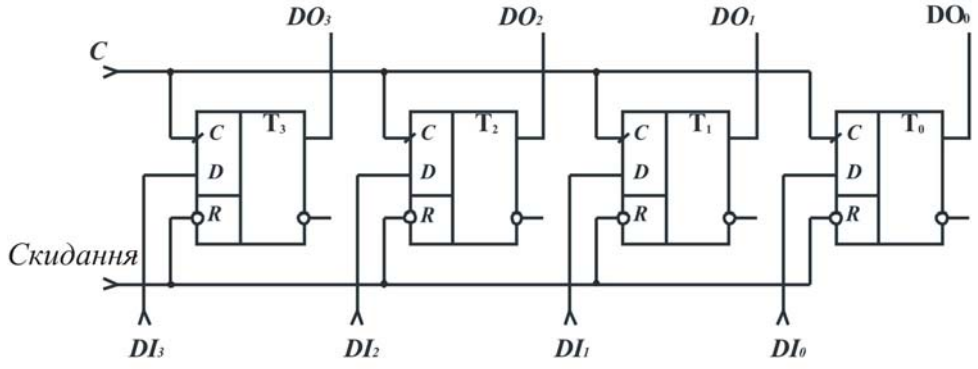
Крім зберігання інформації, в деяких типах регістрів забезпечується можливість виконання додаткових функцій для її перетворення. До них відносяться: перетворення даних з паралельної форми подання в послідовну, і

навпаки; перетворення прямого коду у зворотний; виконання зрушення на один або кілька розрядів убік молодшого або старшого розрядів; інкрементування або декрементирування вмісту регістру. У структурі мікропроцесора це основний елемент.

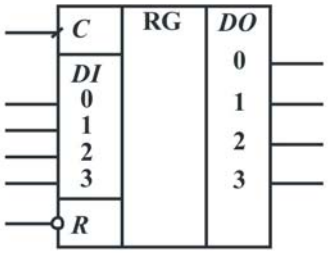
Паралельний регістр

Паралельним називають регістр, у якому n-розрядне двійкове слово записується й зчитується одночасно по всіх n розрядах. Причому, після зчитування інформація, що зберігається в ньому, він не змінюється.

Паралельні регістри найчастіше будуються на синхронних D або $R-S$-тригерах. Приклад схемної реалізації чотирирозрядного паралельного регістру і його умовну позначку наведено на рис. 1.28.



а)



б)

Рис. 1.28. Паралельний регістр:
а) функціональна схема; б) умовна позначка

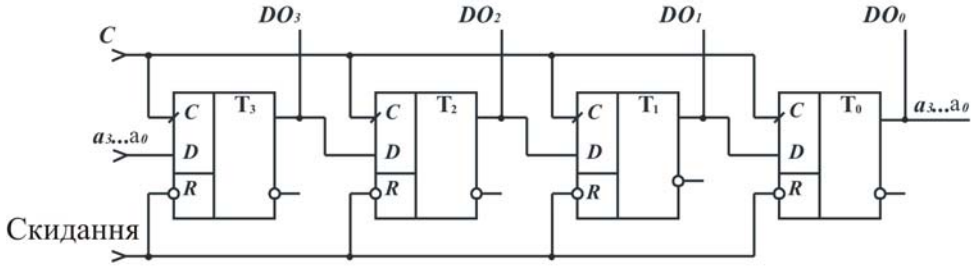
Загальними для розрядів регістру є ланцюги синхронізації (C) й скидання (R). Значення розрядів двійкового слова 0, 1, 2, 3 запишуться у відповідних тригерах тільки після подачі імпульсу синхронізації $C=1$. Після цього тригери переходять у режим зберігання. Зчитування слова з регістру здійснюється із прямих Q-виходів тригерів. Інверсні виходи $(\overline{Q})$ використовуються рідше.

Слід нагадати, що після подачі живлення на схему регістру тригери встановлюються в довільні, заздалегідь непевні стани. Тому в наведеній схемі є окрема лінія установки регістру в «нульовий» або вихідний стан для скидання тригерів. На R-вхід кожного з них подається імпульс скидання. У схемі такий імпульс формується «0», тому що R-входи інверсні.

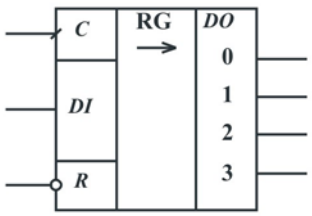
Послідовний регістр

Послідовним називають регістр, у якому здійснюється послідовне (розряд за розрядом) приймання й видача інформації зі зрушенням. Такий регістр називається регістром зрушення. Він являє собою ряд послідовно з'єднаних тригерів, число яких визначається розрядністю двійкового слова. За напрямком зрушення розрізняють регістри прямого зрушення убік молодшого розряду й зворотного зрушення убік старшого розряду. Іноді вживають терміни «зрушення вправо» або «зрушення вліво». Існують також і реверсивні регістри, що допускають зрушення в обох напрямках.

На рис. 1.29 наведено приклад функціональної схеми й умовна позначка чотирирозрядного регістру, що виконує зрушення вправо.



а)



б)

Рис. 1.29. Регістр зрушення вправо: а) функціональна схема;
б) умовна позначка

Регістр побудовано на D-тригерах, які синхронізуються фронтом імпульсу. При записі інформації молодший розряд слова, що вводиться, подається на вхід крайнього лівого тригера T_3. Але запис розряду відбувається тільки при вступі імпульсу синхронізації на вхід C. За наступним синхроімпульсом значення з виходу Q_3 передається в тригер T_2, а на вхід T_3 повинен подаватися наступний розряд слова, що вводиться. Таким чином, із надходженням кожного чергового синхроімпульсу проводиться зрушення інформації на один розряд вправо, і

після четвертого синхроімпульсу регістр буде заповнений усіма розрядами слова, що вводиться.

Для реалізації процедури зрушення вліво використовуються двоступінчасті тригери, у яких за фронтом синхроімпульсу встановлюється перший щабель, а за зрізом – другий. На рис. 1.30 наведено приклад схеми такого регістру. Для передачі інформації з молодшого до старшого розряду вихід кожного тригера молодшого розряду з'єднано із входом тригера старшого розряду. До закінчення синхроімпульсу кожний тригер утримує інформацію, записану на попередньому кроці. Захоплення ж інформації відбувається за фронтом синхроімпульсу, що й забезпечує її зрушення вліво.

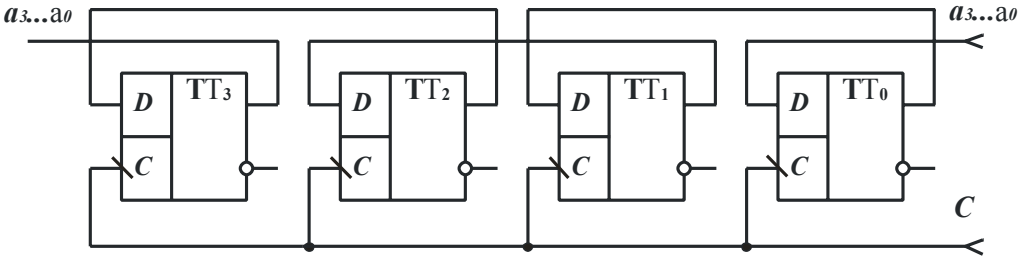


Рис. 1.30. Регістр зрушення вліво

Комбінуючи схеми зрушення вправо й уліво й використовуючи керуючі сигнали, можна побудувати реверсивний регістр. А якщо в схемі забезпечити з'єднання виходу останнього розряду із входом першого, то регістр зрушення легко перетворюється в кільцевий регістр.

Лічильники

Цифровим лічильником називають функціональний вузол, який здійснює підрахунок числа імпульсів, що надходять на його вхід, формує результат обчислення в заданому коді. Такий код звичайно є двійковим й тому ці лічильники називаються двійковими.

Залежно від напрямку підрахунку розрізняють підсумовуючі (із прямим підрахунком), віднімальні (зі зворотним підрахунком) і реверсивні (як із прямим, так і зі зворотним підрахунком) лічильники. Під ємністю лічильника мається на увазі максимальне число імпульсів, яке може бути підраховане лічильником, звичайно 2^k. Де k дорівнює числу розрядів лічильника.

Двійкові лічильники можуть бути побудовані й на синхронних D-тригерах і $J-K$-тригерах, перетворених у T-тригери.

Підсумовуючі двійкові лічильники

На рис. 1.31 наведено схему й тимчасові діаграми чотирирозрядного підсумовуючого двійкового лічильника з ланцюгами послідовного переносу. Інверсний вихід i-го розряду (тригера) з'єднано із входом $(i+1)$-го розряду. Лічильник побудовано на D-тригерах, які перетворені в асинхронні T-тригери, що тактуються фронтом синхроімпульсу.

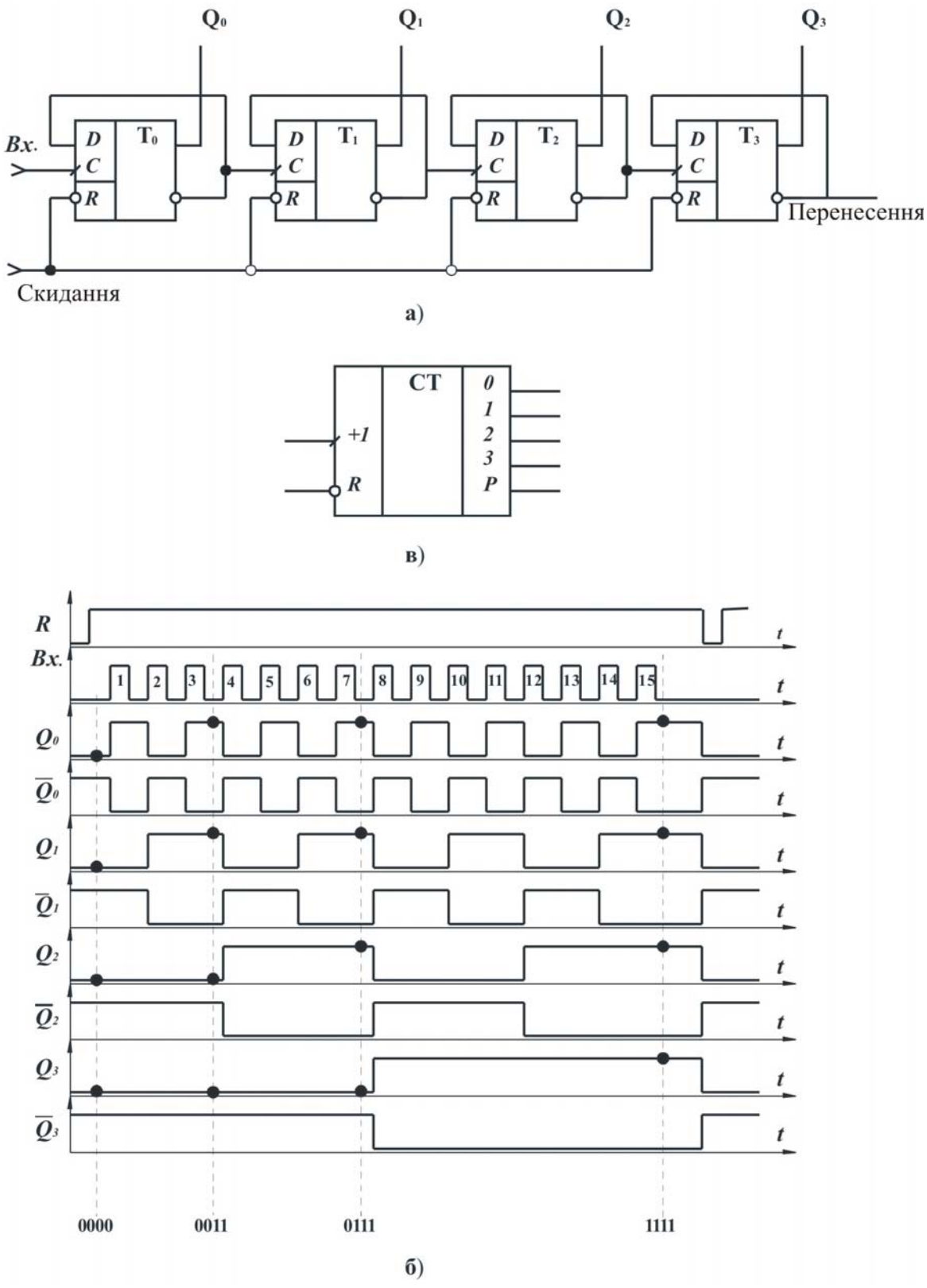


Рис. 1.31. Підсумовуючий двійковий лічильник: а) функціональна схема; б) умовна позначка: в) тимчасові діаграми

Входом лічильника слугує вхід крайнього лівого тригера (T_0). Двійковий код результату підрахунку формується на виходах тригерів Q_0, Q_1, Q_2, Q_3 (Q_0 –

молодший, а Q – старший). Ємність розглянутого лічильника $2^4 = 16$, тому максимальне його показання відповідне до подачі на вхід 15 рахункових імпульсів. 16-й рахунковий імпульс установлює всі тригери у вихідний (нульовий) стан, отже, шина «скидання» (установка «0») необхідна лише на початку роботи лічильника.

Після подачі кожного чергового вхідного імпульсу T-тригер переходить у протилежний стан. З тимчасової діаграми можна побачити, що період проходження імпульсів на виходах кожного розряду у два рази більший, ніж на його вході. У будь-який момент часу стан тригерів лічильника однозначно визначає число імпульсів, що надійшли на його вхід.

1.2.7. Організація пам'яті

Запам'ятовувальні пристрої (ЗП) призначені для зберігання інформації (даних). Вони поділяються на постійні запам'ятовувальні пристрої (ПЗП) і оперативні запам'ятовувальні пристрої (ОЗП).

ПЗП (ROM – Read Only Memory) – пристрій для зберігання й зчитування незмінних даних. Цей вид пам'яті призначено для зберігання програми, а також незмінних констант. Така пам'ять є енергонезалежною – при відключенні живлення записана інформація зберігається. Після поновлення живлення вона може бути зчитана.

ОЗП (RAM – Random-Access Memory) – пристрій для запису, зберігання й зчитування змінюваних даних. У даному виді пам'яті зберігається інформація, що модифікується й використовується в процесі роботи. Це можуть бути різні змінні або результати проміжних і остаточних обчислень. Інформація при такому виді пам'яті губиться після вимикання живлення.

Основним параметром пам'яті є її обсяг, тобто кількість бітів інформації, які можуть у ній зберігатися. Для позначення обсягу використовуються наступні спеціальні одиниці:

- 1К – це 1024 (210) і читається як «кіло-» або «ка-». Обсяг пам'яті приблизно дорівнює однієї тисячі;
- 1М – це 1048576 (220) і читається як «мега-». Обсяг пам'яті приблизно дорівнює одному мільйону;
- 1Г – це 1073741824 (230) і читається як «гіга-». Обсяг пам'яті приблизно дорівнює одному мільярду.

За видом організації пам'ять може бути однорозрядною з індивідуальним доступом до кожного біта та багаторозрядною з паралельним доступом до групи бітів. Групова організація має байтову структуру, від одного до 2^n байтів. У свою чергу кожний байт складається з 8 бітів.

Під коміркою пам'яті може матися на увазі як однорозрядна, так і багаторозрядна структура. У другому випадку в позначенні додається цифра, що вказує на кількість одночасно доступних розрядів. Наприклад, RAM 2Kx8 має 2048 восьмирозрядних комірок пам'яті типу ОЗП.

Оперативні запам'ятовувальні пристрої

Основою елементарної комірки ОЗП є тригер. На рис. 1.32 зображена її функціональна схема. Запам'ятовування й зберігання інформації відбувається в D-тригері, який має динамічний вхід синхронізації із записом інформації за фронтом синхросигналу. Вихід тригера увімкнено через електронний ключ (ЕК), що реалізує лінію із трьома станами.

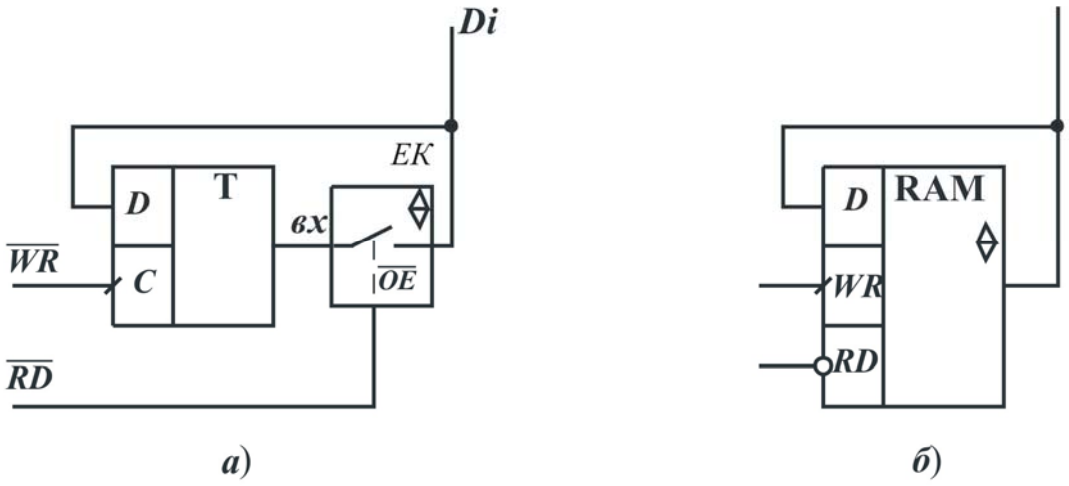


Рис. 1.32. Елементарна комірка ОЗП

Функціональна схема ЕК показана на рис. 1.33. Якщо на керуючому вході OE присутній сигнал високого рівня, то транзистори $VT1$ й $VT4$ замкнені й вихідна лінія перебуває у високоімпедансному стані. Еквівалентно це показано у вигляді розімкнутих ключів. Низький рівень сигналу призводить до відмикання транзисторів $VT1$, $VT4$. На вихідній лінії буде сигнал такого самого рівня, що й на вході. Оскільки ключ, виконаний на транзисторах $VT2$, $VT3$, інвертує сигнал, для забезпечення функції повторення є інвертор на елементі $D2$.

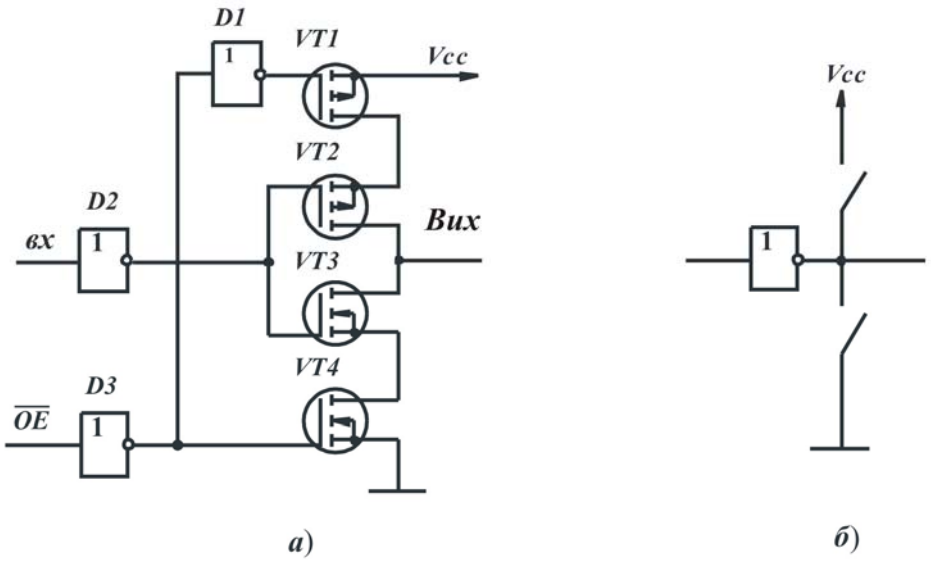


Рис. 1.33. Електронний ключ

Організація комірки пам'яті у такий спосіб дає можливість об'єднати її вхід і вихід на одній лінії даних Di. Для керування записом і читанням інформації є керуючі сигнали WR й RD. Сигнал WR подається на синхровхід тригера, а сигнал RD – на вхід керування електронним ключем OE. Тимчасові діаграми запису й читання наведені на рис. 1.34.

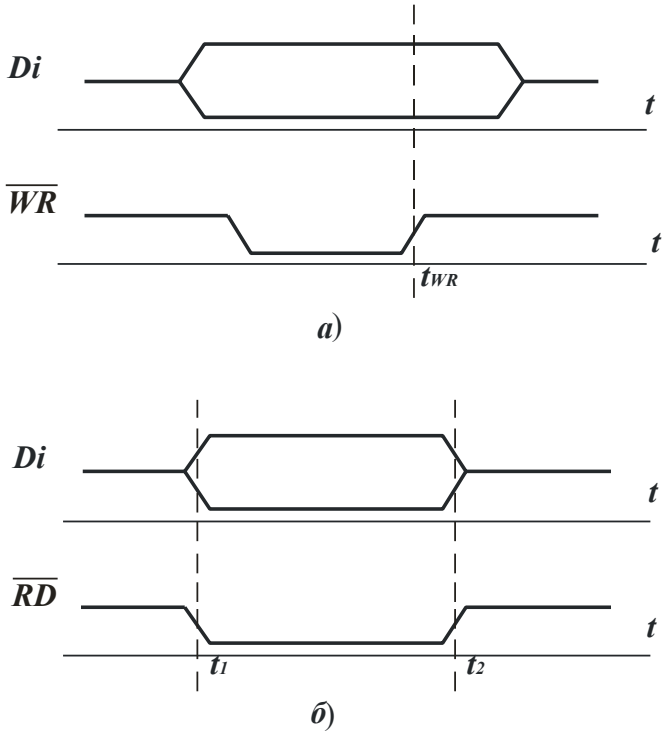


Рис. 1.34. Тимчасові діаграми: а) запис ОЗП; б) читання ОЗП

При записі інформації дані виставляються на лінію Di, після чого формується строб-імпульс на лінії WR. Установка або скидання тригера відбувається за фронтом строб-імпульсу (t_{WR}). Під час читання інформації дані із тригера виставляються на лінію Di в проміжку часу між $t_1 - t_2$ та у момент дії сигналу RD.

Для організації багаторозрядного ОЗП елементарні, однорозрядні комірки пам'яті необхідно об'єднати в реєстрові структури. На рис. 1.35 зображена функціональна схема 8-розрядного ОЗП ємністю 2^n. Вибір конкретної лінійки пам'яті проводиться за допомогою дешифратора адреси, на виходи якого подається відповідна адресна комбінація $A0...An$. Таким чином, кожна комірка пам'яті має адресу, за якою зберігається вміст (інформація).

В ОЗП є три керуючих входи. Для запису або читання – входи WR, RD, для дозволу роботи дешифратора – CS. Декодування адреси й установка 1 на одному з виходів дешифратора відбувається тільки під час подачі сигналу нульового рівня на вхід CS. Високий рівень на одному з виходів дешифратора дозволяє проходження сигналу WR або RD до відповідної лінійки комірок в ОЗП. За сигналом WR проводиться паралельний запис у пам’ять інформації із

шини даних $D0...D7$, а за сигналом RD – читання або видача на шину даних інформації з пам'яті.

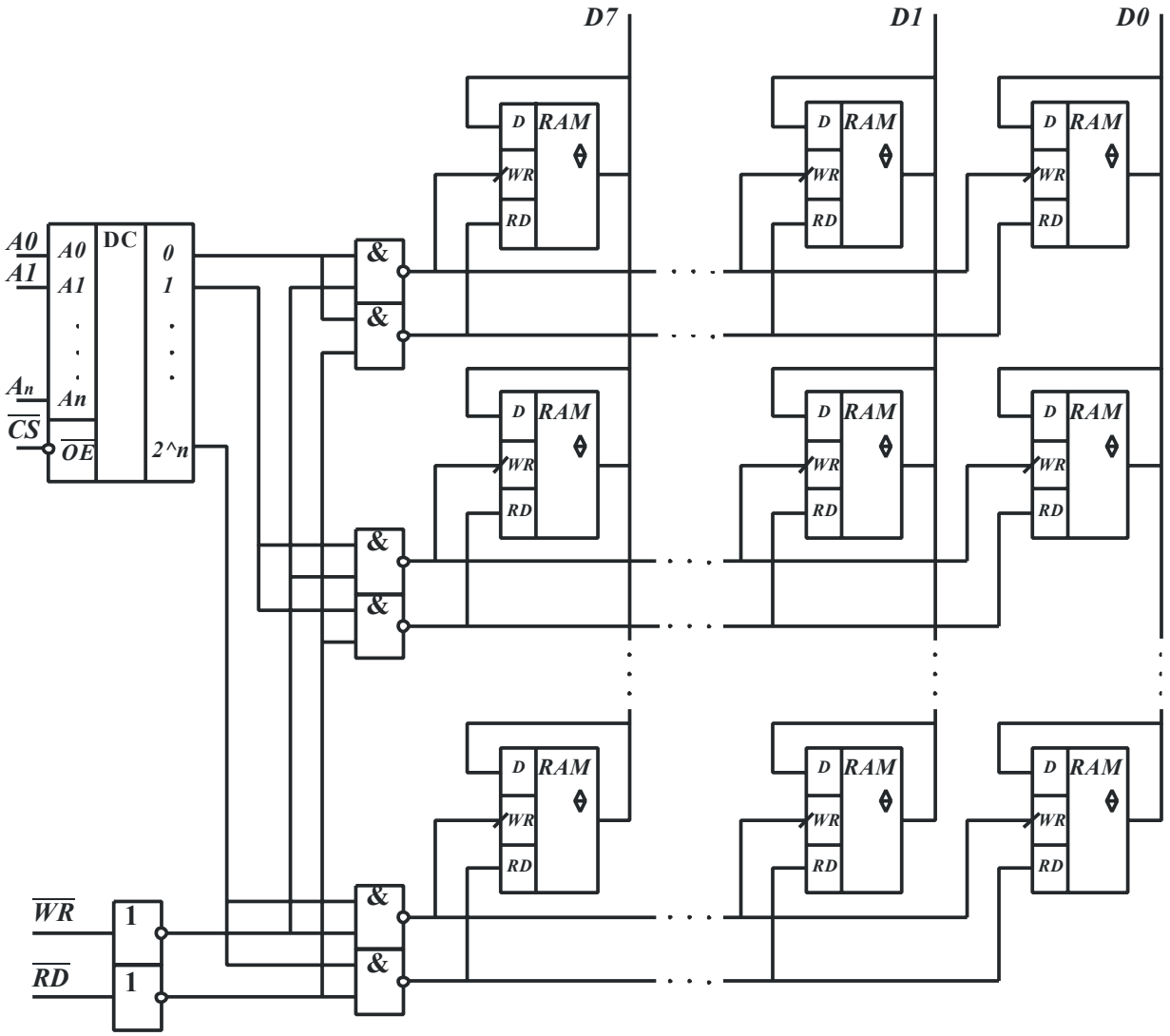


Рис. 1.35. Організація багаторозрядного ОЗП

Умовне зображення ОЗУ й тимчасові діаграми його роботи зображені на рис. 1.36. З діаграм можна побачити, що в першу чергу повинна виставлятися адреса комірки пам'яті, потім сигнал CS і далі під час читання сигнал RD, а при записі – спочатку дані й тільки після цього сигнал WR.

Постійні запам'ятовувальні пристрої

Постійні запам'ятовувальні пристрої (ПЗП) різняться за принципом запису й стирання інформації. Сам процес занесення або запису інформації в ПЗП називається «програмуванням». На сьогоднішній момент існує п'ять видів ПЗП.

ROM (Read Only Memory) – ПЗП програмуються одного разу на стадії виготовлення. Цей вид застосовують при великому тиражі виробів, коли є завдання зниження їх вартості.

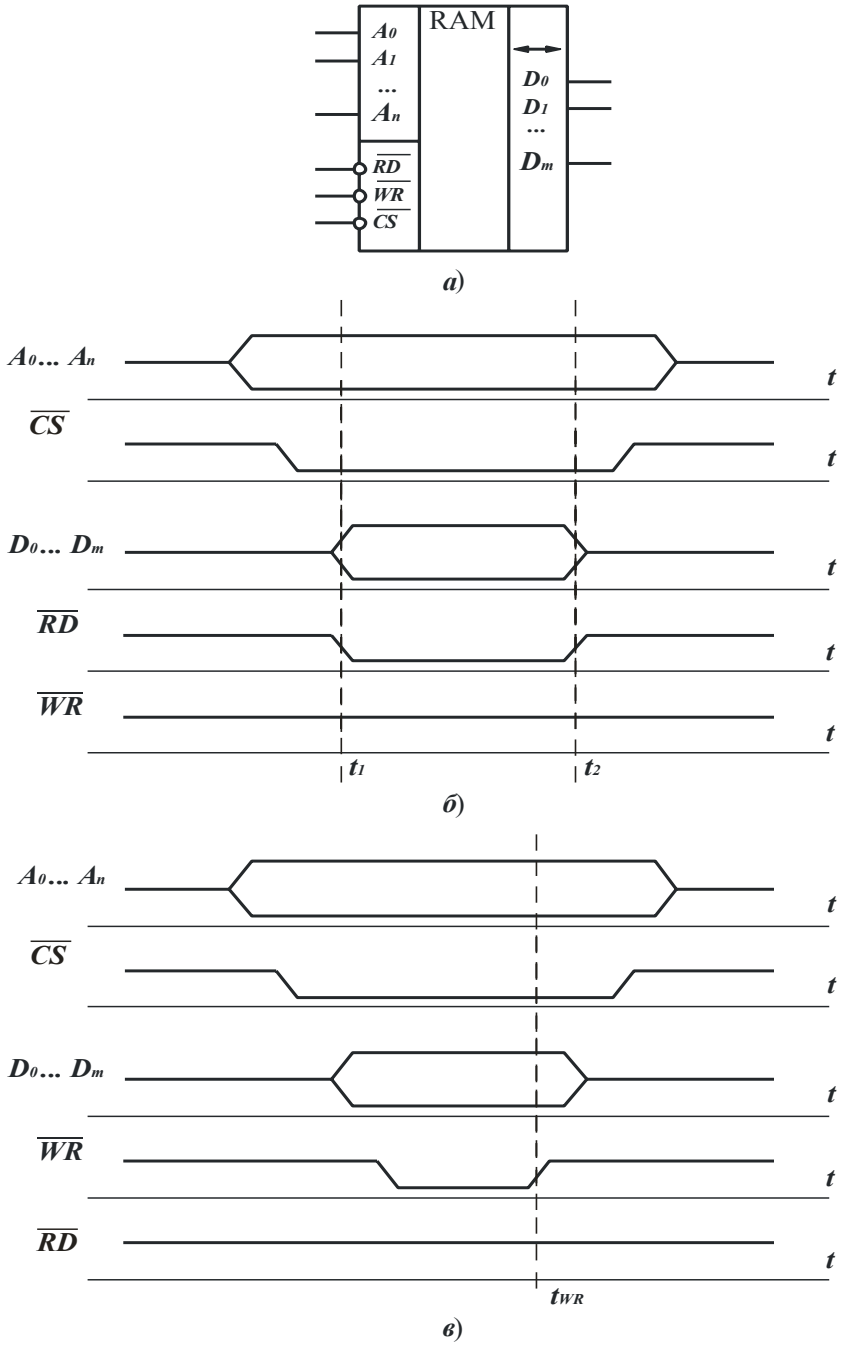


Рис. 1.36. Багаторозрядне ОЗП: а) умовна позначка;
б), в) тимчасові діаграми читання й запису

PROM (Programmable Read Only Memory) – однократно програмувальні ПЗП. Запис інформації в них здійснюється за допомогою спеціального пристрою, який називається програматором.

EPROM (Erasable Programmable Read Only Memory) – багаторазово перепрограмувальні ПЗП. Запис інформації здійснюється за допомогою програматора, стирання – шляхом ультрафіолетового опромінення мікросхеми через спеціальне вікно в корпусі.

EEPROM (Electrically Erasable Programmable Read Only Memory) – електрично програмувальні та такі ПЗП, що стираються. Запис і стирання інформації здійснюється за допомогою програматора.

Flash Memory – електрично програмувальні та такі ПЗП, що стираються. Інформація може записуватися й стиратися за допомогою електричних сигналів без застосування спеціальних програматорів. Використовується найчастіше для енергонезалежного зберігання даних.

У сучасних мікроконтролерах для зберігання програм широко використовуються ПЗП типу EEPROM із багаторозрядною організацією доступу до даних. У якості елементарної комірки пам'яті використовується тригер, що виконано на базі польового транзистора з утримуючою ємністю. Функціональна схема такої комірки зображена на рис. 1.37. При вирядженій ємності $C1$ транзистор $VT1$ замкнено і на його виході втримується високий рівень сигналу (логічна 1). Заряд ємності проводиться при програмуванні ПЗП й у тих бітах, які повинні зберігати стан нуля. Для керування зарядом і розрядом ємності $C1$ використовуються електронні ключі ЕК1 і ЕК3.

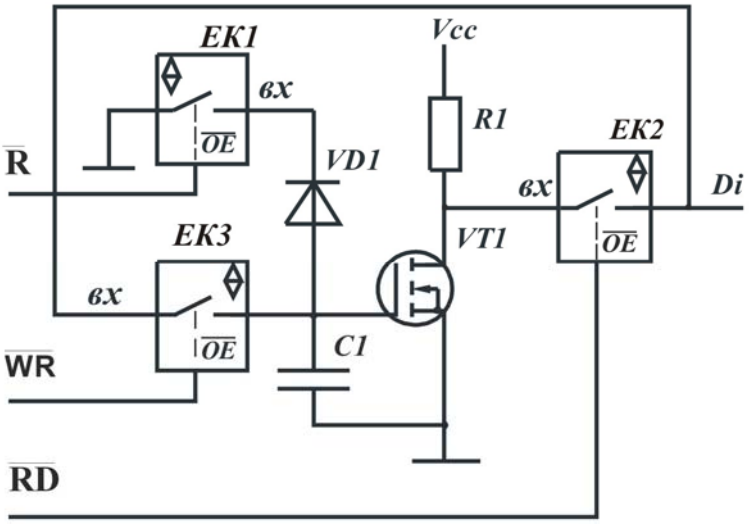


Рис. 1.37. Елементарна комірка ОЗП

При стиранні ПЗП керуючий сигнал подається на ключ ЕК1 і ємність розряджається через діод $VD1$. Після стирання всі комірки пам'яті встановлюються в одиничний стан. Керування записом інформації виконується за допомогою ключа ЕК3. Під час його відмикання на ємність $C1$ надходить сигнал з лінії даних (Di). Його високий рівень виробляє заряд ємності. Зчитування інформації із $VT1$ проводиться через ключ ЕК2, для чого повинен бути сформований керуючий сигнал на лінії RD. Оскільки електронні ключі забезпечують лінії із трьома станами, то вхід і вихід комірки пам'яті об'єднано.

Спосіб організації ПЗП є аналогічним організації багаторозрядного ОЗП. При звичайній роботі із ПЗП проводиться тільки зчитування інформації, тому на схемах умовних позначок лінії запису й стирання часто не зображуються. На рис. 1.38 наведено умовну позначку й тимчасову діаграму роботи ПЗП.

У ПЗП є два керуючих входи RD й CS, а також адресні входи $A0...An$. Кількість адресних входів визначає ємність пам'яті. Так, при n рівному 14 ємність ПЗП становить 32К. З тимчасових діаграм можна побачити

послідовність формування вхідних сигналів. Спочатку встановлюється адреса, потім формується сигнал CS і далі сигнал RD. Читання інформації повинно відбутися в проміжку між $t_1 ... t_2$.

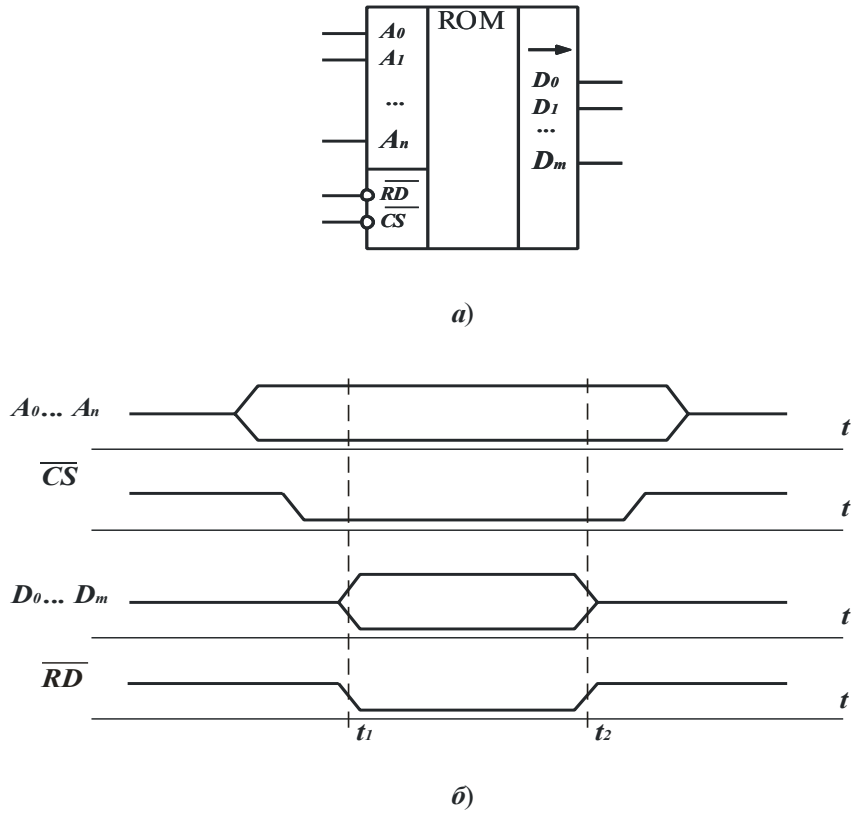


Рис. 1.38. ПЗП: а) умовна позначка; б) тимчасові діаграми

1.2.8. Методи й способи реалізації дискретних і цифрових СУ

На сучасному етапі реалізація дискретних і цифрових пристроїв може бути виконана різними методами, а також на базі різних технічних засобів. Це може бути метод традиційного логічного проектування з використанням дискретних інтегральних схем типу «І», «АБО», «НІ»; або схем середнього ступеня інтеграції, типу дешифраторів або мультиплексорів, їх реалізація з використанням ВІС ПЗП або ПЛМ, нарешті, з використанням програмних методів на базі мікропроцесорної техніки. Усе це є інструментом для розв'язку поставленого завдання. Кожний зі способів має свої переваги й недоліки. Вирішальними факторами під час вибору способу реалізації можуть виступати ступінь складності завдання, вимоги швидкодії або економічна доцільність.

Невеликі й досить прості логічні функції, а відповідно й пристрої можуть бути реалізовані за допомогою дискретних вентильних схем або на ВІС ПЗП, ПЛМ. Крім того, існує чимало стандартних елементів, які випускаються промисловістю. Наприклад, перетворювачі двійково-десяткового коду в двійковий, двійкового коду в додатковий і код знаків, багаторозрядні суматори, арифметико-логічні пристрої та багато інших. У сукупності це цілком може задовольнити потреби розробника. При такому підході слід ураховувати, що якщо

надалі виникне необхідність зміни алгоритму функціонування пристрою, то це приведе до його повної переробки. Фактично – до створення нового пристрою.

Застосовуючи програмні способи реалізації, часто достатньо лише зробити перепрограмування наявного пристрою без будь-якої зміни апаратних засобів. Інакше кажучи, програмний підхід дозволяє застосовувати ті самі елементи пристрою для реалізації складних логічних функцій будь-якого виду. Для цієї мети переважно використовуються однокристальні програмувальні процесори (мікропроцесори) або більш функціонально повні мікроконтролери. Програмно можна реалізувати як логічні, так і математичні функції.

При програмній реалізації логічні функції зображують у вигляді структурних схем, за якими надалі створюється відповідна програма. Між логічними булевими функціями й структурними схемами програм існує взаємно однозначна відповідність. Вони еквівалентні. У цьому можна переконатися на наступному прикладі. Припустимо, необхідно програмно реалізувати логічні функції «І» і «АБО». Структурні схеми програм, відповідні до зазначених функцій, зображено на рис. 1.39.

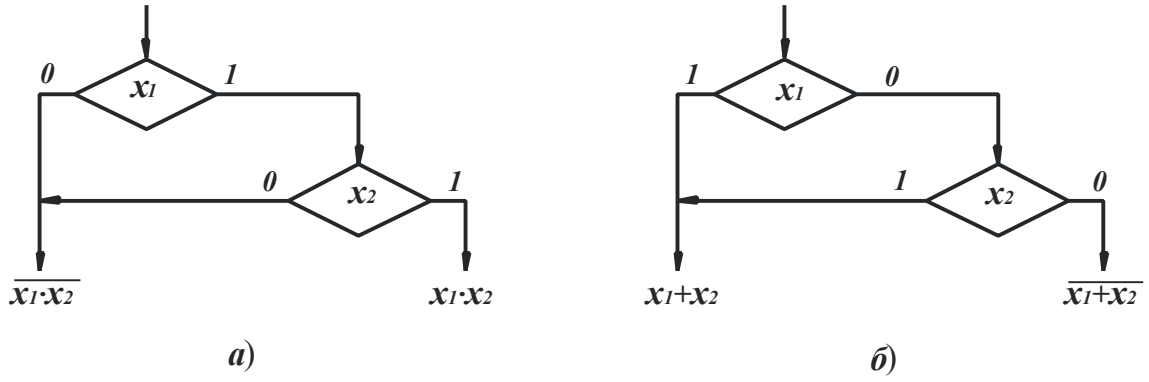


Рис. 1.39. Структурні схеми програм

Як при використанні дискретної схемотехніки, так і їхніх програмних еквівалентів можна отримати логічну функцію будь-якої складності. Причому з меншими апаратними витратами, оскільки вони формуються тим самим пристроєм. Однак функції в цьому випадку формуються послідовно в часі, тобто знижується швидкодія їх виконання. При використанні інтегральної схемотехніки обидві функції можна було б реалізувати одночасно. Ця обставина іноді буває вирішальною на користь застосування вентильних схем.

Таким чином, будь-яка логічна функція, реалізована програмним шляхом, може бути перетворена в еквівалентно реалізовану апаратними засобами, і навпаки. Якщо функція (закон) керування являє собою логічну функцію комбінаційного типу або послідовністного, то реалізація на ПЗП або на ПЛМ являє собою програму, яку потрібно записати в ВІС. При зміні функції керування переписується тільки програма.

Однак у багатьох випадках у процесі реалізації закону керування необхідно здійснювати обчислення керуючих впливів. Вони виконуються у свою чергу за іншими законами. Тобто необхідно мати пристрої керування

(ПК), (АЛУ) і пам'ять для зберігання програми обчислень, а це є загальновідомою структурою ЕОМ. У цій структурі ПК й АЛУ являють собою процесор ЕОМ. Коли досягнення технології дозволили ПК й АЛУ розмістити на одному кристалі, тоді й з'явився термін мікропроцесор (префікс мікро- більше стосується розмірів і вартості мікропроцесора).

Контрольні запитання

1. Які види інформації можуть надходити на вхід системи керування?
2. Що таке об'єкт керування? Які бувають різновиди об'єктів керування?
3. Що таке логічна функція керування?
4. Що є аргументами функції керування?
5. Сформулювати основні закони булевої алгебри.
6. Сформулювати правила складання таблиць істинності.
7. Навести приклади базових логічних елементів.
8. Скласти таблицю істинності для повного лінійного дешифратора на 4 входи. Навести приклад функціональної схеми.
9. Навести приклади функціональних схем мультиплексора й демультиплексора на базі дешифратора на 3 входи.
10. Скласти таблицю істинності для суматора за модулем 2. Навести приклад функціональної схеми.
11. Навести приклади віднімання двійкових чисел для випадків, коли зменшуване більше від’ємника й коли зменшуване менше від'ємника.
12. Скласти таблицю істинності для однорозрядного двійкового суматора. Навести приклад функціональної схеми.
13. Що таке тригер? Які бувають різновиди тригерів?
14. Скласти таблицю переходів і виходів D-*Тригера*. Навести його функціональну схему й тимчасову діаграму.
15. Скласти таблицю переходів і виходів Т-*Тригера*. Навести його функціональну схему, виконану на базі D-*Тригера*, й тимчасову діаграму.
16. Що таке паралельний регістр? Навести приклад його функціональної схеми.
17. Що таке послідовний регістр і які бувають його різновиди? Навести приклади їх функціональних схем.
18. Призначення й принцип роботи двійкового суматора.
19. Призначення й різновиди запам'ятовувальних пристроїв.
20. Які одиниці використовуються для виміру ємності пам'яті?
21. Розкрити принцип побудови елементарної комірки ОЗУ. Навести приклад функціональної схеми й тимчасової діаграми роботи.
22. Розкрити принцип побудови багаторозрядного ОЗУ. Навести приклад функціональної схеми.
23. Розкрити принцип побудови елементарної комірки ПЗУ. Навести приклад функціональної схеми й тимчасову діаграму роботи.
24. Які методи й способи застосовуються для реалізації дискретних і цифрових систем керування?

2. ЗАГАЛЬНІ ВІДОМОСТІ ПРО МІКРОКОНТРОЛЕРИ

Перші мікропроцесорні пристрої з'явилися на ринку в 1973 р. Фірма Intel запропонувала власне мікропроцесорний комплект – сам мікропроцесор Intel 8080 у вигляді одного кристала й комплект периферійних пристроїв у вигляді окремих ВІС, що містять паралельний інтерфейс, контролер переривань, таймер триканальний, контролер прямого доступу до пам'яті й модуль послідовного інтерфейсу. Побудова систем керування, що працюють у реальному часі, коли реакція системи на мінливі параметри об'єкта повинна бути миттєвою (швидкою), виходила громіздкою і недешевою. Удосконалення мікропроцесорів (збільшення розрядності й частоти тактування відбувається відповідно до прогресу технологій, а для систем керування, працюючих у реальному часі, фірмою Intel була запропонована структура ВІС, у якої на один кристал, крім мікропроцесора, інтегровано всі периферійні пристрої: паралельний інтерфейс, таймери, контролер переривань, послідовний інтерфейс, пам'ять програм і пам'ять даних. Це дозволило скоротити апаратні витрати й здешевити системи керування. Такі ВІС одержали назву мікроконтролери (МК). Перший МК фірмою Intel мав марку 8051 АН (у СРСР було випущено аналог ДО1816ВЕ51), а сім’я мікроконтролерів одержала позначення MCS-51.

Архітектура сімейства MCS-51 виявилася настільки вдалою, що серед 8-розрядних мікроконтролерів на світовому ринку вже багато років вона займає лідируючі позиції. Крім фірми Intel, мікроконтролери з архітектурою MCS-51 випускають фірми Siemens, Philips, Atmel та ін.

Удосконалення технології й підвищення ступеня інтеграції розширюють функції периферійних пристроїв, збільшують продуктивність, обсяги пам'яті програм і даних, але саме ядро команд MCS-51 залишається повністю сумісним з молодшими моделями, що дозволяє використовувати величезний накопичений досвід розробки й налагодження програмного забезпечення.

2.1. Структура й функціональні можливості базової моделі MCS-51 (МК51)

Незважаючи на велику різноманітність мікроконтролерів різних виробників, обчислювальне ядро МК51 у всіх контролерів однакове. МК51 містить 8-розрядний мікропроцесор, генератор тактових імпульсів, схеми керування й синхронізації, внутрішню пам'ять програм і внутрішню пам'ять даних, два таймери, паралельні порти вводу/виводу, контролер переривання. Мікросхема живиться від одного джерела напругою +5В, потужністю до 1,5 Вт, допускає експлуатацію в діапазоні температур від –40 до +100°С. Загальними характеристиками для всієї сім’ї МК51 є:

- 32 двонаправлені лінії вводу/виводу, зображені у вигляді чотири-, 8-розрядних портів;
- два 16-розрядних таймери-лічильники;
- контролер переривання;

- 16-розрядний лічильник команд (PC);
- 16-розрядний покажчик даних DPTR;
- синхронно-асинхронний приймач-передавач послідовного порту зі змінюваною швидкістю передачі;
- генератор тактових імпульсів (ГТІ);
- усі вхідні й вихідні сигнали узгоджуються ТТЛ рівнем.

МК51 випускається в корпусі DIP-40. Принципову схему МК51 наведено на рис. 2.1, а призначення виводів – в табл. 2.1.

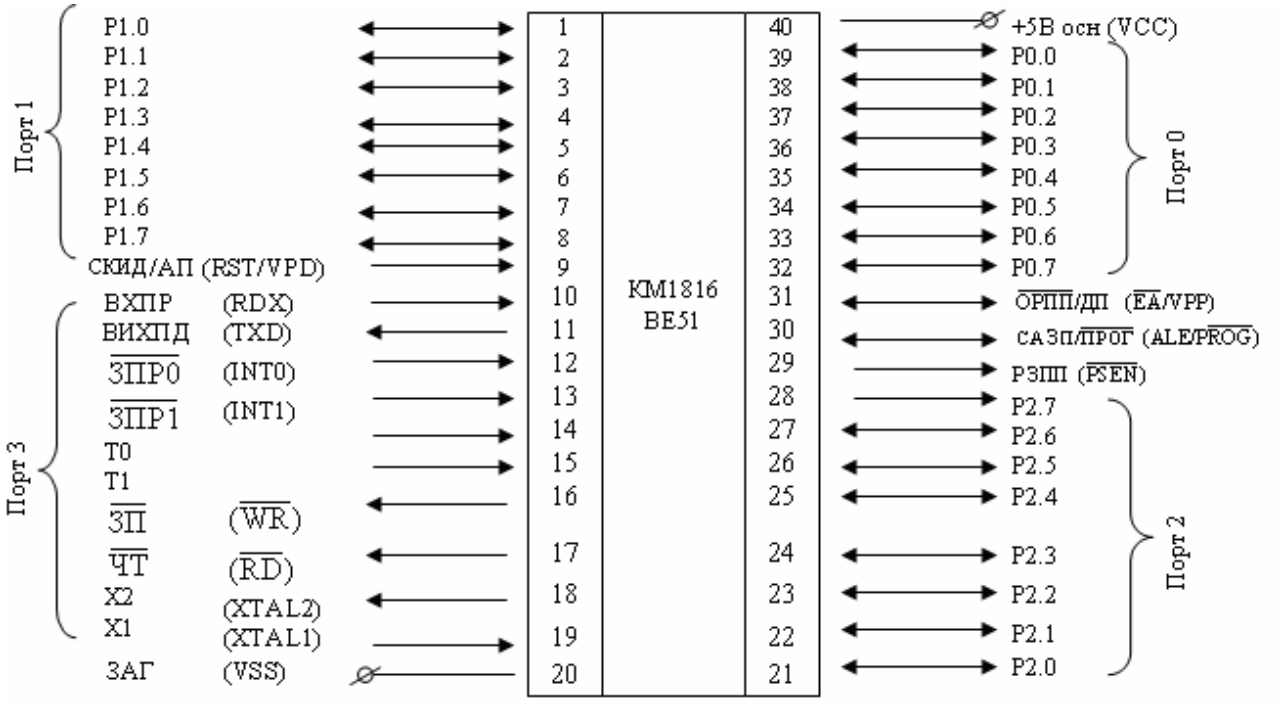


Рис. 2.1. Принципова схема мікроконтролера МК51

Таблиця 2.1

Призначення виводів мікроконтролера МК51

Позначення	Тип	Функція виводу або групи виводів
Порти вводу/виводу		
P0.0…P0.7	Вхід-Вихід	Порт 0 (P0) – восьмирозрядний двонаправлений порт вводу/виводу, передачі коду адреси (молодший байт) або коду даних у мультиплексному режимі під час звертання до зовнішньої пам'яті, вводу/виводу при програмуванні й під час перевірки РПП
P1.0…P1.7	Вхід-Вихід	Порт 1 (P1) – восьмирозрядний квазідвонаправлений порт вводу/виводу для обміну інформацією із зовнішніми пристроями. Використовується також для вводу молодших розрядів коду адреси під час програмування й перевірки РПП МК

Продовження табл. 2.1

P2.0…P2.7	Вхід-Вихід	Порт 2 (P2) – восьмирозрядний квазідвонаправлений порт вводу/виводу, передачі старших розрядів коду адреси під час звертання до зовнішньої пам'яті, а також для вводу старших розрядів коду адреси й сигналів керування під час програмування й перевірки РПП
P3.0…P3.7	Вхід-Вихід	Порт 3 (P3) – восьмирозрядний квазідвонаправлений порт вводу/виводу, звичайно використовується для реалізації периферійних функцій
P3.0	Вхід	Rxd – вхід приймача послідовного порту в асинхронному режимі або вхід-вихід даних у синхронному режимі
P3.1	Вихід	TxD – вихід передавача послідовного порту в асинхронному режимі або видача синхроімпульсів у синхронному режимі
P3.2	Вхід	INT0 – вхід запиту від зовнішнього джерела переривання з умовним номером 0
P3.3	Вхід	INT1 – вхід запиту від зовнішнього джерела переривання з умовним номером 1
P3.4	Вхід	T0 – вхід таймера-лічильника з номером 0
P3.5	Вхід	T1 – вхід таймера-лічильника з номером 1
P3.6	Вихід	$\overline{WR}$ – «Запис» – вихідний сигнал запису байта в зовнішню пам'ять даних (ЗПД). Активний рівень сигналу – логічний «0»
P3.7	Вихід	$\overline{RD}$ – «Читання» – вихідний сигнал читання байта із зовнішньої пам'яті даних (ЗПД). Активний рівень сигналу – логічний «0»
Сигнали керування МК		
ALE/#PROG	Вихід (вхід)	Строб адреси зовнішньої пам'яті. Використовується для керування режимом мультиплексування адреси й даних, які передаються через порт P0 при звертанні до ЗП. Якщо ALE=1, на виводах порту P0 перебуває адреса. При програмуванні МК на вивід подається логічний «0»
#PSEN	Вихід	Дозвіл зовнішньої пам'яті програм. Виконує роль строба приймання байта команди в МК під час вибірки команд із ЗПП. Активний рівень сигналу – логічний «0»
#EA/VPP	Вхід	Сигнал вимкнення резидентної пам'яті програм (РПП). Якщо подано #EA=1, будуть виконуватися команди, розміщені в РПП, якщо (PC)=0000….0FFFH. Якщо подано #EA=0, будуть виконуватися команди, розміщені тільки у ЗПП (РПП повністю недоступна). Під час програмування МК на цей вивід подається імпульс напругою +21В

Закінчення табл. 2.1

#EA/VPP	Вхід	Сигнал вимкнення резидентної пам'яті програм (РПП). Якщо подано #EA=1, будуть виконуватися команди, розміщені в РПП, при цьому (PC)=0000….0FFFH. Якщо подано #EA=0, будуть виконуватися команди, розміщені тільки у ЗПП (РПП повністю недоступна). Під час програмування МК на цей вивід подається імпульс напругою +21В
RST/VRD	Вхід	Сигнал скидання МК (тобто переведення в початковий стан). Рівень сигналу 3,5 В повинен утримуватися не менш ніж 2 мкс. Використовується також для увімкнення аварійного джерела живлення
Сигнали синхронізації МК		
XTAL1	Вхід	Вхід підсилювача-генератора синхросигналів. Підключається до зовнішнього джерела синхронізації (кварцового резонатора, увімкненого за схемою з «середньою крапкою») (рис. 2.5)
XTAL2	Вихід	Вихід підсилювача-генератора синхросигналів. Підключення є аналогічним підключенню XTAL1
Ucc	-	Підключення до джерела живлення напругою Ucc=+5 В±10%
Vss	-	«Загальний» вивід

Внутрішню структуру МК51 наведено на рис. 2.2. Основні функціональні вузли МК51 об'єднано двонапрямленою 8-розрядною магістраллю. Усі операції виконуються в АЛП, яке являє собою 8-розрядний пристрій паралельного типу, що виконує всі арифметичні операції (+, -, *, :), логічні (AND, OR, NOT, XOR), зрушення, скидання, установку бітів. Найважливішою особливістю АЛП є можливість роботи із бітами, що адресуються прямо в ОЗП й РСФ. АЛП оперує із чотирма видами інформації: -1 біт, -4 біта (тетрада), -8 бітів (тайм), -16 бітів (адреса).

Більшість команд МК51 виконуються з використанням акумулятора – регістру з бітовою адресацією, що забезпечує значне спрощення аналізу даних з бітовою структурою інформації. При використанні акумулятора передбачено два види адресації:

- регістрову, код адреси розміщений в коді операції команди (як <A>);
- пряму, при цьому вказується пряма адреса акумулятора (E0H) або символьне чи «ACC» в адресній частині команди.

Регістр PSW є регістром ознак або регістром стану програми. У ньому формуються ознаки результатів, що виконуються в АЛП. Регістр PSW забезпечує керування вибором робочого банку регістрів, допускає побітову адресацію з використанням символічних імен PSW.3, PSW.4.

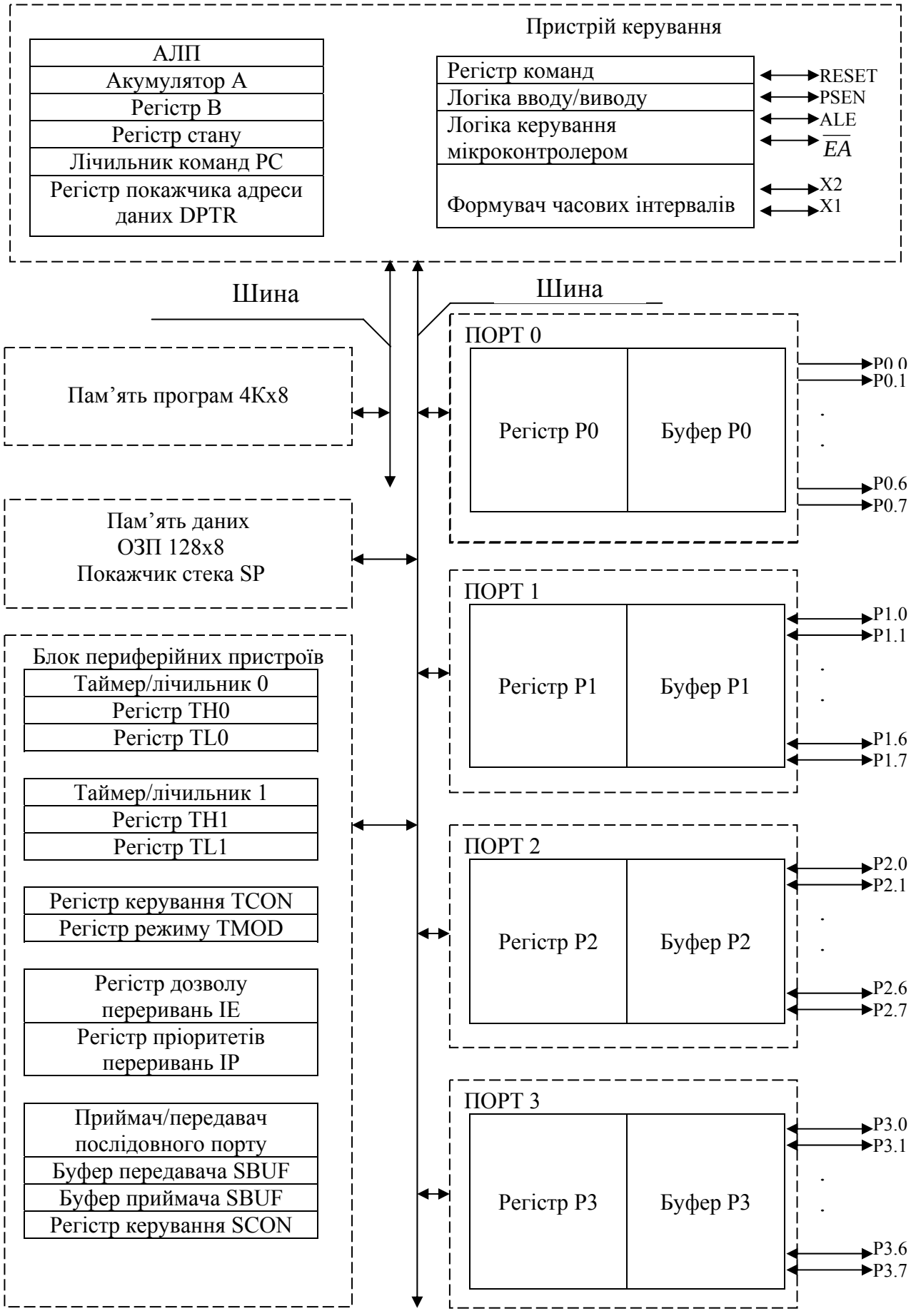


Рис. 2.2. Внутрішня структура МК51

Вказівний регістр DPTR має двобайтову структуру і є базовим при звертанні до зовнішньої пам'яті. Якщо зовнішня пам'ять відсутня, DPTR можна використовувати як регістр загального призначення.

Функціональну структуру зв'язків наведено на рис. 2.3, з якої видно, що всі периферійні пристрої взаємодіють із мікропроцесором через контролер переривань шляхом формування сигналу – запит на переривання (два сигнали від Таймерів T0, T1, два сигнали від зовнішнього середовища контролера INT0, INT1) і один сигнал від буфера послідовного обміну ($RI \vee TI$).

Призначення окремих розрядів регістру PSW наведено в табл. 2.2.

Таблиця 2.2

Призначення розрядів регістру PSW

<table>
<tr><th>Назва біта</th><th>Позиція</th><th>Призначення</th></tr>
<tr><td>C</td><td>PSW.7</td><td>Прапор переносу. Встановлюється й скидається апаратно при виконанні арифметичних, логічних і бітових операцій, а також програмно</td></tr>
<tr><td>AC</td><td>PSW.6</td><td>Прапор допоміжного переносу. Встановлюється й скидається тільки апаратно при виконанні команд підсумовування й віднімання у випадку виникнення переносу або позики в біті 3 акумулятора</td></tr>
<tr><td>F0</td><td>PSW.5</td><td>Вільний прапор. Може бути змінено програмно й використовується за призначенням, встановленим програмістом</td></tr>
<tr><td>RS1
RS0</td><td>PSW.4
PSW.3</td><td>Вибір банку регістрів. Біти встановлюються й скидаються програмно для вибору активного (робочого) банку регістрів:
<table>
<tr><th>RS1</th><th>RS0</th><th>Активний банк</th><th>Адреси РПД</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>00H-07H</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>08P-0FH</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>10H-17H</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>18H-1 FH</td></tr>
</table></td></tr>
<tr><td>OV</td><td>PSW.2</td><td>Прапор переповнення. Встановлюється й скидається апаратно при виконанні арифметичних операцій у випадку переповнення акумулятора. Дає можливість коректно виконувати дії над числами, зображеними в додатковому коді</td></tr>
<tr><td>-</td><td>PSW.1</td><td>Не використовується</td></tr>
<tr><td>P</td><td>PSW.0</td><td>Прапор паритету. Встановлюється й скидається апаратно в кожному циклі команди, фіксує факт непарної кількості «1» в акумуляторі</td></tr>
</table>

Роботу мікроконтролера розділено на машинні цикли, а машинні цикли – на машинні такти (рис. 2.4).

Кварцовий резонатор підключається до зовнішніх виводів X1 і X2 МК51 (рис. 2.5, а) і визначає частоту коливань внутрішнього генератора, який формує всі сигнали керування й синхронізації. При подачі напруги живлення в системі завжди формується сигнал скидання (RST) (рис. 2.5, б) і після цього робота мікроконтролера поділяється на машинні цикли, а машинні цикли – на такти (рис. 2.4).

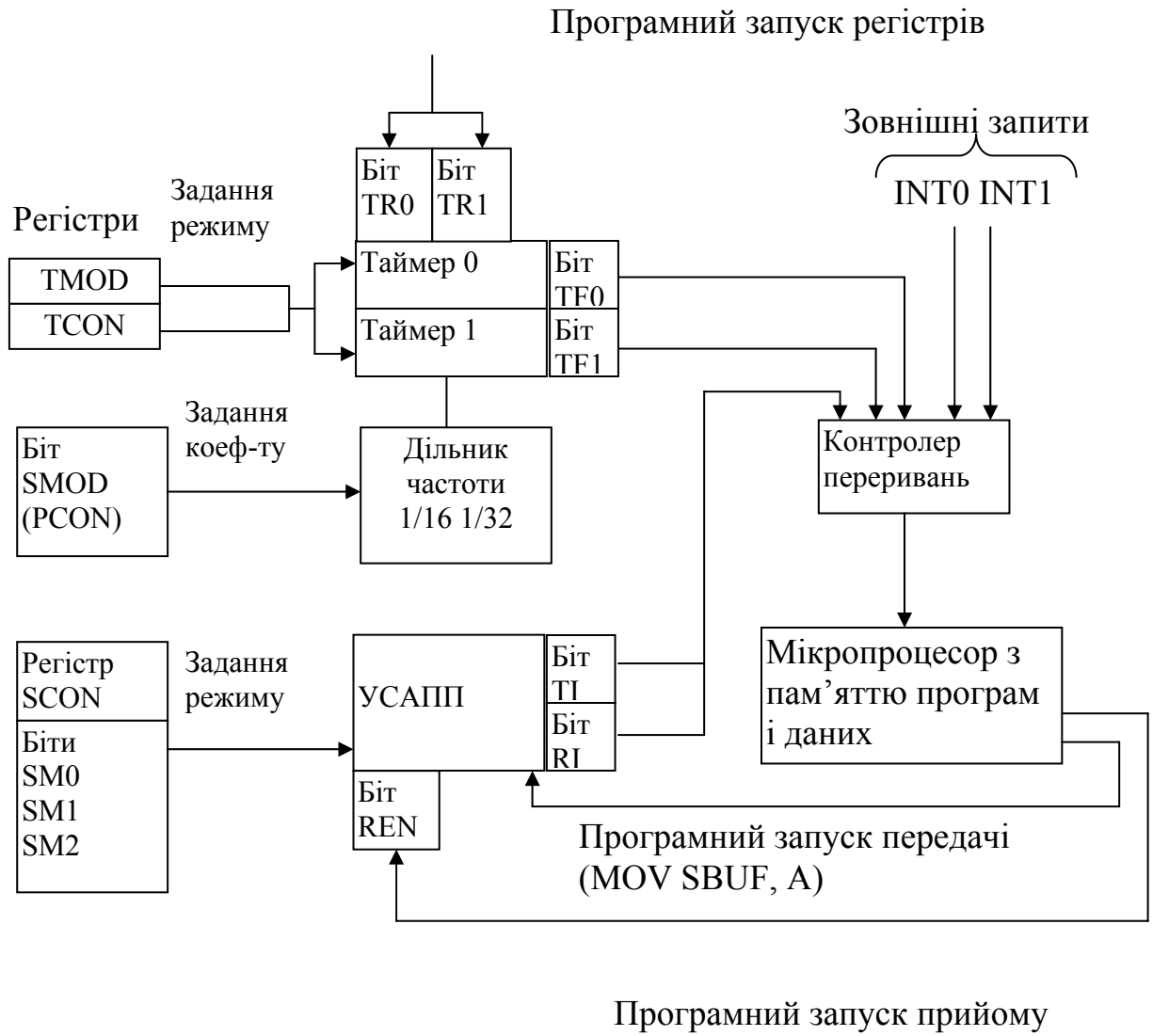


Рис. 2.3. Функціональна структура зв'язків внутрішніх вузлів МК51

Машинні цикли мають фіксовану тривалість, яка дорівнює 12 періодам коливань резонатора або шести станам первинного керуючого автомата (S1-S6). Кожний стан керуючого автомата містить дві фази (P1, P2) сигналів резонатора. У фазі P1, як правило, виконуються операції в АЛП, а у фазі P2 – межрегістрові пересилання інформації. Увесь машинний цикл складається з 12 фаз, починаючи з фази S1P1 і закінчуючи фазою S6P2 (рис. 2.4). Усі заштриховані сигнали, наведені на рис. 2.4, є внутрішніми й недоступні користувачеві для спостереження.

Спостережуваними є сигнали резонатора й сигнал строба адреси зовнішньої пам'яті (САВП), який формується двічі за один машинний цикл

(S1P2-S2P1 і S4P2-S5P1). Цей сигнал використовується для керування процесом пересилання інформації між МК51 і зовнішньою пам'яттю.

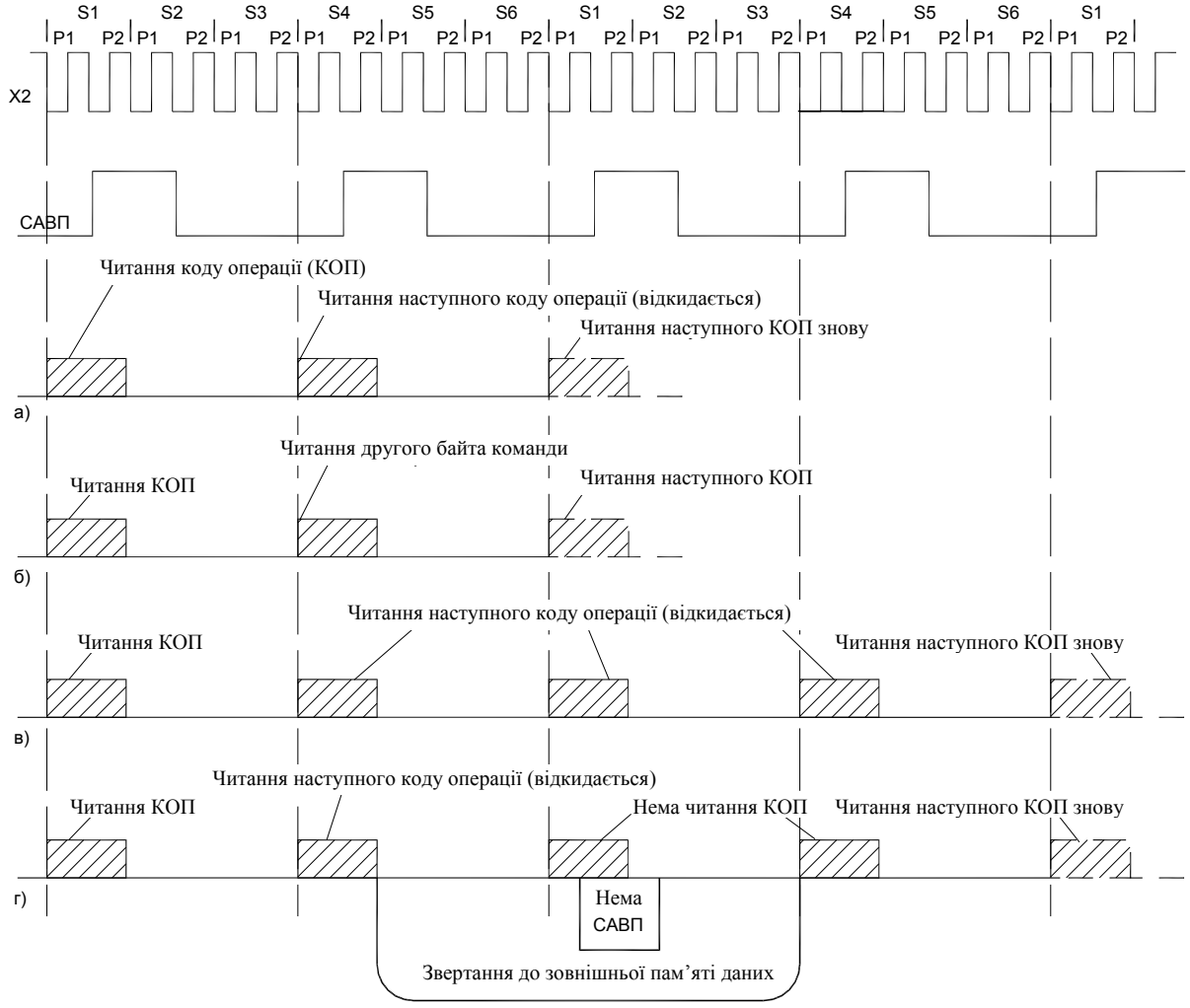


Рис. 2.4. Основні машинні цикли роботи МК51

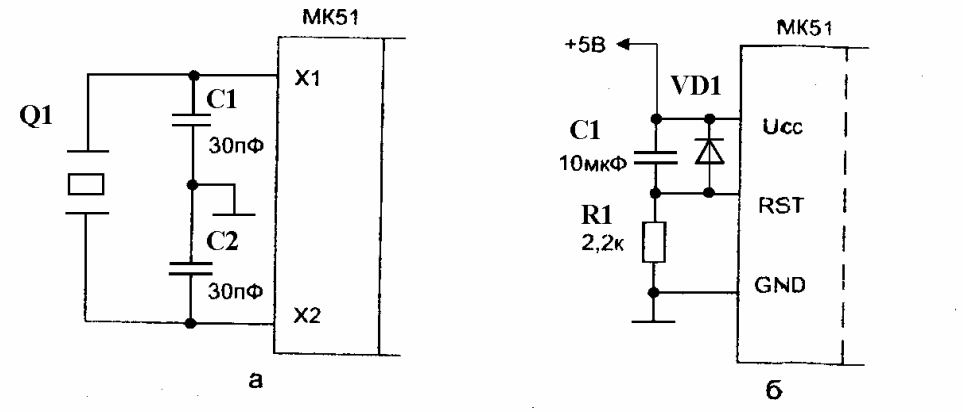


Рис. 2.5. Схеми підключення кварцових резонаторів до МК51

2.2. Програмно доступні ресурси МК51

У МК51 використовується гарвардська архітектура, у якої пам'ять програм і даних розділено, тобто мають свої шини адреси й даних.

Структуру пам'яті можна зобразити у вигляді (рис. 2.6). Пам'ять програм розділено на внутрішню (4 Кбайт) і зовнішню (60 Кбайт). При звертанні до молодших 4 К адрес сигнал $\overline{EA} = 0$. При звертанні до старших 60 К адрес сигнал $\overline{EA} = 1$.

Пам'ять даних також розділено на зовнішню й внутрішню. Внутрішня пам'ять даних у базовій моделі МК51 має розмір 256 байт – 128 байт внутрішнього ОЗП й 128 адрес регістрів спеціальних функцій.

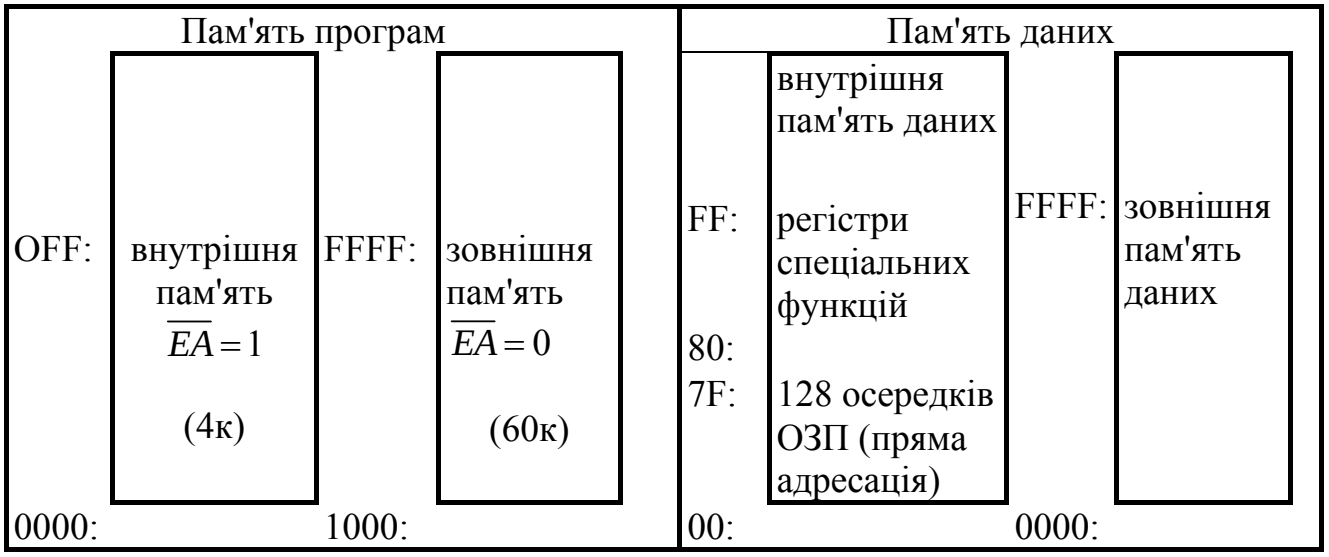


Рис. 2.6. Адресний простір мікроконтролера

З них молодші 32 адреси визначено за банками регістрів від 00F до 1FH. Осередки з адресами 20H до 2FH визначено за бітами, що адресуються прямо.

Область внутрішнього ОЗП визначена з адреси ООН до 7FH; (рис. 2.7). Чотири банки регістрів загального призначення мають по вісім байтів у кожному. При роботі мікропроцесора один з банків вважається робочим (активним). Вибір робочого банку здійснюється програмно, установкою/скиданням двох бітів у регістрі стану програми (PSW). Адресація до робочих регістрів здійснюється символічними іменами R0… R7. Регістри R0 і R1 кожного банку можуть використовуватися для непряморегістрової адресації до 128 осередків внутрішньої пам'яті даних.

Стекова пам'ять (пам'ять магазинного типу – першим увійшов, останнім вийшов) служить для запам'ятовування проміжних даних при виконанні підпрограм. Адресація до осередків стекової пам'яті здійснюється за допомогою 8-розрядного покажчика стека (SP). Звичайно стекова пам'ять розміщується у внутрішній пам'яті даних (ОЗП), починаючи з адреси 40H и вище. При увімкненні живлення або за сигналом скидання (RESET) в SP автоматично завантажується код 07H. При завантаженні стека командами PUSH вміст SP

збільшується, а при вивантаженні командами POP і RET, RETI вміст SP зменшується. Шляхом запису в SP значень від 0 до 127 адресна область стека може бути розміщена в будь-якій області адресного простору РПД.

Шістнадцяткова адреса							Десяткова адреса		
7F:									127
2F:	7F	7E	7D	7C	7B	7A	79	78	47
2E:	77	76	75	74	73	72	71	70	46
2D:	6F	6E	6D	6C	6B	6A	69	68	45
2C:	67	66	65	64	63	62	61	60	44
2B:	5F	5E	5D	5C	5B	5A	59	58	43
2A:	57	56	55	54	53	52	51	50	42
29:	4F	4E	4D	4C	4B	4A	49	48	41
28:	47	46	45	44	43	42	41	40	40
27:	3F	3E	3D	3C	3B	3A	39	38	39
26:	37	36	35	34	33	32	31	30	38
25:	2F	2E	2D	2C	2B	2A	29	28	37
24:	27	26	25	24	23	22	21	20	36
23:	1F	1E	1D	1C	1B	1A	19	18	35
22:	17	16	15	14	13	12	11	10	34
21:	0F	0E	0D	0C	0B	0A	09	08	33
20:	07	06	05	04	03	02	01	00	32
1F: 18:	R7 . . R0				БАНК3				31 24
17: 10:	R7 . . R0				БАНК2				23 16
0F: 08:	R7 . . R0				БАНК1				15 8
07: 00:	R7 . . R0				БАНК0				7 0

Рис. 2.7. Структура внутрішньої пам'яті даних

Шістнадцяткова адреса							Десяткова адреса			Ім'я регістру
0FF:										
0F0:	F7	F6	F5	F4	F3	F2	F1	F0	240	B
0E0:	E7	E6	E5	E4	E3	E2	E1	E0	224	ACC
0D0:	CY D7	AC D6	F0 D5	RS1 D4	RS0 D3	OV D2		P D0	208	PSW
0B8:			PT2 BD	PS BC	PT1 BB	PX1 BA	PT0 B9	PX0 B8	184	IP
0B0:	B7	B6	B5	B4	B3	B2	B1	B0	176	P3
0A8:	EA AF		ET2 AD	ES AC	ET1 AB	EX1 AA	ET0 A9	EX0 A8	168	IE
0A0:	A7	A6	A5	A4	A3	A2	A1	A0	160	P2
99:									153	SBUF
98:	SM0 9F	SM1 9E	SM2 9D	REN 9C	TB8 9B	RB8 9A	TI 99	RI 98	152	SCON
90:	97	96	95	94	93	92	91	90	144	P1
8D:									141	TH1
8C:									140	TH0
8B:									139	TL1
8A:									138	TL0
89:									137	TMOD
88:	TF1 8F	TR1 8E	TF0 8D	TR0 8C	IE1 8B	IT1 8A	IE0 89	IT0 88	136	TCON
87:									135	PCON
83:									131	DPH
82:									130	DPL
81:									129	SP
80:	87	86	85	84	83	82	81	80	128	PO

Рис. 2.8. Структура адресного простору регістрів спеціальних функцій

128 старших адрес внутрішньої пам'яті даних відведені для регістрів спеціальних функцій (РСФ) (рис. 2.8). Призначення регістрів показано на рис. 2.8. Усі регістри мають, крім адреси, як осередки внутрішньої пам'яті даних, так і символічні імена. Частина регістрів допускає побітову адресацію. До РСФ можна звертатися як до звичайних осередків РПД. Більш детальне призначення й функціональні характеристики регістрів спеціальних функцій (РСФ) буде розглянуто при вивченні периферійних пристроїв.

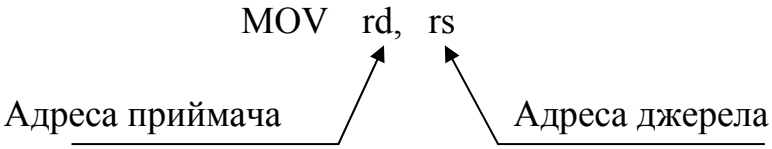
2.3. Методи адресації в МК51

Система команд МК51 має свої особливості, саме ті, які властиві методам обробки й аналізу інформації в системах керування: гнучкість

вводу/виводу інформації через паралельні й послідовні порти, первинна обробка інформації й особливо можливість виконання операцій з бітами.

У кожній команді є повідомлення мікропроцесору про характер (вид) виконуваної операції й методи доступу до операндів, тобто їхньої адреси.

Командний рядок мови асемблер містить мітку (символічна адреса), мнемонічне зображення команди (символічне ім'я, звичайно це перші букви англійських слів), операнди й коментарі. При перерахуванні операндів першим вказується приймач, другим – джерело інформації.



Наприклад, MOV R1, A – переслати вміст акумулятора в регістр R1 поточного банку.

Методи адресації являють собою набір принципів доступу до операндів.

При складанні програм розробник повинен добре знати можливі адресні ресурси мікроконтролера й набір методів адресації до них.

Команди мікроконтролера МК51 мають однобайтову, двобайтову й трибайтову структуру. Перший байт команди завжди код операції, який повідомляє про характер виконуваних операцій мікропроцесором над операндами.

7				0
Мнемокод	Старші біти коду операції	n_2	n_1	n_0
MOV A, R_n	11101	R_n		

Рис. 2.9. Структура однобайтової команди при регістровій адресації

В однобайтових командах з регістровою адресацією (рис. 2.9) адреса одного операнда визначає регістр поточного банку, наведений у команді. Номер регістру вказується трьома молодшими бітами байта коду операції. П'ять бітів байта КОП, що залишилися, визначають адресу другого операнда й характер виконуваних операцій над операндами.

В однобайтових командах з непряморегістровою адресацією (рис. 2.10) для вказівки адреси одного операнда використовуються регістри-покажчики (R0 і R1), які перебувають у регістрових банках 0, 1, 2, 3. Номер регістру вказує молодший біт байта коду операції $(i = 0,1)$, номер поточного банку визначається бітами регістру PSW (PSW.3 і PSW.4).

7	0
Кіп операції	i
MOV A	$@R_i$

Рис. 2.10. Структура команди при непряморегістровій адресації

Команда MOV @Ri,A виконує пересилання вмісту акумулятора в комірку внутрішньої пам'яті даних. Адреса комірки перебуває в регістрі-покажчику. У якості регістра-покажчика можуть бути тільки регістри R0 і R1.

При прямій адресації адреса одного операнда перебуває в другому байті команди (рис. 2.11)

Рис. 2.11. Структура двобайтової команди

Приклад: MOV A, 20H; переслати вміст комірки пам'яті з адресою 20H в акумулятор.

Другий байт команди вказує пряма адреса осередка ОЗП адресного простору програмної моделі МК51. Приклад: MOV A, 20H; переслати вміст комірки пам'яті з адресою 20H в акумулятор.

Цей спосіб адресації дозволяє адресуватися до 128 коміркам оперативної пам'яті й регістрам керування.

При безпосередній адресації числа (константи) перебувають у другому байті команди (рис. 2.12)

Рис. 2.12. Структура двобайтової команди при безпосередній адресації

Приклад: MOV A, #20H; переслати константу 20H в акумулятор.

У трибайтових командах перший байт завжди КОП, а другий і третій байт мають різне призначення залежно від типу команди:

а) КОП	adr 16H	adr 16L
б) КОП	# data 16H	# data 16L
в) КОП	bit	rel
г) КОП	#d 8	rel
д) КОП	ads 8	add 8
е) КОП	adr	rel
ж) КОП	adr	#d 8

Рис. 2.13. Структура трибайтових команд

При завантаженні регістру DPTR командою MOV DPTR, #F0F0H; другий і третій байт команди – це константа (рис. 2.13, б). При пересиланні

безпосереднього байта за прямою адресою (рис. 2.13, ж) один байт указує адресу пересилання, а інший байт – безпосередній операнд #d 8.

При позначенні в третьому байті команди символу «rel» (рис. 2.13, в, г, е) (relative offset) позначає, що це зсув адреси. Як правило, зсув позначає підсумовування до вмісту лічильника команд (PC), зазначеного в команді зсуву, (PC)+ rel) і пересилання цієї суми знову в лічильник команд

PC←(PC)+ rel.

У трибайтовій команді (рис. 2.13, д), у другому й третьому байтах зазначені адреси джерела (add) і адреса приймача (ads) інформації.

Для адресації до бітів використовується пряма 8-бітна адреса. Непряма адресація до бітів неможлива.

Особливістю асемблера МК51 є широке використання для адресації символічних імен регістрів спеціальних функцій, портів і їх окремих бітів. Наприклад, MOV P0, #10H; пересилає в порт 0 константу #10H. З іншої сторони порт 0 має й пряму адресу 80H (рис. 2.8). MOV SBUFF, A пересилає вміст акумулятора в буферний регістр послідовного обміну. Акумулятор має свою пряму адресу (E0H) і буферний регістр послідовного обміну (99H). Символічне позначення «bit» у команді (рис. 2.13) указує на адресу біта у внутрішній пам'яті даних.

2.4. Система команд МК51

У системі команд МК51 можна виділити п'ять груп: арифметичні, логічні, команди пересилання даних, команди роботи з бітами й команди розгалужування або передачі керування.

2.4.1. Арифметичні команди

Арифметичні команди (табл. 2.3) забезпечують виконання операцій додавання, віднімання (у т.ч. з урахуванням переносу) над цілочисельними 8-розрядними операндами, інкрементування й декрементування вмісту регістрів і комірок внутрішньої пам'яті даних, множення й розподіл 8-розрядних цілочисельних операндів.

Таблиця 2.3

Арифметичні команди мікроконтролерів MCS-51

Мнемоніка	Опис	Байт	Тактів	Дія
ADD A, R_n	Додавання регістру з акумулятором	1	12	$(A) \leftarrow (A)+(R_n)$
ADD A, $@R_i$	Додавання комірок внутрішньої пам'яті даних, що адресується непрямо, з акумулятором	1	12	$(A) \leftarrow (A)+((R_i))$
ADD A, R_n	Додавання регістру з акумулятором з урахуванням переносу	1	12	$(A) \leftarrow (A)+(C)+(R_n)$
ADDC A, $@R_i$	Додавання комірок внутрішньої пам'яті даних, що адресується непрямо, з акумулятора з урахуванням переносу	1	12	$(A) \leftarrow (A)+(C)+((R_i))$

Закінчення табл. 2.3

SUBB A, R_n	Віднімання регістру з акумулятора з урахуванням позики	1	12	$(A) \leftarrow (A) - (C) - (R_n)$
SUBB A, @R_i	Віднімання непрямо адресованої комірки внутрішньої пам'яті даних з акумулятора з урахуванням переносу	1	12	$(A) \leftarrow (A) - (C) - ((R_i))$
INC A	Інкремент акумулятора	1	12	$(A) \leftarrow (A) + 1$
INC R_n	Інкремент регістру	1	12	$(R_n) \leftarrow (R_n) + 1$
INC @R_i	Інкремент непрямо адресованої комірки внутрішньої пам'яті даних	1	12	$((R_i)) \leftarrow ((R_i)) + 1$
DEC A	Декремент акумулятора	1	12	$(A) \leftarrow (A) - 1$
DEC R_n	Декремент регістру	1	12	$(R_n) \leftarrow (R_n) - 1$
DEC @R_i	Декремент непрямо адресованої комірки внутрішньої пам'яті даних	1	12	$((R_i)) \leftarrow ((R_i)) - 1$
INC DPTR	Інкремент покажчика даних	1	24	$(DPTR) \leftarrow (DPTR) + 1$
MUL AB	Множення A на B	1	48	$(A)_{7-0} \leftarrow (A) \times (B)$ $(B)_{15-8}$
DIV AB	Ділення A на B	1	48	$(A)_{15-8} \leftarrow (A)/(B)$ $(B)_{7-0}$
DA A	Двійково-десяткова корекція	1	12	Вміст акумулятора у BCD $IF[[(A_{3-0}) > 9] \vee [(AC) = 1]]$ THEN $(A)_{3-0} \leftarrow (A_{3-0}) + 6$ AND $IF[[(A_{7-4}) > 9] \vee [(C) = 1]]$ THEN $(A)_{7-4} \leftarrow (A_{7-4}) + 6$
ADD A, direct	Додавання комірки внутрішньої пам'яті даних з акумулятором	2	12	$(A) \leftarrow (A) + (direct)$
ADD A, #data	Додавання безпосереднього байта даних з акумулятором	2	12	$(A) \leftarrow (A) + \#data$
ADDC A, direct	Додавання комірки внутрішньої пам'яті даних з акумулятором з урахуванням переносу	2	12	$(A) \leftarrow (A) + (C) + (direct)$
ADDC A, #data	Додавання безпосереднього байта даних з акумулятором з урахуванням переносу	2	12	$(A) \leftarrow (A) + (C) + \#data$
SUBB A, direct	Віднімання комірки внутрішньої пам'яті даних з акумулятора з обліком позики	2	12	$(A) \leftarrow (A) - (C) - (direct)$
SUBB A, #data	Віднімання безпосереднього байта даних з акумулятора з урахуванням переносу	2	12	$(A) \leftarrow (A) - (C) - \#data$
INC direct	Інкремент комірки внутрішньої пам'яті даних	2	12	$(direct) \leftarrow (direct) + 1$
DEC direct	Декремент комірки внутрішньої пам'яті даних	2	12	$(direct) \leftarrow (direct) - 1$

У таблиці наведено мнемонічне зображення команд, види операцій над операндами, кількість байтів у команді, кількість тактів і умовне зображення операції, здійснюваної мікропроцесором над операндами.

Наприклад, ADD A, R3; додавання вмісту регістру R3 поточного банку з акумулятором, результат міститься в акумуляторі $< A > \leftarrow < A > + < R3 >$.

В арифметичних командах використовуються всі види адресації, розглянуті вище.

Як можна побачити з таблиці, арифметичні команди не містять трибайтових команд. Однобайтові команди використовують регістрову й непряморегістрову адресацію, результат операції завжди розміщується в акумуляторі

ADD A, R_n; ADD A, $@R_i$; ADDC A, R_n;

SUBB A, R_n; SUBB A $@R_i$; INC R_n; DCC $@R_i$; INC $@R_i$.

Команди інкремента й декремента також усі одно байтові. Винятком є команди інкремента й декремента прямоадресованих комірок внутрішньої пам'яті даних, в яких (INC direct) у другому байті команди завжди знаходиться адреса комірок внутрішньої пам'яті даних. Слід зазначити, що якщо адресація до комірки внутрішньої пам'яті даних використовує непряму регістрову адресацію, наприклад INC @ R0, то команда виходить однобайтовою.

Особливістю системи команд МК51 є наявність команд множення (MUL) і ділення (DIV) цілочисельних 8-розрядних операндів. Команди множення й ділення використовують регістр В. Уміст акумулятора множиться на вміст регістру В и ділиться на вміст регістру В. 16-розрядний результат зберігається в регістрах А и В.

2.4.2. Логічні команди

Логічні команди (табл. 2.4) виконують реалізацію логічних операцій «І» (AND); «АБО» (ORL); «що виключає АБО» (сума за модулем два) (XRL) над операндами, зрушення над умістом акумулятора вправо й уліво, очищення акумулятора й перестановку його тетрад.

Таблиця 2.4

Логічні команди мікроконтролерів MCS-51

Мнемоніка	Опис	Байт	Тактів	Дія
ANL A, R_n	AND регістру й акумулятора	1	12	$(A) \leftarrow (A) \wedge (R_n)$
ANL A, $@R_i$	AND непрямо адресованої комірки внутрішньої пам'яті даних і акумулятора	1	12	$(A) \leftarrow (A) \wedge ((R_i))$
ORL A, R_n	OR регістру й акумулятора	1	12	$(A) \leftarrow (A) \vee (R_n)$
ORL A, $@R_i$	OR непрямо адресованої комірки внутрішньої пам'яті даних і акумулятора	1	12	$(A) \leftarrow (A) \vee ((R_i))$
XRL A, R_n	XOR регістру й акумулятора	1	12	$(A) \leftarrow (A) \forall (R_n)$

Закінчення табл. 2.4

XRL A, @R_i	XOR непрямо адресованої комірки внутрішньої пам'яті даних і акумулятора	1	12	$(A) \leftarrow (A) \forall ((R_i))$
CLR A	Очищення акумулятора	1	12	$(A) \leftarrow 0$
CPL A	Інверсія акумулятора	1	12	$(A) \leftarrow \neg (A)$
RL A	Зрушення акумулятора вліво	1	12	$(A_n + 1) \leftarrow (A_n)\ n = 0-6$ $(A0) \leftarrow (A7)$
RLC A	Зрушення акумулятора вліво через перенос	1	12	$(A_n + 1) \leftarrow (A_n)\ n = 0-6$ $(A0) \leftarrow (C)\ \ (C) \leftarrow (A7)$
RR A	Зрушення акумулятора вправо	1	12	$(A_n) \leftarrow (A_n + 1)\ n = 0-6$ $(A7) \leftarrow (A0)$
RRC A	Зрушення акумулятора вправо через перенос	1	12	$(A_n) \leftarrow (A_n + 1)\ n = 0-6$ $(A7) \leftarrow (C)\ \ (C) \leftarrow (A0)$
SWAP A	Перестановка тетрад акумулятора	1	12	$(A_{3-0}) \leftrightarrow (A_{7-4})$
ANL A, direct	AND комірки внутрішньої пам'яті даних і акумулятора	2	12	$(A) \leftarrow (A) \wedge (direct)$
ANL A, #data	AND безпосереднього байта даних і акумулятора	2	12	$(A) \leftarrow (A) \wedge \#data$
ANL direct, A	AND акумулятора й комірки внутрішньої пам'яті даних	2	12	$(direct) \leftarrow (direct) \wedge (A)$
ORL A, direct	OR комірки внутрішньої пам'яті даних і акумулятора	2	12	$(A) \leftarrow (A) \vee (direct)$
ORL A, #data	OR безпосереднього байта даних і акумулятора	2	12	$(A) \leftarrow (A) \vee \#data$
ORL direct,A	OR акумулятора й комірки внутрішньої пам'яті даних	2	12	$(direct) \leftarrow (direct) \vee (A)$
XRL A, direct	XOR комірки внутрішньої пам'яті даних і акумулятора	2	12	$(A) \leftarrow (A) \forall (direct)$
XRL A, #data	XOR безпосереднього байта даних і акумулятора	2	12	$(A) \leftarrow (A) \forall \#data$
XRL direct, A	XOR акумулятора й комірки внутрішньої пам'яті даних	2	12	$(direct) \leftarrow (direct) \forall (A)$
ANL direct, #data	AND безпосереднього байта даних і комірки внутрішньої пам'яті даних	3	24	$(direct) \leftarrow (direct) \wedge \#data$
ORL direct, #data	OR безпосереднього байта даних і комірки внутрішньої пам'яті даних	3	24	$(direct) \leftarrow (direct) \vee \#data$

Слід зазначити, що всі команди, що використовують регістрову й реєстрову-регістрову-непряморегістрову адресації, також однобайтові.

Якщо логічні операції виконуються над операндами, розміщеними у внутрішній пам'яті, й операндом, розміщеним у команді, то команди в цьому

випадку мають три байти й виконуються за 24 такти. Це команди ANL direct, #data, ORL direct, #data, XRL direct, #data.

2.4.3. Команди пересилання

Велику групу складають команди пересилання даних (табл. 2.5). Команди пересилання зручно розглянути на програмній моделі МК51 (рис. 2.14).

Таблиця 2.5

Команди пересилання даних мікроконтролерів MCS-51

Мнемоніка	Опис	Байт	Тактів	Дія
MOV A, R_n	Пересилання з регістру в акумулятор	1	12	$(A) \leftarrow (R_n)$
MOV A, @R_i	Пересилання з непрямо адресованої комірки внутрішньої пам’яті даних в акумулятор	1	12	$(direct) \leftarrow ((R_i))$
MOV R_n, A	Пересилання з акумулятора в регістр	1	12	$(R_n) \leftarrow (A)$
MOV @R_i, A	Пересилання з акумулятора в непрямо адресовану комірку внутрішньої пам’яті даних	1	12	$((R_i)) \leftarrow (A)$
MOVC A, @A+ DPTR	Пересилання байта коду, пов’язаного з DPTR, в акумулятор	1	24	$(A) \leftarrow ((A) + DPTR))$
MOVC A, @A+PC	Пересилання байта коду, пов'язаного із PC, в акумулятор	1	24	$(PC) \leftarrow (PC) + 1\ (A) \leftarrow ((A) + (PC))$
MOVX A, @R_i	Пересилання байта із зовнішньої пам'яті даних (8-розр. адреса) в акумулятор	1	24	$(A) \leftarrow ((R_i))$
MOVX A, @ DPTR	Пересилання байта із зовнішньої пам'яті даних (16-розр. адреса) в акумулятор	1	24	$(A) \leftarrow ((DPTR))$
MOVX @R_i, A	Пересилання з акумулятора в комірку зовнішньої пам'яті даних (8-розр. адреса)	1	24	$((R_i)) \leftarrow (A)$
MOVX @ DPTR, A	Пересилання з акумулятора в комірку зовнішньої пам'яті даних (16-розр. адреса)	1	24	$(DPTR) \leftarrow (A)$
XCH A, R_n	Обмін між регістром і акумулятором	1	12	$(A) \leftrightarrow ((R_n))$
XCH A, @R_i	Обмін між непрямо адресованою коміркою внутрішньої пам'яті даних і акумулятором	1	12	$(A) \leftrightarrow ((R_i))$
XCHD A, @R_i	Обмін молодшими тетрадами між непрямо адресованою коміркою внутрішньої пам'яті даних і акумулятором	1	12	$(A_{3-0}) \leftrightarrow ((R_{i_{3-0}}))$

Закінчення табл. 2.5

MOV A, direct	Пересилання із комірки внутрішньої пам'яті даних в акумулятор	2	12	$(A) \leftarrow (direct)$
MOV A, #data	Пересилання безпосереднього байта даних в акумулятор	2	12	$(A) \leftarrow \#data$
MOV R_n, direct	Пересилання із комірки внутрішньої пам'яті даних у регістр	2	24	$(R_n) \leftarrow (direct)$
MOV R_n, #data	Пересилання безпосереднього байта даних у регістр	2	12	$(R_n) \leftarrow \#data$
MOV direct, A	Пересилання з акумулятора в комірку внутрішньої пам'яті даних	2	12	$(direct) \leftarrow (A)$
MOV direct, R_n	Пересилання з регістру в комірку внутрішньої пам'яті даних	2	24	$(direct) \leftarrow (R_n)$
MOV direct, $@R_i$	Пересилання з непрямо адресованої комірки внутрішньої пам'яті даних у комірку внутрішньої пам'яті даних	2	24	$(direct) \leftarrow ((R_i))$
MOV $@R_i$, direct	Пересилання із комірки внутрішньої пам'яті даних у непрямо адресовану комірку внутрішньої пам'яті даних	2	24	$((R_i)) \leftarrow (direct)$
MOV $@R_i$, #data	Пересилання безпосереднього байта даних у непрямо адресовану комірку	2	12	$((R_i)) \leftarrow (data)$
PUSH direct	Завантаження комірки внутрішньої пам'яті даних у стек	2	24	$(SP) \leftarrow (SP)+1 \ ((SP)) \leftarrow (direct)$
POP direct	Вивантаження зі стека в комірку внутрішньої пам'яті даних	2	24	$(direct) \leftarrow ((SP)) \ (SP) \leftarrow (SP)-1$
XCH A, direct	Обмін між коміркою внутрішньої пам'яті даних і акумулятором	2	12	$(A) \leftrightarrow (direct)$
MOV direct, direct	Пересилання із комірки внутрішньої пам'яті даних у комірку внутрішньої пам'яті даних	3	24	$(direct) \leftarrow (direct)$
MOV direct, #data	Пересилання безпосереднього байта даних у комірку внутрішньої пам'яті даних	3	24	$(direct) \leftarrow \#data$
MOV DPTR, # 16	Завантаження 16-розрядної константи в покажчик даних	3	24	$(DPTR) \leftarrow \#data_{15-0}$ $DPH \leftarrow \#data_{15-8}$ $DPL \leftarrow \#data_{7-0}$

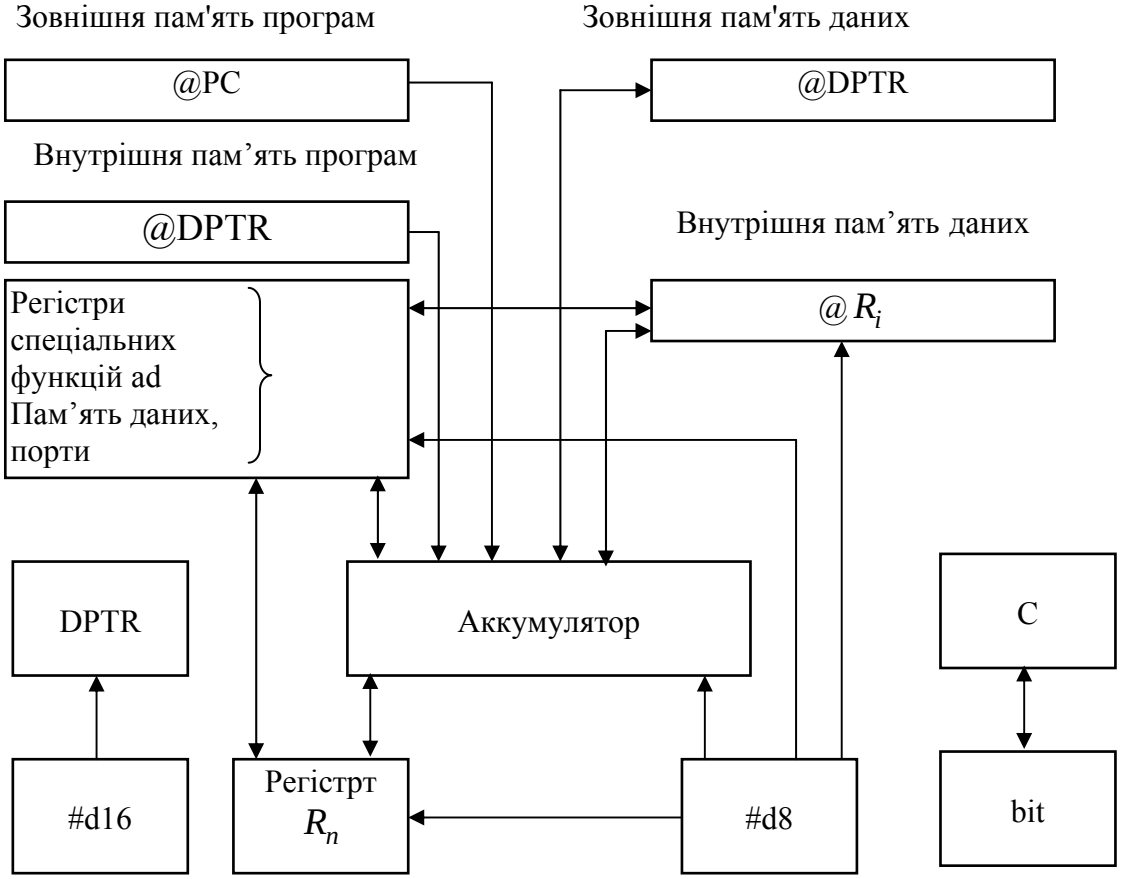


Рис. 2.14. Структура зв'язків програмної моделі МК51

2.4.4. Команди передачі керування

За командами передачі керування або командами переходів мікропроцесор здійснює безумовні або умовні розгалужування (переходи) програми, виклик підпрограм і повернення з них. Найпростішими є команди безумовного переходу, які ще називають командами-стрибками (jmp). Як правило, у команді зазначена адреса переходу. Переходи можна здійснювати у всьому 16-бітному адресному просторі пам'яті, при цьому використовується команда Ljmp 16 adr (табл. 2.6).

Таблиця 2.6

Команди передачі керування мікроконтролерів МК51

Мнемоніка	Опис	Байт	Тактів	Дія
RET	Повернення з підпрограми	1	24	$(PC_{15-8}) \leftarrow ((SP)),\ (SP) \leftarrow (SP)-1,\ (PC_{7-0}) \leftarrow ((SP))\ \ (SP) \leftarrow (SP)-1$
RETI	Повернення з підпрограми переривання	1	24	$(PC_{15-8}) \leftarrow ((SP)),\ (SP) \leftarrow (SP)-1, (PC_{7-0}) \leftarrow ((SP)),\ (SP) \leftarrow (SP)-1$
JMP @A+DPTR	Перехід відносно DPTR	1	24	$(PC) \leftarrow (A)+(DPTR)$
NOP	Ні операції	1	12	$(PC) \leftarrow (PC)+1$

Продовження табл. 2.6

ACALL addr11	Короткий перехід з поверненням	2	24	$(PC) \leftarrow (PC)+2,\ (SP) \leftarrow (SP)+1,\ ((SP)) \leftarrow (PC_{7-0}),$ $(SP) \leftarrow (SP)+1,\ ((SP)) \leftarrow (PC_{15-8}),\ (PC_{10-0}) \leftarrow pageaddress$
AJMP addr11	Короткий перехід без вороття	2	24	$(PC) \leftarrow (PC)+2,\ (PC_{10-0}) \leftarrow pageaddress$
SJMP rel	Короткий перехід з 8-розрядним зсувом	2	24	$(PC) \leftarrow (PC)+2,\ (PC) \leftarrow (PC)+rel$
JZ rel	Перехід, якщо акумулятор дорівнює нулю	2	24	$(PC) \leftarrow (PC)+2,\ IF(A)=0,\ THEN(PC) \leftarrow (PC)+rel$
JNZ rel	Перехід, якщо акумулятор не дорівнює нулю	2	24	$(PC) \leftarrow (PC)+2,\ IF(A) \neq 0,\ THEN(PC) \leftarrow (PC)+rel$
JC rel	Перехід, якщо прапор переносу встановлено	2	24	$(PC) \leftarrow (PC)+2, IF(C)=1, THEN(PC) \leftarrow (PC)+rel$
JNC rel	Перехід, якщо прапор переносу не встановлено	2	24	$(PC) \leftarrow (PC)+2, IF(C)=0, THEN(PC) \leftarrow (PC)+rel$
JB bit, rel	Перехід, якщо біт установлено	3	24	$(PC) \leftarrow (PC)+3, IF(bit)=1, THEN(PC) \leftarrow (PC)+rel$
JNB bit, rel	Перехід, якщо біт не встановлено	3	24	$(PC) \leftarrow (PC)+3, IF(bit)=0, THEN(PC) \leftarrow (PC)+rel$
JBC bit, rel	Перехід, якщо біт установлено, і очищення цього біта	3	24	$(PC) \leftarrow (PC)+3, IF(bit)=1, THEN(bit) \leftarrow 0$ $(PC) \leftarrow (PC)+rel$
DJNZ R_n, rel	Декремент регістру й перехід, якщо він не дорівнює нулю	2	24	$(PC) \leftarrow (PC)+2,\ (R_n) \leftarrow (R_n)-1$ $IF(R_n)>0 or (R_n)<0,\ THEN(PC) \leftarrow (PC)+rel$
LCALL addr16	Довгий перехід з поверненням	3	24	$(PC) \leftarrow (PC)+3,\ (SP) \leftarrow (SP)+1,\ ((SP)) \leftarrow (PC_{7-0}),$ $(SP) \leftarrow (SP)+1,\ ((SP)) \leftarrow (PC_{15-8}),\ (PC) \leftarrow addr_{15-0}$
LJMP addr16	Довгий перехід без вороття	3	24	$(PC) \leftarrow addr_{15-0}$
CJNE A, direct, rel	Перехід, якщо комірка внутрішньої пам'яті даних не є ідентичною акумулятору	3	24	$(PC) \leftarrow (PC)+3,\ IF(A) <> (direct)$ $THEN(PC) \leftarrow (PC)+relativeoffset$ $IF(A)<(direct) THEN(C) \leftarrow 1,\ ELSE(C) \leftarrow 0$
CJNE A, #data, rel	Перехід, якщо безпосередній байт даних не дорівнює акумулятору	3	24	$(PC) \leftarrow (PC)+3,\ IF(A) <> (data)$ $THEN(PC) \leftarrow (PC)+relativeoffset$ $IF(A)<(data) THEN(C) \leftarrow 1,\ ELSE(C) \leftarrow 0$
CJNE R_n, #data, rel	Перехід, якщо безпосередній байт даних не дорівнює регістру	3	24	$(PC) \leftarrow (PC)+3,\ IF((R_i)) <> (data)$ $THEN(PC) \leftarrow (PC)+relativeoffset$ $IF((R_i))<(data) THEN(C) \leftarrow 1,\ ELSE(C) \leftarrow 0$

Закінчення табл. 2.6

CJNE @R_i, #data, rel	Перехід, якщо безпосередній байт даних не дорівнює непрямо адресованій комірці внутрішньої пам'яті даних	3	24	$(PC) \leftarrow (PC)+3,\ IF((R_i)) <> (data)$ $THEN(PC) \leftarrow (PC)+relativeoffset$ $IF((R_i)) < (data) THEN(C) \leftarrow 1,\ ELSE(C) \leftarrow 0$
DJNZ direct, rel	Декремент комірки внутрішньої пам'яті даних і перехід, якщо вона не дорівнює нулю	3	24	$(PC) \leftarrow (PC)+2,\ (direct) \leftarrow (direct)-1$ $IF(direct) > 0 or (direct) < 0,\ THEN(PC) \leftarrow (PC)+rel$

Якщо переходи виконуються в межах однієї сторінки розміром 2048 байт, то використовують команди абсолютного переходу Ajmp adr 11 (буква А початкова буква слова Absolute), при цьому молодші 11 біт умісту лічильника команд (РС) змінюються 11-бітною адресою, зазначеною у самій команді. Переходи можна здійснювати щодо поточної адреси. При цьому використовують команду Sjmp (буква S початкова у слові Short – короткий). Адресний зсув переходу щодо адреси наступної команди становить – 128… + 127 байт. Адреса зсуву вказується в команді Sjmp rel (8); (табл. 2.6). Переходи можна здійснювати за непрямою адресою jmp@A+DPTR щодо вмісту вказівного регістру DPTR. При цьому в лічильник команд завантажується сума вмісту комірки пам'яті, зазначеної акумулятором, і вмісту регістру DPTR $((A)) + DPTR \rightarrow PC$. При цьому переході програма сама обчислює адресу переходу, яка невідома при написанні вихідного тексту програми. Система команд МК51 дозволяє здійснювати й умовні переходи. Наприклад, якщо вміст акумулятора (JZ) дорівнює нулю, уміст акумулятора не дорівнює нулю (JNZ).

Перенос дорівнює одиниці (JC), перенос дорівнює нулю (JNC), адресований біт дорівнює одиниці JB, адресований біт дорівнює нулю (JNB).

Усі команди умовних переходів використовують відносний метод адресації. При цьому відносна адреса переходу втримується в другому байті команди JL rel (8). Команди DJNZ Rn, rel; і DJNZ adr, rel; (табл. 2.6) використовуються для організації програмних циклів. При цьому в якості лічильника використовуються як регістри поточного банку, так і прямоадресовані комірки пам'яті.

Команди порівняння й перехід, якщо не дорівнює CJNEA, adr, rel; CJNEA, #ds, rel; CJNE Rn, #d8, rel; CJNE @Ri, # d8, rel також використовують відносні переходи й використовуються в тих випадках, коли в системі необхідно зафіксувати які-небудь події, наприклад, CJNEA, P2, 20H. Завдяки цій команді порівнюються вміст акумулятора й стан порту Р2. Якщо вони не дорівнюються, то здійснюється перехід зі зсувом 20Н, якщо дорівнюються, то виконується наступна за цією команда програми. Функції розгалужування програми з поверненням до адреси перерваної програми виконують команди

ACALL adr 11, виклик підпрограми в адресному просторі 2 Кбіт і LCALL adr 16, виклик підпрограми у всьому адресному просторі мікроконтролера. При виконанні цих команд мікропроцесор запам'ятовує адресу перерваної програми в стеку $PC+2 \to SP$ й здійснює повернення до перерваної адреси програми командою RET (SP$\to$ PC). За командою RET вміст вершини стека (два байти) пересилається в лічильник команд (SP)$\to$PC. Команда RETI виконує те саме й додатково скидає сигнал запиту переривання.

Завершує систему команд МК51 порожня операція NOP. Завдяки цій команді мікропроцесор нарощує вміст лічильника команд на 1 (PC+1) і не виконує ніяких операцій. При частоті тактування 12 Мгц, команда NOP виконується за 1 мкс.

2.4.5. Команди роботи з бітами

Команди роботи з бітами (табл. 2.7) є відмінною рисою МК51. У деяких описах МК51 говориться про наявність булевого процесора.

Таблиця 2.7

Команди робота з бітами мікроконтролерів MCS-51

Мнемоніка	Опис	Байт	Тактів	Дія
CLR C	Очищення прапора переносу	1	12	$(C) \leftarrow 0$
SETB C	Установка прапора переносу	1	12	$(C) \leftarrow 1$
CPL C	Інверсія прапора переносу	1	12	$(C) \leftarrow \neg(C)$
CLR bit	Очищення біта	2	12	$(bit) \leftarrow 0$
SETB bit	Установка біта	2	12	$(bit) \leftarrow 1$
CPL bit	Інверсія біта	2	12	$(bit) \leftarrow \neg(bit)$
ANL C, bit	AND біта й прапора переносу	2	24	$(C) \leftarrow (C) \wedge (bit)$
ANL C, /bit	AND інверсії біта й прапора	2	24	$(C) \leftarrow (C) \wedge \neg(bit)$
ORL C, bit	OR біта й прапора переносу	2	24	$(C) \leftarrow (C) \vee (bit)$
ORL C, /bit	OR інверсії біта й прапора переносу	2	24	$(C) \leftarrow (C) \vee \neg(bit)$
MOV C, bit	Пересилання біта в прапор переносу	2	12	$(C) \leftarrow (bit)$
MOV bit, З	Пересилання прапора переносу в біт	2	24	$(bit) \leftarrow (C)$

Ці команди встановлюють в 1 (SETB, bit) або скидають у нуль (CLR, bit) прямоадресований біт внутрішньої пам'яті даних, інвертують його значення CPL, bit, виконують операції – логічне «І» і «АБО» (AND і OR) над прапором переносу й прямоадресованим бітом (ANL C, bit), здійснюють пересилання між прапором переносу С і прямоадресованим бітом (MOVC, bit).

Контрольні запитання

1. Скільки банків даних містить мікроконтролер МК51?
2. Де розташована стекова пам’ять в МК51?
3. Яку кількість розрядів містить покажчик стекової пам’яті в МК51?
4. Яку кількість регістрів містить банк 0 в МК51?
5. Яка початкова адреса стекової пам’яті в МК51?
6. Який мінімальний об’єм пам’яті даних в МК51?
7. Який регістр МК51, крім акумулятора, використовується в командах множення і ділення MUL і DIV?
8. Яка команда пересилає інформацію із стека до МК51?
9. Яка команда пересилає інформацію з банка даних до акумулятора?
10. Яка команда пересилає константу до регістру, що знаходиться в банку даних?
11. Яка команда пересилає інформацію з комірки у комірку, що знаходяться в пам’яті даних?
12. Яка команда пересилає інформацію з зовнішньої пам’яті даних до МК51?
13. Яка команда виконує десяткову корекцію акумулятора?
14. Яка команда збільшує вміст акумулятора?
15. Яка команда виконує циклічний зсув вмісту акумулятора ліворуч?
16. Яка команда виконує скидання вмісту акумулятора?

3. ПОРТИ ВВОДУ/ВИВОДУ ІНФОРМАЦІЇ

Мікроконтролер серії MCS-51 має чотири порти вводу/виводу інформації. Вони призначені для двонапрямленого обміну інформацією із зовнішніми пристроями, такими як кнопки, датчики, виконавчі пристрої, індикатори і т.п., а також із зовнішньою пам'яттю програм і даних. Крім цього, порти використовуються під час програмування внутрішньої пам'яті програм контролера.

Спрощені моделі розрядів портів показані на рис. 3.1, 3.2. Базовою є модель порту 1. Вона містить керований тригер-засувку, виконаний у вигляді D-Тригера, вхідний буфер В1, буфер В2 читання тригера-засувки й вихідний драйвер, виконаний на польовому транзисторі з ізольованим затвором і навантажений на резистор. Схему порту 0 доповнено мультиплексором MS, який дозволяє підключати до виходів порту молодший байт адресної шини або шину даних процесора. Крім цього, для підвищення швидкодії порту при роботі із зовнішньою пам'яттю вихідний ключ виконаний за двотранзисторною схемою. Схему порту 2 доповнено мультиплексором MS, який дозволяє підключати до виходів порту старший байт адресної шини процесора при роботі із зовнішньою пам'яттю. Схему порту 3 доповнено буферами підключення периферійних пристроїв. Адресація до портів проводиться так само, як до комірок пам'яті. Фізичні адреси портів вводу/виводу:

P0-80H, при бітовій адресації 80H-87H;

P1-90H, при бітовій адресації 90H-97H;

P 2-A0H, при бітовій адресації A0 H-A7H;

P 3-B0H, при бітовій адресації B0 H-B7H.

При обміні інформацією із зовнішньою пам'яттю даних і програм інформація на портах встановлюється процесором автоматично. При цьому через P0 проводиться видача молодшого байта адреси й приймання або видача даних, а на P2 видається старший байт адреси. У тригері-засувці порту 0 при обміні даними записуються всі одиниці.

У режимі програмування порт 0 використовується в якості шини даних, через порт 1 подається молодший байт адреси, через порт 2 – старші розряди адреси й керуючі сигнали. Розряди порту 3, крім функцій вводу/виводу, мають альтернативні функції, наведені в табл. 3.1.

Навантажувальна здатність порту P0 – два TTL входу, портів P1, P2, P3 – один TTL вхід. Для приєднання потужнішого навантаження необхідно використовувати або буфери (рис. 1.32, розділ 1), або транзисторні підсилювачі (рис. 1.33, розділ 1).

Операцію приймання сигналів від зовнішнього пристрою контролером прийнято називати вводом. Для настроювання порту або окремих бітів порту на ввід у тригери-засувки відповідних розрядів необхідно занести 1.

Прочитати стан портів можна командою

```
MOV A, P0 ; занесення в акумулятор значення порту 0.
```

Крім цього, з розрядами портів можливі бітові операції:

```
MOV C, P1.1    ; пересилання розряду порту P1.1 у перенос.
```

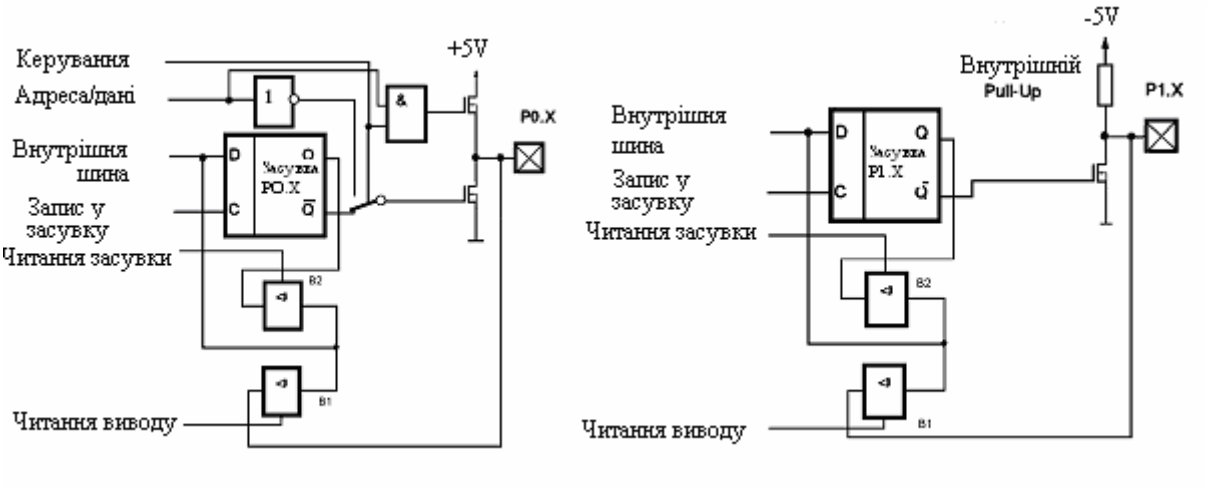


Рис. 3.1. Функціональні схеми каналів портів 0 і 1

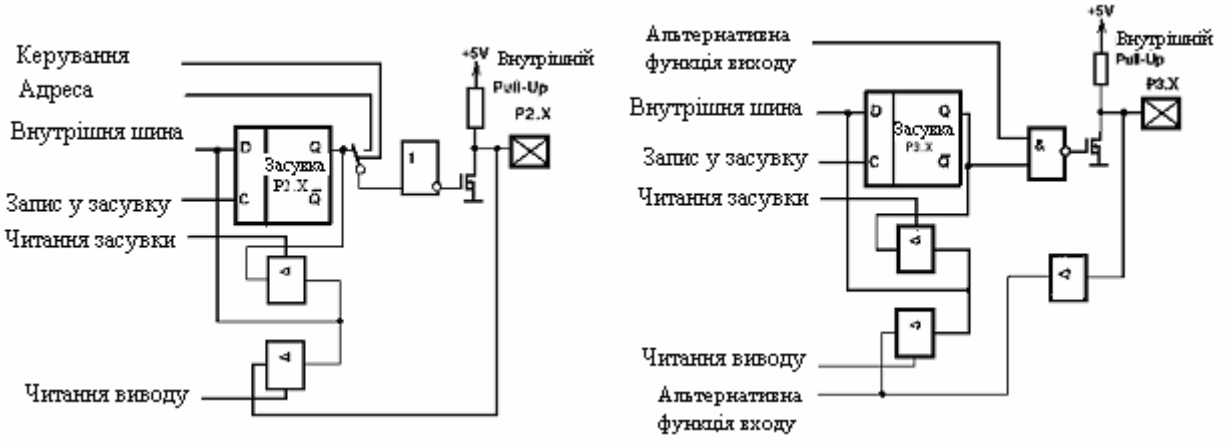


Рис. 3.2. Функціональні схеми каналів портів 2 і 3

Таблиця 3.1

Альтернативні функції порту 3

Сигнал	Розряд порту	Призначення
$\overline{RD}$	P3.7	Читання. Активний сигнал низького рівня формується апаратно при зверненні до ЗПД.
$\overline{WR}$	P3.6	Читання. Активний сигнал низького рівня формується апаратно при зверненні до ЗПД.
T1	P3.5	Вхід таймера лічильника 1 або тест-вхід
T0	P3.4	Вхід таймера лічильника 0 або тест-вхід
$\overline{INT1}$	P3.3	Вхід запиту переривання 1. Сприймається сигнал низького рівня або зріз.
$\overline{INT0}$	P3.2	Вхід запиту переривання 1. Сприймається сигнал низького рівня або зріз.
TxD	P3.1	Вихід передавача послідовного порту в режимі УАПП. Вихід синхронізації в режимі регістру, що зсуває.
RxD	P3.0	Вхід приймача послідовного порту в режимі УАПП. Ввід-вивід даних у режимі регістру, що зсуває.

Команди передачі керування дозволяють за значенням окремих бітів порту змінювати послідовність виконання програми, наприклад:

JB P2.0, M1 ; перехід на мітку M1, якщо розряд P2.0 дорівнює 1

JNB P0.0, M2 ; перехід на мітку M2, якщо розряд P0.0 дорівнює 0.

Операцію передачі сигналів від мікроконтролера до зовнішнього пристрою називають виводом. У цьому випадку на розрядах портів мікроконтролер установлює потрібні логічні рівні сигналів і в такий спосіб проводить керування зовнішніми пристроями. При виконанні операції виводу в порт нове значення записується в засувку у фазі S6P2 останнього машинного циклу команди. Однак новий уміст засувки виводиться безпосередньо на вихідний контакт тільки у фазі S1P1 наступного машинного циклу. Вивід інформації в порт можна виконати командами пересилань, наприклад,

MOV P1, A ; пересилання в порт P1 умісту акумулятора

Якщо порт є одночасно операндом і місцем призначення результату, процесор автоматично реалізує спеціальний режим, який називається «читання–модифікація–запис». Цей режим обігу припускає ввід сигналів не із зовнішніх виводів порту, а з його регістру-засувки, що дозволяє виключити неправильне зчитування раніше виведеної інформації, оскільки передбачається модифікування вмісту тригера-засувки, а не вхідної інформації. Цей режим звертання до портів реалізовано у наступних командах:

ANL – логічне І, наприклад ANL P1, A;

ORL – логічне АБО, наприклад ORL P2, A;

XRL – виключає АБО, наприклад XRL P3, A;

JBC – перехід, якщо в адресованому біті одиниця, і наступне скидання біта, наприклад JBC P1.1,LABEL;

CPL – інверсія біта, наприклад CPL P3.3;

INC – інкремент порту, наприклад INC P2;

DEC – декремент порту, наприклад DEC P2;

DJNZ – декремент порту й перехід, якщо його вміст не дорівнює нулю, наприклад DJNZ P3,LABEL;

MOV PX.Y,C – передача біта переносу в біт Y порту X;

SETB PX.Y – установка біта Y порту X;

CLR PX.Y – скидання біта Y порту X.

Приклади роботи з портами.

Порти в мікроконтролері призначені для вводу вхідних сигналів і реалізації функцій дискретного керування. Фактично, завдання дискретного керування об'єктами з використанням програмувальних контролерів полягає у вводі вхідних сигналів, обчисленні логічної функції, що зв’язує вхідні й вихідні сигнали, й виводу результату обчислення на виходи контролера. У першій главі розглянуто завдання дискретного керування, яке можна записати у вигляді комбінаційних і послідовнісних автоматів. Задання умови для комбінаційного автомата можна зобразити у вигляді таблиці істинності. Розглянемо завдання побудови комбінаційного автомата на прикладі розробки дешифратора для

семисегментного індикатору. Схему підключення семисегментного індикатору наведено на рис. 3.3. Вхідні сигнали надходять на розряди порту P1.0-P1.3, керування індикатором з порту 2 здійснюється через буфер-підсилювач.

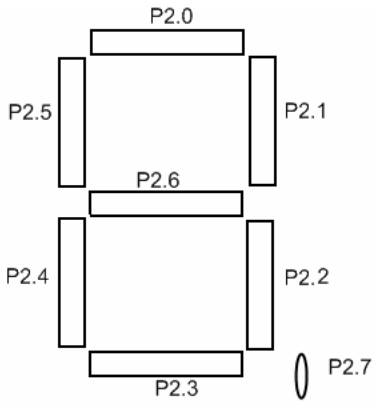


Рис. 3.3. Підключення семисегментного індикатора до мікроконтролера

Розв'язок: Запишемо таблицю істинності для дешифратора.

Таблиця 3.2

Істинності для дешифратора

Входи				Виходи								Символ
1	2	3	4	5	6	7	8	9	10	11	12	13
P1.3	P1.2	P1.1	P1.0	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	
0	0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	0	0	0	0	1	1	0	1
0	0	1	0	0	1	0	1	1	0	1	1	2
0	0	1	1	0	1	0	0	1	1	1	1	3
0	1	0	0	0	1	1	0	0	1	1	0	4
0	1	0	1	0	1	1	0	1	1	0	1	5
0	1	1	0	0	1	1	1	1	1	0	1	6
0	1	1	1	0	0	0	0	0	1	1	1	7
1	0	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	0	1	1	0	1	1	1	1	9
1	0	1	0	0	1	1	1	0	1	1	1	A
1	0	1	1	0	1	1	1	1	1	0	0	B
1	1	0	0	0	0	1	1	1	0	0	1	C
1	1	0	1	0	1	0	1	1	1	1	0	D
1	1	1	0	0	1	1	1	1	0	0	1	E
1	1	1	1	0	1	1	1	0	0	0	1	F

Сформуємо в пам'яті програм контролера масив значень байта виводу згідно до табл. 3.2 (мітка BEGDIM), які потрібно вивести на індикатор для формування шістнадцяткових цифр від 0 до F. Текст підпрограми, яка реалізує дешифратор, наведено нижче.

```
DESH:   MOV DPTR,#BEGDIM; заносимо в DPTR адресу першого
                        ; елемента масиву
        MOV A,P1        ; уводимо вхідні сигнали
```

```
        ANL A,#0FH          ; маскуємо старші розряди вхідних
                            ; сигналів, одержуємо зсув в таблиці для
                            ; даної комбінації
        MOVC A,@A+DPTR; заносимо в акумулятор значення
                             ; елемента масиву, що відповідає
                             ; вхідному коду
        MOV P2,A            ; виводимо значення в порт
        RET                 ; виходимо з підпрограми
BEGDIM: DB 3FH, 06H, 5BH, 4FH, ; 0, 1, 2, 3
        DB 66H, 6DH, 7DH, 07H, ; 4, 5, 6, 7
        DB 7FH, 6FH, 77H, 7CH, ; 8, 9, A, B
        DB 39H, 5EH, 79H,71H  ; C, D, E, F
```

У даній програмі вхідне значення коду, утвореного вхідними сигналами, перетвориться в зсув для масиву вихідних значень і виводиться у вихідний порт.

Такий спосіб реалізації комбінаційного автомата має наступні обмеження: вхідні сигнали повинні бути підключені на один порт і підряд (інакше прийдеться вирішувати завдання зведення різних входів в одне вхідне слово); при досить великій кількості входів (7 і більш) і неповному використанні вихідних комбінацій нераціонально використовується пам'ять програм. Тому пропонується інший спосіб реалізації, заснований на програмній реалізації логічних кон’юнктивно диз'юнктивних рівнянь.

Як і в попередньому випадку, функціонування автомата задається у вигляді таблиці істинності, наприклад:

Таблиця 3.3

Стан бітів портів

Входи				Вихід
P1.2	P1.4	P2.1	P2.3	P1.7
0	1	1	0	1
1	0	1	0	1
1	1	0	0	1

Запишемо логічне рівняння для виходу згідно з наведеною таблицею.

P1.7=!P1.2 & P1.4 & P2.1 & P2.3 +P1.2 & !P1.4 & P2.1 & !P2.3+P1.2 & !P1.4 & P2.1 & !P2.3

Схеми алгоритму програмної реалізації логічних функцій «І» та «АБО» наведені на рис. 3.5 та 3.6.

Програмна реалізація для цієї функції буде наступна:

```
        JB P1.2,M1
        JNB P1.4,M1
        JNB P2.1,M1
        JNB P2.3,M1
        SETB P1.7
        LJMP EXIT
M1:                 ; перевірка наступного умови.
```

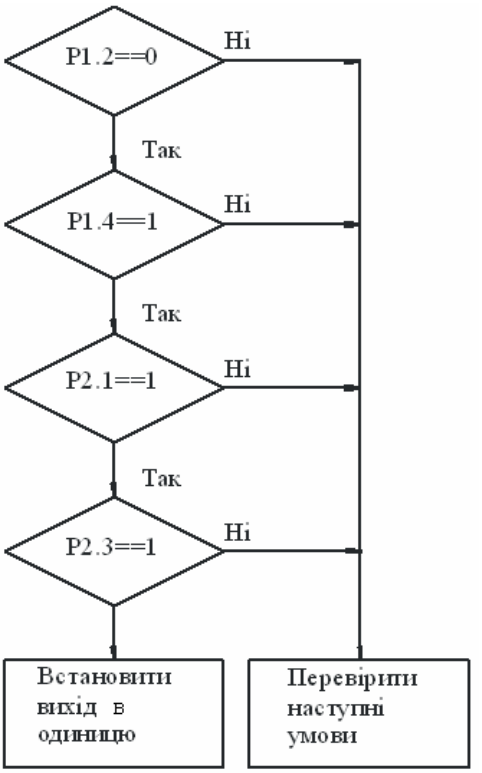


Рис. 3.5. Схема алгоритму програмної реалізації логічної функції «І»

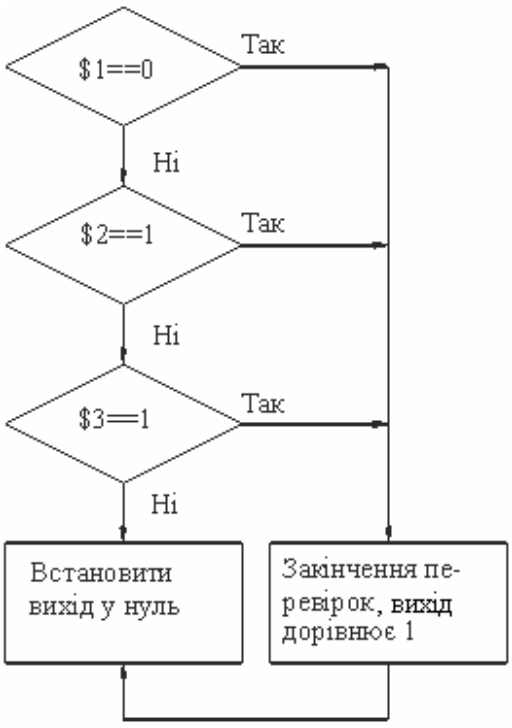


Рис. 3.6. Схема алгоритму програмної реалізації логічної функції «АБО»

Слід звернути увагу, що команди умовного переходу при виконанні умови «порушують» послідовне виконання програми. Тому, для того щоб програма була зручною, пропонується використовувати команди, які перевірять інверсну умову. Тоді послідовне виконання таких команд і буде відповідати виконанню вихідної умови логічної функції.

У цьому алгоритмі під &1, &2 маються на увазі логічні функції «І» наведеного рівняння. Повна програма логічного рівняння, що відповідає до таблиці істинності, буде:

```
        JB    P1.2,M1; Перевірка умови першого рядка таблиці
        JNB P1.4,M1
        JNB P2.1,M1
        JNB P2.3,M1
        SETB P1.7   ; встановлення виходу в одиницю
        LJMP EXIT ; і вихід
M1:
        JNB P1.2,M2; Перевірка умови другого рядка таблиці
        JB    P1.4,M2
        JNB P2.1,M2
        JB    P2.3,M2
        SETB P1.7    ; встановлення виходу в одиницю
        LJMP EXIT  ; і вихід
M2:     JNB P1.2,M3; Перевірка умови третього рядка таблиці
```

```
        JB    P1.4,M3
        JNB P2.1,M3
        JB    P2.3,M3
        SETB P1.77    ; встановлення виходу в одиницю
        LJMP EXIT     ; і вихід
M3:     CLR   P1.7    ; скинути вихід у нуль – не виконалася жодна з
                        умов
EXIT:
```

Таким чином, реалізується формальний перехід від таблиці істинності до програми керування.

Контрольні запитання

1. Який порт МК51, крім порту 0, використовується при зверненні до зовнішньої пам’яті?
2. Який з портів МК51 має подвійне призначення?
3. В якому порті МК-51 розташовані сигнали читання та запису інформації в зовнішню пам’ять даних?
4. Яким чином вхід паралельного порту налагоджується на ввід інформації?
5. Яка навантажувальна здатність порту 0?
6. Скільки паралельних портів має мікроконтролер МК-51?
7. Яка розрядність паралельних портів мікроконтролера МК-51?

4. ОРГАНІЗАЦІЯ ПЕРЕРИВАНЬ У МІКРОПРОЦЕСОРНИХ СИСТЕМАХ

Логіку роботи режиму переривань зручно розглянути на прикладі читання лекції. Професор викладає лекцію (мікропроцесор виконує програму). Якщо студенту щось не зрозуміло, він піднімає руку (виставляє сигнал-запит на переривання), щоб професор дав роз'яснення з тих або інших питань. Професор запам'ятовує, у якому місці перервана лекція (мікропроцесор запам'ятовує у стековій пам'яті адресу, на якій перервана програма), і після цього відповідає на запитання студента (мікропроцесор виконує підпрограму обслуговування цього запиту). Після відповідей на запитання студента (після виконання підпрограми мікропроцесором) професор продовжує лекцію з того місця, де він її перервав (мікропроцесор вертається до виконання основної програми).

Таким чином, термін «переривання» відноситься до функціонування мікропроцесора (МП). При звичайному функціонуванні МП на шину адреси виставляє вміст лічильника команд. Функції, які виконує МП, залежать від типу команди. Якщо виконуються команди пересилання інформації, арифметичні, логічні, то вміст лічильника команд (PC) збільшується відомим способом – збільшується на 1, якщо виконується однобайтова команда, збільшується на 2, якщо виконується двобайтова команда й збільшується на 3, якщо виконується трибайтова команда. Новий уміст лічильника команд і є тепер адресою наступної команди в пам'яті програм.

Під час виконання команд умовних і безумовних переходів (JMP, JC), вміст PC модифікується, як правило, другим і третім байтом цих команд.

І тільки під час виконання команд виклику підпрограм LCALL вміст лічильника команд збільшується на 2 (PC+2), при виконанні команди ACALL уміст лічильника команд збільшується на 1 (PC+1) і запам'ятовується в стеку, а потім уже в лічильник команд завантажується адреса підпрограми, яка перебуває в другому й третьому байті команд LCALL або ACALL.

Таким чином, у стеку запам'ятовується адреса програми, при якій була викликана підпрограма. Після закінчення підпрограми, ця адреса буде відновлена в лічильнику команд (PC). Реагування мікропроцесора на запити переривань полягає в переході від виконання поточної програми до виконання підпрограми обслуговування переривання, відповідної до даного запиту.

У МК51 це реалізується в такий спосіб. При виставлянні сигналу-запиту на переривання від одного із джерел переривань завершується виконання поточної команди, блокується вибірка команд із пам'яті програм наступної команди. Далі контролер переривання на шину даних виставляє код операції команди LCALL з адресою підпрограми, відповідної до даного запиту. Завдяки цій команді МП поточне значення лічильника команд запам'ятовує в стеку, а в лічильник команд заноситься другий і третій байт команди LCALL. Таким чином, звичайний хід виконання програми примусово переривається й мікропроцесор переходить до виконання підпрограми.

Саме на цьому принципі й організовано переривання в мікроконтролері МК51. Функціональну схему взаємодії МП із периферійними пристроями в МК51 наведено на рис. 4.1.

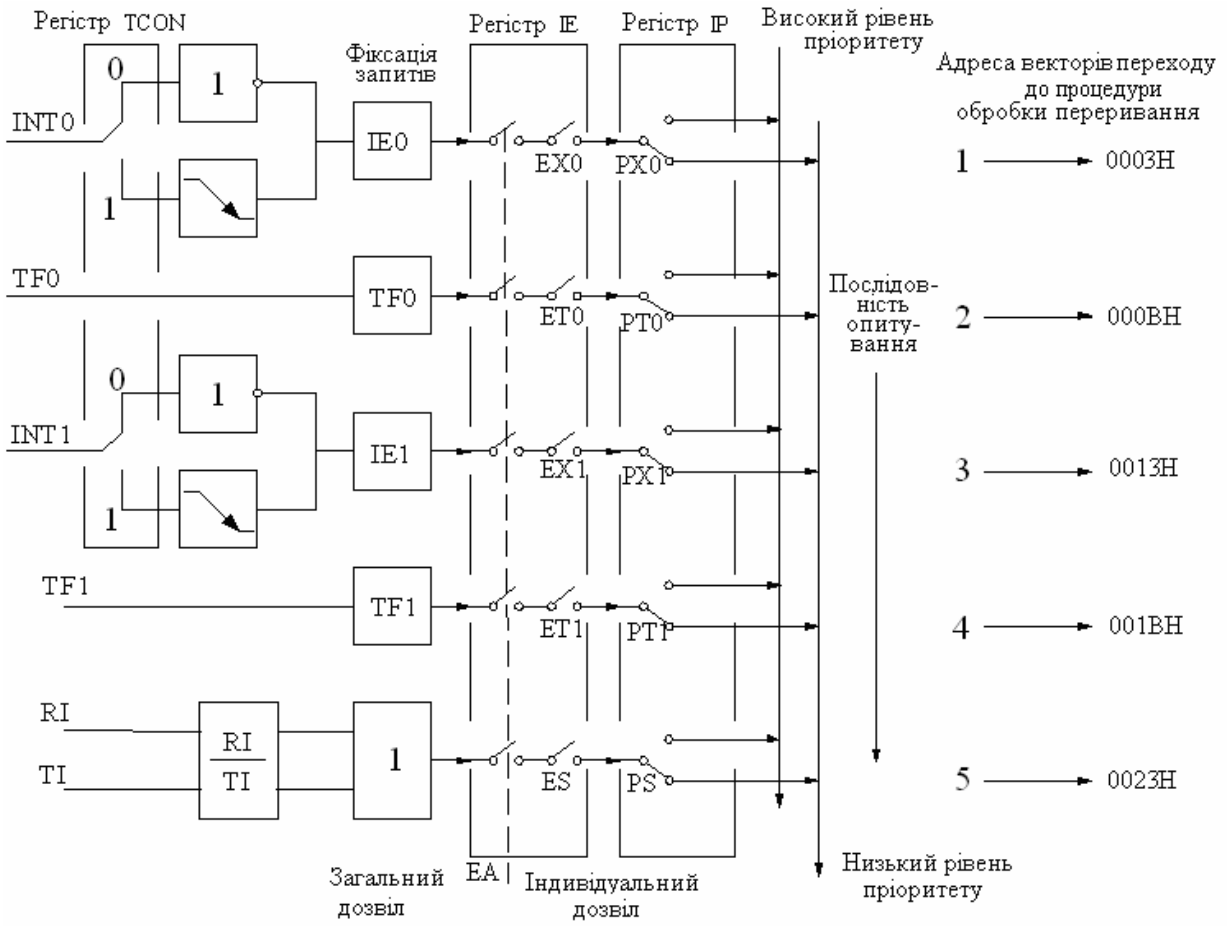


Рис. 4.1. Логіка функціонування системи переривань МК51

Роботу мікропроцесора можуть переривати як внутрішні периферійні пристрої мікроконтролера таймер 0 (T0), таймер 1 (T1), послідовний порт (SBUFF), так і зовнішні пристрої, які перебувають за межами мікроконтролера (усілякі датчики, пристрої захисту і т.п.). Для зовнішніх пристроїв у МК51 передбачено два входи від сигналів-запитів переривань ($\overline{INT0}$ і $\overline{INT1}$).

Таймери T0 і T1 формують сигнали-запити на переривання прапорами переповнення TF0 і TF1 у регістрі керування TCON (TCON.5, TCON.7) табл. 4.1.

Таблиця 4.1

Організація регістру керування TCON

Ім'я біта	Номер біта	Назва й призначення бітів
TF1	TCON.7	Прапор переповнення Таймера 1. Установлюється при переході рахункового регістру таймера зі стану FFH у стан 00H. Очищається при передачі керування на процедуру обробки переривання.

Закінчення табл. 4.1

TR1	TCON.6	Біт запуску Таймера 1. При TR1=1 рахунок дозволено.
TF0	TCON.5	Прапор переповнення Таймера 0. Установлюється при переході рахункового регістру таймера зі стану FFH у стан 00H. Очищається при передачі керування на процедуру обробки переривання.
TR0	TCON.4	Біт запуску Таймера 0. При TR0=1 рахунок дозволено.
IE1	TCON.3	Прапор запиту переривання по вхід INT1#.
IT1	TCON.2	Біт селектора типу активного сигналу на вході INT1#. При IT1=1 активним є перехід «1» – «0», при IT1=0 активним є низький рівень сигналу.
IE0	TCON.1	Прапор запиту переривання на вхід INT0#.
IT0	TCON.0	Біт селектора типу активного сигналу на вході INT0#. При IT0=1 активним є перехід «1» – «0», при IT0=0 активним є низький рівень сигналу.

Послідовний порт формує сигнали-запити переривань прапорами готовність передавача (TI) і готовність приймача (RI) у регістрі керування SCON (SCON.0, SCON.1) табл. 4.2.

Таблиця 4.2

Організація регістру керування SCON

Символічне позначення	Позиційне позначення	Назва й призначення бітів
SM0	SCON.7	Біти керування режимом послідовного порту
SM1	SCON.6	Програмно встановлюються й скидаються. SM0=0, SM1=0 – режим розширення вводу/виводу (зсувний регістр); SM0=0, SM1=1 – 8-бітний послідовний асинхронний порт, швидкість передачі визначається частотою таймера (лічильника) 1; SM0=1, SM1=0 – 9-бітний послідовний асинхронний порт, швидкість передачі: fgq/64 або fgq/32; SM0=1, SM1=1 – 9-бітний послідовний асинхронний порт, швидкість передачі визначається частотою таймера (лічильника) 1
SM2	SCON.5	Біт 2 керування режимом послідовного порту. Програмно встановлюється для заборони приймання кадрів з нульовим 8-м бітом
REN	SCON.4	Біт керування дозволом приймання. Програмно встановлюється й скидається для дозволу (заборони) приймання послідовних даних

Закінчення табл. 4.2

TB8	SCON.3	8-й біт при передачі даних. Програмно встановлюється й скидається для задання 8-го біта в режимі 9-бітного послідовного порту
RB8	SCON.2	8-й біт при прийманні даних. Апаратно встановлюється й скидається залежно від значення 8-го біта й режимі 9-бітного послідовного порту
TI	SCON.1	Прапор переривання при закінченні передачі. Установлюється апаратно після передачі байта, скидається програмно після обслуговування переривання
R1	SCON.0	Прапор переривання при закінченні приймання. Установлюється апаратно після приймання байта, скидається програмно після обслуговування переривання

Спрощено логіка функціонування системи переривань зображена на рис. 4.2, де показано п'ять сигналів запитів переривань. $INT0$, $INT1$ – зовнішні сигнали запитів переривань, $TF0$, $TF1$, SBUFF – сигнали запитів переривань від внутрішніх периферійних пристроїв (таймери Т0 і Т1 і послідовний порт). Причому зовнішні сигнали запитів переривань можуть сприйматися контролером, як за рівнем, так і за перепадом сигналу (зрізу) з «1» в «0». Вибір режиму визначається станами бітів регістру керування TCON (біти TCON.0 і TCON.3) табл. 4.2.

Якщо зазначені біти встановлені в 1, то активним є перехід з 1 в 0, при нульових значеннях цих бітів активним є низький рівень сигналів $\overline{INT0}$ або $\overline{INT1}$.

Оскільки в МК є п'ять запитів переривань, отже, є й п'ять команд LCALL з адресами відповідних підпрограм.

Причому код операції команди LCALL і адреси переходів, відповідні до кожного запиту, фіксовані для кожного джерела й формуються в контролері переривання у відповідності до табл. 4.3.

Таблиця 4.3

Адреси підпрограм відповідних запитів переривання

Запит переривання	Адреса підпрограми
Зовнішній запит INT0	0003H
Запит від таймера Т/З0	000BH
Зовнішній запит INT1	0013H
Таймер лічильник Т/З1	001BH
Послідовний порт	0023H

Як можна побачити з табл. 4.3 інтервал між адресами підпрограм становить 8 комірок пам'яті програм.

Типову структуру підпрограми обслуговування переривання можна уявити в такий спосіб (рис. 4.2). На початку будь-якої підпрограми зберігається в стеку проміжна інформація основної програми (уміст акумулятора й регістрів загального призначення та ін.). Здійснюється це командами PUSH.

```
PUSH A          ; завантаження в стек умісту акумулятора
PUSH PSW        ; завантаження в стек умісту регістру стану програми
Тіло підпрограми
POP PSW         ; вивантаження з вершини стеку в регістр стану
POP A           ; вивантаження з вершини стеку в акумулятор
RETI            ; повернення на адресу перерваної програми
```

Рис. 4.2. Типова структура підпрограми обслуговування

Потім виконується сама підпрограма обслуговування переривання (наприклад, опитування стану датчиків у системі керування). Наприкінці підпрограми за допомогою команд POP відновлюється інформація в регістрах мікропроцесора. Після виконання останньої команди POP у вершині стека буде перебувати адреса перерваної програми. Кількість команд PUSH і POP повинно бути однаковим і визначається самим розробником підпрограми, оскільки від цього залежить час обслуговування переривання. Будь-яка підпрограма обслуговування переривання завершується командою RETI. Завдяки цій команді з вершини стека пересилається в лічильник команд (PC) адреса перерваної програми.

Крім того, за командою RETI скидається прапор запиту переривання, відповідний до цієї підпрограми. Аналіз структури підпрограми показує, що розмістити підпрограму обслуговування переривання у восьми комірках пам'яті програм далеко не завжди можливо. Тому, як правило, за адресою підпрограм розміщається команда безумовного переходу jmp (адреса переходу), яка вказує адресу переходу на область пам'яті, де розміщена основна підпрограма обслуговування переривання.

Кожний із запитів переривання може мати дозвіл на обслуговування або заборону з використанням відповідних бітів регістру керування перериваннями IE (табл. 4.4).

Таблиця 4.4

Організація регістру дозволу переривань IE

Символічне позначення	Позиційне позначення	Назва й призначення
EA	IE.7	Біт дозволу переривань. Програмно скидається для заборони всіх переривань і встановлюється для дозволу переривань залежно від стану бітів IE.4- IE.0

Закінчення табл. 4.4

	IE.6	Не використовується
	IE.5	Не використовується
ES	IE.4	Біт керування перериваннями послідовного порту. Програмно встановлюється (скидається) для дозволу (заборони) переривань за прапорами TI і RL
ET1	IE.3	Біт дозволу переривання від таймера 1. Програмно встановлюється й скидається для дозволу (заборони) переривань від таймера (лічильника) 1
EX1	IE.2	Біт дозволу зовнішнього переривання 1. Програмно встановлюється (скидається) для дозволу (заборони) апаратного переривання INT1
ET0	IE.1	Біт дозволу переривання від таймера 0. Програмно встановлюється (скидається) для дозволу (заборони) переривань від таймера (лічильника) 0
EX0	IE.0	Біт дозволу зовнішнього переривань 0. Програмно встановлюється (скидається) для дозволу (заборони) апаратного переривання INT0

Установка відповідного біта регістру IE (SETB IE.x) в одиницю (еквівалентно замиканню перемикача на рис. 4.1) дозволяє обслуговування цього запиту, скидання в 0 (еквівалентно розмиканню перемикача) забороняє обслуговування. Крім того, скидання біта IE.7 (CLR IE.7) забороняє обслуговування всіх запитів переривання.

З одночасною появою декількох запитів переривання в МК51 передбачено двоступінчастий механізм пріоритетів переривань. Перший щабель механізму пріоритетів забезпечується станами бітів регістру пріоритетів IP (табл. 4.5).

Таблиця 4.5

Організація регістру пріоритетів переривань IP

Символічне позначення	Позиційне позначення	Назва й призначення
	IP.7	Не використовується
	IP.6	Не використовується
	IP.5	Не використовується
PS	IP.4	Біт керування пріоритетом переривань послідовного порту. Програмно встановлюється (скидається) для присвоєння високого (низького) пріоритету за допомогою переривання послідовного порту

Закінчення табл. 4.5

PT1	IP.3	Біт керування пріоритетом переривання від таймера 1. Програмно встановлюється (скидається) для присвоєння високого (низького) пріоритету за допомогою переривання від таймера (лічильника)1
PX1	IP.2	Біт керування пріоритетом зовнішнього переривання 1. Програмно встановлюється (скидається) для присвоєння високого (низького) пріоритету апаратному перериванню 1
PT0	IP.1	Біт керування пріоритетом переривання від таймера 0. Програмно встановлюється (скидається) для присвоєння високого (низького) пріоритету за допомогою переривання від таймера (лічильника) 0
PX0	IP.0	Біт керування пріоритетом зовнішнього переривання 0. Програмно встановлюється (скидається) для присвоєння високого (низького) пріоритету апаратному перериванню INT0

Установка відповідного біта регістру IP в 1 (еквівалентно верхньому положенню перемикача PX) привласнює «високий» рівень, а скидання в 0 (еквівалентно нижньому положенню перемикача PX) «низький».

Другий щабель реалізовано у послідовності опитування запитів переривання, яка встановлює фіксовану черговість обслуговування. При цьому запиті INT0 привласнюється найвищий пріоритет, а послідовному порту (запити RI, TI) найнижчий (рис. 4.1).

Алгоритм функціонування системи обслуговування запитів переривання наведено на рис. 4.3. На ньому можна побачити, що програма з вищим пріоритетом може перервати виконання програми з нижчим пріоритетом. Таку організацію роботи системи переривань називають режимом вкладених переривань (рис. 4.4). З рисунку можна побачити, що основну програму на адресі 0007H переривається підпрограма, що відповідає запиту T1.

Потім виконання цієї підпрограми на адресі 1002H переривається підпрограма з більш високим пріоритетом, що відповідає запиту INT1, яку на адресі 1303H переривається підпрограма з найвищим пріоритетом, відповідна до запиту INT0.

Як можна побачити з рисунку, спочатку завершується виконання підпрограми, відповідної до запиту INT0, потім мікропроцесор перейде до завершення підпрограми, що відповідає запиту TI, і тільки після завершення цієї підпрограми МП переходить до виконання основної програми.

На рис. 4.3 також зображено уміст стеку на момент виконання підпрограми, відповідної до запиту INT0, де можна побачити, що всі адреси перерваних підпрограм і адреса переривання основної програми запам'ятовуються в стеку.

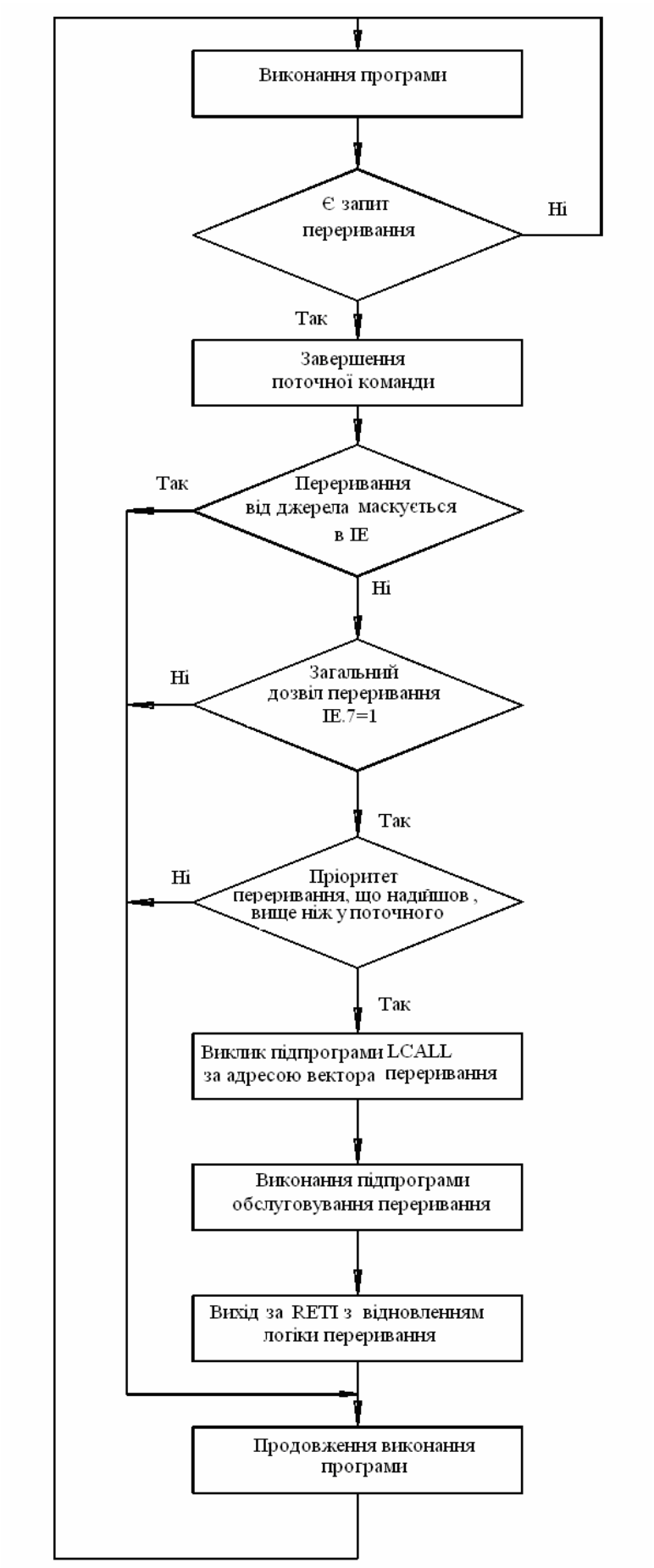


Рис. 4.3. Алгоритм обробки переривань у МК51

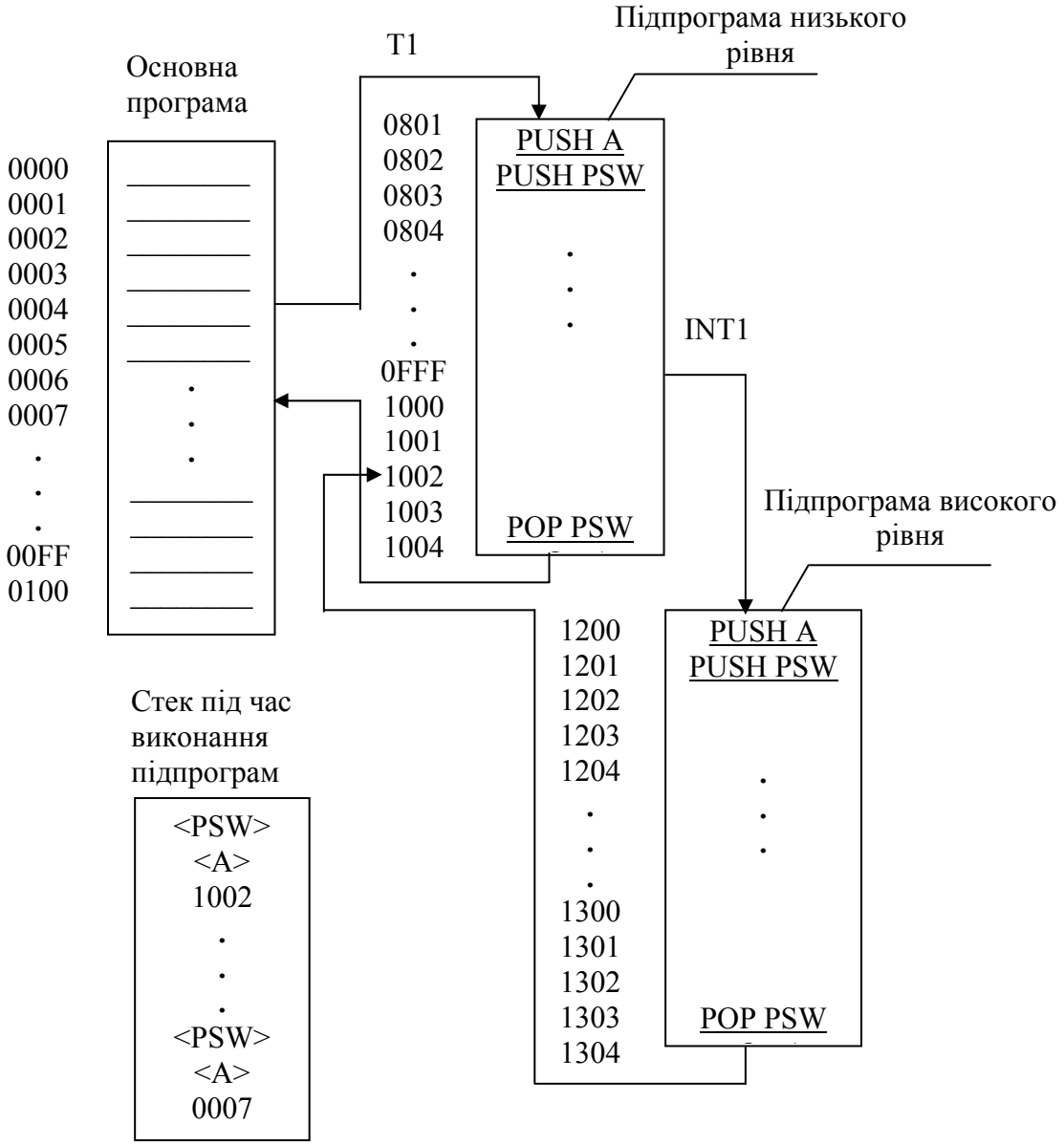


Рис. 4.4. Режим вкладених переривань

Контрольні запитання

1. Скільки сигналів переривань обробляється в МК51?
2. Скільки сигналів переривань обробляється від зовнішніх пристроїв в МК51?
3. Який регістр МК51 призначено для керування режимом переривань?
4. Який регістр МК51 призначено для керування пріоритетом запитів переривань?
5. Який сигнал дозволяє переривання від послідовного порту?
6. Який сигнал дозволяє переривання від таймера 0?
7. Який сигнал забороняє чи дозволяє обслуговування переривань від усіх пристроїв?
8. Який сигнал встановлює найвищий пріоритет запиту від таймера 0?
9. Який сигнал встановлює найвищий пріоритет запиту від послідовного порту?
10. Який сигнал встановлює найвищий пріоритет запиту від зовнішнього пристрою 1?

5. ОРГАНІЗАЦІЯ ТИМЧАСОВИХ ЗАТРИМОК У МІКРОПРОЦЕСОРНИХ СИСТЕМАХ

Реалізація закону керування в мікропроцесорних системах (МПС) здійснюється за допомогою впливу на об'єкт як апаратно (формування усіляких сигналів, наприклад, увімкнення реле, тиристора), так і програмно, при цьому сигнали керування виробляються програмою, яка виконується мікропроцесором. Для прикладу розглянемо роботу звичайного автодорожнього світлофора. Час горіння зеленого й червоного світла – у межах 40 – 150 секунд залежно від інтенсивності руху. Жовтого – 3 – 7 секунд. Якщо світлофор обладнано кнопкою для увімкнення зеленого світла або звуковим сигналом для незрячих, то зелене світло повинно горіти від 50 до 100 секунд. Із цього прикладу можна зробити висновок, що в системі керування світлофором необхідно формувати кілька різних тимчасових інтервалів. Вони залежать від результатів подій у системі. Натиснута пішоходом кнопка – завершився часовий інтервал горіння червоного або жовтого світла і т.п.

Мікроконтролер відрізняється від мікропроцесора тим, що апаратні функції керування реалізуються периферійними пристроями, розміщеними на тому самому кристалі, що й мікропроцесор (таймери, послідовний порт, АЦП, контролер переривань і т.п.) При цьому реалізація апаратного й програмного впливів на об'єкт здійснюється паралельно, як правило, цей час називають системним. Він організований на основі деякого елементарного інтервалу, що формується тактовим генератором мікропроцесора, і являє собою сукупність різних тимчасових інтервалів, яку називають тимчасовою сіткою. Тимчасова сітка, як правило, формується таймерами.

При цьому в системі формуються сигнали початку $(t_н)$, закінчення $(t_о)$ інтервалу й, власне, величина тимчасового інтервалу $(t_з)$ (рис. 5.1).

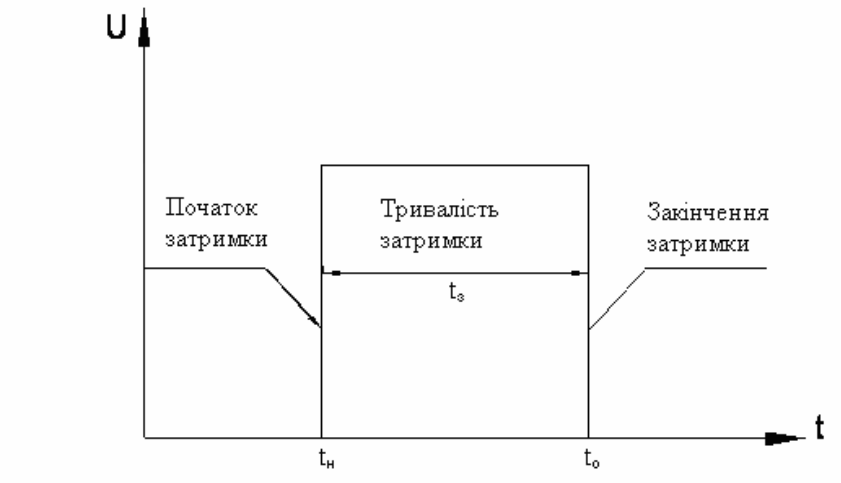


Рис. 5.1. Графічне зображення тимчасового інтервалу

У базовій моделі МК-51 для формування тимчасових інтервалів є два таймери, які являють собою регістри, що працюють у рахункових режимах.

Ідею роботи таймера розглянемо на прикладі посудини, яка наповнюється із крапельниці (рис. 5.2)

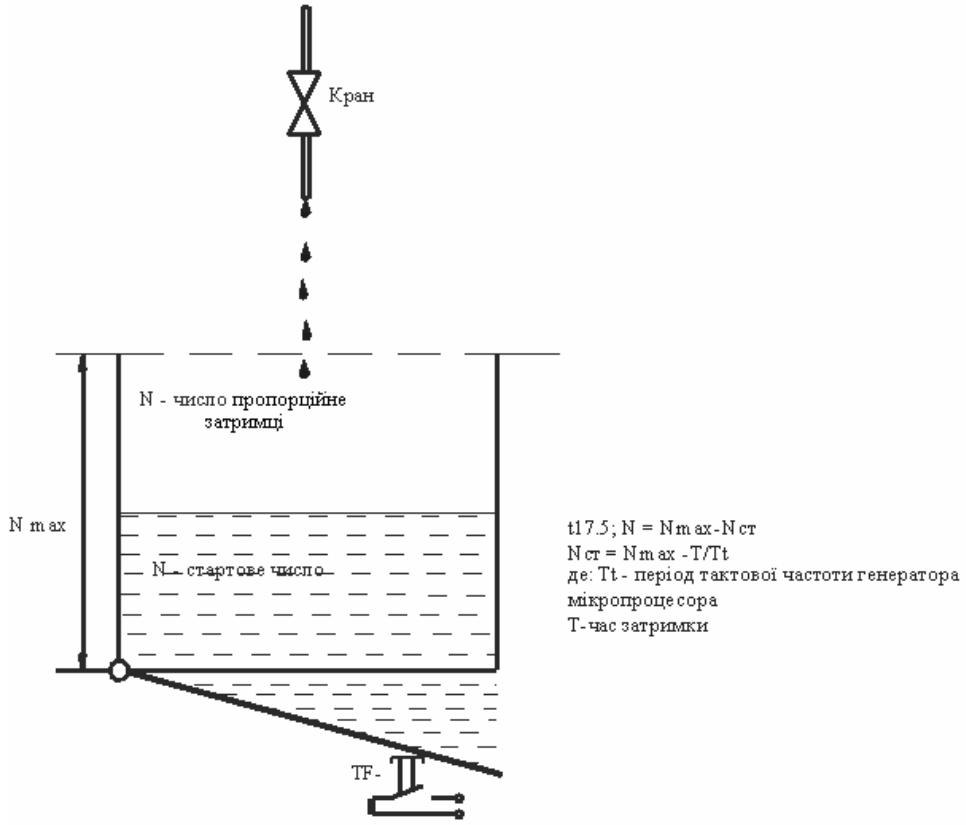


Рис. 5.2. Принцип роботи таймера

Краплі починають надходити в посудину, якщо відкритий кран.

Під час наповнення посудини до верху відбувається відкривання дна, рідина виливається, посудина стає порожньою, дно закривається й система приходить у вихідний стан (роботу механізму відкривання й закривання дна посудини для спрощення не розглядаємо).

Якщо краплі починають капати в порожню посудину, то для її заповнення необхідно N_{max} крапель. Якщо ж попередньо в посудину налити N стартове число крапель і потім відкрити кран, то для її заповнення знадобиться менше число крапель. При постійній швидкості руху краплі час наповнення ємності до верху так само буде менше.

Таким чином, завдяки зміні стартового числа крапель можна змінювати час заповнення посудини до верху.

При цьому початку затримки відповідає відкриття крана для руху крапель, закінчення затримки фіксується замиканням контакту при заповненні посудини до верху й відкриванні її дна. Величина тимчасового інтервалу пропорційна числу крапель, яке необхідно накапати в посудину, щоб вона наповнилася до верху.

У базовій моделі МК-51 використовуються два 16-розрядних лічильники, Таймер 0 і Таймер 1, кожний з яких працює незалежно в чотирьох режимах 0, 1, 2, 3.

Логіку функціонування таймера зображено на рис. 5.3, а графіки тимчасового аналізу роботи на рис. 5.4.

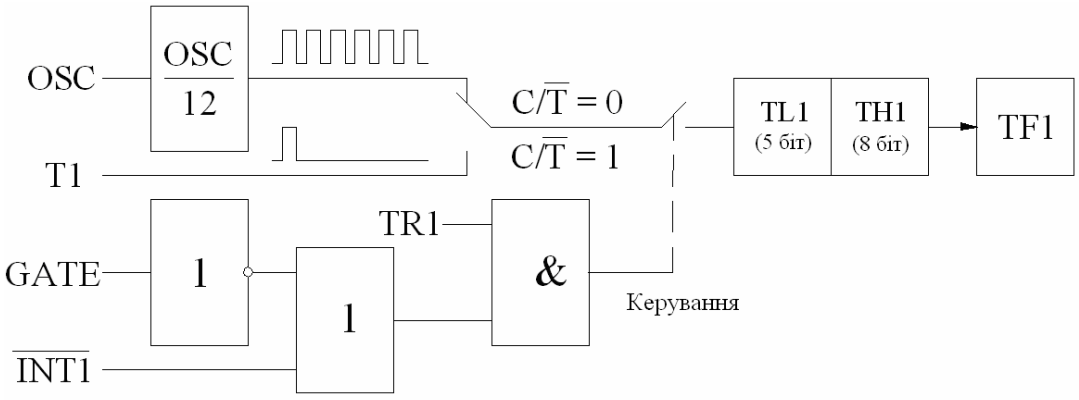


Рис. 5.3. Функціональна схема таймера в режимі 0

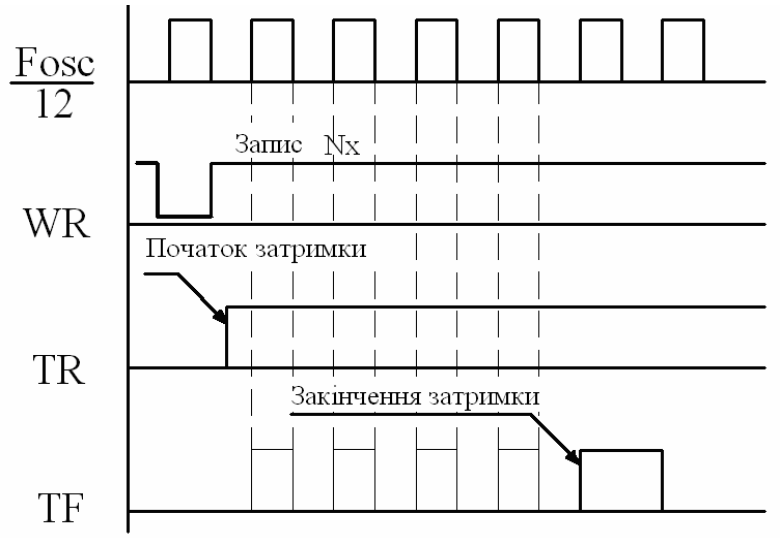


Рис. 5.4. Графік тимчасового аналізу роботи таймера в режимі 0

Кожен таймер може бути запрограмований, власне, у свій режим, у цьому випадку здійснюється підрахунок імпульсів тактового генератора МП (підрахунок крапель). А також – в режим лічильника зовнішніх подій (тобто є два види крапельниць).

Для керування організацією взаємодії таймерів із системою переривання й установлення режимів їх роботи використовується два регістри спеціальних функцій. Регістр режиму TMOD і регістр керування TCON.

Вибір режиму роботи кожного таймера здійснюється шляхом пересилання в регістр режиму TMOD керуючого слова, яке складається відповідно до табл. 5.1.

У режимі таймера вміст регістру лічильника збільшується на 1 (інкрементується) у кожному машинному циклі МП, при цьому частота рахування складає $F_{osc}/12$ (при частоті кварцового резонатора 12 МГц, частота рахування складає 1 МГц).

Таблиця 5.1

Призначення бітів регістрів TMOD

Ім'я біта	Номер біта	Функція
GATE1	TMOD.7	Біт керування Таймером 1. При GATE1=1 Таймер 1 працює завжди при TR1=1. При GATE1=0 для роботи необхідна умова TR1=1 і INT1# =1
З/Т1#	TMOD.6	Біт вибору типу подій для Таймера 1. При С/Т1# = 1 він працює як лічильник, при С/Т1# = 0 як таймер.
М1.1	TMOD.5	Біт 1 визначення режиму роботи Таймера 1.
М1.0	TMOD.4	Біт 0 визначення режиму роботи Таймера 1.
GATE0	TMOD.3	Біт керування Таймером 0. При GATE0 = 1 Таймер 0 працює завжди при TR0=1. При GATE0=0 для роботи необхідна умова TR0=1 і INT0# =1
З/ТО#	TMOD.2	Біт вибору типу подій при таймері 0. При С/ТО# = 1 він працює як лічильник, при С/ТО# = 0 як таймер.
М1.0	TMOD.1	Біт 1 визначення режиму роботи Таймера 0.
М0.0	TMOD.0	Біт 0 визначення режиму роботи Таймера 0

У режимі лічильника інкрементування регістру лічильника (TH, TL) здійснюється при переході вхідного сигналу (T0 або T1) з «1» в «0».

Входи аналізуються у фазі S5P2 кожного машинного циклу. Інкрементування вмісту регістру лічильника здійснюється в циклі, що слідує за тим, у якому було виявлено перехід з «1» в «0». Таким чином, на розпізнавання переходу в цьому режимі необхідно два машинні цикли роботи мікропроцесора, частота рахунку в цьому випадку складе $F_{osc} / 24$ (частота рахунку 500 КГц).

Вибір типу підраховуваних подій визначається станами бітів TMOD.2 і TMOD.6 регістру режиму TMOD. При одиничному стані цих бітів рахункові регістри працюють як лічильники, при нульовому стані як таймери.

Особливістю регістру режиму TMOD є неможливість керування окремими бітами. Таким чином, для встановлення режиму роботи таймера або рахункового регістру необхідно в регістр TMOD переслати керуюче слово, яке складається відповідно до обраного режиму, згідно табл. 5.2.

Біти TMOD.0 і TMOD.1 регістру режиму визначають режим роботи таймера 0, а TMOD.4 і TMOD.5 – режими роботи таймера 1 (табл. 5.3.)

Таблиця 5.2

Структура керуючого слова регістру TMOD

TMOD.7	TMOD.66.	TMOD.55	TMOD.44	TMOD.33	TMOD.22	TMOD.11	TMOD.00
GATE1	C/T	M1	M0	GATE0	C/T	M1	M0
Таймер 1				Таймер 0			

Таблиця 5.3

Вибір режимів роботи таймера

M1	M0	Режим роботи
0	0	**Режим 0**. TH_x як 8-розрядний таймер/лічильник. TL_x як 5-розрядний переддільник.
0	1	**Режим 1.** TH_x і TL_x включені послідовно і являють собою 16-розрядний таймер/лічильник.
1	0	**Режим 2.** TL_x являє собою 8-розрядний таймер/лічильник з автоперезавантаженням значення із TH_x.
1	1	**Режим 3.** TL0 як 8-розрядний таймер/лічильник, який керується бітами регістра TCON Таймера 0, TH0 як 8-розрядний таймер/лічильник, який керується бітами регістра TCON Таймера 1. Таймер 1 не працює.

Наприклад, Таймер 0 потрібно встановити в режим лічильника зовнішніх подій, а Таймер 1 у режим таймера. Тоді біт TMOD.2 повинен бути встановлений в 1, а TMOD.6 в 0. У Таймері 0 і в Таймері 1 використовуємо конфігурацію 0, тобто біти TMOD.0, TMOD.1, TMOD.4 TMOD.5 перебувають у стані 0. Біти TMOD.3, TMOD.7 (сигнали GATE0 і GATE1) ухвалюємо такими, що дорівнюють 0.

Тоді для програмування режиму роботи таймерів у регістр TMOD занесемо керуюче слово

MOV TMOD, #04H (00000100B).

Стан бітів TMOD.7 і TMOD.3 впливає на те, як здійснюється керування початком рахунку регістрів TH і TL (відкривання крана в крапельниці). При одиничному стані зазначених бітів запуск таймера (початок рахунку) здійснюється бітами запуску таймера TR0 або TR1 (біти SETB TCON.6 або SETB TCON.4) регістру керування TCON (табл. 5.4.).

При нульовому стані бітів TMOD7 і TMOD.3 запуск таймера здійснюється попередньою установкою TR0 або TR1 і наявності сигналу INT0 або INT1, які є інверсними (активним нулем), у цьому випадку при переході сигналу на вході INT0 або INT1 з «0» в «1» відбувається запуск таймера, а при переході з «1» в «0» рахунок зупиняється, рис. 5.5.

Таким чином, уміст лічильника прямо пропорційний тривалості імпульсу сигналу INT0 або INT1, тобто в цьому режимі за допомогою таймера можна вимірювати тривалість імпульсів, які подаються на входи INT0 або INT1.

Після того, як лічильник повністю заповнено (посудина наповнилася до верху) стан «усі одиниці» (FFFFH) переходить у стан «усі нулі» (0000H) (відкрилося дно посудини), формується сигнал закінчення рахунку TF0 (замкнувся контакт при відкриванні дна посудини) або TF1, які використовується як запити на переривання від таймерів T0 або T1. Якщо вважати тривалість машинного циклу 1 мкс при частоті тактування T_t 12 МГц,

а максимальне число імпульсів N_{max} (число крапель, яке розміщується в посудині), яке може порахувати рахунковий регістр складає FFFFH (65536 десятковий еквівалент), то максимальний часовий інтервал на одному таймері (на початку рахунку посудина пуста) складе 65,536 мс.

Таблиця 5.4

Призначення бітів регістру TCON

Ім'я біта	Номер біта	Функція
TF1	TCON.7	Прапор переповнення таймера 1. Установлюється при переході рахункового регістру таймера зі стану FFH у стан 00H. Очищається при передачі керування на процедуру обробки переривання.
TR1	TCON.6	Біт запуску Таймера 1. При TR1=1 рахунок дозволено. Установлюється й скидається програмно
TF0	TCON.5	Прапор переповнення таймера 0. Установлюється при переході рахункового регістру таймера зі стану FFH у стан 00H. Очищається при передачі керування на процедуру обробки переривання.
TR0	TCON.4	Біт запуску Таймера 0. При TR0=1 рахунок дозволено.
IE1	TCON.3	Прапор запиту зовнішнього переривання за входом INT1. Установлюється апаратно при виявленні запиту за входом INT1. Скидання залежить від стану біта IT1
IT1	TCON.2	Біт керування типом сигналу переривання INT1, При IT1=1 – визначається за зрізом (перехід 1-0), Скидання IE1 здійснюється апаратно (автоматично); При IT1=0 – визначається за низьким рівнем сигналу на вході INT1. Скидання IE1 здійснюється програмно.
IE0	TCON.1	Прапор запиту переривання по вхід INT0 (аналогічно IE1)
IT0	TCON.0	Біт керування типом сигналу переривання INT0 (аналогічно IT1).

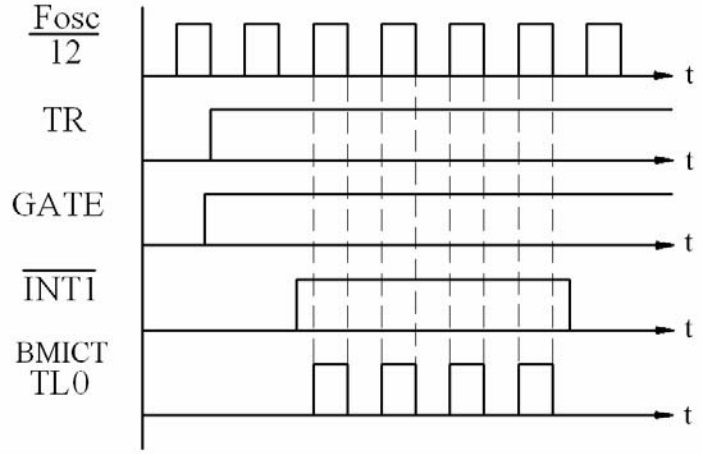


Рис. 5.5. Графік тимчасового аналізу роботи таймера в режимі 0 і 1

Регулювати величину затримки, що формується таймером, можна зміною стартового числа N_{CT}, яке попередньо записується в рахунковий регістр (попереднє накапування рідини в посудину). Тоді тривалість затримки можна визначити відповідно до виразу, оскільки

$$N_з = N_{\max} - N_{CT},$$

а $N_з = \dfrac{T_з}{T_t}$, де $N_з$ – число, яке пропорційне величині часової затримки, тоді

$$N_{CT} = N_{\max} - \frac{T_з}{T_t},$$

$$\frac{T_з}{T_t} = N_{\max} - N_{CT},$$

$$T_з = \left(N_{max} - N_{CT}\right) T_t.$$

Наприклад, потрібно сформувати часову затримку тривалістю $T_з = 1$ мс, тоді

$$N_{CT} = N_{\max} - \frac{T_з}{T_t} = 65536 - \frac{1 \cdot 10^{-3}}{1 \cdot 10^{-6}} = 65536 - 1000 = 64536.$$

Це число попередньо необхідно записати в рахунковий регістр Таймера, а потім встановити біт запуску Таймера (TR0 або TR1).

У режимі 0 рахунковий регістр містить 13 розрядів, у якому регістр TH працює як 8-розрядний лічильник, а регістр TL як 5-бітовий переддільник.

У режимі 1 рахунковий регістр містить 16 розрядів, тобто TH має 8 розрядів і TL – 8 розрядів.

Режим 2 можна назвати режимом програмувального генератора, рис. 5.6.

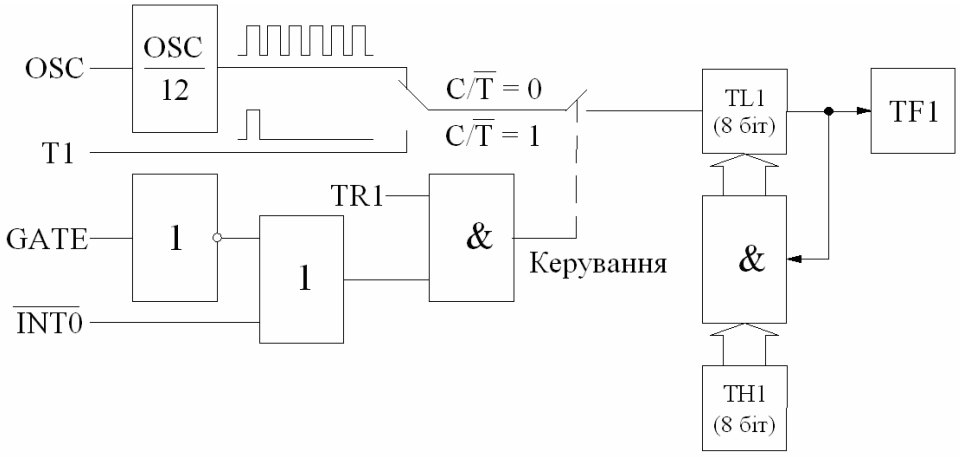


Рис. 5.6. Функціональна схема таймера в режимі 2

При цьому регістр TL працює як 8-розрядний рахунковий регістр, а в TH завантажується число, яке визначає частоту імпульсів на виході таймера.

Після переповнення регістру TL автоматично вміст регістру TH перезавантажується в регістр TL, при цьому вміст регістру TH зберігається. На графіку тимчасового аналізу (рис. 5.7) для прикладу показано, що в регістр TH попередньо занесене число EAH (250 у десятковому еквіваленті, тобто для заповнення посудини до верху необхідно всього 5 крапель) і через кожні 5 імпульсів на виході таймера буде формуватися імпульс, частота повторення якого в розглянутому прикладі складе

$$F = 1\text{МГц} / 5 = 200\text{КГц}.$$

Таким чином, змінюючи вміст регістру TH (MOV Thx, #nx), можна змінювати частоту імпульсів на виході рахункового регістру.

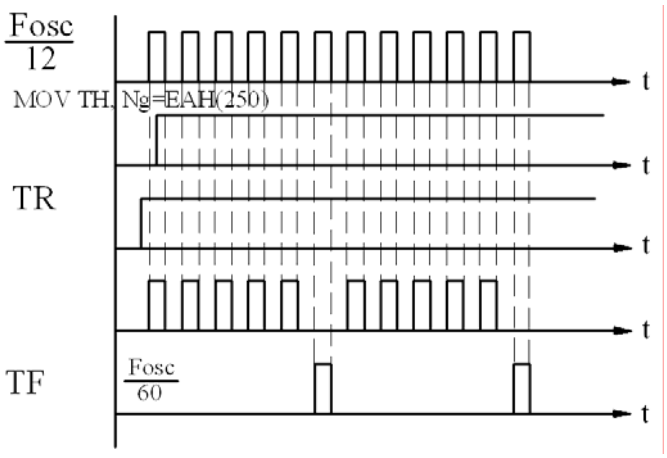


Рис. 5.7. Графік тимчасового аналізу роботи таймера в режимі 2

У режимі 3 Таймер 1 заблоковано еквівалентно (TR1=0), а Таймер 0 працює як два 8-розрядні рахункові регістри, причому регістр TL0 керується бітами Таймера 0, а TH0 – бітами Таймера 1 (рис. 5.8).

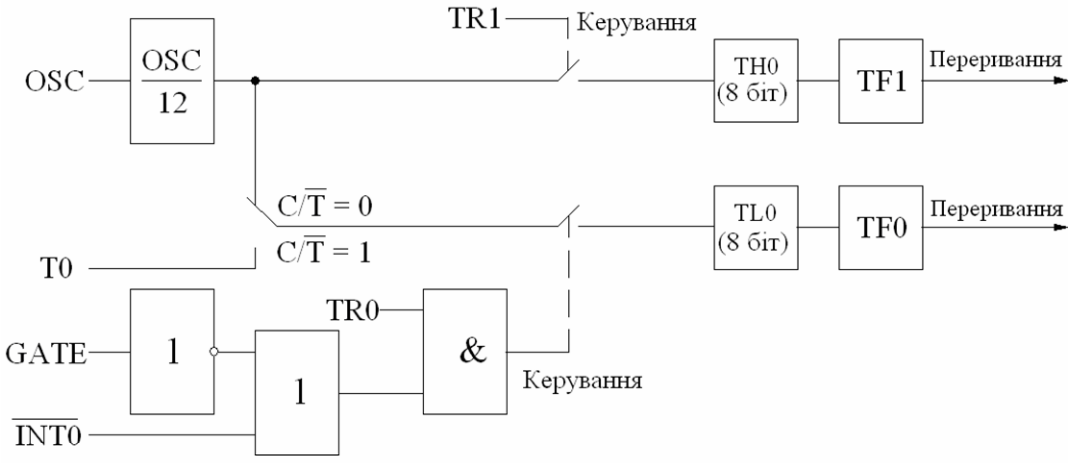


Рис. 5.8. Функціональна схема таймера в режимі 3

Взаємодія таймера із системою переривання визначає регістр TCON, у якому установлена бітова адресація. Призначення бітів цього регістру наведено в табл. 5.4. Біти TCON.7 і TCON.5 фіксують переповнення таймера й використовуються як сигнали запиту переривання від таймера до мікропроцесора. Біти TCON.4 і TCON.6 керують запуском таймерів за командою SETB TCON.4 або SETB TCON.6. Біти TCON.3 і TCON.1 фіксують зовнішні сигнали запитів переривання. Біти TCON.2 і TCON.0 керують системою скидання сигналів INT0 і INT1. При одиничному стані TCON.2 аналіз сигналу INT1 виконується за зрізом, а скидання автоматично (апаратно). При нульовому стані біта TCON.2 аналіз запиту сигналу INT1 здійснюється за низьким рівнем сигналу на вході, а скидання сигналу INT1 виконується програмно після закінчення підпрограми обслуговування переривання.

У режимі 0 рахунковий регістр містить 13 розрядів, у якому регістр TH працює як 8-розрядний лічильник, а регістр TL як 5-бітовий переддільник.

У режимі 1 рахунковий регістр містить 16 розрядів, тобто 8 розрядів TH і вісім TL.

Режим 2 можна назвати режимом програмного генератора. При цьому регістр TL працює як 8-розрядний рахунковий регістр, а в TH завантажується число пропорційне до частоти імпульсів на виході таймера.

Після переповнення регістру TL автоматично вміст регістру TH перезавантажується в регістр TL, при цьому вміст регістру TH зберігається.

Таким чином, змінюючи вміст регістру TH (MOV Thx, #nx), можна змінювати частоту імпульсів на виході рахункового регістру.

У режимі 3 (рис. 5.8) Таймер 1 заблоковано еквівалентно (TR1=0), а Таймер 0 працює як два 8-розрядні рахункові регістри, причому регістр TL0 керується бітами Таймера 0, а THO – бітами Таймера 1.

Контрольні запитання

1. Скільки таймерів містить МК51?
2. Скільки режимів роботи має таймер МК51?
3. Який сигнал МК51 формує початок затримки?
4. Який сигнал формує закінчення затримки?
5. Яке максимальне число можна завантажити в таймер?
6. Яку максимальну затримку можна організувати на одному таймері?
7. Вміст таймера в процесі роботи?
8. Який регістр забезпечує установку режимів роботи таймера?
9. Під дією яких сигналів змінюється вміст таймера?
10. Скільки розрядів використовується при роботі таймера в режимі 0?

6. ПРАВИЛА НАПИСАННЯ ПРОГРАМ ДЛЯ МК 51

6.1. Компілятор для мікроконтролерів сімейства МК51

Системні угоди

Наступні розширення імен файлів будуть використані за замовчуванням програмами пакета фірми 2550 A.D:

asm – вхідний файл для асемблера (компілятора);
obj – вихідний файл із асемблера;
pak – упаковано вихідний файл;
lst – файл лістингу.

Відзначимо, що у вихідний файл Асемблера включається додаткова інформація, і Редактор зв'язків (лінкер) повинен бути виконаний, навіть якщо програма розміщена з потрібної адреси й не містить зовнішніх посилань. При цьому віддаляється додаткова інформація, а також генерується файл у необхідному форматі.

6.2. Синтаксис мови асемблера

Визначники підстави системи числення:

B Binary (двійкова);
O | Q Octal (вісімкова);
D Decimal (десяткова);
H Hex (шістнадцяткова);
"X" | 'X' Ascii (Ascii-Коди).

Визначені значення двох послідовних символів, що укладені в одиночні або подвійні символи. Однак, щоб користуватися ними необхідна директива TWOCHAR ON ('CR' – повернення каретки, 'LF' – переклад рядка, 'SP' – пробіл, 'HT' – горизонтальна табуляція, 'NL' – порожній символ).

Коментарі. Рядки коментарів повинні починатися із крапки, коми або зірочки в першій колонці, за винятком використання директиви COMMENT. Коментарі після інструкцій не супроводжуються крапкою з комою, якщо асемблювання виконується в Spaces Off Mode. Якщо асемблювання виконується в Spaces On Mode, усі коментарі повинні починатися із крапки з комою. Докладніше дивіться директиву SPACES.

Програмний лічильник. Спеціальні символи долар ($) і зірочка (*) слід використовувати у виразах, щоб визначити програмний лічильник. Величина, привласнена знаку долар, відповідає значенню лічильника команд на початку цієї інструкції.

Мітки. Нелокальні мітки можуть складатися з будь-якого числа символів, але тільки 32 символи будуть значущими. Мітки ставляться в будь-якій колонці, якщо ім'я кінчається двокрапкою. Якщо двокрапка не використовується, мітка повинна починатися з першої колонки. Більші й маленькі букви вважаються різними.

Локальні мітки. Локальні мітки можуть використовуватися подібно нелокальним міткам. Відмінність у тому, що локальна мітка визначена тільки між нелокальними міткам. Коли програма переходить від однієї локальної зони до іншої, ім'я локальної мітки використовується повторно. Ця особливість використовується для міток, викликуваних тільки в локальній області, як розглянуто вище. Нові імена міток не потрібні. Асемблер визначає локальні мітки за символом долар на початку або кінці імені. Цей ідентифікатор може бути змінений за допомогою директиви LLCHAR.

Старший байт. Для завантаження старшого байта 16-бітної величини використовується арифметичний символ «більше ніж» «>». Це дозволить бітам з 8 по 15 використовуватися в якості байта, величина якого зміщена.

Молодший байт. Для завантаження молодшого байта 16-бітної величини використовується арифметичний символ «менше ніж» «<». Це дозволить бітам з 0 по 7 використовуватися в якості байта, величина якого зміщена.

Рядкові й прописні символи. Мітки, записані рядковими й прописними буквами, розглядаються як різні. Мітки, використовувані для імен секцій або макросів, різняться аналогічним чином.

6.3. Директиви асемблера

Ця частина описує директиви Асемблера, які дозволяють керувати процесом компіляції. Повний перелік директив Асемблера займає 65 сторінок, тому в даному розділі наводяться найбільш уживані й необхідні директиви.

ORG VALUE

Установлює адресу програми. Якщо директива не виконана, адреса за замовчуванням установлюється в 0000.

END VALUE

Ця директива відзначає кінець програми або підключеного файлу. Вираз, що є наступним за END – додатковий, і якщо існує, то зазначає стартову адресу програми. Вона заноситься у вихідний файл, якщо у вихідному форматі існує тип запису стартової адреси.

LABEL: DB VALUE

Асемблер поміщає величини в послідовні комірки пам'яті. Якщо після директиви не випливає вираз, один байт резервується й обнуляється. Мітка є необов'язковою.

LABEL: DW VALUE

Ця директива поміщає 16-бітні величини у пам’ять. Кілька слів можуть бути задані, якщо записати кілька виразів через кому. Якщо виразу немає, резервується й обнуляється одне слово. Мітка не є обов'язковою.

LABEL: LONG VALUE

Ця директива поміщає 32-бітні величини у пам’ять. Кілька слів можуть бути задані, якщо записати кілька виразів через кому. Якщо виразу немає, резервується й обнуляється одне слово. Мітка не є обов'язковою.

LABEL: DS SIZE,VALUE

Ця директива резервує фіксоване число байтів, обумовлене величиною SIZE. Ніякі величини не запам'ятовуються в резервованій області.

LABEL: EQU VALUE

Привласнює імені LABEL значення VALUE. VALUE може бути іншим символом або дозволеним математичним виразом.

LABEL: VAR VALUE

Привласнює імені LABEL значення VALUE, але значення може змінюватися за текстом програми. Ім'я, певно як VAR, не слід перевизначати директивою EQU.

RADIX <значення>

Підстава	Значення
2 або B	Двійкове
8 або O або Q	Вісімкове
10 або D	Десяткове
16 або H	Шістнадцяткове

Відсутність виразу означає повернення до стандартного (що використовується за замовчуванням) режиму (тобто з основою системи числення 10); при цьому мається на увазі, що всі інші системи числення будуть позначатися за допомогою літер B, Q, D або H після константи. Слід зазначити, що при завданні основи системи числення 16 не існує способу опису десяткового або двійкового числа, оскільки як літера D, так і літера B є припустимими шістнадцятковими цифрами.

INCLUDE <ім'я-файлу>.

Указує асемблеру включити даний файл у процес асемблювання. Імена файлів можуть містити в собі маршрутні імена. Розширення імен повинні задаватися повністю. Вкладене включення файлів не допускається.

6.4. Обчислення під час транслювання

Поданий далі список містить перелік обчислювальних операцій, припустимих у період асемблювання. Зазначені також рівні їх пріоритетів. Операції, які мають рівень пріоритетів 7, повинні бути виконані в першу чергу. Для зміни порядку виконання обчислювальних операцій можуть використовуватися круглі дужки. Обчислення здійснюються за допомогою 80-і розрядних цілих чисел за винятком операцій піднесення у степінь, в яких використовується 8-і розрядне значення показника степеня. Максимальна кількість незавершених операцій дорівнює 16.

Таблиця 6.1

Перелік обчислювальний операцій при асемблюванні програм

Операція	Пріоритет	Опис
Унарний +	7	Може задавати позитивний операнд
Унарний -	7	Змінює знак наступного виразу
\ або .NOT.	7	Інвертує (доповнює) наступний вираз
Унарний >	7	Зберігає старший байт наступної адреси. Повинна використовуватися для отримання значень адрес байтів, що переміщаються

Закінчення табл. 6.1

Унарний <	7	Зберігає молодший байт наступної адреси. Повинна використовуватися для отримання значень адрес байтів, що переміщаються
**	6	Беззнакове піднесення у степінь
*	5	Беззнакове перемножування
/	5	Беззнакове ділення
.MOD.	5	Обчислення залишку
.SHR.	5	Зрушення попереднього виразу праворуч (із заповненням нулями) на кількість позицій, заданих наступним виразом
.SHL.	5	Зрушення попереднього виразу ліворуч (із заповненням нулями) на кількість позицій, заданих наступним виразом
+	4	Додавання
-	4	Віднімання
&або.AND.	3	Логічне І
або.OR.	2	Логічне АБО
.XOR.	2	Логічне, що виключає АБО

6.5. Порівняння під час транслювання

У поданому далі списку наводяться операції порівняння періоду асемблювання, що повертають значення «всі одиниці» (якщо висловлення є істинним) або «всі нулі» (якщо висловлення є неправильним).

Таблиця 6.2

Операції порівняння періоду асемблювання програм

Операція	Опис
= або .EQ.	Дорівнює
> або .GT.	Більше ніж
< або .LT.	Менше ніж
.UGT.	Беззнакове більше ніж
.ULT.	Беззнакове менше ніж

6.6. Повідомлення про помилки асемблювання

SYNTAX ERROR. (Синтаксична помилка). Звичайно ця помилка виникає через пропущену кому або круглу дужку.
CAN'T RESOLVE OPERAND. (Неможлива ідентифікація звертання до операнду). Незрозуміло, що в цьому випадку передбачалося програмістом.
ILLEGAL ADDRESSING MODE. (Неправильний режим адресації). Адресація операнда з використанням даної форми адресації неприпустима.
ILLEGAL ARGUMENT. (Неправильний аргумент). У даному контексті операнд не може бути використаний.
MULTIPLY DEFINED SYMBOL. (Символ уже описаний). Даний символ уже описаний раніше (не включаючи '.VAR').
ILLEGAL MNEMONIC. (Неправильне мнемонічне позначення). Дане мнемонічне позначення не існує і не було використано для завдання макросу.

TOO LARGE. (Число занадто велике). Результат занадто великий для даного операнда.
ILLEGAL ASCII DESIGNATOR. (Неправильний код таблиці ASCII). Неправильний символ пунктуації або код таблиці Ascii.
HEX # AND SYMBOL ARE IDENTICAL. (Шістнадцяткове число й символ є ідентичними). Існує деяка мітка, яка в точності ідентична шістнадцятковому числу, використовуваному в якості операнда. Для того, щоб виникла ця помилка, навіть індикатор шістнадцяткового числа повинен бути на тій самій позиції.
UNDEFINED SYMBOL. (Символ не визначено). Символ не був описаний у період виконання першого проходу асемблера.
RELATIVE JUMP TOO LARGE. (Занадто далекий відносний перехід). Результуюча адреса переходу перебуває на іншій сторінці.
EXTRA CHARACTERS AT END OF OPERAND. (Наприкінці операнда присутні зайві символи). Звичайне виникнення цього повідомлення пов'язано з наявністю синтаксичної помилки або помилки формату.
Примітка: дана помилка виникає при останній перевірці якої-небудь команди перед тим, як Асемблер переходить до наступного рядка, і вона позначає, що після припустимого символу завершення операнда знаходяться зайві символи.
LABEL VALUE CHANGED BETWEEN PASSES. (Значення мітки змінилося між проходами Асемблера). Значення символу (ідентифікатора), декодоване в період виконання проходу 1, не дорівнюється виявленому при виконанні проходу 2.
Примітка: ця помилка звичайно виникає в тому випадку, коли Асемблер обробляє інші частини програми при виконанні проходу 1 у порівнянні із проходом 2 через зміну значень аргументів директив умовної компіляції. Для виявлення подібних типів помилок може виявитися корисною директива PASS1 ON/OFF.
ATTEMPTED DIVISION BY ZERO. (Почата спроба ділення на нуль). При виконанні ділення дільник дорівнює 0.
ILLEGAL EXTERNAL REFERENCE. (Неправильне зовнішнє посилання). У даному контексті зовнішнє посилання використовуватися не може.
NESTED CONDITIONAL ASSEMBLY UNBALANCE DETECTED. (Виявлено дисбаланс при аналізі вкладених структур асемблера). Виявлено яку-небудь директиву класу '.IF', котрий не відповідає парна директива '.ENDIF'.
ILLEGAL REGISTER. (Неприпустимо використання регістру). Для даної команди неприпустимо використання зазначеного регістру.
CANT RECOGNIZE NUMBER BASE. (Неможливо визначити основу системи числення числа). Задана основа системи числення числа не допускається асемблером.
NOT ENOUGH PARAMETERS. (Не вистачає параметрів). У макросі число аргументів перевищує число параметрів.
ILLEGAL LABEL 1ST CHARACTER. (Неправильний перший символ мітки). Мітка повинна починатися з алфавітного символу.

MAXIMUM EXTERNAL SYMBOL COUNT EXCEEDED. (Перевищено максимальне число зовнішніх символів). У модулі було задано занадто велике число зовнішніх символів (ідентифікаторів).
Примітка: у модулі припустимо приблизно 500 зовнішніх символів (ідентифікаторів).
MUST BE IN SAME SECTION. (Повинен перебувати в тій самій секції). Операнд команди перебуває в іншій секції.
NON-EXISTENT INCLUDE FILE. (файл, що включається, не існує). Файл не може бути знайдений.
ILLEGAL NESTED INCLUDE. (Неправильно вкладене увімкнення файлів). Один файл, що увімкнено, містить директиву .INCLUDE. Дане повідомлення про помилку може також зазначати на відсутність оператора END у якому-небудь файлі, що увімкнено.
NESTED SECTION UNBALANCE. (Дисбаланс вкладених секцій). В описі вкладеної секції відсутня ENDS.
MISSING DELIMITERS ON MACRO CALL LINE. (У рядку виклику макросу пропущені обмежники). Виявлена розбіжність обмежників під час виклику макросу.
MULTIPLE EXTERNALS IN THE SAME OPERAND. (У одному й тому самому операнді виявлена множина зовнішніх символів (ідентифікаторів). У одному й тому самому операнді існує більш одного зовнішнього символу (ідентифікатора)).
A LABEL IS ILLEGAL ON THIS INSTRUCTION. (Мітка не може бути застосована до даної команди). Дане повідомлення про помилку використовується для того, щоб позначити мітки, що не одержують переміщуваного значення, такі як ENDM або MACEND. Використання цих міток неприпустимо для даної команди.
MACRO STACK OVERFLOW. (Переповнення стека макросів). Використано занадто великий ступінь вкладення макросів.
Примітка: дана помилка може бути викликана занадто великою кількістю рекурсивних викликів макросів. Стек допускає використання приблизне 700 вкладених або рекурсивних викликів макросів. На кількість макросів впливає число аргументів, які використовуються в макросі.
MISSING LABEL. (Пропущена мітка). У даній команді потрібна присутність мітки.
OPERAND MUST BE DEFINED AS AN 8 BIT RELOCATABLE VALUE. (Операнд може бути описаний як 8-бітове переміщуване значення). Це повідомлення про помилку виникає, коли 16-бітова адреса використовується в команді, що допускає 8-бітовий операнд. Для того щоб зробити це значення переміщуваним, необхідно використовувати знак <або>.
MISSING RIGHT ANGLE BRACKET. (Пропущена права кутова дужка). Права кутова дужка є обов'язковою.
MACRO NAME MUST APPEAR ON SAME LINE AS MACRO DEFINITION. (Ім'я макросу повинно з'являтися на тому самому рядку, що й опис макросу).

ILLEGAL LOCAL LABELS. (Неприпустиме використання локальних міток). Мітки не можуть бути описані як локальні. Наприклад, .VAR.
MISSING MODULE DIRECTIVE. (Пропущена директива опису модуля MODULE).
MISSING ENDMOD DIRECTIVE. (Пропущена директива ENDMOD (задання кінця модуля)).
'Module' CAN'T BE IN 'Include' FILE. (Директива 'Module' не може перебувати у 'приєднувальному' файлі).
'Endmod' CAN'T BE IN 'Include' FILE. (Директива 'Endmod' не може перебувати у 'приєднувальному' файлі).

6.7. Методика роботи з компілятором x8051

Компілятор X8051 може працювати в діалоговому режимі та режимі командного рядка.

6.7.1. Діалоговий режим

Для виклику Асемблера необхідно завантажити x8051.exe. Асемблер у відповідь запросить:
Listing Destination?(N,T,D,E,L,<CR>=N).
Де абревіатури означають наступне:
N друку немає;
T термінал;
P принтер;
D диск;
E тільки помилки;
L друкування увімкн./вимкн.

Потім Асемблер запросить ім'я файлу, що містить вихідні коди:

Input filename:

При вводі імені файлу можна вилучити розширення, якщо воно asm. Далі Асемблер запросить ім'я вихідного файлу:

Output filename:

Якщо користувач відповість поверненням каретки, ім'я вихідного файлу буде утворено з імені вхідного з розширенням obj. Якщо ім'я файлу у відповіді не буде містити розширення, то воно дорівнює obj.

Якщо видача лістингу йде під керуванням директиви асемблера LIST ON/OFF, то виникає додатковий запит:
LIST ON/OFF Listing Destination (T,P,D,<CR>=T):
Скорочення відповідають попереднім.

6.7.2. Режим командного рядка

Асемблер може сприймати командний рядок. У цьому випадку ім'я вхідного файлу визначається першим, додатково може йти ім'я вихідного файлу й список опцій. Загальна форма команди (необов'язкові поля показані у квадратних дужках) буде наступною:

asm8051[-q]input_filename[output_filename][-t,-p,-d,-px,-dx]
Якщо введена опція -q, на екран виводяться тільки повідомлення про помилки й відповідні рядки. Ця опція повинна передувати імені файлу:

-t вивід на термінал
-p вивід на принтер
-x вивід лістингу з таблицею перехресних посилань
-d вивід на диск
-e вивід тільки повідомлень про помилки
-l вивід блоків, що позначено у тексті LIST ON/OFF.

6.8. Редактор зв'язків для компілятора мікроконтролера сімейства МК51

Далі під лінкером скрізь мається на увазі програма редактора зв'язків. Лінкер дозволяє користувачеві записати програму мови асемблера, що містить кілька програмних модулів. Лінкер ураховує зовнішні посилання й виконує розміщення в адресному просторі. Він здатен створювати вихідні файли для всіх найбільш застосовуваних форматів.

Лінкер може бути викликаний у діалоговому, командному режимах або під керуванням з файлу. Вихідний формат вибирається директивою у вихідному файлі або в параметрах команди LINK. Повний опис можливостей і методики роботи з лінкером викладається в керівництві обсягом 38 сторінок, тому для практичної роботи розглядається тільки режим командного рядка.

РЕЖИМ КОМАНДНОГО РЯДКА.

Лінкер може бути викликаний у командному рядку. Формат такої команди показано нижче з наступним розшифруванням елементів:
Drive:\Path\Link.exe [-q]-c file1[-Innnn]file2[-Innnn]...[-ofile][-Llibfile][-options]

-q Лінкер у режимі Quit. У цьому випадку видається тільки повідомлення про помилки на консоль.
-c Потрібен для вказівки, що буде використано режим командного рядка, а не керуючого файлу. Слідом за ключем -I йде список вхідних файлів, що складається (для наведеного рядка) з файлів fil1 і fil2. За кожним файлом може випливати адреса зсуву секцій, що починається з -I. Якщо ця адреса відсутня, то поточна секція є продовженням попередньої з тим самим іменем.
-o Використовується для вказівки вихідного файлу. Цей елемент не обов'язковий. Якщо він відсутній, лінкер створить вихідний файл із тим самим іменем, що й перший вхідний, та з розширенням, обумовленим типом формату, що генерується.
-L Використовується для задання бібліотек. Максимум може бути зазначено 50 бібліотек.
-options. Визначає додаткові параметри. Знак мінус потрібен на початку списку, і в нього може входити стільки параметрів, скільки необхідно (детально це описано в «Параметри Лінкера»).

ПАРАМЕТРИ ЛІНКЕРА:

Параметри вказуються в діалоговому режимі після імені бібліотечних файлів. Вони можуть бути задані й у режимі керування з файлу й у командній Mode. Коли зазначено кілька конкуруючих параметрів, останній скасовує дію попереднього.

D Створити дисковий файл, що містить усі помилки лінкування, символьну таблицю глобальних змінних і карту пам'яті (MAP) завантаження. Файл має те саме ім'я, що й вихідний файл, але з розширенням map.

S Створити символьний файл для процесу налагодження. Цей файл містить усі глобальні символи і їх величини. Кожний символ має у довжину 32 букви.

A Створити символьний файл, але із символами в 10 букв. Він використовується для сумісності з лінкером 2500 A.D. версії 3.0.

M Створити символьний файл із метою налагодження у форматі Microtek. Він містить усі символи як глобальні, так і локальні. Для того, щоб цей формат міг бути створений, у вихідному файлі повинна бути присутня директива SYMBOLS ON.

Z Створити символьний файл із метою налагодження у форматі ZAX. Цей файл містить локальні й глобальні змінні. Для того, щоб цей формат міг бути створений, у вихідному файлі повинна бути присутня директива SYMBOLS ON.

X Створити виконуваний вихідний файл.

H Створити шістнадцятковий файл формату Intel.

E Створити шістнадцятковий файл розширеного формату Intel.

T Створити шістнадцятковий файл формату Tektronix.

1 Створити вихідний файл формату S19 фірми Motorola.

2 Створити вихідний файл формату S28 фірми Motorola.

3 Створити вихідний файл формату S37 фірми Motorola.

ПРИКЛАД. Є об'єктний файл first.obj, необхідно виконати процес лінкування зі створенням виконуваного (у двійковому коді) вихідного файлу.

Для цього необхідно виконати наступну команду:

Drive:\Path\link.exe -c first.obj -x

При цьому буде створено вихідний файл із розширенням «tsk» (у нашім прикладі first.tsk), код програми буде розташовано з адреси 0000H (за замовчуванням).

7. ПРИКЛАДИ ВВОДУ ІНФОРМАЦІЇ З ДИСКРЕТНИХ ДАТЧИКІВ

У реальних системах керування майже завжди присутні всілякі контактні дискретні датчики, кнопки, органи блокування і т.п. Схеми підключення датчиків до МК51 наведено на рис 7.1, а. Ввід інформації із цих датчиків має свої особливості.

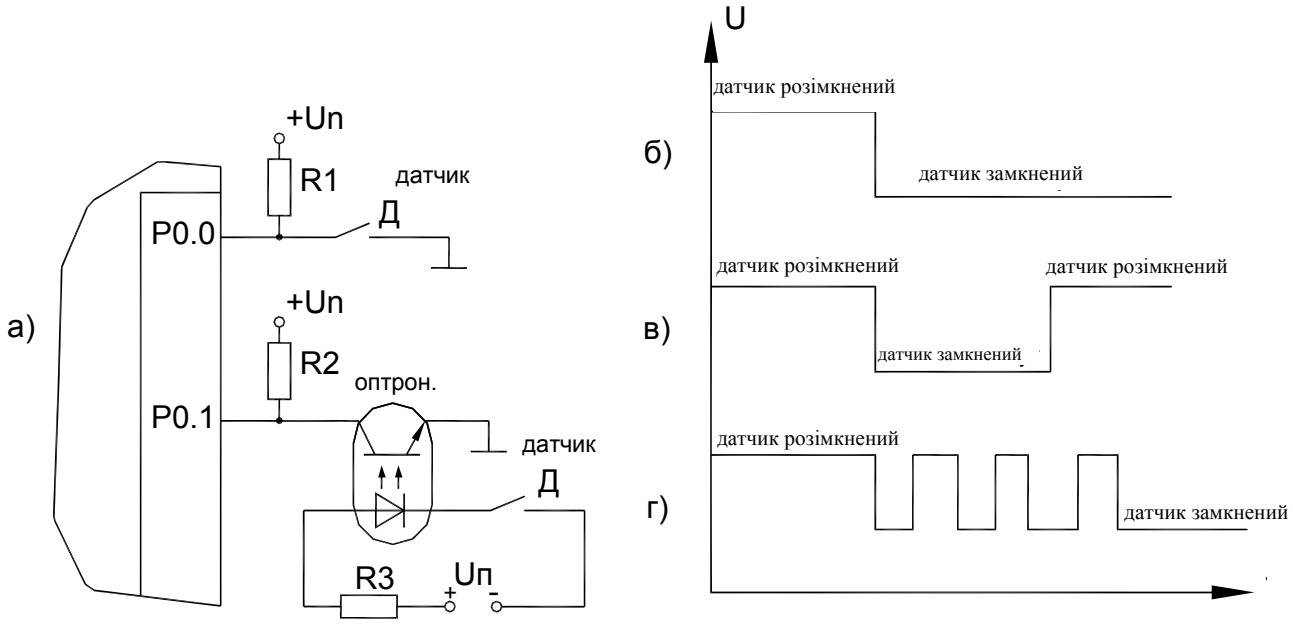


Рис. 7.1. Схеми підключення дискретних датчиків

По-перше, сигнал при спрацьовуванні може бути статичним, тобто при спрацьовуванні датчика формується тільки перехід з 1 в 0, як зображено на рис. 7.1, б. Але може формуватися й імпульсний сигнал (рис. 7.1, в). Крім того, при спрацьовуванні таких датчиків виникає деренчання контакту. При замиканні контакту виникає відскік контактів і в систему може бути введена випадкова послідовність нулів і одиниць (рис. 7.1, г).

Причому датчик може бути приєднаний до входів T0 або T1 порту P3 або до входів портів P0…P3.

При приєднанні до входів T0 або T1 здійснюється контроль тривалості спрацьовування датчика з використанням таймерів T0 або T1 у режимі 0 або режимі 1.

Контроль стану датчика при приєднанні до входів портів P0…P2 і сигналі, як зображено на рис. 7.1, а, на виході датчика можна здійснити всього однією командою, наприклад, JB P1.4, мітка; мітка переводить мікропроцесор на виконання підпрограми, яка здійснює команди операції, які відповідають замкненому стану контакту. Схему алгоритму опитування стану датчика наведено на рис. 7.2, а.

Якщо спостерігається деренчання контакту, то усунути його можна шляхом вводу в схему алгоритму лічильника заданого числа станів, що збігаються (рис. 7.2, б), або шляхом вводу тимчасової затримки (рис. 7.2, в).

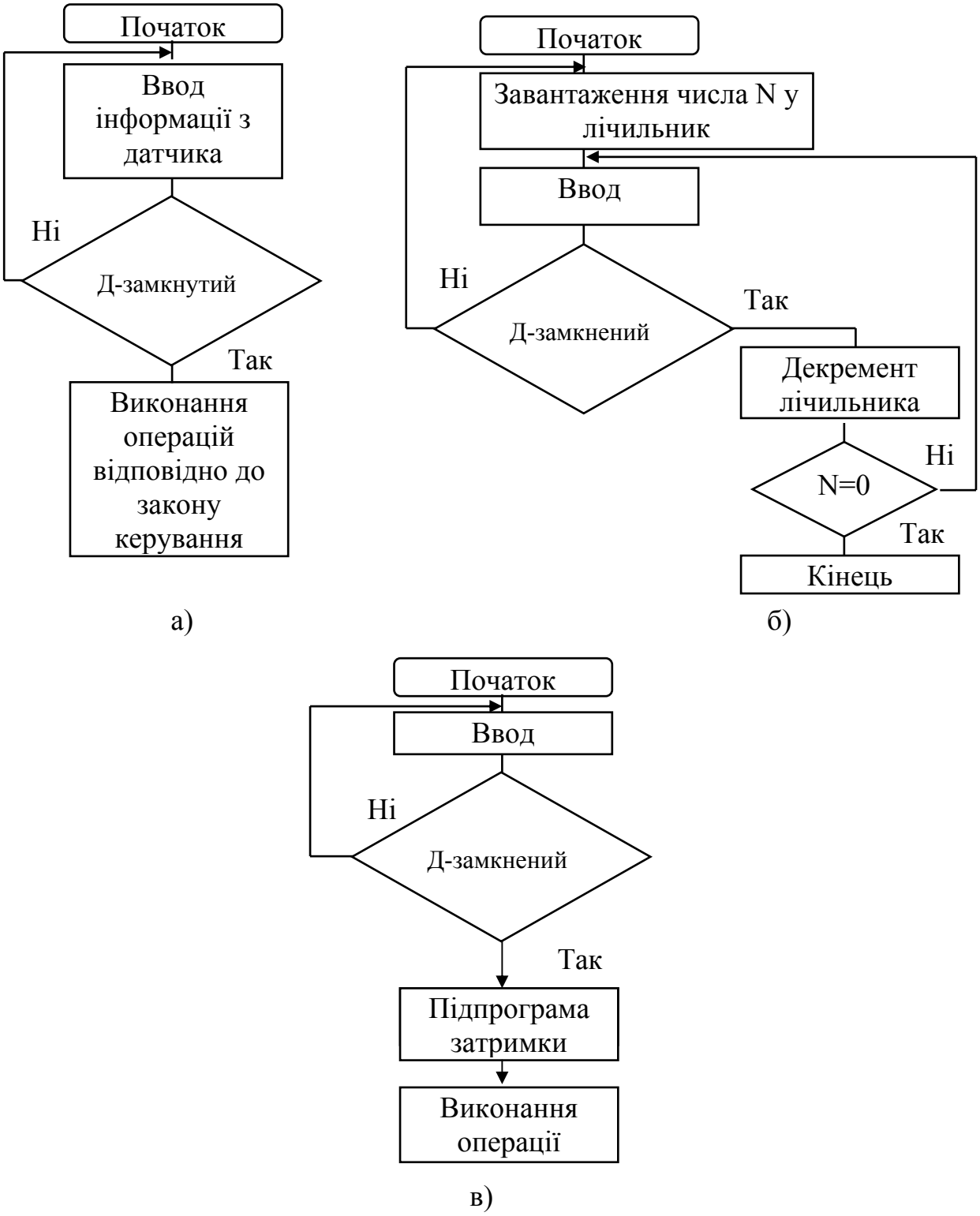


Рис. 7.2. Схема алгоритму аналізу інформації з дискретного датчика:
а) опитування датчика; б) з використанням лічильника; в) з використанням часової затримки

Розглянемо приклад розробки мікропроцесорної системи керування шахтною водовідливною установкою. Вона містить основний насос, заливальний насос, засувку основного ставу, датчики верхнього й нижнього рівнів (рис. 7.3). Відкачка води починається, якщо рівень води у водозбірнику

досягне значення датчика верхнього рівня (ДВ), вимкнутись насос повинен, якщо рівень води у водозбірнику досяг значення нижче датчика нижнього (ДН) рівня. Перед увімкненням основний насос повинен бути заповнений водою за допомогою заливального насоса. Тривалість заливання основного насоса – 30 секунд. Перед увімкненням основного насоса повинна бути відкрита засувка водонапорного ставу. Таким чином, відповідно до завдання, насосна установка містить два датчики рівня, два насоси, яким надають руху асинхронні двигуни з короткозамкненим ротором, засувку водонапорного ставу (рис. 7.3).

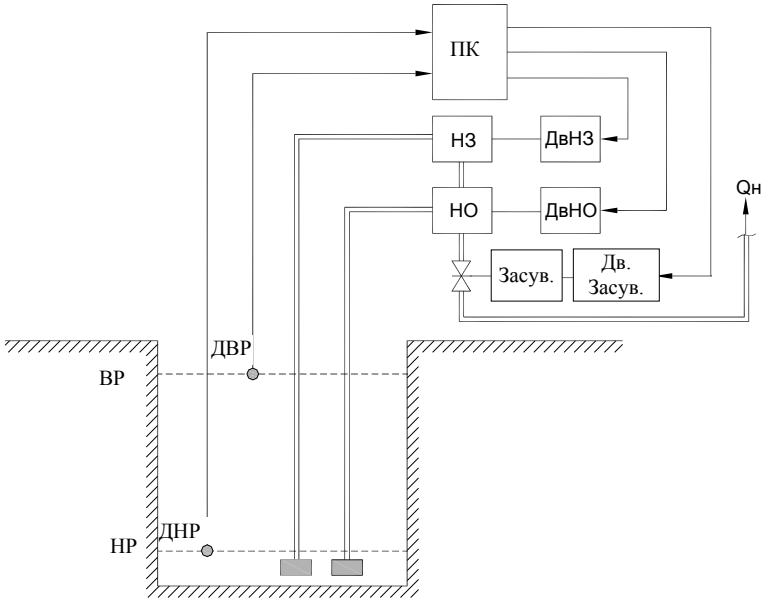


Рис. 7.3. Структурна схема системи керування водовідливом

Завдання конкретне, викладено в текстовому вигляді, з якого випливає, що до МК51 необхідно підключити датчики верхнього й нижнього (ДВ, ДН) рівнів і три котушки пускачів для увімкнення приводів заливального, основного насосів і засувки. Датчики рівня на виході мають сухий контакт, а напруга живлення котушок пускачів становить 12 вольт. Виходячи із цього, функціональна схема мікропроцесорної системи керування буде мати вигляд, який наведено на рис. 7.4.

Алгоритм функціонування системи зобразимо у вигляді графа станів системи (рис. 7.5). Система відповідно до умов перебуває у вихідному (А0) стані доти, поки рівень води не досягне значення верхнього рівня. Після цього система переходить у стан (А1). У цьому стані вмикається заливальний насос і запускається затримка 30 секунд. Після закінчення затримки система переходить у стан (А2). У цьому стані вимикається заливальний насос, вмикається привід засувки на відкривання (вимкнення приводу засувки виконує локальна автоматика) і вмикається основний насос. У стані А2 система буде перебувати доти, поки рівень води не опуститься нижче датчика нижнього рівня (воду відкачено). У цьому випадку система перейде у вихідний стан (А0), при цьому вимкнеться основний насос, закриється засувка водонапорного ставу. Далі процеси повторюються.

Після складання графа станів системи привласнимо символічні імена змінним:

datn – датчик нижнього рівня;
datv – датчик верхнього рівня;
no – основний насос;
nz – заливальний насос;
zadv – засувка водонапорного ставу;
zd – затримка;
state – стан системи;
prescal – переддільник для організації інтервалу в 1 секунду.

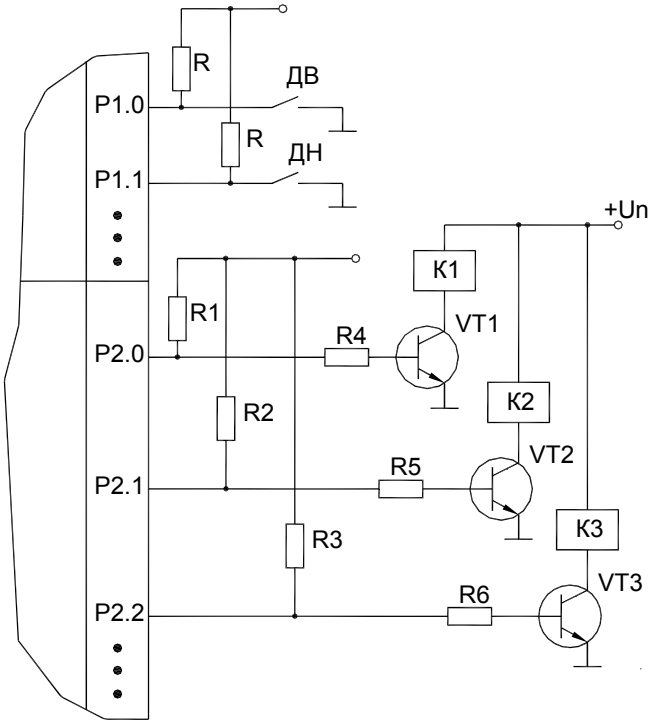


Рис. 7.4. Функціональна схема мікропроцесорної системи керування водовідливом

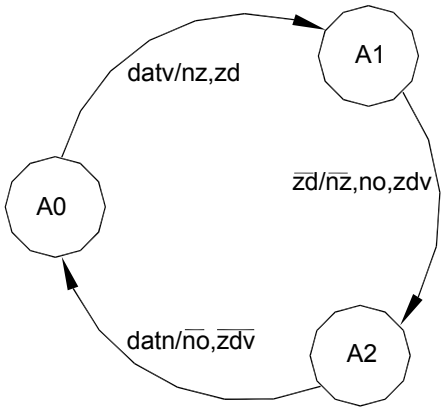


Рис. 7.5. Граф стану системи керування водовідливом

Після цього визначимо адреси ОЗУ, де будуть перебувати ці змінні:

```
datn      equ 81h ; адреса датчика нижнього рівня,
datv      equ 80h ; адреса датчика верхнього рівня,
no        equ 90h ; адреса основного насоса,
nz        equ 91h ; адреса насоса, що заливає основний насос,
zd        equ 31h ; адреса комірки формування затримки,
zadv      equ 32h; адреса засувки водонапорного ставу;
state     equ 40h ; адреса комірки стану системи,
prescal   equ 30h ; адреса комірки переддільника.
```

Текст програми, що відповідає графу станів системи (рис. 7.5), наведено нижче:

```
        ljmp m0;
        org 0bh;
```

```
        ljmp tim0
m0:     clr no;переводимо вихідний стан системи в початковий скиданням до нуля
        clr     nz ; змінних nz, no, zadv,state
        clr     no
        clr     zadv
        mov   state,#0 ;
        mov   sp,#70h ; призначаємо початкову адресу стекової пам'яті,
        mov   tmod,#1h ; установлюємо режим роботи Таймера 0,
        mov   th0,#3ch ; заносимо в Таймер 0 стартове число, яке
        mov   tl0,#0b0h ; визначає величину затримки 50 мілісекунд,
        setb tcon.4 ; запуск таймера (установка біта TR у регістрі TCON)
        setb ie.7         ; дозвіл переривань
        setb ie.1          ; дозвіл переривань від Таймера 0
        mov prescal,#20 ; коефіцієнт множення затримки 50 мілісекунд,
loop: mov a,state   ; основний цикл системи,
        cjne a,#0,st1; перевірка стану системи, якщо 0, то перехід до аналізу стану
        ;датчика ДВ, якщо не нуль, то перехід на мітку st1,
        jnb datv,loop ; перевірка стану датчика верхнього рівня, якщо спрацював,
        ;запускаємо затримку, якщо немає вертаємося у вихідний стан,
        mov zd,#1eh ; встановлення змінної zd значення 30 (секунд),
        setb nz         ; увімкнути заливальний насос,
        mov state1,#1 ; фіксація переходу системи в стан a1,
        ajmp loop ; перехід на початок циклу,
st1:    cjne a,#1,st2 ; перевірка стану системи,
        mov a,zd1; перевірка умови закінчення затримки,
        jnz loop;
        clr nzo; вимкнути заливальний насос,
        setb no ; увімкнути основний насос
        mov state,#2 ;зафіксувати перехід системи в стан 2
        ajmp,loop; перехід на початок циклу,
st2:    jb datn,loop ; перевірка стану датчика нижнього рівня,
        clr no; вимкнути основний насос,
        clr zadv; закрити засувку водонапорного ставу,
        mov state,#0 ; зафіксувати перехід системи в стан 0,
        ajmp loop ;
tim0: push acc;
        mov th0,#3ch ; перезавантаження таймера (50 мілісекунд)
        mov tl0,#0b0h ;
        djnz prescal,m1; перевірка затримки 1 секунда,
        mov prescal,#20 ; перезавантаження переддільника числом 20,
        mov a,zd; пересилання змінної zd в акумулятор,
        jz, m1 ; перевірка закінчення затримки, якщо ні,
        dec zd; декрементуємо змінну zd,
 m1:  pop acc ; відновлюємо вміст акумулятора,
        reti; вихід з підпрограми переривання.
```

8. ОРГАНІЗАЦІЯ ПЕРЕДАЧІ ІНФОРМАЦІЇ В ПОСЛІДОВНОМУ ФОРМАТІ В МІКРОПРОЦЕСОРНИХ СИСТЕМАХ (МПС)

Сучасні МПС мають багаторівневу структуру, де виникає завдання обміну інформацією не тільки між різними рівнями, але й між окремими мікроконтролерами на одному рівні. Крім того, у системах, як правило, передбачається зв'язок з персональним комп'ютером (ПК), який виконує реєстрацію (моніторинг) основних параметрів процесу керування. При цьому всі мікроконтролери об'єднані в локальну керуючо-обчислювальну мережу, через яку й здійснюється обмін інформацією між мікроконтролерами.

8.1. Принципи передачі інформації з послідовного каналу зв'язку

Обмін даними між пристроями у мікроконтролері відбувається по паралельній шині. Вона складається із провідників, які з'єднують пристрої. При цьому по кожному провіднику передається окремий біт, у такий спосіб шина даних являє собою вісім провідників, на кожний з яких подається від пристроїв значення бітів від D0 до D7.

Такий спосіб обміну виправдовує себе, коли обмін відбувається на невеликі відстані, наприклад, усередині одного пристрою або мікросхеми. Якщо є необхідність передавати інформацію між пристроями на значні відстані, такий спосіб неефективний через значну вартість багатопровідних шин. Тому обмін з вилученими пристроями, як правило, проводиться за дво- або трипровідною схемою. При цьому використовується так званий спосіб тимчасового ущільнення каналу. Його суть показано на рис. 8.1.

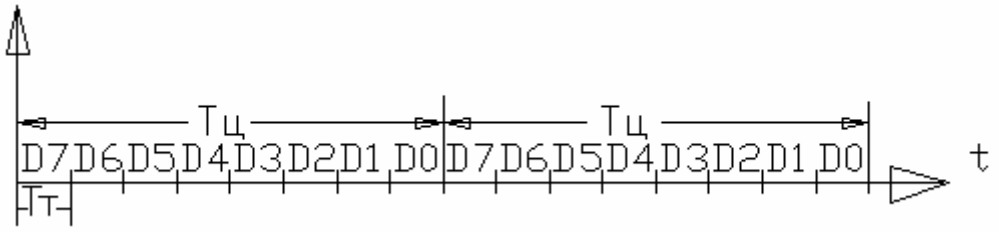


Рис. 8.1. Тимчасове ущільнення каналу зв'язку

У цьому випадку інформація передається по двох проводах – по сигнальному відповідно до загального.

На сигнальний провід подається послідовно в часі стан розрядів шини даних, починаючи з D0 по D7. Значення кожного розряду втримується на сигнальному проводі протягом часу t_u. Таким чином, відбувається перетворення даних з паралельного формату в послідовний, яке здійснюється за допомогою зсувного регістру й тактового генератора. Структурна схема цього пристрою показана на рис. 8.2.

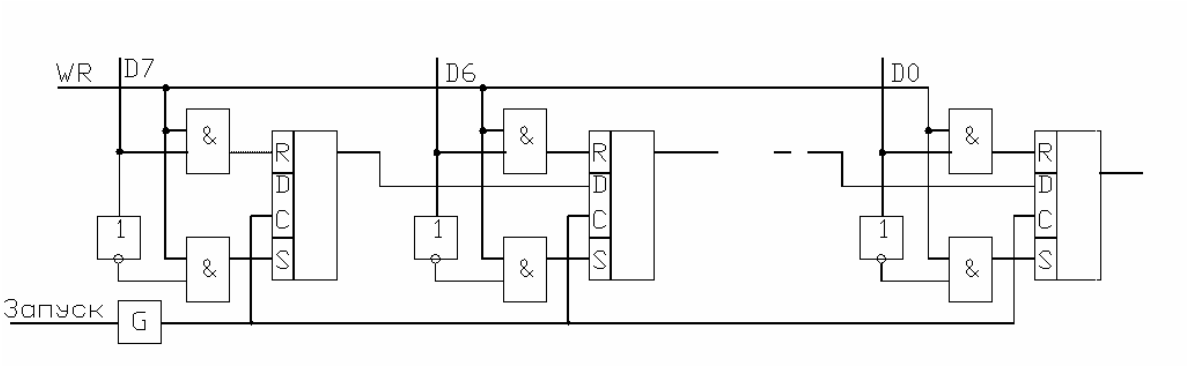


Рис. 8.2. Схема зсувного регістру з паралельним занесенням даних

Він складається зі зсувного регістру, виконаного на D-Тригерах, схем занесення інформації в тригери (D0-D7) (паралельний формат) й генератора синхроімпульсів G.

Після занесення інформації в тригери запускається генератор. На останньому тригері послідовно в часі формується значення D0, за другим імпульсом генератора – D1 і т.п.

Час утримання значень бітів на виході визначається періодом генератора.

Під час такого способу передачі виникають наступні проблеми: синхронізація прийнятих даних за тактами, байтами і повідомленнями. Синхронізація за тактами передбачає, що фіксація інформації на приймальній стороні відбувається в моменти часу, коли на виході лінії зв'язку встановлюється значення переданого біта (після закінчення перехідного процесу). Найбільш простим способом такої синхронізації є використання ще одного сигнального провідника для синхронізації приймального пристрою. Структуру такої передачі зображено на рис. 8.3.

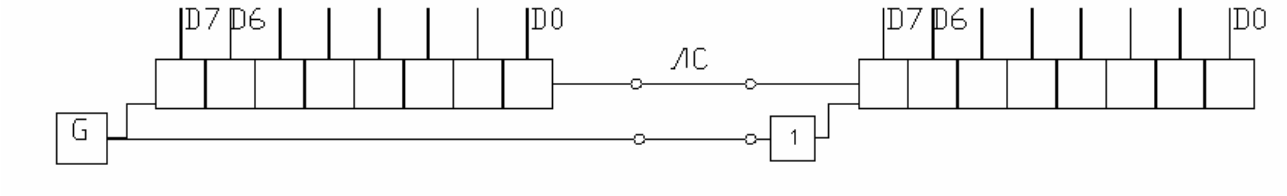


Рис. 8.3. Передача інформації з додатковим сигналом синхронізації

Такий спосіб дозволяє передавати інформацію з максимальними швидкостями передачі. Однак наявність додаткового провідника обмежує область застосування цього способу передачі. Він використовується частіше для обміну інформацією з периферійними пристроями в межах одного апарата, наприклад, для зв'язку з АЦП, ЦАП, ЖК індикаторами, флеш-пам'яттю і т.п.

Осцилограма цього способу передачі зображена на рис. 8.4.

Інвертування тактового сигналу приймача приводить до того, що підтвердження значення в ЛС проводиться у середині бітового інтервалу, при значенні сигналу, що вже встановилось.

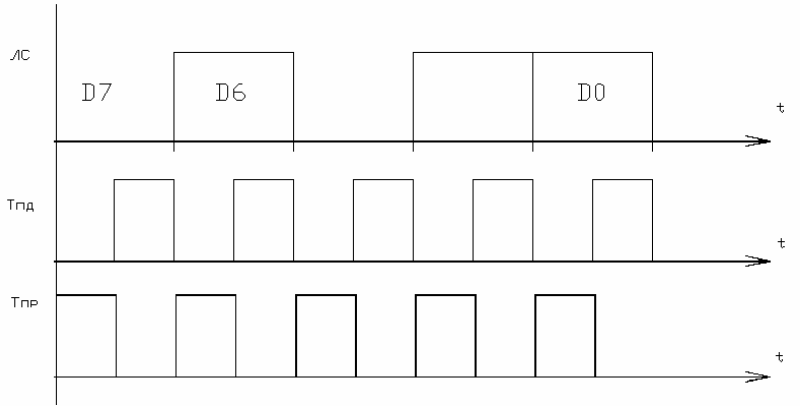


Рис. 8.4. Осцилограма обміну інформацією при використанні додаткового тактового сигналу

У більш досконалих системах приймач синхронізується від власного генератора, але, тому що генератори розділені, їх необхідно синхронізувати. Для цього в схему вводиться додатковий пристрій, який синхронізує фазу приймального генератора за фронтами і зрізами, які виділяються із приймального сигналу. Структурна схема такої системи передачі наведена на рис. 8.5.

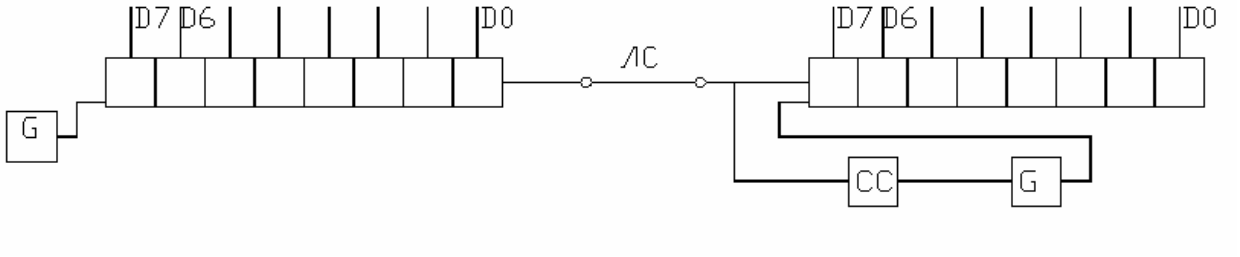


Рис. 8.5. Обмін інформацією з незалежним тактуванням

8.2. Послідовний інтерфейс у мікроконтролері MCS51

Для обміну інформацією в послідовному із зовнішніми пристроями форматі до складу мікроконтролера належить послідовний інтерфейс або універсальний синхронно-асинхронний приймач-передавач УСАПП. До його складу належать: зсувні регістри, що приймають й передають, буфери приймача й передавача, а також схеми тактування передавача й синхронізації приймача. Наявність додаткових буферів дозволяє суміщати операції зчитування вже прийнятого байта із прийманням чергового.

Послідовний інтерфейс може працювати за чотирма режимами і настроюється на відповідний за допомогою запису керуючого слова в регістрі SCON. Призначення бітів регістру керування наведено в табл. 3.2.

Для обміну інформацією із зовнішніми пристроями УАПП використовують сигнали Rxd і Txd – альтернативне призначення бітів порту 3 розряди 0 і 1 відповідно. Залежно від режиму роботи УСАПП призначення цих розрядів різне.

8.3. Режим роботи 0

Режим 0 виконує завдання синхронного обміну. Під час роботи в цьому режимі застосовується додатковий провід тактування для синхронізації приймача зовнішнього пристрою. Схему підключення зовнішнього пристрою зображено на рис. 8.6.

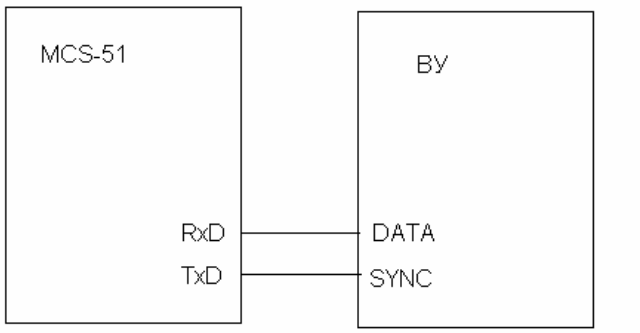


Рис. 8.6. Схема підключення периферійних пристроїв
до контролера в режимі 0

Передача й приймання інформації здійснюється через вхід Rxd, імпульси синхронізації – через Txd. Обміну підлягають 8-бітові дані. Мікроконтролер у цьому режимі є ведучим. Він ініціює обмін, здійснюючи початкову передачу даних. Якщо потрібно, передає запит на приймання інформації, після чого переходить у режим приймання. Потім формує тактові імпульси для зовнішнього пристрою й ухвалює призначену для нього інформацію. Швидкість обміну при цьому фіксована й дорівнює 1/12 Fosc. Для ініціювання передачі необхідно занести до регістра SCON у розряди MSO, MS1 нулі й записати в SBUF перший байт повідомлення. Наступний байт може бути записано в SBUF тільки після передачі байта. Прапор закінчення передачі перебуває в біті SCON.1 (TI), формується апаратно після закінчення передачі байта. Він може викликати переривання за адресою 23H. Після закінчення обробки переривання прапор необхідно скинути програмно. Допускається опитування прапора без використання переривання. Після встановлення прапора TI в SBUF можна записати наступний біт повідомлення. Якщо було передано останній байт повідомлення, досить скинути прапор TI і більше він формуватися не буде. Для поновлення передачі необхідно в SBUF записати перший байт повідомлення й процес передачі повторюється.

Для переведення УСАПП у режим приймання потрібно встановити в SCON.4 одиницю.

Режим 0 призначено для обміну інформацією із пристроями, що розташовані на платі контролера, при дотриманні правил перешкодозахисту й мінімальній довжині провідників. Він забезпечує максимальну швидкість обміну по послідовному каналу.

Режим роботи 1.

Режим асинхронного обміну найбільш часто застосовується. Це пояснюється тим, що формат обміну даними у цьому режимі такий самий, як і у послідовному інтерфейсі персональних комп'ютерів (COM).

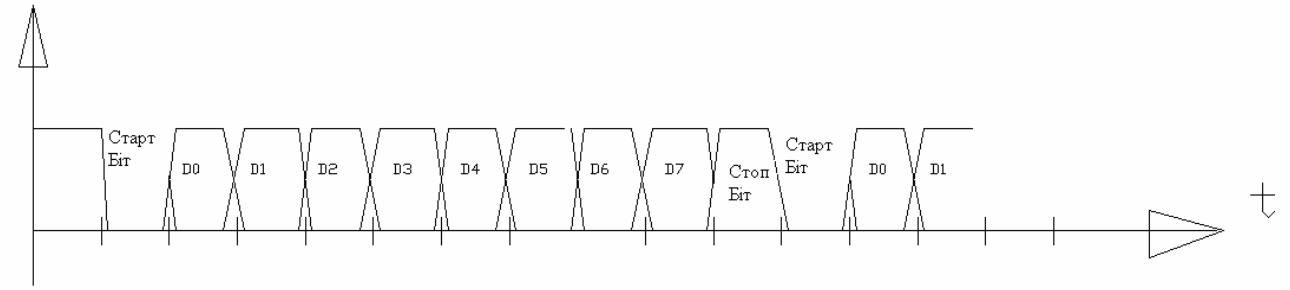


Рис. 8.7. Структура посилки в режимі 1

Структуру посилки в режимі 1 зображено на рис. 8.7. У посилці передаються 8-бітові дані, які доповнюються 2 службовими бітами: старт-біт (передається на початку посилки) і стоп-біт (передається наприкінці посилки).

Передача даних проводиться через вихід Txd. Приймання даних – через вхід Rxd.

Якщо передача в режимі 1 відсутня, то вихід Txd перебуває в стані 1. Після занесення даних в SBUF передавача, до лінії зв'язку передається старт-біт, за ним 8 бітів даних, а потім стоп-біт. Старт-біт передається нульовим, а стоп-біт одиничним рівнем. Таким чином, навіть якщо серед даних передаються всі 0 або всі 1, у повідомленні можна виділити початок і кінець передачі байта.

Структурну схему УСАПП у режимі 1 зображено на рис. 8.8.

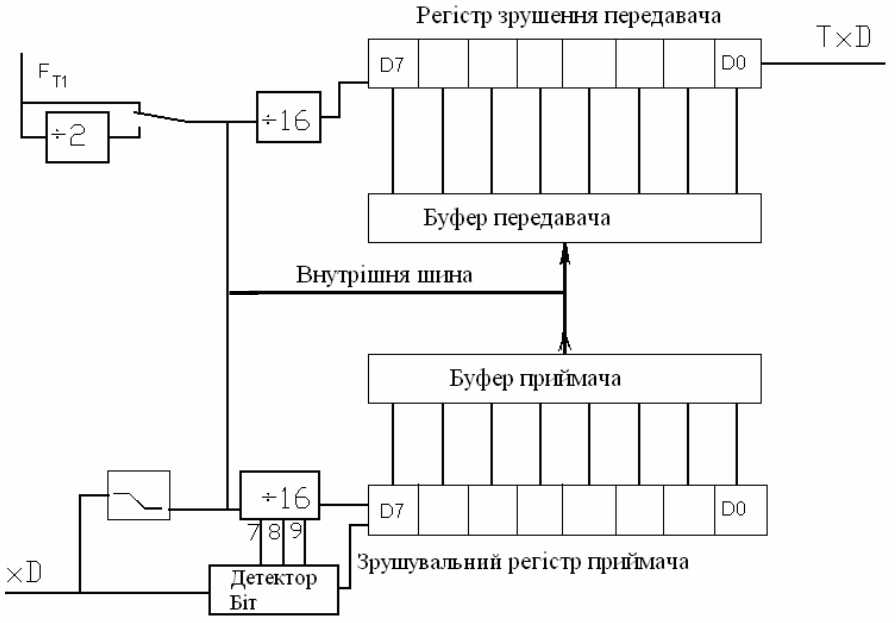


Рис. 8.8. Структурна схема УСАПП у режимі 1

Тактування приймання й передачі в режимі 1 здійснюється від таймера 1, що працює в режимі 2 (автоматичне перезавантаження). Швидкість передачі визначається значенням константи, що завантажується в TH1.

На тактування зсувного регістра передавача частота з виходу таймера надходить через дільник на 16.

У режимі 1 відсутній сигнал зовнішнього тактування приймача. Імпульси синхронізації приймача формуються з тактової частоти таймера 1 спільно зі схемами детектора спаду, детектора бітів і керованого дільника на 16.

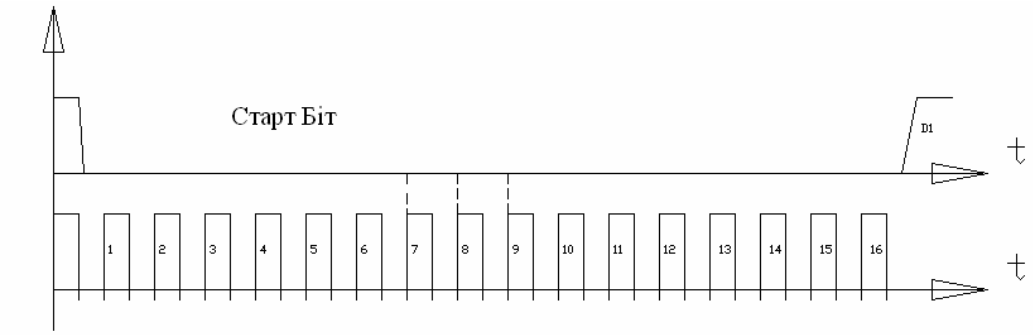


Рис. 8.9. Перевірка старт-біта в середині інтервалу для захисту від імпульсної перешкоди

Приймання починається під час виявлення детектором спаду переходу з 0 в 1. При цьому скидається в 0 дільник на 16 приймача. На 7-му, 8-му і 9-му такті дільника опитується значення входу приймача RxD (рис. 8.9) і, якщо за трьома вимірами більше 1, старт-біт не зачитується, йде його подальший пошук. Таким чином, відбувається підстроювання фази тактування приймача за сигналами з лінії зв'язку. Вона забезпечує підтвердження значення прийнятих даних у середині тактового інтервалу. А мажоритарне опитування (за трьома крапками на інтервалі) – захист від імпульсних перешкод.

При прийманні, крім контролю старт-біта, здійснюється контроль стоп-біта. Якщо він не дорівнює 1, посилка не зачитується й дані губляться.

Як правило, обмін даними між контролерами проводиться декількома байтами. Цю послідовність байтів називають повідомленням. Для визначення початку повідомлення використовують або не застосовуване в посилці оригінальне значення першого байта, або так званий тайм-аут, коли протягом певного часу передача даних перед передачею чергового повідомлення не відбувається.

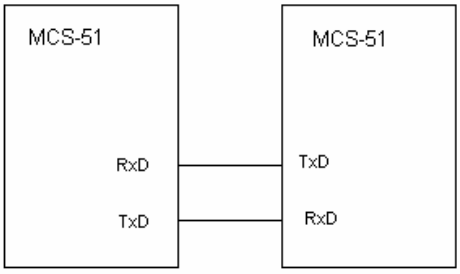


Рис. 8.10. Схема підключення двох контролерів у режимі 1

Як правило, обмін між пристроями в послідовному форматі відбувається за алгоритмом запит – відповідь, або передача – підтвердження. При цьому ініціатором обміну є провідний контролер або Master. Другий контролер називають веденим або Slave. Задання на програму обміну інформацією між контролерами можна сформулювати в такий спосіб: необхідно передати через послідовний інтерфейс масив даних, розміщених з адреси Adr1, довжиною в L байт; прийняти у відповідь масив довжиною Lotv і записати його з адреси Adr2.

На рис. 8.11 наведено схему алгоритму й програми, що виконує такий обмін без використання переривань.

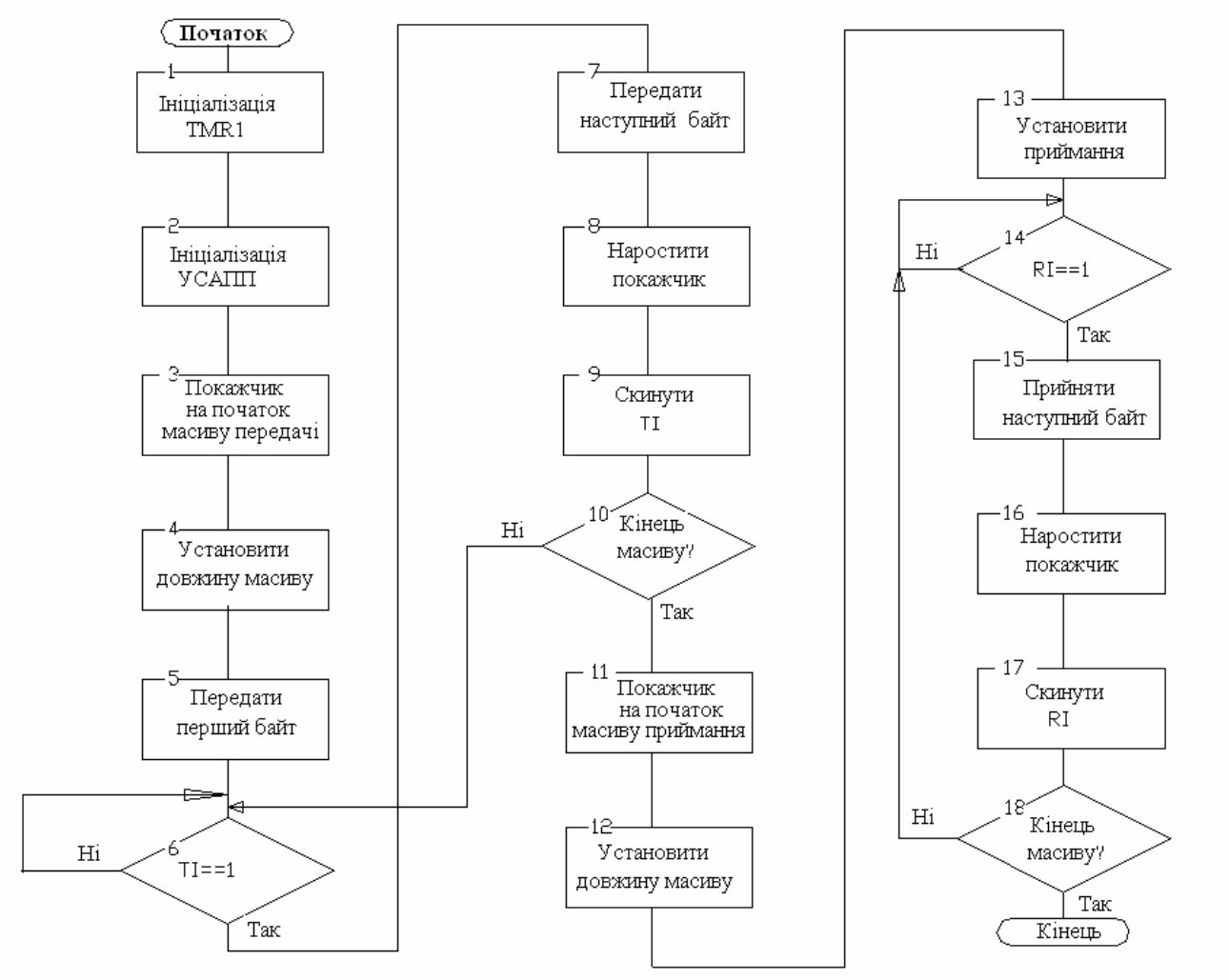


Рис. 8.11. Блок-схема алгоритму обміну повідомленнями між контролерами в режимі 1

Приклад програми:

```
        Mass equ 30h; початок переданого масиву
        Recivmas equ 50h ; початок прийнятого масиву
        Flag equ 0h ; ознака закінчення обміну даними
        Razmer equ 2h ; R2
        Ukaz equ 0h ; R0
;OSNOVNAYA PROGRAMMA
```

```
main: mov TMOD,#20h; reg 2 taim 1
        mov PCON,#80h;SMOD=1
        mov TL1,#0fdh
        mov TH1,#0fdh; частота обміну 19,2 kgc
        setb TCON.6; режим роботи USART
        setb SCON.6
        setb SCON.7
        setb SCON.3
        mov Ukaz,#mass+1
        mov Razmer,#9; довжина масиву
        setb Flag
;PEREDACHA
        mov SBUF, Mass; передача першого байта
m1t:    jnb SCON.1,m1t
m2t:    mov SBUF,@R0
        inc r0
m2:     clr SCON.1
        djnz r2,m1t; контроль кінця масиву передачі

m1:
;PRIEM
        mov Ukaz,#recivmas
        mov Razmer,#10; довжина масиву приймання
        setb SCON.4
m1r:    jnb SCON.0,m1r
reciw:  mov @R0,SBUF; зчитування прийнятого байта
        inc R0
        clr SCON.0
        djnz r2,m1r; контроль кінця приймання
m2r:    ljmp m2r
        END
```

На початку програми задається другий режим таймера 1, а також заноситься константа в TH1, що забезпечує швидкість обміну 19,2 кбіт/с.

Потім настроюється УСАПП на перший режим роботи, а також заноситься перший байт масиву в SBUFF. Контроль передачі байта здійснюється шляхом програмного опитування біта TI в SCON. Після закінчення передачі чергового байта цей біт установлюється в одиницю. В SBUFF заноситься значення наступного елемента масиву, нарощується покажчик масиву, скидається в 0 біт TI, декрементується й перевіряється на нуль лічильник елементів масиву. Якщо останній елемент масиву передано, програма переходить на приймання відповіді на запит, якщо він був не останнім, то вертається на контроль біта TI. Після закінчення передачі масиву УСАПП переводиться в режим приймання, для цього біт REN (SCON.4) установлюється в 1. Потім формується значення покажчика приймального

масиву, установлюється його довжина й відбувається опитування й контроль біта RI (SCON.0). Одиниця в цьому біті означає, що приймач закінчив приймання чергового байта, тепер необхідно зчитати буфер, щоб звільнити його для приймання наступного байта. Після зчитування з SBUFF значення заноситься в масив приймання, нарощується покажчик елементів масиву, скидається біт RI, декрементується й перевіряється на нуль лічильник елементів масиву. Якщо останній байт масиву прийнято, біт REN скидається в 0, приймання припиняється й програма переходить на продовження виконання завдання керування.

Режим роботи 2.

У цьому режимі, крім старт- і стоп-біта та 8 бітів даних, додається дев'ятий біт, значення якого можна задавати програмно. Він використовується для підвищення вірогідності приймання інформації (для передачі значення біта контролю парності), або для передачі ознаки початку повідомлення в багатоконтролерних мережах, де першим байтом передається номер адресата. Частота приймання/передачі може дорівнювати 1/32 або 1/64 частоти резонатора залежно від значення SMOD (PCON.7).

Режим роботи 3.

Режим 3 збігається з режимом 2 за форматом повідомлення але швидкість обміну може задаватися таймером 1, як і в режимі 1.

Контрольні запитання

1. Яку кількість бітів містить посилка УАПП МК51 в режимі 1?
2. Яку кількість бітів містить посилка УАПП МК51 в режимі 2?
3. Який біт регістра SCON УАПП МК51 дозволяє роботу контролера в мережевому режимі?
4. Який сигнал УАПП МК51 використовується в системі переривань при передачі інформації?
5. Який сигнал УАПП МК51 використовується в системі переривань при прийманні інформації?
6. Скільки режимів роботи має послідовний порт МК51?
7. Яку кількість бітів містить посилка УАПП МК51 в режимі 0?
8. Який перехід в лінії зв’язку характеризує початок посилки?
9. Який перехід в лінії зв’язку характеризує завершення посилки?
10. Який біт регістра SCON УАПП МК51 дозволяє приймання інформації?

9. ПОСЛІДОВНІ ШИННІ СИСТЕМИ (КАНАЛИ ЗВ'ЯЗКУ)

9.1. Загальні поняття про електронні шинні системи

В сфері електроніки під шинними системами розуміють зв’язок між більш ніж двома електронними компонентами або системами задля передачі інформації між ними через спільний канал зв’язку.

В перекладі на німецьку «шинна система» буде «Bussystem», де «Bus» перекладається також як «автобус». Таким чином, можна припустити, що це слово було запозичено зі сфери транспортної техніки, де автобус для пасажирів є транспортним засобом, який везе їх за єдиним маршрутом, хоча, як правило, вони підсаджуються в різних точках шляху і мають різні пункти призначення.

Середовище передачі даних в електронних шинах часто має гальванічний характер, тобто воно складається зі з’єднань, що проводять електричний струм, до яких підключені елементи системи. Також існують та використовуються інші середовища передачі даних, – оптичні з’єднання (наприклад, за допомогою оптичного волокна) та безпровідне з’єднання.

В цьому розділі мова йтиме про неоптичні лінії передачі даних.

Таким чином, згідно з визначенням, до однієї шини можуть бути приєднанні більш ніж два об’єкти, тобто існує декілька джерел передачі даних. Звідси одразу стає зрозуміло, що однією з основних особливостей шини має бути обробка збоїв на лінії зв’язку або ж необхідні стратегії, які зможуть з самого початку ліквідувати подібні збої під час передачі інформації.

Електронні шинні системи поділяються на паралельні та послідовні.

В паралельних шинних з’єднаннях перенесення даних відбувається паралельно, тобто одночасно крізь декілька паралельних ліній зв’язку. Паралельні шинні з’єднання часто розглядають як шини задньої стінки в комп’ютерних чи мікропроцесорних системах для з’єднання декількох електронних блоків, які знаходяться на різних платах.

В промисловій та непромисловій сферах використовується велика кількість різноманітних паралельних шин. Як приклад з промислової сфери можна назвати шину VME, що використовується у вимірювальній техніці та техніці автоматизації.

До характерних особливостей паралельних шинних з’єднань відносяться:

- висока швидкість передачі даних;
- коротка довжина шин;
- високі витрати на процеси з’єднань через велику кількість ліній зв’язку.

У шинних з’єднаннях послідовного обміну передача даних відбувається послідовно в часі, тобто в формі один за одним переданих бітів через один канал зв’язку.

До характерних особливостей послідовних шин відносяться:

- більш низька у порівнянні з паралельними шинами такої самої тактової частоти швидкість обміну даними;

- більш низькі витрати на процеси з'єднань через меншу кількість ліній зв'язку;
- більша просторова протяжність шини, яка може коливатись від декількох десятків метрів до кількох кілометрів;
- необхідні паралельно-послідовні та послідовно-паралельні перетворювачі даних, оскільки інформація в передавачі та приймачі, як правило, і видається, і приймається в паралельній формі.

Більше того, шини послідовного обміну розрізняються за сферою їх використання. Ми можемо спостерігати в оточуючому нас середовищі, наприклад, в офісі «нормальні» шини, для яких не існує окремого терміна. Відомий приклад – USB (Universal Serial Bus – універсальна послідовна шина) зі сфери персональних комп'ютерів.

Системам, що позначаються спеціальним терміном «польові шини», властивий послідовний обмін більш високої надійності, особливо це стосується електромагнітної сумісності для жорстких умов – використання в промисловості або в транспортних засобах. Перш за все, за допомогою польових шин було створено велику кількість досить різноманітних систем, що поширені також дуже по-різному. В якості приклада можна привести одну з найпоширеніших систем CAN (Controller Area Network). Це система, що застосовується в мільйонах транспортних засобів, а також у великій кількості інших сфер.

9.2. Різновиди послідовних шинних систем

Кількість шинних систем, пов'язаних з керуванням в сфері офісної техніки, не дуже велика. До них відносять USB (Universal Serial Bus), Ethernet , а також IEEE 1394 «Firewire».

На відміну від офісних, існує велика кількість різноманітних систем в сфері промисловості. В цьому випадку потрібно розрізняти так звані патентовані та «відкриті» шини. Патентовані системи мають різноманітні характеристики, в залежності від специфіки роботи фірми, і вони, як правило, не розголошуються.

«Відкритою» називають таку шинну систему, до специфікацій якої відкрито вільний доступ, і прилади, що підключаються до шини, можуть бути змінені та допрацьовані будь-якою фірмою. Однак, стосовно ступеня відкритості існують чіткі відмінності. Наприклад, щоб мати право використовувати бренд з шини деяких систем, необхідно платити за ліцензію. Або треба перебувати в певних групах, проводити тести на сумісність.

Ще одна явна відмінність відносно відкритості системи стосується різних комунікаційних рівнів шинної системи, що будуть розглядатися в наступному розділі. Таким чином, існують шини, які відкриті тільки на певних сеансових рівнях, на інших рівнях знаходяться патентовані елементи. Класичним прикладом можуть бути шинні системи в транспортних засобах, таких як CAN, де обидва нижні рівні комунікації (так званий фізичний рівень та рівень біт-передачі) відповідають міжнародним стандартам, однак, на рівні прикладного

протоколу (тобто значення переданих даних) виробники автомобілів часто використовують свої особисті закриті специфікації.

Вже йшла мова про те, що для розуміння шинних систем дуже корисно знати їх загальну класифікацію за ISO/OSI (базова модель для систем зв'язку). Ця модель включає в себе 7 рівнів, які виконують різні функції в шинних системах, що тут розглядаються. Нижній рівень має характеристики середовища передачі. Другий рівень (рівень біт-передачі) містить процеси зображення бітів та доступу до шини, включаючи рішення або попередження комунікаційних колізій.

Рівні 4, 5 і 6 в шинних системах, що розглядаються, виражені слабо і через свою простоту до уваги не приймаються. Рівень 7 (прикладний рівень або протокол прикладної програми) є важливим. Варто відзначити, що приписування окремих специфікацій, процесів або протоколів до певних рівнів не завжди можливо та потрібно. Не дивлячись на це, багаторівнева модель допомагає все структурувати та зрозуміти.

На прикладі використання CAN в транспортних засобах вже було показано, що в шинних системах різні рівні можуть бути стандартизованими чи нестандартизованими. Дуже важливо враховувати це, коли мова йде про одиничні шини.

Часто мета відкритої шинної системи полягає в тому, щоб різні виробники мали змогу розробляти устаткування для даної системи, а прилади від них працювали на тій самій шині. При цьому варто чітко розрізняти, чи є ці прилади «сумісними» або «мають можливість взаємодіяти». В цьому випадку під сумісністю розуміють можливість приладів не заважати комунікації в шині, а приймати в ній активну участь. Функціональна сумісність потребує загальних специфікацій значення, тобто інтерпретації переданих даних. В цьому відношенні функціональна сумісність передбачає специфікатори на всіх реалізованих шинною системою комунікаційних рівнях, зокрема, на верхніх рівнях протоколу (прикладних рівнях).

Через широкий спектр вимог та умов використання виникла велика різноманітність промислових шинних систем. Сьогодні відомо більш ніж 50 шин з відкритим протоколом, частково з підваріантами застосування, що потребують особливої безпеки, та тільки декілька шин, спеціально розроблених для забезпечення безпеки критично важливих додатків. Крім того, відомо понад 15 відкритих промислових систем Ethernet, що базуються на комунікаційних системах.

Щоб мати уяву про різноманіття шин, які використовуються в сфері промисловості, перерахуємо деякі з них: ARCNET, AS-Interface (Aktuator-Sensor-Interface), BACnet (Building Automation and Control Network), Bitbus, byteflight, CAN (Controller Area Network), CC-Link, EIB (European Installation Bus), FIP (Factory Instrumentation Protocol), Flexray, HART (Highway Addressable Remote Transducer), INTERBUS, LIN (Local Interconnecting Network), LON (Local Operating Network), Modbus, MOST (Media Oriented System Transfer), PROFIBUS (mehrere Varianten), Safetybus-P, SERCOS, SwiftNet. І це ще не весь перелік.

Наведемо декілька прикладів промислових систем Ethernet, тобто систем, що базуються на різноманітних модифікаціях Ethernet, що були вдосконалені та адаптовані до використання в промисловому середовищі: EtherCAT, Ethernet/IP, Ethernet Powerlink, Profinet, SERCOS III, VARAN.

Звісно, виникає питання, яка шинна система є найбільш відповідною до тих чи інших вимог, та системи, які мають розширення. У [GRU2001] детально розібрано засіб для порівняння, оцінки та вибору послідовної шини.

В наступних часткових главах коротко наведені деякі приклади різноманітних систем зв’язку, які, без сумніву, мають відношення до мікрокомп’ютерів чи мікроконтролерів.

В наступних підпунктах будуть розглянуті деякі приклади абсолютно різноманітних комунікаційних систем, які, звісно, безпосередньо пов’язані з мікрокомп’ютерною та мікропроцесорною технікою.

9.3. Окремі приклади інтерфейсів та послідовних шин

Далі в якості приклада будуть наведені інтерфейс RS 232, система RS 485, локальна шина I2C, а також польові шини CAN і Modbus.

Зв'язок між комунікаційною системою та мікропроцесором або мікроконтролером здійснюється за допомогою зв’язкового контролера та трансивера. Шинні контролери генерують та обробляють логічні фронти імпульсу та послідовність бітів, в той час як трансивери в якості безпосереднього інтерфейсу до комунікаційних ліній надають та сприймають сигнальні фронти імпульсу та рівні біта безпосередньо як сигнали на лінії.

З одного боку зв’язкові контролери можуть приєднуватись користувачем до мікроконтролерів як окремі модулі за допомогою адресних шин, ліній передачі даних та шини керування. З іншого – зараз багато мікроконтролерів вже обладнано вбудованими зв’язковими контролерами, що, звісно, суттєво полегшує роботу розробника.

Таким чином, на сьогодні доступні багато модифікацій родини мікропроцесорів 8051 з вже вбудованими комунікаційними контролерами, що відіграють особливу роль в цій книзі.

9.3.1. Інтерфейс RS 232

Інтерфейс зв’язку RS 232 являє собою EIA-Norm (Electronic Industries Association). Тут мова йде не про шинну систему, а про зв’язковий інтерфейс виключно між двома користувачами, тобто двокрапкове з’єднання або однорангове з’єднання «ЕВМ-ЕВМ» (оверлейна комп’ютерна мережа).

9.3.1.1. Визначення логічного рівня

Визначення логічного рівня відносять до електричних властивостей інтерфейсу RS 232. Варіант такого інтерфейсу демонструється на прикладі прямого імпульсу (рис. 9.1).

Типове значення рівня лінії передачі даних складає +12V для логічного нуля та -12 V для логічної одиниці. Середні значення знаходяться в діапазоні

+3V – логічний нуль, -3V – логічна одиниця, та максимально допустимі значення – +15V та -15V відповідно.

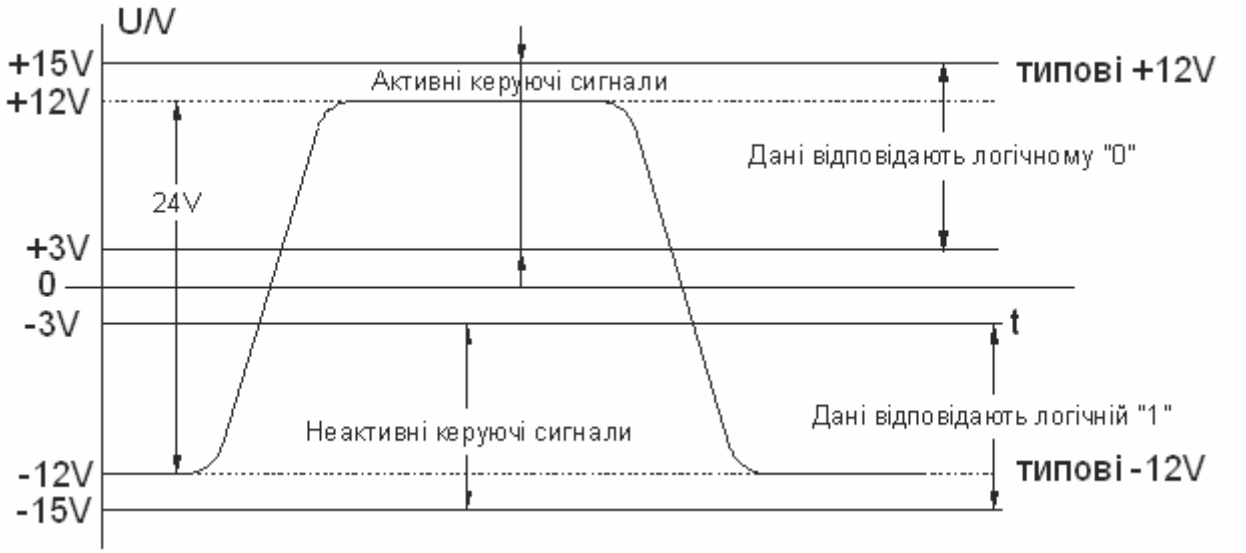


Рис. 9.1. Визначення логічного рівня RS 232 для ліній передачі даних та керуючих сигналів

Таким чином, біт даних буде розшифровано як логічну одиницю, якщо рівень U менший за -3V, і як логічний нуль при рівні U більший за +3V. Навпаки, сигнали, що сповіщають та керують, є активними, коли напруга більша за +3V, та неактивними, коли рівень U менший за -3V.

9.3.1.2. Розведення контактів та сигналів

В якості роз'ємного з'єднання RS 232 використовують, наприклад, штекерний підроз'єм D. Його вигляд згори зображено на рис. 9.2.

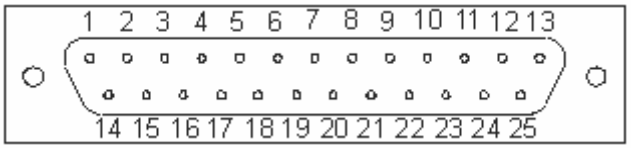


Рис. 9.2. 25-контактний штекерний підроз'єм D інтерфейсу з призначенням виводів для інтерфейсу RS 232

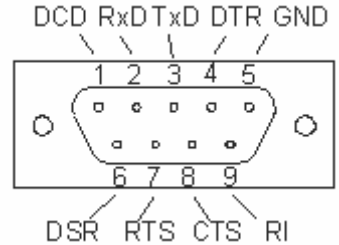


Рис. 9.3. 9-контактний штекерний підроз'єм D інтерфейсу з призначенням виводів для інтерфейсу RS 232

Магістралі, що використовуються для передачі даних, можуть бути змонтовані у відповідності до розпайки виводів, що демонструється вище (рис. 9.3).

Функціональні характеристики інтерфейсу RS 232 можна виділити з наведених нижче ліній передач (рис. 9.4). Тут демонструються найбільш важливі лінії передач.

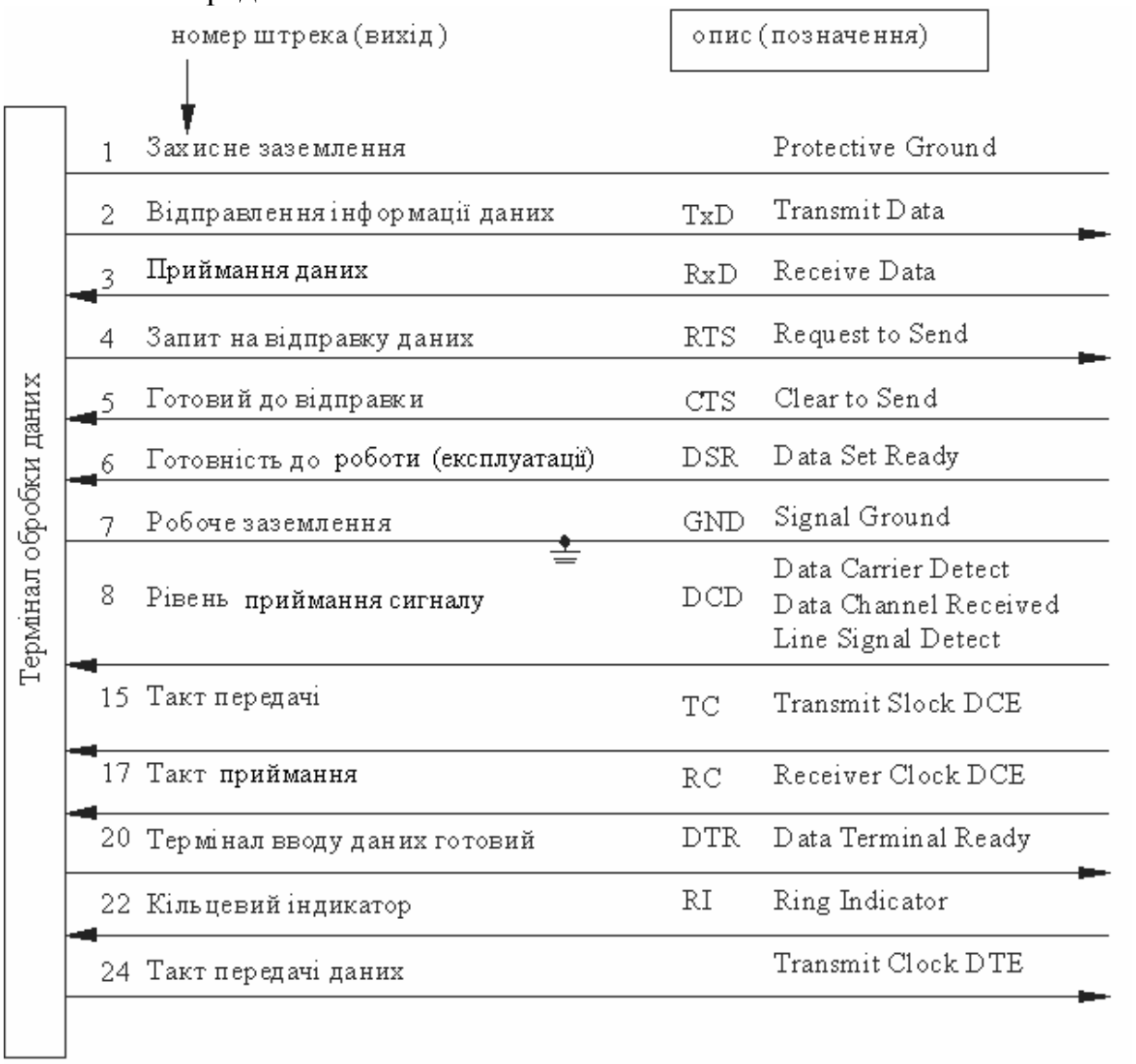


Рис. 9.4. Призначення сигналів інтерфейсу RS 232

На рис. 9.5 зображено просте двопровідне з’єднання між двома терміналами на основі RS 232.

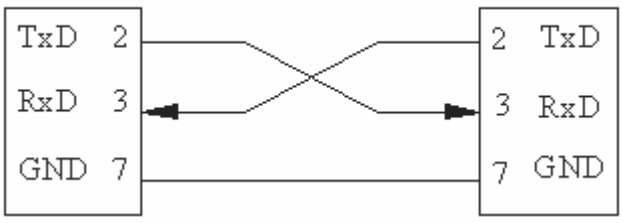


Рис. 9.5. Мінімальна конфігурація з’єднання RS 232 між двома терміналами

9.3.1.3. Довжина кабелю

На магістралях інтерфейсу RS 232 сигнали є біполярними. Логічний рівень може коливатись від +3V до +15V та відповідно від -3V до -15V. Завдяки цьому може бути забезпечена швидкість передачі даних 19200 бодів на відстань приблизно 15 метрів, якщо паразитний зв'язок інших приладів не дуже сильний. Цього можна запобігти, якщо використовувати екрановані кабелі. Двічі екранований кабель з маленькою ємністю дає можливість використовувати в два та навіть і в три рази більшої довжини кабель RS 232 шляхом зниження спотворення форми імпульсу та значного зниження паразитних зв’язків.

9.3.2. З’єднання RS 485

З’єднання за допомогою EIA RS 485 дає можливість підключення до шини, комунікаційної лінії зв’язку до 32 користувачів.

Інтерфейс RS 485 задає, так само як і EIA-Standard, тільки електричні або, кажучи іншими словами, фізичні умови. Його було розроблено для того, щоб мати можливість приєднати більшу кількість користувачів до шини. В цьому інтерфейсі можливо підключення 32 користувачів (як передавачів, так і приймачів даних) до одного двопровідного шинного кабелю з симетричним перенесенням даних. Завдяки симетричній конструкції інтерфейсу досягається високий рівень захищеності від перешкод.

Для передачі логічного рівня використовується метод різниці напруг. Всі прилади, які є у передавача та приймача даних, приєднуються до єдиної двопровідної лінії передач, це означає, що вони ввімкнені паралельно (див. рис. 9.6)

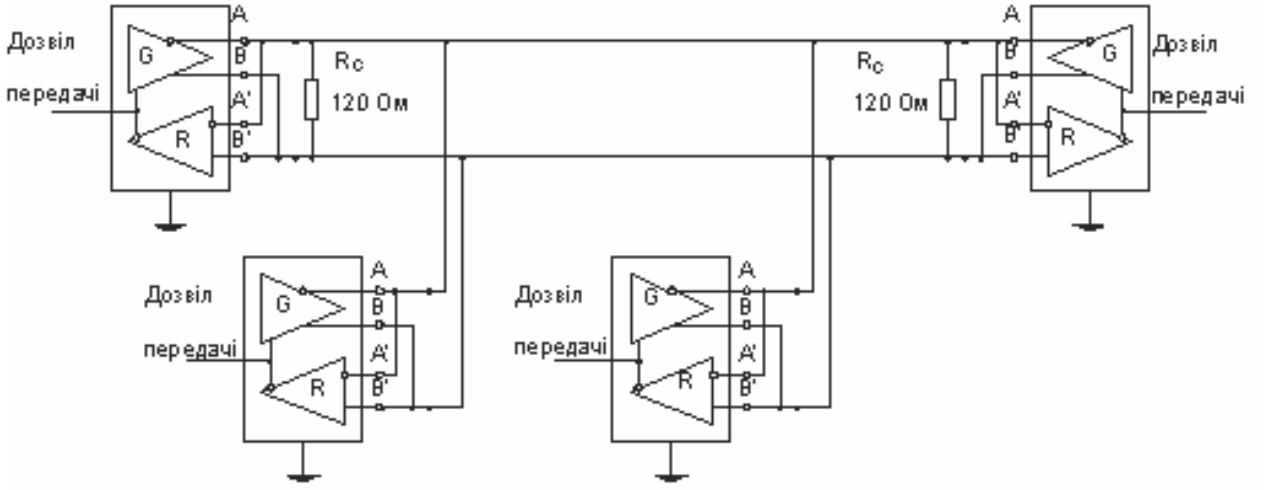
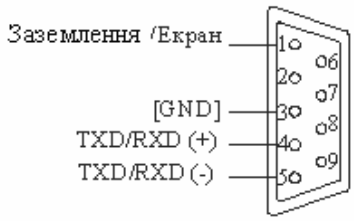


Рис. 9.6. Шинна система з інтерфейсом RS 485

Часто для з’єднання RS 485 використовується 9-контактний штекерний роз’їм. На рис. 9.7 зображено розташування виводів та призначення сигналів такого інтерфейсу.



Вивід	Функція
1	Schutzerde/Schirm => Заземлення/Екран
2	-
3	GND/Signalmasse => З'єднання не обов'язкове
4	TXD/RXD (+) Відправлені/Отримані дані (+)
5	TXD/RXD (-) Відправлені/Отримані дані (-)
6	-
7	-
8	-
9	-

Рис. 9.7. Розташування виводів та сигналів інтерфейсу RS 485

Основні характеристики з'єднання RS 485: максимальна довжина лінії передач 1200 м (при зменшеній швидкості обміну даними 100 Кбіт/с) та максимальна швидкість передачі інформації 10 Мбіт/с (при дуже маленькій довжині магістралі).

EIA-Norm визначає спочатку тільки електричні (фізичні) характеристики інтерфейсу, коли на одну шинну лінію передач приєднується декілька користувачів (тут до 32). Проблему комунікаційного конфлікту можна розв'язати за допомогою відповідного протоколу обміну даними. Тому з'єднання використовується завжди разом з протоколом обміну. Протокол може бути як власної розробки, так і стандартизований. Це означає, що описуючи з'єднання, реквізити «RS 485» вказують тільки на фізичні характеристики (фізичний рівень 1). Додатково потрібні дані про протокол, що використовується (фізичний рівень 2 та вище). Оскільки з'єднання RS 485 приєднується до шини, то за його допомогою можуть бути реалізовані також і польові шинні системи. Таким чином, наприклад, раніше реалізовувалися частини протоколу CAN на основі фізичних з'єднань RS 485. Як приклад польових шин, що частково використовують RS 485б, можна назвати ще PROFIBUS, MODBUS та INTERBUS.

9.3.3. Шина I^2C

Шина I^2C є послідовною. Однак вона не є польовою. Створена для відносно коротких ліній передач та не забезпечує достатню стійкість до перешкод. Де ж тоді її застосовують?

$I^2C = IIC$ – це скорочення для Inter-IC-Bus (шина на основі внутрішньої IC). Вона була розроблена для з'єднання мікропроцесорів та інших інтегральних мікросхем на одній платі. Так само може йти мова про плати в

одній гніздовій системі, однак, в таких випадках I^2C використовується дуже рідко. Таким чином, ця шина точно виконує ті задання, що зазвичай в мікропроцесорній системі обробляються паралельною системною шиною. Однак навіщо було для цього розробляти додаткову послідовну шину, яка набагато довша, ніж звичайна паралельна? Зараз фірма Philips, що є власником патенту на цю шину, виробляє велику кількість електронних приладів широкого споживання. У зв'язку з різноманітними причинами сьогодні майже кожен прилад має мікропроцесор або мікроконтролер. Процесори та периферійні мікросхеми, що з'єднані за допомогою паралельних шин, потребують більшої кількості ліній зв'язку. Останні і покривають поверхню плати. Мікрочипи мають зарезервувати багато штиркових виводів для приєднання шини. В результаті використання таких ІС отримуємо досить велику конструкцію, що займає багато місця на платі. Також одержуємо дуже велику кількість місць запаювання плати. Зараз вона, як правило, по відношенню до кількості інформації, що передають прилади, достатньо низька. Шина I^2C за своїм початковим стандартом надає максимальну швидкість передачі даних біля 10 Кбайт/с. Але цього достатньо майже для всіх випадків її застосування. Оскільки шина I^2C має лише дві лінії передачі сигналу, на платі буде розташовано менше ліній. ІС також буде менше. Таким чином, зменшується трудомісткість виготовлення та загальний розмір схеми. Між тим, шина I^2C стала широко відомою як стандарт для «повільного» з'єднання на одній платі. Фірма Philips, як і інші виробники, пропонує велику кількість інтегральних схем з інтерфейсом шини I^2C.

9.3.3.1. Концепція шини I^2C

Дві лінії передач, SDA (Serial DAta) та SCL (Serial CLock), використовуються для обміну інформацією. Кожен користувач шини має свою власну адресу. При цьому зовсім немає різниці, чи мова йде про модуль ЗУ, чи про ЖК-індикатор або мікроконтролер. Кожен з цих користувачів може виконувати функції відправника та одержувача інформації в залежності від їх призначення. Клавіатурний інтерфейс, як правило, тільки надсилає дані, а ЖК-дісплей тільки приймає. Модуль пам'яті, так само як і керований мікроконтролер, навпаки, може виконувати обидві ці функції. Важливо розділяти прилади на ті, що ведуть, та ті, котрих ведуть. Ті, що ведуть, це такі пристрої, що ініціюють саму передачу даних та її напрям, для цього вони генерують імпульси синхронізації на SCL лінії. Пристрій, що реагує на сигнал того, що веде, та таким чином приєднується до передачі даних, називають тим, котрого ведуть. Деякі пристрої в певний момент можуть брати на себе виконання функцій як тих, що ведуть, так і тих, котрих ведуть. Зміна стану виконується динамічно. Таким чином, мікроконтролер в один певний момент часу може бути в ролі пристрою, що веде, та надсилати дані на ЖК-дісплей, а в інший момент може отримувати інформацію від іншого контролера, наприклад, дані вимірювань (як прилад, якого ведуть). Важливо те, що кожен користувач шини може виконувати функції як ведучого пристрою, так і того, що ведуть,

однак не всім мікросхемам це властиво. Більшість інтегральних схем, сумісних з шиною I^2C, ніколи самостійно не ініціюють передачу, і таким чином можуть приймати тільки режим пристрою, якого ведуть. Фактично лише модулі з власним інтелектом спроможні виконувати обидві функції. Тому що будь-яка кількість інтегральних схем на шині може ініціювати пересилання сигналів, шина I^2C вважається мультиведучою. Звісно, в один момент часу може бути активним та використовувати шину лише один ведучий пристрій. Протокол шини I^2C пропонує різноманітні засоби для попередження конфліктів. Тому ведучі пристрої перед початком передачі інформації перевіряють, чи вільна шина. Випадок, коли обидва ведучі пристрої одночасно готові розпочати передачу, може бути сприйнята двома способами: тактовою синхронізацією та керуванням доступом до спільного ресурсу. При цьому перший використовують для того, щоб синхронізувати тактовий час обох пристроїв. Керування доступом приводить до того, що ведучий прилад спочатку видає 1 як задане, в той час як інший ведучий прилад видає 0 та втрачає право доступу до спільної шини. Тому нульовий логічний рівень на шині I^2C вважається «домінантним рівнем».

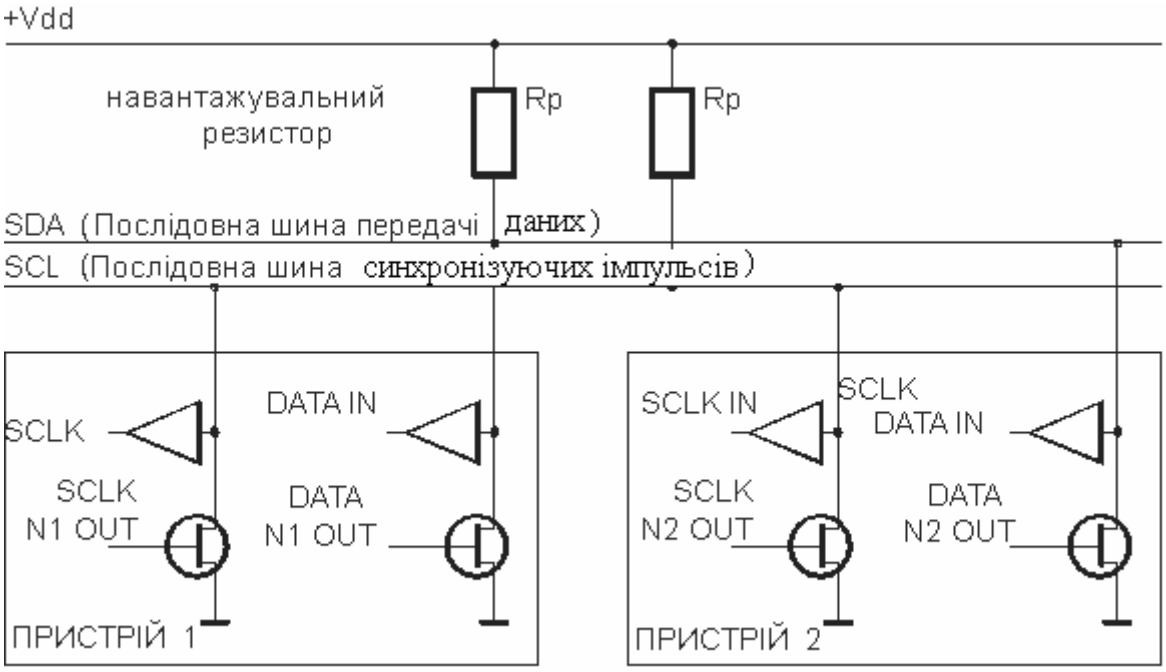


Рис. 9.8. Дротяне з’єднання «I»

Як SCL, так і SDA лінії функціонують в двох напрямках. Вони постійно тримаються на високому рівні за допомогою резистора, що підтягує рівень. Коли шина вільна, обидві лінії мають високий рівень. Мікросхеми, підключені до шини I^2C, повинні мати вихід з відкритим стеком або колектором. З ними відповідна лінія може «спуститися» до видачі нуля на заземлення. Сукупність подібних мікрочипів на шині реалізує так зване «дротяне з’єднання «I» (рис. 9.8). Максимальна швидкість передачі даних зазвичай складає 100 Кбіт/с,

а в так званому швидкісному режимі при розширені протока I^2C – 400 Кбіт/с. Кількість користувачів шиною обмежується лише максимальною ємністю шини 400 пФ.

9.3.3.2. Передача даних на рівні розрядів

Основою передачі текстових даних є передача окремих байтів. Це завдання розрядного протоколу шини I^2C. В цілому окремі розряди кодуються через рівень на лінії синхронізуючих імпульсів та SDA. SCL функціонує в цьому випадку як послідовний синхронізатор (як при здвиговому регістрі). Спеціальний рівень – комбінації фронтів обох ліній передач кодують так звані початкові та кінцеві умови. Таким чином, за допомогою обох ліній SDA та SCL можна отримати сигнал про початок та кінець передачі даних. Це можливо не в усіх послідовних шинах. В інших для цього потрібні додаткові лінії передач. Для обох ліній передач рівень нижче 1,5 V вважається низьким, а вище 3 V – високим. Рівень, що знаходиться між заданими значеннями, є невизначеним.

Достовірність даних

Дані на лінії SDA мають бути стабільними до того часу, поки SCL лінія має високий рівень. В іншому випадку зміни на лінії SDA будуть інтерпретовані як спеціальні умови початку та кінця передачі. Це означає, що пристрої, які передають, мають перевести свої дані по низькому рівню SCL на лінію SD, а, пристрої які приймають, звісно, читають цю лінію на високому рівні SCL.

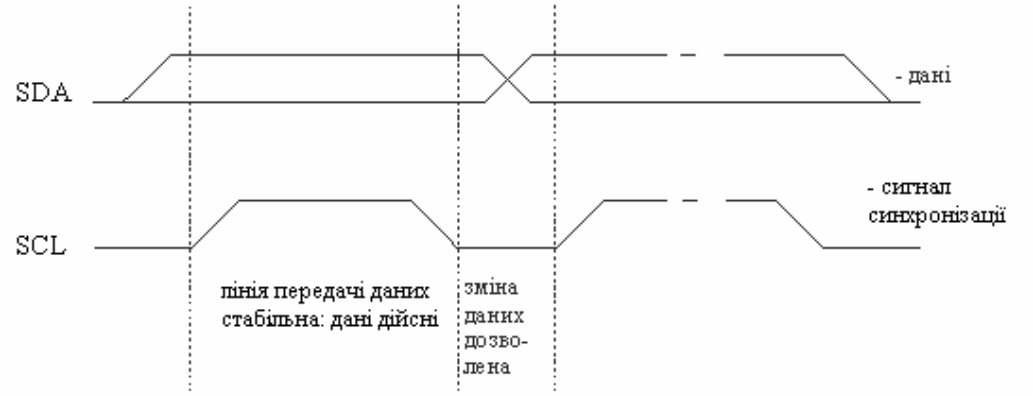


Рис. 9.9. Перевірка достовірності даних

Умови початку та кінця передачі

Умови початку та кінця передачі кодуються через фронти SDA лінії при високому логічному рівні ведучої SCL лінії. Для генерації умови початку передачі потрібно сформувати перепад з 1 в 0 сигналу SDA, для умови кінця – з 0 в 1. Це продемонстровано на рис. 9.10. Згідно з специфікаціями, умови початку та кінця можуть генеруватися до кожного моменту часу поточної передачі даних. Але рекомендується обережність, коли до шини підключені прилади, що лише частково відповідають стандартам, які сюди відносяться. Це

часто стосується випадків програмної реалізації протоколу шини на мікроконтролері.

Після умови початку передачі шина вважається зайнятою. В такому випадку жоден інший ведучий пристрій, крім того, що задав умови початку, не може генерувати умови кінця передачі даних. Після того, як задано умову кінця передачі, шина вільна.

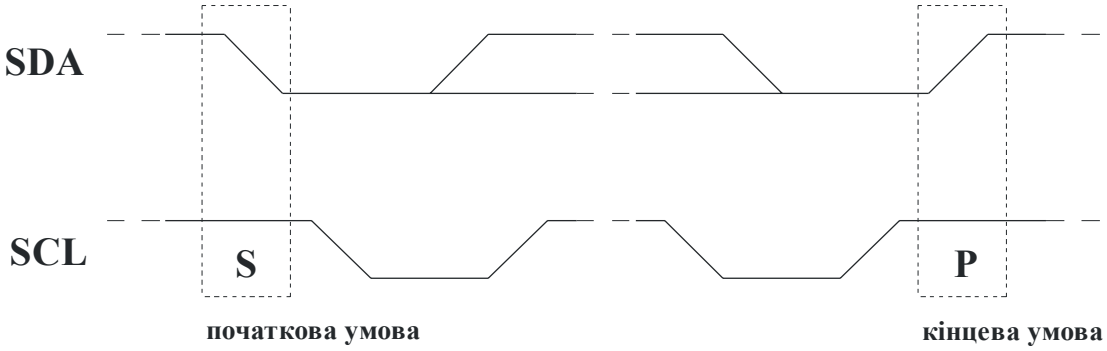


Рис. 9.10 Умови початку та кінця передачі даних

9.3.3.3. Керування доступом до спільного ресурсу (розподіл доступу до шини) та тактова синхронізація

Тактова синхронізація

Всі ведучі прилади самостійно генерують тактовий імпульс. Однак при реалізації лінії шини у вигляді дротового з'єднання «І» виникає можливість синхронізувати тактові імпульси конкуруючих ведучих пристроїв. Один ведучій пристрій, що видає низький рівень на лінію SCL, буде потім віддавати його назад, тобто видавати високий рівень. З цього моменту ведучий пристрій не тільки продовжує працювати далі, а й в свою чергу зчитує повторений сигнал лінії SCL, доки вона не буде мати високий рівень. Це означає, що доти, доки інший механізм знижує лінію SCL, ведучий пристрій синхронізується з ним. Цей інший механізм може бути як ведучим пристроєм, так і тим, якого ведуть, що й викликає затримку. В цілому здійснюється синхронізація найдовшого низького рівня та найповільнішого механізму.

Арбітраж доступу (керування доступом до спільного ресурсу)

Під арбітражем розуміють процес розподілу прав доступу до шини між кількома ведучими пристроями. На шині I^2C реалізується спеціальна операція, що називається CSMA/CA (Carrier Sense, Multiple Access with Collision Avoidance). Carrier Sense означає, що ведучий пристрій, який хоче виконати передачу даних, сканує лінії передач, щоб виявити, чи зайнята шина. Шина I^2C визначається як зайнята, коли одна з ліній передач знаходиться у стані з низьким рівнем напруги. Інтервал, під час якого обидві лінії передачі даних мають бути в стані з високим рівнем напруги (стан логічної «1»), щоб шина була розпізнана як вільна, зображено в технічному описі параметрів Philips як $t_{HD;STA}$. В стандартному протоколі він складає мінімум 4,7 мікросекунди. Якщо ведучий пристрій розпізнав лінію як вільну, він задає умови початку передачі і

в першому біті видає адресу бажаного пристрою, якого ведуть. Якщо одночасно з першим ту саму дію виконує інший ведучий пристрій, то спочатку потрібно зрівняти імпульси на SCL лінії (див. попередній розділ). Під час зчитування синхронізованих фаз рівня високої напруги ведучі пристрої кожен раз порівнюють SDA-біти, котрі видають, з тими, котрі вони зчитують. Це функціонує так само як при керуванні синхронізованою лінією передач. Ведучий пристрій, який видає «1» та приймає «0» (так як «0» через дротове з'єднання «I» є домінантним), розрізняє таким чином, що інший механізм в цей момент використовує шину, і переривае передачу даних.

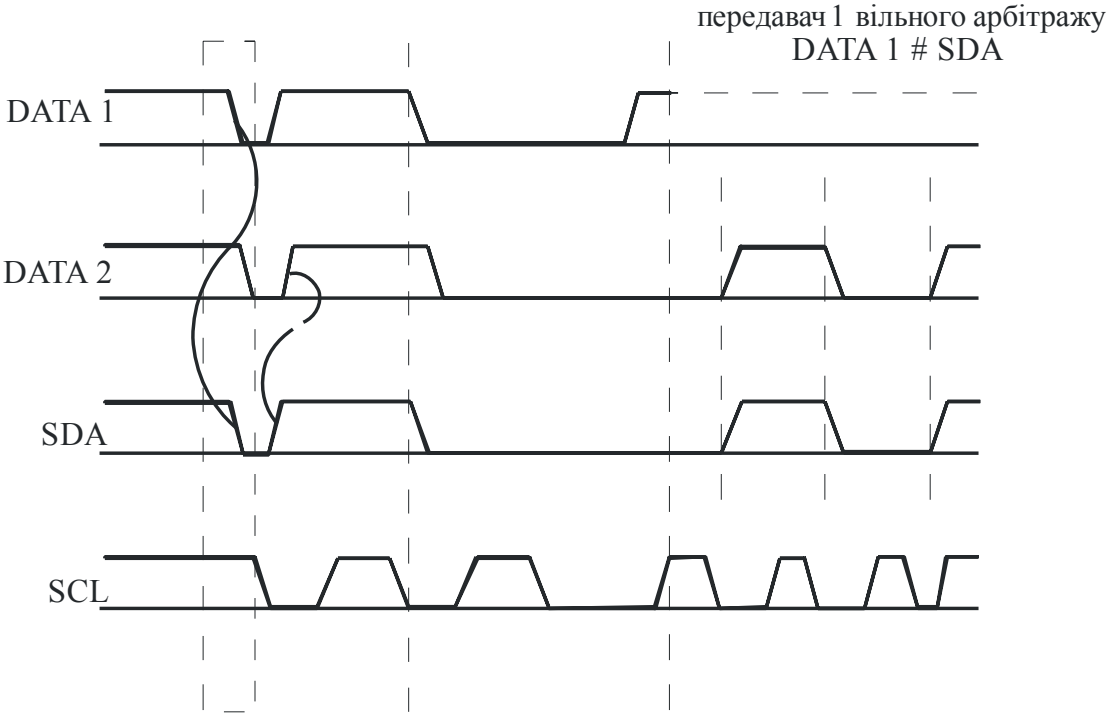


Рис. 9.11. Засіб попередження конфліктів

Цей процес являє собою *Collision Avoidance* (попередження конфліктів), тому що ведучий пристрій, що «переміг», від спроби отримання доступу конкурентом нічого не отримує. Це є протилежністю до іншого процесу, який називають *Collision Detection* (виявлення конфліктів), під час якого обидва ведучі пристрої мають звільнити шину. За допомогою процесу попередження конфліктів підвищується пропускна здатність шини, тому що доступ завжди отримує тільки один ведучій пристрій.

9.3.3.4. Адресація користувачів шини (I^2C – блоки)

Загальні поняття

Кожен з блоків, що мають зв'язок з шиною, повинен мати певну адресу (адреса підлеглого пристрою), щоб його можна було відрізнити від інших блоків. Блоки, котрі виконують функції тільки ведучих пристроїв, не потребують адреси. З самого початку концепт шини I^2C передбачав 7-бітові адреси. За допомогою 7 бітів максимально можна підключити 128

різноманітних блоків. Певні адреси цього адресного простору зарезервовані, таким чином кількість блоків ще більше скорочується. Philips специфікував розширення адресного простору, при якому довжина адреси складає 10 біт. Але, до моменту видання цієї книги ще не було таких інтегральних мікросхем, котрі можуть працювати на 10-бітовій адресації. Таку адресацію використовують тільки мікроконтролери з певним програмним забезпеченням. Оскільки 7- та 10-бітову адресу можна використовувати одночасно, 10-бітовий режим здається трохи перенавантаженим. Відповідні правила протоколу через це виглядають дуже складними. Далі адресація ускладнюється за допомогою спеціального метода – Start Byte Procedure. Ідея цього методу полягає в спрощенні програмного забезпечення підлеглого блока без використання спеціальної апаратної частини шини I^2C завдяки відповідному заголовку для бажаної транзакції. Однак на практиці цей метод не застосовується.

7-бітова адресація

Якщо ведучий пристрій хоче з'єднатися з веденим, спочатку він має отримати за допомогою умови старту передачі доступ до шини (арбітраж). Потім він надсилає перший байт, 7 перших бітів якого мають адресу бажаного веденого пристрою. Останній біт має необхідний напрям передачі (0 = запис, 1 = зчитування). Формат такого байта показано на рис. 9.12.

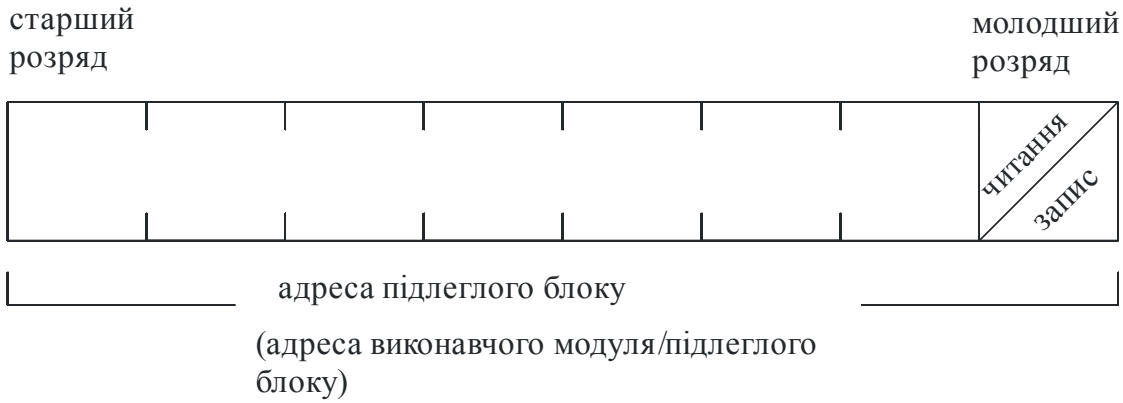


Рис. 9.12. Формат бітів адресації

Після видання умови початку передачі всі підключені ведені пристрої мають контролювати шину, щоб прочитати цей байт. При цьому повільно функціонуючі, підпорядковані пристрої можуть примусити ведучий пристрій чекати. Після того, як всі 8 бітів отримано, ведені пристрої порівнюють отриману адресу з власною. Вона або попередньо задана для певного блоку, або встановлюється DIP-перемикачами, перемичками або спаяними містками. Часто ці варіанти комбінуються. Блоки, котрим адреса не підходить, відмикаються від шини до наступного сигналу початку передачі даних. Якщо адреса підійшла, блок, що зреагував, генерує підтвердження приймання інформації (див. розділ «Передача байтів»). Тепер ведучий пристрій знає, що ведений з адресою, що запитується, готовий до передачі. Якщо підтвердження

не надійшло, то або даний блок зайнятий, або блока з заданою адресою не існує.

Розподіл адресних байтів

У посібнику вже розглядалось, що початкові сім бітів першого байта мають адресу блока, який підключений до шини I^2C, з котрим потрібно з'єднатися. Адреса 0 необхідна для особливих випадків. Якщо надсилається нульова адреса разом з бітом напряму передачі 0 (для запису), це називають «загальним викликом» («General Call»). Тому адреса 0 також називається «адресою загального виклику» («General Call Address»). За допомогою цієї адреси розпочинається широкомовлена розсилка, тобто розсилаються пакети даних, призначених для групи користувачів шиною I^2C. І звісно, цей пакет даних може пересилатись тільки від ведучого пристрою до веденого, але не навпаки.

9.3.4. CAN (Controller Area Network –мережа контролерів)

9.3.4.1. Основи

Відтоді, як з 1980 року електроніка почала займати все більше місця в автомобільній галузі, підвищувалась потреба в комунікації між окремими керуючими блоками. Звичайна проводка вже не відповідала цим вимогам. Компанія Bosh, як великий виробник керуючих блоків, дуже рано виявила потребу у винаході абсолютно нової, спеціально призначеної для використання в автопромисловості, шинної системи. Таким чином з'явилась мережа контролерів CAN (Controller Area Network). Важливими завданнями при її виробництві були надійність, низька вартість, ефективний протокол і можливість функціонування за наявності одночасно кількох ведучих пристроїв. В кінці вісімдесятих з'явився перший CAN-контролер, розроблений фірмою Intel. Протокол CAN було регламентовано у міжнародному стандарті ISO 11898.

Тим часом CAN затвердився як стандарт в організації мереж для автомобілів і почав використовуватись майже всіма заводами-виробниками транспортних засобів. В автоматиці, де часто переважають такі самі вимоги до шинної системи, як і в автомобільній галузі, CAN широко застосовується з початку дев'яностих. На базі CAN (мережевий рівень 1+2) було створено багато відкритих протоколів на рівні 7 з частково власними групами користувачів в автоматиці, до того ж в цій галузі тоді можна було зустріти і протоколи, що належали певним фірмам. Міжнародна організація користувачів та виробників CAN in Automation (CiA) – CAN в автоматизації – виробляє в усьому світі засновані на CAN програми та пристрої, особливо в автомобільній промисловості. CiA виконує функції нормування, маркетингу та розвитку CAN. Відповідно до даних CiA, в усьому світі виробляється та відповідно застосовується більше міліарда CAN-вузлів.

9.3.4.2. Фізичні характеристики

CAN має лінійну архітектуру. Як середовище передачі даних застосовується вита пара з електричними рівнями, що відповідають ISO 11898. Згідно з цим, максимальна протяжність складає 40 метрів при максимальній швидкості передачі в 1 Мбіт/с. Однак при знижені швидкості передачі даних до 50 Кбіт/с мережа може мати довжину до 1000 метрів. Принцип підключення користувачів до шинної лінії за допомогою різниці напруг є аналогічним вже розглянутим вище принципам функціонування інтерфейсу RS 485, котрі зображені на рис. 9.6.

9.3.4.3. Характеристики протоколу рівня передачі бітів

На рівні передачі бітів CAN використовується метод CSMA/CA, основи якого вже було розглянуто на прикладі шини I^2C (див. розділ 9.3.3.3 «Арбітраж доступу»).

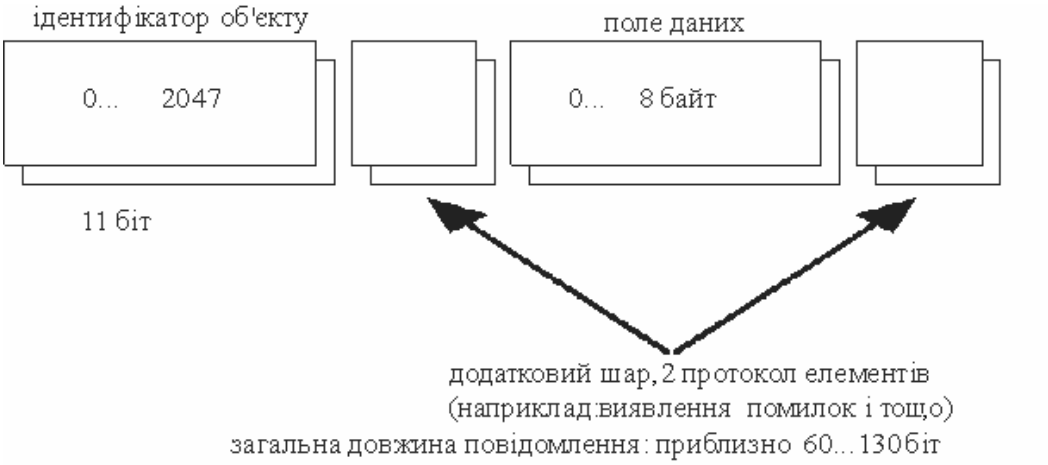


Рис. 9.13. Структура телеграми стандарту CAN

Поле даних може мати різну довжину від 0 до 64 бітів і мати інформацію, призначену для передачі.

Поле «Ідентифікатор об’єкта» (11 біт) має декілька функцій. Воно служить як мітка для телеграми. Через це воно і використовується для регулювання доступу до шини, тобто для ліквідації колізій. Дане поле надає сигналу певний пріоритет. Останній, в свою чергу, тим вище, чим менше сигналів логічної 1 має поле. Таким чином, найвищий пріоритет у сигналі з ідентифікатором об’єкта 0.

CAN – це мережа з кількома ведучими пристроями, в якій сигнали різного пріоритету передаються широкомовлено. Її ключовий концепт – це спосіб розподілу доступу до шини, заснований на попередженні конфліктів (CSMA/CA). Далі ми розглянемо принцип функціонування розподілу доступу до шини (рис. 9.14).

Якщо шина вільна, будь-яка станція може розпочинати передачу сигналу, в якому вона надсилає заголовок телеграми. Він містить ідентифікатор, спеціально призначений для даного сигналу. Процес арбітражу тепер залежить

від цього ідентифікатора. Під час арбітражу доступу рецесивні біти ідентифікації («1») переписуються від домінантних бітів ідентифікації («0») іншого вузла, який в той самий час намагається отримати доступ до шини. Кожен передавач даних одночасно чекає сигналу від шини і одразу ж перемикається з функції передачі на отримання даних, як тільки зчитує домінантний біт, в той час як він надсилає рецесивний біт. Вузол відправлення інформації з найбільш високим ідентифікатором пріоритету виграє арбітраж і таким чином отримує доступ до шини. Його сигнал відправляється без втрати часу. Таким чином, арбітраж доступу не є деструктивним. Після програного арбітражу автоматично відбувається нова спроба доступу до шини.

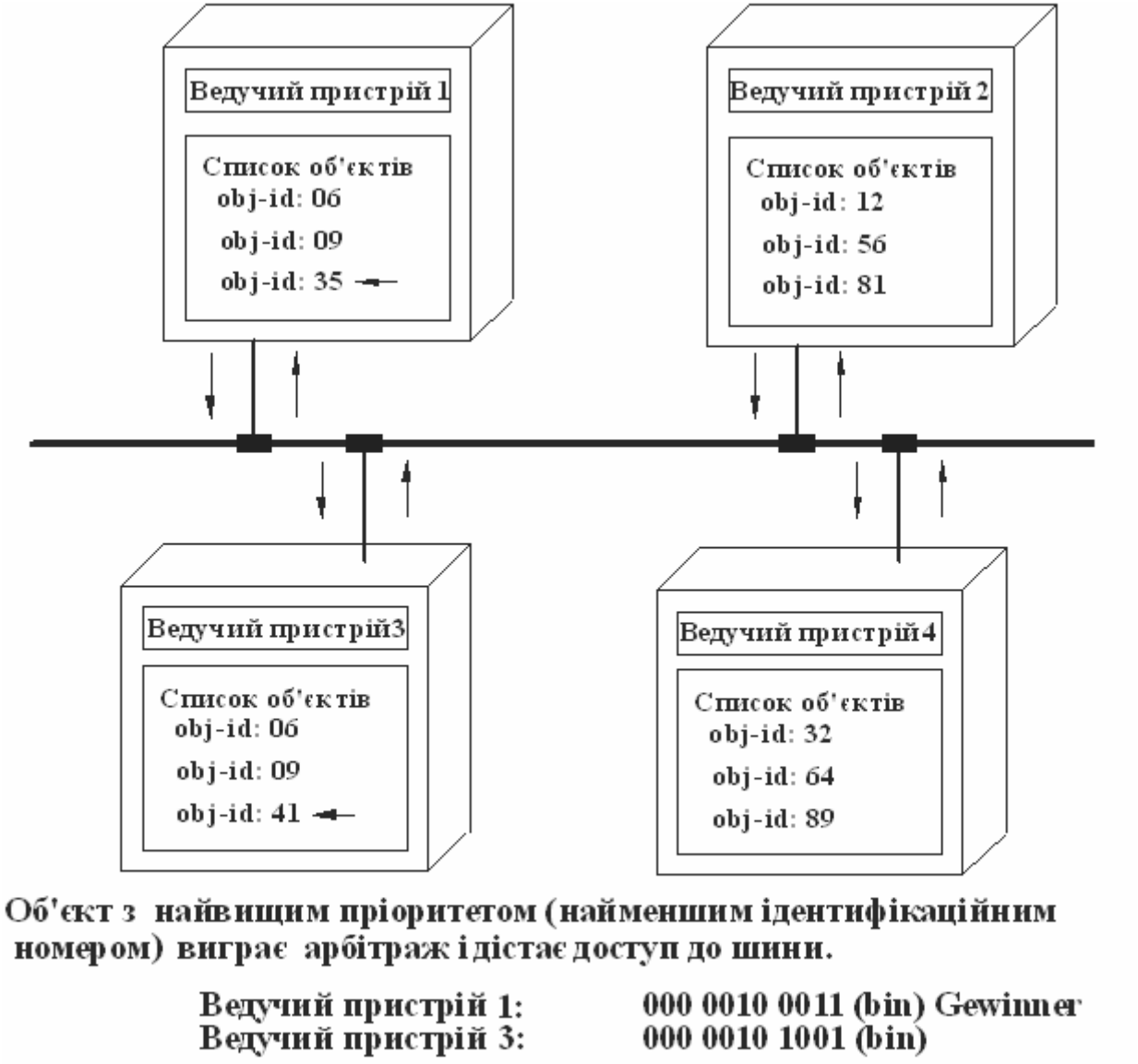


Рис. 9.14. Процес доступу до загальної шини CAN і недеструктивний арбітраж

Це означає, що сигнал з найвищим пріоритетом затримується тільки доти, доки сигнал, що відправляється в даний момент, не буде доставлено. Таким чином, для цього сигналу може бути задано максимальний час пошуку. Однак, доступ до шини сигналів з більш низьким пріоритетом при дуже високому її

завантажені може бути на деякий час ускладнено. Таким чином, CAN розподіляє доступну потужність передачі сигналів відповідно до їх пріоритету, а саме незалежно від завантаженості шини.

Принцип арбітражу доступу CAN передбачає, що максимальний час проходження сигналу в мережі значно менший, ніж час біта, тому що за час передачі біта кожен вузол має встигнути передати свою реакцію. Тому максимальна протяжність мережі тут залежить в більшій мірі від обраної швидкості передачі даних, ніж в інших шинних системах, в яких пригнічення сигналу частіше всього є обмежуючим фактором.

CAN має високу надійність передачі інформації. Арбітраж, фільтрація сигналів, виявлення та попередження помилок та підтвердження отримання сигналу реалізовані в апаратній частині. Для виявлення помилок CAN використовує комбінацію з п’яти різних механізмів: контроль розрядного рівня, контроль циклічного надлишкового коду (ЦНК), контроль підтвердження, заповнення бітів та контроль в межах сигналу.

Вузли мають систему самоконтролю та самостійно перемикаються при певній частоті помилок з *error active* на *error passive* (дані режими відрізняються засобом обробки помилок) і, нарешті, на *bus-off* режим.

Згідно (ISO) нормам 11898 мережа CAN специфікована тільки до ISO/OSI-Рівеня 2. На цьому інтерфейсі контролери можуть обмінюватись сигналами, що містять до восьми бітів корисної інформації, та ідентифікаторами.

9.3.4.4. Протоколи застосування на базі CAN

Специфікації ISO/OSI рівня 2 не достатньо для обміну корисною інформацією між кількома користувачами та її інтерпретації. Потрібні подальші інструкції, наприклад: які ідентифікатори об’єкта (тобто, пріоритети), до яких сигналів застосовуються, і яку структуру та значення вони мають в полі даних переданих бітів. Ці та подальші інструкції визначаються на більш високих рівнях протоколу. Вони розроблюються самостійно, або використовують вже існуючі стандартизовані протоколи. В транспортних засобах, як правило, використовуються розроблені виробником закриті протоколи застосування. В інших сферах застосування, перш за все в автоматиці, більш раціонально використовувати стандартизовані та відкриті протоколи застосування, тому що з їх допомогою можна зекономити витрати на вдосконалення і без будь-яких складностей експлуатувати пристрої різних виробників на загальній шині.

На основі специфікації рівня 2 від CAN було створено багато відкритих стандартів рівня застосування. Основні відкриті вищі протоколи в сфері автоматизації та інших споріднених сферах – CANopen та DeviceNet.

Протокол CANopen дуже поширений та застосовується в багатьох інших сферах окрім автоматизації.

На рис. 9.15 зображено базовий концепт комунікації пристрою з інтерфейсом CANopen.

CANopen виділяє два основних класи сигналів: Service Data Objects (SDO) з низьким пріоритетом та Process Data Objects (PDO) з високим пріоритетом. Пріоритет, тобто номер ідентифікатора об’єкта, що використовується, встановлюється в специфікації CANopen для різноманітних пристроїв та застосувань. Разом з цим існують ще спеціальні сигнали з вже визначеними ідентифікаторами, як наприклад сигнал синхронізації (SYNC).

Сигнали SDO використовують для передачі великого набору даних, як, наприклад, файлів або наборів параметрів. При цьому дані можуть сегментуватись, тобто розділятись на декілька сигналів. PDO використовують для швидкої передачі невеликих наборів даних з високим пріоритетом сигналу.

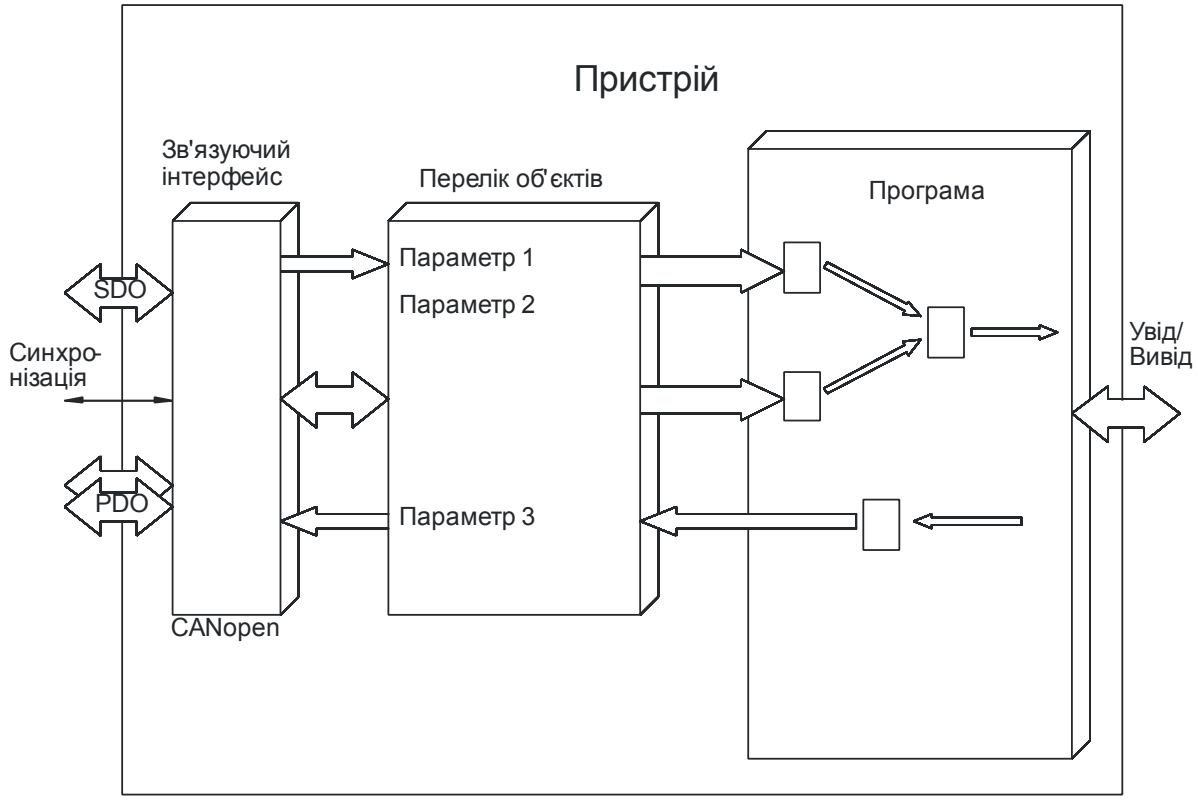


Рис. 9.15. Концепт комунікації пристрою CANopen

В пристрої існує так званий словник об’єктів. Це список, в якому містяться інструкції для структури, значення та інтерпретації переданих корисних даних. Тепер ідентифікатори об’єкта переданих сигналів (SDO, PDO та ін.) можуть використовуватись як індекс в записах словника об’єктів, і завдяки цьому поля даних можуть бути інтерпретовані відповідно до збережених інструкцій. Це звісно стосується не тільки прийнятих сигналів. Також сигнали, котрі мають бути надіслані, кодуються у відповідності до попередньо внесених інструкцій. Це дуже великий та універсальний концепт комунікації.

Специфікації CAN open існують для більшої кількості пристроїв різного типу, але основні механізми комунікації (SDO, PDO та ін.) завжди однакові.

9.3.4.5. Основні характеристики CAN

В табл. 9.1 ще раз зібрані основні характеристики мережі CAN

Таблиця 9.1

Основні характеристики CAN

Походження	Розроблена компаніями Bosch и Intel для застосування в сфері автоматизації (головним чином), стандартизована в ISO 11898
Ієрархія	Система з кількома ведучими пристроями
Топологія	Лінія (шина)
Швидкість передачі даних	До 1 Мбіт/с
Шинне середовище	ISO 11898, оптоволокно та ін.
Максимальна довжина шини	40 м при 1 Мбіт/с, до 2000 м при швидкості 25 Кбіт/с з 1 повторювачем сигналу
Доступ до середовища	CSMA/CA (недеструктивний арбітраж)
Кількість вузлів шини	ISO 11898: не вказана кількість; стандартно приблизно 30, розширені драйвери (100 і більше), доступні повторювачі
Адресація	Об’єктно-орієнтована (за ідентифікатором)
Довжина сигналу	дані от 0 до 8 байтів, загальна – приблизно130 біт
Ефективність кадру даних	0 – 53 % для довжини кадру даних 0 – 8 байтів
Пріоритети	2032 при 11 ідентифікаторах біта, $2032*2^{18}$ при 29 ідентифікаторах біта (розширена мережа CAN)
Механізми виявлення та попередження помилок	Контроль за допомогою циклічного надлишкового кода, визначення розрядного рівня, заповнення бітами, визначення довжини інформаційного кадру, обробка помилок в активному режимі, відстань Хемінга:6
Застосування	Автомобілі (легкові, вантажні, автобуси), промислова та будівельна автоматизація, ткацькі верстати, медичне устаткування, верстати та ін.

9.3.5. MODBUS

MODBUS також є шинною системою, що застосовується в промисловій автоматизації, котра підтримується більшою кількістю фірм. Як середовище передачі даних, використовується екранована двопровідна лінія. Устаткуванням для передачі виступає оптимізований інтерфейс RS485 зі швидкістю передачі 1 Мбіт/с. Спосіб отримання доступу до шини Modbus Plus – передача маркера. Топологія являє собою фізичну лінію з максимальною загальною протяжністю 450 м. При використанні повторювачів сигналу кількість користувачів може бути

збільшена до 64. Протяжність в такому випадку збільшується до макс. 1800 м. Є можливість створити так звану «двокабельну мережу», завдяки якій користувачі мережі пов'язані між собою подвійною витою парою. За допомогою цього підвищується захищеність від відмов через пошкодження кабелю або помилок за бітами – користувачі постійно контролюють обидві лінії. В подібній техніці кабельного з'єднання повторювачі сигналів мають підключатись завжди між рівними користувачами (див. рис. 9.17).

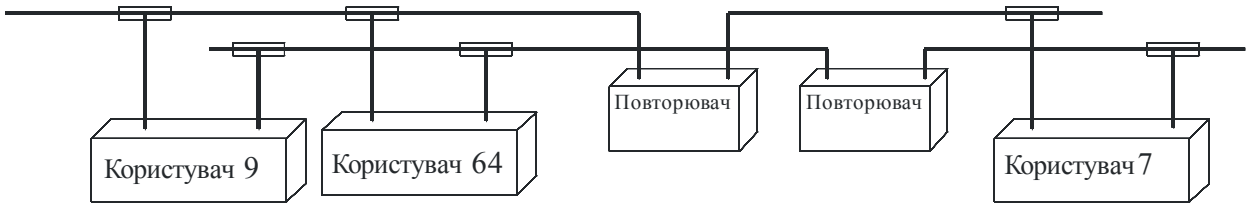


Рис. 9.17. Двокабельна мережа

За допомогою спеціальних повторювачів топологія може бути розширена, зіркоподібна або деревоподібна.

Відповідно до еталонної моделі ISO/OSI в Modbus Plus реалізовано разом з рівнями 1 і 2 також рівень 3. Мережі можуть з'єднуватись шляхом мостів. В такому випадку кожна мережа має свій власний маркер та функціонує незалежно від інших мереж. За допомогою реалізації рівня 3 сигнали можуть передаватись через один або кілька мостів. Міст є активним користувачем шини з двома незалежними одна від одної адресами. (див. рис. 9.18).

Якщо від одної мережі до іншої надсилається сигнал, то міст отримує відповідний сигнал, зберігає його та надсилає далі, як тільки сигнал отримав маркер іншої мережі. Таким чином можна посилати сигнали через 4 мости. Передача сигналу через різні мережі має задаватись прикладною програмою. Шляхом раціонального використання мостів можна знизити кількість користувачів мережею і значно підвищити потужність та захищеність від відмов кожної мережі окремо та відповідно до всієї системи в цілому. Адреси користувачів не залежать від їх фізичного положення в мережі, до того ж в одній логічній кільцевій схемі не має бути двох однакових адрес.

Якщо два користувачи в одній кільцевій схемі мають однакові адреси, це передається прикладній програмі.

Коли інсталюється мережа, кожен користувач складає список користувачів. Кругообіг маркера починається з користувача, що має найнижчу адресу, та передається завжди далі на наступну більш високу за значенням адресу. Користувач з найвищою адресою передає маркер користувачу з найнижчою адресою назад. Під час функціонування користувачі можуть вмикатися або вимикатися. Коли один користувач лишає мережу, генерується новий маркер. Процес ініціалізації триває до 100 мс.

Якщо підключається новий користувач, його буде внесено до списку адрес інших користувачів вже через 15 с (найгірший випадок, зазвичай 5 с). Якщо користувач отримує маркер, то в нього є можливість підключатись до

шини. Він може надсилати сигнали довжиною максимум 512 на одного користувача. Є можливість так зконфігурувати користувача, що він зможе послати в цілому 8 Кбіт різним користувачам. На сигнал завжди має надійти відповідь про його отримання. Якщо від відправника не прийнято сигнал підтвердження, той самий сигнал буде повторюватись. Якщо під час другої спроби сигнал підтвердження відправника не надходить, він надсилає повідомлення про помилку, яке може бути запитано додатком.

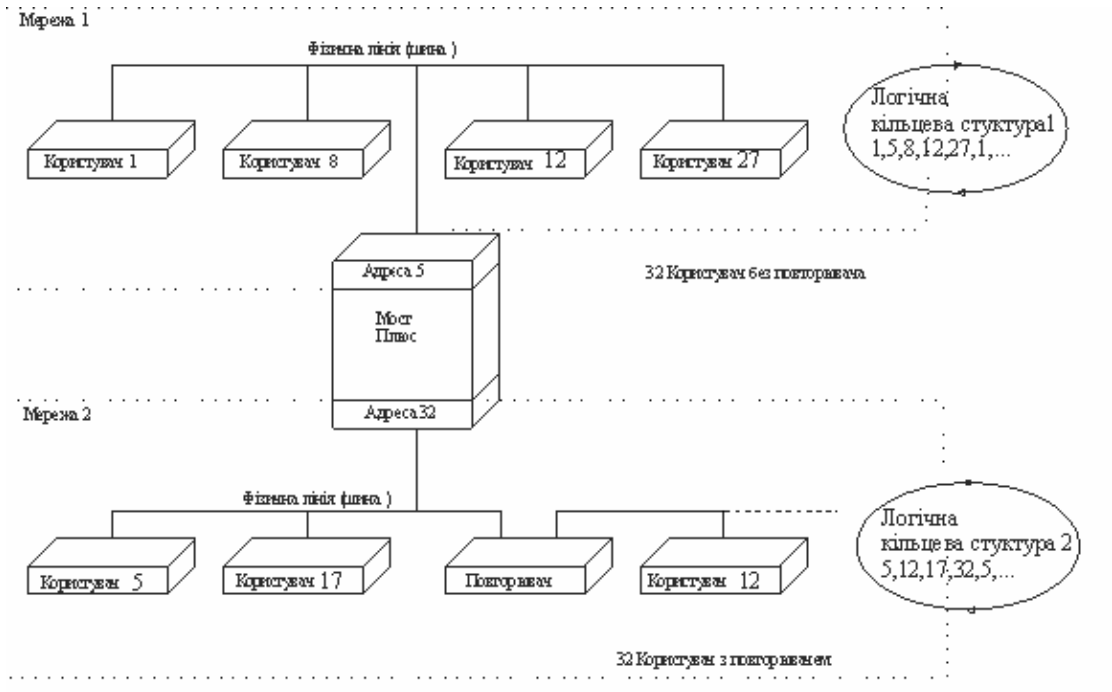


Рис. 9.18. Фізична та логічна структура Modbus

При передачі маркера одночасно може бути надіслано до 512 біт широкомовлених повідомлень. Вони знаходяться у фреймі маркера і зберігаються в глобальній базі даних з інформацією про їх відправника. У кожному логічному колі є власна база даних, до якої мають доступ тільки користувачі кола.

Після передачі маркера користувач слідкує за шиною. Якщо від отримувача маркера не надходить жодної реакції, то маркер передається вдруге. Якщо отримувач і після цього не реагує, то мережа заново ініціалізується, починається новий кругообіг маркера. Поруч з цими механізмами попередження помилок передається також 16-бітний циклічний надлишковий код.

Додатково до модифікації Modbus існує також Modbus TCP/IP на основі оптимізованого з’єднання RS 485. Тут в якості фізичного середовища передачі використовується Ethernet. Застосовується встановлений на нього протокол передачі TCP/IP. Таким чином, виникає апаратна сумісність зі стандартними системами Ethernet. Застосування стеків протоколу TCP/IP в програмному забезпеченні приводить до скорочення витрат на його вдосконалення. В сфері шарів користувачів протоколу також потрібні спеціальні реалізації Modbus.

Контрольні запитання

1. Скільки об’єктів можна підключити до шини?
2. Перелічити особливості послідовних шин.
3. Які різновиди польових шин ви знаєте?
4. Інтерфейс RS 232. Назвати приклади використання та логічні рівні.
5. Скільки користувачів можна підключити до інтерфейсу RS 485?
6. Призначення шини I^2C.
7. Скільки ліній необхідно для роботи шини I^2C?
8. Як кодуються умови початку та кінця передачі по шині I^2C?
9. Як виконується арбітраж CAN шини?
10. Які протоколи зв’язку використовують CAN шину?
11. Яким чином отримують доступ до шини Modbus?
12. Яку топологію може мати шина Modbus?

10. ПРИКЛАДИ РОЗРОБКИ СИСТЕМ КЕРУВАННЯ НА БАЗІ МК-51

10.1. Система керування з неперервними характеристиками

Організацію вводу/виводу аналогових сигналів розглянемо на прикладі керування нагрівальною піччю для сушіння керамічних заготовок. Об'єкт керування являє собою камеру, у якій нагрівання забезпечується електронагрівником. Керування потужністю нагрівача проводиться за допомогою широтно-імпульсної модуляції (ШІМ) з періодом від 1 с до 200 с. Контроль температури виконується за допомогою терморезистора з перетворювачем, який має лінійну струмову характеристику від 4 до 20 mA. При 0˚С – 4 mA, при 400˚С – 20 mA.

Функціональна схема такої системи регулювання наведена на рис. 10.1.

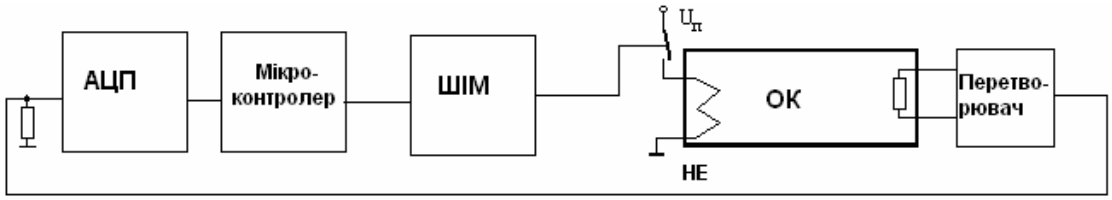


Рис. 10.1. Функціональна схема керування сушильною піччю

Автоматичний ПІД-регулятор реалізовано на MCS-51. Схема підключення АЦП наведена на рис. 10.2.

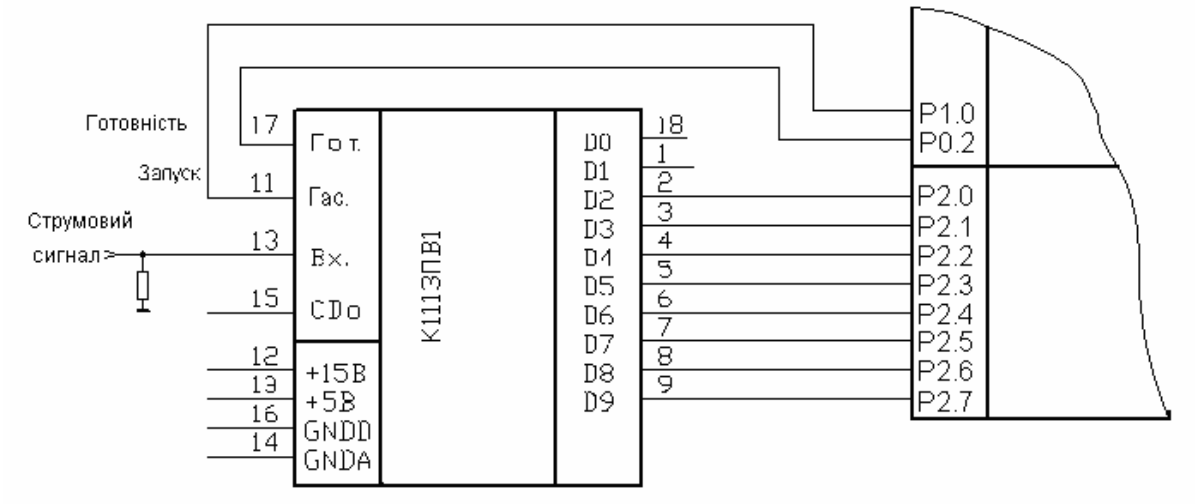


Рис. 10.2. Схема підключення АЦП до мікроконтролера

Подібні системи відносяться до класу неперервних. У таких системах вихідний параметр об'єкта керування може займати нескінченну кількість значень або станів.

Найбільше поширення одержали системи керування із заданням регульованого сигналу. У цих системах регульована змінна $Y(t)$ повинна якомога точно відпрацьовувати сигнал $w(t)$, тобто забезпечувати мінімальну помилку керування.

$$\Delta = w(t) - y(t).$$

Системи, у яких задавальна змінна може змінюватися в часі, називаються слідкувальними. Якщо задавальна змінна постійна, то така система має назву стабілізації.

Розрізняють системи із прямим і зворотним зв'язком. Структурну схему системи із прямим зв'язком показано на рис. 10.3.

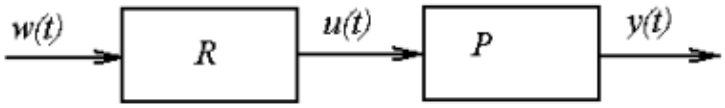


Рис. 10.3. Структура системи із прямим зв'язком

Такі системи застосовуються, коли значення параметра на виході об'єкта регулювання пов'язано із вхідною величиною детерміновано, включаючи збурювання, аналітичним виразом або табличною відповідністю. Якщо ж у процесі роботи на систему впливають процеси, які не можна врахувати, використовують системи зі зворотними зв'язками. Їх структурну схему зображено на рис. 10.4.

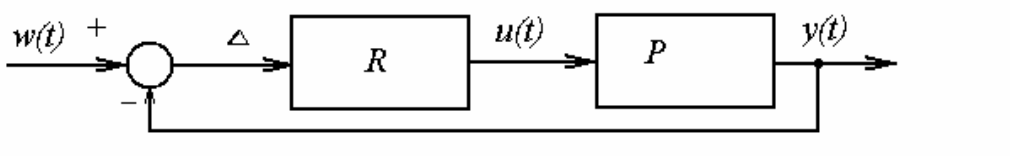


Рис. 10.4. Структура системи зі зворотним зв'язком

Дослідження подібних систем розглядається в курсі ТАК – в розділі «лінійні системи керування». Задання якості регулювання в таких системах забезпечується використанням ПІД-регуляторів і їх модифікацій.

Функція керування (ФК) у таких регуляторах обчислюється за наступною формулою:

$$u(t) = k_П \cdot \Delta + 1 / T_И \int \Delta dt + T_Д d\Delta / dt,$$

де: $k_П$ – коефіцієнт підсилення пропорційного регулятора; $T_И$ – постійна часу при

інтегральній частині; $T_Д$ – постійна часу при диференціальній частині; Δ – різниця між заданням і поточним значенням вихідного параметра.

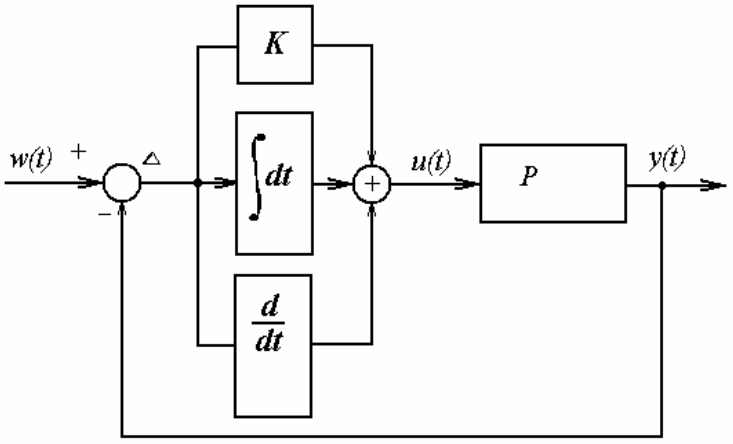


Рис. 10.5. Структурна схема системи керування з ПІД-регулятором

При переході до цифрової системи керування неперервні перетворення інтегрування й диференціювання заміняються їхніми дискретними аналогами. Оскільки:

$$d\Delta = \left[w(k) - Y(k) - \left(w(k-1) - Y(k-1) \right) \right],$$

то обчислення функції керування здобуває вигляд:

$$u(k) = k_П \Delta(k) + 1 / T_И \sum_{i=1}^{k} \left(\Delta(i) \Delta t \right) + T_d \left(Y(k-1) - Y(k) \right) / \Delta t ,$$

де: Δt – період квантування або час такту; Y – вихідне значення.

У наведеній формулі для обчислення суми необхідно зберігати значення помилки регулювання на попередніх кроках, тому використовують рекурентний спосіб обчислення інтеграла.

$$u(k) = k_П \Delta(k) + I(k-1) + \Delta(k) \Delta t / T_И + \left(Y(k-1) - Y(k) \right) T_d / \Delta t ,$$

де: $I(k-1)$ – значення інтегральної частини регулятора на попередньому кроці.

Для мінімізації обчислень заміняємо:

$$k_И = \Delta t / T_И ; \qquad k_Д = Td / \Delta t .$$

Остаточний вираз для цифрового ПІД-регулятора має вигляд:

$$u(k) = k_П \Delta(k) + I(k-1) + \Delta(k) k_И + \left(Y(k-1) - Y(k) \right) k_d \ .$$

Обчислення функції керування для ПІД-регулятора складається з декількох етапів і повинно виконуватися відповідно до діаграми, наведеної на рис. 10.6.

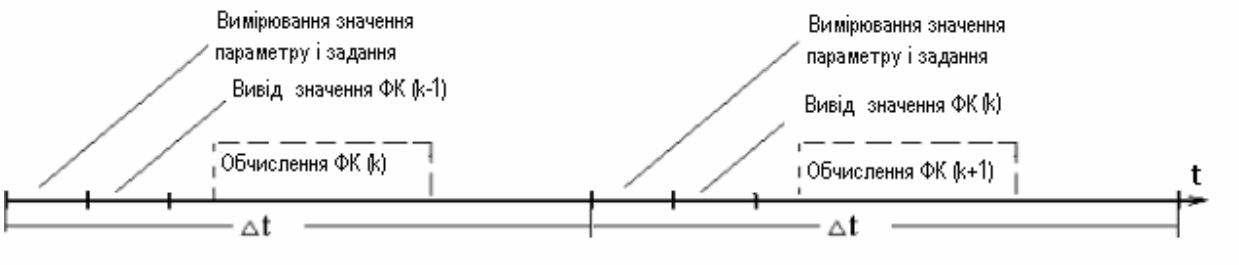


Рис. 10.6. Діаграма формування функції керування

Початок кожного такту ініціюється відмітником часу (системним таймером). Вимір поточного значення вихідного параметра, визначення заданого значення регульованої величини й висновок, обчисленого на попередньому кроці значення функції керування, проводиться в самому оброблювачі переривання від таймера. Цим забезпечується мінімальна тимчасова погрішність виміру й формування керуючого сигналу. Обчислення функції керування не вимагає такої твердої прив'язки до часу, тому може бути виконано в будь-який момент часу між викликами переривання від таймера.

Як правило, цифрова реалізації ПІД-регулятора виконується в системі, у котрій реалізовано режим багатозадачності. У цьому випадку задання ПІД-регулятора виконується на тлі задань дискетного керування, індикації, обміну за послідовним портом і т.п. Виклик задань у таких системах реалізовано з використанням циклічного алгоритму планування, який має на увазі почергове виконання задань. Підвищення продуктивності сучасних мікроконтролерів дозволяє досить швидко обчислювати функцію керування навіть у багатозадачному режимі. Але для неперервних систем важливо не швидко виконувати їх обчислення, а вчасно, тому для правильного функціонування ПІД-регулятора після однократного виконання задання протягом такту вони повинні бути заблоковані.

У процесі вводу значення регульованої величини з об'єкта керування й заданого значення регульованої величини, а також під час виводу значення функції керування сигнали перетерплюють кілька перетворень. Тому для правильного обчислення функції керування це потрібно враховувати. Наприклад, при керуванні нагрівальною піччю для сушіння керамічних заготовок контроль температури виконується за допомогою терморезистора з перетворювачем, який має лінійну струмову характеристику від 4 до 20 mA. При 0˚C – 4 mA, при 400˚C – 20 mA.

Розглянемо проходження сигналу з датчика температури. Вихідний, струмовий сигнал з перетворювача температури перетворюється в напругу, це виконується за допомогою прецизійного резистора. Далі ця напруга

перетворюється в код за допомогою аналогово-цифрового перетворювача. За умови лінійності системи виміру температури для визначення коефіцієнтів лінійного перетворення досить двох крапок. Візьмемо 8-розрядний АЦП із діапазоном перетворення від 0 до 5,12 В. При температурі 0˚С на виході перетворювача буде струм 4 mA. При опорі навантаження для перетворювача 240 Ом це відповідає напрузі на вході АЦП в 0,96 В. При температурі 400˚С струм перетворювача – 20 mA й на вході АЦП – 4,8 В. Визначимо крок квантування АЦП: $\Delta x = 5{,}12 / 256 = 0{,}02\ B$.

Тоді 0˚С на виході АЦП буде відповідати значення: $0{,}96 / 0{,}02 = 48$, а $400^{\circ}\text{C} \rightarrow 4{,}8 / 0{,}02 = 240$.

Для визначення коефіцієнтів лінійного перетворення $\left(y = k \cdot x + B\right)$ будуємо систему із двох рівнянь:

$$400 = k240 + B$$
$$0 = k48 + B\,.$$

Для одержання значення коефіцієнта k віднімемо від першого рівняння друге:

$$400 = k\left(240 - 48\right) \rightarrow k = 400 / 192 \approx 2{,}0833\,.$$

Далі з першого рівняння одержуємо В:

$$0 = 2{,}0833 \cdot 48 + B \rightarrow B = -2{,}0833 \cdot 48 = 100\,.$$

Округляємо k до 2, одержуємо, що для обчислення значення температури в градусах Цельсія отримане з АЦП значення необхідно помножити на 2, а потім від добутку відняти 100.

Вихідним сигналом функції керування в цій системі буде тривалість імпульсу ШІМ. Період ШІМ буде визначати час кванта. Значення тривалості імпульсу на виході ПІД-регулятора буде виражено у відсотках від максимального значення періоду ШІМ. При тривалості імпульсу, що дорівнює періоду ШІМ, на нагрівач подається максимальна потужність. При такій побудові регулятора під час налагодження легко можна перевірити правильність функціонування регулятора. Наприклад, при дослідженні пропорційного регулятора задане значення регульованої величини – 50˚С, змінне значення температури в камері – 40˚С, коефіцієнт підсилення пропорційної ланки – 5, інтегральний коефіцієнт і диференціальний дорівнюють 0, вихідний сигнал регулятора – 50 %. Якщо період ШІМ – 200 с, це буде відповідати тривалості імпульсу 100 с. Розглянемо ті самі умови для інтегральної частини. Якщо при розрахунках постійної інтегрування для аналогового регулятора вийшло значення, наприклад, 100 с, то для ШІМ с періодом 200 с коефіцієнт інтегратора буде дорівнювати:

$$k_И = \Delta t / T_И = 200 / 100 = 2 .$$

При різниці між поточним значенням температури й заданням 10˚С збільшення тривалості імпульсу на кожному періоді буде дорівнювати 20 %.

```
;Програма керування сушильною камерою
 Block equ 0; біт блокування завдання ПІД-регулятора
 tmra equ 37h; температура з АЦП
 t1c equ 30h; дільник до 1 з
 Tshim equ 31h; період ШІМ
 timp equ 32h; тривалість імпульсу
 ACP equ 33h; значення з АЦП
 Rez_pid equ 41h; вихід ПІД
 zadan equ 80; задання 80 град Цельсія
 zint equ 22h; знак інтегратора поточний
 zdelt equ 21h; знак погрішності
 kprop equ 5; коефіцієнт пропорційного регулятора
 kdif equ 3; коефіцієнт диференціального регулятора
 kint equ 2; коефіцієнт інтегрального регулятора
 delt equ 36h; погрішність регулювання поточна
 ptemp equ 37h; температура на попередньому кроці
 sprop equ 38h; старший байт пропорційного
 mprop equ 39h; молодший байт пропорційного
 iregm equ 3ah; значення інтегрального
 iregs equ 3bh; значення інтегрального
 ttemp equ 3ch; поточне значення температури
 Dregm equ 3dh; значення диференціального
 Dregs equ 3eh; значення диференціального
 spidm equ 3fh; молодший байт суми регулятора
 spids equ 40h; старший байт суми регулятора.
```

Наведений фрагмент є описом призначення символьних змінних і вказівкою, де зберігаються значення проміжних і кінцевих змінних

```
 ljmp m0      ; обходимо вектор переривання
org 0bh       ; вектор переривання від таймера ТМ0
push acc
mov th0,#3ch
mov tl0,#0b0h
djnz t1c,m2
mov t1c, #20
mov a,Tshim                ; період ШІМ
 mov th0,#3ch
 mov tl0,#0b0h
```

```
        djnz t1c,m2             ; програмний дільник
        mov t1c, #20            ; відмітник часу дорівнює 1 с.
        mov a,Tshim             ; період ШІМ 200 з
        jz m3
        dec a
        mov Tshim,a
        ljmp m4
m3:     setb P1.0;              старт АЦП
kp:     jb P0.2,kp    ; очікування кінця перетворення
        clr P1.0
        mov A,P2; зчитування значення з АЦП
        setb Block; розблокування задання ПІД-регулятора
        mov ACP,a; збереження значення АЦП
        mov Tshim,200; новий період ШІМ
        mov timp,Rez_pid; задання ширини імпульсу
        setb P1.1
        clr Block
        ljmp m2
m4:     mov a,timp; ширина імпульсу
        jz m5
        mov a,timp
        dec a
        mov timp,a
        ljmp m2
m5:     clr P1.1; закінчення імпульсу ШІМ
m2:     pop PSW
        pop acc
        reti                    ; повернення з переривання.
```

У даному фрагменті наведена підпрограма оброблювача переривання від таймера. Вона викликається апаратно за переповненням таймера кожні 50 mC. За допомогою програмного лічильника *t*1*c* період відмітника часу для ПІД-регулятора збільшується до 1 с. Змінна Tshim забезпечує значення періоду ШІМ 200 с. При досягненні змінної значення 0 запускається АЦП, зчитується значення з АСП і встановлюється значення тривалості імпульсу ШІМ (змінна timp). Після чого розблокується задання ПІД-регулятора (змінна Block).

```
;Ініціалізація таймера
m0:     mov sp,#70h
        mov tmod,#1h            ; програмування таймера 0 у режим 1
        mov th0,#3ch
        mov tl0,#0b0h
        setb tcon.4
        setb ie.7
```

```
        setb ie.1
        mov t1c,#20
        clr P1.0
        clr P1.1
        clr Block
;Основна програма
LOOP:   call Pidregul
        ljmp LOOP
```

З периферійних обладнань у програмі використовується таймер 0 і порт P1 і P0.

При ініціалізації системи таймер 0 настроюється на режим 1 з періодом переповнення 50 mC, установлюються в 0 порти висновку і початкові значення змінних. Цикл LOOP дозволяє періодично викликати задання ПІД-регулятора на виконання. У циклі перебуває всього одне задання керування, але може бути і кілька – послідовно викликуваних.

```
;Підпрограма ПІД-регулятора
Pidregul:
        jnb Block,rt; якщо заблокована на вихід
        mov A,ACP; визначаємо дійсне значення температури
        rl A
        clr c
        subb A,100
        mov ttemp,A
        subb A,zadan; визначаємо погрішність регулювання
        mov delt,a
        jnc pol
        setb C; погрішність негативна
        setb zdelt
        ljmp prop
pol:    clr C
        clr zdelt
prop:   mov B,kprop; коефіцієнт пропорційного регулятора
        call mullz
        mov sprop,B
        mov mprop,A
integrp: mov A,delt
        mov B,kint; коефіцієнт інтегрального регулятора
        mov c,zdelt; знак погрішності
        call mullz
        add a,Iregm;
        mov Iregm,a; інтегральний регулятор
        mov a,b
        addc a,Iregs
```

```
        mov Iregs,a
diff:   mov A,ptemp; диференціальний регулятор
        subb A,ttemp; різниця поточного й попереднього виміру
        mov B,kdif; коефіцієнт диференціального регулятора
        call mullz
        mov Dregm,a; молодший байт значення
        mov Dregs,b; старший
sympid: mov a,mprop; складаємо пропорційну частину
        add a,iregm; з інтегральної
        mov spidm,a
        mov a,sprop
        addc a,Iregs
        mov spids,a
        mov a,spidm
        add a,Dregm; додаємо диференціальну частину
        mov spidm,a
        mov a,spids
        addc a,Dregs
        mov spids,a
        jb Acc.7,zero
        jz gorez
        mov rez_PID,#0100; якщо результат більший за 255 на ШІМ 200
        ajmp rt
zero:   mov rez_PID,#0
        ajmp rt
gorez:  mov a,spidm
        clr c
        subb a,#100; при значенні >100
        jc gorez1
        mov spidm,#100; виводимо 100
gorez1: mov a,spidm
        rl a
        mov rez_PID,a
rt:     setb Block
        ret
```

Підпрограма ПІД-регулятора починає свою роботу з перевірки блокування задання. Якщо задання не заблоковано, зберігається попереднє значення температури й обчислюється поточне значення температури за даними, прийнятими з АЦП. Потім обчислюється помилка керування, як різниця між поточним значенням температури й заданим значенням регульованої величини.

Окремо обчислюються значення функції керування всіх регуляторів: пропорційного (мітка prop), інтегрального (мітка integrp) і диференціального

(мітка diff). Для обчислення використовується підпрограма перемножування однобайтового числа зі знаком на 7-бітове без знака. У такий спосіб коефіцієнти для даного ПІД-регулятора обмежені значенням від 0 до 128. Конкретні значення коефіцієнтів регуляторів подаються в описі змінних. Тип вихідних значень регуляторів – двобайтові зі знаком.

Далі отримані значення кожного регулятора підсумовуються (мітка sympid). Спочатку складаються побайтово значення пропорційного й інтегрального регуляторів, потім отримана сума складається зі значенням, що обчислено для диференціального регулятора, й результат перевіряється за наступними умовами. Якщо значення на виході регулятора негативне, то тривалість вихідного імпульсу дорівнює 0. Якщо воно більше 100, то йому привласнюється значення 100. Після чого вихідне значення масштабується відповідно до тривалості ШІМ (у нашому випадку множиться на 2 шляхом зрушення вліво).

```
;підпрограма множення 7-бітового без знака на 8-бітове зі знаком
;значення без знака міститься в B, зі знаком в Acc, знак у C
;молодший байт результату міститься в Acc, старший в B
mullz:  jc min;
        mul ab; позитивний результат
        ret
min:    cpl a;негативний результат
        inc a
        mul ab
        cpl a
        mov r2,a
        mov a,b
        cpl a
        mov b,a
        mov a,r2
        add a,#1
        jnc goret
        inc b
goret:  ret
```

Оскільки коефіцієнти для всіх регуляторів позитивні, то для обчислення досить ураховувати знак тільки одного співмножника – помилки регулювання. Множення негативного співмножника проводиться за наступним алгоритмом: переведення негативного значення в додатне множення додатних чисел, переведення двобайтового результату в додатковий формат. Обмеження на розрядність коефіцієнтів уведене для того, щоб старший біт результату множення не потрапив в останній знаковий розряд двобайтового числа зі знаком.

Таким чином, розглянутий приклад показує, що при роботі з неперервними об'єктами вирішуються наступні завдання:

1. Квантування сигналів за рівнем за допомогою АЦП.

2. Формування вихідних неперервного сигналу функції керування за допомогою ЦАП або ШІМ.

3. Обчислення керуючої функції з урахуванням тимчасових параметрів.

4. Визначаються коефіцієнти регуляторів за методами теорії автоматичного керування.

10.2. Система керування кроковим двигуном

Кроковий двигун (КД) є традиційним виконавчим пристроєм багатьох електронних приладів і систем. Він являє собою безколекторний двигун постійного струму з фіксованими положеннями вала. КД призначено у першу чергу для точного позиціювання вала без застосування систем зворотного зв'язку. Обмотки КД є частиною статора. На роторі розташовано постійний магніт або у випадках зі змінним магнітним опором зубчастий блок з магнітом’якого матеріалу. Усі комутації проводяться за зовнішніми схемами. Відносно до контролера відмінність між ними відсутня. На двигунах з постійними магнітами звичайно є дві незалежні обмотки.

Крокові двигуни мають широкий діапазон кутових дозволів. Більш грубі мотори, звичайно с постійними магнітами, обертаються на 90° за крок, у той час як прецизійні двигуни можуть мати дозвіл 1,8° або 0,72° на крок.

Для правильного керування біполярним кроковим двигуном необхідна електрична схема, яка повинна виконувати функції старту, зупинки, реверсу й зміни швидкості. Кроковий двигун транслює послідовність цифрових перемикань у рух. «Обертове» магнітне поле забезпечується відповідними перемиканнями напруг на обмотках. Слідом за цим полем буде обертатися ротор, з'єднаний за допомогою редуктора з вихідним валом двигуна.

Потужність крокових двигунів знаходиться у діапазоні від одиниць ватів до одного кіловата. Кроковий двигун має не менш двох положень стійкої рівноваги ротора в межах одного оберту. Напруга живлення обмоток керування кроковом двигуном являє собою послідовність однополярних або двополярних прямокутних імпульсів, що надходять від електронного комутатора (К). Результуючий кут відповідає числу перемикань комутатора, а частота обертання двигуна – частоті перемикань електронного комутатора.

Комутатор повинен забезпечувати зміну полярності струму в обмотках. Він являє собою два транзисторні мости, в діагональ яких включені статорні обмотки. На рис. 10.7 зображено положення ротора крокового двигуна залежно від комутації обмоток. Послідовно комутуючи струм в обмотках відповідно до діаграм, наведених на рис. 10.8, можна змусити обертатися вектор магнітного поля, а за ним і ротор, у прямій або зворотній послідовності. При такому керуванні двигун має 4 стійких стану на одному оберті ротора.

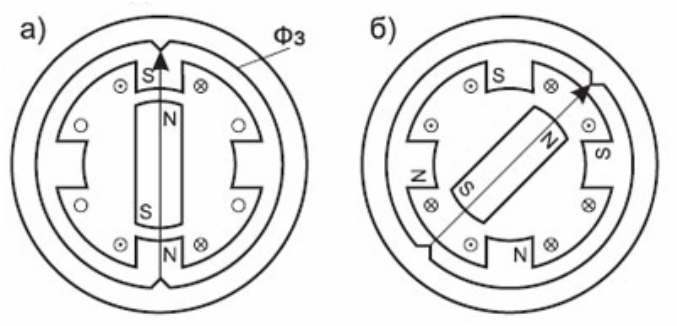


Рис. 10.7. Положення ротора при кроковому (а) і напівкроковому керуванні (б)

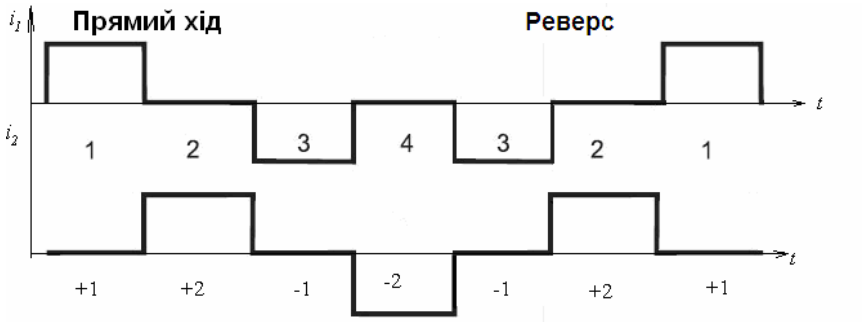


Рис. 10.8. Прямий і зворотний хід у кроковому двигуні

Одночасне включення двох обмоток приводить до орієнтування вектора магнітного поля із кроком 45° щодо вертикальної осі. Таке керування називається напівкроковим. Напівкроковий режим дозволяє вдвічі підвищити точність позиціювання крокових двигунів. Послідовність комутації обмоток двигуна в напівкроковому режимі наведена на рис. 10.9. При комутації по черзі включаються одна обмотка, а за нею дві разом і т.п. При такому керуванні двигун має 8 стійких станів.

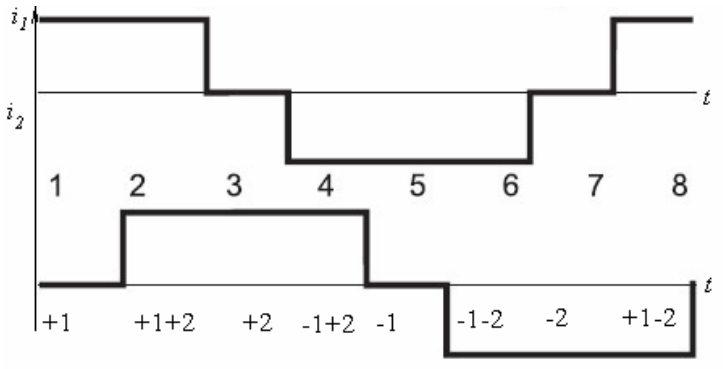


Рис. 10.9. Напівкрокове керування кроковим двигуном

У крокових приводах електрична енергія перетворюється в механічну у вигляді обертального руху. Тому рівняння руху записується як рівняння рівноваги

всіх моментів. Для більшості робочих механізмів і машин, що мають постійний момент інерції, цей вираз має вигляд:

$$M - M_C = J\frac{\omega - \omega_0}{\Delta t}, \tag{10.1}$$

де M – обертаючий момент електродвигуна, дж; M_C – статичний момент робочого механізму, наведений до вала двигуна, дж; J – момент інерції системи, наведений до вала двигуна, дж·с^2; α – кут повороту, рад.; t – час, с.

$$\frac{\omega - \omega_0}{\Delta t} \text{ – кутове прискорення, } 1/\text{с}^2; \tag{10.2}$$

$$\omega = \frac{\alpha - \alpha_0}{\Delta t} \text{ – кутова швидкість, } 1/\text{с}^2. \tag{10.3}$$

Наявність моменту інерції приводить до того, що для розгону або гальмування двигуна необхідний час, інакше вектор положення ротора почне відставати або випереджати вектор поля й двигун втратить синхронізм, що призведе або до пропуску кроків, або зупинки двигуна.

Наведемо приклад розрахунків розгону й гальмування крокового двигуна. Двигун з моментом 1500 Дж. повинен обертати об'єкт керування зі швидкістю 60 об/хв. Момент інерції об'єкта – 500 Дж. Керування двигуном проводиться в напівкроковому режимі. Необхідно визначити час виходу двигуна на задану швидкість, а також кількість кроків для розгону й тривалість кожного кроку. Розрахунки проводяться без обліку моменту опору обертанню.

Розв'язок: оскільки початкова швидкість дорівнює нулю, то $\omega' = 0$. Тоді (1) можна переписати

$$M = J\frac{\omega}{\Delta t}, \tag{10.4}$$

$$\Delta t = \omega\frac{J}{M}, \tag{10.5}$$

$$\omega = \frac{60 \cdot 2\pi}{60} = 6,28\frac{рад}{с}. \tag{10.6}$$

Звідси

$$\Delta t = 6,28\frac{500}{1500} = 2,1 \text{ с}.$$

Визначимо, скільки кроків повинен виконати двигун до повного розгону.

Перед початком розгону $\alpha = 0$

$$M = J\frac{2\alpha}{\Delta t^2}, \tag{10.7}$$

звідси

$$\alpha = \frac{\Delta t^2}{2}\cdot\frac{M}{J} = \frac{2{,}1^2}{2}\cdot\frac{1500}{500} = 6{,}62 \text{ рад.}$$

У напівкроковому режимі кроковий двигун робить 8 кроків за один оберт. Тоді один крок – це поворот на кут

$$S = \frac{2\pi}{8} = \frac{\pi}{4} \text{ рад.}$$

Виходить, до кінця розгону двигун зробить: $\frac{6{,}62\cdot 4}{\pi} = 8{,}4$ кроків.

Тривалість першого кроку буде дорівнювати, виходячи з (7),

$$t_1 = \sqrt{2\cdot\alpha_1\cdot\frac{J}{M}} = \sqrt{2\cdot\frac{\pi}{4}\cdot\frac{500}{1500}} = 0{,}72 \text{ с.}$$

Тривалість другого кроку визначимо як різницю часу повороту на два кроки мінус поворот на один крок. Тривалість третього – як різницю часу повороту на три й два кроки і т.п. Розрахунки значень зручніше за все проводити в програмі EXEL.

$$t_2 = \sqrt{2\cdot 2\cdot\alpha_1\cdot\frac{J}{M}} - \sqrt{2\cdot\alpha_1\cdot\frac{J}{M}} = 0{,}3 \text{ с;}$$

$$t_3 = \sqrt{3\cdot 2\cdot\alpha_1\cdot\frac{J}{M}} - \sqrt{2\cdot 2\cdot\alpha_1\cdot\frac{J}{M}} = 0{,}23 \text{ с;}$$

. . . .

$$t_9 = \sqrt{9\cdot 2\cdot\alpha_1\cdot\frac{J}{M}} - \sqrt{8\cdot 2\cdot\alpha_1\cdot\frac{J}{M}} = 0{,}124 \text{ с.}$$

Тривалість кроку для швидкості обертання 60 об/хв. обчислюємо в такий спосіб:

$$t_s = \frac{S}{\omega} = \frac{\pi}{4 \cdot 6,28} = 0,125 \text{ с.}$$

Таблиця 10.1

Таблиця тривалості імпульсів для розгону двигуна

№	1	2	3	4	5	6	7	8	9
T	0,72	0,3	0,23	0,19	0,17	0,15	0,14	0,13	0,12

Керування кроковим двигуном у мікропроцесорній системі керування складається із частин. Перша оформляється у вигляді підпрограми з передачею параметрів. У якості параметрів задається напрямок руху, кількість кроків. Підпрограма розраховує кількість кроків розгону й гальмування двигуна, а також кількість кроків з номінальною швидкістю. Алгоритм підпрограми, яка ініціалізує керування двигуном із прикладу розрахунків, наведений на рис. 10.10.

Друга частина розміщається в підпрограмі обслуговування переривань від таймера. У ній за відмітником часу 10 mC формуються імпульси керування струмами в обмотках. Алгоритм відпрацьовує спочатку розгін, потім роботу на номінальній швидкості, а після цього гальмування двигуна. Для визначення тривалості імпульсу під час розгону й гальмування програма використовує таблицю, записану в пам'яті програм. Алгоритм роботи цієї частини наведено на рис. 10.11. Значення на виході порту керування мостами формує підпрограма «крок», для цього вона використовує таблицю активізації виходів для кожного положення ротора. Алгоритм цієї підпрограми показано на рис. 10.12.

Схему підключення крокового двигуна до мікроконтролера зображено на рис. 10.13. Вона складається із двох транзисторних мостів, у діагональ яких включені статорні обмотки крокового двигуна ОВ1 і ОВ2. Керування мостами проводиться чотирма молодшими бітами порту P0. Коли подається одиниця на P0.0, відкривається транзистор Q3, а за ним і Q2, і через першу обмотку проходить струм позитивної полярності. Під час подачі одиниці на P0.1 відкривається транзистор Q4, а за ним і Q1, і через першу обмотку проходить струм негативної полярності. Другий міст працює аналогічно.

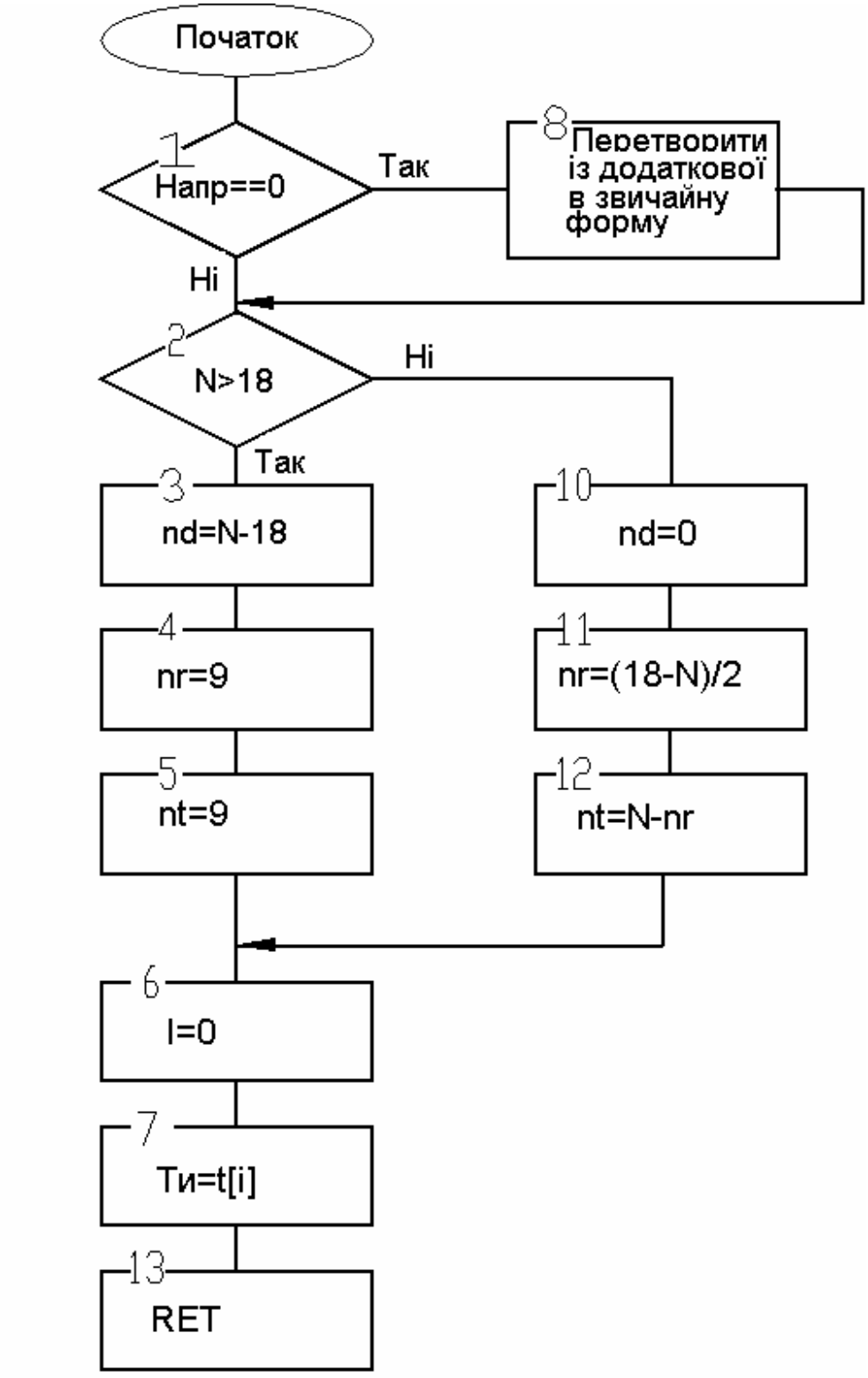


Рис. 10.10. Підпрограма ініціалізації руху

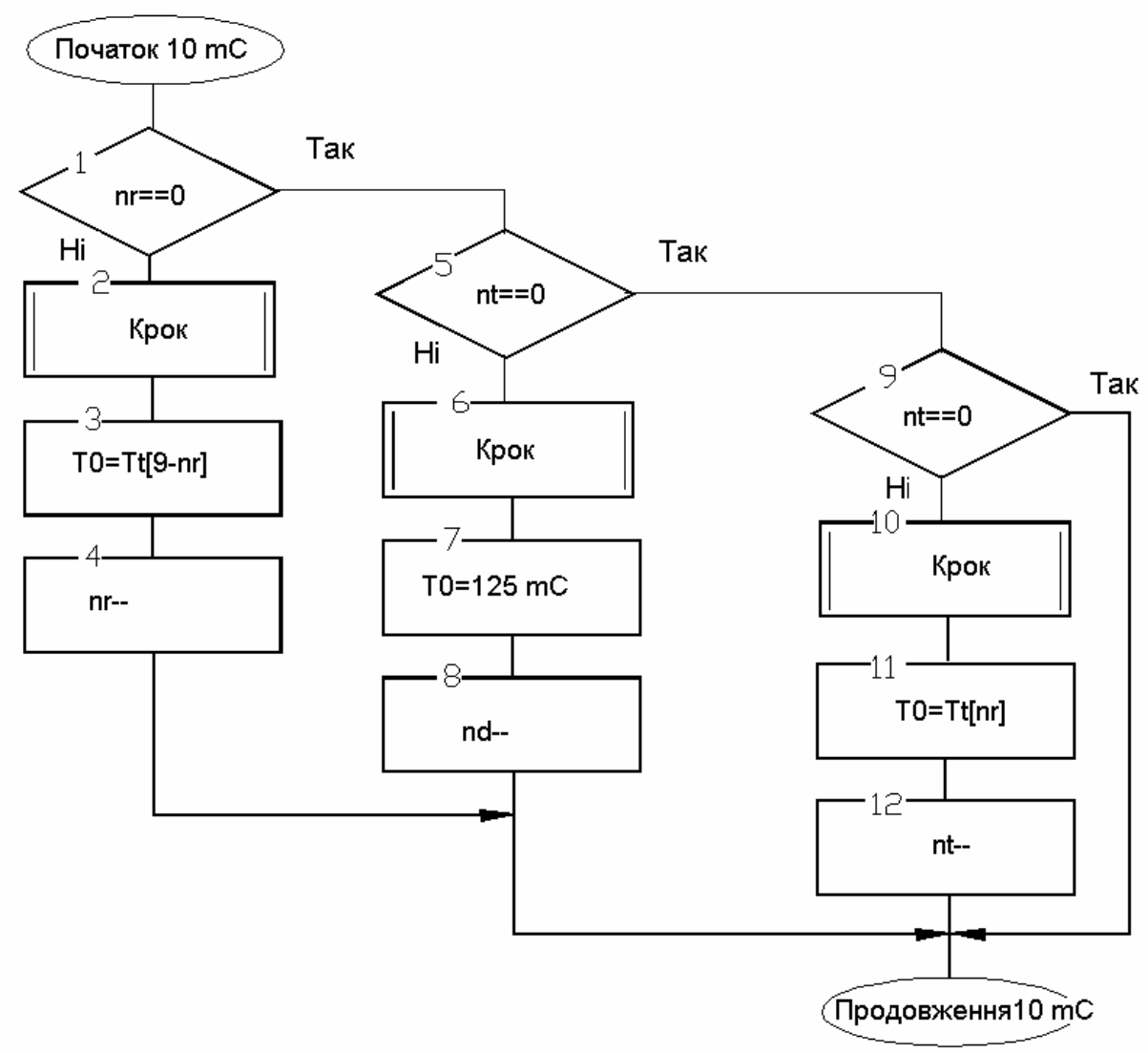


Рис. 10.11. Алгоритм відпрацьовування руху крокового привода

Програма керування кроковим двигуном наведена нижче. Підпрограма Step_ini формує вихідні дані для прямого й реверсивного керування кроковим двигуном від 2 до 127 кроків з розгоном і гальмуванням.

```
N EQU 20h; Кількість кроків і напрямок
nr EQU 30H; Кількість кроків розгону
nd EQU 31h; Кількість кроків руху
nt EQU 32h; Кількість кроків гальмування
ns EQU 33h; поточне положення ротора
tim EQU 34h; тривалість імпульсу
sr EQU 35h; номер кроку розгону гальмування
fp EQU 08h; прапор непарної кількості кроків розгону гальмування
```

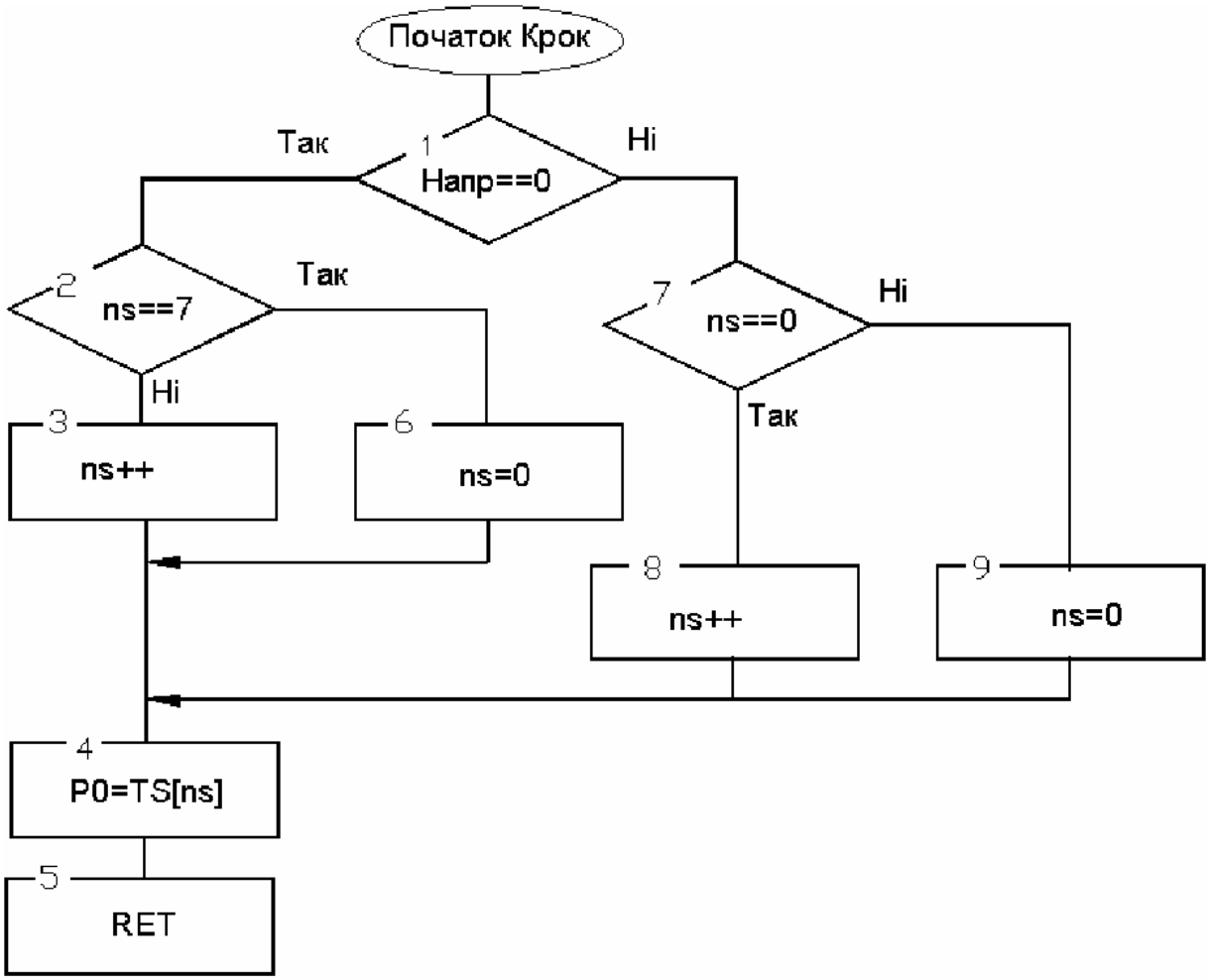


Рис. 10.12. Підпрограма формування сигналів керування мостами

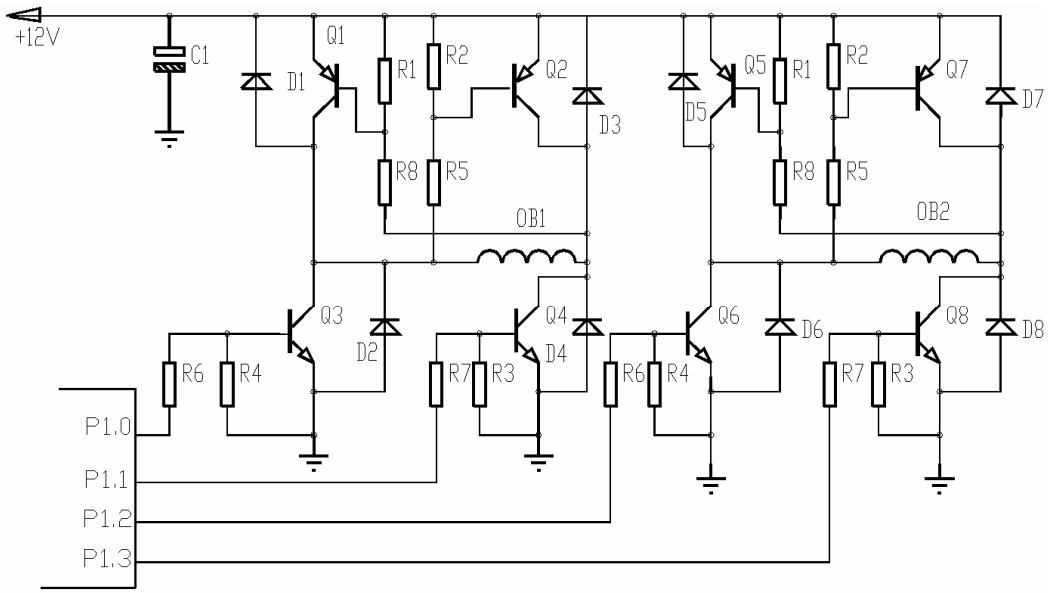


Рис. 10.13. Підключення крокового двигуна до мікроконтролера

Наведений фрагмент є описом призначення символьних змінних і вказівкою, де зберігаються значення проміжних і кінцевих змінних

```
        ljmp main
        org 0bh; вектор оброблювача від таймера
        ljmp Tim_0
main:   mov TMOD,#01h; ініціалізація таймера
        mov TL0,#0f0h; настроювання відмітника на 10 mC
        mov TH0,#0d8h;
        setb tcon.4   ; запуск таймера
        setb ie.7     ; дозвіл переривань
        setb ie.1
        mov P1,#0
```

З периферійних пристроїв у програмі використовується таймер 0 і порт P1. При ініціалізації системи таймер 0 настроюється на режим 1 з періодом переповнення 10 mC, в 0 установлюються порти виводу і початкові значення змінних.

```
        .       .       .
        mov N,#8h
         lcall Step_ini
        .       .       .
```

Вихідні дані, напрямки руху й кількість кроків передаються в підпрограму через комірку 20 h, символьне позначення N.

```
Step_ini:       ; підпрограма ініціалізації руху
        mov sr,#0  ; номер кроку розгону гальмування
        mov a,N
        jnb Acc.7,m1
        cpl a  ; перетворення додаткового формату у звичайний
        inc a
        mov N,a
        setb 07h; збереження прапора негативного напрямку
m1:     clr c
        subb a,#18
        jc m3    ; якщо менше 18 на м3
        mov nd,A   ; кількість кроків рівномірного руху
        mov nr,#9  ; кількість кроків розгону
        mov nt,#9  ; кількість кроків гальмування
        ret
m3:     mov nd,#0 ; якщо <18, кількість кроків рівномірного руху дорівнює 0
        mov a,N
        anl a,#7fh
        clr c
```

```
        rrc a
        jnc m12
        setb fp; установлюємо ознаку, що кроків гальмування менше ніж розгону
m12:  mov nt,a  ; кількість кроків розгону
        mov a,N
        anl a,#7fh
        clr c
        subb a,nt
        mov nr,a  ;кількість кроків гальмування
        ret
```

Результатом виконання цієї підпрограми є кількість кроків розгону (nr), рівномірного ходу (nd) і гальмування (nt).

```
;**************************************
;оброблювач переривань від таймера
;**************************************
Tim_0:        push acc
        push PSW
        mov TL0,#0f0h
        mov TH0,#0d8h; наступна оцінка через 10 mC
        mov a,tim ; час кроку закінчено?
        jz raz
        dec tim  ; ні
        ljmp ex
 raz:  mov a,nr ; розгін закінчено?
        jz m4     ;так
        lcall step ; формуємо сигнали керування на мости
        clr c
        mov a,sr ; визначаємо покажчик на тривалість імпульсу
        mov DPTR,#razgon
        movc a,@a+DPTR
        mov tim,a ; установлює тривалість імпульсу
        inc sr    ; номер наступного кроку
        dec nr
        ljmp ex
m4:   mov a,nd; рівномірний рух закінчено?
        jz m5    ; так
        lcall step
        mov tim,#12  ; тривалість імпульсу фіксована
        dec nd      ; номер наступного кроку
        ljmp ex
m5:   mov a,nt   ; гальмування закінчено?
        jz ex
```

```
        jnb fp,m13
        dec sr
        clr fp
        m13:  lcall step
        mov a,sr ; визначаємо покажчик на тривалість імпульсу
        dec a
        mov DPTR,#razgon
        movc a,@a+DPTR
        mov tim,a  ; установлює тривалість імпульсу
        dec nt  ; номер наступного кроку
        dec sr
ex:     pop PSW
        pop Acc
        reti
```

В оброблювач переривання програма надходить кожні 10 mC, змінна tim є програмним таймером тривалості імпульсу кроку. При розгоні й гальмуванні в неї заносяться значення з таблиці razgon, при рівномірному русі туди заноситься число 12, яке відповідає тривалості кроку 120 mC.

```
;**********************************
;Підпрограма формування значення сигналів для мостів
;**********************************
step:   mov DPTR,#faza
        mov a,ns
        jb 07h,m6
        cjne a,#7,m7
        mov ns,#0ffh
m7:     inc ns
m9:     mov a,ns
        movc a,@a+DPTR
        mov P1,a
        ret
m6:     cjne a,#0,m8
        mov ns,#8
m8:     dec ns
        ljmp m9
faza:
        db 01h,05h,04h,06h,02h,0Ah,08h,09h ; значення бітів порту P0 для 8 положень
                                           ; ротора
razgon:
         db  72,30,23,19,17,15,14,13,12; значення тривалості кроку при розгоні
        END
```

Підпрограма step формує сигнали керування мостами. Змінна ns зберігає положення ротора. Оскільки крокові двигуни використовуються без датчиків зворотного зв'язку за положенням, алгоритм задання початкової системи координат виглядає в такий спосіб. У системі, як правило, є дискретний кінцевий датчик початку координат. При увімкненні живлення, якщо датчик не спрацював, крековий двигун включається на рух до початку координат на мінімальній швидкості для перших 8 кроків. Таким чином, тривалість імпульсу дорівнює першому значенню з таблиці razgon. Це дозволяє засинхронізувати положення вектора поля й ротора. При спрацьовуванні кінцевого вимикача процес зупиняється й фіксується початок координат. Мікропроцесорна система керування дозволяє максимально використовувати можливості крокового приводу. Вона забезпечує гнучке керування, особливо в перехідних режимах, зберігаючи точність позиціювання й виконуючи будь-які програми відпрацьовування напрямку й швидкості ходу.

11. ПРАКТИЧНА ЧАСТИНА

Лабораторна робота №1

Ознайомлення із середовищем розробки й налагодження програмного забезпечення для контролерів серії MCS-51

Ціль роботи: вивчити технологію налагодження програмного забезпечення MCS-51, вивчити команди пересилань із прямою адресацією й арифметичних і логічних операцій із прямою адресацією.

Загальні відомості. Програма для мікроконтролера являє собою послідовність машинних інструкцій (кодів операцій), розглянутих у главі «система команд мікроконтролера». Однак написання програми у вигляді послідовності двійкових значень кодів операцій досить трудомісткий процес і погано піддається редагуванню. Тому для автоматизації праці програмістів розроблені транслятори з мов різних рівнів. Мовою найнижчого рівня є Асемблер. Фактично – це запис команд у вигляді мнемокоду (текстового опису) машинних інструкцій процесора плюс підтримка різних директив транслятора. Розробка програми для мікроконтролера мовою Асемблер полягає в наступному:

1. Складання тексту програми у вигляді мнемокоду процесора.
2. Трансляція в машинні інструкції процесора.
3. Налагодження програми на програмному симуляторі мікроконтролера.
4. Запис налагодженої програми в мікроконтролер.
5. Перевірка функціонування програми на реальному об'єкті.

Дані лабораторні роботи будуть виконуватися в інтегрованій системі Mcstudio. Вона являє собою інтегровані в одну оболонку наступні компоненти: текстовий редактор, компілятор, програмний симулятор мікроконтролера, формувач моделі зовнішнього оточення контролера. Прикладна програма може бути виконана як у вигляді одного файлу (модуля), так і у декількох модулів. Крім цього, в процесі трансляції, налагодження і т.д. система створює додаткові службові файли. Для керування цією сукупністю файлів використовується концепція проекту.

Методика виконання.

1. Проект створюється з головного меню Mcstudio послідовним вибором пунктів Файл – Створити – Проект. Для вказівки місця розташування проекту, імені проекту й модулів, типу процесора заповнюється форма (рис. 1).

У результаті на екрані відкриваються вікна: «Проект», «Ресурси» і вікно редактори тексту програми із прописаним шляхом до вихідного модуля програми (рис. 2).

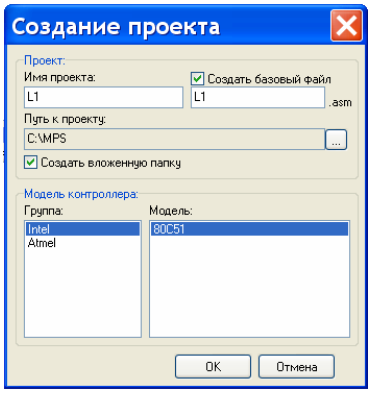


Рис. 1. Вікно створення проекту

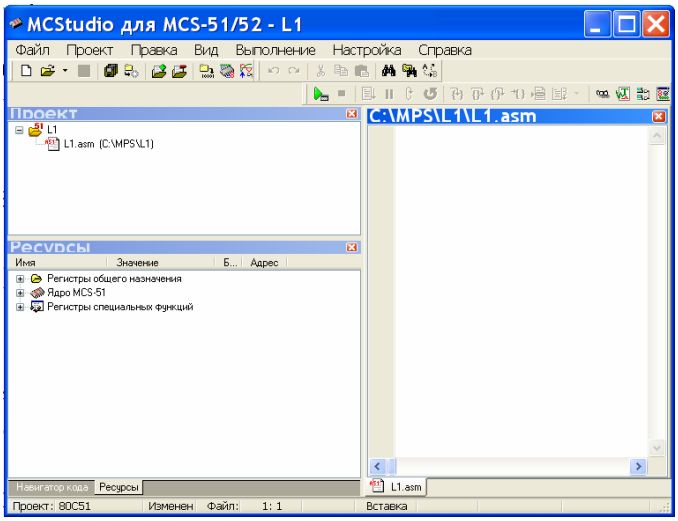


Рис. 2. Екран Mcstudio після створення проекту

1. У текстовому редакторі набираємо текст програми, наведений у лабораторній роботі. При наборі тексту необхідно дотримуватись наступних правил:

– перші 8 позицій у рядку – це поле міток;

– за ними розташовується поле команд;

– за полем команд розташовується поле операндів;

– за полем операндів може розташовуватися поле коментарів, які відокремлюються від інших полів крапкою з комою.

2. За описом команд, наведених в «Робочому зошиті №1», вивчаємо їх роботу.

3. Компілюємо програму. Для цього натискаємо комбінацію клавіш Ctrl+F9 або кнопку на панелі інструментів  . У результаті компіляції у вікні текстового редактора додається дві колонки ліворуч, рис. 3. Перша колонка містить адресу, за якою розташовуються машинні коди мікроконтролера. Друга колонка – це значення машинних кодів.

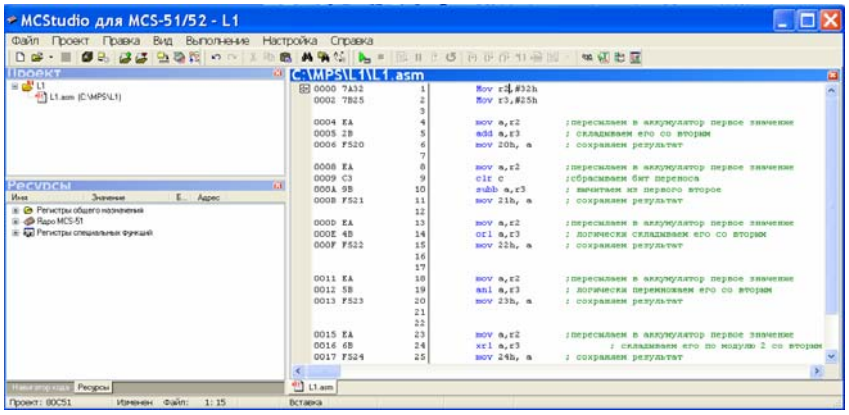


Рис. 3. Результат компіляції програми

Якщо в програмі були синтаксичні помилки, компілятор виявить і видасть повідомлення про характер помилки (рис. 4). Якщо у вікні повідомлень про помилку кликнути лівою кнопкою миші, то курсор буде поміщено у рядок програми, де була виявлена помилка.

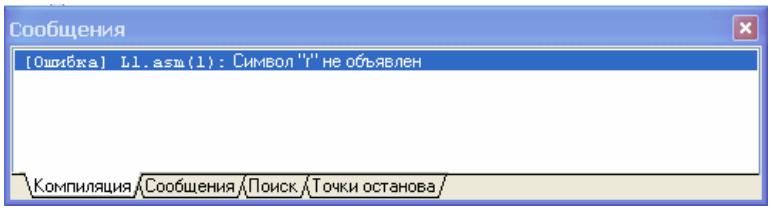


Рис. 4. Вигляд вікна повідомлень про помилку

1. Виконуємо симуляцію відкомпільованої програми в покроковому режимі. Симуляція завжди починається з натискання кнопки  або клавіші F9. Після натискання синя смуга в тексті програми вказує, яка команда буде виконуватися на наступному кроці. Черговий крок програми при покроковому налагодженні виконується натисканням клавіші F7. Покрокове налагодження необхідне, щоб переконатися, чи однаково розуміють виконання команди контролер і програміст.

2. Результати виконання команд переглядаємо у вікні відображення дерева ресурсів мікроконтролера (рис. 5).

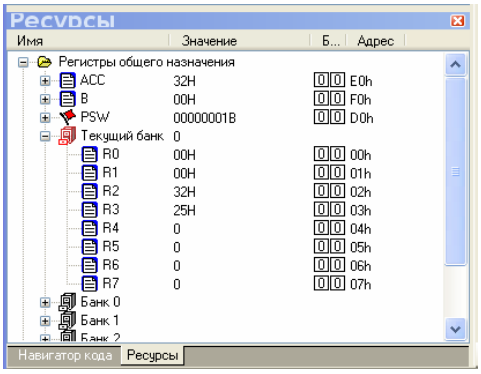


Рис. 5. Дерево ресурсів контролера на 6 кроці виконання програми

3. Підставляємо в програму значення згідно з варіантом, за результатами симуляції підготовляємо звіт лабораторної роботи.

Примітка: після закінчення симуляції натисніть клавіші Ctrl+ F2, щоб вийти з режиму симуляції. Без цього ви не зможете закрити програму Mcstudio та зберегти свої дані

Теоретичні відомості. Нижче наведені фрагменти вікон робочого середовища Mcstudio і пояснення виконання команд пересилань із прямою адресацією.

Пересилання константи.

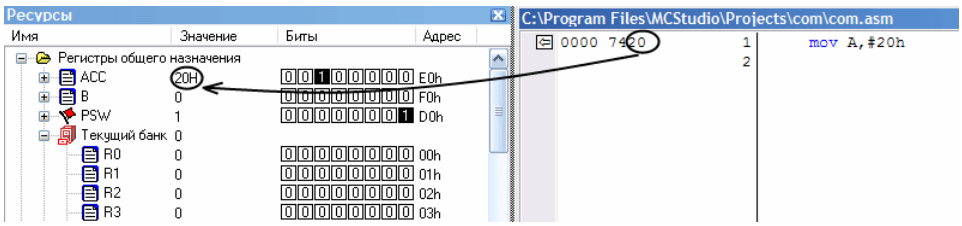


Рис. 6. Виконання команди mov A,# 20h

Команда MOV A,# d виконує пересилання константи, яке зберігається в другому байті команди, в акумулятор.

Пряма адресація

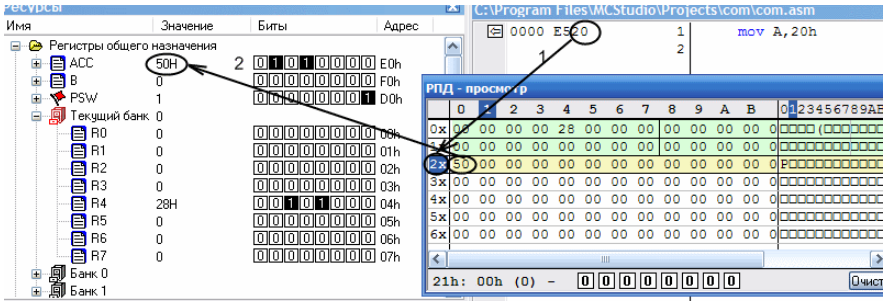


Рис. 7. Виконання команди mov A,20h

Команда MOV A,ad виконує пересилання даних із внутрішньої пам'яті в акумулятор. Адреса комірки пам'яті перебуває в другому байті команди.

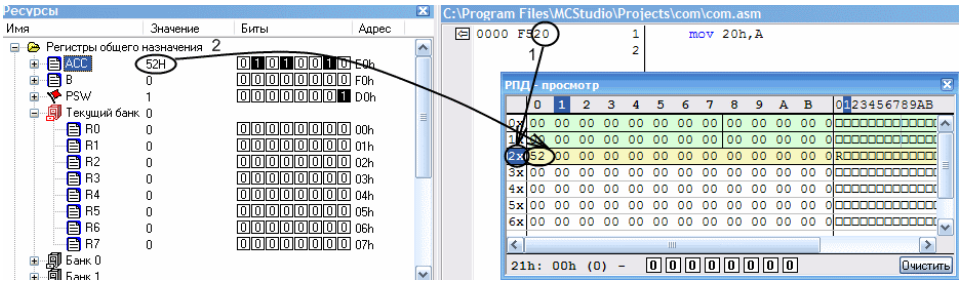


Рис. 8. Виконання команди mov 20h,A

Команда MOV ad, A виконує пересилання даних з акумулятора у внутрішню пам'ять. Адреса комірки пам'яті перебуває в другому байті команди.

Пряма регістрова адресація.

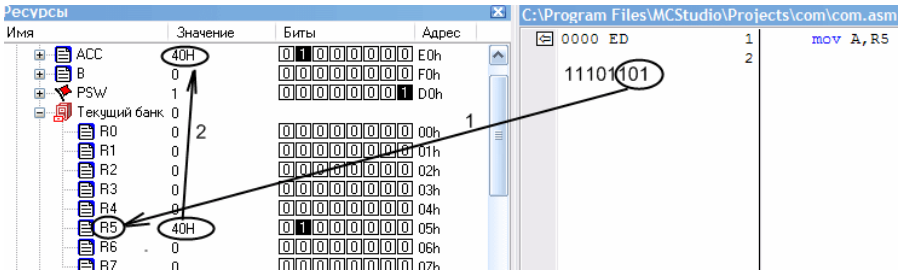


Рис. 9. Виконання команди mov A,R5

Команда MOV A,Rn, виконує пересилання значення регістру поточного банку в акумулятор. Адреса регістру зазначена в молодших трьох бітах коду операції.

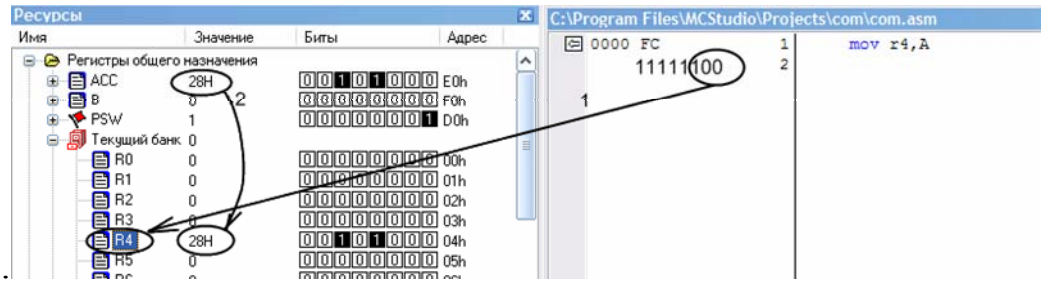


Рис. 10. Виконання команди mov R4, A

Команда MOV Rn,A виконує пересилання значення акумулятора в регістр поточного банку. Адреса регістру зазначена в молодших трьох бітах коду операції.

Задання лабораторної роботи. Написати програму роботи мікроконтролера МК51, яка виконує арифметичні й логічні дії над двома числами, поміщеними в регістри R2 і R3, і результати записує у внутрішню пам'ять даних з адреси 20H.

Текст програми мовою Асемблера

```
Mov r2,#32h        ; занесення в регістри вихідних значень
Mov r3,#25h

mov a,r2     ; пересилаємо в акумулятор перше значення
add a,r3     ; складаємо його із другим
mov 20h, a   ; зберігаємо результат

mov a,r2     ; пересилаємо в акумулятор перше значення
clr c        ; скидаємо біт переносу
subb a,r3    ; віднімаємо від першого друге
```

```
mov 21h, a  ; зберігаємо результат

mov a,r2    ; пересилаємо в акумулятор перше значення
orl a,r3    ; логічно складаємо його із другим
mov 22h, a  ; зберігаємо результат

mov a,r2    ; пересилаємо в акумулятор перше значення
anl a,r3    ; логічно перемножуємо його із другим
mov 23h, a  ; зберігаємо результат

mov a,r2    ; пересилаємо в акумулятор перше значення
xrl a,r3    ; складаємо його за модулем 2 із другим
mov 24h, a  ; зберігаємо результат
end         ; кінець програми.
```

Вихідні дані для програми:

– перше значення дорівнює номеру за списком, помноженому на 5.

– друге значення дорівнює номеру за списком, помноженому на 7.

Звіт повинен містити текст програми й результати виконання кожної операції.

Лабораторна робота №2
Дослідження внутрішньої й зовнішньої пам'яті даних і пам'яті програм

Ціль роботи: закріпити знання студентів при роботі з відлагоджувачем, вивчення команд пересилань із непрямою адресацією, покажчиків, роботу з масивами.

Теоретичні відомості. Пам'ять даних необхідна для приймання, зберігання й видачі інформації, використовуваної в процесі виконання програми. Пам'ять програм призначена для їх зберігання, а також для приймання, зберігання й видачі інформації, використовуваної в процесі виконання програми.

Для звертання до зовнішньої пам'яті даних використовується команда MOVX. Є два типи команд, що відрізняються забезпеченням 8-бітової або 16-бітової непрямої адреси до зовнішнього ОЗП даних. У першому випадку необхідно використовувати регістри R0 і R1 банків регістрів, уміст яких забезпечує 8-бітову адресу, яка мультиплексується з даними порту P0. У другому випадку при виконанні вищевказаної команди покажчик даних DPTR генерує 16-бітова адреса.

Команда MOVC A,@A+dptr здійснює звертання до пам'яті програм і пересилає з неї байт коду в акумулятор. Адреса зчитуваного байта обчислюється як сума 8-бітового вихідного вмісту акумулятора без знака й умісту DPTR. Тому в лабораторній роботі перед звертанням до ПП рекомендується скинути вміст акумулятора.

При асемблерному програмуванні мікроконтролерів сім'ї МК51 здійснювати бітову адресацію можна, указавши адресу біта або безпосередньо

сам біт. Наприклад, запис JNB A.4,M1 або JNB E4H,M1 (адреса четвертого біта акумулятора) буде сприйматися відлагоджувачем однаково.

Нижче наведені фрагменти вікон робочого середовища Mcstudio і пояснення виконання команд пересилань із непрямою адресацією

Непряма адресація

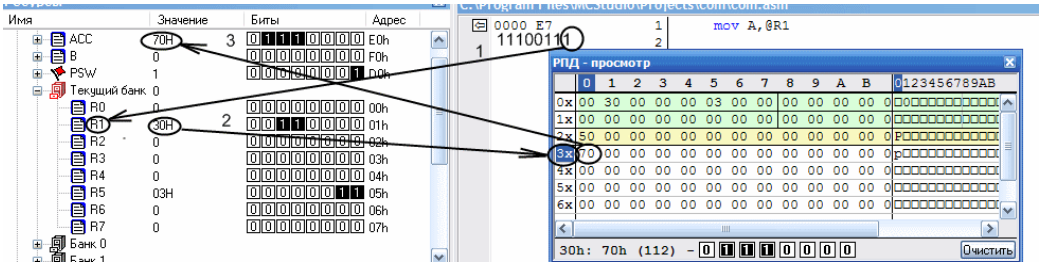


Рис. 1. Виконання команди mov A,@R1

Команда MOV A,@Ri виконує пересилання в акумулятор умісту гнізда внутрішньої пам'яті даних. Адреса регістру (галузі 1) вказується в молодшому біті коду операції. Адреса гнізда (галузі 2) перебуває в регістрі-покажчику. Регістром-покажчиком можуть виступати тільки регістри R0 і R1.

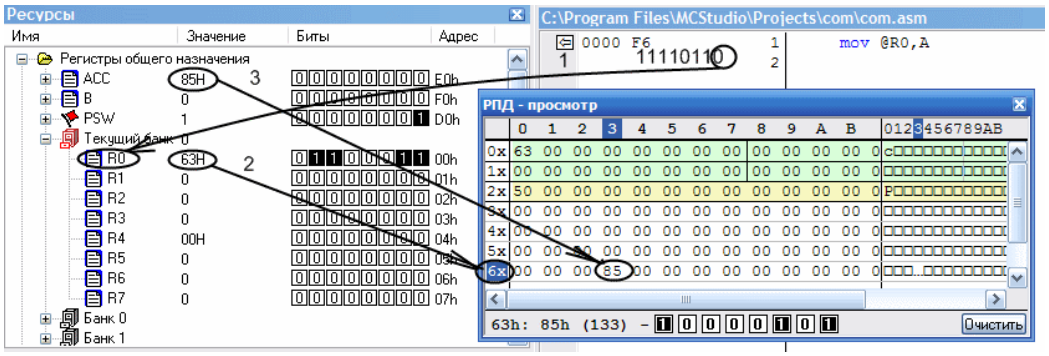


Рис. 2. Виконання команди mov @r0,A

Команда MOV @Ri,A виконує пересилання вмісту акумулятора в гніздо внутрішньої пам'яті даних. Адреса гнізда перебуває в регістрі-покажчику. Регістром-покажчиком можуть виступати тільки регістри R0 і R1.

Приклад задання. Написати програму для мікроконтролера MCS-51, яка складає за модулем два елементи двох масивів довжиною 10 байтів. Перший масив перебуває у внутрішній пам'яті даних контролера, починаючи з адреси 60h. Другий – у пам'яті програм контролера, починаючи з адреси 40h. Результат додавання за модулем масивів записується на місце першого. Якщо в четвертому біті результату додавання за модулем два є «1», то, починаючи з адреси 51h у зовнішній пам'яті даних, записується адреса цього байта. Після обробки масиву в гніздо 50h зовнішньої пам'яті даних записується кількість отриманих покажчиків. При роботі із зовнішньою пам'яттю використовується однобайтова адресація.

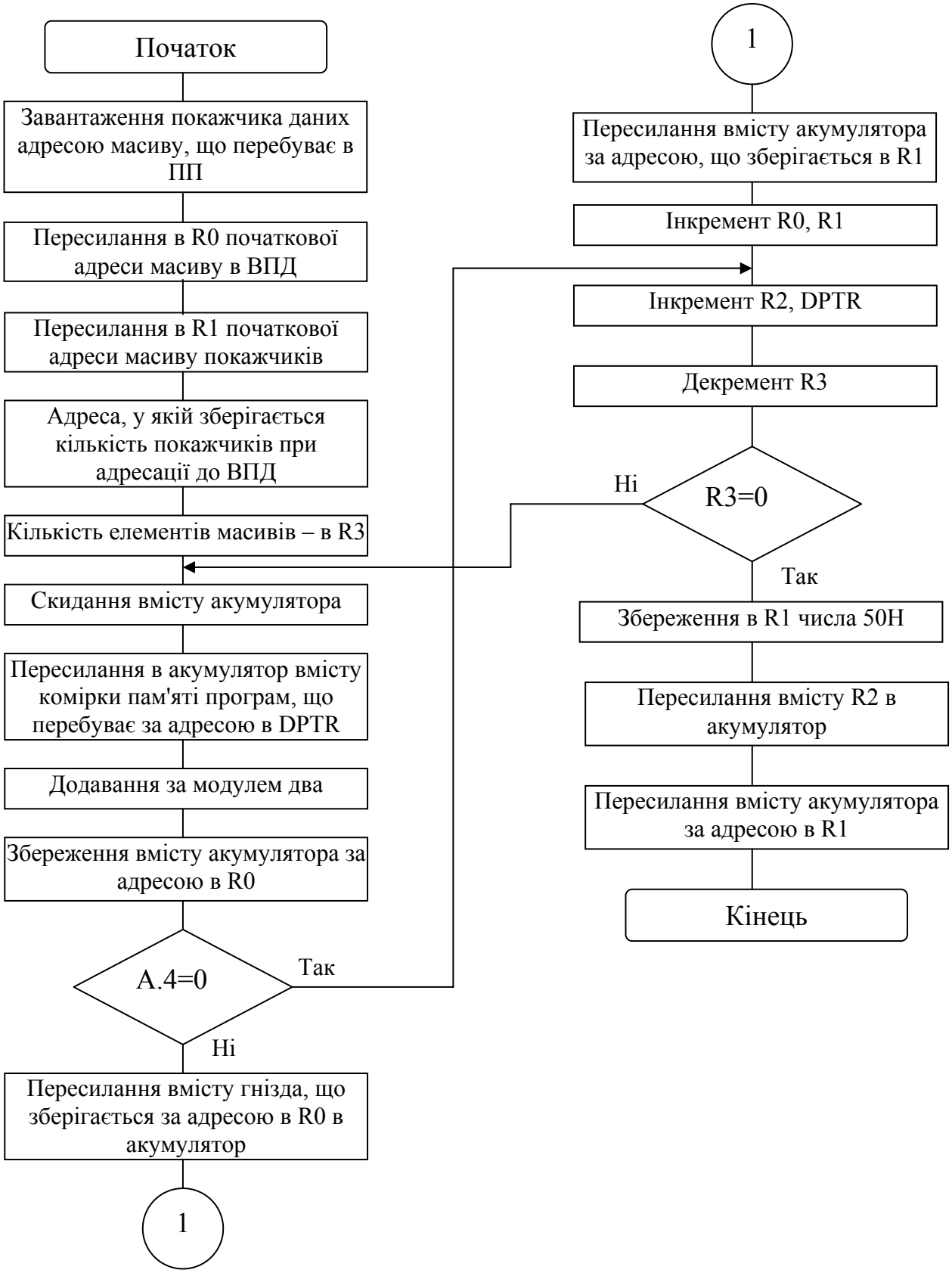
Примітка. При написанні програми задається лише один масив, що перебуває в пам'яті програм, починаючи з адреси 40H. Другий масив можна

задати безпосередньо в процесі роботи з відлагоджувачем, записавши елементи масиву в заданій області внутрішньої пам'яті даних (у цьому випадку – з адреси 60H) і зберігши їх в окремому файлі. Результат виконання програми значення вихідного масиву можна побачити у вікні резидентної пам'яті даних (РПД). Для цього в головному меню необхідно вибрати пункт Вид – Резидентна пам'ять даних.

Текст програми мовою Асемблера

```
        org 00h
        MOV DPTR,#M3 ; завантаження покажчика даних адресою масиву, що перебуває в ПП
        MOV R0,#60H  ; початкова адреса масиву, що перебуває у ВПД
        MOV R1,#51H  ; початкова адреса масиву в зовнішній ПД
        MOV R2,#00H  ; кількість отриманих покажчиків при адресації до зовнішньої ПД
        MOV R3,#0AH  ; кількість елементів масивів
M2:     CLR A        ; скидання вмісту акумулятора
        MOVC A,@A+dptr ; додавання за модулем два n-х елементів масивів
        XRL A,@R0      ; і пересилання результату на місце першого
        MOV @R0,A
        JNB A.4,M1   ; перевірка на «1» четвертого біта результату додавання за модулем два
        MOV A,R0     ; запис адреси результату додавання за модулем два, у якому
                     ; четвертий біт одиничний
        MOVX @R1,A
        INC R1       ; інкремент адрес масивів
        INC R2
M1:     INC R0
        INC dptr
        DJNZ R3,M2   ; перевірка масивів на їхнє закінчення
        MOV R1,#50H  ; запис кількості отриманих покажчиків при адресації
        MOV A,R2     ; до зовнішньої пам'яті в гніздо 50h зовнішньої пам'яті даних
        MOVX @R1,A
        org 40h
M3:     DB 01H,12H,43H,36H,0feh,37H,63H
        DB 80h,0ach,91H          ; задання масиву в пам'яті програм
End
```

Звіт повинен містити текст програми, текст індивідуального завдання й блок-схему алгоритму, результат виконання програми.



Блок-схема програми складання за модулем двох масивів

Індивідуальні задання:

Варіант 1. Написати програму, яка обчислює двобайтову суму елементів масиву, розташованого в пам'яті програм з адреси 50H. Довжина вихідного масиву – 12 однобайтових елементів.

Варіант 2. Написати програму, яка виконує множення елементів масиву, розташованого в пам'яті програм з адреси 80H на 25, і двобайтовий добуток зберігає у вигляді масиву у внутрішній пам'яті даних з адреси 30H. Довжина вихідного масиву – 8 однобайтових елементів.

Варіант 3. Написати програму, яка виконує сортування елементів масиву, розташованого в пам'яті програм з адреси 70H. Парні елементи зберігаються у внутрішній пам'яті даних з адреси 30H, а непарні – в зовнішній пам'яті даних з адреси 50H. Довжина вихідного масиву – 10 однобайтових елементів.

Варіант 4. Написати програму, яка виконує ділення елементів масиву, розташованого в пам'яті програм з адреси 80H, на 3. Ціле від ділення зберігається у внутрішній пам'яті даних з адреси 50H, а залишок – у зовнішній пам'яті даних з адреси 20H. Довжина вихідного масиву – 15 однобайтових елементів.

Варіант 5. Написати програму, яка виконує додавання за модулем 2 елементів масиву, розташованого в пам'яті програм з адреси 20H із умістом комірки пам'яті за адресою 55H. Довжина вихідного масиву – 12 однобайтових елементів. Результат розмістити за адресою 30H.

Варіант 6. Написати програму, яка виконує додавання елементів масиву, розташованого в пам'яті програм з адреси 100H із умістом комірки пам'яті за адресою 25H, і результат зберігає у вигляді масиву у внутрішній пам'яті даних з адреси 30H. Довжина вихідного масиву – 10 однобайтових елементів.

Варіант 7. Написати програму, яка виконує сортування елементів масиву, розташованого в пам'яті програм з адреси 50H. Негативні елементи зберігаються у внутрішній пам'яті даних з адреси 20H, а позитивні – в зовнішній пам'яті даних з адреси 60H. Довжина вихідного масиву 10 однобайтових елементів. Ознакою негативного числа є 1 в D7.

Варіант 8. Написати програму, яка виконує множення елементів масиву, розташованого в пам'яті програм з адреси 80H, на 5. Результат зберігається у внутрішній пам'яті даних з адреси 50H. Довжина вихідного масиву – 15 однобайтових елементів.

Варіант 9. Написати програму, яка виконує логічне додавання елементів масиву, розташованого в пам'яті програм з адреси 30H із константою A5H. Довжина вихідного масиву – 9 однобайтових елементів. Результат розмістити за адресою 40H.

Варіант 10. Написати програму, яка виконує сортування елементів масиву, розташованого в пам'яті програм з адреси 50H. Елементи, що більше 30H, зберігаються у внутрішній пам'яті даних з адреси 20H, інші – в зовнішній пам'яті даних з адреси 60H. Довжина вихідного масиву – 10 однобайтових елементів.

Лабораторна робота №3
Написання програми керування комбінаційним автоматом табличним способом

Ціль роботи: вивчити використання теорії цифрових автоматів для формалізації написання програм комбінаційних автоматів.

Вихідні дані. Вхідні сигнали для семисегментного індикатору надходять на розряди порту P1.0-P1.3, а керування індикатором здійснюється з порту 2 через буферний підсилювач.

Написати програму перекодування двійкового значення, які надходить з порту P1, у значення, що відповідають цифрам шістнадцяткового алфавіту, котрі відображаються на семисегментному індикаторі. Підключення виходів порту P2 до індикатору зображено на рис. 3.3.

Теоретичні відомості. Дешифратори будуються на основі масивів вихідних сигналів, розташовуваних у пам'яті програм. Звертання до потрібного елемента масиву здійснюється заданням зсуву відносно до початку масиву.

Розв'язок: Запишемо таблицю істинності для дешифратора (табл. 1).

Таблиця 1

Істинності для дешифратора

Входи				Виходи								Символ
P1.3	P1.2	P1.1	P1.0	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	
1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	0	0	0	0	1	1	0	1
0	0	1	0	0	1	0	1	1	0	1	1	2
0	0	1	1	0	1	0	0	1	1	1	1	3
0	1	0	0	0	1	1	0	0	1	1	0	4
0	1	0	1	0	1	1	0	1	1	0	1	5
0	1	1	0	0	1	1	1	1	1	0	1	6
0	1	1	1	0	0	0	0	0	1	1	1	7
1	0	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	0	1	1	0	1	1	1	1	9
1	0	1	0	0	1	1	1	0	1	1	1	A
1	0	1	1	0	1	1	1	1	1	0	0	B
1	1	0	0	0	0	1	1	1	0	0	1	C
1	1	0	1	0	1	0	1	1	1	1	0	D
1	1	1	0	0	1	1	1	1	0	0	1	E
1	1	1	1	0	1	1	1	0	0	0	1	F

Сформуємо в пам'яті програм контролера масив значень байта виводу згідно табл. 1 (мітка BEGDIM), які потрібно вивести на індикатор для формування шістнадцяткових цифр від 0 до F. Текст підпрограми, яка реалізує дешифратор, наведено нижче.

```
DECOD:  MOV DPTR,#BEGDIM ; заносимо в DPTR адресу першого
                         ; елемента масиву
        MOV A,P1         ; уводимо вхідні сигнали
        ANL A,#0FH       ; маскуємо старші розряди вхідних
                         ; сигналів, одержуємо зсув в
                         ; таблиці для даної комбінації
        MOVC A,@A+DPTR   ; заносимо в акумулятор значення
                         ; елемента масиву, відповідного
                         ; вхідному коду
        MOV P2,A         ; виводимо значення в порт
        RET              ; виходимо з підпрограми
BEGDIM: DB 3FH, 06H, 5BH, 4FH,    ; 0, 1, 2, 3
        DB 66H, 6DH, 7DH, 07H,    ; 4, 5, 6, 7
        DB 7FH, 6FH, 77H, 7CH,    ; 8, 9, A, B
        DB 39H, 5EH, 79H,71H      ; C, D, E, F
```

У даній програмі вхідне значення коду, утвореного вхідними сигналами, перетвориться в зсув для масиву вихідних значень. Потім за його допомогою зчитуємо з таблиці значення й виводимо у вихідний порт.

Методика виконання:

1. Заносимо програму у вікно редактора програми.

2. Контроль над роботою програми здійснюємо за допомогою редактора оточення. Для цього натискаємо кнопку  панелі інструментів. Після входу в редактор створюємо середовище оточення й у вікні «Оточення контролера» налаштовуємо виходи порту P2 на вихід (рис. 1).

3. Переходимо на вкладку «Зовнішні пристрої» і встановлюємо у вікні оточення чотири кнопки й семисегментний індикатор зі списку елементів.

4. Активізуючи по черзі кожний елемент, заповнюємо форму підключення його до контролера. Наприклад: кликнувши на кнопку в колонці, що з'явився ліворуч, натискаємо на кнопку у полі адреси, у вікні, що з'явилося, вказуємо, що вона підключена до виводу контролера, і в сусідній колонці до біта 0 порту P1 (рис. 2).

1. Після підключення кнопок і індикатору виходимо з редактора оточення, зберігши файл оточення

2. Після виходу з редактора в робочому вікні з'явиться вікно «Оточення». Якщо тепер перейти в режим налагодження, то за допомогою підключених кнопок можна задавати логічні сигнали на підключених входах, а індикатор буде відображати значення сигналів на виходах.

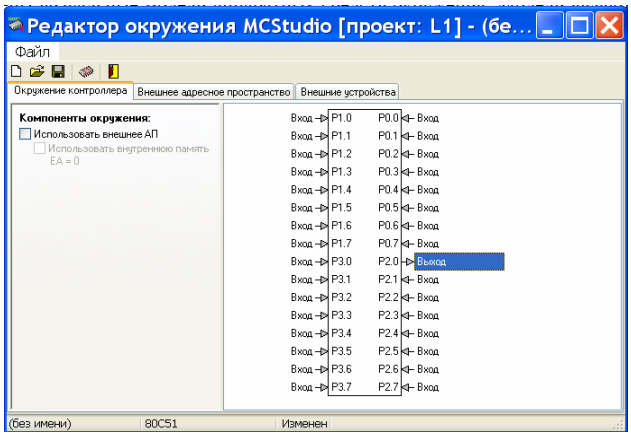


Рис. 2. Вікно настроювання оточення контролера

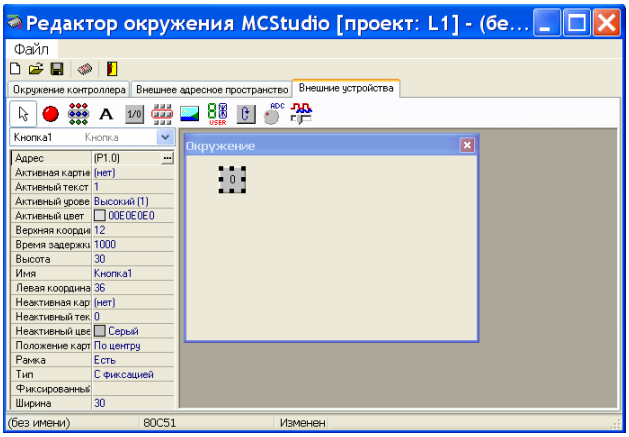


Рис. 3. Підключення зовнішніх пристроїв до контролера

Звіт повинен містити текст індивідуального задання й текст програми.

Задання:

Варіант 1. Написати програму, яка перетворить код на входах P0.0-P0.2 у символи від 3 до D.

Варіант 2. Написати програму, яка перетворить код на входах P0.4-P0.6 у символи 0,2,4,6,8,A,3,E.

Варіант 3. Написати програму, яка перетворить код на входах P1.0-P1.2 у символи 1,3,5,7,9,B,D,F.

Варіант 4. Написати програму, яка перетворить код на входах P 0-P2 у символи 1,1,2,2,3,3,4,4.

Варіант 5. Написати програму, яка перетворить код на входах P 0-P2 у символи 2,5,8,A,C,D,E,F.

Варіант 6. Написати програму, яка перетворить код на входах P 0-P2 у символи 1,3,5,7,9,B,A,D.

Варіант 7. Написати програму, яка перетворить код на входах P 0-P2 у символи 2,6,A,E,0,4,8,C.

Варіант 8. Написати програму, яка перетворить код на входах P 0-P2 у символи 1,4,7,A,D,0,3,6.

Варіант 9. Написати програму, яка перетворить код на входах P 0-P2 у символи 0,3,6,9,C,F,2,5.

Варіант 10. Написати програму, яка перетворить код на входах P 0-P2 у символи 5,9,C,0,4,8,C,0.

Програмний дешифратор виконати у вигляді підпрограми з параметрами, що задаються.

Лабораторна робота №4
Написання програми комбінаційного автомата за таблицею істинності

Ціль роботи: вивчити методи розв'язку логічних рівнянь для бітових змінних і засвоїти формальні методи написання програми для комбінаційних автоматів.

Теоретичні відомості. Спосіб реалізації комбінаційного автомата, розглянутий в 3 лабораторній роботі, має наступні обмеження: вхідні сигнали повинні бути підключені на один порт і підряд (інакше необхідно буде збирати інформацію з різних входів в одне вхідне слово); при досить великій кількості входів (7 і більше) і неповному використанні вихідних комбінацій нераціонально використовується пам'ять програм. Тому пропонується інший спосіб розв'язку задання керування, заснований на програмній реалізації логічних кон’юнктивно диз'юнктивних рівнянь.

Як і в попередньому випадку, функціонування автомата задається у вигляді таблиці істинності, наприклад, табл. 1

Таблиця 1

Приклад задання алгоритму у вигляді таблиці істинності

Входи				Вихід
P1.2	P1.4	P2.1	P2.3	P1.7
0	1	1	0	1
1	0	1	0	1
1	1	0	0	1

Запишемо логічне рівняння для виходу згідно з наведеною таблицею.

P1.7=!P1.2 & P1.4 & P2.1 & P2.3 +P1.2 & !P1.4 & P2.1 & !P2.3+P1.2 & !P1.4 & P2.1 & !P2.3 (3.1)

Складемо схему алгоритму для першої функції «І» рівняння (рис. 3.1).

Слід звернути увагу, що команді переходу за умовою відповідає умовний оператор на блок-схемі алгоритму з результатом «ні» унизу. Тому для того, щоб програма була зручною, пропонується використовувати команди, які перевіряють інверсну умову. Тоді послідовне виконання таких команд і буде відповідати виконанню вихідної умови логічної функції.

У результаті програмна реалізація для першої функції «І» рівняння має вигляд:

```
JB P1.2,M1
JNB P1.4,M1
JNB P2.1,M1
JNB P2.3,M1
SETB P1.7
LJMP EXIT
M1: * * * ;перевірка наступної умови
```

Схема алгоритму для функцій «АБО» наведена на рис. 3.6 у розділі 3.

У цьому алгоритмі під &1, &2 маються на увазі логічні функції «І» по рядках таблиці наведених рівнянь.

Повна програма логічного рівняння, відповідного до таблиці істинності, має вигляд:

```
Loop: JB P1.2,M1; перевірка умови першого рядка таблиці
      JNB P1.4,M1
      JNB P2.1,M1
      JNB P2.3,M1
      SETB P1.7; установка виходу в одиницю
      LJMP Loop; і вихід
M1:
      JNB P1.2,M2; ; перевірка умови другого рядка таблиці
      JB P1.4,M2
      JNB P2.1,M2
      JB P2.3,M2
      SETB P1.7; установка виходу в одиницю
      LJMP Loop; і вихід
M2:
      JNB P1.2,M3; перевірка умови третього рядка таблиці
      JB P1.4,M3
      JNB P2.1,M3
      JB P2.3,M3
      SETB P1.7; установка виходу в одиницю
      LJMP Loop; і вихід
M3: CLR P1.7; скинути вихід у нуль – не виконалася жодна з умов
      LJMP Loop; і вихід
```

Методика виконання:

1. Заносимо програму у вікно редактора програми.

2. Контроль над роботою програми зробимо за допомогою редактора оточення. Для цього підключаємо на входи відповідні до задання кнопки, а на виходи – світлодіоди.

3. Правильність складання умов появи вихідних сигналів перевіряється під час симуляції виконання в покроковому режимі, остаточна перевірка – в автоматичному режимі. Для переходу на симуляцію в автоматичному режимі необхідно після включення повторно нажати клавішу F9.

Звіт повинен містити текст програми, вихідні дані й блок-схему алгоритму.

Варіанти задань

Варіант 1	Комбінація натискання клавіш				Стан портів виводу			
Порти МК	P0.0	P1.2	P1.4	P1.5	P2.1	P2.2	P1.3	P2.3
Кодова комбінація 1	1	0	0	0	1	0	0	1
Кодова комбінація 2	0	1	0	0	0	1	1	0

Варіант 2	Комбінація натискання клавіш				Стан портів виводу			
Порти МК	P0.0	P1.2	P1.4	P1.5	P2.1	P2.2	P1.3	P2.3
Кодова комбінація 1	1	0	0	1	1	1	0	0
Кодова комбінація 2	0	1	1	0	0	1	1	0

Варіант 3	Комбінація натискання клавіш				Стан портів виводу			
Порти МК	P0.0	P1.2	P1.4	P1.5	P2.1	P2.2	P1.3	P2.3
Кодова комбінація 1	1	1	0	0	1	0	0	1
Кодова комбінація 2	0	1	0	0	0	1	1	0

Варіант 4	Комбінація натискання клавіш				Стан портів виводу			
Порти МК	P0.0	P1.2	P1.4	P1.5	P2.1	P2.2	P1.3	P2.3
Кодова комбінація 1	1	0	0	0	1	1	1	1
Кодова комбінація 2	0	1	0	0	0	1	1	0

Варіант 5	Комбінація натискання клавіш				Стан портів виводу			
Порти МК	P0.0	P1.2	P1.4	P1.5	P2.1	P2.2	P1.3	P2.3
Кодова комбінація 1	1	0	0	1	1	0	0	1
Кодова комбінація 2	0	1	0	1	0	1	0	0

Варіант 6	Комбінація натискання клавіш				Стан портів виводу			
Порти МК	P0.0	P1.2	P1.4	P1.5	P2.1	P2.2	P1.3	P2.3
Кодова комбінація 1	1	0	0	1	1	0	0	1
Кодова комбінація 2	0	1	1	0	0	1	1	0

Варіант 7	Комбінація натискання клавіш				Стан портів виводу			
Порти МК	P0.0	P1.2	P1.4	P1.5	P2.1	P2.2	P1.3	P2.3
Кодова комбінація 1	1	0	0	0	1	1	0	1
Кодова комбінація 2	1	1	0	0	0	1	1	0

Варіант 8	Комбінація натискання клавіш				Стан портів виводу			
Порти МК	P0.0	P1.2	P1.4	P1.5	P2.1	P2.2	P1.3	P2.3
Кодова комбінація 1	1	1	0	1	1	1	0	0
Кодова комбінація 2	0	1	1	0	0	1	1	0

Варіант 9	Комбінація натискання клавіш				Стан портів виводу			
Порти МК	P0.0	P1.2	P1.4	P1.5	P2.1	P2.2	P1.3	P2.3
Кодова комбінація 1	1	1	1	0	1	0	0	1
Кодова комбінація 2	0	1	1	0	0	0	1	0

Варіант 10	Комбінація натискання клавіш				Стан портів виводу			
Порти МК	P0.0	P1.2	P1.4	P1.5	P2.1	P2.2	P1.3	P2.3
Кодова комбінація 1	1	0	0	1	1	0	0	1
Кодова комбінація 2	0	1	0	1	0	1	1	0

Варіант 11	Комбінація натискання клавіш				Стан портів виводу			
Порти МК	P0.0	P1.2	P1.4	P1.5	P2.1	P2.2	P1.3	P2.3
Кодова комбінація 1	1	0	0	1	1	0	0	1
Кодова комбінація 2	0	1	0	0	0	0	0	0

Варіант 12	Комбінація натискання клавіш				Стан портів виводу			
Порти МК	P0.0	P1.2	P1.4	P1.5	P2.1	P2.2	P1.3	P2.3
Кодова комбінація 1	1	1	0	1	1	0	0	1
Кодова комбінація 2	0	1	1	0	0	0	1	0

Варіант 13	Комбінація натискання клавіш				Стан портів виводу			
Порти МК	P0.0	P1.2	P1.4	P1.5	P2.1	P2.2	P1.3	P2.3
Кодова комбінація 3	0	0	1	0	1	1	1	0
Кодова комбінація 4	0	1	0	1	0	1	1	1

Варіант 14	Комбінація натискання клавіш				Стан портів виводу			
Порти МК	P0.0	P1.2	P1.4	P1.5	P2.1	P2.2	P1.3	P2.3
Кодова комбінація 3	0	1	1	0	1	1	1	0
Кодова комбінація 4	1	0	0	1	0	1	1	1

Варіант 15	Комбінація натискання клавіш				Стан портів виводу			
Порти МК	P0.0	P1.2	P1.4	P1.5	P2.1	P2.2	P1.3	P2.3
Кодова комбінація 3	0	0	0	0	1	1	1	0
Кодова комбінація 4	0	0	0	1	0	0	0	1

Варіант 16	Комбінація натискання клавіш				Стан портів виводу			
Порти МК	P0.0	P1.2	P1.4	P1.5	P2.1	P2.2	P1.3	P2.3
Кодова комбінація 3	0	0	1	1	0	1	1	0
Кодова комбінація 4	0	0	0	1	1	1	1	1

Лабораторна робота №5
Дослідження тимчасових затримок у МП системах

Ціль роботи: навчитися писати програми й розраховувати значення змінних для програмних тимчасових витримок.

Теоретичні відомості. Тимчасові витримки в мікропроцесорних системах легше всього організувати через програмні лічильники. У якості лічильників зручно використовувати регістри R0 – R7. При цьому виходимо з того, що команди виконуються за кінцевий час. Вихідними даними для розрахунків є частота кварцу мікроконтролера. У середовищі Mcstudio вона встановлюється в меню **Виконання** – **Опції симуляції**. Програми формування затримок зручно оформляти у вигляді підпрограми з передачею параметрів. Параметр – значення регістрів, що забезпечує задану тимчасову затримку.

У прикладі наведено програму тимчасової затримки для двобайтового лічильника.

Приклад:

```
        Mov r2,#25h; завантаження молодшого байта затримки 1 ц
        Mov r3,#47h ; завантаження старшого байта затримки 1 ц
        Lcall zd1; виклик підпрограми затримки 2 ц
M1:     ajmp M1; нескінченний цикл
zd1:    mov a,r2; пересилання вхідного значення в акумулятор 1 ц вип. r3 раз
m2:     djnz a,m2 ; декремент молодшого байта затримки 2 ц вип. r2*r3 раз
        djnz r3,zd1; декремент старшого байта затримки2 ц вип. r3 раз
        ret             ;вихід з підпрограми 2 ц
```

Розрахунки часу затримки. За основу береться тривалість одного машинного циклу контролера. При тактовій частоті процесора 12 МГц тривалість машинного циклу 1 мкС. Загальна тривалість обчислюється за наступною формулою

$$Тзд=1ц+1ц+2ц+1ц*r3+2ц*r2*r3+2ц*r3+2=$$
$$=1+1+2+71+2*37*71+2*71+2=10869\ [мкс].$$

Методика виконання:

1. Контроль витримки затримки проводиться за покажчиком часу, що перебуває у вікні «Ресурси» (пункт «Ядро MCS-51» (рис. 1).

2. Встановлення заданої частоти генератора виконується через пункт меню Виконання – Опції симуляції.

3. Для контролю виконання програми в автоматичному режимі використовувати крапку зупинки, встановлену на мітці M1. Для задання крапки зупинки необхідно маркер помістити на потрібну крапку програми й натиснути клавішу F5.

4. Задається крапка зупинки. Програма запускається на симуляцію в автоматичному режимі після переривання виконання (в крапці зупинки), час виконання не повинен відрізнятися від встановленого в заданні більш ніж на 5 мкс.

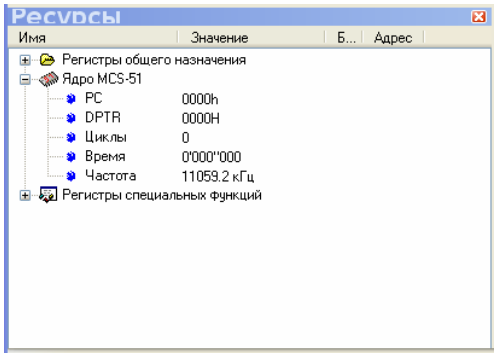


Рис. 1. Вікно ядра MCS-51

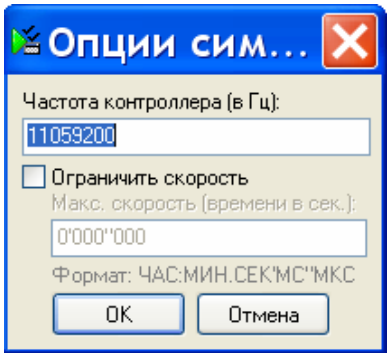


Рис. 2. Вікно установки частоти генератора мікроконтролера

5. Значення регістрів визначається за формулою, що наведена вище. Для цього треба задати значення старшого байта затримки, що дорівнює 255, і за рівнянням знайти значення молодшого байта. Якщо отримане значення перевищує число 255, додаємо ще один цикл у програмі. З урахуванням цього коректуємо формулу й знаходимо потрібні значення.

Варіанти задань.

Час затримки повинен дорівнювати номеру за списком, що помножений на 27 мілісекунд.

Звіт повинен містити текст програми, вихідні дані й блок-схему алгоритму, отримане реальне значення затримки.

Лабораторна робота №6
Побудова апаратного відмітника часу з використанням таймера

Ціль роботи: вивчити принципи написання оброблювачів переривання, навчитися розраховувати значення константи, що завантажується в таймер за заданим періодом формування міток часу, навчитися працювати у відлагоджувачі при використанні переривань, спостерігати за таймером у режимі відмітника часу.

Теоретичні відомості. Відмітник часу – пристрій, який формує переривання через фіксовані проміжки часу.

Робота з таймером у режимі відмітника часу з використанням переривання складається з декількох етапів.

Спочатку необхідно встановити таймер у режимі 1 (16-розрядний таймер). Задання режиму проводиться за допомогою занесення керуючого слова в регістр TMOD. За режим TM0 відповідають біти D1, D0.

Для формування необхідного коефіцієнта ділення заносимо в регістри TH0, TL0 число, що дорівнює різниці 65535 і значенню необхідного коефіцієнта ділення.

Для дозволу роботи TM0 установлюємо в регістрі TCON.4 логічну «1».

Для дозволу проходження переривання від TM0 у регістрі IE.1 і IE.7 необхідно встановити «1».

Для організації тимчасових затримок з використанням відмітника часу використовуємо програмно організовані лічильники, розташовувані в пам'яті даних. Для запуску затримки в комірку заноситься число. Значення числа визначається як результат ділення необхідної тимчасової затримки на період відмітника часу. Контроль закінчення затримки виконується перевіркою на нуль значення в програмному лічильнику. Декрементування лічильників проводиться в оброблювачі переривання, якщо лічильник не дорівнює нулю.

Текст програми мовою Асемблера, що реалізує миготіння світлодіоду порту P1.0, яка здійснює керування двома затримками. Лічильники затримок організовано в 31 і 32 комірках пам'яті даних.

```
        ljmp m0             ; обходимо вектор переривання
        org 0bh             ; вектор переривання від таймера TM0
        push acc
        mov th0,#3ch
        mov tl0,#0b0h       ; налаштування таймера на період в 50 мс
        djnz 30h,m2
        mov 30h, #20        ; організація постдільника
        mov a,31h
        jz m3
        dec a               ; організація першої затримки
        mov 31h,a
m3:     mov a,32h
        dec a               ; організація другої затримки
        mov 32h,a
m2:     pop acc
reti                        ; повернення з переривання
m0:     mov sp,#70h
        mov tmod,#1h        ; установка таймера 0 у режим 1
        mov th0,#3ch
        mov tl0,#0b0h
        setb tcon.4
        setb ie.7           ; дозвіл роботи TM0 і проходження
        setb ie.1           ; переривання від TM0
        mov 30h,#20
On:             setb P1.0       ;включення світлодіоду
mov a,#03h
                mov 31h,a       ;запуск затримки 1 на 3з
zdoff: mov a,31h    ;очікування затримки на вимикання
                jnz zdoff
Of:             clr P1.0
                mov a,#03h
                mov 32h,a       ; запуск затримки 1 на 3 с
zdon:           mov a,32h       ; очікування затримки на включення
                jnz zdon
                ljmp On
end
```

Пояснення до програми. Дана програма використовує таймер 0 для створення апаратного відмітника часу з періодом 50 mc. Вектор переривання від таймера 0 перебуває за адресою 0bh. В оброблювачі переривань проводиться перезавантаження таймера, яке забезпечує час до наступного переривання, що дорівнює 50 mc. У комірці 30h організовано програмний постдільник на 20. Таким чином, за допомогою постдільника одержуємо період відмітника 1 с. Така змінна дуже зручна, якщо необхідно робити періодичні дії з періодом 1 с. У комірках 31 h і 32 h організовуємо затримки від 1 до 255 с.

В основній програмі (мітка м0) робимо ініціалізацію таймера й переривання від нього, після чого запалюємо світлодіод, підключений до біта 0 порту 1, і за допомогою затримки 1 залишаємо увімкненим його на 3 с. Після цього гасимо його й залишаємо вимкненим на 3 с за допомогою затримки 2. Командою LJMP On зациклюємо програму на миготіння.

Установивши крапки зупинки на рядках з мітками On і Off, у моменти зупинки програми можна бачити, що зміна стану світлодіоду відбувається кожні 3000000 мкс.

Методика виконання:

1. Заносимо програму у вікно її редактора.
2. Контроль над роботою програми здійснюємо за допомогою редактора оточення. Для цього підключаємо на вихід P1.0 світлодіод.
3. Установлюємо крапки зупинки відповідно до пояснень до програми.

Створюємо вікно для спостереження за змінними в процесі налагодження. Для цього в меню вибираємо пункти Вид – Перегляд значень. Ініціюємо правою кнопкою миші й додаємо у вікно регістри, що цікавлять нас, і комірки пам'яті (th0, tl0, 30h, 31h, 32h) (рис. 1). Через вікно створення перегляду й конструктор виразів.

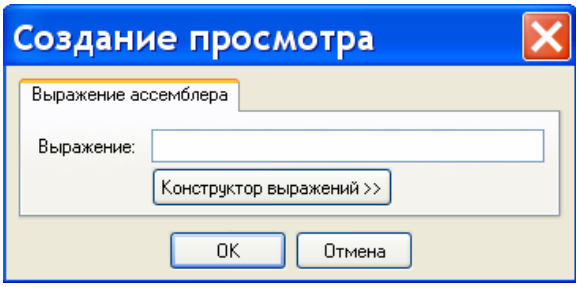


Рис. 1. Вікно для заповнення вікна налагодження

4. У результаті одержуємо вікно перегляду значень зі змінними які нам необхідні для налагодження (рис. 3).
5. Для більш повного розуміння процесів, пов'язаних з перериванням програми таймером, виконання програми до першого переривання робимо симуляцією в покроковому режимі. Для цього перший цикл переривання при ініціалізації виконаний укороченим.

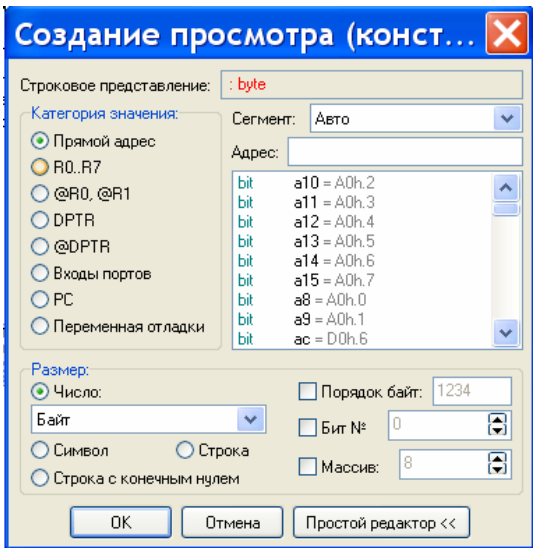


Рис. 2. Вікно конструктора виразів

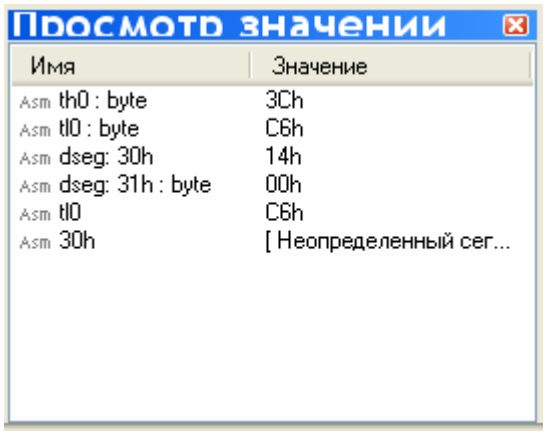


Рис. 3. Вікно перегляду значень

6. Запускаємо симуляцію в автоматичному режимі й контролюємо час зупинки за часом виконання в ядрі MCS-51.

Звіт повинен містити тексти програми та вихідного задання.

Задання:

Варіант 1. Реалізувати затримку в 2 с і вивести на порти P1.0-P1.3 у вигляді хвилі, що біжить, зліва направо.

Варіант 2. Реалізувати затримку в 1 с і вивести на порти P1.0-P1.3 у вигляді тіні, що біжить, зліва направо.

Варіант 3. Реалізувати період миготіння 2 с результату комбінації, що спрацювала, для лабораторної роботи 4.

Варіант 4. Реалізувати рахунок і індикацію на семисегментний індикатор змінної від 0 до 8 з періодом 2 с.

Варіант 5. Реалізувати затримку в 2 с і вивести на порти P1.0-P1.3 у вигляді хвилі, що біжить, справа наліво.

Варіант 6. Реалізувати затримку в 1 с і вивести на порти P1.0-P1.3 у вигляді тіні, що біжить, справа наліво.

Варіант 7. Реалізувати рахунок і індикацію на трьох світлодіодах двійкового коду від 0 до 7 з періодом 2 с.

Варіант 8. Реалізувати програму, яка включає світлодіод, через 5 с після натискання кнопки, а виключає, коли відпускають кнопку.

Варіант 9. Реалізувати програму, яка включає світлодіод, після натискання кнопки, а виключає через 5 с після відпускання кнопки.

Варіант 10. Реалізувати програму, яка блимає світлодіодом з періодом 3 с, доки натиснута кнопка, а виключає, коли відпускають кнопку.

Лабораторна робота №7
Дослідження послідовного інтерфейсу в мікропроцесорних системах

Ціль роботи: навчитись писати програми обміну інформацією в мікропроцесорних системах з послідовним інтерфейсом

Теоретичні відомості. Послідовний інтерфейс у мікроконтролері серії MCS-51 призначено для обміну інформацією з іншими обчислювальними системами на відстані від декількох метрів до декількох кілометрів.

Послідовний інтерфейс має власну назву, оскільки він перетворює дані, що надходять до нього, в паралельному форматі по восьмирозрядній шині в послідовний код, тобто в послідовний у часі потік бітових даних на виході передавача. Крім цього, він виконує зворотне перетворення. Дані в послідовному форматі, що надходять на вхід приймача, перетворяться в паралельні і можуть бути зчитані в акумулятор.

Перетворення виконується за допомогою зсувного регістру. Швидкість передачі задається за допомогою таймера 1, що працює в другому режимі без переривань.

Швидкість передачі визначається за формулою:

$$C = \left(2^{SMOD} / 32\right)\left(f_{osc} / 12\right)\left(256 - \left(TH1\right)\right),$$

де: f_{osc} – частота кварцового резонатора мікроконтролера.

Текст програми мовою Асемблера, що реалізує передачу масиву даних, які розташовуються у внутрішній пам'яті даних з адреси 30H довжиною 10 байтів. Після передачі проводиться приймання відповідних даних у масив, що розташовується з адреси 50H довжиною 10 байтів. Швидкість обміну даними – 19200 біт/с, у режимі 9 бітів змінювана швидкість обміну.

```
        Mass equ 30h
        Recivmas equ 50h; початок масиву, що приймається
        Flag equ 0h; кінець обміну даними
        Razmer equ 12h; R2 2 Банк
        Ukaz equ 10h; R0 2 Банк
        ljmp main; перехід на основну програму
        org 23h; вектор переривання від послідовного інтерфейсу
        ljmp transmit
;ОСНОВНА ПРОГРАМА
main:   mov TMOD,#20h; режим 2 таймеру 1
        mov PCON,#80h;SMOD=1
        mov TL1,#0fdh
        mov TH1,#0fdh; частота обміну 19.2 kgc
        setb TCON.6; режим роботи USART
        setb SCON.6
        setb SCON.7
```

```
        setb SCON.3
        mov Ukaz,#mass+1
        mov Razmer,#10; довжина масиву
        setb Flag
;ПЕРЕДАЧА
        mov SBUF, Mass; перший байт пішов
        setb IE.7; дозвіл переривань
        setb IE.4
m1:     jb Flag,m1; контроль закінчення передачі
;ПРИЙМАННЯ
        mov Ukaz,#recivmas; початок масиву приймання
        mov Razmer,#10; довжина масиву приймання
        setb SCON.4
        setb Flag
m1r:    jb Flag,m1r
me:     ljmp me
;ПІДПРОГРАМА ОБРОБКИ ПЕРЕРИВАНЬ ВІД USART
transmit:
        push Acc
        push PSW
        setb PSW.4; активний банк 2
        clr PSW.3
        jb SCON.0,reciw; виявлення від кого запит
        djnz r2,m2t; контроль закінчення передачі
        clr Flag; передачу закінчено
        ljmp m2
m2t:    mov SBUF,@R0; передача елемента масиву в передавач
        inc r0
m2:     clr TI
m2r:    pop PSW
        pop Acc
        reti
reciw:  mov @R0,SBUF; приймання
        inc R0
        clr RI
        djnz r2,m2r; контроль кінця приймання
        clr  SCON.0
        clr Flag
        ljmp m2r
        end
```

Опис програми. В основній програмі ініціалізується таймер і послідовний інтерфейс. А так само проводиться запуск передачі за допомогою запису першого байта в SBUFF. Контроль над передачею здійснюється апаратно через виклик переривання від біта TI. Оскільки вектор переривання від приймача й передавача один – 23H, то при вході в оброблювач визначається

від кого запит. Передача інших байтів масиву в SBUFF проводиться в оброблювачі переривання. Зчитування прийнятих байтів із приймача виконується так само в оброблювачі переривань.

Методика виконання.

1. Заносимо програму у вікно її редактора.

2. Програму виконуємо в режимі симуляції з використанням підсистеми послідовного порту (натиснути кнопку ) і вікна «Статистика UART».

3. Симулятор послідовного порту дозволяє спостерігати процес передачі і приймання даних із прив'язкою до часу тактування мікроконтролера. Вікно симулятора УСАПП зображено на рис. 1. Крім цього, у вікні можна сформувати повідомлення для приймача й імітувати в ручному або автоматичному режимі вступ його на вхід приймача.

4. Емуляцію появи інформації на вході приймача виконуємо в ручному режимі. Формувати список байтів для приймання можна, натиснувши кнопку + у полі керування буфером приймання. Для того, щоб черговий байт із буфера потрапив на вхід приймача, потрібно натиснути кнопку «Надіслати».

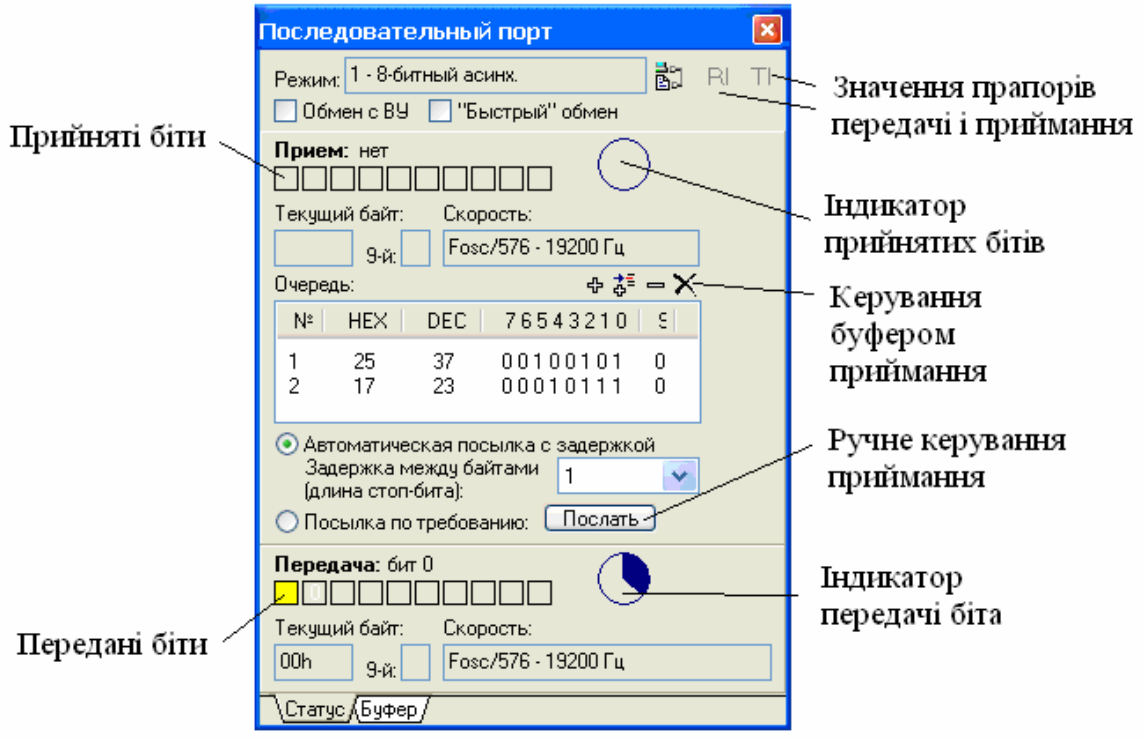


Рис. 1. Вікно симулятора послідовного обміну

Звіт повинен містити текст програми, вихідні дані й блок-схему алгоритму.

Задання: Організувати обмін інформацією мікроконтролера з іншим мікроконтролером. Переданий масив розташовується в пам'яті програм з адреси й довжиною, що зазначено в таблиці, згідно варіанта. Прийняті дані треба розмістити у внутрішній пам'яті з адреси й довжиною, що зазначена в таблиці, згідно варіанта. В оброблювачі переривань треба використовувати

банк регістрів загального призначення згідно варіанта. Швидкість обміну даними – згідно варіанта.

Таблиця 1

Індивідуальні задання на лабораторну роботу

Номер варіанта	Початкова адреса переданого масиву	Довжина переданого масиву	Початкова адреса прийнятого масиву	Довжина прийнятого масиву	Номер банку оброблювача переривання	Швидкість обміну біт/с.
1	100h	8	20h	9	1	1200
2	150h	6	50h	11	2	2400
3	220h	4	40h	13	3	4800
4	50h	10	10h	7	2	9600
5	60h	5	60h	12	1	19200
6	80h	7	20h	10	3	1200
7	120h	9	50h	8	1	2400
8	170h	11	40h	6	2	4800
9	30h	13	10h	4	3	9600
10	10h	4	60h	13	1	19200

СПИСОК ЛІТЕРАТУРИ

1. Каган, Б.М. Основы проектирования микропроцессорных устройств автоматики [Текст] / Б.М. Каган, В.В. Сташин. – М.: Энергоатомиздат, 1987. – 304 с.

2. Бродин, В.Б. Системы на микропроцессорах и БИС программируемой логике [Текст] / В.Б. Бродин, А.В. Калинин. – М.: ЭКОМ, 2002. – 400 с.

3. Костинюк, Л.Д. Мікропроцесорні засоби та системи [Текст] / Л.Д. Костинюк, Я.С. Парганчук. – Львів.: Львівська політехніка, 2001. – 200 с.

4. Сташин, В.В. Проектирование цифровых устройств на однокристальных микроконтроллерах [Текст] / В.В. Сташин, А.В. Урусов, О.Ф. Мологонцева. – М.: Энергоатомиздат. – 1990. - 224 с.

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

Навчальне видання

ТКАЧОВ Віктор Васильович
ГРУЛЕР Герхард
НОЙБЕРГЕР Ніколаус
ПРОЦЕНКО Станіслав Миколайович
КОЗАРЬ Микола Володимирович

МІКРОПРОЦЕСОРНА ТЕХНІКА
Навчальний посібник

Редактор Є.М. Ільченко

Комп'ютерний набір та верстка Н.М. Безгінова

Підписано до друку 31.07.2012 р. Формат 30х42/4
Папір офсет. Ризографія. Ум. друк. арк. 11
Обл.-вид. арк. 15,3. Тираж 300 пр. Зам. №

Підготовлено до друку та видрукувано
у Державному ВНЗ «Національний гірничий університет»
Свідоцтво про внесення до Державного реєстру ДК № 1842
від 11.06.2004 р.

49005, м. Дніпропетровськ, просп. К.Маркса, 19