Міністерство транспорту та зв'язку України
Державний департамент з питань зв'язку та інформатизації України
ОДЕСЬКА НАЦІОНАЛЬНА АКАДЕМІЯ ЗВ'ЯЗКУ ім. О.С. ПОПОВА

Кафедра обчислювальної техніки та мікропроцесорів

Антонов О.С., Хіхловська І.В.

ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА МІКРОПРОЦЕСОРИ

Навчальний посібник
з дисципліни
"Обчислювальна техніка та мікропроцесори"
напряму Телекомунікації

Частина 2

ЗАТВЕРДЖЕНО
Методичною радою факультету
телекомунікаційних систем.
Протокол № 4
від 27 листопада 2007 р.

Одеса 2008

УДК 004 + 00431 + 621.39 (075) План НМВ 2007/2008 навч. р.

Антонов О.С., Хіхловська І.В. Обчислювальна техніка та мікропроцесори. Навчальний посібник. Частина 2. – Одеса: Вид. центр ОНАЗ ім. О.С. Попова, 2008. – 200 с.: іл.

Розглянуто основні принципи побудови й функціонування обчислювальних та мікропроцесорних систем, їхні основні вузли, у тому числі мікропроцесори. На прикладах мікропроцесорів та мікроконтролерів фірми Motorola показані принципи проектування мікропроцесорних систем, у тому числі для цифрового оброблення сигналів та їхнього програмування. Наведено приклади застосування мікропроцесорів та мікроконтролерів різних моделей у пристроях телекомунікацій. Кожен розділ супроводжується запитаннями вхідного та вихідного контролю.

СХВАЛЕНО
на засіданні кафедри
обчислювальної техніки
та мікропроцесорів
і рекомендовано до друку.
Протокол № ____
від ___._____.200__р.

ЗМІСТ

3 МОДУЛЬ
11 МІКРОПРОЦЕСОРНІ СИСТЕМИ НА УНІВЕРСАЛЬНИХ МП ФІРМИ MOTOROLA

Вхідний контроль:

1 Які 32-розрядні універсальні МП фірми Intel Вам знаєте?

2 У яких пристроях обчислювальної техніки вони використовуються?

3 У яких пристроях телекомунікацій вони використовуються?

4 Під якими операційними системами працюють Intel-сумісні мікропроцесори?

5 Як зорганізовується захищений режим МП фірми Intel?

6 Який механізм віртуалізації пам'яті зреалізовують Intel-сумісні процесори?

7 Яку архітектуру – RISC чи CISC – мають універсальні МП фірми Intel?

МП сімейства MC680X0 фірми Motorola належать до універсальних МП з класичною CISC архітектурою і застосовуються у персональних комп'ютерах, серверах, цифрових системах комутації, маршрутизаторах, мобільному зв'язку, контрольно-вимірювальній апаратурі, літакобудівництві, військово-промисловому комплексі, системах промислової автоматики тощо. Фірма Motorola випускає також широкий спектр напівпровідникових пристроїв від транзисторів до надвеликих інтегральних мікросхем. Вироби фірми Motorola відрізняються високими технічними характеристиками, якістю та надійністю, а вихід бездефектної продукції становить 99,9997 %. Все це зумовило широке розповсюдження виробів фірми Motorola на світовому ринку.

Базовою моделлю МП MC680X0 є МП MC68000.

11.1 16-розрядні мікропроцесори фірми Motorola

Вхідний контроль:

1 Які пристрої входять до програмної моделі мікропроцесора I8086?

2 Відмінності фоннеймановської і гарвардської архітектури МП.

3 До якої з архітектур належать МП фірми Intel?

4 Призначення регістра прапорців (ознак результату).

5 В який спосіб змінюється вміст регістра прапорців?

6 Призначення регістра-лічильника команд.

7 В який спосіб змінюється вміст регістра-лічильника при виконуванні різних команд?

8 Назвіть способи адресування МП фірми Intel.

Сімейство 16- та 32-розрядних мікропроцесорів MC680x0 фірми Motorola містить низку процесорів, контролерів та співпроцесорів, які мають однакову базову архітектуру. Ця архітектура складається з наборів однакових для всіх представників сімейства регістрів, внаслідок чого всі пристрої мають ідентичні базові способи адресування, базову систему команд, ідентичні принципи

взаємодії з пам’яттю і взаємодії із зовнішніми пристроями. Для всіх пристроїв сімейства виконується принцип програмної сумісності «знизу-уверх». При цьому системи команд наступних пристроїв доповнюється новими командами і способами адресування, зберігаючи усі базові.

Мікропроцесори цього сімейства використовуються в персональних комп’ютерах Macintosh, а також в розподілених системах керування складними об’єктами, обмін даними з якими відбувається стандартною шиною VME. Для використовування в системах керування розроблено низку модифікацій МП MC68000, які сполучено під назвою MC68EC0x0 (EC – Embedded Controller). До їхнього складу входять: інтегровані процесори, комунікаційні контролери тощо.

Базова модель – мікропроцесор MC68000 – має 16-розрядну зовнішню шину даних та 24-розрядну шину адреси, що дозволяє адресувати простір пам’яті 16 Мбайт.

Максимальна тактова частота для цього процесора – 16,7 МГц. МП MC68000, який продукується за n-МОП-технологією, споживає близько 1,6 Вт, а мікропроцесор MC68HC000, який створено за КМОП-технологією, – 260 мВт. Тактову частоту процесора MC68EC000 доведено до 20 МГц, за споживання потужності 380 мВт.

Залежно від класу розв’язуваних завдань, що вирішуються, програмні й апаратні ресурси системи, в складі якої працюють МП сімейства MC680x0, поділяються в такий спосіб, щоби забезпечити передусім керування роботою власне мікропроцесорної системи за допомогою системного програмного забезпечення, а також задля розв’язування прикладних завдань користувача. Задля розв’язування згаданих завдань зреалізовано два режими МП:

- режим супервізора;
- режим користувача.

В режимі супервізора дозволено виконання усіх команд і забезпечено доступ до усіх регістрів МП. В режимі користувача заборонено виконування деяких команд і обмежено доступ до певних регістрів, задля запобігання внесення таких змін у режимі МП, щоби розв’язання низки завдань стало неможливим.

Поточний режим функціонування визначається значенням біта S регістра стану. При включенні МПС мікропроцесор автоматично встановлює режим супервізора і працює під керуванням операційної системи. Перехід до режиму користувача здійснюється встановлянням біта S до стану логічного 0. У режимі користувача змінювання режиму роботи МП не дозволяється; перехід може статися у разі виникання виняткових ситуацій (переривань) чи внаслідок нового запуску МП (RESET).

До складу регістрової моделі МП MC68000, наведеної на рис. 11.1, входять два набори 32-розрядних регістрів, кожний з котрих вміщує по вісім однакових регістрів, програмний лічильник та регістр стану.

Вісім регістрів даних D7...D0 призначено для зберігання даних.

Вісім регістрів адреси A7...A0 призначено для зберігання адрес комірок пам’яті, визначених програмістом. Регістр A7 (A´7) є дубльований і має

заздалегідь визначене функціональне призначення. Він використовується як вказівник стека. Залежно від режиму функціонування МП, він слугує вказівником стека користувача USP або вказівником стека супервізора – SSP. Це дозволяє розділити стеки при розв'язуванні завдань користувача і супервізора.

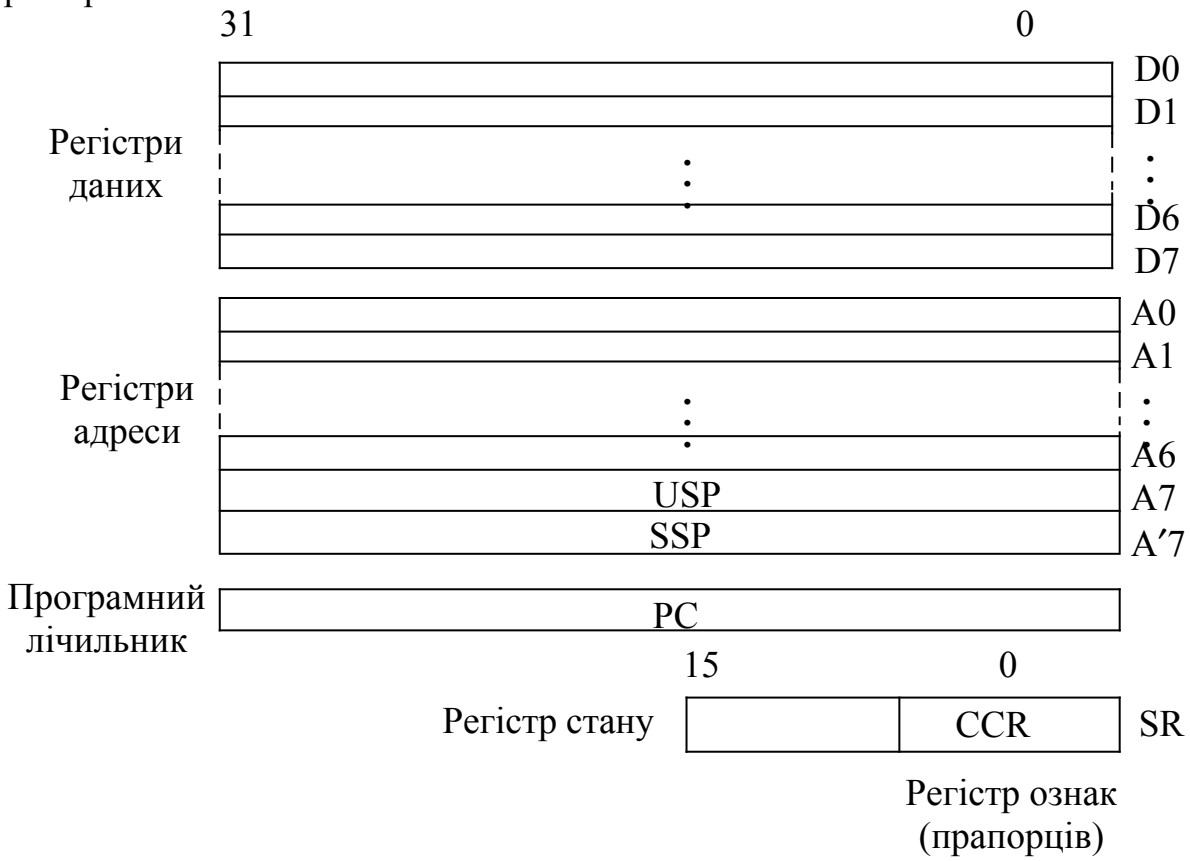


Рисунок 11.1 – Регістрова модель мікропроцесора MC68000

Програмний лічильник PC призначено для зберігання адрес команд. Зважаючи на те, що МП має 24-розрядну шину адрес, в програмному лічильнику використовуються лише 24 розряди.

Регістр стану SR складається з системного байта та байта користувача. Повністю цей регістр є доступний лише в режимі супервізора. В режимі користувача доступ є лише до байта користувача – регістра ознак CCR, який використовується окремо. Формат регістра стану подано на рис. 11.2.

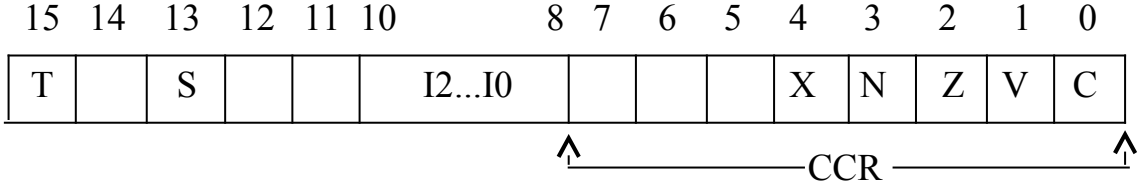


Рисунок 11.2 – Формат регістра стану SR

Регістр ознак (прапорців) CCR складається з восьми тригерів, п'ять з яких призначено для зберігання певних ознак, якими можна схарактеризувати результат виконання операції АЛП.

Окремі біти регістра ознак мають таке призначення:

C (ознака перенесення) – перенесення зі старшого розряду при виконуванні арифметичних операцій та зберігання вмісту висуваного розряду при виконуванні операцій зсувів. Набирає значення C = 1 за виникнення перенесення зі старшого розряду результату виконуваної операції;

V (ознака переповнення) – набуває значення V = 1 в разі переповнення розрядної сітки при обробці операндів зі знаком;

Z (ознака нульового результату) – за здобуття результату, який дорівнює 0, ознака набуває значення 1;

N (ознака знаку) – зберігає копію знакового розряду результату операції. Значення ознаки N = 1 відповідає від'ємному результатові;

X (ознака розширення) – при виконуванні переважної більшості операцій копіює значення біта C. В певних випадках значення цього біта формується залежно від виконуваної операції.

Біти системного регістра SR визначають режим функціонування МП і мають призначення:

T (ознака трасування) – за допомогою цієї ознаки здійснюється перехід до покрокового виконування програми за T = 1;

S (ознака супервізора) – визначає режим роботи МП. Значення ознаки S = 1 відповідає функціонуванню МП в режимі супервізора, а S = 0 – в режимі користувача;

I2...I0 – поле маски переривань, визначає мінімальний рівень обслуговуваних запитів переривань.

Інші біти регістра SR не використовуються або їх зарезервовано задля використовування в наступних моделях.

Мікропроцесор MC68000 обробляє дані – операнди, які можуть бути бітами, байтами, словами та довгими словами, а також дані, які подано у вигляді двійково-десяткових чисел. Операнди можуть розміщуватись у регістрах мікропроцесора (даних чи адреси), а також у комірках пам'яті. При цьому, позаяк шина даних МП має місткість 16 біт, а пам'ять має байтову організацію, то для роботи зі словами чи довгими словами слід задавати адресу старшого байта операнда, яка має бути парною чи кратною до 4. Це дозволяє звертатися спочатку до старшого байта слова, а потім до молодшого, який розміщується у комірці пам'яті, адреса котрої є більша на 1 за адресу старшого байта. Таке послідовне розміщення операндів у пам'яті відповідає звичайному розміщенню розрядів числа при написанні чисел. Такий порядок адресування байтів називається big-endian, на відміну від порядку адресування little-endian, який використовується в МП фірми Intel. Розміщення слів (а) і довгих слів (б) у пам'яті та їхнє адресування подано на рис. 11.3. Адреса N завжди має бути парною.

Підключення ВІС мікропроцесора до шин МПС здійснюється за допомогою виводів, функціональне призначення котрих та їхні умовні назви подано на рис. 11.4 (виводи живлення на схемі не зазначено). Напрям проходження сигналів зазначено стрілками. Підключення МП до шини адреси МПС здійснюється за допомогою 24-розрядної шини адреси (A23...A0), на яку

виставляється адреса пристрою, до якого здійснюється звернення. Команди та дані надходять на 16-розрядну шину даних (D15...D0), якою здійснюється обмін словами. Передавання довгих слів здійснюється за два цикли шини (спочатку старша частина довгого слова, потім молодша).

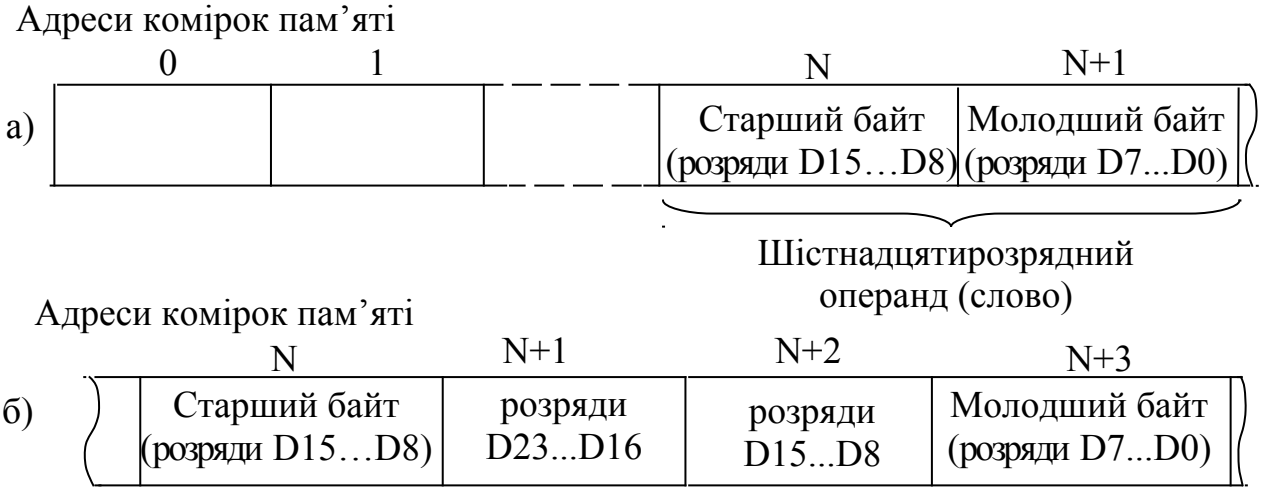


Рисунок 11.3 – Адресування байтів в слові (а) і довгому слові (б)

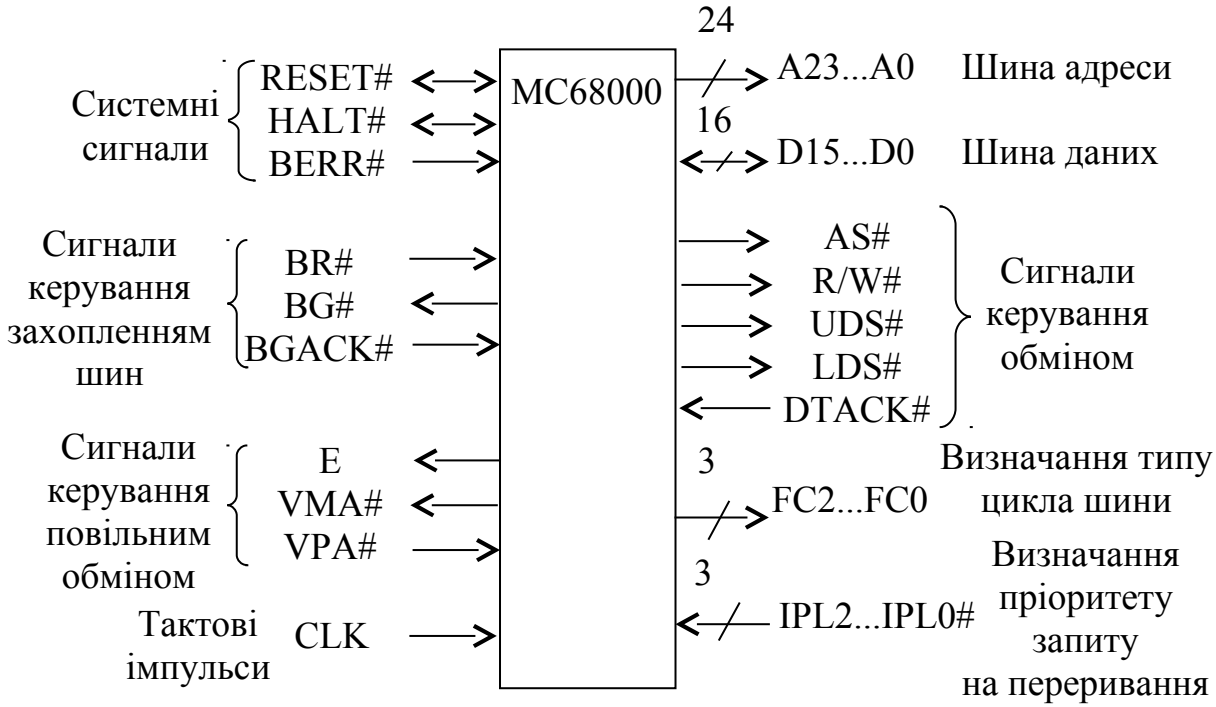


Рисунок 11.4 – Призначення виводів ВІС MC68000

Сигнали керування поділено на групи відповідно до функціонального призначення, як подано на рис. 11.4. Знак # засвідчує, що активним рівнем сигналу є значення логічного 0 (низький рівень).

CLK – сигнал тактової синхронізації. Частота цього сигналу зумовлює тривалість такту машинного часу.

Групу системних сигналів становлять три сигнали:

– RESET# – сигнал скидання. Вихідний сигнал RESET# формується при виконанні команди RESET і встановлює всі пристрої МПС у початковий стан. Вхідний сигнал RESET# призводить до апаратного виключення, яке встановлює початковий стан МП. При цьому вміст регістрів адреси обнулюється, значення біта S встановлюється в 1, що відповідає режимові супервізора. Далі з комірки пам'яті з адресою $000000 завантажується значення вказівника стека супервізора SSP, а з комірки з адресою $000004 до програмного лічильника PC – адреса підпрограми ініціалізації системи. Ця підпрограма завантажує потрібні початкові значення до регістра SR, регістрів адреси та даних, а також виконує ініціалізацію пристроїв, які входять до складу МПС. Програмне скидання виконується в режимі супервізора командою RESET, яка також встановлює початковий стан МП.

– HALT# – сигнал зупину. Вхідний сигнал HALT# припиняє виконання поточної програми, переводить шину адреси та шину даних до відключеного стану, а виходи керування до неактивного стану. Вихідний сигнал HALT# формується за подвійної помилки звернення до шини. Вихід зі стану зупину відбувається подаванням вхідного сигналу RESET#.

– BERR# – вхідний сигнал помилки звернення до шини адреси. Цей сигнал формується схемою контролера шини, якщо на шину адреси подано адресу пристрою, що не існує, або адресу, яка не входить до адресного простору МПС. Цей сигнал також формується за тривалої відсутності сигналу готовності DTACK# і за інших порушень обміну.

Групу сигналів керування обміном подавано п'ятьма сигналами:

– AS# – адресний строб є сигнал, формування якого зумовлює момент завершення формування адреси на шині; активний стан цього сигналу зберігається до завершення циклу обміну.

– R/W# – читання/запис. Значення сигналу зумовлює напрям передавання даних. Сигнал R/W#, котрий дорівнює 1, зумовлює читання (введення) даних в МП; значення сигналу, що він дорівнює 0, – навпаки, виведення (запис) даних з МП.

– UDS# – передавання старшого байта даних.

– LDS# – передавання молодшого байта.

Сигнали UDS#, LDS# зумовлюють розмір і порядок передавання даних шиною. Значення LDS# = 0 водночас з UDS# = 1 відповідає передаванню молодшого байта, LDS# = 0 та UDS# = 0 – передаванню слова.

– DTACK# – вхідний сигнал підтвердження готовності до обміну, який надходить від пам'яті чи зовнішніх пристроїв.

Групу сигналів керування захопленням шин становлять сигнали, які визначають порядок спільного використання шини даних кількома пристроями системи. Ці сигнали використовуються для забезпечення режиму безпосереднього доступу до пам'яті, а також для організації спільної роботи кількох МП у багатопроцесорних системах. До цієї групи входять сигнали:

– BR# – вхідний сигнал будь-якого зовнішнього пристрою на захоплення шини. Отримавши такий сигнал, якщо переривання дозволено, МП

завершує виконання поточного циклу обміну, припиняє виконування команди, переводить власні виводи A23...A0 та D15...D0 до відключеного стану, а виходи керування до неактивного стану.

– BG# – вихідний сигнал дозволу захоплення шин. Формується після відключення МП від шини.

– BGACK# – вхідний сигнал підтвердження захоплення шини зовнішнім пристроєм. Формується зовнішнім пристроєм після отримання сигналу BG# і переходу до режиму керування шиною. Формування сигналу BR# після цього припиняється.

Після завершення обміну зовнішній пристрій встановлює значення сигналу BGACK у 1 і МП продовжує виконання своєї роботи, яку він перервав.

Сигнали керування повільним обміном дозволяють підключати до МП зовнішні пристрої, які мають відносно невелике значення тактової частоти. Для організації роботи з ними використовуються відповідні сигнали керування. До цих сигналів належать:

– E – тактовий сигнал для зовнішнього пристрою. На цьому виході формується сигнал, який має частоту вдесятеро меншу за тактову частоту МП (CLK). Цей сигнал може використовуватися зовнішніми пристроями в якості сигналу тактової частоти. При цьому сигнали адреси й сигнали AS# та R/W# формуються як за звичайного обміну даними.

– VPA# – сигнал готовності до повільного обміну. Цей сигнал формує зовнішній пристрій, адресу якого встановлено на шині адреси і він є готовий до роботи.

– VMA# – сигнал підтвердження повільного обміну. Цей сигнал формує МП у відповідь на отримання ним сигналу VPA#. Після його встановлення розпочинається обмін даними у повільному режимі.

В кожному циклі обміну МП визначає тип циклу, який виконується. Для цього МП формує код, який виставляється на шину визначення типа циклу FC2...FC0. Відповідність типа циклу і коду наведено у табл. 11.1.

Таблиця 11.1 – Типи виконуваних циклів

FC2	FC1	FC0	Тип циклу
0	0	0	резервовано
0	0	1	вибирання даних користувача
0	1	0	вибирання команд користувача
0	1	1	резервовано
1	0	0	резервовано
1	0	1	вибирання даних супервізора
1	1	0	вибирання команд супервізора
1	1	1	підтвердження переривання

Дешифрування цього коду надає можливість розділити пам’ять для режимів супервізора та користувача, сформувати сигнал підтвердження переривання INTA# або ідентифікувати поточний стан МП.

Переривання і виключення – це спеціальні процедури, виконувані МП, якщо при роботі МПС виникають ситуації, котрі потребують тимчасового припинення головної програми й обслуговування інших програм (обробки переривань) задля відповідної реакції системи на обставини, які призвели до цього.

МП може обслуговувати до семи запитів на переривання від зовнішніх пристроїв, які формують сигнали IRQ1...IRQ7, відповідно до свого пріоритету. Ці сигнали оброблюються за допомогою пріоритетного шифратора і надходять на входи IPL#. Відповідність пріоритетів запиту і сигналів IPL2...IPL0# подано у табл. 11.2.

Таблиця 11.2 – Кодування запитів на переривання

Пріоритет	Запити на переривання відповідно до пріоритету	Код запиту IPL2...IPL0
0 (мінімальний)	відсутність запиту	111
1	IRQ1	110
2	IRQ2	101
3	IRQ3	100
4	IRQ4	011
5	IRQ5	010
6	IRQ6	001
7 (максимальний)	IRQ7	000

Виключення. Виключення (Exception) – особливі ситуації при роботі МПС, які є реакцією системи на непередбачувані, переважно аварійні, ситуації, обслуговування яких передбачає переривання головної програми і перехід до підпрограми обробки цього виключення. Виключення бувають апаратними або програмними. Апаратні виключення зумовлено будь-якими порушеннями функціонування пристроїв системи, як периферійних, так і внутрішніх, або аварійними (відключення живлення, відмикання з'єднувальних ліній тощо). Програмні виключення виникають за неправильного функціонування програми: формування неіснуючої адреси, формування помилкового коду команди, ділення на 0, покрокового виконання програми тощо. При обслуговуванні виключень МП автоматично переходить до режиму супервізора, що надає підпрограмі обслуговування виключення доступ до всіх ресурсів системи.

Апаратні виключення зумовлено такими причинами:

- надходження зовнішнього сигналу RESET# ;
- формування зовнішнього сигналу помилки звернення до шини BERR#;
- надходження від периферійних пристроїв запитів на переривання;
- відсутність сигналів підтвердження готовності до обміну DTACK# або сигналу готовності до повільного обміну VPA#;
- надходження запиту на переривання від периферійного пристрою, який попередньо не ініційовано;
- відключення живлення та інші аварійні ситуації.

Програмні виключення виникають за таких причин:

– надходження команд, які спричинюють виключення RESET, TRAP, TRAPV, CHK;

– надходження привілейованих команд у режимі користувача;

– ділення на 0 (команди DIVS, DIVU);

– надходження команди, яка має помилковий код операції;

– встановлення покрокового режиму роботи (встановлення біта T = 1);

При цьому виключення відбувається на кожному кроці виконування програми і МП виводить на екран монітора результати виконання поточної команди й дозволяє змінювати ці результати в перебігу налагодження;

– надходження команди, яка має у полі коду операції значення 1010. Цей код забезпечує перехід до підпрограми-емулятора команд, які не входять до системи команд МП;

– формування непарної адреси.

Обслуговування всіх виключень відбувається однаково: МП переходить до режиму супервізора, встановлюючи значення біта S = 1, потім МП зберігає у стеку вміст програмного лічильника PC і регістра стану SR задля можливості повернення до перерваної програми. Потім до програмного лічильника завантажується вектор виключення (V_e), котрий є адресою початку відповідної програми обслуговування. Якщо виключення зумовлено помилкою шини або формуванням помилкової адреси, то до стека також завантажується значення адреси, яка спричинилася до виключення, код виконуваної команди і код циклу (FC2...FC0), який при цьому виконувався. Ця інформація використовується підпрограмою обслуговування переривання для з'ясування причини, яка зумовила виключення.

Вектори виключень розміщено у пам'яті МПС і поєднано у таблицю виключень. Ця таблиця розміщена в області пам'яті з адресами $000...$3FF. Кожний вектор адресовано двома словами, тому в таблиці є входи для 256 підпрограм виключень. Кожний вектор має власний номер N_e від 0 до 255 і кожному номеру відповідає власна адреса A_e, в якій зберігається вектор V_e відповідної підпрограми. Адреса визначається в результаті множення номера вектора на 4 – ($A_e = 4\ N_e$). Множення відбувається у результаті логічного зсуву на 2 розряди ліворуч. Таблиця різновидів виключень та їхніх векторів подана у табл. 11.3.

Повернення з підпрограми обслуговування виключення здійснюється за допомогою команд RTE або RTS. Командою RTE завершується підпрограма обслуговування виключення в режимі супервізора, а командою RTS, – якщо при виконанні підпрограми обслуговування МП перейшов до режиму користувача.

Виключення з номером $N_e = 0$ може бути програмним, за надходження команди RESET, або апаратним при формуванні сигналу зовнішнього скидання RESET#. В обох випадках з комірки пам'яті з адресою $000 завантажується значення вказівника стека супервізора – SSP, а з комірки пам'яті з адресою $004 – початкова адреса підпрограми ініціалізації всієї системи. Отже, це виключення має два вектори виключення V_e.

Таблиця 11.3 – Різновиди виключень та їхні адреси

Номер виключення N_e	Адреса виключення A_e	Вид виключення
0	$000	Встановлення початкового стану (завантаження SSP)
–	$004	Встановлення початкового стану (завантаження PC)
2	$008	Помилка звернення до шини
3	$00C	Формування помилкової адреси (непарної)
4	$010	Помилковий код операції
5	$014	Ділення на 0
6	$018	Команда CHK
7	$01C	Команда TRAPV
8	$020	Порушення привілеїв
9	$024	Покроковий режим (T = 1)
10	$028	Код емуляції 1010
11	$02C	Код емуляції 1111
12...14	$030...$038	Резервовано
15	$03C	Неініційоване переривання
16...23	$040...$05E	Резервовано
24	$060	Помилкове переривання
25...31	$064...$07C	Автовекторні переривання
32...47	$080...$0BF	Команди TRAP з номерами N_t = 0...15
48...63	$0C0...$0FF	Резервовано
64...255	$100...$3FF	Векторні переривання користувача

Виключення з номером N_e = 2 (помилка звернення до шини) здійснюється за сигналом спеціального пристрою, якщо на шині адреси формується адреса неіснуючого (непідключеного) пристрою. Також це виключення може відбуватися за сигналом вартового таймера, якщо певного часового інтервалу не формується сигнал підтвердження готовності до обміну DTACK# від пам’яті або зовнішніх пристроїв. Підпрограма обслуговування цього виключення визначає тип помилки, адресу і команду, яка зумовила виключення. Якщо при обслуговуванні виникає повторний сигнал помилки звернення, то МП переходить до режиму зупину і формує відповідний вихідний сигнал HALT#.

Виключення кодів емуляції 1010 та 1111 дозволяють виконувати команди, які не входять до системи команд МП MC6800. Так, підпрограма обслуговування виключення, зустрівши команду, яка має код 1010, аналізує інші розряди слова команди і залежно від результатів аналізу виконує певну послідовність команд, яка відповідає виконанню макрокоманд, що розширює можливості МП і сприяє реалізації макроасемблерів. Код емуляції 1111 використовується для реалізації команд арифметичного співпроцесора. Отже,

існує можливість за допомогою МП MC68000 налаштовувати і виконувати програми, які використовують команди арифметичного співпроцесора.

Виключення з номерами 15 (неініційоване переривання) і 24 (помилкове переривання), а також векторні й автовекторні переривання обслуговуються при надходженні запитів від зовнішніх пристроїв.

Номери виключень команди TRAP формуються відповідно до виразу $N_e = 32 + D_t$. Це виключення використовується для завдання контрольних точок зупину при налагоджуванні програми, а також задля включення підпрограм обслуговування пристроїв, які використовуються спільно з МП.

Переривання (Interruption) – особливі ситуації, які виникають при функціонуванні МПС для обслуговування певних пристроїв, робота яких відбувається за участі центрального процесора і не передбачена програмою, яка виконується. Як було розглянуто у розділі 7.2.2, вони виникають при надходженні відповідних команд (програмні переривання) або сигналів від зовнішніх пристроїв (апаратні переривання).

Для обслуговування переривань МП припиняє виконання головної програми, зберігає у стеку адресу останньої команди, яку він виконав (адреса повернення), завантажує початкову адресу підпрограми обслуговування переривання і розпочинає її виконувати. Після завершення підпрограми МП поновлює у програмному лічильнику адресу повернення і продовжує виконання головної програми. Обслуговування переривання розпочинається за запитом на переривання, який формується пристроєм, що він потребує обслуговування.

Зовнішні пристрої, які можуть спричинити переривання, поділено відповідно до важливості їхнього функціонування і кожному з них призначено власний пріоритет обслуговування. МП може обслуговувати до семи запитів на переривання від різних пристроїв. Кодування запитів наведено у табл. 11.2.

Обслуговування зовнішніх пристроїв відбувається відповідно вмісту поля маски переривання (I2...I0) у регістрі SR. Це поле визначає рівень пріоритету пристрою (IRQ) і дозволяє чи не дозволяє його обслуговування. Відповідність коду запитів і можливості його обслуговування наведено у табл. 11.4.

Таблиця 11.4 – Кодування можливості обслуговування переривань

Запит на переривання	Маска I2...I0	Запити, що обслуговуються
відсутність запиту	000	IRQ1...IRQ7
IRQ1 ($L_i = 1$)	001	IRQ2...IRQ7
IRQ2 ($L_i = 2$)	010	IRQ3...IRQ7
IRQ3 ($L_i = 3$)	011	IRQ4...IRQ7
IRQ4 ($L_i = 4$)	100	IRQ5...IRQ7
IRQ5 ($L_i = 5$)	101	IRQ6...IRQ7
IRQ6 ($L_i = 6$)	110	IRQ7
IRQ7 ($L_i = 7$)	111	IRQ7

Адресування підпрограм обслуговування запитів переривань може виконуватися у два способи. При автовекторному перериванні кожному запитові з рівнем L_i відповідає номер виключення згідно з табл. 11.3 з адресами A_e = \$064...\$07C. За векторного переривання зовнішній пристрій формує і передає до МП будь-яке значення номера N_e. Пристрої, які можуть спричинити векторні переривання, мають спеціальний векторний регістр IVR, в якому зберігається відповідне значення N_e, яке записується до нього при ініціалізації. Якщо ініціалізацію не проведено, то пристрій формуватиме код з номером N_e = 15 (неініціалізоване переривання).

Кожне переривання розпочинається з того, що пристрій, який потребує обслуговування, формує запит IRQ#, котрий надходить на пріоритетний шифратор, який формує сигнали коду запиту IPL0...IPL2, відповідно до призначеного пріоритету пристрою. Якщо код запиту є не менший за значення I2...I0 у регістрі SR, то МП завершує виконання поточної команди і виконує цикл підтвердження переривання. В цьому циклі МП формує на виводах FC2...FC0 код 111 – підтвердження переривання (див. табл. 11.1), який перетворюється на сигнал IACK#, котрий відповідно до адреси, надходить на пристрій, який потребує обслуговування. Приклад схеми формування сигналів наведено на рис. 11.5.

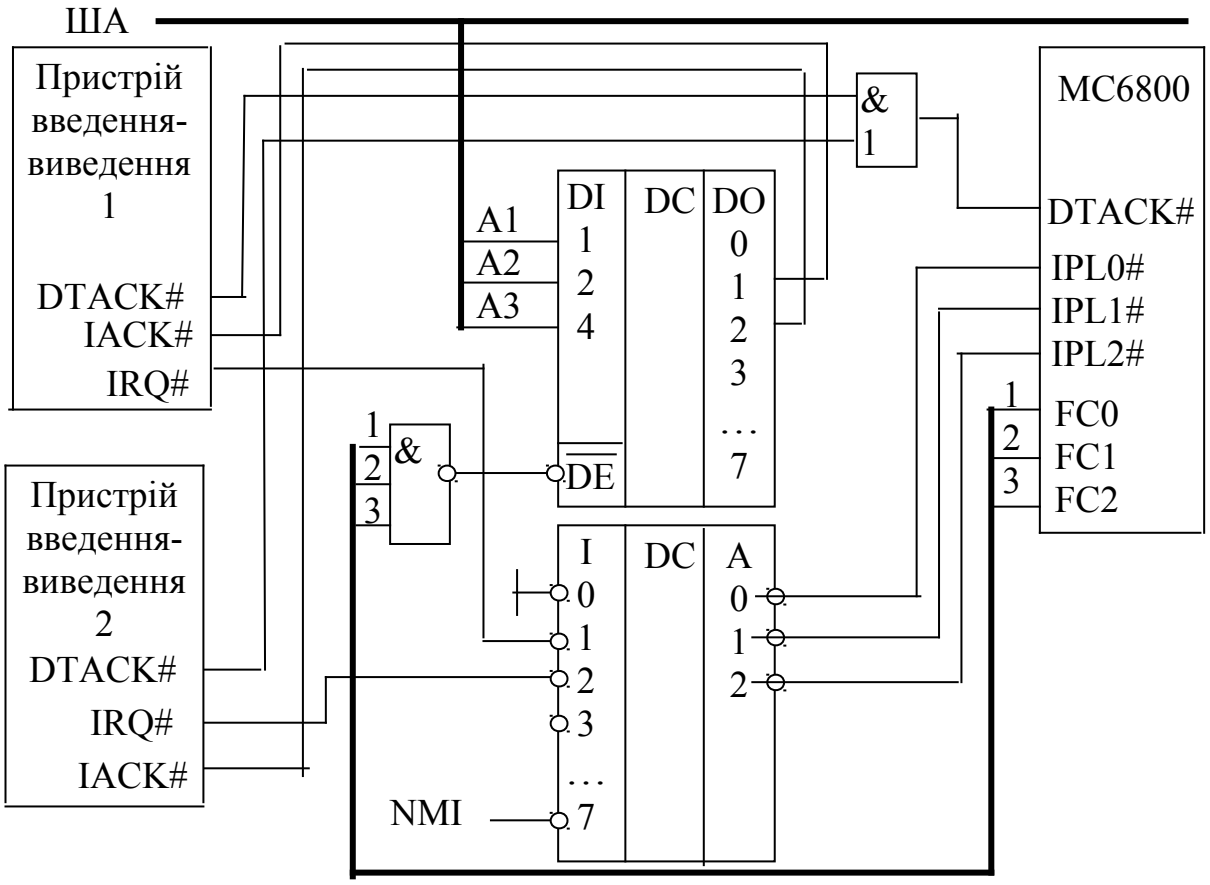


Рисунок 11.5 – Схема формування сигналів переривання

При виконуванні автовекторного запиту на переривання зовнішній пристрій, який отримав сигнал IACK#, підтверджує свій запит, формуючи сигнал низького рівня на вході VPA# мікропроцесора. Після цього МП формує

адресу вектора виключення A_V = \$064...\$07C відповідно до рівня пріоритету пристрою, котрий потребує обслуговування.

При опрацьовуванні векторного переривання зовнішній пристрій, який отримав сигнал IACK#, встановлює на молодші розряди шини даних D7...D0 свій номер виключення N_e і підтверджує передання номера, формуючи сигнал DTACK#. МП приймає значення номера і відповідно до нього формує адресу вектора виключення A_V = \$100...\$3FF.

Якщо при підтвердженні переривання не буде сформовано сигнали VPA# або DTACK#, то МП буде виконувати виключення з номером 24 – «помилкове переривання».

Наступна модель MC68010 передбачає підключення мікросхеми диспетчера пам'яті MC68851 (Memory Management Unit – MMU).

Блок MMU зорганізовує сторінкове адресування пам'яті з розміром сторінки від 256 байт до 32 кбайт. Адреса комірки пам'яті, яку формує МП, сприймається як ефективна або виконавча. На виходи MMU ця адреса транслюється як фізична. Якщо логічна адреса відповідає сторінці, яка є відсутня в оперативній пам'яті, то формується сигнал помилки звернення до пам'яті BERR#. Цей сигнал спричинює переривання, – і операційна система виконує перезапис вмісту відсутньої сторінки з жорсткого диску до адресованої оперативної пам'яті замість сторінки, до якої довш за все не було звернення. Вміст старої сторінки переписується на жорсткий диск, а МП обирає команду або дані зі сторінки, перенесеної до оперативної пам'яті. Зреалізований в такий спосіб механізм віртуалізації пам'яті дозволяє здійснювати обмін сторінками із зовнішньою пам'яттю великого обсягу, використовуючи оперативну пам'ять меншого обсягу. Сигнал BERR# формується також у разі помилки при заданні адреси пам'яті або пристроїв введення-виведення.

Контрольні запитання:

1 Чим відрізнюється програмна модель МП MC68000 від програмної моделі фірми Intel?

2 В яких двох режимах може працювати МП MC68000?

3 Чим вони відрізнюються?

4 Які регістри входять до регістрової моделі користувача МП MC68000?

5 В який спосіб можна перейти до режиму супервізора при роботі з МП MC68000?

6 З даними якої розмірності працює МП MC68000 і в який спосіб відбувається звернення до шини даних при роботі з кожним з них?

7 Як виконується адресування байтів даних при обробці слів та довгих слів?

8 Які сигнали визначають розмірність даних, котрі оброблюються командою?

9 В чому полягає призначення сигналу BERR# і за яких умов відбувається його формування?

Контрольні запитання підвищеної складності:

1 Якими сигналами обмінюються МП та зовнішній пристрій за взаємодії за різних способів обміну?

2 Які типи циклів роботи може зреалізовувати МП МС68000?

3 Чим відрізнюються тип циклу вибору даних супервізора від вибору даних користувача?

11.2 Побудова МПС на 16-розрядних мікропроцесорах фірми Motorola

11.2.1 Підсистема центрального процесорного елемента МС68000

Вхідний контроль:

1 Яку розрядність мають ШД та ША МП МС68000?
2 Які системні сигнали ВІС МП68000 Вам відомі?
3 Чи є згадані в п. 2 сигнали односпрямовані або двоспрямовані й чому?
4 За яким алгоритмом працює пріоритетний шифратор?
5 На якій частоті працює МП МС68000?

До підсистеми центрального процесорного елемента (ЦПЕ) МПС входять пристрої (ВІС), які забезпечують його роботу, сам мікропроцесор МС68000 і пристрої, до яких належать:

– генератор тактових імпульсів, який формує послідовність імпульсів тактової частоти для всієї МПС;

– формувач сигналів керування МПС, який формує всі сигнали, необхідні для вибору вузлів МПС, керування вибором розрядності операндів, контролю за формуванням адреси пристроїв, перериваннями тощо;

– буфер шини даних – пристрій, який забезпечує необхідний рівень навантажувальної здатності виходів шини даних ВІС МС68000. Він являє собою двоспрямований приймач-передавач, який підключається до виходів центрального процесора (ЦП).

Умовне графічне позначення ВІС МС68000 наведено на рис. 11.6. На цьому рисунку також подано рекомендоване фірмою підключення виводів ВІС до джерела живлення. Призначення виводів відповідає рис. 11.4.

Формування сигналів синхронізації та скидання (RST) виконується за допомогою схеми генератора тактових імпульсів, поданої на рис. 11.7. В схемі використовується мікросхема МС88916 фірми Motorola. Для стабілізації частоти використовується кварцовий резонатор Z1. Підключення виводів генератора до ЦП проводиться відповідно до назв виводів; з’єднуються лінії, що мають однакові назви. Сигнал тактової частоти подається на ЦП й інші пристрої схеми.

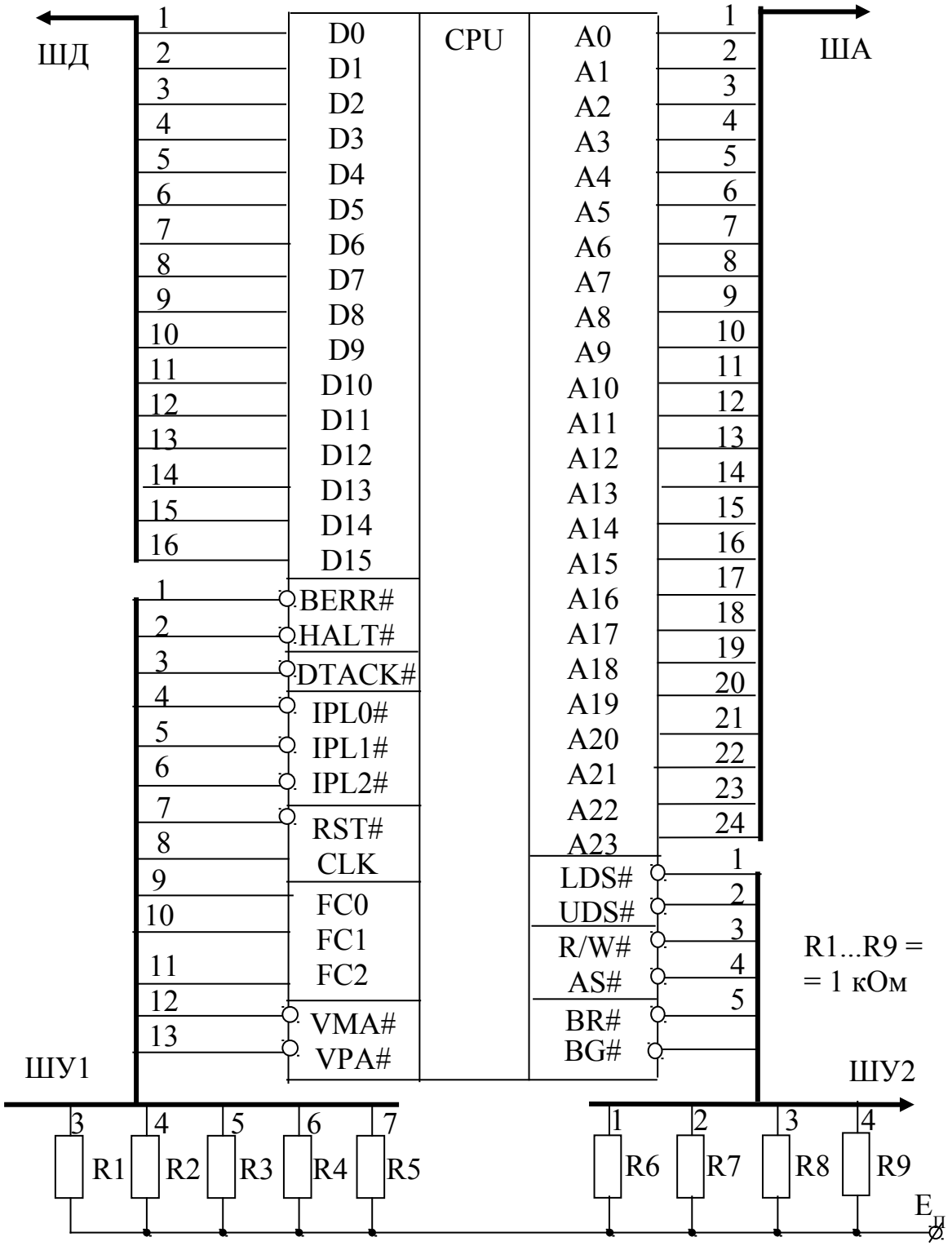


Рисунок 11.6 – Умовне графічне позначення і принципова схема підключення ВІС МС68000

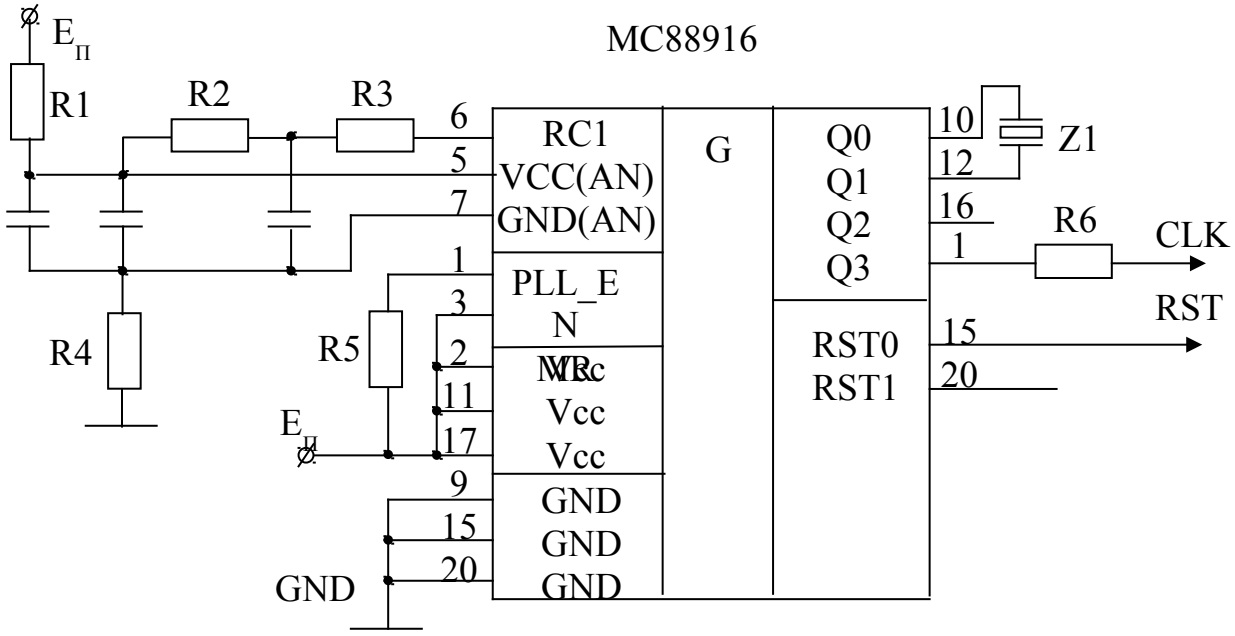


Рисунок 11.7 – Схема генератора тактових імпульсів

Виходи шини даних ВІС MC68000 не мають вбудованих підсилювачів потужності вихідних сигналів, тому для використання у МПС ця шина потребує використання спеціальних схем – буферів. В якості таких схем рекомендовано використовувати приймачі-передавачі 74F245, які є 8-розрядними буферними схемами. Кількість мікросхем визначається розрядністю шини даних ВІС MC68000 – 16, тому в МПС слід використовувати дві мікросхеми 74F245, одна з котрих обслуговує молодший байт шини, а друга – старший. Обробка довгих слів здійснюється за два такти, тому молодша і старша частини будуть обслуговуватися окремо. Для керування роботою схеми слід дешифрувати сигнали UDS, LDS, а також сигнали формування типу циклу. Сигнали керування буфером шини даних на схемі позначено PDEN0, PDEN1. Принципову схему буфера шини даних наведено на рис. 11.8.

Сигнали керування МПС формуються при дешифруванні сигналів ЦП та сигналів адреси. Отже, ця схема являє собою низку різних дешифраторів, підключених до відповідних кіл схеми. Для уніфікації схеми та зручності використання рекомендовано використовувати програмовану логічну інтегральну схему FPGA, запрограмовану відповідно до алгоритмів роботи всіх необхідних дешифраторів та інших схем, потрібних для керування МПС. Умовне графічне позначення програмованої логічної інтегральної схеми FPGA подано на рис. 11.9. Також на цьому рисунку подано вхідні й вихідні сигали на виводах схеми FPGA.

Приміром, сигнали BYTE0 (1,3) формуються при дешифруванні сигналів UDS, LDS, відповідно до табл. 11.5. Обробка довгих слів здійснюється за два такти, під час виконання котрих зберігається значення коду. Значення сигналу BYTE0 зберігається при обробці байтів, слів та довгих слів, а BYTE1 – при обробці слів та довгих слів, що дозволяє спрощувати організацію багатошарової пам’яті.

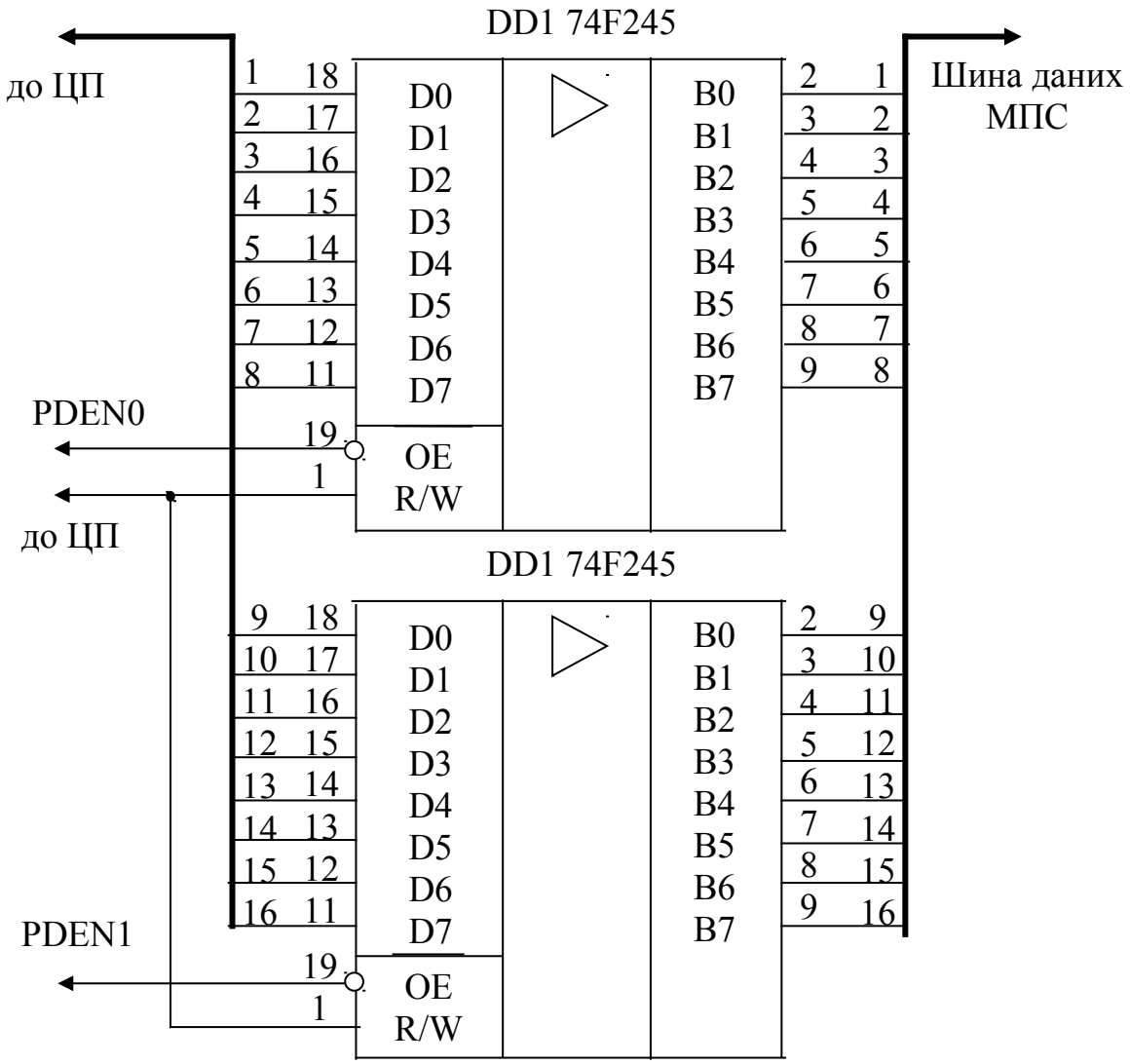


Рисунок 11.8 – Принципова схема буфера шини даних

Таблиця 11.5 – Визначення довжини операнда

LDS	UDS	
0	0	Обробка слова
0	1	Обробка байта

Сигнали BACK1 (2, 3, 4, 5) використовуються для керування пристроями, які входять до складу МПС, формуючи сигнали, котрі переводять певний пристрій до активного стану. Для формування цих сигналів використовуються сигнали адреси і виконуваного циклу. Визначено, що сигнал BACK1 відповідає переведенню до активного стану ПЗП, BACK2 – ОЗП, BACK3 – асинхронний послідовний приймач-передавач, BACK4 – паралельний периферійний інтерфейс, BACK5 – таймер.

Умовне графічне позначення схеми FPGA, сигнали і виводи на яких вони формуються, а також адреси підключення показано на рис. 11.9.

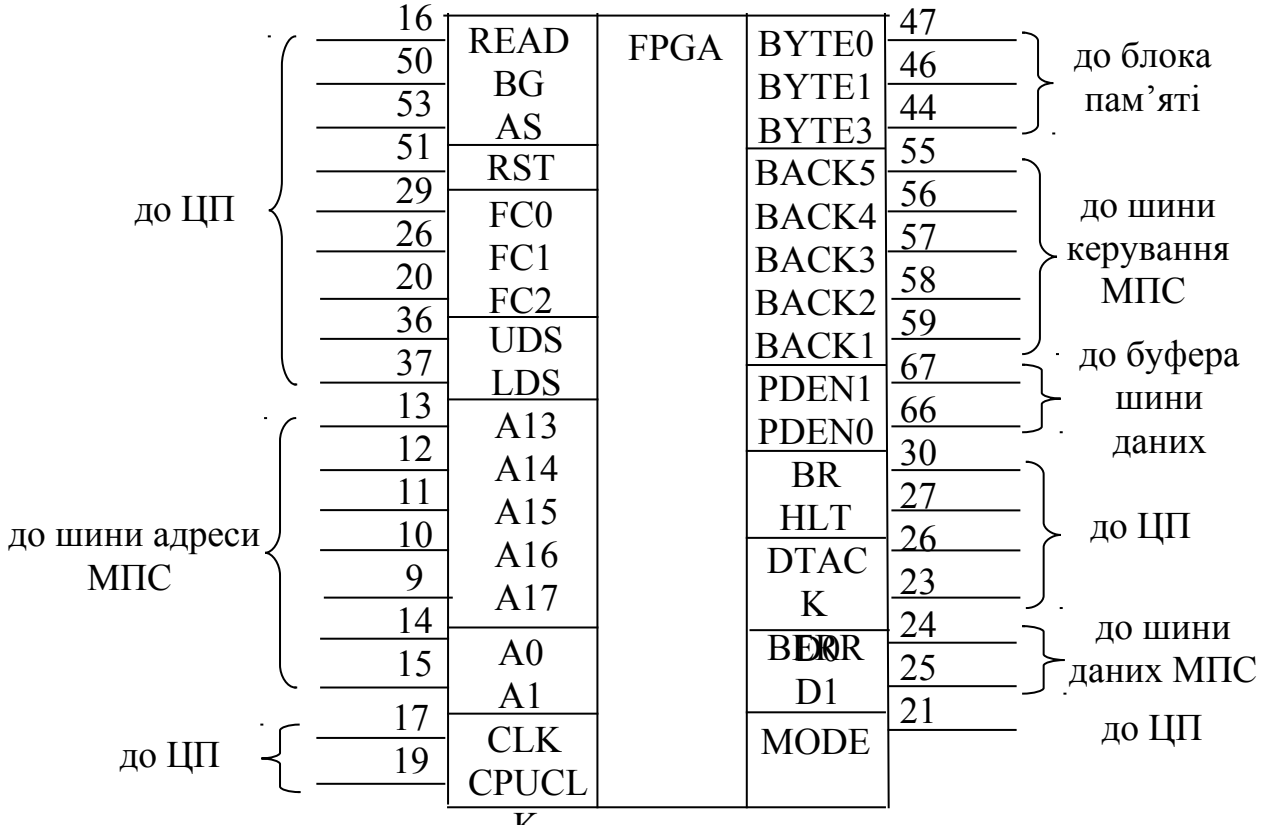


Рисунок 11.9 – Умовне графічне позначення ВІС FPGA

Контрольні запитання:

1 З якою метою до підсистеми ЦП MC68000 включено буфер шини даних?

2 Чим відрізнюються сигнали на входах і виходах буфера шини даних?

3 Чим зумовлюється стабільність частоти тактового генератора MC88916?

Контрольні запитання підвищеної складності:

1 Для чого використовуються сигнали BACK5…BACK1?

2 Для чого на схему FPGA надходять сигнали адреси A17…A13?

3 В чому полягає призначення сигналу BERR, який формується схемою FPGA?

4 Для формування яких сигналів використовуються сигнали UDS, LDS?

5 Чи використовуються сигнали FC2...FC0 для реалізації переривань?

11.2.2 Розподіл адресного простору МПС

Вхідний контроль:

1 З якою метою адресний простір розподіляється поміж різних типів підсистем МПС?

2 Які вимоги визначають розподіл адресного простору між підсистемами МПС?

3 Які підсистеми МПС входять до адресного простору?

4 Чим визначається максимальний обсяг адресного простору МПС?

Кожна з підсистем МПС має власну область адрес у адресному просторі МПС. Розподіл адресного простору визначається мапою адрес, котра задає межі адресного простору для кожної з підсистем. Розподіл адресного простору МПС на базі MC68000 здійснено відповідно до мапи адрес материнської плати інтегрованої платформи M68EC0X0IDP. Мапу адрес для МПС наведено на рис. 11.10. Адресний простір, відповідно до мапи, розподіляється поміж модулями: постійного запам'ятовувального пристрою (ПЗП) – ROM, оперативного запам'ятовувального пристрою (ОЗП) – RAM, послідовного порту (інтерфейсу) – DUART, паралельного порту (інтерфейсу) та таймера – PI/T.

Розподіл адресного простору зреалізовано на ВІС FPGA, яка була розглянута вище, і сигнали BACK1 (2, 3, 4, 5) формуються відповідно до мапи. Звернення до ПЗП під час включення і при перезавантаженні системи виконується апаратно – сигнал BACK1 у ці моменти формується примусово.

Рисунок 11.10 – Мапа розподілу адресного простору МПС MC68000

Системний ПЗП призначено для зберігання системних програм: самотестування, початкового завантаження системи тощо.

Область системного таймера призначено для обслуговування вбудованого таймера, який визначає часові інтервали при роботі системи. Він є енергонезалежним і продовжує функціонувати за відключення живлення.

Контрольні запитання:

1 Визначте інформаційну ємність кожної з областей мапи розподілу адресного простору МПС МП MC68000?

2 Які підсистеми МПС позначено в адресному просторі?

3 Яка з підсистем МПС займає більший адресний простір і чому?

Контрольні запитання підвищеної складності:

1 Призначення програмованої логічної інтегральної схеми FPGA у складі підсистеми центрального процесорного елемента MC68000.

2 Для керування якими пристроями використовуються сигнали BACK1 (2, 3, 4, 5), які формуються на виходах програмованої логічної інтегральної схеми FPGA?

3 За допомого яких сигналів здійснюється визначення довжини даних, які оброблюються?

11.2.3 Організація підсистеми пам'яті

Вхідний контроль:

1 Якими параметрами схарактериззовуються пристрої пам'яті?

2 Визначте кількість мікросхем RAM, необхідних для побудування блока пам'яті з організацією 64К×8, з мікросхем, які мають організацію 32К×4?

3 Скільки входів адреси повинна мати мікросхема ROM з організацією 32Кх8?

4 Які сигнали необхідні для запису інформації до мікросхем пам'яті RAM?

5 Які сигнали необхідні для зчитування інформації з мікросхеми пам'яті RAM?

6 Яких станів можуть набувати вихідні сигнали тристабільних мікросхем пам'яті?

7 В який спосіб взаємодіють сигнали OE, CS, CE поміж собою для виконання запису до мікросхеми пам'яті RAM?

Побудова підсистеми пам'яті здійснюється відповідно до положень розділу 5 даного підручника, котрі доповнюються тим, що ПЗП і ОЗП МПС MC68000 повинні працювати з даними, які можуть бути байтами, словами та довгими словами. Відповідно до цього, водночас можливе звернення до однієї, двох чи чотирьох комірок пам'яті. Шина даних в МПС MC68000 16-розрядна, і робота з довгими словами виконується за два цикли шини, тому при роботі з байтами і словами відбувається звернення до комірок пам'яті з однією адресою, а молодше і старше слова довгих слів розміщуються у двох сусідніх парах комірок. Для реалізації такого принципу побудови пам'яті необхідно будувати пам'ять з чотирьох блоків, кожен з яких призначено для роботи з байтами даних, поєднуючи їх відповідно до довжини операндів. Організацію такої пам'яті подано на рис. 11.11. Блоки ПЗП та ОЗП будуються в однаковий спосіб.

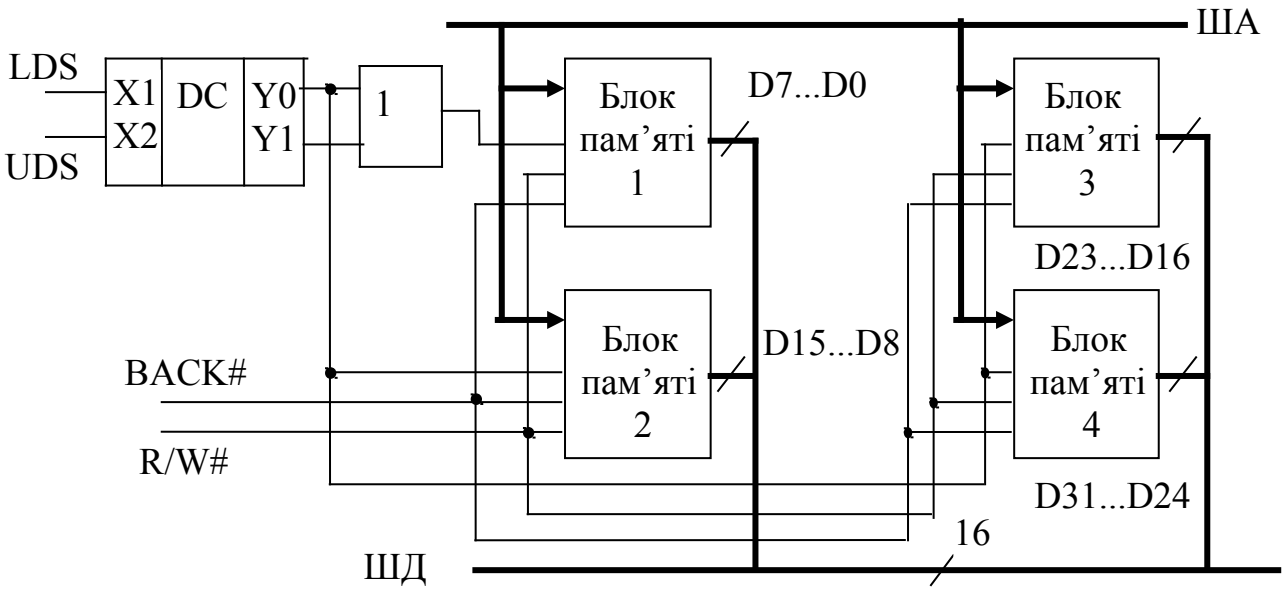


Рисунок 11.11 – Організація чотириблокової пам'яті

В цій пам'яті всі блоки пам'яті до шини адреси під'єднуються паралельно (до одних і тих самих розрядів), що забезпечить звернення до комірок з однаковими номерами. Сигнали вибору блока (BACK) і читання/запис (R/W) також надходять одночасно. Вибір відповідного блока здійснюється за допомогою дешифратора і схеми АБО. Якщо на входи дешифратора надходить код 00 (робота зі словами), то активний сигнал формується лише на виході Y0 і надходить на відповідні входи всіх блоків (на вхід блоку 1 він проходить через логічну схему АБО). Отже, водночас всі 16 виводів двох блоків пам'яті будуть з'єднані з шиною даних. Якщо на входи дешифратора надходить код 01 (робота з байтом), то сигнал Y1 дозволить роботу лише блокові 1; інші блоки в цей момент будуть перебувати в режимі зберігання інформації (будуть неактивними). Для нарощування інформаційної ємності блока пам'яті збільшується кількість однотипних мікросхем пам'яті у кожному з блоків, відповідно до положень розділу 5.5 підручника. Сукупність мікросхем пам'яті в усіх чотирьох блоках, які обслуговують однакові адреси, можна назвати шаром пам'яті. Отже, якщо кожен з блоків пам'яті складається з кількох мікросхем пам'яті, то можна говорити про використовування багатошарової пам'яті.

Побудова кожного з блоків пам'яті здійснюється відповідно до положень розділу 5.5 підручника. Припустимо, що треба побудувати ОЗП з організацією 115К×8 з мікросхем АМ21С512, які було розглянуто в розділі 5.5. ОЗП має працювати з байтами, словами та подвійними словами. Кількість мікросхем, потрібних для побудови визначатиметься за виразом

$$N = 4\frac{115K \times 8}{64K \times 8} = 7{,}1872 \approx 8,$$

де 4 – кількість блоків пам’яті; 115К×8 – організація ОЗП кожного з блоків; 64К×8 – організація мікросхеми АМ21С512.

Якщо у результаті здобуто дробове число, то його слід заокруглювати **обов’язково** до більшого цілого числа.

Отже, блок ОЗП вміщуватиме чотири блоки пам’яті, кожен з яких складатиметься з двох мікросхем. Схему цієї пам’яті наведено на рис. 11.12.

Керування схемою здійснюється сигналами, які формуються ВІС FPGA. Сумарний обсяг пам’яті кожного блока 128К. Блок 1 може працювати з байтами, сумісно з блоком 2 – зі словами і всі чотири блоки – з довгими словами. Молодша частина довгого слова оброблюється блоками 1 та 2, старша – блоками 3 та 4. Керування шарами пам’яті в кожному з блоків здійснюється за сигналом адреси А16 у такий спосіб, що молодші комірки пам’яті з адресами $00000...$0FFFF оброблюються верхньою (за схемою) мікросхемою, а старші з адресами $10000...$1FFFF – нижньою. Для керування використовується вхід ОЕ, на котрий подається або сам сигнал А16, або його інверсія. Отже, якщо на вхід верхньої мікросхеми надходить сигнал безпосередньо з шини адреси, то в діапазоні адрес $10000...$1FFFF робота цього блока буде забороненою.

Контрольні запитання:

1 Чому при використовуванні МП МС68000 рекомендовано використовувати пам’ять, яка складається з чотирьох блоків?

2 В який спосіб організовується робота блоків при роботі з даними різної розрядності?

3 Який пристрій керує роботою блоків пам’яті МПС на МП МС68000?

4 Для чого використовується багатошарова пам’ять?

5 Чи є обов’язковим використовування багатошарової пам’яті?

6 За допомогою яких сигналів та пристроїв здійснюється вибір шарів блоків пам’яті?

Контрольні запитання підвищеної складності:

1 Розробити схему ПЗП з організацією 128×8, задля зберігання даних у вигляді байта, слова і довгого слова, використовуючи мікросхеми пам’яті, які наведено в розділі 5.5.

2 Які сигнали керують вибором шару для роботи блоків пам’яті і з якого пристрою вони надходять?

3 Як взаємодіють поміж собою блоки пам’яті при роботі з операндами різної розрядності?

4 За скільки циклів 32-розрядні дані може бути записано до пам’яті RAM?

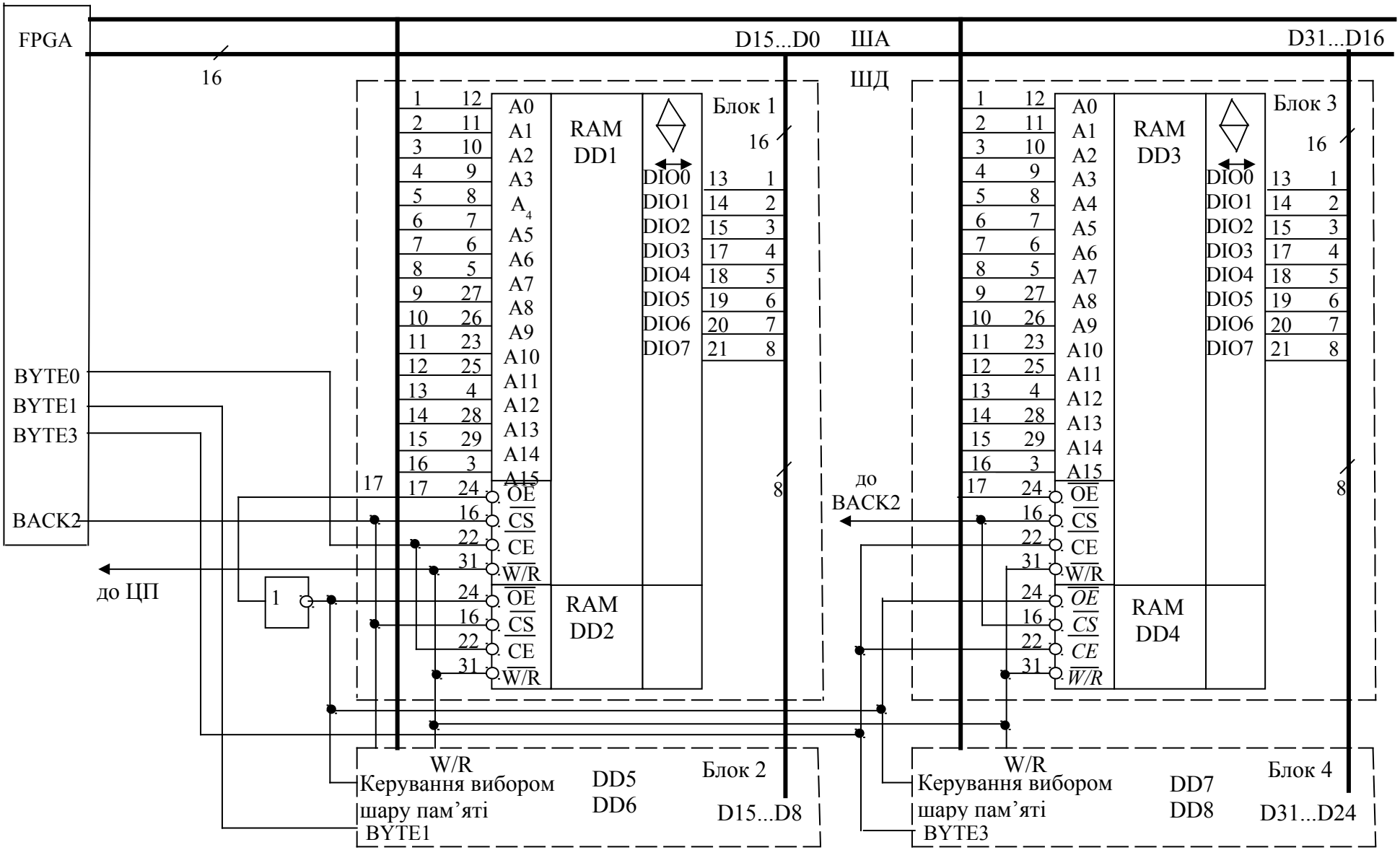


Рисунок 11.12 – Принципова схема блока пам'яті

11.2.4 Організація підсистем введення-виведення

Вхідний контроль:

1 Чим відрізняються паралельний і послідовний спосіб обміну даними у МПС?

2 Даними якої розмірності може обмінюватися МПС через послідовний порт RS-232C?

3 Які службові символи можуть входити до складу інформації, яка передається через послідовний порт RS-232C?

4 В чому полягає різниця поміж синхронним і асинхронним способами обміну інформацією?

5 В який спосіб слід зорганізувати послідовний обмін інформацією поміж МПС, якщо в кожній з них інформацію подано у паралельному коді?

6 У яких випадках застосовується паралельний обмін даними поміж МПС та периферійними пристроями й чому?

7 У яких випадках дані поміж МПС та периферійними пристроями пересилаються у послідовному коді й чому?

8 Які послідовні адаптери та приймачі-передавачі Вам є відомі?

9 З яких бітів складається формат даних у послідовному адаптері RS-232-C?

Асинхронний послідовний інтерфейс DUART – MC68681 фірми Motorola

BIC фірми Motorola MC68681 (DUART – DUAL ASYNCHRONOUS RESEIVER/TRANSMITTER) – подвійний асинхронний приймач-передавач, призначений для організації двох незалежних послідовних асинхронних дуплексних каналів обміну (A та B) із зовнішніми пристроями; також за його допомогою існує можливість організації обміну службовою інформацією через 6-розрядний паралельний порт приймання і 8-розрядний паралельний порт введення-виведення інформації.

DUART забезпечує такі можливості як:

– організація двох незалежних асинхронних дуплексних послідовних каналів введення-виведення;

– робота кожного з каналів на 18-х фіксованих швидкостях обміну, які можуть програмуватися незалежно для кожного з каналів;

– робота з даними, довжина котрих може становити від 5 до 8 бітів даних;

– у повідомлення встановлюються стартовий і один-два стопових біти, додатково може використовуватися біт паритету;

– можуть вибиратися такі режими роботи каналів:

нормальний (повний дуплекс);
автоматичного ехо-сигналу;
місцева кільцева перевірка;
віддалена кільцева перевірка;

– робота в режимі селекторного виклику;

– багатофункціональний 16-розрядний лічильник/таймер може працювати у двох режимах:

формування часових інтервалів;

генерування імпульсів;

– 6-розрядний паралельний порт введення може слугувати за багатофункціональний пристрій введення інформації або використовувати чотири входи для вимірювання часових характеристик сигналів;

– багатофункціональний 8-розрядний паралельний порт введення-виведення дозволяє індивідуально виконувати встановлення інформації на кожному з розрядів і задавати режим роботи;

– порт забезпечує багатовекторну систему переривань (до восьми пристроїв);

– визначення помилок роботи (зупин лінії, фальшстарт, надходження переривання тощо);

– робота від внутрішнього генератора з кварцовим стабілізуванням частоти або від зовнішнього генератора імпульсів;

– ВІС сумісна з ТТЛ-логікою;

– живлення від джерела +5 В.

Структурну схему DUART наведено на рис. 11.13. До структурної схеми входять:

– системний інтерфейс – схема, яка забезпечує керування роботою вузлів DUART, а також керування обміном інформацією з ЦП. За допомогою цієї схеми проводиться обмін сигналами поміж ЦП і вузлами DUART. На схему надходять сигнали:

R/W# – читання /запис;

CS# – вибір мікросхеми, при надходженні цього сигналу ВІС переводиться до активного стану;

RESET# – сигнал скидання;

RS4-RS1 (Register Select) – відповідні розряди шини адреси, які визначають адресу внутрішнього вузла регістра DUART;

DTACK# – сигнал підтвердження готовності ВІС до обміну, підтверджує наявність інформації на шині даних;

– буфер шини даних – схема, котра забезпечує правильність обміну поміж шиною даних МПС і внутрішньою шиною DUART;

– схема керування перериваннями – забезпечує роботу за перериваннями. На її входи надходить сигнал IACK# – сигнал підтвердження переривання – і формується сигнал IRQ# – номер запиту на переривання. До схеми входять такі вузли:

IMR (Interrupt Mask Register) – регістр маски переривання;

ISR (Interrupt Status Register) – регістр стану переривання;

ACR (Auxiliary Control Register) – допоміжний регістр керування;

IVR (Interrupt Vector Register) – регістр вектора переривань;

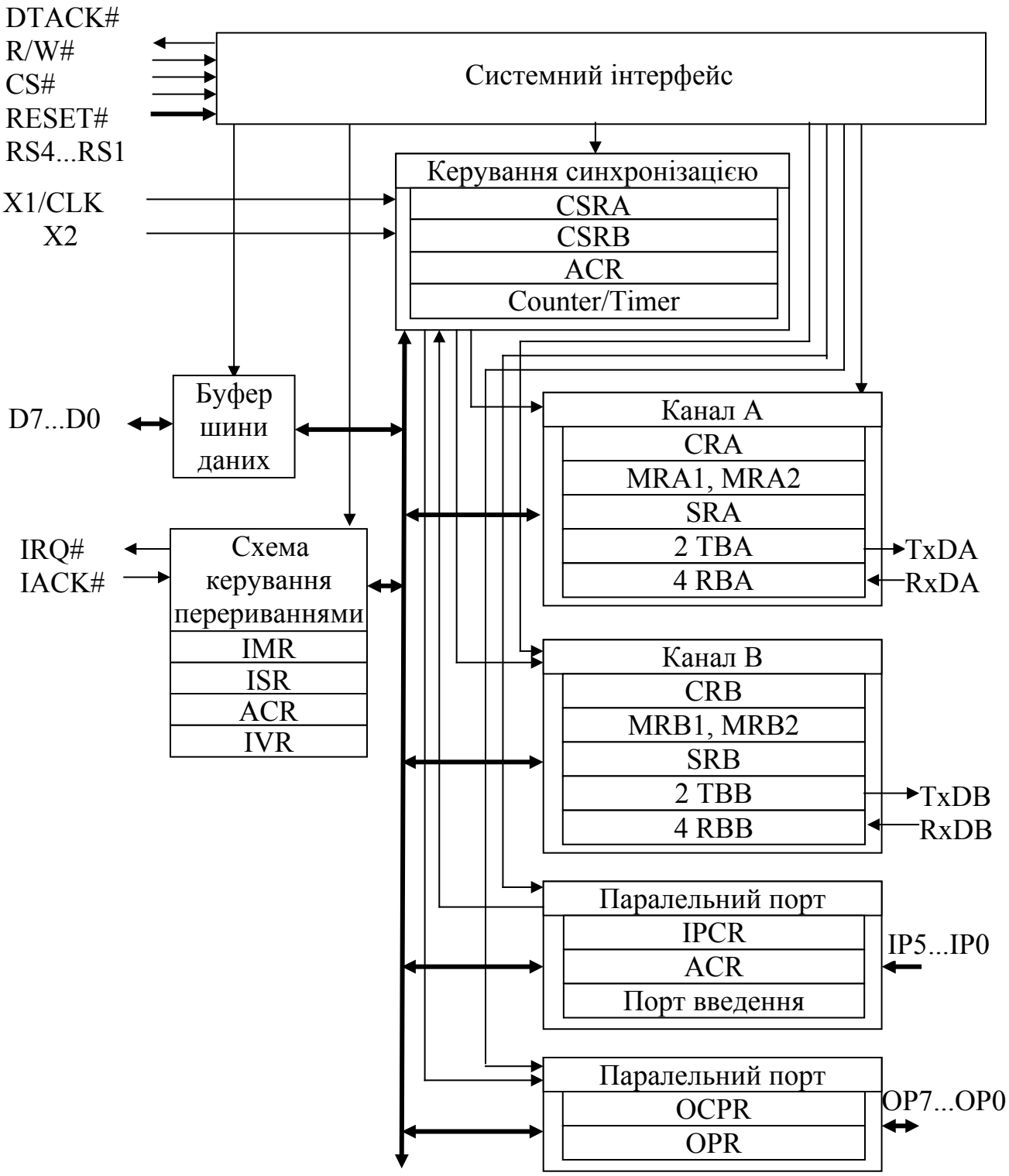


Рисунок 11.13 – Структурна схема DUART

– схема керування синхронізацією. Схема визначає швидкість обміну інформацією у послідовних каналах. До її складу входять:

CSRA (Clock Select Register channel A) – регістр визначення швидкості обміну інформацією в каналі A;

CSRB (Clock Select Register channel B) – регістр визначення швидкості обміну інформацією в каналі B;

ACR (Auxiliary Control Register) – допоміжний регістр контролю;

Counter/Timer – 16-розрядний лічильник/таймер, складається з двох 8-розрядних регістрів, кожен з яких програмується окремо;

– два аналогічних асинхронних послідовних канали – A і B, які побудовано з однакових вузлів:

CRA (B) (Command Register channel A (B)) – командний регістр, який керує виконанням команд;

MRA1, MRA2 (Mode Register 1, 2) – регістри, вміст котрих визначає режим роботи каналів введення-виведення даних;

SRA (B) (Status Register Channel A (B)) – регістри стану, зберігають інформацію про результати обміну;

TBA (B) (Transmit Buffer channel A (B)) – буферні регістри передавача відповідного каналу, до їхнього складу входять два регістри: THRA (B) – регістр тимчасового зберігання передавача і TSRA (B) – регістр зсуву передавача;

RBA (B) (Receive Buffer channel A (B)) – буферні регістри приймача відповідного каналу. Кожний складається з чотирьох регістрів: трьох регістрів RHRA (B) – регістр тимчасового зберігання приймача, і RSRA (B) – регістр зсуву приймача;

– 6-розрядний паралельний порт введення. За необхідністю виводи каналу може бути запрограмовано на виконання певних допоміжних функцій обслуговування каналів зв’язку. До його складу входять:

IPCR (Input Port Change Register) – регістр стану порту введення. За відсутності сигналів перебуває у скинутому стані;

ACR (Auxiliary Control Register) – допоміжний регістр контролю;

порт (6-розрядний регістр) введення;

– 8-розрядний паралельний порт введення-виведення. Багатоцільовий універсальний порт. Усі розряди порту може бути індивідуально скинуто або встановлено. До його складу входять:

OPCR (Output Port Configuration Register) – регістр стану. Програмується для визначення конфігурації паралельного порту (OP7...OP0) для поточного режиму роботи.

OPR (Output Port Register) – 8-розрядний паралельний порт.

Розподіл адресного простору DUART відбувається відповідно до табл. 11.6.

Кожен з двох каналів забезпечує роботу на 18-ти фіксованих швидкостях від внутрішнього генератора, частота котрого визначається кварцовим стабілізатором, який має частоту 3,6864 МГц, або від зовнішнього генератора, сигнал з котрого надходить на вивід X1/CLK. Взаємозв’язок швидкості й частот зовнішнього генератора наведено в табл. 11.7.

Таблиця 11.6 – Розподіл адресного простору DUART

RS4	RS3	RS2	RS1	Читання (R/W = 1)	Запис (R/W = 0)	
0	0	0	0	Mode Register A (MRA1, 2)	Mode Register A (MRA1, 2)	
0	0	0	1	Status Register A (SRA)	Clock Select Register (CSRA)	
0	0	1	0	Не використовується	Command Register A (CRA)	
0	0	1	1	Receive Buffer (RBA)	Transmit Buffer A (TBA)	
0	1	0	0	Input Port Change Register (IPCR)	Auxiliary Control Register (ACR)	
0	1	0	1	Interrupt Status Register (ISR)	Interrupt Mask Register (IMR)	
0	1	1	0	Старший байт лічильника/таймера	Старший байт лічильника/таймера	
0	1	1	1	Молодший байт лічильника/таймера	Молодший байт лічильника/таймера	
1	0	0	0	Mode Register B (MRB1,2)	Mode Register B (MRB1,2)	
1	0	0	1	Status Register B (SRB)	Clock Select Register (CSRB)	
1	0	1	0	Не використовується	(Command Register B (CRB)	
1	0	1	1	Receive Buffer B (RBB)	Transmit Buffer B (TBA)	
1	1	0	0	Interrupt Vector Register (IVR)	Interrupt Vector Register (IVR)	
1	1	0	1	Порт введення	Output Port Configuration Register (OPCR)	
1	1	1	0	Старт лічби	Output Port Register	Встановлення біта
1	1	1	1	Стоп лічби		Скидання біта

Таблиця 11.7 – Відповідність швидкості й частоти зовнішнього генератора

Швидкість, Бод	Частота зовнішнього генератора, кГц	Швидкість, Бод	Частота зовнішнього генератора, кГц
50	0,8	1200	19,2
75	1,2	1800	28,8
100	1,759	2000	32,056
134,5	2,153	2400	38,4
150	2,4	4800	76,8
200	3,2	7200	115,2
300	4,8	9600	153,6
600	9,6	19,2 кБод	307,2
1050	16,756	38,4 кБод	614,4

Програмування ВІС DUART здійснюється за допомогою запису сигналів керування (байтів) до відповідних регістрів.

Керування роботою каналів А і В здійснюється за допомогою вмісту регістрів MRA1, MRA2 (MRB1, MRB2). Призначення бітів регістру MRA1 (MRB1) подано в табл. 11.8.

Програмування регістрів MRA2 та MRB2 задля визначення режиму роботи каналу здійснюється відповідно до табл. 11.9.

Вибір необхідної швидкості роботи передавача та приймача здійснюється завантаженням до регістрів CSRA та CSRB байта, вміст котрого формується відповідно до табл. 11.10. За коду 1111 для синхронізації використовуються виводи IP3 або IP4 паралельного порту введення.

Вміст регістрів команд CRA (B) керує виконанням програми функціонування BIC DUART. Його програмування наведено у табл. 11.11.

Передавачі каналів А і В програмуються відповідно до наведених таблиць перед початком передавання інформації. Про готовність до передавання повідомляється встановленням біта 2 у регістрі статусу (SRA (B)). Цей біт також може використовуватися для генерації запиту на переривання для каналу А на виводі OP6 паралельного порту або OP7 – для каналу В. Передавання дозволяється встановленням біта 2 і скиданням біта 3 у командному регістрі CRA (B).

Передавання інформації розпочинається з її завантаженням до буферного регістру передавача TBA (B), звідки вони передаються до регістру зсуву TSRA (B). В цьому регістрі здійснюється їхнє перетворення на послідовний код, формування стартового біта і передавання інформації до каналу. Завершується передавання формуванням необов’язкового біта парності та обов’язкових одного чи двох стопових бітів. Передача розпочинається за зрізом сигналу у каналі TxD, що встановлює значення сигналу TxRDY у 1. Скидання цього сигналу буде відбуватися після завантаження до буфера передавання нової інформації, що призводить до формування імпульсу сигналу DTACK, котрий дозволяє передавання інформації. Після проходження стопового біта лінія залишається у стані логічної 1, якщо немає наступного байта. Скидання бітів 2 і 3 командного регістру CRA (B) є забороною передавання, при цьому сигнали TxRDY і DTACK не формуються і інформація D4 не передається.

Передавання інформації подано за допомогою часових діаграм, які наведено на рис. 11.14.

Про готовність до приймання інформації повідомляється встановленням біта 0 в регістрі статусу (SRA (B)). Приймачі відповідних каналів розпочинають працювати при надходженні на вхід зрізу вхідного сигналу і визначають тривалість одного біта. Тривалість біта визначається кількістю імпульсів тактової частоти від зрізу до першого фронту вхідного сигналу. Зазвичай тривалість біта становить 7,5 періодів тактової частоти; якщо таке не виконується, подальша інформація ігнорується і приймач продовжує

Таблиця 11.8 – Програмування режиму роботи каналів регістрів MRA1 (MRB1)

Контроль RxRTS	Вибір RxIRQ#	Помилка	Перевірка парності		Тип перевірки	Довжина повідомлення	
біт 7	біт 6	біт 5	біт 4	біт 3	біт 2	біт 1	біт 0
0 – не дозволено 1 – дозволено	0 – RxRDY 1 – повний канал FIFO	0 – обнулення 1 – блокування	0 0	Перевірка парності	З перевіркою 0 – дорівнює 1 – не дорівнює – Адреси/дані 0 – дані 1– адреси	0 0 – 5 біт 0 1 – 6 біт 1 0 – 7 біт 1 1 – 8 біт	
			0 1	–			
			1 0	Немає перевірки			
			1 1	Біти використовуються як адреси/дані			

Таблиця 11.9 – Програмування режиму роботи каналів регістрів MRA2 (MRB2)

Режим роботи каналу		Контроль TxRTS (вивід OP0)	CTS дозвіл передавання	Кількість стопових бітів	
біт 7	біт 6	біт 5	біт 4	біт 3 біт 2 біт 1 біт 0	
		0 – не дозволено 1 – дозволено	0 – не дозволено 1 – дозволено	6-8 бітів даних	5 бітів даних
0 0	Нормальний			0000 – 0,563	1,063
0 1	Автоматичного еха-сигналу			0001 – 0,625	1,125
1 0	Місцева кільцева перевірка			0010 – 0,688	1,188
1 1	Віддалена кільцева перевірка			0011 – 0,750	1,250
				0100 – 0,813	1,313
				0101 – 0,875	1,375
				0110 – 0,938	1,438
				0111 – 1,000	1,500
				1000 – 1,563	1,563
				1001 – 1,625	1,625
				1010 – 1,688	1,688
				1011 – 1,750	1,750
				1100 – 1,813	1,813
				1101 – 1,875	1, 875
				1110 – 1,938	1,938
				1111 – 2,000	2,000

Таблиця 11.10 – Програмування швидкості обміну DUART

Вибір швидкості приймача						Вибір швидкості передавача					
Біт 7	Біт 6	Біт 5	Біт 4	Швидкість, Бод залежно від ACR		Біт 3	Біт 2	Біт 1	Біт 0	Швидкість, Бод залежно від ACR	
				Біт 7 = 0	Біт 7 = 1					Біт 7 = 0	Біт 7 = 1
0	0	0	0	50	75	0	0	0	0	50	75
0	0	0	1	110	110	0	0	0	1	110	110
0	0	1	0	134,5	134,5	0	0	1	0	134,5	134,5
0	0	1	1	150	150	0	0	1	1	150	150
0	1	0	0	200	200	0	1	0	0	200	200
0	1	0	1	300	300	0	1	0	1	300	300
0	1	1	0	600	600	0	1	1	0	600	600
0	1	1	1	1050	1050	0	1	1	1	1050	1050
1	0	0	0	2400	2400	1	0	0	0	2400	2400
1	0	0	1	4800	4800	1	0	0	1	4800	4800
1	0	1	0	7200	1800	1	0	1	0	7200	1800
1	0	1	1	19,2 кБод	9600	1	0	1	1	19,2 кБод	9600
1	1	0	0	38,4 кБод	19,2 кБод	1	1	0	0	38,4 кБод	19,2 кБод
1	1	0	1	Таймер	Таймер	1	1	0	1	Таймер	Таймер
1	1	1	0	Зовн. сигн.	Зовн. сигн.	1	1	1	0	Зовн. сигн.	Зовн. сигн.
1	1	1	1	Зовн. сигн.	Зовн. сигн.	1	1	1	1	Зовн. сигн.	Зовн. сигн.

Таблиця 11.11 – Програмування командного регістра CRA(B)

	Різні команди	Команди передавача	Команди приймача
Біт 7	Біт 6 Біт 5 Біт 4	Біт 3 Біт 2	Біт 1 Біт 0
	000 – Команди немає 001 – Скидання режиму 010 – Скидання приймача 011 – Скидання передавача 100 – Скидання статусу помилок 101 – Скидання канальних переривань 110 – Зупин 111 – Скидання зупину	00 – Не виконуються активні дії. Зберігається поточний стан режимів	00 – Не виконуються активні дії. Зберігається поточний стан режимів
		01 – Передавання дозволено	01 – Приймання дозволено
		10 – Передавання заборонено	10 – Приймання заборонено
		11 – Не використовується	11 – Не використовується

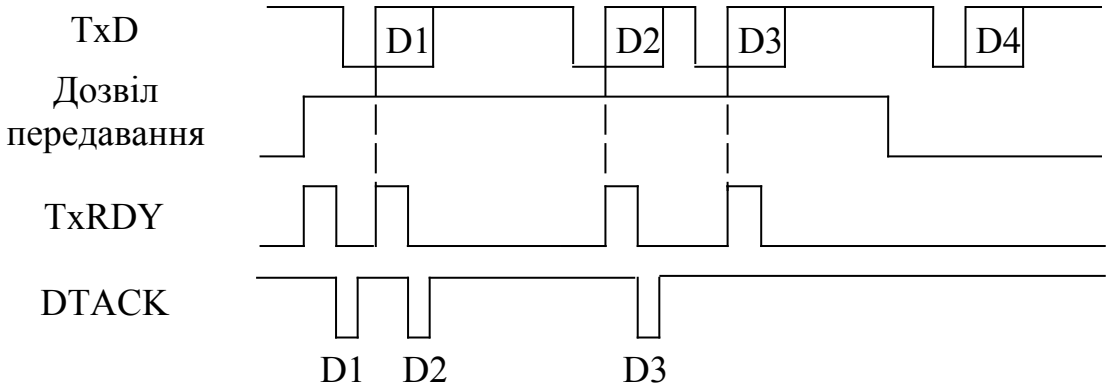


Рисунок 11.14 – Передавання інформації послідовним каналом

контролювати надходження імпульсів на його вхід. Якщо часові співвідношення виконуються, то приймач записує дані до буфера приймача RHR і скидає біт 0 у регістрі статусу (SRA (B)). Якщо віднайдено помилку кадрової синхронізації (відсутність стопового біта), то у лінії зберігається нульовий (L0) стан впродовж половини періоду тривалості біта, після чого робота продовжується так само, як і після отримання стартового біта.

Поява інших помилок (парності, зміна швидкості надходження інформації тощо) змінює значення відповідних прапорців (ознак) у регістрах статусу (SRA (B)).

Після завершення приймання байта на лінії повинен встановлюватися високій рівень сигналу (L1) тривалістю не менш за половину періоду тривалості біта, після чого порт продовжує пошук наступного стартового біта.

Функціонування DUART у режимах контролю забезпечується встановленням бітів 7, 6 у регістрі режиму роботи каналів MRA2 (MRB2) відповідно до табл. 11.8.

В режимі автоматичного eха-сигналу (код бітів 7, 6, ..., 1) в каналі автоматично ретранслюються дані. Місцевий зв’язок поміж центральним процесором і приймачем продовжує нормально функціонувати, а зв’язок поміж центральним процесором і передавачем заблоковано. В такий спосіб, можна контролювати стан лінії і роботу приймача станції.

В режимі місцевої кільцевої перевірки (код – 10) вихід передавача у BIC з’єднується з входом приймача. Передавач і приймач в цьому режимі працюють відповідно до встановлених режимів у штатному розписі. Передаючи дані і контролюючи їхнє приймання, можна перевірити правильність роботи місцевого каналу приймання-передавання DUART.

В режимі віддаленої кільцевої перевірки каналом автоматично ретранслюються дані і канал передавача головної станції сполучено з іншими станціями мережі (дозволено до 256 інших станцій). При цьому місцевий зв’язок поміж центральним процесором і приймачем заблоковано. До складу даних, що передаються, входять стартовий біт, блок даних, біт адреси/даних

(A/D) і стопові біти. Цей блок призначено одній зі станцій. Станції мережі невпинно здійснюють моніторинг каналу й ідентифікують біт адреси/даних задля визначення виду інформації. Якщо біт A/D встановлено, то передається адреса, якщо скинуто – дані. Станція визначає власну адресу, сповіщає про це головну станцію, яка вмикає власний приймач, і генерує переривання, після чого отримується повідомлення, і знову заблоковує власний канал.

Багатофункціональний 16-розрядний лічильник/таймер може працювати в режимі формування часових інтервалів і режимі генерування імпульсів. В кожному з режимів лічильник/таймер можна запрограмувати на отримання сигналу тактової частоти від кількох джерел і виведення результату через вивід OP3 паралельного порту. До лічильника можна завантажити число від $0002 до $FFFF, котре можна змінити будь-якого моменту. Програмування здійснюється завантаженням до допоміжного регістра керування (ACR) байта, що відповідає потрібному режимові роботи. Формат цього регістра наведено у табл. 11.12. Біт 7 цього регістра визначає швидкість обміну інформацією відповідно до табл. 11.6. В режимі формування часових інтервалів лічильник/таймер запускається і зупиняється ЦП, тому цей режим можна використовувати як системний годинник реального часу задля генерування переривань і як вартовий пристрій. В режимі генерування імпульсів лічильник/таймер працює невпинно незалежно від ЦП, тому такий режим можна використовувати для програмної реалізації сигналів тактової частоти для каналів A і B, періодичного звернення до певних пристроїв і в якості автогенератора періодичних імпульсних сигналів.

В режимі формування часових інтервалів лічильник/таймер працює як лічильник, який декрементує попередньо завантажений вміст лічильника. За сигнал тактової частоти може слугувати сигнал від зовнішнього генератора, який надходить на вхід X1/CLK, поділений на 16, або сигнал зі входу IP2. Після програмування і запуску лічильник/таймер розпочинає працювати від значень попереднього завантаження до досягнення значення $0000, після чого встановлює прапорець у біті 3 регістру ISR і продовжує лічбу від значення $FFFF. Для перевантаження лічильник/таймер потрібно заново зініціалізувати.

В режимі генерування імпульсів таймер формує періодичну послідовність прямокутних імпульсів, частота слідування котрих визначається значеннями попередніх завантажень і сигналом, який надходить від джерел: внутрішній кварцовий генератор; зовнішній сигнал на виводі X1/CLK або сигнал на цьому виводі, частота котрого поділена на 16; зовнішній сигнал, що надходить на вивід IP32. Таймер виконує лічбу до досягнення значення $0000, після чого перевантажує значення попередніх завантажень і продовжує роботу.

Таблиця 11.12 – Програмування режиму роботи таймера

Вибір швид-кості	Програмування режиму і джерела сигналу		Виводи запитів переривання			
			IP3 IRQ	IP2 IRQ	IP1 IRQ	IP0 IRQ
Біт 7	Біт 6 | Біт 5 | Біт 4		Біт 3	Біт 2	Біт 1	Біт 0
	Режим	Джерело часових сигналів	0 – дозвіл, 1 – заборона	0 – дозвіл, 1 – заборона	0 – дозвіл, 1 – заборона	0 – дозвіл, 1 – заборона
0 або1 (див. табл. 11.5	000 – лічильник	зовнішній IP2				
	001 – лічильник	TxCA – 1X визначення часових інтервалів каналу передавача A				
	010 – лічильник	TxCB – 1X визначення часових інтервалів каналу передавача B				
	011 – лічильник	кварцовий резонатор або зовнішній сигнал (X1/CLK), поділений на 16				
	100 – таймер	зовнішній IP2				
	101 – таймер	зовнішній IP2				
	110 – таймер	кварцовий резонатор або зовнішній сигнал (X1/CLK)				
	111 – таймер	кварцовий резонатор або зовнішній сигнал (X1/CLK), поділений на 16				

Нижче наведено фрагмент програми роботи послідовного порту DUART задля передавання байта зі швидкістю 300 Бод каналом A:

```
MOVEA.L #$B00000,A0     ; Завантаження базової адреси DUART у A0
MOVE.B #$07,(A0)        ; Завантаження до регістра режиму порту A
                        ; MR2A даного $07 (00000111) для
                        ; встановлення нормального режиму
                        ; передавання з одним стоповим бітом
MOVE.B #$44,($01,A0)    ; Завантаження до регістра визначання
                        ; швидкості обміну інформацією у каналі A
                        ; даного $44 (01000100) для  забезпечення
                        ; швидкості 300 Бод
MOVE.B #$34,($02,A0)    ; Завантаження до регістра команд каналу A
                        ; даного $34 (01010101), яке перевантажує
                        ; передавач
```

```
MOVE.B #$A5,($03,A0)      ; Завантаження до буферного регістра
                          ; передавача каналу А даного $A5 (10100101),
                          ; яке є інформацією, котру буде передано
                          ; каналом А
MOVE.B ($03,A0),($0E,A0)  ; Передавання інформації каналом А
```

Підключення ВІС DUART до шин МПС проводиться відповідно до функціонального призначення виводів мікросхеми та сигналів, що формуються іншими пристроями. Умовні графічні позначення та схему підключення асинхронного послідовного порту DUART подано на рис. 11.15. Збільшення кількості послідовних портів здійснюється при дешифруванні сигналів адреси. Кожна з ВІС DUART займає адресний простір 32 байти. Отже, для адресування наступних ВІС можна використовувати розряди А5 і наступні.

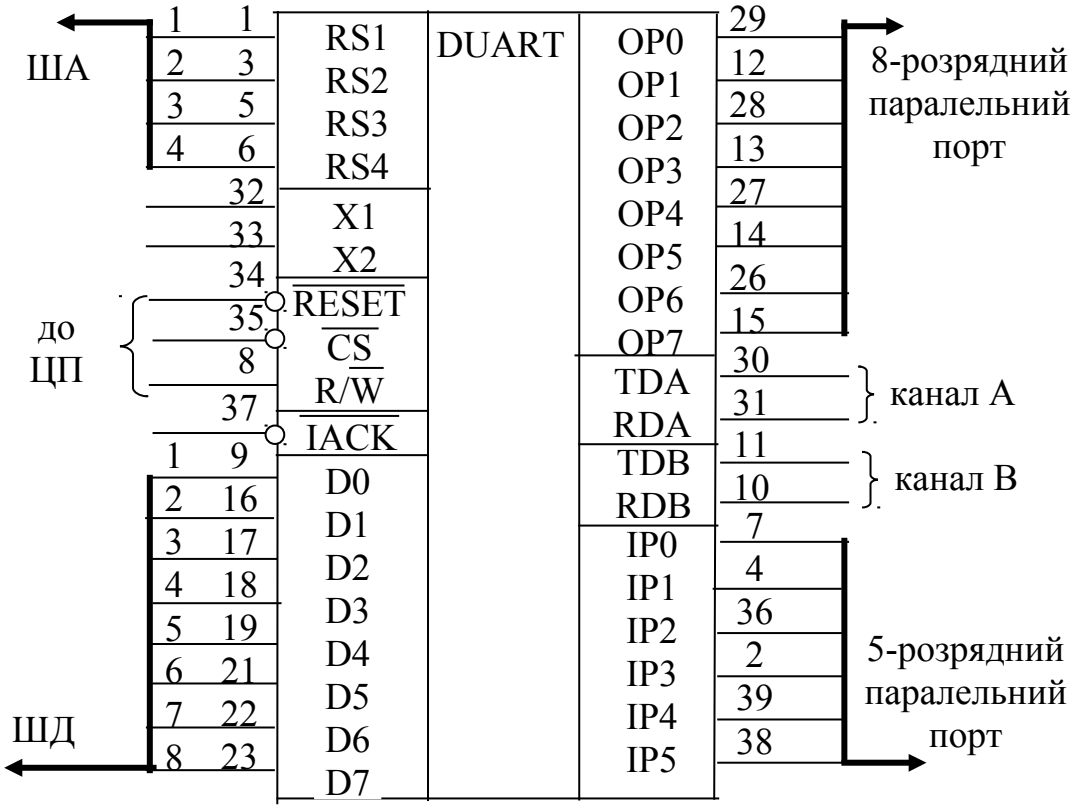


Рисунок 11.15 – Схема підключення ВІС DUART

Контрольні запитання:

1 Які функціональні можливості має ВІС DUART?

2 Які пристрої входять до складу ВІС DUART?

3 Який обсяг пам’яті відведено для адресування блоків ВІС DUART?

4 У яких режимах може працювати блок приймача-передавача ВІС DUART?

5 Скільки дуплексних каналів можливо зорганізувати за допомогою однієї ВІС DUART?

6 Для чого можуть використовуватися виводи порту IP0...IP5?

Контрольні запитання підвищеної складності:

1 Для чого призначено режим автоматичного ехо-сигналу?
2 У яких режимах може працювати 16-розрядний лічильник/таймер?
3 У який спосіб програмується швидкість обміну інформацією ВІС DUART?
4 Чи є можливість перевірки працездатності тракту приймача-передавача без наявності абонента?
5 Скільки окремих швидкостей обміну можна зреалізовувати за допомогою ВІС DUART?

Периферійний інтерфейс/таймер (PIT) MC68230 фірми Motorola

ВІС PI/T є паралельний інтерфейс/таймер, призначений для обміну 8-розрядними даними поміж мікропроцесором та зовнішніми пристроями (датчиками, пристроями керування тощо). Структурну схему PI/T наведено на рис. 11.16.

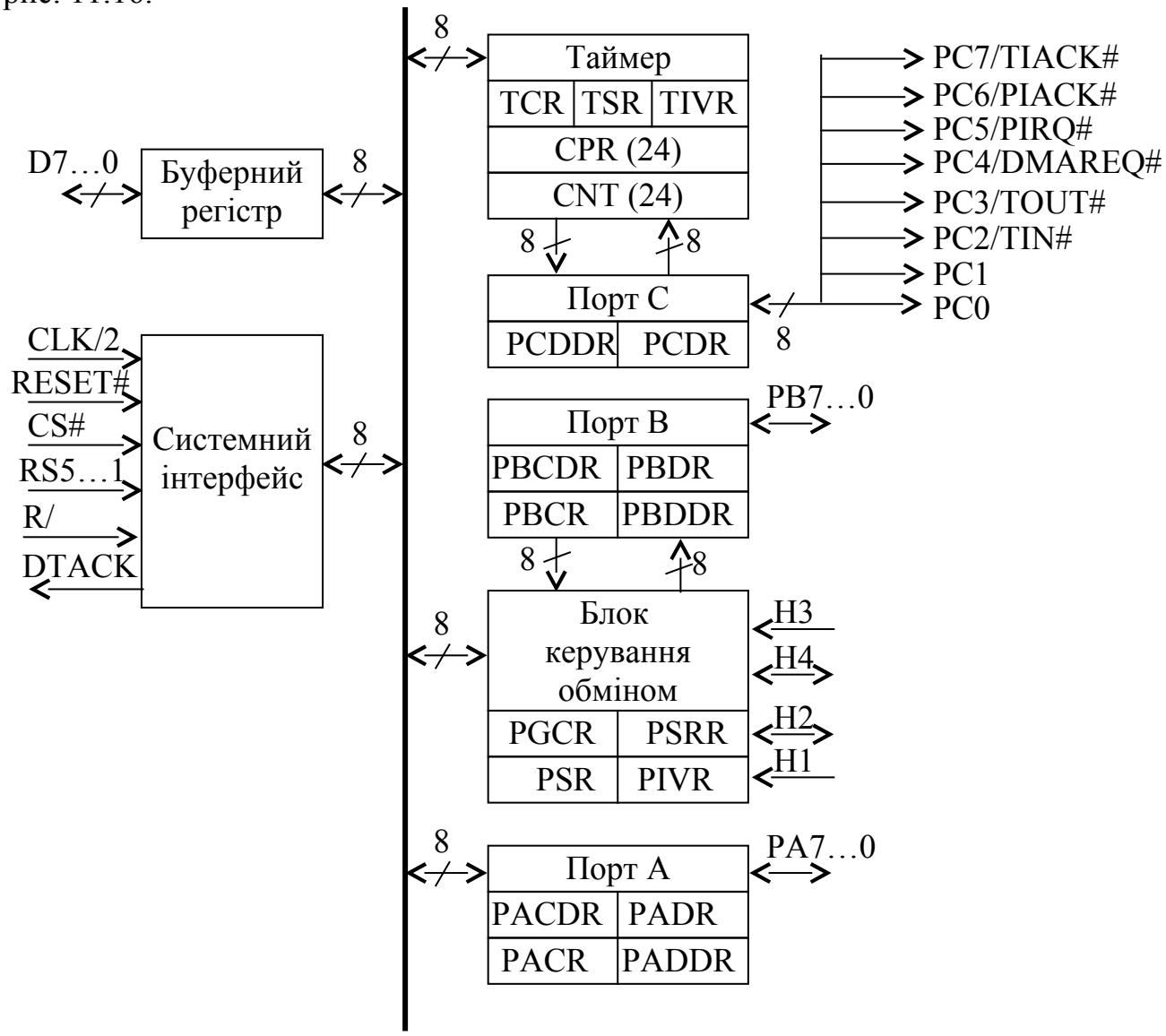


Рисунок 11.16 – Структурна схема PI/T

PI/T складається з блоків, які забезпечують зв'язок з мікропроцесором (буферний регістр, системний інтерфейс), і блоків, які обслуговують зовнішні пристрої (8-розрядні порти A, B, C, 24-розрядний таймер, блок керування обміном, який може використовуватись для реалізації переривань, паралельного введення-виведення даних або формування сигналів квитування при пересиланні даних через порти A, B). Виводи PC7...PC2 можуть програмуватись для передавання сигналів таймера TIN#, TOUT#, переривання PIRQ#, PIACK#, TIACK, запиту прямого доступу DMAREQ#. Зв'язок PI/T з МП зреалізовується шляхом обміну даними по виводах D7…D0 у циклі читання або запису. МП видає на вхід системного інтерфейсу сигнал $R/\overline{W}$, а PI/T видає сигнал підтвердження готовності DTACK#.

Дані зчитуються або записуються до одного з регістрів таймера, порту A, B, C або блока керування обміном. Вибір регістра визначається кодом адреси, який надходить на входи адреси RS5…RS1. В адресному просторі PI/T займає 32-байтові комірки, з яких 23 зайнято регістрами, а решта не використовуються. Всі регістри, окрім лічильника CNT та регістра попереднього встановлення лічильника CPR, які мають 24 розряди, є 8-розрядні і адресуються як байт. CNT та CPR адресуються як три окремих 8-розрядні регістри. У табл. 11.13 показано адреси регістрів PI/T та зазначено зміщення відносно базової адреси $800001, які використовуються у команді MOVEP для формування ефективної адреси, регістрів PI/T.

При зверненні до PI/T на входи RS5...RS1 надходить адреса регістра, що її формує МП, а на вхід вибирання CS# подається сигнал CS# від адресного дешифратора, який визначає саме цей PI/T серед усіх, які входять до МПС. На вхід CLK надходять синхросигнали CLK/2, а на вхід RESET# – загальний для всієї системи сигнал скидання.

PI/T є програмована ВІС, у керувальні регістри якої для реалізації різних режимів роботи треба записати керувальні коди, тобто зініціалізувати її.

Порти A та B забезпечують паралельний обмін даними поміж МП та зовнішніми пристроями. Задля виведення даних МП записує їх до регістра даних відповідного порту: PADR або PBDR, а їхнє введення здійснюється шляхом їхнього зчитування з регістрів даних. Кожний вивід портів A та B може програмуватись окремо на введення або виведення встановленням у 1 (виведення) або у 0 (введення) бітів відповідного регістра – PADDR або PBDDR. Порти мають також додаткові регістри даних – PACDR та PBCDR, які дозволяють зберігати дані, якщо необхідно ввести від зовнішнього пристрою наступний біт до того, як попередній було зчитано мікропроцесором, або вивести наступний біт з мікропроцесора до того, як попередній біт було прийнято зовнішнім пристроєм. Отже, PACDR та PBCDR слугують за буферні регістри.

Блок керування обміном здійснює загальне керування портами A та B, а також керування функціями виводів H3...H1. При ініціалізації портів A та B до регістрів PGCR, PSRR блока керування, регістрів PACR, PADDR та PBCR, PBDDR портів МП записує керувальні коди, які визначають їхнє функціонування.

Таблиця 11.13 – Адреси регістрів PI/T

Назва регістра	Призначення регістра	Адреса	Зміщення
PGCR	Головний регістр керування	$800001	$00
PSRR	Регістр обслуговування переривань	$800003	$02
PADDR	Регістр напрямку обміну даними порту A	$800005	$04
PBDDR	Регістр напрямку обміну даними порту B	$800007	$06
PCDDR	Регістр напрямку обміну даними порту C	$800009	$08
PIVR	Регістр вектора переривань	$80000B	$0A
PACR	Регістр керування порту A	$80000D	$0C
PBCR	Регістр керування порту B	$80000F	$0E
PADR	Регістр даних порту A	$8000011	$10
PBDR	Регістр даних порту B	$8000013	$12
PAAR	Додатковий регістр даних порту A	$8000015	$14
PBAR	Додатковий регістр даних порту B	$8000017	$16
PCDR	Регістр даних порту C	$8000019	$18
PSR	Регістр стану PI/T	$800001B	$1A
TCR	Регістр керування таймером	$8000021	$20
TIVR	Регістр вектора переривань таймера	$8000023	$22
CPRH	Регістр попереднього встановлення старшого байта лічильника	$8000027	$26
CPRM	Регістр попереднього встановлення середнього байта лічильника	$8000029	$28
CPRL	Регістр попереднього встановлення молодшого байта лічильника	$800002B	$2A
CNTRH	Регістр старшого байта лічильника	$800002F	$2E
CNTRM	Регістр середнього байта лічильника	$8000031	$30
CNTRL	Регістр молодшого байта лічильника	$8000033	$32
TSR	Регістр стану таймера	$8000035	$34

Фрагмент програми демонструє керування виведенням даних PI/T:

```
MOVEA.L #$800001,A0     ; Завантаження базової адреси PI/T до A0
MOVE.B #0,($0C,A0)      ; Завантаження до регістра керування порту A PACR
                        ; 0 для встановлення нульового режиму
MOVE.B #$FF,(4,A0)      ; Завантаження до регістра напрямку обміну даними
                        ; порту A даного $FF для забезпечення виведення по
                        ; всіх розрядах
MOVE.B #$55,($0,A0)     ; Завантаження до регістра даних порту A даного
                        ; $55 (01010101)
```

```
JSR.B *+12              ; Звернення до підпрограми затримки
BRA.B *-16              ; Циклування програми
MOVE.L #1000000,D0      ; Завантаження лічильника підпрограми затримки
SUBQ.L #1,D0            ; Декрементування лічильника
BNE.B *-2               ; Організація підпрограми затримки на 1 с
RTS                     ; Повернення з підпрограми затримки
```

Наведений фрагмент програми є драйвер виведення PI/T.

Фрагмент драйвера введення наведено нижче:

```
MOVE.B $800015.L,D0     ; Введення молодшого байта даного з додаткового
                        ; регістра порту A PADR до регістра D0
```

За бажання ввести до регістра D0 слова з портів A та B можна застосовувати команду MOVEP, яка виконується тільки з непрямою регістровою адресацією:

```
MOVEA.L $800001,A5
.
.
.
MOVEP ($14,A5),D0
```

За командою MOVEP біти D15...D8 з додаткового регістра порту A пересилаються до першого байта регістра D0, а біти D7...D0 пересилаються з додаткового регістра порту B до нульового байта регістра D0.

Виводи H3...H1 може бути запрограмовано на виконання функцій входів на запит переривання, входів-виходів даних або передавання сигналів квитування при обміні даними через порти A, B. Якщо виводи H3 та H1 використовуються для введення даних, то поточні сигнали на цих виводах дублюються у чотирьох розрядах регістра стану PSR. Для виведення даних використовуються виводи H2 та H4, значення даних на них встановлюються відповідно до вмісту певних розрядів регістра PSR. Через читання або запис вмісту PSR мікропроцесор керує введенням або виведенням даних.

Порти A, B можуть програмуватись для реалізації чотирьох різних режимів обміну.

Режим 0 – односпрямований обмін 8-розрядними даними через окремі порти A і B. Напрямок обміну для кожного виводу задається вмістом регістрів PADDR та PBDDR. Виводи H1 і H2 можуть використовуватись для квитування передавання через порт A, а виводи H3 і H4 – для квитування передавання через порт B. При введенні даних до відповідного порту сигнали H1 і H3 вказують на надходження даних від зовнішнього пристрою, при виведенні даних – на їхнє приймання зовнішнім пристроєм. Сигнали H2 і H4 при введенні

підтверджують отримання даних від зовнішніх пристроїв, а при виведенні слугують за запити на приймання даних зовнішніми пристроями.

Режим 1 – односпрямований обмін 16-розрядними даними через подвійний порт A-B. Для квитування обміну використовуються сигнали на виводах H3, H4, функціональне призначення яких при введенні та виведені даних є таке ж саме, як і в режимі 0. Виводи H1 та H2 використовуються для введення-виведення даних або для подавання запитів на переривання.

Режим 2 – двоспрямований обмін 8-розрядними даними через порт B. Виводи H1 та H2 слугують для квитування виведення даних до зовнішнього пристрою, виводи H3 та H4 – для квитування введення даних до порту B. Порт A у цьому режимі використовується для односпрямованого обміну даними без квитування.

Режим 3 – двоспрямований обмін 16-розрядними даними, які зчитуються мікропроцесором через подвійний порт A-B. Виводи H1 та H2 слугують для квитування виведення даних до зовнішнього пристрою, а виводи H3 та H4 – для квитування введення даних до порту.

Порт C може використовуватись для обміну 8-розрядними даними, які записуються або зчитуються мікропроцесором через регістр PCDR. Обмін використовується без квитування. Напрямок передавання даних для кожного виводу PCi задається відповідним розрядом вмісту регістра PCDDR, яке вводиться до нього у перебігу ініціалізації порту C. Виводи PC2, PC3, PC7 може бути запрограмовано для обслуговування таймера, виводи PC5, PC6 – для запиту на переривання та приймання підтвердження переривання від мікропроцесора. Вивід PC4 може використовуватись для формування запиту на прямий доступ до пам’яті.

Якщо виводи H3...H1 або частина з них програмується для роботи на приймання запитів на переривання за ініціалізації блока керування обміном, то надходження на них запиту зумовлює формування сигналу PIRQ# = 0 на виводі PC5 порту C. Цей сигнал подається на вхід приорітетного шифратора PRCD, який формує відповідний код запиту на переривання IPL2...IPL0, який подається на мікропроцесор. Коли мікропроцесор переходить до обслуговування цього запиту, він видає сигнал підтвердження, який надходить на вхід PIACK# – вивід PC6. Мікропроцесор формує адресу регістра векторів переривань PIVR і зчитує вектор при зверненні до таблиці переривань для адресування підпрограми обслуговування цього переривання. Кожному запитові Hi відповідає свій вектор переривань. Старші шість розрядів цього вектора записуються до регістра PIVR PI/T у перебігу ініціалізації, а два молодших розряди відповідають номерові входу Hi, на який надійшов запит. Пріоритети запитів від зовнішніх пристроїв і порядок їхнього обслуговування визначаються вмістом регістра PSSR PI/T, який завантажується у перебігу ініціалізації.

Таймер, який входить до складу PI/T, зреалізовано на базі 24-розрядного лічильника CNT, який працює на віднімання. Початковий стан лічильника встановлюється за ініціалізації через запис 3-х байтів до регістра попереднього встановлення CPR. Запуск таймера здійснюється через запис до регістра

керування таймером TCR керувального коду, який визначає також режим його функціонування. Сигнали на CNT можуть надходити як від генератора синхроімпульсів CLK, так і від зовнішніх пристроїв на вхід TIN# (вивід PC2).

У режимі лічби імпульсів CLK таймер через інтервали часу, які визначаються вмістом CPR, формують сигнали на виході TOUT# (вивід PC3). У режимі лічби подій (лічба сигналів на вході TIN#) поточний вміст CNT вказує на кількість імпульсів, що надійшли. Також можна запрограмувати ділення частоти вхідних імпульсів на 32.

У режимі лічби синхроімпульсів сигнал на вході TIN# слугує для запуску (TIN# = 1) або зупину (TIN# = 0) лічильника. Сигнал на виході таймера TOUT# (вивід PC3) має початкове значення TOUT# = 1 і змінює своє значення, коли вміст лічильника дорівнює 0. Після цього лічильник може відновити початкове значення і повторити лічбу або завантажити нове значення з регістра CPR. Коли стан лічильника стає нульовим, у регістрі TSR встановлюється значення біта ZDS = 1 (прапорець нуля). Поточний стан таймера контролюється мікропроцесором через зчитування та аналіз вмісту регістра TSR.

При роботі таймера на виході TOUT# формуються прямокутні імпульси, які можуть використовуватись для керування зовнішніми пристроями (періодичне включення-виключення, синхронізація тощо). Сигнал TOUT може подаватися на вхід приорітетного шифратора як сигнал запиту для переривання для мікропроцесора. Сигнал підтвердження переривання надходить на вхід TIACK (вивід PC7). Для зчитування вектора переривання мікропроцесор звертається до регістра TIVR, до якого цей вектор записується за ініціалізації таймера.

Таймери використовуються у вартових пристроях, для затримки сигналів, для запуску підпрограм у задані моменти часу тощо.

На рис. 11.17 подано часові діаграми передавання слова через 8-розрядний інтерфейс PI/T. Сигнали SIZ1, SIZ0 мають відповідно значення 10...2 байти в перебігу першого циклу і 1 байт (значення 01) – в перебігу другого циклу. Сигнали DSACK1# та DSACK0, які вказують на готовність порту до роботи в кожному циклі, мають значення 11 в перебігу чекання і 10 (1 байт) – у кожному з циклів передавання даних до PI/T.

Контрольні запитання:

1 Які функції можна зреалізовувати на BIC PI/T?
2 Які пристрої входять до складу BIC PI/T?
3 Скільки регістрів може бути адресовано до BIC PI/T?
4 У скількох режимах роботи може працювати BIC PI/T і чим вони відрізнюються один від одного?
5 Поясніть термін – обмін інформацією з квитуванням.
6 Для чого використовуються виводи H1...H4 в різних режимах роботи?
7 В яких режимах може працювати таймер BIC PI/T?
8 Які виводи BIC PI/T використовує таймер?

Контрольні запитання підвищеної складності:

1 Чи можна використовувати ВІС PI/T для обміну словами?

2 Проілюструйте за допомогою часових діаграм процес передавання слова.

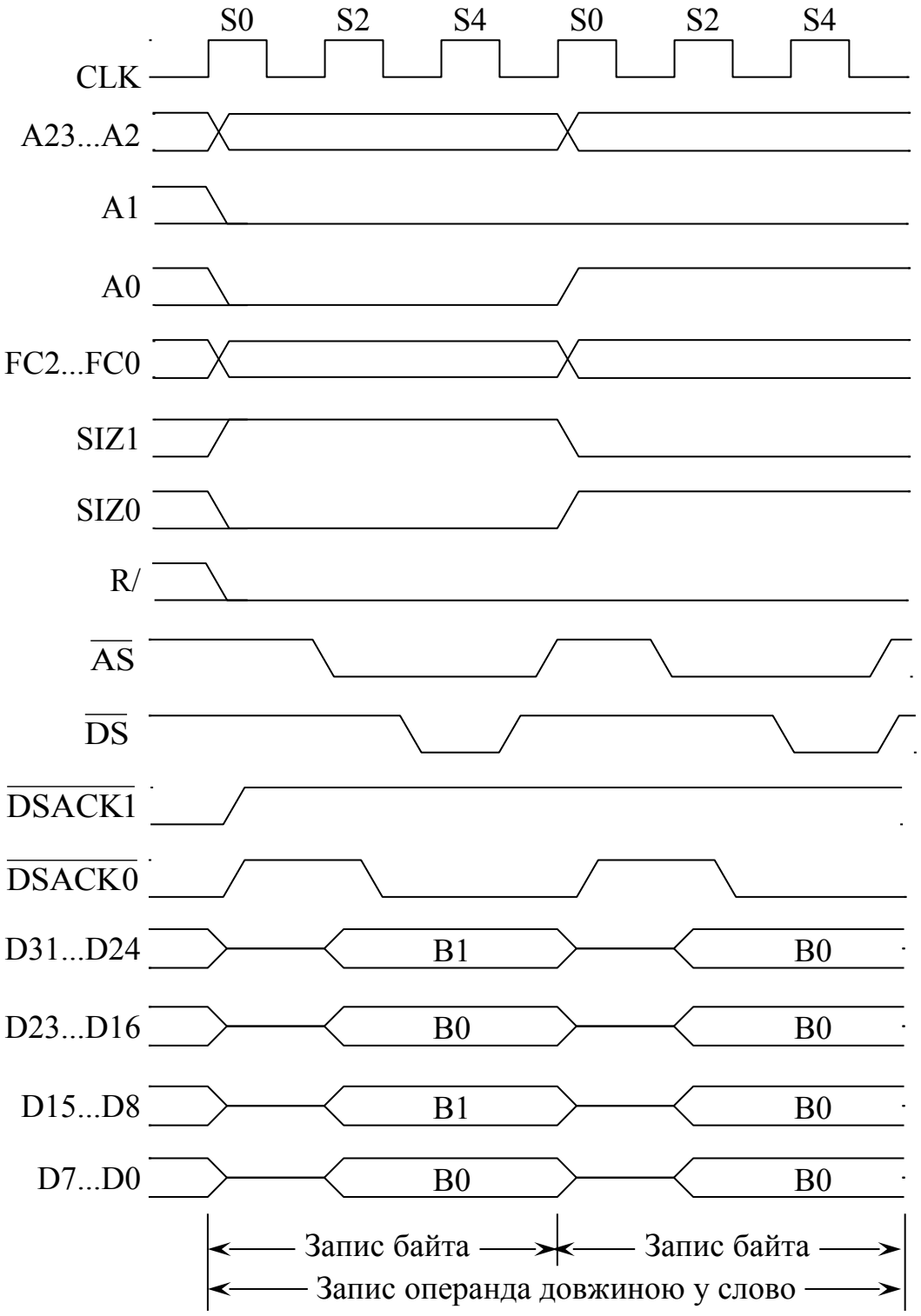


Рисунок 11.17 – Часові діаграми передавання слова через 8-розрядний інтерфейс PI/T

11.3 32-розрядні мікропроцесори сімейства M680XX фірми Motorola

Вхідний контроль:

1 Яку назву мають регістри загального призначення МП фірми Intel?

2 Чи є згадані у п. 1 регістри закріплено за умовчанням за певними функціями?

3 Що таке сегментування пам'яті і в яких МП воно використовується?

4 З якою метою пам'ять поділяється на сегменти?

5 Чи є сумісні різні моделі універсальних МП фірми Intel і, якщо так, то як цього досягається?

6 Розпочинаючи з якої моделі МП фірми Intel використовують конвеєр команд?

7 Розпочинаючи з якої моделі МП фірми Intel використовують кеш-пам'ять?

32-розрядні універсальні мікропроцесори фірми Motorola MC68020, MC68030, MC68040, MC68060 побудовано на підставі базової моделі MC68000. Їхньою характерною особливістю є наявність засобів, які забезпечують їхню спільну роботу з співпроцесорами для обробки чисел з плаваючою точкою. Розпочинаючи з моделей MC68040, 32-розрядні процесори побудовано за гарвардською архітектурою з розділенням потоків даних та команд; кеш-пам'ять є багаторівневою; є апаратна підтримка реалізації мультипроцесорних систем; введено внутрішні засоби самотестування та налагодження. МП MC68060 має суперскалярну структуру з паралельною роботою двох виконавчих пристроїв обробки цілочисельних операндів і одного пристрою обробки операндів з плаваючою точкою. На базі моделі MC68020 розроблено сімейство спеціалізованих комунікаційних контролерів M683XX.

32-розрядні МП MC680X0 сумісні програмно "знизу догори" і мають ту ж саму програмну модель користувача, що й базова модель МП MC68000.

МП MC68020 зреалізовує розширений набір команд та способів адресування, які стали базовими для всіх наступних моделей сімейства, має внутрішню кеш-пам'ять команд на 256 байт та передбачає підключення зовнішніх співпроцесорів. МП MC68030 і старші моделі вміщують вже додатково кеш-пам'ять даних та блок керування пам'яттю, який виконує сторінкове адресування.

Програмну модель користувача МП сімейства M680X0 подано на рис. 11.18.

Програмна модель складається з восьми 32-розрядних регістрів даних – D7…D0, восьми 32-розрядних регістрів адрес – A7…A0, з яких регістр A7 є вказівник стека користувача USP, 32-розрядного регістра-лічильника команд PC та 16-розрядного регістра стану SR**,** молодші 8 розрядів якого є регістр коду умов (прапорців) CCR користувача, а 8 старших розрядів вміщують біти стану процесора в режимі супервізора. Регістрову модель супервізора подано на рис. 11.19.

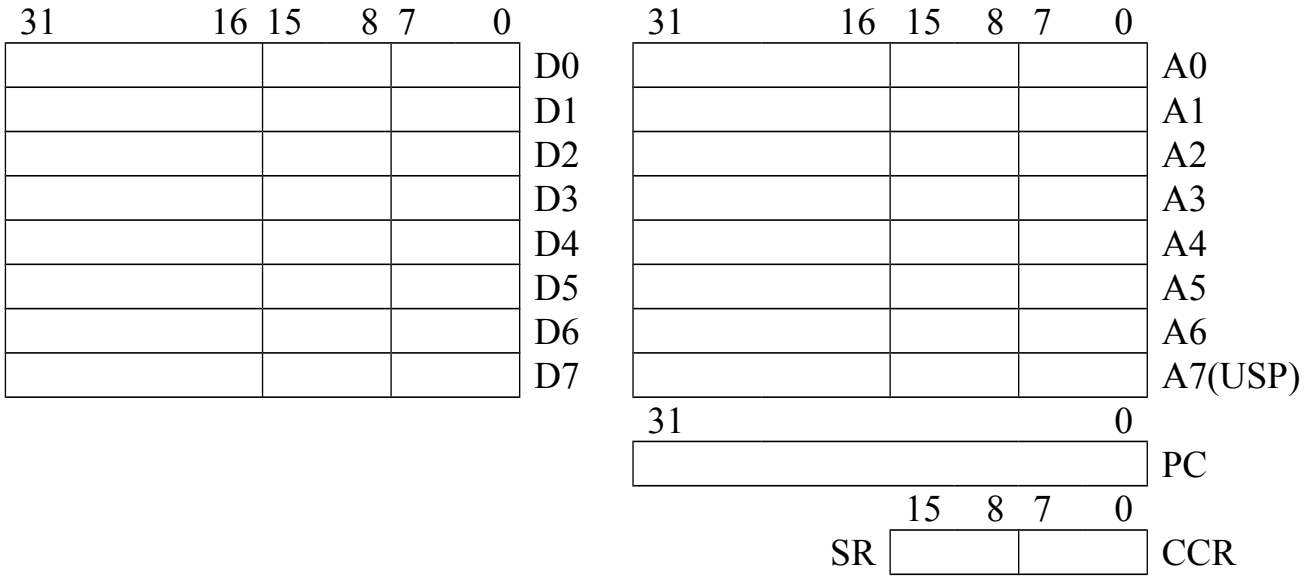


Рисунок 11.18 – Програмна модель користувача

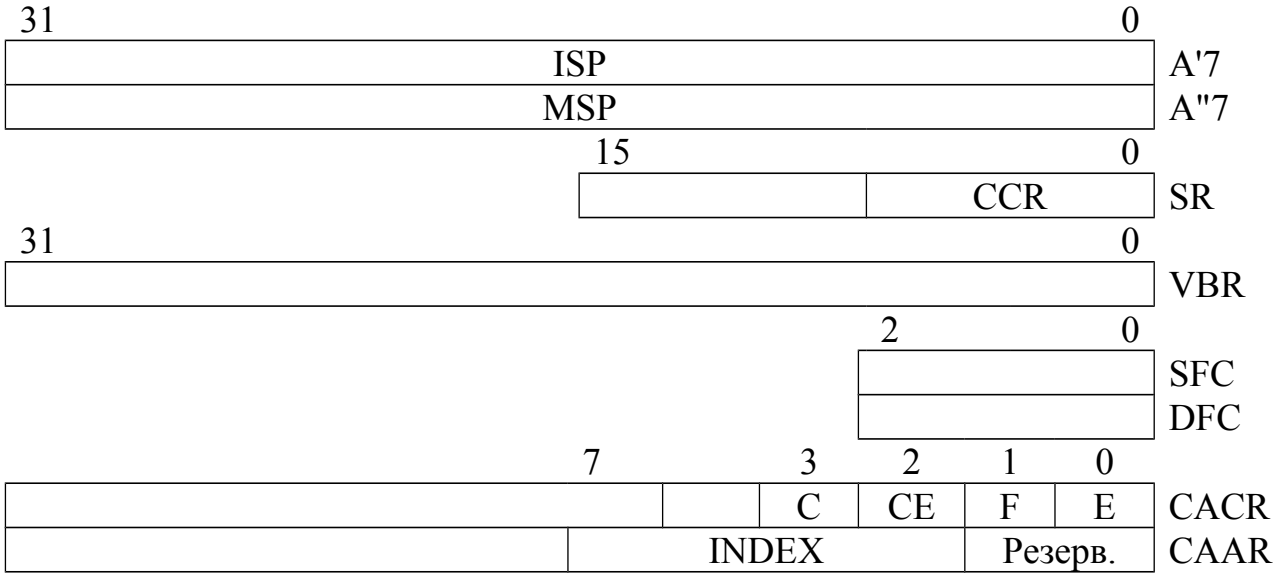


Рисунок 11.19 – Регістрова модель супервізора MC68020

Регістрова модель вміщує додатково два 32-розрядні регістри ISP (A7') та MSP (A7"). Регістр ISP виконує функції вказівника стека супервізора при обробці переривань та виключень, а регістр MSP використовується операційною системою при роботі у багатозадачному режимі. Він вказує на головний стек, до якого при перемиканні задач заноситься адреса повернення до попередньої задачі. Регістр VBR вміщує базову адресу таблиці векторів переривань, яка може переміщуватись у адресному просторі.

До трьох молодших розрядів регістрів SFC та DFC заноситься код адресного простору, який надходить на зовнішні виводи FC2…FC0 при пересиланні даних поміж регістрами процесора D7…D0 або A7…A0 та зовнішньою пам’яттю у привілейованих командах.

Під час запису даних з регістра Dn або An у пам’ять функціональний код FC2…FC0 відповідно до таблиці:

FC2	FC1	FC0	Простір пам’яті
0	0	1	Пам’ять даних користувача
0	1	0	Пам’ять команд користувача
1	0	1	Пам’ять даних супервізора
1	1	0	Пам’ять команд супервізора

надходить на входи МП з регістра DFC, а при зчитуванні з пам’яті – до регістра Dn або An – з регістра SFC. Одночасно функціональний код надходить до блока керування пам’яттю MMU, який звертається до відповідного адресного простору в пам’яті.

Регістри CACR та CAAR керують кеш-пам’яттю; окремі біти регістра CACR мають таке призначення:

E = 1 дозволяє роботу кеш-пам’яті, E = 0 забороняє звернення до кеша, але вміст кеша зберігається;

CE = 1 анулює вміст рядка кеша, номер якого задається вмістом поля INDEX у регістрі CAAR;

F = 1 забороняє заповнення рядка кеша у разі кеш-промаху, фіксуючи поточний стан кеша;

C = 1 анулює вміст всіх рядків кеш-пам’яті.

На рис. 11.20 подано біти умов регістра стану.

Рисунок 11.20 – Регістр стану SR

Біти T1...T0 задають режими трасування:

T1	T0	Режим
0	0	Трасування відсутнє
0	1	Покроковий режим
1	0	Зупин виконання програми після команд JMP, JSR, BRA тощо

Біт S = 1 визначає режим супервізора, S = 0 – режим користувача.

Біт M = 1 при значенні біта S = 1 зумовлює вибір головного вказівника стека MSP, а M = 0 – вибір регістра ISP – вказівника стека переривань.

Контрольні запитання:

1 Які нові регістри порівняно з МП MC68000 вміщує програмна модель користувача МП MC68020?

2 Які основні регістри має регістрова модель супервізора MC68020?

3 Чи має МП MC68020 конвеєр команд?

4 Яка різниця у розміщенні у пам’яті таблиць векторів переривань у мікропроцесорів фірми Intel та Motorola?

5 Які нові прапорці вміщують старші 8 розрядів регістра SR і на що вони вказують?

6 Що таке маска переривань і з якою метою вона використовується?

Контрольні запитання підвищеної складності:

1 Що таке MMU і які функції він виконує?

2 Як працює блок MMU за взаємодії з жорстким диском?

3 Як зорганізовується у МП MC68010 і старших його моделей процедура перезапису відсутньої у ОЗП сторінки з жорсткого диску: апаратно чи паралельно?

11.4 Побудова МПС на 32-розрядних мікропроцесорах фірми Motorola

11.4.1 Підсистема центрального процесорного елемента

Вхідний контроль:

1 Яку розрядність мають ШД та ША МП MC68000?

2 Які системні сигнали ВІС МП68000 Вам відомі?

3 Чи є згадані в п. 2 сигнали односпрямовані або двоспрямовані й чому?

4 За яким алгоритмом працює пріоритетний шифратор?

5 На якій частоті працює МП MC68000?

6 Який сигнал треба подати на вхід AVEC# ВІС МП MC68000?

За основу МПС було взято контролер – модуль розвитку фірми Flight Electroniks – FLIGHT-68EC020 EVM.

У якості центрального процесора в МПС використовується МП MC68EC020, який має 32-розрядну шину даних та 24-розрядну шину адреси, що є достатнім для більшості застосувань МПС у складі пристроїв телекомунікацій. В іншому параметри MC68EC020 збігаються з параметрами MC68020, збігаються також програмні моделі супервізора і користувача. Однаковими, за незначними винятками, є і системи команд. Зовнішні виводи МП MC68EC020 подано на рис. 11.21.

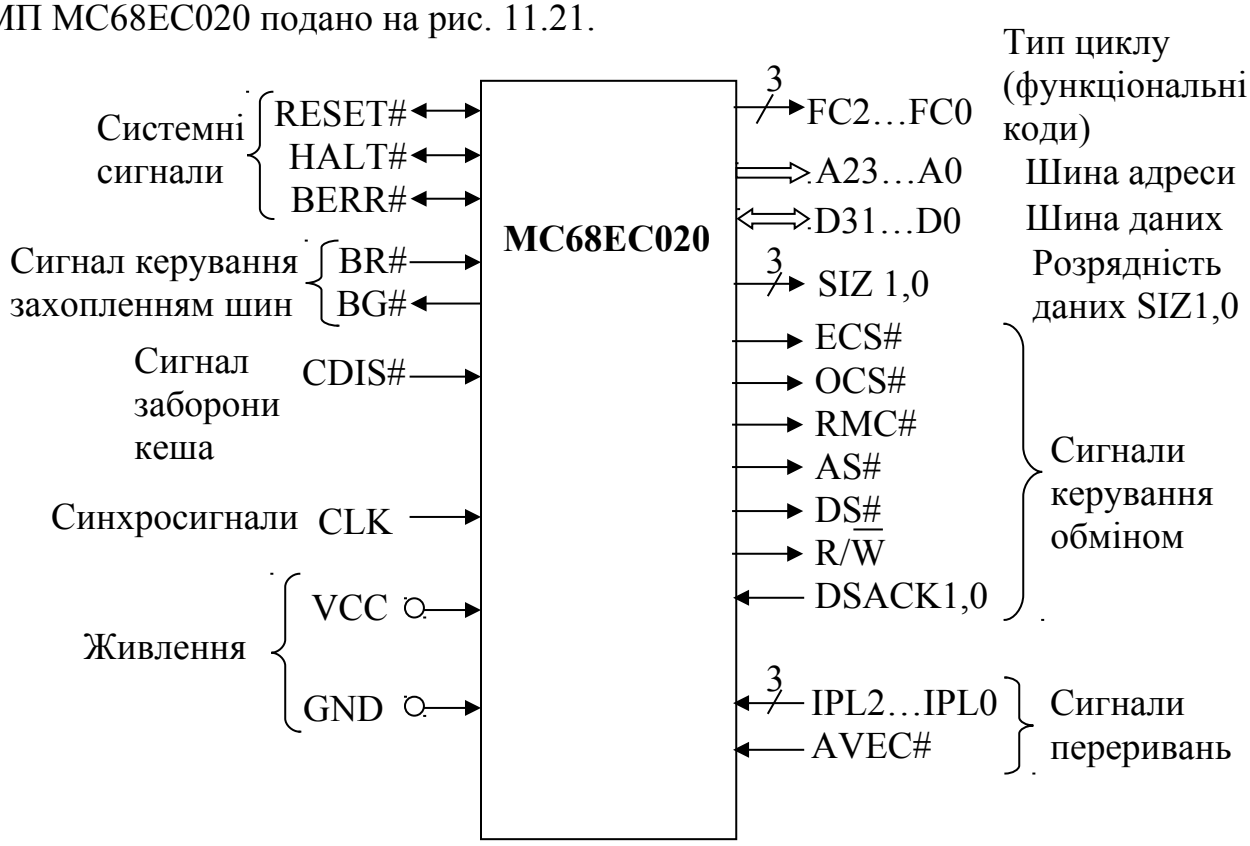


Рисунок 11.21 – Зовнішні виводи МП MC68EC020

На виводи МП A23...A0, D31...D0 надходять 24-розрядні адреси та 32-розрядні дані. Сигнали на виходах FC2...FC0 зазначають тип виконуваного циклу відповідно до табл. 11.14.

За FC2 = FC1 = FC0 = 1 може працювати підключений співпроцесор.

Таблиця 11.14 – Типи виконуваних циклів

FC2	FC1	FC0	Тип виконуваного циклу
0	0	0	резервовано для подальшого розвитку
0	0	1	вибирання даних користувача
0	1	0	вибирання команд користувача
0	1	1	резервовано
1	0	0	резервовано
1	0	1	вибирання даних супервізора
1	1	0	вибирання команд супервізора
1	1	1	цикл центрального процесора

Особливістю 32-розрядних мікропроцесорів є динамічне визначення використовуваної розрядності шини даних, яка може буди 8-, 16-, 32-розрядною. Значення вихідних сигналів SIZ1,0 вказують на розрядність даних, які передаються у циклі (табл. 11.15).

Таблиця 11.15 – Вихідні сигнали SIZ1,0, які визначають розрядність передаваного операнда

SIZ1	SIZ0	Розрядність операнда
0	0	4 байти
0	1	1 байт
1	0	2 байти
1	1	3 байти

Передавання трьох байтів (SIZ1,0 = 11) здійснюється при виборі довгого слова за непарною адресою, яке здійснюється за два послідовних цикли при 32-разрядній шині даних.

Для керування обміном використовуються такі сигнали:

AS# – адресний стробімпульс, набирає активного значення при надходженні адреси на шину A23...A0 і зберігає це значення до завершення циклу обміну;

$R/\overline{W}$ – сигнал, який визначає напрямок передавання даних шиною D31…D0, введення (читання) відбувається за високого рівня сигналу $R/\overline{W} = 1$, а виведення (запис) – за $R/\overline{W} = 0$;

DSACK1,0 – вхідні сигнали підтвердження готовності зовнішніх пристроїв до обміну, які вказують на розрядність шини даних, яка використовується, згідно з табл. 11.16; вийти з циклу очікування можна при надходженні сигналу BERR# = 0 від зовнішньої схеми або при зміні сигналів DSACK1# = DSACK0 = 1 на інше сполучення. Сигнали DSACK1# та DSACK0# вказують на кількість байтів, які залишилось передати у даному циклі, наприклад, при передаванні довгого слова через 8-розрядний пристрій виведення;

Таблиця 11.16 – Вхідні сигнали DSACK1,0, які вказують на розрядність шини даних

DSACK1#	DSACK0#	Розрядність шини даних
0	0	4 байти
0	1	2 байти
1	0	1 байт
1	1	Цикл очікування

DS# – стробімпульс даних, який у циклі читання дорівнює 0 водночас зі стробом адреси; у циклі запису набирає значення DS# = 0 через один такт після адресного стробімпульсу; він сигналізує, що МП видав дані на шину даних;

ECS# – вихідний сигнал початку циклу обміну, який набирає значення ECS ≠ 0 упродовж першого такту кожного нового циклу;

OCS# – вихідний сигнал початку циклу обміну даними, який має значення OCS# = 0 упродовж першого такту початкового циклу передавання даних;

RMC# – вихідний сигнал циклу “читання-модифікація-запис”, який зреалізовується при виконанні команди TAS, значення RMC# = 0 встановлюється на початку першого циклу цієї команди і зберігається в перебігу її виконання.

Сигнали переривань IPL2, IPL1, IPL0, системні сигнали початкового встановлення RESET#, зупину HALT#, помилки звернення до шини BERR#, а також сигнали керування захопленням шини BR#, BG# виконують ті ж самі функції, що й у мікропроцесорі MC68000.

Сигнал заборони роботи кеша CDIS# 0 зазвичай використовується у режимі налагодження МПС.

Сигнал AVEC#, який дорівнює логічному нулю, дозволяє автовекторні переривання.

На вхід CLK подаються зовнішні синхросигнали з частотою 16 МГц. Напруга живлення процесора становить 5 В, споживана потужність не перевищує 2 Вт.

МП MC68EC020 підтримує оброблення до семи запитів на переривання від пристроїв введення-виведення, які повинні подати відповідні сигнали IRQ7#...IRQ1# (INT7#...INT1#) на входи схеми пріоритетного шифратора PCD (рис. 11.22). Для кожного запиту встановлено пріоритет обслуговування: найвищий Li = 7 для запиту на вході IRQ7, найнижчий Li = 1 для запиту на вході IRQ1. Запит переривання ініціюється при надходженні логічного нуля на відповідний вхід IRQi#; на виходах PCD формується трирозрядний код IPL2#, IPL1#, IPL0#, який відповідає номерові запиту, який має максимальний пріоритет. Вхід IRQ0 завжди підмикається до низького потенціалу і за відсутності запитів на входах PCD формує код IPL0#, IPL1#, IPL2# = 111, поданий у інверсній формі, який не спричинює переривань.

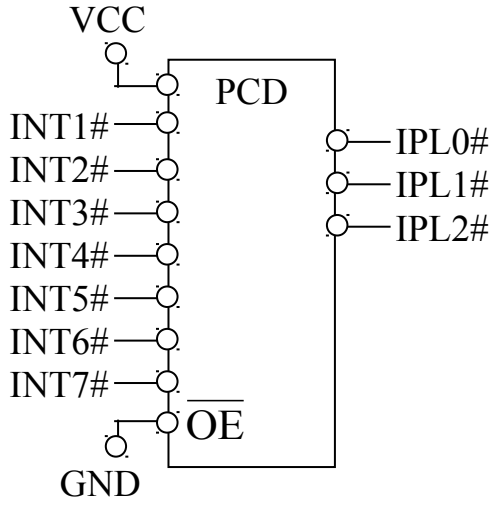


Рисунок 11.22 – Схема формування кодів пріоритетів запитів переривань

Контрольні запитання:

1 З якою метою МП M68EC020 формує сигнали керування обміном ECS# та OCS#?

2 Що визначають вихідні сигнали SIZ1 та SIZ0?

3 Що визначають вхідні сигнали DSACK1# та DSACK0#?

4 Чому не можна переривати цикл “читання–модифікація–запис”?

5 На який вхід пріоритетного шифратора PCD треба підключити пристрій введення-виведення, щоби надати йому 3-й пріоритет у інверсному коді?

6 Який код має виставити МП на виходах FC2, FC1, FC0 у режимі циклу центрального процесора (циклу підтвердження переривань)?

7 З якою метою подається рівень логічного нуля на вхід $\overline{\text{OE}}$ пріоритетного шифратора?

Контрольні запитання підвищеної складності:

1 На входи пріоритетного шифратора надходять одночасно запити на переривання INT1#, INT5#, INT6#. Який код – IPL0#, IPL1#, IPL2# – буде сформовано на виходах пріоритетного шифратора?

2 Маска переривань має значення I2 = 1, I1 = 0,I0 = 0. Зазначте номери пріоритетів обслуговуваних запитів на переривання.

11.4.2 Розподіл адресного простору МПС

Вхідний контроль:

1 З якою метою адресний простір розподіляється поміж підсистемами МПС різних типів?

2 Чи можуть займати адресний простір периферійні пристрої?

3 Чи займає адресний простір центральний процесор?

4 Чи займає адресний простір співпроцесор?

За основу підходу до розділу адресного простору, який формує МП MC68EC020, було взято мапу пам’яті контролера FLIGHT-68EC020EVM. Розподіл адресного простору МПС подано на рис. 11.23.

Адресний простір розподіляється поміж ПЗУ – адреси $000000...$3FFFFF, ОЗП – адреси $400000...$7FFFFF, паралельними периферійними інтерфейсами-таймерами PI/T – адреси $800000...$9FFFFF, та асинхронними послідовними адаптерами DUART – адреси A00000...FFFFFF.

Розподіл адресного простору між типами пристроїв можна зреалізовувати на програмованій логічний матриці – ПЛМ, наприклад 16L8D (PAL2) фірми Motorola (рис. 11.24).

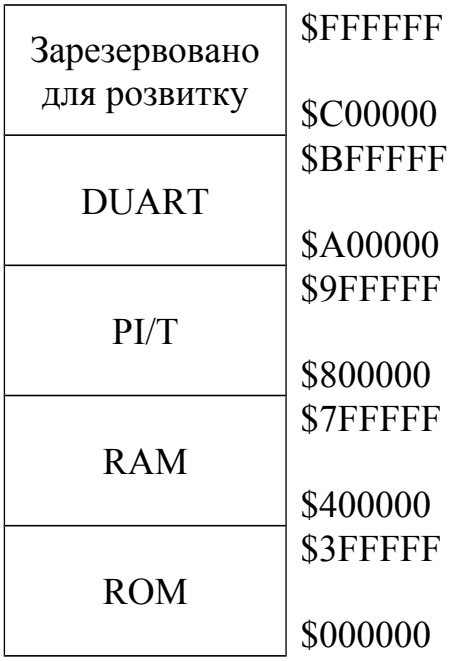


Рисунок 11.23 – Розподіл адресного простору МПС

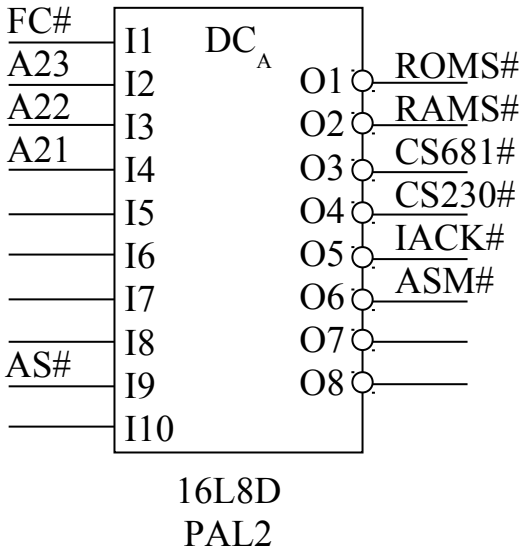


Рисунок 11.24 – Схема формування сигналів розподілу адресного простору

Вихідні логічні функції – це сигнали ROMS# – дозвіл на роботу декодерів адреси ВІС постійної пам’яті, RAMS# – дозвіл на роботу декодерів адреси ВІС оперативної пам’яті, CS230# – дозвіл на роботу декодерів адреси ВІС PI/T, CS681# – дозвіл на роботу декодерів адреси ВІС DUART. Отже, розподіл адресного простору зреалізовується за допомогою ієрархічної структури: ПЛМ розподіляє адресний простір на підпростори для пристроїв різних типів, а всередині кожного підпростору адреси розподіляються поміж окремими ВІС одного призначення за допомогою декодерів адреси.

Перед тим, як застосовувати декодер адресного простору PAL2, треба за допомогою логічної схеми сформувати сигнал $FC\# = \overline{FC2 \wedge FC1 \wedge FC0}$. На рис. 11.25 подано схему формування сигналу FC# – циклу центрального процесора в інверсній формі.

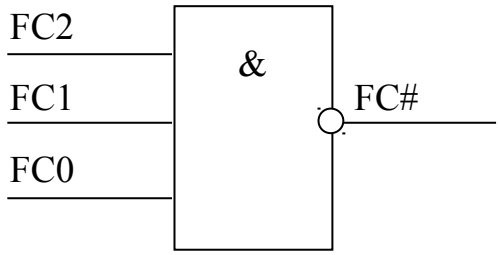


Рисунок 11.25 – Формування сигналу FC#

Вихідні сигнали PAL2 може бути описано в аналітичний формі як функції вхідних сигналів:

$$IACK\# = \overline{FC} \wedge \overline{AS}$$

$$ASM\# = FC \wedge \overline{AS}$$

$$ROMS\# = \overline{ASM} \wedge IACK \wedge \overline{A23} \wedge \overline{A22} \wedge \overline{A21}$$

$$RAMS\# = \overline{ASM} \wedge IACK \wedge \overline{A23} \wedge A22 \wedge \overline{A21}$$

$$CS230\# = \overline{ASM} \wedge IACK \wedge A23 \wedge \overline{A22} \wedge \overline{A21}$$

$$CS681\# = \overline{ASM} \wedge IACK \wedge A23 \wedge \overline{A22} \wedge A21$$

З цих виразів видно, що сигнал ROMS# матиме низький рівень, коли розряди адреси A23, A22, A21 дорівнюватимуть 0, тобто розпочинаючи з адреси $000000.

Сигнал RAMS# дорівнюватиме 0, коли адресні розряди A23, A22, A21 матимуть відповідно значення 010, тобто розпочинаючи з адреси $400000.

Сигнал CS230# дорівнюватиме 0, коли адресні розряди A23, A22, A21 матимуть відповідно значення 100, тобто розпочинаючи з адреси $800000.

Сигнал CS681# дорівнюватиме 0, коли адресні розряди A23, A22, A21 матимуть відповідно значення 101, тобто розпочинаючи з адреси $A00000.

Нульові значення сигналів ROMS#, RAMS#, CS230#, CS681# і слугують розподілу адресного простору.

Формування сигналу читання/запису RWAS#, який подається на входи RAM, здійснюється на логічних елементах (рис. 11.26).

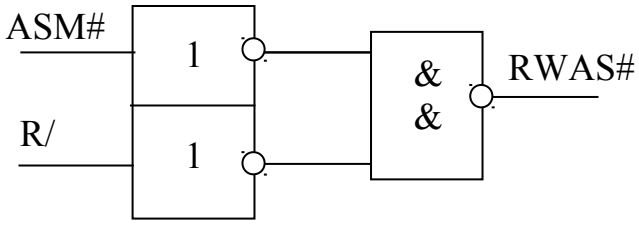


Рисунок 11.26 – Схема формування сигналу RWAS#

Контрольні запитання:

1 Який обсяг пам’яті у Мбайтах займає кожний підпростір пам’яті у мапі пам’яті FLIGHT-68EC020EVM?

2 Які початкові адреси мають адресні підпростори, призначені для адресування кожного типу пристроїв, які входять до підсистем МПС?

3 Чим визначаються значення початкових адрес адресних підпросторів?

4 З якою метою формується сигнал читання/запису RWAS#?

5 Як можна інакше сформувати сигнал RWAS#?

6 З якою метою формується сигнал FC#?

Контрольні запитання підвищеної складності:

1 Як програмуються ПЛМ у якості дешифратора адреси?

2 Проілюструйте, як можна на ПЛМ 16L8D сформувати сигнал дозволу роботи співпроцесора?

11.4.3 Організація підсистеми пам’яті МПС

Вхідний контроль:

1 Скільки мікросхем RAM з організацією 16х4 треба задіяти і як їх поєднати, якщо треба побудувати підсистему пам’яті з організацією 16х8?

2 Скільки мікросхем ROM з організацією 16х8 треба задіяти і як їх поєднати, якщо треба побудувати підсистему пам’яті з організацією 64х8?

3 За допомогою якого пристрою формуються сигнали вибирання мікросхем пам’яті $\overline{\text{CS}}$?

4 Які вхідні сигнали треба подати на мікросхеми пам’яті RAM?

5 Які вхідні сигнали треба подати на мікросхеми пам’яті ROM?

6 Які вихідні сигнали знімаються з мікросхем пам’яті і скільки їх може бути?

Шина даних D31…D0 МП MC68EC020 є тристабільна, двоспрямована, немультиплексована паралельна шина, яка слугує для обміну даними поміж мікропроцесором, пам’яттю та пристроями введення-виведення. За один цикл шини можуть пересилатися 8-, 16-, 24- або 32-розрядні дані.

Пам’ять заданого типу – постійна або оперативна – великого розміру, яка сягає сотні кбайтів або десятків Мбайтів, може бути побудована банками,

кожний з яких складається з чотирьох однакових ВІС. Кожна ВІС призначена для зберігання байтів з однаковими індексами: всіх нульових (В0), перших (В1), других (В2) та третіх (В3). Така конфігурація пам'яті дозволяє легко адресувати будь-яку комірку пам'яті у будь-якому з чотирьох шарів банка. Кількість банків залежить від заданого обсягу пам'яті і ємності ВІС, які утворюють ці банки. Припустимо, що треба забезпечити чотиришарову організацію пам'яті M x 8 кбайт, а в наявності є ВІС оперативної пам'яті з організацією n x 8. Враховуючи,. що один банк складається з чотирьох ВІС, ємність одного банка дорівнює 4 x n x 8 кбайт, а кількість банків, з яких буде побудовано пам'ять, можна обчислити за формулою

$$P = \frac{M \times 8}{4 \times n \times 8},$$

де M – заданий обсяг пам'яті, а n – обсяг пам'яті однієї ВІС. Припустимо, що заданий обсяг пам'яті дорівнює 750 кбайт, а ємність однієї ВІС – 64 кбайт. Ємність одного банка становить 4 x 64 = 256 кбайт, а кількість банків дорівнюватиме трьом.

Зрозуміло, що адресування будь-якої комірки пам'яті такої конфігурації повинна мати ієрархічну структуру. У адресному просторі пам'яті заданого типу спочатку адресується банк пам'яті, потім шар у вибраному банку, а у шарі вже адресується комірка пам'яті.

Адресування банків оперативної чи постійної пам'яті можна зреалізовувати за допомогою звичайного декодера адреси, дозвіл на роботу якого дають сигнали RAMS# або ROMS#, сформовані PAL2. На рис. 11.27 наведено формування сигналів дозволу на роботу банків RAM.

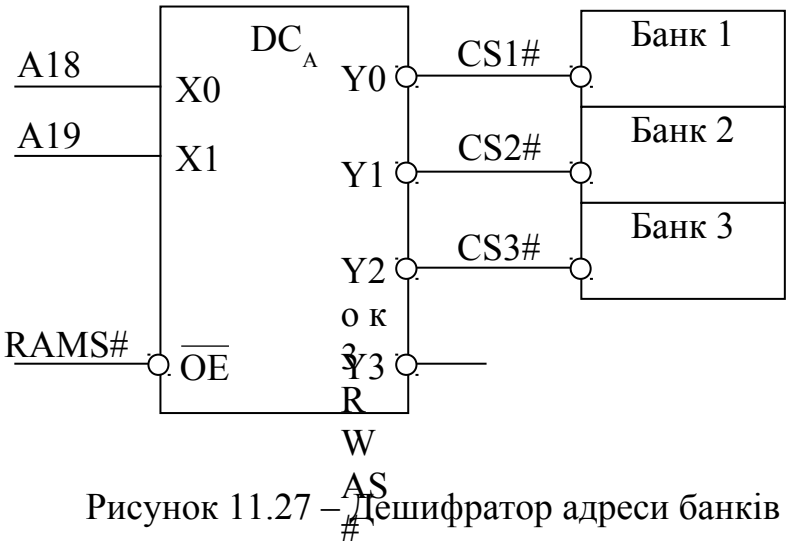


Рисунок 11.27 – Дешифратор адреси банків RAM

ВІС оперативної пам'яті повинні мати два керувальних входи – $\overline{CS}$, $\overline{OE}$ та вхід $R/\overline{W}$, на який подаються сигнали читання-запису RWAS#. В одному банку всі входи $\overline{CS}$ може бути поєднано через те що банк вибирається у

цілому. Вибір шару здійснюється сигналами RAMU# (B3), RAMMU# (B2), RAMLU# (B1), RAML# (B0), які можуть формуватися ПЛМ PAL 16L8D фірми Motorola (PAL1). Формування вихідних сигналів здійснюється відповідно до аналітичних виразів:

$$
\begin{aligned}
\text{RAMU\#} &= \overline{\text{RDY}} \wedge \text{R}/\overline{\text{W}} \wedge \overline{\text{DS}} \vee \overline{\text{RDY}} \wedge \overline{\text{A1}} \wedge \overline{\text{A0}} \wedge \overline{\text{DS}} \\
\text{RAMMU\#} &= \overline{\text{RDY}} \wedge \text{R}/\overline{\text{W}} \wedge \overline{\text{DS}} \vee \overline{\text{RDY}} \wedge \overline{\text{A1}} \wedge \text{A0} \wedge \overline{\text{DS}} \vee \\
&\quad \overline{\text{RDY}} \wedge \overline{\text{A1}} \wedge \text{SIZ1} \wedge \overline{\text{DS}} \vee \overline{\text{RDY}} \wedge \overline{\text{A1}} \wedge \overline{\text{SIZ0}} \wedge \overline{\text{DS}} \\
\text{RAMLU\#} &= \overline{\text{RDY}} \wedge \text{R}/\overline{\text{W}} \wedge \overline{\text{DS}} \vee \overline{\text{RDY}} \wedge \text{A1} \wedge \overline{\text{A0}} \vee \\
&\quad \overline{\text{RDY}} \wedge \overline{\text{A1}} \wedge \overline{\text{SIZ0}} \wedge \overline{\text{SIZ1}} \wedge \overline{\text{DS}} \vee \\
&\quad \overline{\text{RDY}} \wedge \overline{\text{A1}} \wedge \text{A0} \wedge \overline{\text{SIZ0}} \wedge \overline{\text{DS}} \\
\text{RAML\#} &= \overline{\text{RDY}} \wedge \text{R}/\overline{\text{W}} \wedge \overline{\text{DS}} \vee \overline{\text{RDY}} \wedge \text{A0} \wedge \text{A1} \wedge \overline{\text{DS}} \vee \\
&\quad \overline{\text{RDY}} \wedge \text{A0} \wedge \text{SIZ0} \wedge \text{SIZ1} \wedge \overline{\text{DS}} \vee \\
&\quad \overline{\text{RDY}} \wedge \overline{\text{SIZ0}} \wedge \overline{\text{SIZ1}} \wedge \overline{\text{DS}} \vee \\
&\quad \overline{\text{RDY}} \wedge \text{A1} \wedge \text{SIZ1} \wedge \overline{\text{DS}}
\end{aligned}
$$

На рис. 11.28 подано схему декодера адреси шарів пам’яті у банку.

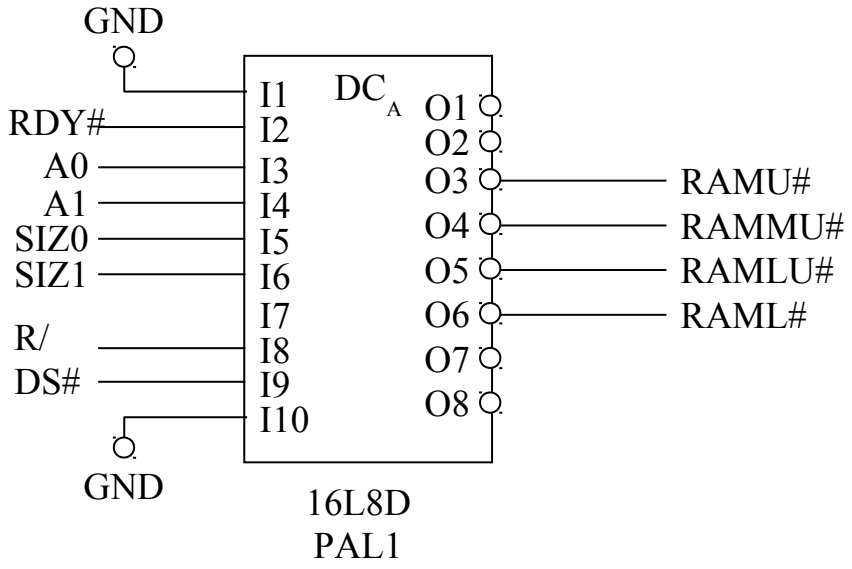


Рисунок 11.28 – Декодер адреси шарів пам’яті у банку

Сигнал RDY є сигнал дозволу роботи RAM або ROM; він може бути сформований на логічних елементах як логічна сума сигналів ROMS# та RAMS#:

$$\text{RDY\#} = \text{ROMS\#} \vee \text{RAMS\#}.$$

Оскільки розряди адреси A1...A0 використовуються для формування сигналів адреси шарів, на адресні входи ВІС оперативної пам'яті на 64 К подаються адресні розряди A17...A2, тому на вхід дешифратора банків можна подавати розряди A18, A19 і в разі необхідності, – A20 (адресування 8-ми банків).

На рис. 11.29 подано схему підсистеми пам'яті, яка вміщує оперативну пам'ять. Підсистема пам'яті на ROM будується аналогічно, але сигнал $R / \overline{W}$ не подається (рис. 11.30).

Контрольні запитання:

1 З якою організацією доцільно використовувати мікросхеми RAM при побудові банка пам'яті обсягом 64 К, яка є чотиришарова?

2 Скільки мікросхем ROM з організацією 64 К треба задіяти для побудови пам'яті з організацією 612 К і скільки банків, які вміщують чотири шари, треба зорганізувати?

3 Який пристрій чи програма забезпечують організацію окремих банків пам'яті для режимів супервізора та користувача МП MC680X0 фірми Motorola?

4 За допомогою якого пристрою розподіляється підпростір адрес, призначених для адресування банків підсистеми пам'яті?

5 За допомогою якого пристрою формуються сигнали дозволу роботи мікросхем різних шарів?

6 Як називаються сигнали, які дозволяють роботу шарів банків пам'яті? Як розшифровуються назви цих сигналів?

7 З якою метою з шини адреси МП M680X0 на адресування мікросхем пам'яті надходять адресні розряди, розпочинаючи з A2?

8 Чому на входи X0 та X1 DC_A (див. рис. 11.27) подаються розряди адреси саме A18, A19, а не інші?

Контрольні запитання підвищеної складності:

1 Чи потребують виходи адресних розрядів A31...A0 шинних формувачів і чому?

2 Чи можуть бути доступними для МП усі 32 розряди даних з чотиришарового банку пам'яті за один цикл шини?

11.4.4 Організація підсистеми введення-виведення

Вхідний контроль:

1 За скільки циклів мікропроцесора можна передати слово через периферійний інтерфейс/таймер MC68230?

2 За скільки циклів мікропроцесора можна прийняти довге слово з периферійного інтерфейса/таймера MC68230?

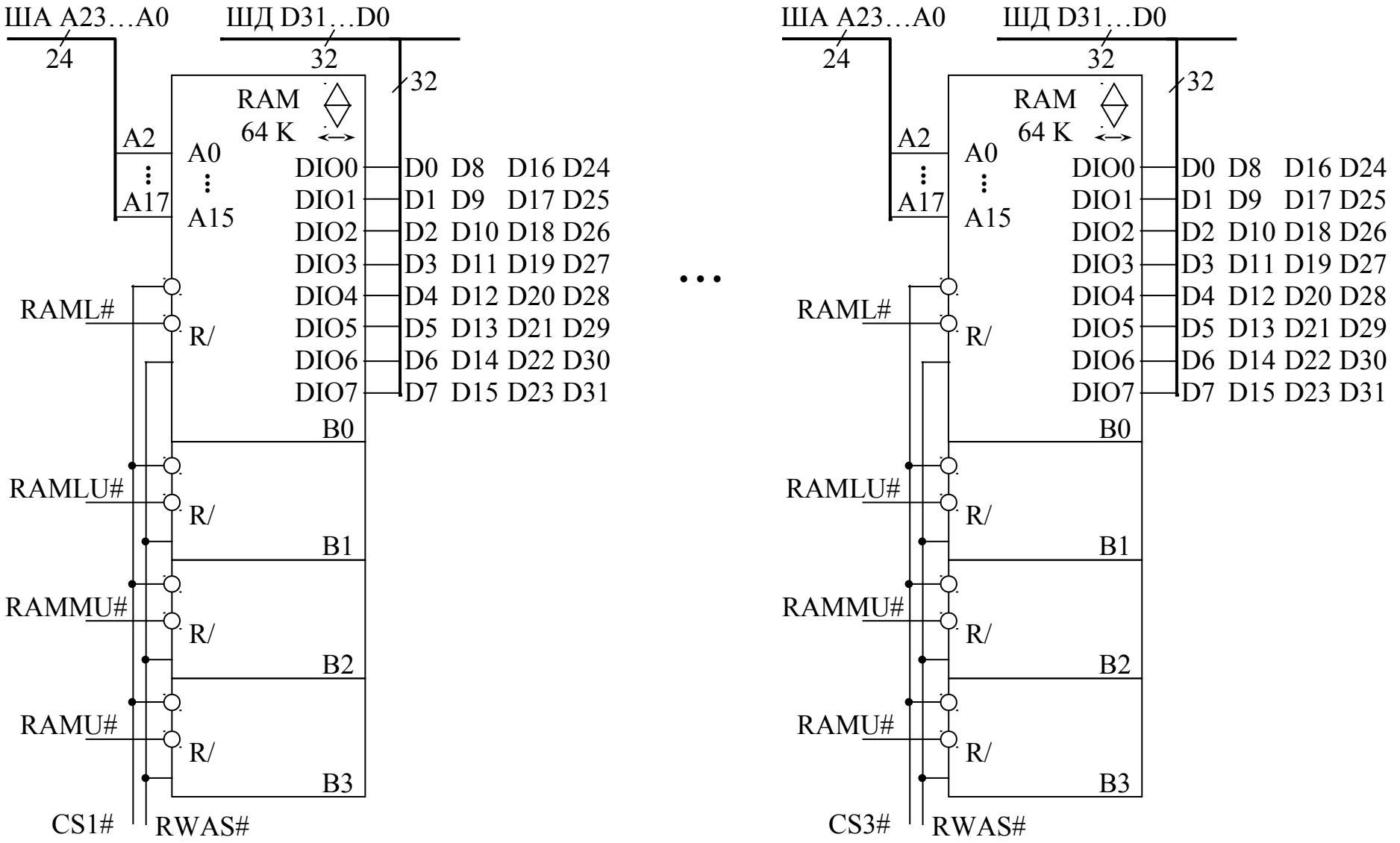


Рисунок 11.29 – Схема підсистеми пам’яті RAM

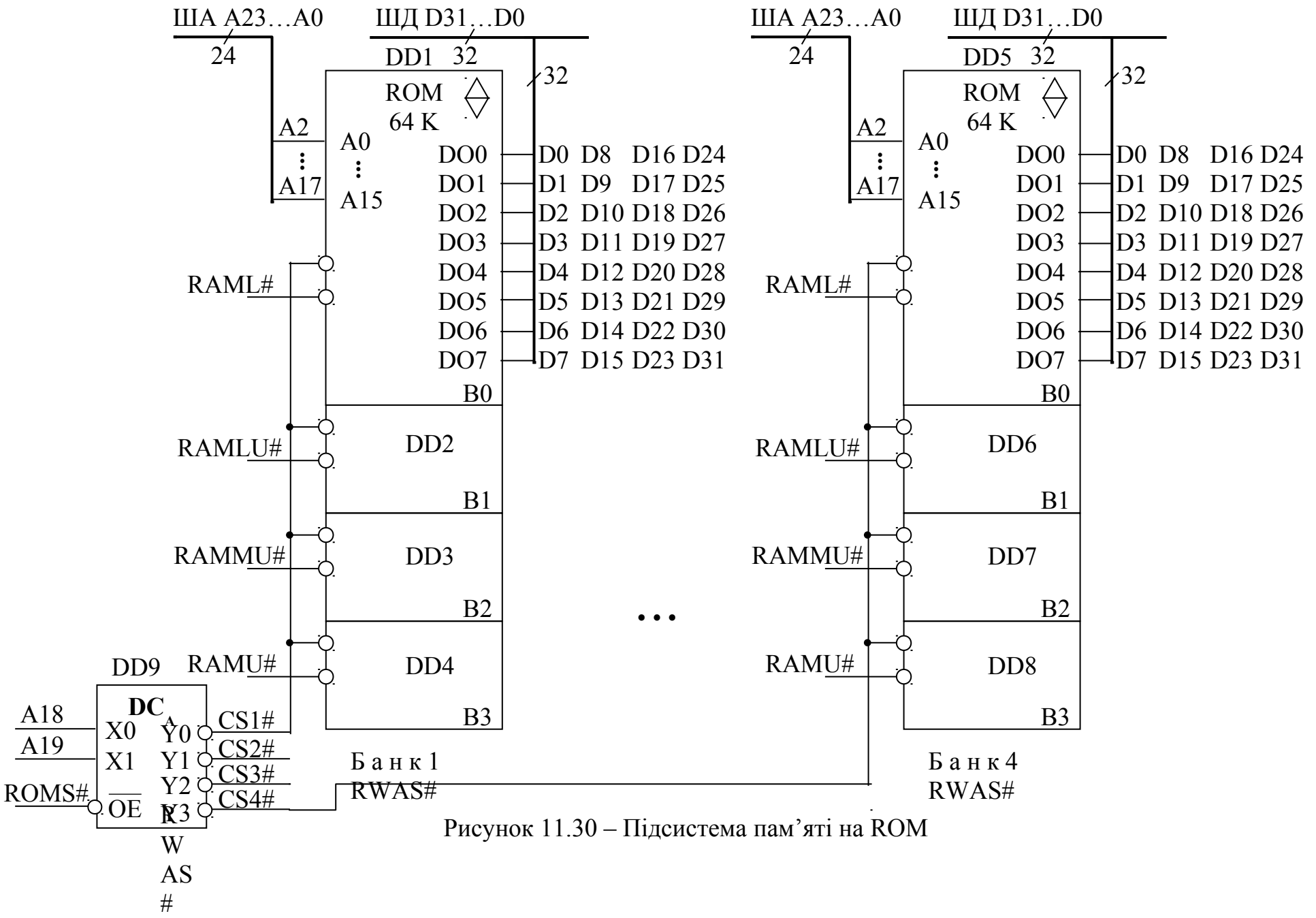


Рисунок 11.30 – Підсистема пам’яті на ROM

3 До яких розрядів шини даних МПС підмикаються входи PI/T?

4 Чи може таймер у складі PI/T викликати переривання роботи МП?

5 Чи можна через PI/T обмінюватись даними поміж периферійними пристроями та пам’яттю у режимі ПДП?

6 Якщо у МПС треба зорганізувати два послідовних канали на приймання та три на передавання, скільки ВІС DUART MC68681 має бути задіяно?

Підключення кількох периферійних інтерфейсів-таймерів та їхнє адресування здійснюється аналогічно до підключення банків пам’яті за допомогою декодерів адреси. У якості сигналу дозволу роботи дешифратора адреси використовується сигнал CS230#. На рис. 11.31 подано підключення трьох PI/T.

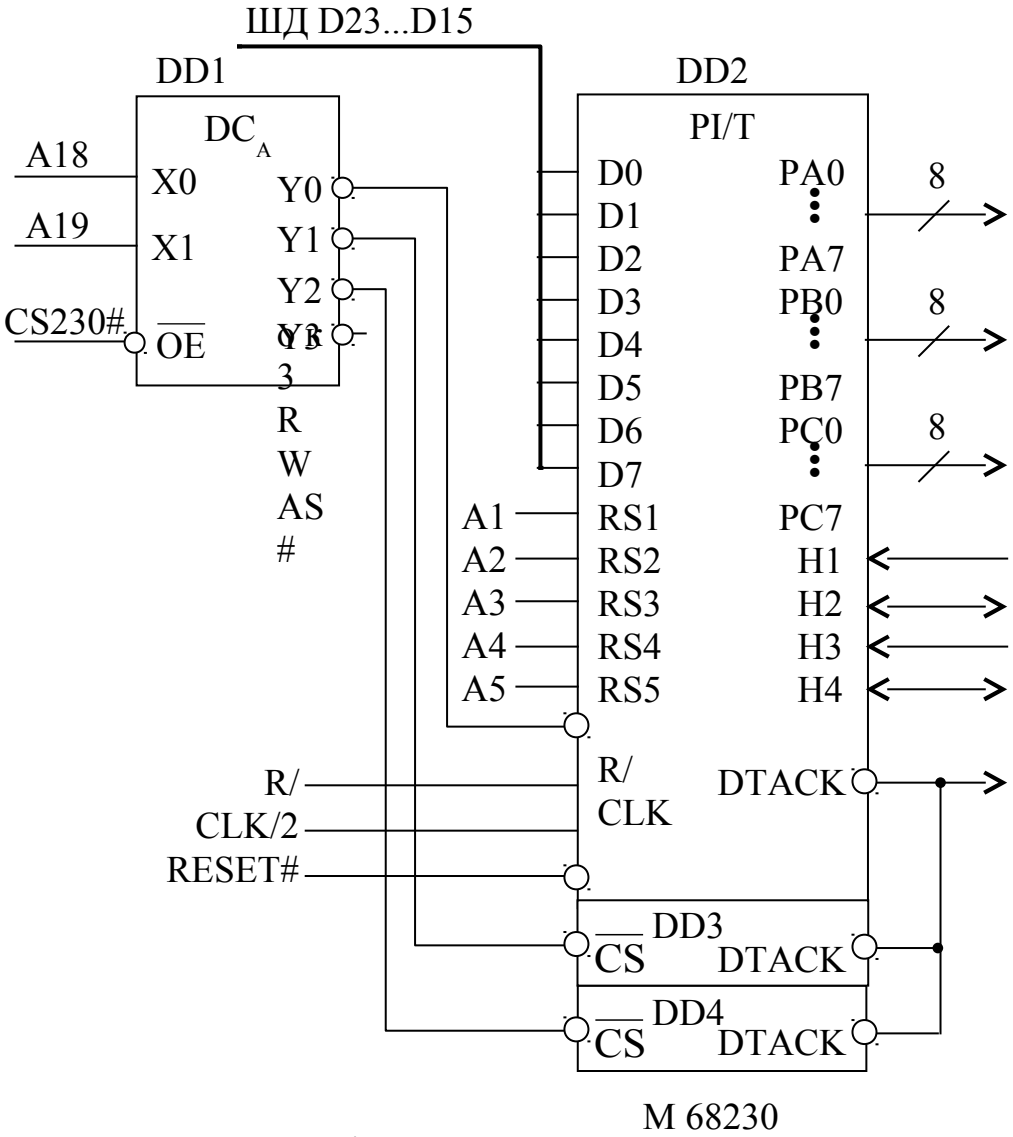


Рисунок 11.31 – Підсистема введення-виведення на трьох PI/T

Завдяки наявності тристабільних виводів PI/T при побудові підсистеми введення-виведення з кількох ВІС відповідні розряди D7…D0, виводи DTACK можна підмикати, так само, як і входи R/$\overline{W}$, CLK та $\overline{RESET}$. Сигнали вибору

чипа підмикаються з виходів декодера окремо до входів $\overline{CS}$ кожного PI/T. Виходи портів PA7...PA0, PB7...PB0, PC7...PC0, а також виводи H1, H2, H3, H4 підмикаються до зовнішніх пристроїв через розмикачі, які і на рис. 11.31 зазначено лише для верхньої ВІС.

Побудова підсистеми введення-виведення на ВІС DUART здійснюється аналогічно, а адресний простір для побудови підсистеми можна робити спільним для різних видів інтерфейсів – паралельних та послідовних, якщо їхня кількість невелика.

На рис. 11.32 наведено частину підсистеми введення-виведення, яка є побудована на DUART.

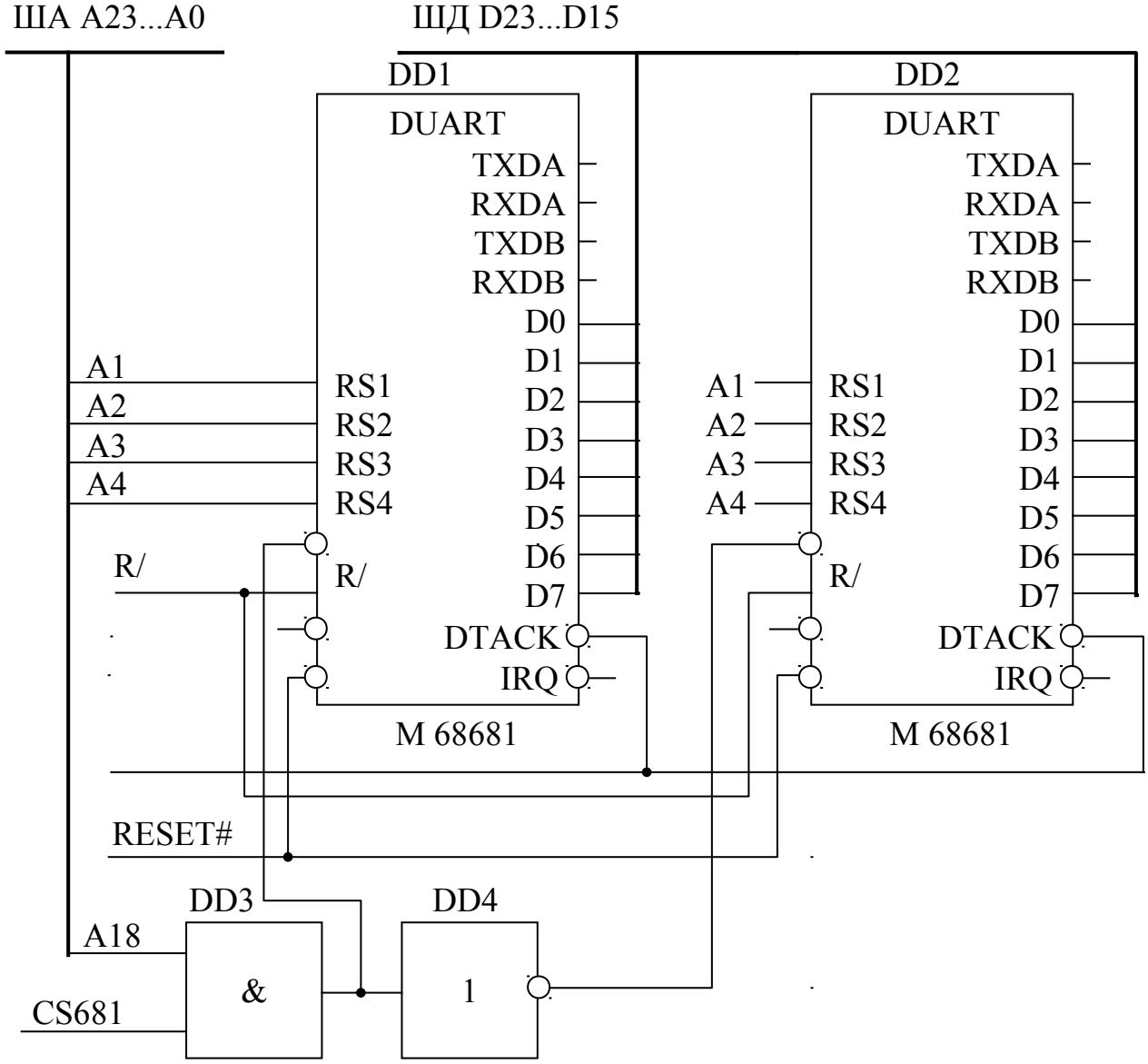


Рисунок 11.32 – Підсистема введення-виведення на двох DUART

Сигнали D7…D0 DUART можна підмикати до шини даних, поєднуючи однакові розряди, так само можна поєднувати сигнали R/$\overline{W}$, RESET# та DTACK. На входи $\overline{CS}$ різних ВІС подаються у загальному випадку сигнали з декодера адреси; у простому випадку, коли ВІС DUART є лише дві, розряд

A18, у протилежних фазах на входи $\overline{CS}$ подається поєднаний з сигналом CS681 на логічному елементі ТА.

Виходи двох приймачів RXDA та передавачів TXDA через підсилювачі-формувачі підмикаються до лінії зв'язку, яка підмикає їх до зовнішніх пристроїв (датчиків, вимірювальних приладів тощо).

На рис. 11.33 наведено схему дешифратора, який формує сигнали підтвердження переривань IACK, які надходять на сім пристроїв введення-виведення (PI/T та DUART).

Рисунок 11.33 – Схема формування сигналів IACK

На рис. 11.34 наведено передавання слова через PI/T за два цикли шини.

15слово0B1B0D23D16ШДSIZ1SIZ0A1A0DSACK1DSACK01-й циклB11000102-й циклB0010110

Рисунок 11.34 – Передавання слова через 8-розрядний інтерфейс

Сигнали DTACK# у разі обміну байтами можуть поєднуватись від усіх пристроїв введення-виведення через логічний елемент АБО-НІ і подаватись на входи DSACK1 та AVEC мікропроцесора.

На вхід DSACK0 можна подати рівень логічного нуля для визначення розміру передаваного операнда як байта. Сигнал BERR# формувати не треба через те, що усі розряди шини адрес є задіяні. Режим ПДП не є передбачений, тому на вхід BR мікропроцесора треба подати рівень L1.

Контрольні запитання:

1 Який адресний простір займають три ВІС МС68230?

2 За допомогою якого комбінаційного вузла можна підключити до МПС три ВІС М68230?

3 Який адресний простір для кожної ВІС М68230 виокремлюється у схемі, яку наведено на рис. 11.31?

4 Який адресний простір виокремлюється для кожної ВІС МС68681 у схемі, яку наведено на рис. 11.32?

Контрольні запитання підвищеної складності:

1 Куди надходять сигнали DTACK ВІС МС68230 та ВІС МС68681?

2 Як формуються сигнали DSACK1 та DSACK0, які подаються на входи МП?

11.4.5 Підключення співпроцесора

Вхідний контроль:

1 Які функції виконує співпроцесор у складі двопроцесорної системи: центральний процесор – співпроцесор?

2 Які типи співпроцесорів Ви знаєте?

3 Який пристрій – центральний процесор або співпроцесор– є ініціатором їхньої спільної роботи?

Робота співпроцесора здійснюється в циклі, коли FC2 = FC1 = FC0 = 1, або, інакше, у просторі CPU.

Мікропроцесор MC68EC020 припускає підключення співпроцесора обробки чисел з плаваючою точкою типу MC68881; тип процесора визначається кодом, який вміщує поле формату команди CPID співпроцесора (рис. 11.35). Мнемоніка команди співпроцесора розпочинається з символу F (1111).

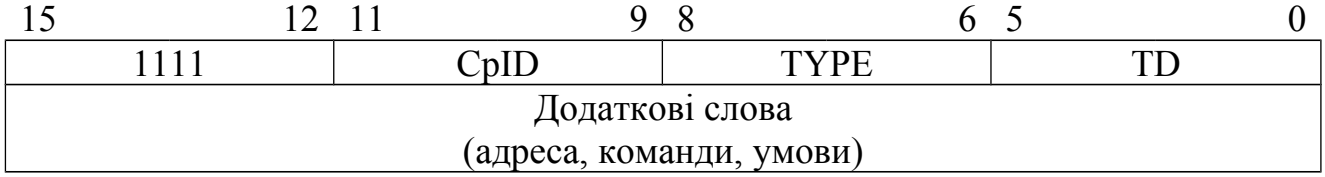


Рисунок 11.35 – Формат команд співпроцесора

Поле TYPE визначає тип команди, яку виконує співпроцесор, – код операції, розгалуження за умовою, реалізація циклу тощо. Поле CD й решта слів команди вміщують додаткову інформацію: адресу операнда, вибір операції, умови тощо.

Після отримання команди співпроцесора мікропроцесор виконує цикл звернення до співпроцесора: FC2, FC1 та FC0 дорівнюють 1, а на шину адреси

(рис. 11.36) надходить інформація, яка описує стан шини адреси при зверненні до співпроцесора. Вміст розрядів A19...A13 використовується для адресування операнда зазначеного в команді співпроцесора, а розряди A4...A0 обирають регістр у складі інтерфейсу співпроцесора, який слугує за джерело або приймач інформації.

31 ... 20	19 ... 16	15 ... 13	12 ... 5	4 ... 0
0 ... 0	0010	CpID	0 ... 0	Регістр

Рисунок 11.36 – Призначення розрядів шини адреси при зверненні до співпроцесора

При зверненні до співпроцесора мікропроцесор видає сигнали AS#, DS#, R/$\overline{W}$, RESET# і приймає від нього сигнали підтвердження DSACK1#, DSACK0#.

При виконанні команди співпроцесора програмний лічильник мікропроцесора PC вміщує адресу кода операції (першого слова команди). На рис. 11.37 подано ВІС математичного співпроцесора MC68881. Сигнал вибору співпроцесора CPCS формується на ПЛМ або на декодері адреси.

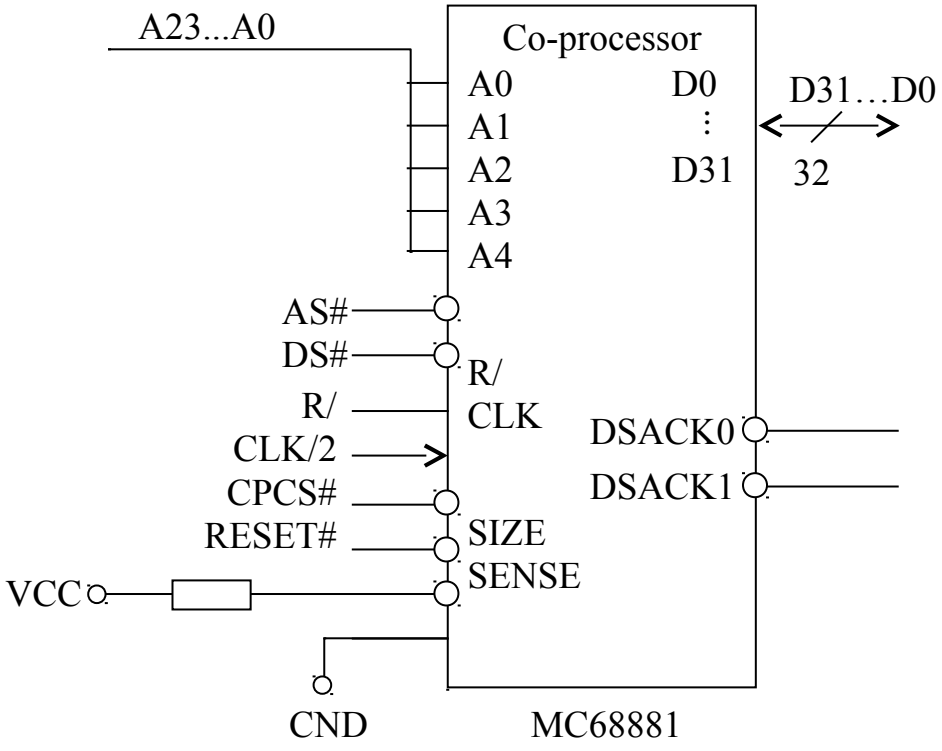


Рисунок 11.37 – ВІС співпроцесора та її підключення

Контрольні запитання:

1 Які поля входять до складу формату команди співпроцесора й що вони визначають?

2 З якого символу розпочинається кожна команда співпроцесора й чому?

3 Які пристрої можуть формувати сигнал CS# вибору мікросхеми співпроцесора?

4 Яку розрядність має шина даних співпроцесора й чому?

Контрольні запитання підвищеної складності:

1 Яку функцію виконують у МПС вихідні сигнали співпроцесора DSACK0# та DSACK1#?

2 Які типи співпроцесорів може кодувати поле CpID:

а) 000;

б) 001?

12 ПРОГРАМУВАННЯ УНІВЕРСАЛЬНИХ МП ФІРМИ MOTOROLA

12.1 Мова Асемблер програмування МП фірми Motorola

Вхідний контроль:

1 Який формат мають типові команди мови Асемблер-86?

2 Де у форматі команди пересилань мови Асемблер-86 є джерело, а де приймач?

3 В який спосіб у командах мови Асемблер-86 зазначається розрядність операндів?

4 У яких системах числення може бути подано операнди у командах мови Асемблер-86?

5 Які способи адресування підтримують МП фірми Intel?

Мова Асемблер МП сімейства M680X0 є спільна для МП MC68000 як базової моделі і подальших моделей і використовує формат двоадресної типової команди.

Формат типової двоадресної команди подано на рис. 12.1.

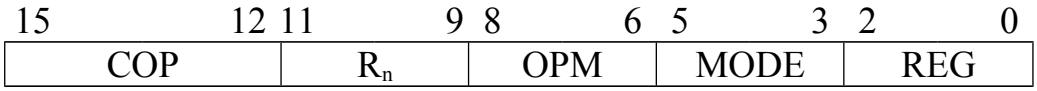


Рисунок 12.1 – Формат типової двоадресної команди

Поле COP визначає код виконуваної операції, поле R_n вміщує номер регістра – n, в якому зберігається операнд “приймач”, OPM – розрядність і розміщення результату; поля MODE та REG – спосіб адресування і розміщення операнда “джерело”.

Якщо виконується адресування з індексуванням, код команди вміщує друге слово, формат котрого наведено на рис. 12.2.

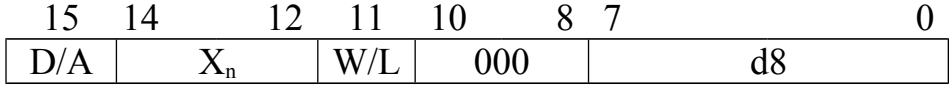


Рисунок 12.2 – Формат другого слова команди при адресуванні з індексуванням

Поле D/A визначає тип регістра, який використовується в якості індексного. За D/A = 0 в якості індексного використовується один з регістрів даних, а за D/A = 1 – регістр адреси.

Поле X_n вміщує номер регістра, який використовується в якості індексного.

Поле W/L визначає розрядність індексу. За W/L = 0 в якості індексу використовуються 16 молодших розрядів індексного регістра з поширенням

знаку до 32-х розрядів. За W/L = 1 за індекс служить 32-розрядний вміст індексного регістра.

Поле d8 вміщує 8-розрядне зміщення, яке подано у доповнювальному коді.

Деякі способи адресування потребують доповнення кодів команди, які розглянуто. Приміром, при безпосередньому адресуванні код команди доповнюється одним чи двома словами, які вміщують безпосередній операнд Im8, Im16 або Im32. Якщо при адресуванні операнда “джерело” чи “приймач” використовується 16-розрядне зміщення або використовується абсолютна адреса, яка має розмір слово (Abs.W) або подвійне слово (Abs.L), то код команди доповнюється відповідними словами.

У загальному вигляді типова двоадресна команда мови Асемблер МП MC68000 має вигляд

COP.x <src>,<dst>,

де COP – мнемокод відповідної команди, замість x ставиться символ, який визначає розрядність операндів: B – байт, W – слово (16 розрядів), L – довге слово (32 розряди); якщо символ розрядності є відсутній, то за умовчанням операнд є слово. Операнди умовно позначаються як <src> – джерело, <dst> – приймач, який слугує за власне приймач результату операції. При записуванні конкретних команд замість <src> та <dst> зазначаються символічні адреси операндів мовою Асемблера відповідно до способу їхнього адресування. За безпосереднього адресування замість <src> зазначається число, перед яким ставиться префікс #.

Операнди, адреси та зміщення у командах можуть подаватись у системах числення, які зазначаються символами:

& – десяткова система,
% – двійкова система,
@ – вісімкова система,
$ – шістнадцяткова система,

символи розміщують перед даними.

За відсутності символу, який зазначає систему числення, дане сприймається як десяткове число.

Мікропроцесор MC68000 виконує читання з пам’яті 16-розрядних операндів, так само як їхній запис до пам’яті, навіть коли у команді зазначено в якості операнда байт; у циклі звернення до пам’яті обирається слово, молодший байт якого використовується як операнд. Довге слово обирається за два послідовних цикли шини, причому першими обираються старші 16 розрядів. Отже, адреси команд чи даних, що їх формує МП, мають бути парними.

Мікропроцесори MC68000 та старші моделі використовують такі способи адресування операндів:

- регістрове (операнд у регістрі даних чи адреси), наприклад
  MOVE.B D1,D0
  MOVEA.L A2,A3;

− непряме регістрове (операнд у комірці пам’яті, яка є адресована вмістом регістра адреси), наприклад

```
MOVE (A0),D1
```

− непряме регістрове з постінкрементом (операнд у комірці пам’яті, адресу якої розміщено в адресному регістрі і після виконання команди нарощується на 1, 2 чи 4 залежно від довжини зазначеного операнда), наприклад

```
MOVE D3,(A1)+          ; Вміст A1 після виконання команди
                       ; нарощується на 2
```

− непряме регістрове з предекрементом (операнд у комірці пам’яті, адресу якої розміщено в адресному регістрі і перед виконанням команди зменшується на 1, 2 чи 4 залежно від довжини зазначеного операнда), наприклад

```
MOVE.L -(A1),D2        ; Вміст A1 перед виконанням команди
                       ; зменшується на 4
```

− непряме регістрове зі зміщенням (операнд у комірці пам’яті, адреса якої є алгебраична сума вмісту регістра адреси та 16-розрядного зміщення, яке задається у команді), наприклад

```
MOVE.B -$A(A0),D3    ; Для МП MC68000
MOVE.B ($44,A3),D4   ; Для старших моделей
```

− непряме регістрове з індексуванням (операнд у комірці пам’яті, адреса якої є сума вмісту регістра адреси, індексного регістра та зміщення, заданого у команді), наприклад

```
CLR.B $40(A0,D3.W)   ; Для МП MC68000
CLR ($1234,A2,D3.L)  ; Для старших моделей
```

− пряме (операнд у комірці пам’яті, адреса якої задається числом зі знаком, прямо зазначеним у команді), наприклад

```
JMP $1234              ; Для МП MC68000
MOVE.B D0,-$07FF.W     ; Для старших моделей ефективна адреса
                       ; становить $FFF801
```

− відносне (операнд у комірці пам’яті, адреса якої є сума поточного вмісту програмного лічильника PC та заданого у команді зміщення); поточний вміст PC, який використовується для обчислення відносної адреси, дорівнює адресі першого слова виконуваної команди плюс 2, наприклад

```
JMP *+$10              ; Для МП MC68000
JMP (*+$10,PC)         ; Для старших моделей
```

− відносне з індексуванням (операнд у комірці пам’яті, адреса якої є сума вмісту PC, індексного регістра та зміщення, заданого у команді), наприклад

```
CLR $10(PC,A2.W)       ; Для старших моделей
CLR ($1000,PC,A2.W) ; Для старших моделей
```

− безпосереднє (значення операнда задано безпосередньо у команді), наприклад

```
MOVEQ #$40,D3          ; Швидке завантаження
MOVEQ #64,D3           ; безпосередньо заданого операнда у D3
```

МП сімейства M680X0 підтримують усі способи адресування, що і МП MC68000, й окрім того, додаткові типи непрямого регістрового адресування з індексуванням.

Непряме регістрове адресування з постіндексуванням

Синтаксис Асемблера має вигляд

([bd,An],Ri.s*SCALE,od),

де bd – базове зміщення; Ri.s*SCALE – значення індексного регістра An або Dn, яке множиться на масштабний множник SCALE – 1, 2 ,4 або 8; od – вихідне зміщення; s – символ розрядності індексного регістра, може дорівнювати W або L. Вміст індексного регістра Ri.s трактується як число зі знаком і у разі s = W знак поширюється на 32 розряди.

Наприклад, запис A3.W*2 означає, що за індекс у команді слугує вміст 16 молодших розрядів регістра адреси A3, зсунутий на 1 розряд ліворуч та розширений знаком до 32 розрядів.

Команда

CLR ([$1234,A2],D3.L,$5678)

очищуватиме комірку пам’яті з ефективною адресою EA, яка підраховується в такий спосіб. Припустимо, що вміст регістра A2 становить $400600, а регістра D3 – відповідно $1000. Частина ефективної адреси EA, подана у квадратних дужках, має вигляд

$$\begin{array}{rl} & \$400600 \\ + & \underline{\$001234} \\ & \$401834 \end{array}$$

Припустимо, що за цією адресою у пам’яті зберігається число $600600. Тоді повна ефективна адреса, зазначена у команді, матиме вигляд

EA = $600600 + $1000 + $5678 = $606C78.

Внаслідок виконання команди буде обнулено 16 молодших розрядів даного, яке зберігається у пам’яті, розпочинаючи з адреси $606C78.

Непряме регістрове адресування з преіндексуванням

Синтаксис Асемблера має вигляд

([bd,An,Ri.s*SCALE],od)

Ефективна адреса EA підраховується аналогічно до попереднього способу.

Команда

CLR ([$1234,A2,D3.L]$5678)

очищуватиме комірку пам’яті з ефективною адресою EA, частина якої, подана у квадратних дужках, матиме вигляд

$400600 + $1234 + $1000 = $402834.

Припустимо, що розпочинаючи з цієї адреси, у пам’яті зберігається довге слово $600600, тоді

EA = $600600 + $5678 = $605C78.

Внаслідок виконання команди буде обнулено 16 молодших розрядів даного, яке зберігається у пам’яті, розпочинаючи з адреси $605C78.

Непряме відносне адресування з індексуванням

Синтаксис Асемблера має вигляд

(*+d,PC,Ri.s*SCALE)

або

(xxx,PC,Ri.s*SCALE).

Вмістом PC вважається адреса наступної команди.

За непрямого відносного адресування з постіндексуванням та преіндексуванням ефективна адреса формується як за непрямого регістрового адресування, але замість вмісту регістра An використовується вміст програмного лічильника PC:

EA = [PC + bd] + (Xn) * SCALE + od – у разі постіндексування
EA = [PC + (Xn) * SCALE + bd] + od – у разі преіндексування

Якщо замість PC вказати ZPC, то можна задати в команді значення програмного лічильника, дорівнюване 0.

Нижче наводяться приклади використовування команди JMP з різними способами адресування:

JMP ($400610,PC,A2.W)
JMP (*+$10,PC,A2.W)
JMP ($4,PC,A2.W)
JMP (*+$1000,PC,A2.W)

JMP ($400610,ZPC,A2.W)

Команди Асемблера МП68000 можуть займати від одного до 5 байт, а старших моделей – до 6 байт.

Контрольні запитання:

1 Наведіть формат типової команди МП М680Х0.

2 Зазначте, де у форматі типової команди є джерело, а де – приймач.

3 В який спосіб у форматі команди зазначається розрядність операндів?

4 У яких системах числення можна подавати операнди, адреси та зміщення у командах мови Асемблера МП фірми Motorola?

5 Які способи адресування операндів підтримує МП МС68000?

6 Які способи адресування операндів підтримує додатково МП МС68020 і старших моделей і з чим це пов’язано?

Контрольні запитання підвищеної складності:

1 З якою метою, на Ваш погляд, використовується при роботі з масивами непряме регістрове адресування з постіндексуванням?

2 З якою метою, на Ваш погляд, використовується при роботі з масивами непряме регістрове адресування з преіндексуванням?

3 У яких випадках доцільно використовувати кожний із зазначених в пп.. 1 та 2 видів адресування?

12.2 Система команд МП МС680Х0 (Для самостійного вивчення)

Вхідний контроль:

1 Зазначте, де у команді мови Асемблер-86 SUB AX,BX є джерело, а де – приймач?

2 Зазначте, де у команді Асемблера МП М680Х0 SUB D0,D1 є джерело, а де – приймач?

3 Яку розрядність має операнд у команді мови Асемблер-86 MOV AX,70H?

4 Яку розрядність має операнд у команді мови Асемблера МП М680Х0 MOVE D0,D1?

5 Чи виставляють команди пересилань мови Асемблер-86 прапорець знаку?

6 Чи виставиться прапорець ZF при виконуванні команди MOV BX,0000H?

12.2.1 Команди пересилань

Команда MOVE пересилає вміст джерела до приймача, наприклад:

```
MOVE.L #$12345678,$400700.L  ; Запис безпосереднього даного до
                             ; комірки пам’яті з адресою $ 400700
MOVEA A3,A4                  ; Пересилання молодших 16-ти розрядів
```

```
                        ; адресного регістра A3 у A4, старші
                        ; 16 розрядів A4 розширюють знаком
MOVEA.L D0,A0           ; Пересилання вмісту регістра D0 до A0
PEA (A0)                ; Запис вмісту регістра D0 до стека
MOVEA.L #$800015,A0     ; Завантаження адреси додаткового
                        ; регістра PAAR порту A PI/T до A0
MOVEP (0, A0),D0        ; Введення слова з додаткових регістрів
                        ; PAAR та PBAR за адресами $800015 та
                        ; $800017 до D0
MOVEP D0,(-4,A0)        ; Збереження слова з D0 у регістрах
                        ; даних PADR та PBDR, які мають
                        ; відповідно адреси $800011 та $800013
```

Команда MOVEP забезпечує пересилання 16- або 32-розрядних операндів через 8-розрядні периферійні пристрої за два або чотири цикли обміну.

```
EXG D1,D3               ; Обмін вмістом регістрів D1 та D3
EXG A4,D2               ; Обмін вмістом регістрів A4 та D2
```

Слід відзначити, що команда пересилання MOVE виставляє прапорець N = 1 у разі негативного операнда, який пересилається, і скидає всі інші прапорці. Якщо пересилається операнд, який дорівнює 0, встановлюється прапорець Z = 1, а інші скидаються. Прапорець X є індиферентний відносно значення операнда, який пересилається.

12.2.2 Команди арифметичних операцій

```
ADD.B (A3),D2           ; Додавання молодшого байта регістра D2
                        ; до вмісту комірки пам’яті, яка є
                        ; адресована A3
ADDA D3,A0              ; Додавання слова з A0 до слова у D3,
                        ; результат записується до A0
ADDI #$1234,(A3)+       ; Додавання слова з комірок пам’яті, які
                        ; мають адреси (A3) та (A3) + 1
                        ; до числа $1234, після чого до вмісту A3
                        ; додається два
ADDQ.L #8,(22,A3)       ; Довге слово з комірок пам’яті, які є
                        ; адресовані вмістом (A3) + 22, (A3) + 23,
                        ; (A3) + 24 та (A3) +25, додається до даного
                        ; 8, результат записується за цими ж
                        ; адресами, швидке додавання
ADDX -(A3),-(A4)        ; Перед додаванням адреси комірок пам’яті
                        ; A3 та A4 зменшуються на 2, після чого
                        ; до вмісту комірки пам’яті, адресованої A4,
```

```
                        ; додається вміст комірки пам'яті,
                        ; адресованої A3, та значення прапорця X
SUB.L #1,D0             ; Віднімання від вмісту 32-розрядного
                        ; регістра D0 одиниці
SUBA.L #$400,A6         ; Віднімання від довгого слова, яке
                        ; вміщується у адресному регістрі A6,
                        ; безпосереднього даного $400
SUBI #250,(A6)+         ; Віднімання від слова, розташованого у
                        ; комірках пам'яті з адресами A6 та
                        ; A6 + 1, десяткового даного 250; після
                        ; операції вміст A6 збільшиться на два
SUBQ.L #2,A2            ; Швидке віднімання безпосередньо
                        ; заданого операнда, який не може
                        ; перевищувати 3-х розрядів
NEG D0                  ; Віднаходження доповнення до 2 – D0: =
                        ; (D0) + 1, знак даного змінюється
CMP (A4),D7             ; Порівняння вмісту регістра D7 із вмістом
                        ; комірки пам'яті, яка адресується (A4) за
                        ; рахунок внутрішнього віднімання,
                        ; результат нікуди не записується, а
                        ; виставлені прапорці використовуються
                        ; для реалізації розгалужень
CMPM.B (A0)+,(A1)+      ; Порівняння відповідних однобайтових
                        ; елементів масивів, адресованих вмістом
                        ; A0 та A1; при виконуванні команди від
                        ; елементів масиву (A1)+ віднімаються
                        ; елементи масиву (A0)+
```

Команди додавання та віднімання встановлюють усі прапорці залежно від здобутого результату, окрім прапорця X.

```
MULS (A3),D2            ; Множення зі знаком слова з D2 на слово, яке є
                        ; розташоване у комірках пам'яті, адресованих
                        ; вмістом A3 та A3 + 1; 32-розрядний результат
                        ; розміщується у регістрі D2
MULU (A3),D0            ; Множення без знаку слова з регістра D0 на слово,
                        ; розташоване у пам'яті, адресу якого зазначено у
                        ; регістрі A3; 32-розрядний результат розміщується
                        ; в регістрі D0
DIVS D1,D3              ; Ділення зі знаком довгого слова з D3 на довге
                        ; слово, розміщене у D1; біти результату
                        ; розміщуються у бітах D0...D15 регістра D3, а
                        ; стача у бітах D16...D31 регістра D3
DIVSL D1,D3             ; Ділення зі знаком довгого слова з D3 на довге
                        ; слово, розміщене у D1; біти результату
```

```
                    ; розміщуються у бітах D0...D31 регістра D3, а
                    ; стача губиться
```

Розширення набору команд 32-розрядних процесорів пов'язано із введенням нових груп команд, які забезпечують операції з бітовими полями, поширюють групу арифметичних та інших операцій, а також команд, які забезпечують роботу зі співпроцесором.

Команди пересилань з регістрів CCR та SR та їхнє завантаження мають вигляд

```
MOVE CCR,D3     ; Пересилання вмісту регістра CCR у D3
MOVE #0,CCR     ; Обнулення регістра CCR
MOVE SR,D0      ; Пересилання вмісту регістра SR у D0
MOVE #$700,SR   ; Встановлення маски пріоритетів переривань, яка
                ; дорівнює 7
```

Команди обміну поміж вказівником стека користувача USP та адресним регістром мають вигляд

```
MOVE USP,A0     ; Пересилання вказівника стека USP до A0
MOVE A7,USP     ; Задання вказівника стека
```

Команди множення 32-розрядних операндів мають вигляд

```
MULS.L (A3),D2      ; Множення зі знаком 32-розрядного даного з D2 на
                    ; 32-розрядне дане, адресоване A3; результат
                    ; записується до D2; якщо він перевищує
                    ; 32 розряди, встановлюється прапорець V = 1
                    ; до CCR
MULS.L (A3),D2:D3   ; Множення зі знаком 32-розрядного даного з D3 на
                    ; 32-розрядне дане, адресоване A3; 64-розрядний
                    ; результат записується до регістрової пари D2:D3,
                    ; старші 32 біти – до D2, молодші – до D3
```

Команди MULU та MULS, відповідно беззнакове множення та множення зі знаком, встановлюють такі прапорці:

N = 1, якщо результат від'ємний;
Z = 1, якщо результат дорівнює нулю;
V = 1, якщо переповнення є;
C = 0, завжди;
X – не встановлюється.

Команди ділення 64-розрядного операнда на 32-розрядний дільник:

```
DIVSL D1,D2:D3     ; Ділення зі знаком 64-розрядного операнда з
                   ; регістрової пари D2:D3 на 32-розрядний операнд
                   ; з D1; 64-розрядний результат записується до
                   ; регістрової пари D2:D3, старші 32 розряди – до D2,
                   ; молодші – до D3
DIVSL.L D1,D2:D3   ; Ділення зі знаком 32-розрядного операнда з D3 на
                   ; 32-розрядний операнд з D1; 32-розрядний
                   ; результат – біти 0...31 – записуються до D3, а
                   ; стача – біти 0...31 – до регістра D2
```

Команди DIVU та DIVS, відповідно беззнакове ділення та ділення зі знаком, встановлюють такі прапорці:

N = 1, якщо частка є від'ємна, та N не визначений, якщо є переповнення або ділення на нуль;

Z = 1, якщо частка дорівнює нулю;

V = 1, у разі переповнення;

C = 0, завжди;

X – не змінюється.

Команди бітових операцій наводяться у табл. 12.1. Якщо операнд розміщено у регістрі даних, він трактується як довге слово, якщо у комірці пам'яті, – як байт. Номер тестованого біта задається вмістом регістра Dn або безпосереднім операндом Nb. Значення Z = 1 встановлюється, якщо тестований біт $b_n = 0$, і Z = 0, якщо $b_n = 1$.

Команда BTST зберігає значення тестованого біта незмінним, **команда BSET** після тестування встановлює значення $b_n = 1$, а **команда BCLR** – значення $b_n = 0$. **Команда BCHG** інвертує значення біта b_n після тестування.

Зазначені команди можуть виконуватися на МП MC68000 та MC68020.

У МП68020 введено також нову групу команд, яка виконує операції з бітовими полями, довжиною до 32-х бітів. Для обрання бітового поля <bf> у команді зазначається ефективна адреса EA операнда, зміщення Of, яке визначає номер молодшого (першого) біта в полі <bf>, а також розмір поля Wf (кількість бітів у полі). При виконанні кожної з цих команд у регістрі CCR встановлюються ознаки N, Z, які схарактеризовують вміст зазначеного поля <bf>: ознака N набирає значення старшого біта поля, ознака Z = 1 встановлюється, якщо усі біти поля мають значення 0. Ознаки C та V обнулюються, X не змінюється. Після встановлення ознак **команда BFCHG** інвертує значення бітів у полі <bf>, **команда BFCLR** заповнює поле нулями, **команда BFSET** – одиницями. **Команда BFTST** тільки встановлює ознаки і не змінює значення бітів. **Команди BFEXTS** та **BFEXTU** завантажують обране бітове поле до регістра Dn з розширенням його до 32-х розрядів знаком або нулями. **Команда BFINS** виконує обернену процедуру і пересилає Wf

молодших розрядів з регістра Dn до зазначеного бітового поля. **Команда BFFFO** визначає Nb – номер першого біта у полі <bf>, який має значення 1, і заносить цей номер до регістра Dn.

Таблиця 12.1 – Команди бітових операцій

Синтаксис Асемблера	Розрядність	Операції	Адресування
BTST Dn,<EA>	B, L	$\overline{b}_n \rightarrow Z$	Dn – регістрове, операнда – усі види
BTST #Nb, <EA>	B, L	$\overline{b}_n \rightarrow Z$	Nb – безпосереднє, операнда – усі види
BSET Dn, <EA>	B, L	$\overline{b}_n \rightarrow Z$, $1 \rightarrow b_n$	Dn – регістрове, операнда – усі види
BSET #Nb, <EA>	B, L	$\overline{b}_n \rightarrow Z$, $1 \rightarrow b_n$	Nb – безпосереднє, операнда – регістрове, непряме регістрове, відносне, відносне з індексуванням
BCLR Dn, <EA>	B, L	$\overline{b}_n \rightarrow Z$, $0 \rightarrow b_n$	Dn – регістрове, операнда – регістрове, непряме регістрове, відносне, відносне з індексуванням
BCLR #Nb, <EA>	B, L	$\overline{b}_n \rightarrow Z$, $0 \rightarrow b_n$	Nb – безпосереднє, операнда – регістрове, непряме регістрове, відносне, відносне з індексуванням
BCHG Dn, <EA>	B, L	$\overline{b}_n \rightarrow Z$, $\overline{b}_n \rightarrow b_n$	Dn – регістрове, операнда – регістрове, непряме регістрове, відносне, відносне з індексуванням
BCHG #Nb, <EA>	B, L	$\overline{b}_n \rightarrow Z$, $\overline{b}_n \rightarrow b_n$	Nb – безпосереднє, операнда – регістрове, непряме регістрове, відносне з індексуванням

Значення зміщення й розміру поля {Of:Wf} можуть задаватися у команді безпосередньо числами або зазначатися вмістом регістрів даних як Of та Wf, наприклад: {D1:D2} або {10:D3}. Значення Wf повинні перебувати у діапазоні 0…31, а Wf = 0 визначає розмір поля у 32 біти. Значення Of, яке задається безпосередньо, також повинне бути у діапазоні 0…31. Якщо для віднайдення Of зазначається регістр даних, то його вміст трактується як число зі знаком.

Приклади команд бітових операцій:

```
BFCHG          D1{#6: #3}
```

Команда інвертує 3 біти числа, яке вміщено до регістра D1, розпочинаючи з 6-го біта ліворуч.

```
MOVEA.L     #400700,A4    ; Завантаження адреси до регістра A4
   CLR.L    (A4)          ; Обнулення комірки пам‘яті за EA = (A4)
   MOVEQ    #10,D0        ; Завантаження номера першого
                          ; перетворюваного біта до D0
   MOVEQ    #15,D1        ; Завантаження кількості перетворюваних
                          ; бітів до D1 (F)
   BFCHG    (A4) {D0: D1}; Інвертування бітів
```

До виконання команди BFCHG комірку пам‘яті було обнулено. Після виконання цієї команди у комірці буде число

0 0000 0011 1111 1111 1111 1000 0000 B=$003FFF00

BFCLR ([$400700]){#2: #10}

Команда обнулить поле у 16 бітів довжиною, розпочинаючи зі зміщення 2 біти у довгому слові з адресою $400700.

BFEXTS ($400600,A0){D0: D1},D2

З комірки пам‘яті за адресою ($400600,A0) бітове поле, розпочинаючи зі зміщення, зазначеного у D0, і довжиною у кількість бітів, зазначену у D1, завантажується до регістра D2 і поширює знак до 32-х бітів.

BFEXTU D1{#0: #1},D2

Команда пересилає з регістра D1 один біт, розпочинаючи зі зміщення 0 до регістру D2 і заповнює нулями інші розряди.

BFINS D2,D3{#2:#$10}

Команда пересилає $10 молодших розрядів з регістра D2 у задане бітове поле регістра D3, розпочинаючи з другого розряду ліворуч. Якщо регістр D2 вміщує число $FFFFFFFF, а регістр D3 – усі нулі, то після виконання команди регістр D2 вміщуватиме число $FFFFFFFF, а регістр D3 – число $3FFFC000.

BFSET D1{D2:D3}

Команда встановлює одиниці у бітах числа, яке вміщується у D1, розпочинаючи зі зміщення (D2) у кількості (D3).

BFTST D0{$10:#4}

Команда тестує біти з b4 по b7 операнда довжиною у слово в регістрі D0. Якщо всі біти обнулено, у регістрі CCR встановлюється ознака Z = 1, якщо b7 дорівнює 1, встановлюється ознака N у регістрі CCR.

12.2.3 Команди логічних операцій

```
AND.B D1,D3          ; Операція логічного ТА над молодшими
                     ; байтами операндів , розташованих у D1
                     ; та D3; результат записується до D3
ANDI #$FBFB,SR       ; Задання маски переривань відповідно до
                     ; рівня 4
ORI.B #4,CCR         ; Встановлення біта Z = 1 у регістрі
                     ; прапорців CCR
EOR.L DO,D2          ; Виконання операції виключного АБО над
                     ; довгими словами, які зберігаються у D2 та
                     ; D0
EORI.B #$55,$400700.L ; Інвертування парних бітів у байті, який
                     ; зберігається у комірці пам'яті з адресою
                     ; $400700
```

Прапорець N у регістрі CCR при виконанні логічних операцій встановлюється залежно від знаку результату. Прапорець Z = 1, якщо результат є нульовий . Прапорці V та C завжди встановлюються такими, що дорівнюють нулю. Прапорець X не змінюється.

12.2.4 Команди зсувів

Команди арифметичних та логічних зсувів зреалізовуються за схемами (рис. 12.3...12.5):

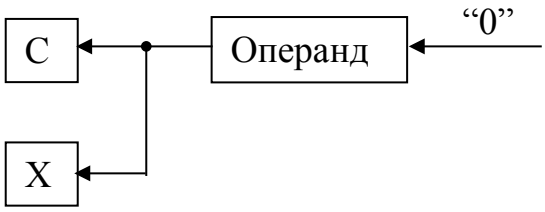


Рисунок 12.3 – Арифметичний та логічний зсуви ліворуч (**команди ASL, LSL**)

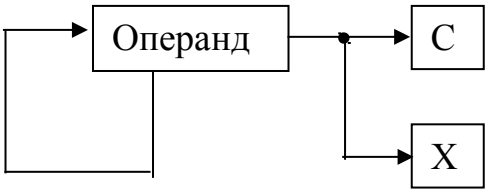


Рисунок 12.4 – Арифметичний зсув праворуч (**команда ASR**)

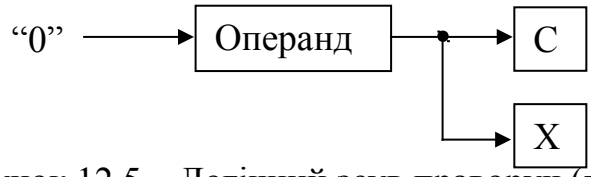


Рисунок 12.5 – Логічний зсув праворуч (**команда LSR**)

Команди циклічних зсувів зреалізовуються за схемами (рис. 12.6...12.9):

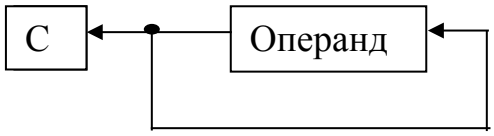


Рисунок 12.6 – Циклічний зсув ліворуч (**команда ROL**)

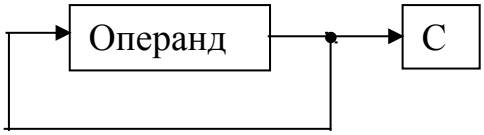


Рисунок 12.7 – Циклічний зсув праворуч (**команда ROR**)

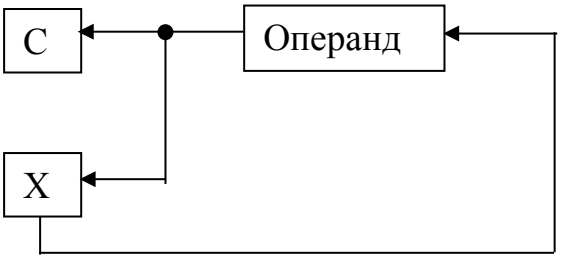


Рисунок 12.8 – Циклічний зсув ліворуч через прапорець X (**команда ROXL**)

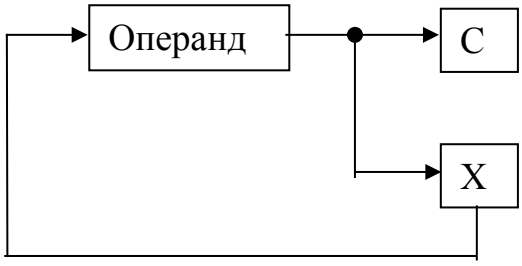


Рисунок 12.9 – Циклічний зсув праворуч через прапорець X (**команда ROXR**)

Кількість розрядів, на яку зреалізовується зсув, записується або безпосередньо у команді (1...8) або у регістрі даних, лічильнику зсувів. Операнд, який записано у пам’яті, можна зсувати тільки на 1 біт і його розмір має бути не більше за слово.

```
ASR #3,D4          ; Арифметичний зсув молодшого слова у регістрі
                   ; D4 праворуч на 3 розряди
ASL.L D0,D6        ; Зсув ліворуч довгого слова у регістрі D6 на
                   ; кількість розрядів, зазначену у D0
```

```
ASR (-2,A3,D2,L)        ; Зсув праворуч на 1 розряд слова, яке
                        ; зберігається у комірках пам'яті, ефективну
                        ; адресу першої зазначено за типом "непряме
                        ; регістрове адресування зі зміщенням та
                        ; індексуванням" та наступною
```

За арифметичного та логічного зсувів ліворуч за допомогою команд ASL та LSL молодші розряди операнда, які звільнюються, заповнюються нулями. За арифметичного зсуву праворуч за допомогою команди ASR старші розряди операнда, які звільнюються, заповнюються значенням його старшого (знакового) розряду. Це дозволяє зберегти значення знаку та форми подання операнда (двійковий код або доповняльний код). За логічного зсуву праворуч за командою LSR старші розряди операнда заповнюються нулями. При виконанні команди ASL значення V = 1 встановлюється, якщо відбулося хоча б одне змінення знакового розряду за кількох зсувів. За виконання команди ASR старший розряд не змінюється. За виконання усіх видів циклічних зсувів останній розряд операнда, що він висувається, зберігається як прапорець C.

12.2.5 Команди безумовних переходів

Команди безумовних переходів (БП) дозволяють перейти у програмі на команду із заданою адресою без запам'ятовування адреси повернення. Команди безумовних переходів зреалізовуються у МП MC68000 з коротким та довгим зміщеннями та різними типами адресування. Короткі зміщення становлять 2^8 адрес, а довгі – 2^{16} адрес. **Команда безумовного переходу JMP** завантажує до програмного лічильника PC ефективну адресу EA, яка подається за типами адресування: прямим, непрямим регістровим зі зміщенням, з індексуванням, відносним та відносним з індексуванням. Ця ефективна адреса і є адресою команди, якій передається керування програмою.

Команда безумовного переходу BRA відрізняється від JMP способом формування нового вмісту програмного лічильника. Він формується як сума вмісту програмного лічильника PC (адреса першого байта наступної команди програми) та зміщенням Ds, яке задається у команді. Величина Ds, яка вміщує до 16-ти розрядів, задається числом зі знаком, тому нове значення PC може бути більше чи менше за його поточне значення. Якщо вміст Ds становить не більше за 8 розрядів (коротке зміщення), то він розміщується у молодшому байті першого слова команди. Якщо ж він становить 16 розрядів (довге зміщення), то задається окремим словом команди. Тому команди з коротким зміщенням займають менший обсяг пам'яті й виконуються швидше, ніж такі ж самі команди з довгим зміщенням.

Нижче наводяться приклади деяких команд безумовних переходів з різним типом адресування для МП MC68000.

Команди безумовного переходу

```
$107008          JMP $1234
```

та

$107008 BRA.L $00107030

подано відповідно з коротким та довгим адресуванням.

Команди

$107030 JMP *+$10

та

$107030 BRA.S *+$10

виконуються аналогічно, адреса переходу зазначається відносно першого байта команди, наступної за командою безумовного переходу.

Мова Асемблера МП MC68020 дозволяє використовувати команди з коротким зміщенням до 2^{16} адрес та довгим – до 2^{32} адрес.

Команди

$400620 JMP $400634.L

та

$400620 JMP $500020.L

здійснюють відповідно короткий та довгий безумовний переходи.

Аналогічно будуть виконуватись команди

$400606 BRA.B *+$10,

$400606 BRA *+$10,

$400606 BRA.L *+$10

та

$ 400606 BRA.L $600000

Команди

JMP (*+ $10,PC)

та

JMP (*+ $100,PC)

використовують відносне адресування, а команди

JMP ($400610,PC,A2.W)

JMP (*+$10,PC,A2.W)

JMP ($8,PC,A2.W)

JMP (*+$1000,PC,A2.W)

JMP ($400610,ZPC,A2.W)

JMP ($400610,A2.W)

використовують непряме регістрове адресування з індексуванням.

12.2.6 Команди умовних переходів

Команди умовних переходів (розгалужень) B_{cc} мають 14 варіантів, які відрізняються умовами. Якщо зазначена в команді умова виконується, то програма переходить на команду, адреса якої формується стосовно вмісту програмного лічильника PC, як у **команді BRA;** Якщо – ні, то виконується наступна за командою умовного переходу команда. Як умови розгалуження використовуються 14 різних значень ознак N, Z, C, V та їхніх комбінацій. У багатьох випадках розгалуження програм виконується залежно від результату порівняння двох операндів за допомогою команди порівняння знакових та беззнакових чисел: **CMP, CMPA, CMPI, CMPM, TST та TAS**.

Порівняння операндів відбувається за їхнього віднімання згідно з табл. 12.2, внаслідок чого встановлюються ознаки N, Z, V, C. Сам результат не зберігається і значення операндів не змінюються.

Окремі команди дозволяють порівнювати операнд, що адресується EA, із вмістом регістра даних (**команда CMP**), регістра адреси (**команда CMPA**), безпосереднім операндом (**команда CMPI**). **Команда CMPM** використовується для порівняння розташованих у пам'яті елементів двох масивів операндів, **команди тестування TST та TAS** є одноопераднними варіантами команд порівняння.

Таблиця 12.2 – Команди порівняння та тестування

Синтаксис Асемблера	Розрядність	Операції	Адресування
CMP <EA>,Dn	B, W, L	Dn – <scr>	<scr> – усі, Dn – регістрове
CMPA <EA>,An	W, L	An – <scr>	<scr> – усі, An – регістрове
CMPI #Im,<EA>	B, W, L	<dst> – Im	Im – безпосереднє, <EA> – регістрове, непряме регістрове з усіма модифікаціями або пряме, коротке чи довге
CMPM (Ay)+,(Ax)+	B, W, L	<dst> – <scr>	Обидва операнди подаються з постінкрементуванням
TST <EA>	B, W, L	<dst> – 0	Регістрове або непряме регістрове з усіма модифікаціями
TAS <EA>	8	<dst>–0;1→b7	Регістрове або непряме регістрове з усіма модифікаціями

При виконанні цих команд встановлюються ознаки N, Z, відповідно зі знаком і значенням (дорівнює чи не дорівнює 0) операнда, що адресується EA. **Команда TAS** після тестування встановлює у 1 старший біт операнда b7. Виконання цієї команди не можна перервати запитанням прямого доступу до пам'яті. **Команда TAS** використовується у мультипроцесорних системах задля встановлення семафора – спеціального біта, який дозволяє чи забороняє різним МП доступ до окремих блоків спільної пам'яті. **Команда CMP2** є властива тільки МП MC68020, вона перевіряє перебування операнда у регістрі даних або адреси в регістрі адреси у зазначених в команді межах. Нижня границя LB обирається з комірки з адресою EA, верхня UB – з наступної. Якщо операнд дорівнює LB або UB, то Z = 1, якщо не дорівнює, то Z = 0. Якщо операнд перебуває у заданих межах, C = 0, якщо виходить, то C = 1.

Рівність операндів визначається залежно від значення ознаки Z: умови EQ та NE. Якщо порівнюються числа без знаку, то їхні відносні значення: вище (>), вище або дорівнює (>=), нижче (<), нижче або дорівнює (<=) – визначаються

відповідно з умовами HI, HS, LO, LS. Як мнемокоди умов “вище або дорівнює” (HS) та нижче (LO) можна використовувати відповідно CC та CS.

Якщо порівнюються числа зі знаком, їхні відносні значення визначаються умовами GT, GE, LT, LE. Решта умов визначається знаком результату (PL, MI) та наявністю або відсутністю переповнення (VS, VC). Слід зауважити, що процесор MC680x0 не встановлює прапорець парності чи непарності кількості одиниць у результаті, який достатньо широко використовується, наприклад, при перевірці результату в обчислювальній техніці. Тому нижче наводиться підпрограма штучного формування цієї ознаки.

У запис команд умовного переходу мовою Асемблер мнемокод відповідної умови вводиться замість символу cc. Наприклад, B + EQ = BEQ – мнемокод команди розгалуження, якщо операнди дорівнюють один одному; B + MI = BMI – мнемокод команди розгалуження за від‘ємного результату. Види умов, які використовуються у командах розгалужень, наводяться в табл. 12.3.

Таблиця 12.3 – Види умов

Символи мови	Умова, що перевіряється	Значення ознак
NE	Не дорівнює (ненульовий результат)	Z = 0
EQ	Дорівнює (нульовий результат)	Z = 1
HI	Вище	C + N = 0
LS	Нижче або дорівнює	C + N = 1
HS (або CC)	Вище або дорівнює (перенесення немає)	C = 0
LO (або CS)	Нижче (перенесення є)	C = 1
GE	Більше або дорівнює	N⊕V = 0
LT	Менше	N⊕V = 1
PL	Результат додатний	N = 0
MI	Результат від’ємний	N = 1
GT	Більше	Z + (N⊕V) = 0
LE	Менше або дорівнює	Z + (N⊕V) = 1
VC	Переповнення немає	V = 0
VS	Переповнення є	V = 1
T	Розгалуження є	1
F	Розгалуження немає	0

Умови T та F використовуються у складі **команд DBF** – безумовне виконання заданої кількості циклів та **DBT** – безумовний вихід з циклу.

12.2.7 Команди організації програмних циклів

Для організації циклів використовується **команда DB_{cc}**. Зазначений у команді регістр Dn є лічильником циклів у цьому циклі. При виконанні команди DB_{cc} спочатку перевіряється виконання умови, заданої у команді. Якщо умова виконується, то МП обирає наступну команду програми (умовний вихід з циклу). Якщо ж умова не виконується, то вміст регістра Dn декрементується. Якщо при цьому вміст регістра Dn становить –1, то також обирається наступна команда (цикл завершується). Якщо ж вміст Dn не дорівнює –1, то виконується перехід до команди з адресою (PC + Ds), яка є початком циклу. **Команда DB_{cc}** припускає використовування будь-яких умов, зазначених у табл. 12.3. **Команда організації циклу DBF** зреалізовує безумовне виконання заданої кількості циклів, а **команда DBT** – безумовний вихід з циклу.

12.2.8 Команди звернення до підпрограм

Команда виклику підпрограм JSR завантажує у PC з комірки пам’яті, адресу якої зазначено в команді, адресу першої команди підпрограми. Перед цим поточний вміст PC (адреса наступної команди) запам’ятовується у стеку. Виклик підпрограми виконується також **командою BSR**, яка використовує відносне адресування, аналогічно до **команди BRA**. Повернення до програми, яка викликає, після завершення підпрограми здійснюється **командою RTS**, яка повертає зі стека поточне значення PC, чим забезпечує перехід до виконання наступної команди програми. Як правило, перша команда підпрограми завантажує до стека поточне значення регістра SR з метою збереження ознак останнього результату. У цих випадках слід при поверненні з підпрограми використовувати **команду RTR**, яка відновлює поточне значення байта CCR (ознаки X, N, Z, V, C) та вміст лічильника команд PC.

Команди передавання керування наведено у табл. 12.4.

До команд керування також належать команди організації переривань:

TRAP (TRAP) – команда звернення до підпрограми обслуговування виключень. Команда завантажує до стека супервізора поточний вміст регістрів SR і PC, а потім завантажує до PC початкову адресу (вектор) підпрограми обслуговування відповідного виключення, яке відповідає числу $\#D_t = 0...15$, що входить до команди;

TRAPV (TRAP ON OVERFLOW) – команда виконується аналогічно до команди TRAP за умови встановлення ознаки переповнювання V = 1 і викликає виключення переповнювання;

ILLEGAL (TAKE ILLEGAL INSTRUCTION TRAP) – команда включає відповідне виключення при надходженні помилкового коду команди;

RTE (RETURN FROM EXCEPTION) – повернення з підпрограми обслуговування виключень. Команда є привілейованою і може виконуватися лише в режимі супервізора. При її виконанні відбувається поновлення вмісту програмного лічильника і регістра SR зі стека

Таблиця 12.4 – Команди передавання управління

Синтаксис Асемблера	Операція	Адресування
JMP <EA>	<dst> → PC	Непряме регістрове (усі види), пряме (коротке та довге), відносне, відносне з індексуванням
BRA ds	PC + ds → PC	Відносне
JSR <EA>	SP – 4 → SP, PC → (SP), <dst> → PC	Непряме регістрове (усі види), пряме (коротке та довге), відносне, відносне з індексуванням
BSR ds	SP – 4 → SP, PC → (SP), PC + ds → PC	
RTS	(SP) → SP, SP + 4 → SP	
RTR	(SP) → CCR, SP + Z → SP, (SP) → PC, SP + 4 → SP	
Bcc ds	Якщо (cc) виконується, то PC + ds → PC	
DBcc ds	Якщо (cc) не виконується, то Dn–1→Dn; якщо Dn ≠ -1, то PC + ds → PC	

і спеціальні команди:

NOP (NOP OPERATION) – команда здійснює перехід до наступної команди без виконання будь-яких операцій;

STOP (LOAD STATUS REGISTER AND STOP) – є привілейованою командою і виконується в режимі супервізора. Завантажує до регістра SR слово, зазначене в команді, після чого процесор припиняє роботу;

RESET (RESET EXTERNAL DEVICES) – команда формує сигнал RESET на відповідному виводі МП; використовується для початкового встановлення підсистем МПС;

CHK (CHECK REGISTER AGAINST BOUNDS) – спеціальна команда, яка виконує порівняння вмісту відповідного регістра з межею, завданою вмістом ефективної адреси; якщо значення виходить за межу, то виконується відповідне переривання;

LINK (LINK AND ALLOCATE) – зменшує значення вказівника стека на 4, завантажує вміст регістра адреси, який зазначено в команді, до стека, завантажує значення вказівника стека до регістра адреси, збільшує значення вказівника стека на 4. Таким чином, команда завантажує нове значення вказівника стека, зберігаючи у стеку і регістрі адреси дані для повернення до вихідних значень;

UNLK (UNLINK) – команда скасовує зміни, які відбулися внаслідок виконання команди LINK.

Синтаксис команд організації переривань і спеціальних команд подано у табл. 12.5.

Таблиця 12.5 – Перелік команд організації переривань і спеціальних команд

Синтаксис Асемблера	Операція	Адресування
TRAP #D_t	(SSP) – 2 → (SSP), (SR) → SSP (SSP) – 4 → (SSP), (PC) → SSP; V_e(#D_t) → PC	
TRAPV	Якщо V_e = 1, то виконується аналогічно до TRAP, якщо V_e = 0, то команда не виконується	
ILLEGAL	Якщо код команди є помилковий, то аналогічно до TRAP	
RTE	(SP) → SR; SR + 2 → (SP); SP → PC; (SP) + 4 → SP	
NOP	(PC) + 2 → PC;	
STOP #W_S	W_S → SR і зупин програми	
RESET	Встановлення початкового стану пристроїв МПС	
CHK <EA>,D_n	Якщо D_n < 0 або D_n > <EA>, то аналогічно до TRAP	Непряме регістрове (усі види), пряме (коротке та довге), відносне, відносне з індексуванням
LINK A_n,d_S	(SP) – 4 → (SP); A_n → SP; (SP) → A_n; (SP) + d_S → (SP)	
UNLK A_n	A_n → SP; (SP) → A_n; (SP) + 4 → (SP)	

Контрольні запитання:

1 Які прапорці виставляє МП фірми Motorola при виконуванні команд пересилань?

2 Які прапорці виставляються при виконуванні арифметичних операцій?

3 Які прапорці виставляються при виконуванні логічних операцій?

4 З якою метою використовується команда CMP <src>,<dst> і в який спосіб вона виконується?

5 Наведіть приклади виконання команд ASL та LSL з операндом, який дорівнює $82 на два розряди і поясність результати.

6 З якою метою виконуються команди циклічних зсувів?

7 Поясніть, як виконується команда CMPM (A3)+,(A4)+.

8 Де вміщується адреса переходу при виконуванні команди JMP <EA>?

9 В який спосіб формується адреса нового вмісту програмного лічильника PC при виконуванні команди BRA ds?

10 Які умови можна зазначати у команді умовного переходу B_{cc}?

11 В який спосіб перевіряють команди DB_{cc} та DBF умову завершення циклу, задану в команді, яка виконує вихід з циклу?

Контрольні запитання підвищеної складності:

1 Які дії МП MC68xxx спричинює виконання команди виклику ПП JSR <EA>?

2 Які дії МП MC68xxx спричинює виконання команди повернення з ПП RTS?

3 Яка команда перевіряє потрапляння заданого операнда до зазначеного в ній діапазону значень і як вона працює?

4 Які команди виконують операції з бітовими полями і в який спосіб вони виконуються?

12.3 Побудова програм з різною структурою мовою Асемблер МП фірми Motorola

12.3.1 Лінійні програми

Вхідний контроль:

1 В який спосіб будуть розміщуватись у пам’яті байти команди мовою Асемблер-86 MOV AX,7000H, якщо команду розташовано, розпочинаючи з адреси 7000:0100?

2 Команда з якою адресою буде виконуватись наступною у лінійній програмі?

3 Яка адреса вміщується у вказівнику команд IP МП фірми Intel на лінійних ділянках програми?

За приклад побудови лінійної програми розглянемо ділення 16-розрядного числа $5679 зі знаком на 16-розрядне число $0004.

```
MOVE.L #$00005679,D2
MOVE.L #$00000004,D1
DIVS D1,D2
NOP
```

Результатом виконання фрагмента буде частка від ділення (розряди 0...15), яка дорівнює $159E, вона буде розміщена у D2 (розряди 0...15), і стача $0001 (розряди 16...31), яку буде розміщено також у регістрі D2.

Зробимо перевірку правильності здобутого результату за допомогою фрагмента програми

```
EOR.L D3,D3
MOVE.L D2,D3
MULS D1,D2
SWAP D3
ADD D3,D2
```

Результатом виконання фрагмента буде наявність у регістрі D2 числа $5685, вміст регістра D1 не зміниться.

Контрольні запитання:

1 Яку частку програм, на Ваш погляд, займають лінійні ділянки?

2 В який спосіб будуть розміщуватись у пам'яті байти команди мовою Асемблер МП M680X0 MOVE #$1234,D0, якщо команда розташована розпочинаючи з адреси $400600?

3 Команда з якою адресою буде виконуватись наступною у лінійній програмі?

4 Яка адреса вміщується у лічильнику команд РС МП фірми Motorola на лінійних ділянках програми?

Контрольні запитання підвищеної складності:

1 Віднайти добуток даних $1234 та $2 усіма відомими Вам способами: написати фрагменти програми, які їх зреалізовують.

2 Віднайти частку від ділення здобутого у попередньому завданні результату на $2 усіма відомими Вам способами: написати фрагменти програми, які їх зреалізовують.

12.3.2 Розгалужені та циклічні програми. Підпрограми

Вхідний контроль:

1 Наведіть приклад застосування арифметичного циклу.

2 Наведіть приклад застосування ітераційного циклу.

3 За яким принципом зреалізовуються підпрограми часової затримки?

Приклад 12.3.1 Написати фрагмент програми, який здійснював би часову затримку на термін, визначуваний найбільшим числом, котре можна трактувати як байт.

```
400600 MOVE.B #$FF,D6   ; Затримка здійснюється за рахунок
400602 SUB #1,D6        ; повторення у циклі команди віднімання 1 з
400604 BNE *-2          ; лічильника D6; цикли повторюються, доки
400608 NOP              ; його вміст не дорівнюватиме нулю
```

Фрагмент програми, який зреалізовує затримку, може бути оформлено у вигляді підпрограми.

Приклад 12.3.2 Написати фрагмент програми, який виводить слово $1234 до додаткового регістра PAAR PI/T, а через час, визначуваний вмістом D2 у підпрограмі TIME, слово $5678 до того самого регістра PAAR PI/T.

```
        MOVE,B #$1234,D0     ; Завантаження даного $1234 до регістра D0
        MOVE,L #$800015,A0   ; Завантаження адреси регістра
                             ; PAAR PI/T у A0
        MOVEP.B D0,(A0)      ; Запис даного $1234 до регістра PAAR
        JSR TIME             ; Звернення до підпрограми TIME
        MOVE.B #$5678,D1     ; Завантаження даного $5678 до регістра D1
        MOVEP.B D0,(A0)      ; Запис даного до регістра PAAR PI/T
TIME: MOVE #$AB,D2           ; Підпрограма
  M2 : SUB #1,D2             ; TIME
        BNE M2               ;
        RTS                  ;
```

Приклад 12.3.3 Написати підпрограму визначення парності чи непарності кількості одиниць у байті, який міститься в регістрі.

Задача розв’язується шляхом логічного зсуву байта у циклі, наприклад праворуч, та підрахування кількості разів, коли встановлювався прапорець перенесення C. Структурну схему алгоритму підраховування кількості одиниць у байті зображено на рис. 12.10.

Програма розв’язання задачі на Асемблері МП MC68020:

```
EVEN:  MOVE    SR,D5      ; Завантаження регістра стану до D5
       CLR.L   D2         ; Обнулення D2, лічильника суми
       MOVE.L  #$7,D3     ; Організація лічильника циклів у D3
       MOVE.L  #$09,D0    ; Завантаження байта $9 до регістра D0
M1:    LSR.B   #$1,D0     ; Зсув праворуч регістра D0 на один розряд
       BCS.B   M2         ; Перехід до лічильника суми
       BRA.B   M3         ; Обхід підсумовування
M2:    ADDI    #$1,D2     ; Додавання 1, якщо C = 1
M3:    DBF     D3,M1      ; Організація повторення циклів
       BTST    #$0,D2     ; Перевірка парності кількості
                          ; одиниць суми у D2
       BNE.B   M4         ; Число непарне?
       CLR.L   D2         ; Ні, обнулення D2
       BRA.B   M5         ; Обхід запису числа FDH до D2
M4:    MOVE.L  #$FD,D2    ; Так, запис до D2 числа $FD
M5:    MOVE    D5,SR      ; Відновлення регістра стану з D5
       RTS                ; Повернення з підпрограми
```

Якщо кількість одиниць є непарна, то до регістра D2 записується довільна позначка $FD, а якщо ж парна, – то регістр D2 обнулюється.

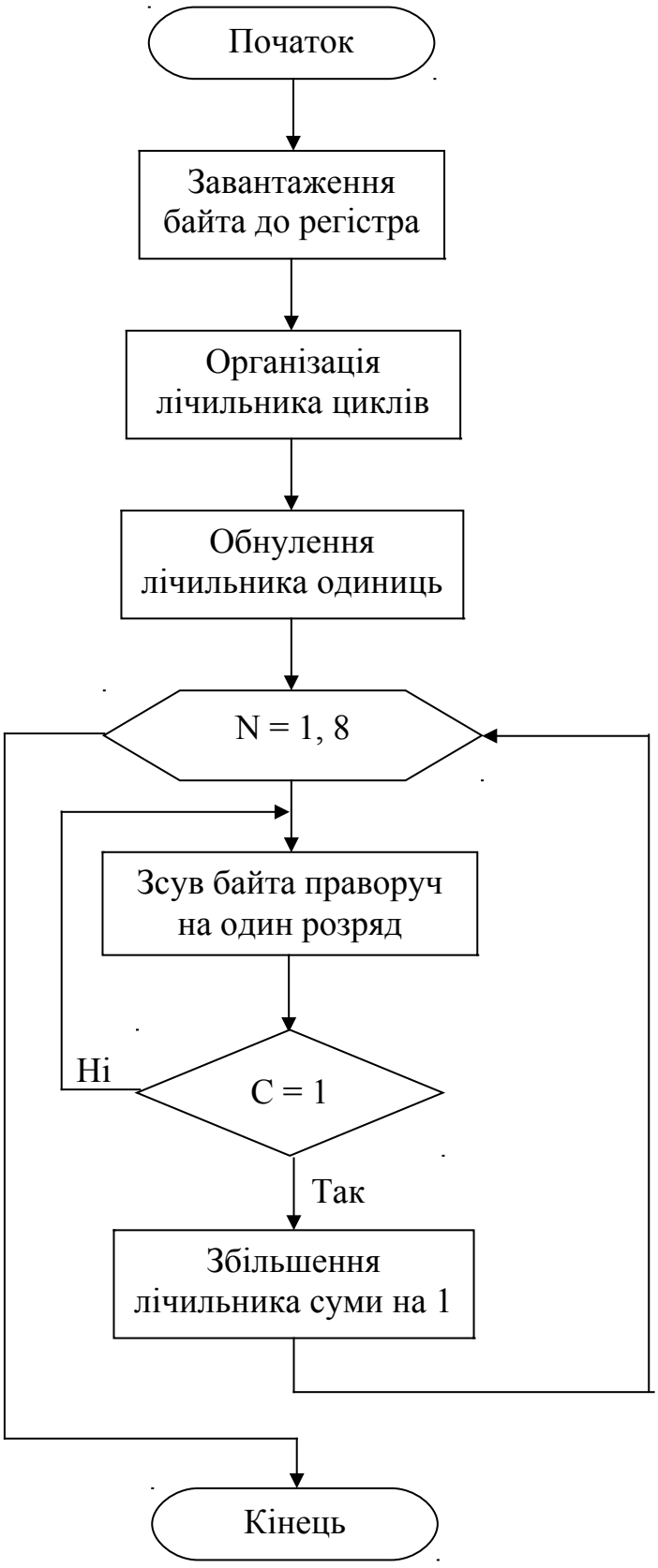


Рисунок 12.10 – Структурна схема алгоритму підраховування кількості одиниць у байті

Приклад 12.3.4 Порівняти значення байтів у масивах (A0) та (A1), які складаються з чисел зі знаками у межах шести пар. За умовою DBGT зорганізувати вихід з циклу, зазначити, скільки циклів буде виконано.

	(A0)	(A1)
400600 MOVEA.L #$400700,A0	5F	40
400606 MOVEA.L #$400800,A1	3F	F4 (-12)
40060C MOVE #$5,D0	(-1)FF	F0 (-16)
400610 CMPM.B (A0)+,(A1)+	2F	F0 (-16)
400612 DBGT D0,$400610	(-32)E0	A3 (-93)
400616 NOP	(-16)F0	AF (-81)

Оскільки елементи масивів трактуються як числа зі знаками, то від'ємні числа подано у доповняльному коді (поруч ці самі числа подано зі знаком у десятковій системі числення). Враховуючи особливості виконання команди CMPM (A0)+, (A1)+ (від елемента, який адресується A1, віднімається елемент, адресований A0), можна свідчити, що вихід з циклу відбудеться за наявністю у лічильнику циклів числа -1, за окресленою умовою DBGT цикл не завершується в межах порівняння шести пар елементів. Кількість циклів становить 6.

Контрольні запитання:

1 Напишіть підпрограму визначення або непарності кількості одиниць у байті, який міститься у регістрі (приклад 12.3.3) при зсуві вмісту цього регістра ліворуч; дайте пояснення здобутим результатам.

2 Поясніть, які команди у підпрограмі TIME визначають постійну складову затримки, а які – варіаційну.

Контрольні запитання підвищеної складності:

1 Визначити час затримки, який зреалізовується фрагментом програми у прикладі 12.3.1.

2 Визначити час затримки, який зреалізовує підпрограма TIME у прикладі 12.3.2.

3 Написати підпрограму часової затримки на 2 мкс для реалізації на МП MC68000 і на МП MC68020.

12.4 Створення програмного забезпечення МПС на МП фірми Motorola

Вхідний контроль:

1 Які пристрої входять до периферії МПС?

2 В який спосіб периферійні пристрої можуть підключатись до МПС?

3 В який спосіб задаються режими роботи пристроїв введення-виведення?

4 В який спосіб програмно зреалізовується часова затримка?

МПС, яка розв’язує складні завдання керування або збирання та обробки даних, має розвинену периферію і повинна працювати під операційною системою. МП фірми Motorola мають всі необхідні структурні риси – режими супервізора та користувача, механізм підтримки сторінкового обміну даними поміж жорстким диском та оперативною пам’яттю для побудови таких МПС, у тому числі комп’ютерів. Реалізація переривань, багатозадачного режиму, оброблення виключень виконується в режимі супервізора під керуванням ОС. Коли в якості обчислювального пристрою використовується комп’ютер або багатопроцесорна система, наприклад у цифрових системах комутації, вони водночас можуть розв’язувати завдання статистики, білінгу, підготовки документів тощо. При виборі обчислювального пристрою, в тому числі мікропроцесора, треба мати на увазі, що він визначається типом розв’язуваних завдань, вимогами до продуктивності, характеристиками периферійних пристроїв. МПС може бути просторово розподіленою – датчики, вимірювальні прилади, виконавчі та керувальні пристрої, МП (обчислювальний пристрій). Мікропроцесори намагаються розташувати якомога ближче до периферійних пристроїв, якщо це є технічно можливим, і зв’язок поміж ними здійснюється за допомогою паралельних пристроїв введення-виведення для досягнення найбільшої швидкості обміну даними. Водночас до МПС можуть входити послідовні адаптери та приймачі-передавачі для зв’язку з віддаленими периферійними пристроями.

При створюванні програмного забезпечення МПС завжди слід розробляти підпрограми ініціалізації кожного з пристроїв введення-виведення на заданий режим. Головна робоча програма, яка керує МПС, незважаючи на розмаїття розв’язуваних задач, вміщує такі фрагменти:

- циклічний опит датчиків та вимірювальних приладів, перевірка готовності їх до роботи;
- введення даних з датчиків;
- оброблення даних за заданим алгоритмом;
- циклічне виведення результатів на виконавчі або керувальні пристрої;
- відображення проміжних або кінцевих результатів на моніторі чи інших пристроях.

З метою узгодження швидкостей роботи МП та периферії часто треба затримувати команди або сигнали за допомогою підпрограм затримки.

З метою економії пристроїв введення-виведення периферійні пристрої можуть підмикатись до одного чи кількох портів за допомогою комутаторів. У такому разі треба виводити у циклі головної програми номери або адреси кожного з периферійних пристроїв.

В головній програмі, як правило, є фрагменти порівняння введених параметрів зі стандартними, які зберігаються у пам’яті, результати порівняння визначають подальше виконання програми.

Нижче наведено приклади програм, які зреалізовують спрощений алгоритм роботи МПС різного призначення.

Приклад 12.4.1 Приклад демонструє можливість реалізації пристрою телекомунікацій на комп'ютері або МПС з МП М680Х0.

Скласти програмні моделі скремблера та дескремблера, які дозволяли б за допомогою генератора випадкових чисел кодувати 255 відліків цифрового сигналу, а потім їх декодувати. Надходження цифрових сигналів з лінії зв'язку у паралельному коді імітується зчитуванням однобайтових елементів масиву з пам'яті, розпочинаючи з адреси $600500. Кодовані скремблером значення відліків записуються до пам'яті, розпочинаючи з адреси $600600, потім декодуються у моделі дескремблера і первинні значення відліків для порівняння запам'ятовуються, розпочинаючи з адреси $600700. Для генерації псевдовипадкової послідовності (ПВП), як правило, використовуються кільцеві регістри зсуву з суматорами за модулем 2 у введених колах зворотного зв'язку. Не кожна конфігурація структури зворотного зв'язку забезпечує максимальний період псевдопослідовності, що дорівнює $2^N - 1$, де N – кількість розрядів регістру. На рис. 12.11 подано одну з максимальних структур з трьома колами зворотного зв'язку, які пов'язують 0, 2, 4, та 7-й розряди 8-розрядного регістру зсуву. Ця структура забезпечує максимальний період генерації числової псевдопослідовності, що дорівнює 255.

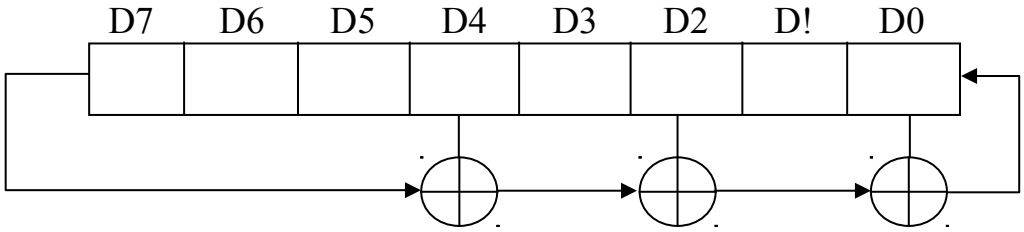


Рисунок 12.11 – Генератор псевдовипадкової послідовності

Задана структура зворотного зв'язку регістру зсуву здійснюється за допомогою маски $95. Для реалізації трьох послідовних операцій складання за модулем 2 прийнятої структури зручно використовувати ознаки парності чи непарності кількості одиниць у байті на кожному кроці зсуву, який зреалізовується програмно за допомогою команди ROXL.B – циклічного зсуву ліворуч з розширенням.

Скремблювання та дескремблювання відліків сигналів здійснюється шляхом операції додавання їх за модулем 2 до чергового псевдовипадкового числа. Вхідні та вихідні відліки сигналів при скремблюванні та дескремблюванні записуються у регістр зсуву та зчитуються з нього у паралельному коді.

Структурну схему алгоритму зображено на рис. 12.12.

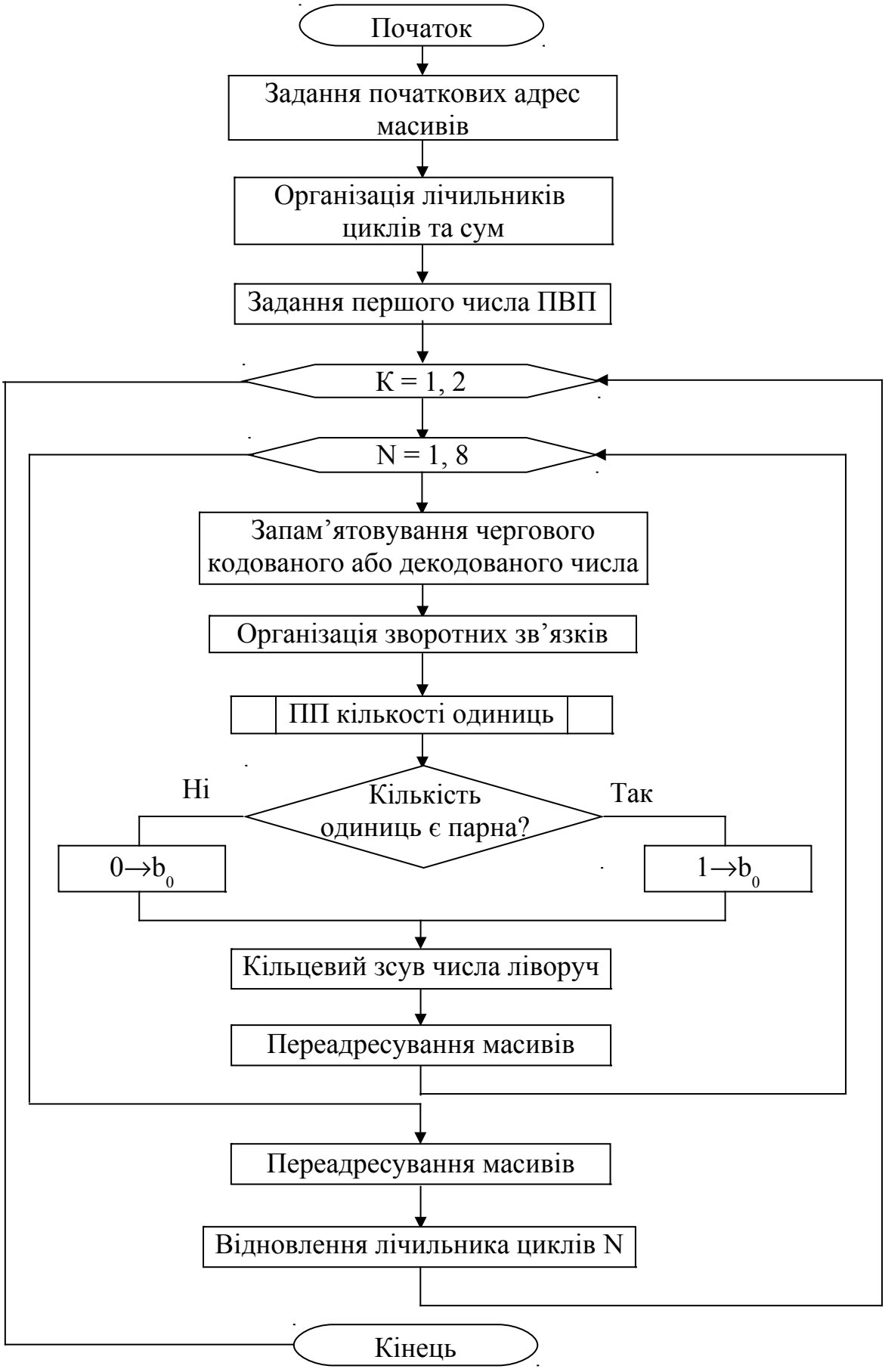


Рисунок 12.12 – Структурна схема алгоритму скремблера-дескремблера

Програма розв’язання завдання:

```
        MOVEA.L  #$400700,A0  ; Завантаження адреси масиву вхідних
                              ; сигналів до A0
        MOVEA.L  #$400800,A1  ; Завантаження адреси масиву кодованих
                              ; сигналів до A1
        MOVE.L   #$FE,D3      ; Завантаження лічильника циклів ПВП
        MOVE.L   #$1,D4       ; Завантаження лічильника циклів
                              ; скремблювання–дескремблювання
M4:     MOVE.L   #$9,D0       ; Завантаження першого числа ПВП
M1:     MOVE.L   D0,D1        ; Запам’ятовування чергового числа ПВП
                              ; у D1
        MOVE.B   (A0)+,D2     ; Скремблювання чергового елемента
        EOR.B    D0,D2        ; масиву вхідних сигналів
        MOVE.B   D2,(A1)+     ; та запам’ятовування його за адресою
                              ; (A1)+
        ANDI.B   #$95,D0      ; Організація зворотних зв’язків
        BSR      EVEN         ; Звернення до ПП визначення парності-
                              ; непарності кількості одиниць у байті
        MOVE     SR,D5        ; Ні, пересилання SR до D5
        BTST     #$0,D2       ; Кількість одиниць є
        BEQ.B    M2           ; парна?
        ANDI.B   #$FD,D5      ; Встановлення X до “0”
        BRA.L    M3           ; Обхід встановлення X до “1”
M2:     ORI.B    #$10,D5      ; Так, встановлення X до “1”
M3:     MOVE     D5,SR        ; Відновлення регістра стану
        MOVE.L   D1,D0        ; Відновлення чергового числа ПВП
        ROXL.B   #1,D0        ; Циклічний зсув D0 ліворуч через X
        DBF      D3,M1        ; Повернення до початку внутрішнього
                              ; циклу
        MOVEA.L  #$400800,A0  ; Переадресування вхідного
        MOVEA.L  #$400900,A1  ; та дескрембльованого масивів сигналів
        MOVE.L   #$FE,D3      ; Перевантаження лічильника циклів
        DBF      D4,M4        ; Повернення до початку зовнішнього
                              ; циклу
        NOP                   ; Завершення програми
```

Слід зауважити, що при зверненні до ПП визначення кількості одиниць у байті значення байта, що міститься у D0, треба передавати з головної програми, а команду MOVE.L #$9,D0 вилучити з ПП.

Результатом виконання програми є три масиви: перший, що складається з відліків вхідного сигналу, розпочинаючи з адреси $400700; другий розпочинається з адреси $400800 і складається з кодованих сигналів, а третій,

розташований за адресою $400900, складається з розкодованих відліків вхідного сигналу. Кожний з масивів складається з 255 елементів.

Приклад 12.4.2 Вимірювальна МПС оцінює якість N каналів зв'язку за P параметрами і розпочинає свою роботу після надходження із зовнішнього пристрою ПВВ1 байта дозволу B1. Через пристрій виведення ПВИВ МПС видає номер поточного каналу, а через інтервал часу T1 – номер поточного вимірюваного параметра. До шини даних підмикається відповідний вимірювальний прилад, який через пристрій введення ПВВ2 вводить до МПС значення вимірюваного параметра у вигляді байта. Якщо це значення перебуває у заданих межах, МПС переходить до вимірювання наступного параметра, якщо ж ні, – то записує до пам'яті, розпочинаючи з адреси ADDR1 номер хибного каналу, номер хибного параметра та його значення, а потім також переходить до вимірювання наступного параметра. Задані межі відхилення кожного параметра зберігаються у пам'яті, розпочинаючи з адреси ADDR2. Скласти програму, яка забезпечує роботу МПС за заданим алгоритмом.

Розподіл регістрів:
D0 – лічильник кількості каналів
D1 – лічильник кількості параметрів
D2 – лічильник кількості циклів у підпрограмі затримки DELAY
A0 – адресний регістр, до якого завантажується ADDR1
A1 – адресний регістр, до якого завантажується ADDR2
A2 – адресний регістр, до якого завантажується ADDR3 (адреса ПВВ1)
A3 – адресний регістр, до якого завантажується ADDR4 (адреса ПВИВ)
A4 – адресний регістр, до якого завантажується ADDR5 (адреса ПВВ2)
B1 – байт дозволу роботи МПС
Bi – поточний байт, який вводиться з ПВВ2

Перед виконанням програми до пам'яті, розпочинаючи з адреси ADDR2, треба послідовно записати стандартні значення кожного з вимірюваних параметрів.

```
        MOVEA.L #$ADDR1,A0      ; Завантаження ADDR1 до A0
        MOVEA.L #$ADDR3,A2      ; Завантаження адреси ПВВ1
        MOVEA.L #$ADDR4,A3      ; Завантаження адреси ПВИВ
        MOVEA.L #$ADDR5,A4      ; Завантаження адреси ПВВ2
MN:     MOVE #$N-1,D0           ; Завантаження кількості каналів N-1
MP:     MOVE #$P-1,D1           ; Завантаження кількості параметрів P-1
M5:     MOVEA.L #$ADDR2-1,A1    ; Завантаження ADDR2-1 до A1
M1:     MOVE.B (A2),D3          ; Введення чергового байта Bi з ПВВ1
        CMP #$B1,D3             ; Порівняння введеного байта з B1
        BNE M1                  ; Повернення на початок циклу
        MOVE.B D0,(A3)          ; Виведення номера каналу до ПВИВ
        JSR DELAY
        MOVE.B D1,(A3)          ; Виведення номера параметра до
```

```
                                ; ПВИВ через час T
        JMP M3
DELAY: MOVE #$K,D2              ; Завантаження до D2 числа K, яке
                                ; визначає час затримки
M2:     SUBQ #1,D2              ; Декрементування регістра D2
        BNE M2                  ; Повернення на дві адреси назад, якщо
                                ; у регістрі D2 не 0
        RTS                     ; Повернення з підпрограми DELAY
M3:     MOVE.B (A4),D3          ; Введення значення вимірюваного
                                ; параметра
        ADDA,L #1,A1            ; Отримання адреси ADDR2+2
        CMP2.B (A1)+,D3         ; Перевірка перебування введеного
                                ; параметра у межах (A1) ≤ D3 ≤ (A1+1)
        BEQ M4
        MOVE.B D0,(A0)+         ; Завантаження до пам'яті номера
                                ; хибного каналу
        MOVE.B D1,(A0)+         ; Завантаження до пам'яті номера
                                ; хибного параметра
        MOVE.B D3,(A0)+         ; Завантаження до пам'яті значення
                                ; хибного параметра
M4:     DBF D1,M3               ; Повернення на початок внутрішнього
                                ; циклу по P, якщо D1 не -1
        MOVE #$P-1,D1           ; Відновлення лічильника циклів D1
        DBF D0,M5               ; Повернення на початок зовнішнього
                                ; циклу по N, якщо D0 не -1
        JMP MN                  ; Циклування програми
```

При налагодженні програми у разі неможливості підключати пристрої введення-виведення треба імітувати надходження байтів параметрів зчитуванням їх з пам'яті як елементів масивів.

Приклад 12.4.3 Керувальна МПС послідовно перевіряє готовність до роботи N пристроїв з адресами ADDR1…ADDRN. МП видає через пристрій виведення ПВИВ адресу пристрою, який перевіряється на готовність, на комутатор. Комутатор підмикає пристрій до пристрою введення ПВВ, і МП може зчитати байт з виходу пристрою; якщо він дорівнює еталонному байтові $AA, то пристрій є готовий до роботи. Адреси готових до роботи пристроїв запам'ятовуються у послідовних комірках пам'яті, розпочинаючи з адреси ADDRN+1. Якщо в перебігу першого опитування виявляється, що жоден пристрій не є готовий до роботи, через час T опитування повторюється і в разі неготовності жодного пристрою МПС завершує роботу.

```
        MOVE.L #$ADDR1,A0       ; Завантаження адреси 1 пристрою до A0
        MOVE #$N-1,D0           ; Організація лічильника циклів за
                                ; кількістю пристроїв
```

```
        MOVE #$1,D1             ; Організація лічильника циклів за
                                ; кількістю перевірок
        EOR D3,D3               ; Обнулення регістра D3 – лічильника
                                ; кількості готових пристроїв
        MOVEA.L #$ADDRO,A1      ; Завантаження адреси пристрою
                                ; виведення ПВИВ до A1
        MOVEA. L #$ADDRI,A2     ; Завантаження адреси пристрою ПВВ
                                ; до A2
        MOVEA. L #$ADDRR,A3     ; Завантаження початкової адреси
                                ; масиву адрес готових до роботи
                                ; пристроїв
M1:     MOVEA.L A0,(A1)         ; Виведення адреси і-го пристрою через
                                ; порт виведення ПВИВ
        ADDQ #1,A0              ; Нарощування адреси пристроїв
        MOVE.B (A2),D2          ; Введення байта даних з пристрою
                                ; ПВВ
        CMP $AA,D2              ; Порівняння (D2) з байтом дозволу
                                ; $AA
        BNE M2                  ; Ні, (D2) не дорівнює $AA
        ADDI #1,D3              ; Так, (D2) дорівнює $AA,
                                ; інкрекрементування лічильника D3
        MOVEA. L A0,(A3)+       ; Завантаження адреси готового до
                                ; роботи пристрою
M2:     DBF D0,M1               ; Організація циклу за кількістю
                                ; пристроїв
        CMPI 0,D3               ; Перевірка лічильника D3 на рівність 0
        BEQ M3                  ; Всі прилади не є готові
        JMP M5
M3:     JSR DELAY               ; Звернення до підпрограми DELAY
        MOVEA.L #$ADDR1,A0      ; Відновлення адреси першого пристрою
        MOVEA. L #$ADDRR,A3     ; Відновлення початкової адреси
                                ; масиву адрес готових до роботи
                                ; пристроїв
M4:     DBF D1,M1               ; Завершення циклу за кількістю
                                ; перевірок
M5:     NOP                     ; Завершення програми
DELAY:  MOVE #$K,D4             ; Завантаження до D4 числа K, яке
                                ; визначає тривалість затримки
M6:     SUBQ #1,D4              ; Організація
        BNE M6                  ; програмної затримки
        RTS                     ; Повернення з підпрограми
```

Приклад 12.4.4 Керувальна МПС обробляє дані, які находять у циклі від N датчиків через пристрій введення ПВВ у вигляді байтів за алгоритмом $Y = \overline{x_7} \vee x_6 \vee \overline{x_4} \vee x_3 \vee x_2 \vee \overline{x_0}$. Якщо для поточного даного Y = 1, то через час T через пристрій виведення ПВИВ МПС виводить керувальний сигнал на відповідний виконавчий механізм і переходить до опитування наступного датчика. Якщо ж 0, то МПС відразу переходить до приймання наступного даного.

Задамо номери N датчиків з N-2 по -1. Ці номери МПС буде видавати на зовнішній комутатор, який підмикає і датчики, і відповідні виконавчі механізми синхронно до ПВВ та ПВИВ.

Алгоритм оброблення даного, який є заданий у вигляді логічної функції, описано у [8]. Для його реалізації треба виконати такі дії:

1 Помножити логічно дане на маску mask1, яка вміщує 1 у розрядах, які відповідають наявним у даному розрядам, і 0 – у розрядах, які є відсутні, вилучаючи їх з розгляду, і здобути результат Y1.
2 Зробити операцію виключного АБО над результатом Y1 та маскою mask2, яка буде мати 1 у розрядах, де x є інвертований, і 0 – у всієї решти, і здобути результат Y2.
3 Якщо Y2 ≠ 0, то Y = 0, і навпаки, якщо Y2 = 0, Y = 1.

Фрагмент програми

```
        MOVEA.L #$ADDR1,A1      ; Задання адреси ПВВ
        MOVEA.L #$, ADDR2,A2    ; Задання адреси ПВИВ
LOOP:   MOVE #$N-1,D0           ; Задання лічильника циклів
M1:     MOVE D0,(A2)            ; Виведення адреси N-го датчика
        MOVE (A1),D1            ; Введення даного Yi
        ANI.B #mask1,D1         ; Виключення розрядів, які є
                                ; відсутні у Yi, знаходженняY1i
        EORI.B #$mask2,D1       ; Знаходження Y2i
        BEQ M3                  ; Yi = 1
M2:     DBF D0,M1               ; Y = 0, звернення за наступним даним
        JMP LOOP                ; Циклування програми
M3:     JSR DELAY               ; Звернення до підпрограми DELAY
        MOVE D1,(A2)            ; Виведення керувального слова
        JMP M2
DELAY:  MOVE #$K,D3             ; Завантаження до D3 числа K, яке
                                ; визначає тривалість затримки
M4:     SUBQ #1,D4              ; Організація
        BNE M4                  ; програмної затримки
        RTS                     ; Повернення з підпрограми
```

Програма виконується у циклічному режимі неперервно.

Контрольні запитання:

1 Які фрагменти має вміщувати робоча програма, яка керує МПС?

2 Чи завжди МПС працює під керуванням операційної системи?

3 Які пристрої введення-виведення можуть входити до складу МПС?

4 Як називається програмний модуль, призначений керувати пристроями введення-виведення?

Контрольні запитання підвищеної складності:

1 Напишіть фрагмент програми оброблення даних, які надходять у циклі від датчика через пристрій введення у вигляді байтів за алгоритмом $Y = x_6 \wedge \overline{x_5} \wedge x_3 \wedge \overline{x_1} \wedge x_0$. Якщо для поточного даного $Y = 1$, то через час T через пристрій виведення ПВИВ МПС виводить керувальний сигнал на відповідний виконавчий механізм і переходить до опитування наступного датчика. Якщо ж 0, то МПС відразу переходить на прийом наступного даного. Кількість датчиків і значення даних задати самостійно. Програма повинна працювати в циклі.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ ДО 3-го МОДУЛЯ

1 Шагурин И. И. Микропроцессоры и микроконтроллеры фирмы Motorola: Справ. пособие. – М.: Радио и связь, 1998. – 560 с.: ил.
2 Шагурин И. И. Современные микроконтроллеры и микропроцессоры Motorola: Справ. пособие. – М.: Горячая линия – Телеком, 2004. – 952 с.: ил.
3 MC68681 DUAL ASYNCHRONOUS RECEIVER/TRANSMITTER (DUART) (INCLUDES DATA ON MC2681 AND MC2682) – Motorola Semiconductors. – September, 1985.
4 PROGRAMMER'S REFERENCE MANUAL (Includes CPU32 Insructions) M68000 Family Programmer's Reference Manual – ©MOTOROLA INC., 1992.
5 M68EC000 CPU Module Users Manual. Rev 3.0 – ©MOTOROLA, 1993.
6 M68EC0x0 Integrated Development Platform Users Manual. Rev 3.0 – ©MOTOROLA, 1993.
7 THE FLIGHT-68EC020 EVM. Users Manual by: R.F. Coates. Revised and Edited by: J.H.R. Selby. – Flight Tltctronics LTD., Southamton.
8 Проектирование импульсных и цифровых радиотехнических систем: Учеб. пособие для радиотехн. спец. вузов / Ю. П. Гришин, Ю. М. Казаринов, В. М. Катипов и др.; Под ред. Ю. М. Казаринова. – М: Высшая школа, 1985. – 340 с.: ил.

4 МОДУЛЬ
13 МІКРОПРОЦЕСОРНІ СИСТЕМИ НА МІКРОКОНТРОЛЕРАХ ФІРМИ MOTOROLA ТА ЇХНЄ ПРОГРАМУВАННЯ

Вхідний контроль:

1 Які пристрої називаються мікроконтролерами?

2 Які пристрої можуть входити до складу мікроконтролера?

3 У складі якого обладнання можуть використовуватися мікроконтролери?

Фірма Motorola є провідним виробником мікроконтролерів (МК) у світі. Вона випускає 8-розрядні мікроконтролери, 16-, 32-розрядні модульні комунікаційні мікроконтролери та 32-розрядні RISC мікроконтролери PowerPC.

8-розрядні МК є найбільш поширеними представниками мікропроцесорної техніки. Їхня вартість не є високою, а доволі широкий спектр функціональних можливостей дозволяє використовувати ці контролери у різноманітних пристроях та приладах. Усі 8-розрядні МК входять до складу трьох сімейств – 68HC05, 68HC08 та 68HC11.

МК M68HC05 є найбільш прості й, відповідно, такі, що мають обмежені функціональні можливості, але є доволі дешевими. До складу цього сімейства входять різні типи мікроконтролерів, які відрізнюються обсягом та типом пам’яті, номенклатурою периферійних пристроїв, що їх розміщено на кристалі, а також іншими експлуатаційними характеристиками.

МК сімейства 68HC08 мають більшу продуктивність, більший обсяг внутрішньої пам’яті та розширені функціональні можливості.

На кристалі МК сімейства 68HC11 зреалізовано велику номенклатуру периферійних пристроїв, вони також забезпечують можливість розширення обсягу пам’яті за рахунок підключення зовнішніх запам’ятовувальних пристроїв.

Загальна кількість різних моделей 8-розрядних МК, які сьогодні продукуються, становить близько 100.

16-розрядні МК фірми Motorola представлено двома сімействами – 68HC12/912 та 68HC16. МК 68HC12/912 мають повну архітектурну сумісність з сімейством 68HC11, в них вбудовано внутрішню Flash-пам’ять з великим обсягом – до 128 кбайт. Іхньою особливістю є виконування низки операцій, які використовуються для керування об’єктами з використанням “нечіткої логіки”. Вони мають доволі великий набір периферійного обладнання, високу продуктивність і відносно низьке енергоспоживання. МК 68HC16 є подальшим розвитком архітектури 8-розрядних мікроконтролерів 68HC11 і відрізнюється розширеним набором регістрів, доповненням системи команд і способів адресування. Їхнє використовування, у певних випадках, обмежено необхідністю використовування зовнішньої пам’яті.

Сімейство 32-розрядних МК MC68300 побудовано з використанням 32-розрядного процесора, який доповнено низкою високоефективних периферійних пристроїв, що дозволяє використовувати їх в системах керування

складними об’єктами та процесами з використанням мінімальної кількості інших компонентів. Вони використовуються у робототехніці, авіаційно-космічній техніці, у різноманітному телекомунікаційному обладнанні.

Розпочинаючи з 90-х років фірма Motorola також випускає комунікаційні 32-розрядні мікроконтролери, призначені для використовування у складі цифрових телекомунікаційних систем і забезпечують обробку даних та організацію швидкого обміну масивами даних по каналах зв’язку. До них належать мікроконтролери MC68302, MC68360, MC68356 та їхні модифікації.

13.1 Типові мікроконтролери фірми Motorola

Вхідний контроль:

1 Які пристрої входять до програмної моделі мікропроцесора M680x0?

2 Які відмінності мають фоннейманівська і гарвардська архітектури МП?

3 До якої з архітектур належать МП фірми Motorola, які вивчалися в попередніх розділах?

4 Призначення регістра прапорців (ознак результату)?

5 В який спосіб змінюється вміст регістра прапорців?

6 Стекова пам’ять та її використовування у МП.

7 Призначення регістра-лічильника команд.

13.1.1 8-розрядні мікроконтролери

Сімейство M68HC05/705

Всі моделі найбільш поширеного сімейства 8-розрядних МК M68HC05/705 мають однакове процесорне ядро CPU05 і відрізнюються за наявністю, обсягом та типом пам’яті, яка входить до їхнього складу, певними характеристиками, а також номенклатурою периферійних пристроїв, розміщених на кристалі. Загальна кількість 8-розрядних контролерів, рекомендованих сьогодні до використовування, становить близько 40 моделей.

Мікроконтролери M68HC05 та M68HC705 відрізнюються лише наявністю у складі останнього репрограмованого ПЗП з електричним стиранням – РПЗП-ЕС (EEPROM).

Загальну структурну схему МК сімейства M68HC05/705 наведено на рис. 13.1.

До складу всіх 8-розрядних контролерів входять: процесор CPU05, блок програмування, блок контролю функціонування, генератор тактової частоти і блок внутрішнього запам’ятовувального пристрою. Склад паралельних портів і периферійних пристроїв відрізнюється у МК різних типів . Блок внутрішнього запам’ятовувального пристрою складається з оперативного запам’ятовувального пристрою даних, обсягом від 32 до 920 байт (для різних моделей), постійного запам’ятовувального пристрою програм, обсягом до 32 кбайт та службового постійного запам’ятовувального пристрою, обсягом

240 байт. До складу деяких моделей входять репрограмовані ПЗП з електричним стиранням – РПЗП-ЕС (EEPROM), обсягом до 920 байт.

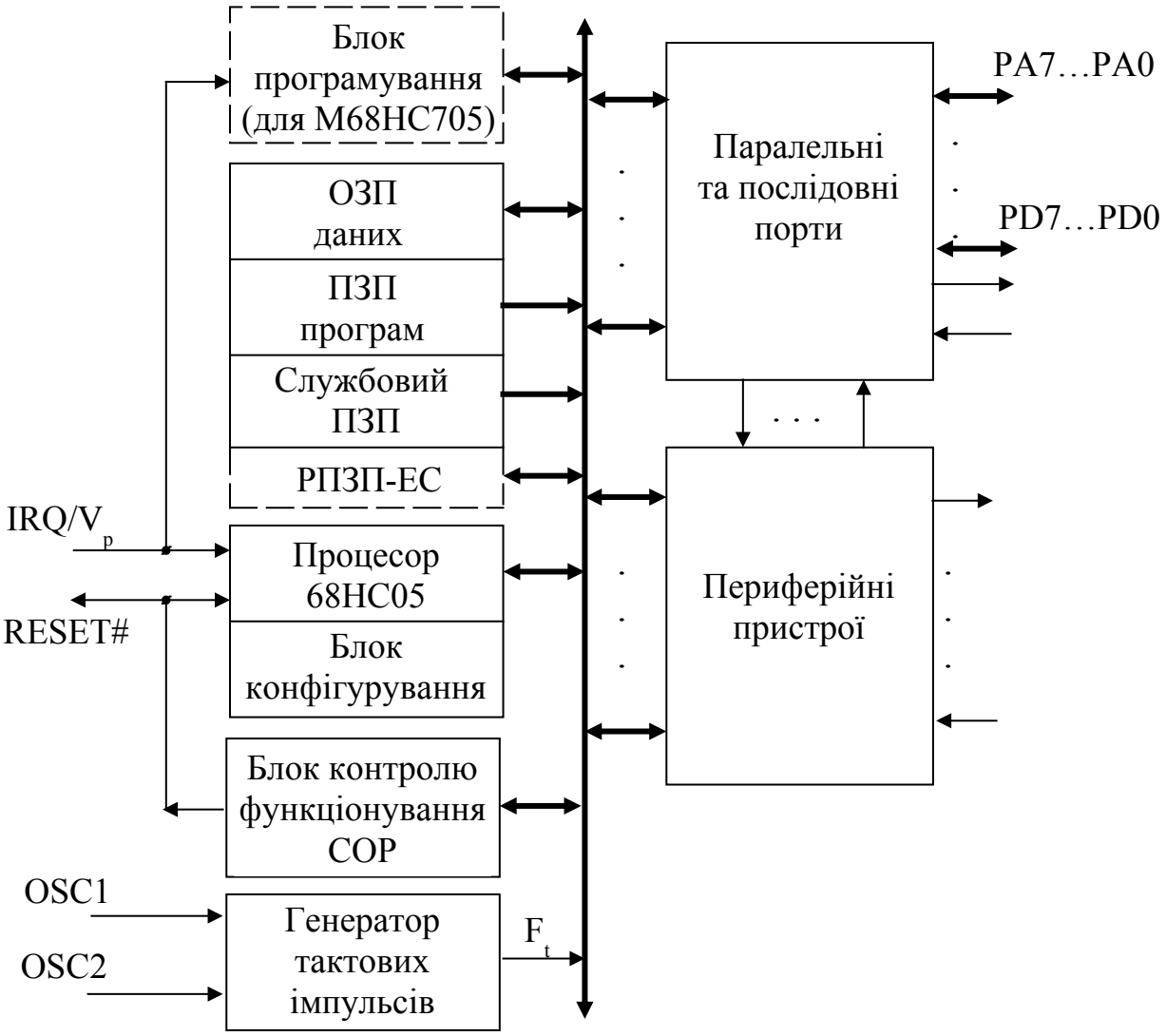


Рисунок 13.1 – Загальна структурна схема МК сімейства M68HC05/705

Генератор тактової частоти формує імпульси, частота котрих визначається кварцовим або керамічним резонатором, RC-ланцюгом або зовнішнім генератором, які підключаються до виводів OSC1, OSC2. Частота імпульсів тактової частоти МК для більшості моделей становить F_t = 2,1 МГц за напруги живлення $V_п$ = 5 В і F_t = 1,0 МГц за $V_п$ = 3,3 В. Деякі моделі мають підвищену тактову частоту F_t = 3,0 або 4,0 МГц за $V_п$ = 5 В.

Блок програмування використовується задля завантаження інформації до РПЗП-ЕС. Завантаження здійснюється через асинхронний послідовний порт відповідно до програми завантаження, яка зберігається у службовому ПЗП. При цьому здійснюється контроль правильності запису кожного байта.

Блок конфігурування використовується задля визначення режиму роботи пристроїв, які входять до складу МК.

Блок контролю функціонування забезпечує контроль за виконанням програми за допомогою вартового таймера WDT (Watch-Dog Timer), а також контроль частоти імпульсів тактової синхронізації.

Блок паралельних та послідовних портів, у різних моделях МК вміщує від двох до чотирьох 8-розрядних паралельних портів, всі виводи котрих можуть використовуватися задля двоспрямованного обміну. В деяких моделях окремі виводи може бути призначено лише для виведення або введення, а також задля приймання сигналів запитів переривань, входів аналого-цифрового перетворювача, входу таймера, виходів сигналів контролера рідинно-кристалічного індикатора і для організації виводів синхронних та асинхронних портів.

Задля послідовного обміну використовуються асинхронні порти SCI (Serial Communication Interface) або SCI+ і синхронні порти SPI (Serial Peripheral Interface) або SIOP (Simple Synchronous Serial Input-Output Port) або SSPI (Simple Serial Peripheral Interface). Деякі серії мають у своєму складі спеціалізовані послідовні порти, призначені для організації мікроконтролерних мереж. Такі МК широко використовуються в складі пристроїв автоматики, вимірювальної апаратури тощо.

До складу **блока периферійних пристроїв** можуть входити таймери, широтно-імпульсний модулятор, спеціальні вихідні схеми задля підключення пристроїв індикації тощо.

МК сімейства M68HC05/705 можуть працювати у двох режимах зі зниженим енергоспоживанням: очікування і зупину.

У режимі очікування (Wait mode) зупиняється робота процесора, але продовжується робота послідовних портів, таймерів, інших периферійних пристроїв та блока контролю функціонування. Перехід до цього режиму виконується при надходженні команди WAIT. Повернення до робочого режиму здійснюється при надходженні зовнішнього запиту переривання, сигналів переривання від таймера, сигналу запуску від блока контролю функціонування і зовнішнього сигналу RESET#. Повернення відбувається за один період тактової частоти. Залежно від джерела надходження сигналу повернення до робочого режиму до програмного лічильника PC буде завантажено або адресу відповідного вектора переривання, або вектор початкового завантаження. В цьому режимі споживана потужність зменшується в кілька разів.

Перехід до режиму зупину відбувається при надходженні команди STOP. В цьому режимі припиняється робота генератора тактових імпульсів, процесора, послідовних портів, таймерів, інших периферійних пристроїв та блока контролю функціонування. В цьому режимі споживаний струм зменшується до одиниць і навіть часток мікроамперів. Вихід з цього режиму відбувається при надходженні зовнішнього запиту на переривання (на вхід IRQ# або на вивід МК, який запрограмовано як вхід сигналів переривань) або зовнішнього сигналу RESET#. До програмного лічильника PC буде завантажена адреса зовнішнього переривання або вектор початкового завантаження. Повернення до робочого режиму відбувається за час, що дорівнює $4064T_t$.

При переході до режиму зниженого енергоспоживання у регістрі ознак CCR встановлюється значення ознаки $I = 0$, щоби забезпечити дозвіл переривання і повернення до робочого режиму.

В деяких моделях МК сімейства передбачено режим неповного зупину (Halt mode), який відрізнюється від режиму очікування лише тим, що генератор тактових імпульсів відключається від процесора, що зменшує споживану потужність. Задля повернення до робочого режиму в такому разі потрібен більший час, щоби забезпечити встановлення заданих параметрів тактових імпульсів.

Регістрову модель процесора CPU05 подано на рис. 13.2.

Процесор виконує обробку 8-розрядних операндів і зреалізовує 65 команд. До складу регістрової моделі входять:

– 8-розрядний акумулятор, використовуваний для зберігання одного з операндів та результату виконування арифметичних чи логічних операцій;

– 8-розрядний індексний регістр X, використовуваний для формування адреси при індексному адресуванні. Цей регістр також є джерелом одного з операндів при виконуванні операції множення, а також може використовуватися задля тимчасового зберігання операндів та результатів;

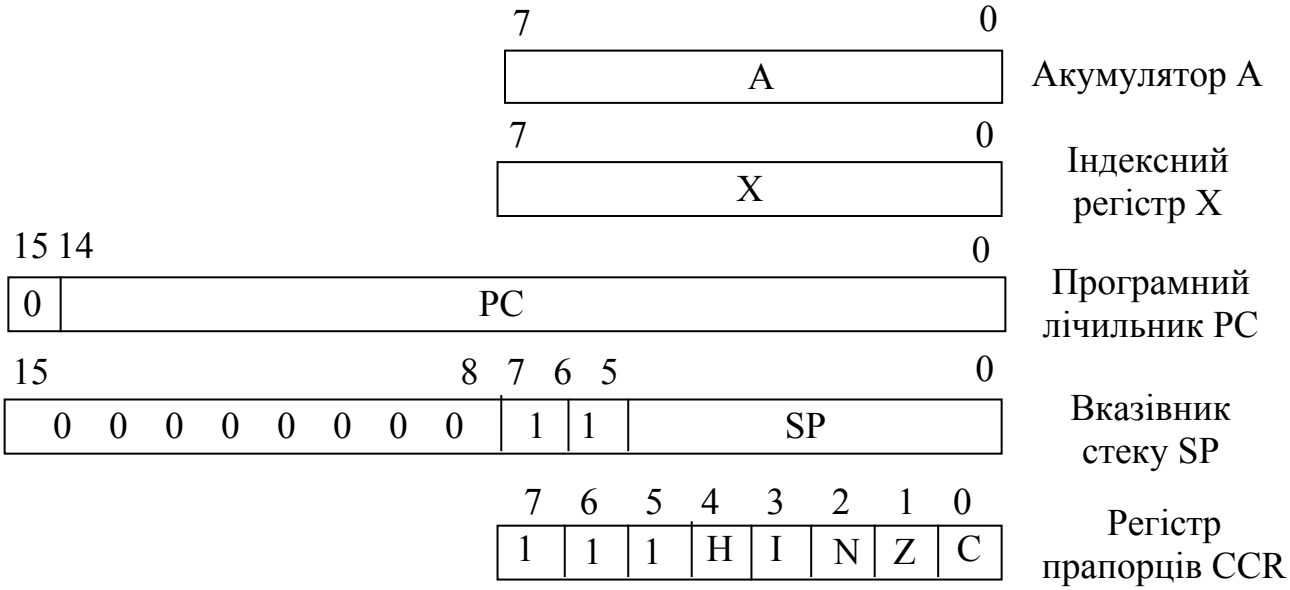


Рисунок 13.2 – Регістрова модель процесора CPU05

– 8-розрядний регістр прапорців CCR, який зберігає значення п’яти ознак результату, як показано на рис. 13.2:

– C (Carry/Borrow Flag)– ознака перенесення (набирає значення C = 1, якщо при виконуванні операції відбувається перенесення зі старшого розряду результату);

– Z (Zero Flag) – ознака нульового результату (Z = 1, якщо результат операції дорівнює нулю);

– N (Negative Flag) – ознака знаку результату (N = 0, якщо результат додатний, та N = 1 за від’ємного результату);

– I (Interrupt Mask) – ознака дозволу переривання (I = 0 – обробляння переривань дозволено, I =0 – заборонено);

– H (Half-Carry Flag) – допоміжне (міжтетрадне) перенесення. Використовується при виконуванні операцій над двійково-десятковими числами.

– 16-розрядний програмний лічильник PC вміщує адресу поточної виконуваної команди. Кількість використовуваних розрядів залежить від обсягу внутрішньої пам'яті МК. Максимальний обсяг, що може адресуватися, становить 32 кбайти (розряди 0...14). При включенні МК (виконання команди RESET) до програмного лічильника автоматично завантажується адреса першої команди програми (вектор початкового завантаження) з двох останніх комірок пам'яті ($FFFE...$FFFF);

– 16-розрядний вказівник стека SP, який використовується для роботи зі стеком. Стек в МК сімейства M68HC05/705 має обсяг 64 байти і розміщується наприкінці молодших 256 байтів ОЗП даних, тому для адресування використовується лише 6 молодших розрядів регістра SP, куди при запуску МК автоматично завантажується $00FF. Стан старших 10 розрядів вказівника стека при роботі не змінюється.

Блок конфігурування, призначений для визначення певних характеристик і режимів роботи МК і складається з регістрів конфігурування. Ці регістри в МК підсімейства 68HC05 є комірками пам'яті ПЗП, програмованими маскою на етапі виготовлення мікросхеми, тому їхній вміст при експлуатації не змінюється. В МК підсімейства 68HC705 ці регістри є комірками одноразово програмованих ПЗП, які програмуються при першому включенні мікросхеми і за подальшої роботи не змінюються. Лише при використовуванні МК з РПЗП-ЕС вміст цих комірок можна змінювати. Кількість регістрів і призначення окремих бітів в них зумовлюється типом МК. Так, в МК MC68HC705J1A, який досліджується при виконанні лабораторного практикуму з дисципліни, блок конфігурування представлено одним регістром MOR. Формат вмісту цього регістра подано на рис. 13.3.

7	6	5	4	3	2	1	0
SOSCD	EPMCEC	OSCRES	SWAIT	SWPDI	PIRQ	LEVEL	COPEN

Рисунок 13.3 – Формат вмісту регістра MOR

Біти цього регістра мають таке призначення:

– SOSCD – визначає значення затримки включення при запуску МК, яка потрібна задля входження до робочого режиму генератора тактових імпульсів. Якщо біт SOSCD = 0, то затримка становить $4064T_t$, а при SOSCD = 1 – $128T_t$. Цей варіант зреалізовується при використовуванні зовнішнього генератора тактових імпульсів;

– EPMCEC – біт захисту. Значення біта EPMCEC = 1 забороняє читання вмісту РПЗП-ЕС;

– OSCRES – при значенні цього біта OSCRES = 1 зреалізовується підключення внутрішнього резистора поміж виводами OSC1 та OSC2 при

використовуванні високочастотного кварцового резонатора в якості елемента, який визначає часові характеристики;

– SWAIT – значення цього біта SWAIT = 1 забезпечує переключення МК до режиму неповного зупину при надходженні команди STOP;

– SWPDI – значення цього біта SWPDI = 1 забезпечує підключення резисторів, які “підтягують” їхній потенціал до рівня логічного 0, для виводів портів, запрограмованих на введення даних;

– PIRQ – значення цього біта PIRQ = 1 забезпечує роботу виводів молодшої тетради порту А (PA3 – 0) в якості входів для зовнішніх запитів переривання;

– LEVEL – визначає вид сигналу зовнішнього переривання. LEVEL = 1 відповідає сигналові переривання, котрий має низький рівень (логічний 0), LEVEL = 0 визначає сигнал переривання як зріз потенціалу;

– COPEN – дозволяє при значенні COPEN = 1 роботу блока функціонування.

Найбільш складну організацію має регістр конфігурування МК MC68HC705C8A, котрий зреалізовується за допомогою трьох регістрів: OPTION, MOR1 та MOR2.

Регістр OPTION входить до складу блока пам’яті МК і має адресу $1FDF, регістри MOR1 та MOR2 входять до складу блока конфігурування. Формат цих регістрів подано на рис. 13.4.

а)

7	6	5	4	3	2	1	0
RAM1	RAM0	0	0	SEC	0	IRQ	0

б)

7	6	5	4	3	2	1	0
PBPU7	PBPU6	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0 COPC

в)

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

Рисунок 13.4 – Формат вмісту регістрів OPTION (а), MOR1 (б) та MOR2(в)

Регістр OPTION визначає тип розділів пам’яті, що конфігуруються, і вид сигналів зовнішнього переривання. Його біти мають такі значення:

– RAM0 визначає тип першого розділу пам’яті, що конфігурується, (адреси комірок $0020...$004F). Значення RAM0 = 0 свідчить, що у цьому розділі розташовано комірки РПЗП-ЕС, а за RAM0 = 1 у цьому розділі розташовано комірки ОЗП;

– RAM1 визначає тип другого розділу пам’яті, що конфігурується, (адреси комірок $0020...$015F). Значення RAM1 = 0 свідчить, що у цьому розділі розташовано комірки РПЗП-ЕС, а за RAM1 = 1 у цьому розділі розташовано комірки ОЗП;

– IRQ визначає вид сигналу зовнішнього переривання, який сприймається як запит переривання. IRQ = 1 відповідає сигналові переривання, що має низький рівень (логічний 0), а IRQ = 0 схарактеризовує сигнал переривання як зріз імпульсу;

– SEC – біт захисту. Значення біта SEC = 1 забороняє читання вмісту РПЗП-ЕС.

Регістр MOR1 (адреса $1FF0) визначає призначення виводів порту B: при значенні PBPUx = 0 відповідний вивід PUx використовується як вхід зовнішніх запитів переривання, при PBPUx = 0 цей вивід використовується для обміну даними. Молодший біт (PBPU0 COPC) цього регістру також використовується для скидання вартового таймера у блоці контролю функціонування.

Регістр MOR2 (адреса $1FF1) використовується в режимі емуляції моделі MC68HC0512A і виконує функції скидання вартового таймера у блоці контролю функціонування цієї моделі.

Блок контролю функціонування складається з блоків контролю функціонування COP (Computer Operating Property), до складу котрих входить вартовий таймер, а також монітор тактових імпульсів (в деяких моделях).

Вартовий таймер блока контролю функціонування (WDT) МК сімейства M68HC05/705 контролює правильність виконування програми. Правильність виконування перевіряється за результатами запису певних значень до відповідних регістрів. Якщо такий запис не виконується за певний час T_w, то блок контролю формує сигнал RESET# – і відбувається перезавантаження контролера. Період контролю визначається при конфігуруванні МК.

В МК MC68HC705J1A робота WDT дозволяється при встановленні біта COPEN = 1 до регістра конфігурування MOR. Скидання вартового таймера відбувається під час запису нуля до молодшого біта регістра COPR, адреса якого є $07F0. Період скидання має не перевищувати

$$T_w = 262144\ K_w / F_t,$$

де K_w – коефіцієнт задля завдання можливих інтервалів часу контролю, визначається значенням бітів RT1-0 у регістрі керування таймером MFT. Визначається за табл. 13.1; F_t – тактова частота МК.

Таблиця 13.1 – Відповідність значень K_w бітам RT1-0

RT1-0		K_w
0	0	1
0	1	2
1	0	4
1	1	8

Така процедура використовується у більшості моделей і має назву – непрограмоване скидання. Вона зреалізовується періодичним завантаженням нуля до молодшого біта регістру COPR:

```
LDA #$x0      ; Завантаження до акумулятора числа, яке має значення 0
              ; у молодшому розряді
STA $07F0     ; Перезапис числа з акумулятора до регістру COPR
```

Ці команди слід включити до тексту робочої програми і вони мають виконуватися не рідше одного разу за час контролю T_w.

Режим непрограмованого скидання для МК MC68HC705C8A зреалізовується при запису 1 до біта NCOPE регістра MOR2, а скидання вартового таймера відбувається під час запису 0 до молодшого біта регістра MOR1. Для визначення періоду скидання в цьому разі значення коефіцієнта $K_w = 1$.

В деяких моделях МК, в тому числі MC68HC705C8A, можна зреалізовувати програмне скидання замість описаного вище. При цьому для керування роботою вартового таймера використовується вміст двох регістрів у блоці контролю функціонування: регістра керування COPCR (адреса $001E) і регістра скидання COPR. Вміст регістра COPCR подано на рис. 13.5.

7	6	5	4	3	2	1	0
0	0	0	COPF	CME	COPE	CM1	CM0

Рисунок 13.5 – Формат вмісту регістра COPCR

Біти регістра COPCR мають таке призначення:

– COPF – ознака перезапуску блока контролю функціонування; біт набирає значення COPF = 1, якщо відбувся перезапуск МК після спрацьовування вартового таймера чи зниження частоти тактових імпульсів (значення цього біта можна лише читати);

– CME – значення CME = 1 дозволяє роботу схеми контролю частоти тактових імпульсів, протилежне значення – забороняє;

– COPE – значення COPE = 1 дозволяє контроль за виконанням програми за допомогою вартового таймера, котрий зреалізовує непрограмне скидання;

– CM1, CM0 – визначають значення коефіцієнта K_w аналогічно до значень RT1-0 регістра керування таймером MFT.

Читання вмісту регістра COPCR призводить до скидання значення біта COPF = 0. При включенні МК всі біти, окрім COPF, встановлюються в 0. Після запуску біти CME, COPE, CM1–0 можуть встановлюватися в 1 лише один раз, а потім лише читатися.

Для контролю виконання програми, в цьому разі, встановлюється значення біта PCOPE = 1, а до регістра COPR (адреса $001D) здійснюється періодичний запис чисел $55, потім $AA. Часовий інтервал поміж цими записами не повинен перевищувати значення T_w, яке здобувається як

$$T_w = 32768\ K_w / F_t,$$

де значення K_w та F_t віднаходяться в наведений вище спосіб.

Якщо за час T_w запису обох чисел не відбувається, то формується сигнал RESET# і здійснюється перезавантаження МК.

Блок контролю функціонування МК MC68HC705C8A також вміщує монітор тактових імпульсів, котрий контролює значення частоти генератора тактових імпульсів. Ця схема формує сигнал RESET# , якщо за час T_x не надходить тактового імпульсу. Цю схему слід виключати, якщо передбачається робота з низькими тактовими частотами.

Блок паралельних та послідовних портів МК MC68HC705J1A вміщує лише два паралельних порти А та В з можливих п’яти, передбачених у складі блока паралельних та послідовних портів сімейства M68HC05/705.

Вісім виводів порту А МК MC68HC705J1A може бути запрограмовано на введення/виведення даних у паралельному форматі або чотири з них можуть використовуватися в якості входів для сигналів переривань. Порт В має лише шість виводів для організації обміну у паралельному форматі.

Робота портів зорганізовується за допомогою регістрів даних PORTA та PORTB, котрі мають адреси відповідно $00 та $01, а також двох регістрів напрямку пересилання DDRA та DDRB. Регістри напрямку пересилання визначають режим роботи відповідних виводів, котрі після початкового завантаження запрограмовані на приймання даних (в регістрах DDRx записано 0 у всіх розрядах). Для використання виводу в якості виходу даних, відповідні розряди регістрів напрямку слід встановити у стан 1.

Встановлення біта SWPDI = 1 дозволяє підключення (”підтягування”) входів портів до рівня логічного нуля задля виключення завад, які можуть виникати у вхідних колах МК, при переході виходів джерела сигналів до високоімпедансного стану.

Виходи портів забезпечують значення струмів навантаження $I_{н0}$ = 1,6 мА при формуванні сигналу логічного 0 і $I_{н1}$ = 0,8 мА – при формуванні сигналу логічної 1.

У складі МК MC68HC705J1A блока послідовних портів немає.

У складі інших МК сімейства M68HC05/705 він складається з таких пристроїв.

Асинхронний послідовний порт (SCI), який входить до складу МК типу MC68HC705C8A, забезпечує стандартний асинхронний формат приймання-передавання даних у вигляді кадру, який вміщує один стартовий і один стоповий біти, вісім інформаційних бітів (D0...D7) і за потреби може доповнюватися одним додатковим контрольним бітом (D8*). Формат кадру подано на рис. 13.6.

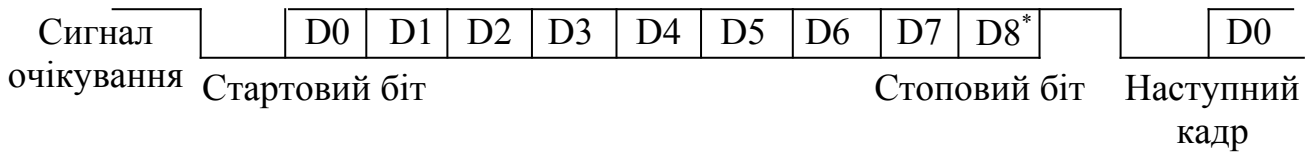


Рисунок 13.6 – Формат кадру даних порту SCI

Обмін даними здійснюється через буферний регістр SCDR, вхід і вихід котрого з'єднуються з відповідними виводами паралельного порту D, які використовуються для послідовного обміну (PD0 та PD1).

Перед передаванням даних на лінії встановлюється сигнал очікування (сигнал логічної 1) з часом передавання не менше за період тактової синхронізації $T_s = 1/F_s$. Відсутність цього сигналу упродовж часу, понад $(10...11)T_s$, сприймається як сигнал завершення обміну – BREAK.

Якщо кадр вміщує в усіх інформаційних бітах, а також у контрольному біті сигнал, що дорівнює 1, то такий кадр сприймається як сигнал завершення передавання – BREAK, а в разі передавання сигналу логічного 0 – як сигнал очікування наступного циклу передавання даних – IDLE.

Керування обміном даними через порт SCI здійснюється відповідно до вмісту 8-розрядних регістрів керування SCCR1 (адреса $0E) і SCCR2 (адреса $0F), регістра стану SCSR (адреса $10) і регістра BRR (адреса $0D), формат вмісту котрих подано на рис. 13.7.

	7	6	5	4	3	2	1	0
а)	R8	T8	–	M	WAKE	–	–	–
б)	TIE	TCIE	RIE	TLIE	TE	RE	RWU	SBK
в)	TDRE	TC	RDFR	IDLE	OR	NF	RE	–
г)	–	–	SCP1	SCP0	–	SCR2	SCR1	SCR0

Рисунок 13.7 – Формат вмісту регістрів SCCR1 (а), SCCR2 (б), SCSR (в), BRR (г)

Регістр SCCR1 (рис. 13.7, а) вміщує такі біти:

– WAKE – сигнал активізації приймача;

Значення WAKE = 0 активізує приймач при надходженні на вхід даних RDI символа IDLE, а значення WAKE = 1 – якщо старший (адресний) розряд даних, що надійшли на вхід, має одиничне значення;

– M – сигнал, що визначає розрядність даних;

M = 0 визначає, що дані мають розрядність 1 байт (контрольного біта немає), а значення M = 1 відповідає даним, що мають довжину 9 розрядів (дані вміщують контрольний біт);

– R8 – розряд призначено для запису значення біта контролю до прийнятого сигналу;

– T8 – біт, значення котрого записується у кадр і передається як біт контролю.

Біти регістра SCCR2 (рис. 13.7, б) мають таке призначення:

– TIE – біт дозволу формування запиту контролю переривання; значення TIE = 1 дозволяє формування сигналу при встановленні в регістрі SCSR ознаки звільнення буферного регістра при передаванні даних – TDRE = 1;

– TCIE – значення TCIE = 1 дозволяє формування запиту контролю переривання при встановленні в регістрі SCSR ознаки завершення передавання даних – TC = 1;

– RIE – значення RIE = 1 дозволяє формування запиту контролю переривання при встановленні в регістрі SCSR ознаки заповнення буфера або переповнення приймача – RDRF = 1 або OR = 1;

– ILIE – біт дозволу формування запиту контролю переривання (ILIE = 1) при надходженні на вхід даних RDI сигналу завершення обміну (встановлення біта IDLE = 1 в регістрі SCSR);

– TE – біт включення передавача (TE = 1), а також його відключення (TE = 0);

– RE – біт включення приймача (RE = 1), а також його відключення (RE = 0);

– RWU – біт керування приймачем; значення RWU = 1 переводить приймач до режиму очікування, а також дозволяє (відповідно до значення WAKE) активізацію приймача;

– SBK – біт завершення обміну; при значенні SBK = 1 формується і передається на вихід передавача символ завершення обміну BREAK (послідовність 0).

Перед передаванням даних, записаних до регістра SCDR, біт SBK встановлюється в 1, потім це значення скидається і передавач передає символ завершення обміну (10 або 11 нулів) – встановлює на виході рівень 1 упродовж одного періоду сигналу синхронізації; після цього розпочинається передавання кадру даних.

Регістр стану SCSR (рис. 13.7, в) вміщує ознаки, формовані в перебігу передавання чи приймання даних. Для користувача він є доступний лише для читання. Біти регістра стану мають таке призначення:

– TDRE – ознака звільнення буфера передавача; цей біт набирає значення TDRE = 1 після завершення перезапису вмісту регістра SCDR до вихідного зсувового регістра для подальшого передавання лінією зв’язку;

– TC – ознака завершення передавання; набирає значення TC = 1 після передавання останнього біта даних з вихідного зсувного регістра;

– RDRF – ознака заповнення буфера приймача; значення RDRF = 1 встановлюється після завершення перевантаження даних з вхідного зсувного регістра до буферного регістра;

– IDLE – ознака надходження сигналу очікування; якщо на вході даних сигнал дорівнює 1 упродовж 10...11 тактів сигналу синхронізації, то біт набирає значення IDLE = 1;

– OR – ознака переповнювання приймача. Встановлюється значення OR = 1, якщо поточний символ надходить до зсувного регістра приймача до

зчитування з регістра SCDR попереднього. В такому разі вміст SCDR зберігається, а символ, що надійшов, втрачається;

– NF – ознака наявності шумів на лінії приймання. Встановлюється значення NF = 1, в разі зміни стану входу до завершення приймання поточного біта;

– FE – ознака порушення кадру. Значення FE = 1 встановлюється, якщо тривалість стопового біта є меншою за період сигналу синхронізації T_s.

Ознаки запитів переривань TDRE, TC, RDRF, IDLE, OR формуються, якщо це дозволено відповідними бітами регістра SCCR2. Програми обслуговування цих переривань повинні зчитати і проаналізувати вміст регістра SCSR задля правильної обробки переривання. Ознаки NF та FE не формують запитів на переривання, читаються і аналізуються за результатами програмного опитування.

Значення ознак TDRE і TC скидаються в 0, якщо при передаванні даних читання вмісту SCSR виконується після запису даних до регістра SCDR, а значення ознак RDRF, IDLE та OR скидаються в 0, якщо при прийманні даних читання вмісту SCSR виконується після зчитування даних з регістра SCDR.

Вміст регістра BRR (рис. 13.7, г) зумовлює швидкість обміну даними встановленням значення тактової частоти F_s відповідно до виразу $F_s = F_t / (K_d \cdot K_s)$. Значення коефіцієнтів K_d та K_s обираються встановленням відповідних значень бітів регістра BRR. Значення K_d зумовлюється значенням бітів SCP1-0, а коефіцієнта K_s – SCP2-0. Це надає змогу зреалізовувати великий діапазон значень швидкостей обміну від 10 біт/с до 100 кбіт/с.

В деяких моделях МК сімейства використовується модифікований асинхронно-синхронний порт SCI+. Використовування такого порту надає змогу визначати швидкість обміну окремо для лінії передавання і приймання даних, а також забезпечує можливість синхронного передавання даних. Для забезпечення синхронного передавання даних порт має окремий вихід синхросигналу, який не суміщено з виходами паралельних портів.

Синхронний послідовний порт (SPI), який входить до складу порту D МК MC68HC705C8A, складається з регістра даних SPDR (адреса $0C), регістра керування SPCR (адреса $0A) і регістра стану SPSR (адреса $0B).

Обмін здійснюється 8-розрядними даними, які зчитуються з регістра даних SPDR або записуються до нього.

Керування процесом обміну здійснюється відповідно до стану бітів регістра SPCR, формат вмісту якого подано на рис. 13.8, а схемна реалізація організації послідовного обміну поміж двома абонентами через синхронний послідовний порт SPI – на рис. 13.9.

Вивід SCK (рис. 13.9) призначено для передавання сигналу синхронізації. На боці ведучого SPI цей вивід використовується як вихід сигналу, а на боці веденого як – вхід. Лінія MOSI використовується для передавання інформації від ведучого до веденого SPI. Лінією MISO здійснюється передавання інформації від веденого SPI до ведучого. Виводи SS# використовуються для вибору режиму функціонування; рівень 1, який надходить на цей вивід, визначає відповідний SPI як ведучий, а рівень 0 – як ведений.

7	6	5	4	3	2	1	0
SPIE	SPE	–	MSTR	CPOL	CPHA	SPR1	SPR0

Рисунок 13.8 – Формат вмісту регістру керування SPCR

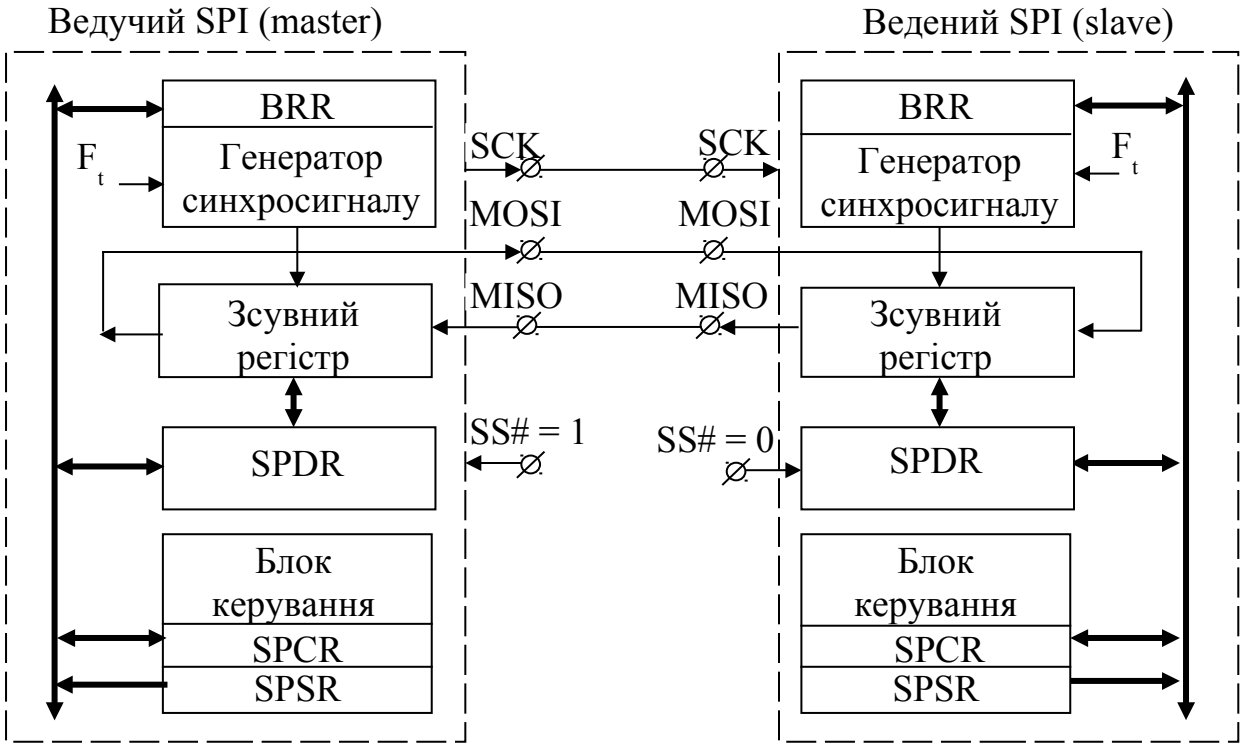


Рисунок 13.9 – Організація синхронного обміну даними через порт SPI

Обмін проводиться по лініях MOSI або MISO і в кожному такті здійснюється запис одного біта до молодшого розряду зсувного регістру і вивід одного біту зі старшого. Обмін проводиться байтами, які зчитуються або записуються до регістру SPDR.

Обмін здійснюється відповідно до коду, який записано у регістрі SPCR (див. рис. 13.8):

– значення SPE = 1 призводить до включення порту і використання його виводів (див. рис. 13.9);

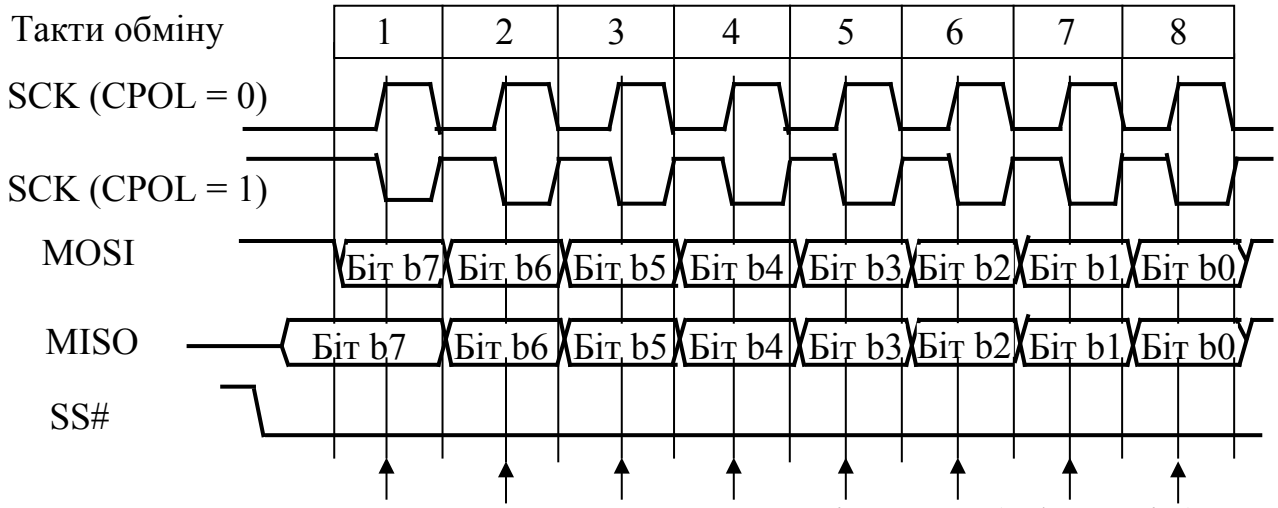
– значення біта MSTR визначає порт для програмування в якості веденого або ведучого; MSTR = 1 визначає відповідний порт як ведучий, а значення 0 – як ведений;

– значення бітів SPR1-0 ведучого SPI зумовлює швидкість обміну; значення цих бітів задають значення коефіцієнта ділення K_d у виразі $F_s = F_t/K_d$;

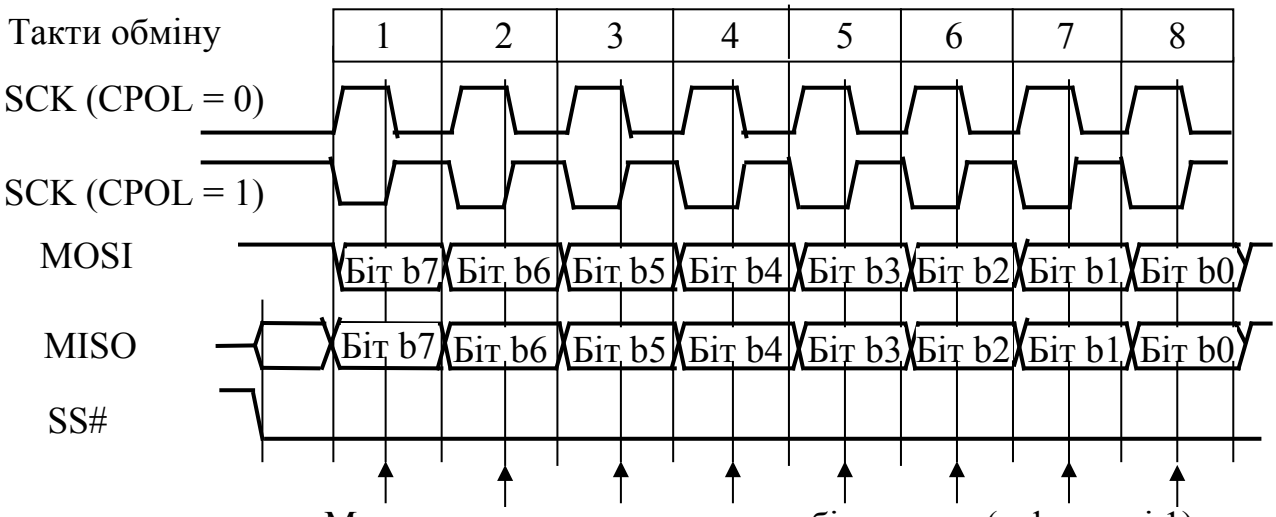
– біт CPOL визначає полярність синхросигналу. Значення CPOL = 0 підтримує на лінії SCK незмінний рівень 0 до початку циклу обміну, а значення CPOL = 1 – рівень 1. Після завантаження даних до регістра SPDR ведучого порту на його вихід SCK надходять сигнали тактової синхронізації. Фронт або зріз цих сигналів, залежно від значення біта CPOL, зумовлюють момент

початку передавання поточного біта або початок запису його до зсувного регістра;

– CPHA – біт фази синхронізації. Визначає активний перепад сигналу синхронізації, який визначає формат передавання даних, як показано на рис. 13.10. Робота порту SPI може здійснюватися у форматі 0 (рис. 13.10, а) або у форматі 1 (рис. 13.10, б).



а)



б)

Рисунок 13.10 – Часові діаграми процесу передавання даних через SPI

Порти SPI працюють у форматі 0 при встановленні у регістрі SPCR значення біта CPHA = 0. При цьому ведений SPI формує старший біт сигналу, що передається після отримання від ведучого SPI сигналу SS# = 0 (як показано на рис. 13.10, а). Значення біту CPOL = 0 означає, що після запису даних до регістра SPDR ведучого SPI, розпочинається процес обміну даними і видавання даних з ведучого та веденого портів здійснюється при встановленні на виводі SCK рівня 0, а введення даних до зсувних регістрів – при встановленні на

виводі SCK рівня 1. Якщо значення біта CPOL = 1, то виведення даних відбувається при значенні SCK = 1, а введення – при рівні 0.

Після завершення передавання символу на вхід SS# веденого порту треба подати сигнал рівня 1, а перед початком передавання наступного символу встановити рівень 0.

При роботі у форматі 1 ведучий і ведений порти розпочинають передавання даних водночас. Якщо значення біта CPOL = 0, то початок передавання визначається надходженням фронту сигналу синхронізації і надходженням зрізу, якщо значення CPOL = 1.

Введення даних здійснюється при надходженні фронту, якщо CPOL = 1 і зрізу, – якщо CPOL = 0. При цьому значення сигналу SS# на вході веденого порту може зберігати своє значення поміж циклами передавання різних символів.

Контроль за процесом обміну здійснюється за допомогою вмісту регістра стану SPSR, вміст котрого подано на рис. 13.11. Значення бітів – ознаки виконування поточного циклу обміну – встановлюються після завершення кожного циклу обміну.

7	6	5	4	3	2	1	0
SPIF	WCOL	–	MODF	–	–	–	–

Рисунок 13.11 – Формат вмісту регістра стану SPSR

Біти регістра стану SPSR мають таке призначення:

– SPIF – ознака завершення обміну. Після передавання байта даних цей біт набирає значення SPIF = 1. При цьому формується запит переривання SPI, якщо в регістрі SPCR біт дозволу переривання встановлено SPIE = 1;

– WCOL – ознака помилки запису. Встановлюється значення WCOL = 1, якщо здіснюється спроба запису до регістру даних SPDR до завершення процесу зчитування даних з нього;

– MODF – ознака помилки режиму. Встановлюється значення MODF = 1 за спроби перевести порт, який працює в режимі ведучого, до режиму веденого.

Вміст регістра SPSR може бути лише зчитано. Обнулення бітів цього регістра відбувається після зчитування його поточного стану і запису до регістру SPDR даних для наступного циклу обміну.

Послідовний периферійний порт (SIOP)**,** котрий використовується в МК MC68HC705P6A, може використовуватися для обміну даними, який буде відбуватися розпочинаючи зі старшого або молодшого розряду.

Для обміну використовуються три виводи: SDI – для передавання інформації від веденого порту до ведучого (аналог MISO), SDO – для передавання інформації від ведучого порту до веденого (аналог MOSI) і лінії SCK – для передавання сигналу синхронізації. Виводи порту SIOP зорганізовуються на виводах PB5-7 паралельного порту B. Сигнал SS# – відсутній, тому що в цьому порту не формується запит переривання після

завершення передавання і стан порту після завершення поточного циклу обміну можна контролювати лише програмно.

Послідовний периферійний порт (SSPI) переважно є аналогічний до порту SIOP.

Блок периферійних пристроїв.

Таймер є одним з головних пристроїв, які входять до складу цього блока і призначений для формування певних часових інтервалів. Необхідність введення таймера до складу блока периферійних пристроїв зумовлено призначенням МК для керування певними об'єктами та процесами.

До складу МК MC68HC705J1A входить 15-розрядний багатофункціональний таймер MFT, структурну схему котрого подано на рис. 13.12.

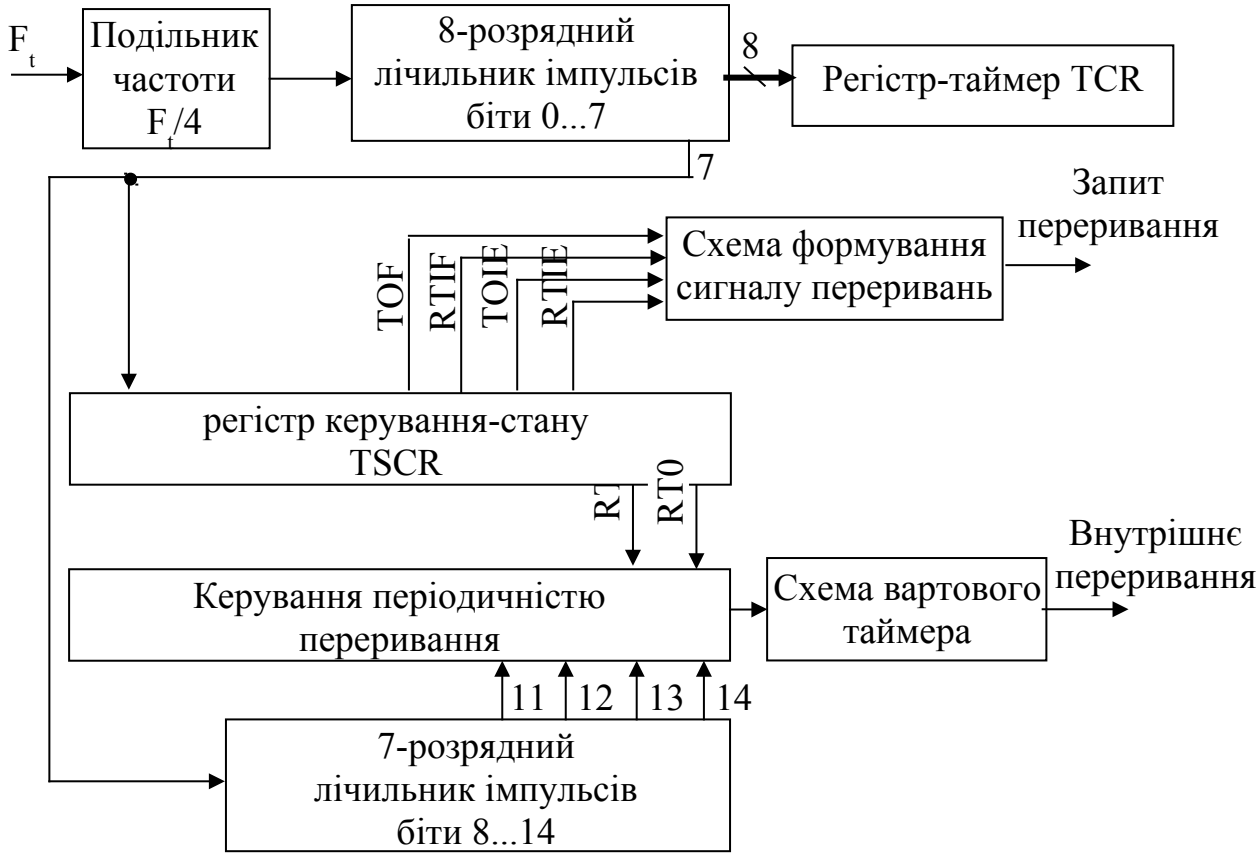


Рисунок 13.12 – Структурна схема багатофункціонального таймера

Він побудований на базі 15-розрядного лічильника, 8 молодших розрядів котрого являють собою програмно-доступний регістр-таймер TCR (адреса $09), а 7 старших розрядів використовуються як подільник частоти при формуванні періодичних переривань. Для контролю за роботою таймера використовується вміст регістра-таймера, який може бути прочитано за будь-якого моменту часу. Вихід старшого, 7-го, біта може використовуватися задля переривання роботи таймера через кожні 1024 періоди тактової частоти. Частота роботи лічильника становить $F_c = F_t / 4$.

Вартовий таймер – COP (Computer operation property), який входить до складу багатофункціонального таймера, призначено задля контролю за роботою програмного забезпечення і реалізації переривань через певні відліки часу.

До складу багатофункціонального таймера MFT також входить регістр керування-стану TSCR (адреса $08), формат котрого подано на рис. 13.13.

7	6	5	4	3	2	1	0
TOF	RTIF	TOIE	RTIE	TOFR	RTIFR	RT1	RT0

Рисунок 13.13 – Формат регістра керування-стану TSCR

Біти цього регістра відповідають певним ознакам, використовуються для керування роботою таймера і мають таке призначення:

– TOIE (Timer Overflow Interrupt Enable) – ознака дозволу переривання переповнювання;

– TOF (Timer Overflow Flag) – ознака переповнювання. Значення TOF = 1 встановлюється при зміні вмісту регістра-таймера з $FF на $00. Якщо, при цьому ознака дозволу переривання переповнювання встановлена TOIE = 1, то відбувається переривання роботи МК. Отже, переривання переповнювання може відбуватися через кожні 2·10 = 1024 періоди тактової частоти;

– RTIF (Real-Time Interrupt Flag) – ознака періодичного переривання;

– RTIE (Real-Time Interrupt Enable) – ознака дозволу періодичного переривання;

– TOFR (Timer Overflow Flag Reset) – біт скидання ознаки переповнювання. Запис 1 до цього біта після переривання дозволяє встановлення ознаки TOF = 0;

– RTIFR (Real-Time Interrupt Flag Reset) – біт скидання ознаки періодичного переривання. Запис 1 до цього біта після переривання дозволяє встановлення ознаки RTIF = 0;

– RT1-0 (Real-Time Interrupt) – біти, що встановлюють часові інтервали встановлення ознаки періодичного переривання. Значення RTIF = 1 встановлюється через інтервали, визначувані відповідно до вмісту цих бітів:

RT1-0	Період встановлення ознаки RTIE
00	$T_p = 2 \cdot 14\ T_t$
01	$T_p = 2 \cdot 15\ T_t$
10	$T_p = 2 \cdot 16\ T_t$
11	$T_p = 2 \cdot 17\ T_t$

Використовування багатофункціонального таймера MFT дозволяє виконувати вимірювання часових інтервалів поміж певними подіями, формування сигналів з певною затримкою, періодичний виклик потрібних підпрограм, формування імпульсів потрібної частоти та довжини, а також формувати сигнали внутрішніх переривань від вартового таймера.

До складу інших МК сімейства можуть входити таймерні блоки з різною організацією для керування певними об’єктами. Більшість з них побудовано на базі 16-розрядного таймерного блока, котрий доповнюється 8-розрядним лічильником-таймером та 14-розрядним базовим таймером. До складу МК MC68HC705B16 також входять два широтно-імпульсних модулятори, котрі можуть формувати послідовність імпульсів з програмованою шпаруватістю.

Сімейство M68HC08/908

Сімейство M68HC08/908 є подальшим розвиненням МК M68HC05/705. Вони зберігають їхні архітектурні особливості, однак дозволяють підвищувати техніко-економічні характеристики пристроїв, до складу котрих вони входять. До складу МК підсімейства M68HC908 входить Flash-пам’ять, що дозволяє широко використовувати його у пристроях, які випускаються малими серіями. Програмна сумісність з МК M68HC05/705 дозволяє використовувати програмне забезпечення, розроблене для них задля програмного керування новими пристроями.

Сьогодні випускається і рекомендовано для використовування понад 30 моделей МК цього сімейства, які відрізнюються складом паралельних і послідовних портів, а також периферійного обладнання.

До складу МК сімейства M68HC08/908 входять: процесорне ядро CPU08; внутрішня пам’ять програм – ПЗП, програмований маскою, обсягом до 32 кбайт або Flash-пам’ять, обсяг котрої становить 60 кбайт; ОЗП даних, який має ємність від 128 байт до 2 кбайт. В деяких моделях є РПЗП-ЕС обсягом 512 байт або 1 кбайт.

Залежно від функціонального призначення МК, до їхнього складу може входити від 5-ти до 8-ми паралельних портів, послідовні порти SCI та SPI. До складу МК деяких серій входять спеціалізовані послідовні порти, призначені для організації мікроконтролерних мереж. Такі МК зреалізовують обмін даними по мультиплексованій шині J1850, зреалізовують інтерфейс з послідовною шиною USB і спеціалізовані інтерфейсні модулі CAN та I2C. МК серії EY вміщує послідовний порт ESCI, яки зреалізовує протокол LIN задля обміну даними по однопровідній лінії зв’язку.

До складу периферійних пристроїв всіх МК сімейства M68HC08/908 входять 16-розрядні таймери. Більшість моделей вміщує 8- або 10-розрядні АЦП.

Окрім того, МК серії LD мають спеціальні виходи сигналів синхронізації і використовуються для керування цифровими моніторами, а МК серії RF мають у своєму складі радіопередавач.

Характерною особливістю побудови МК цього сімейства є побудова МК зі стандартним набором модулів (модульний принцип). Поєднання окремих модулів на одному кристалі надає змогу формувати МК різноманітного функціонального призначення.

Загальну структурну схему МК цього сімейства наведено на рис. 13.14. До складу цієї схеми входять стандартні модулі, кожен з котрих має власне функціональне призначення.

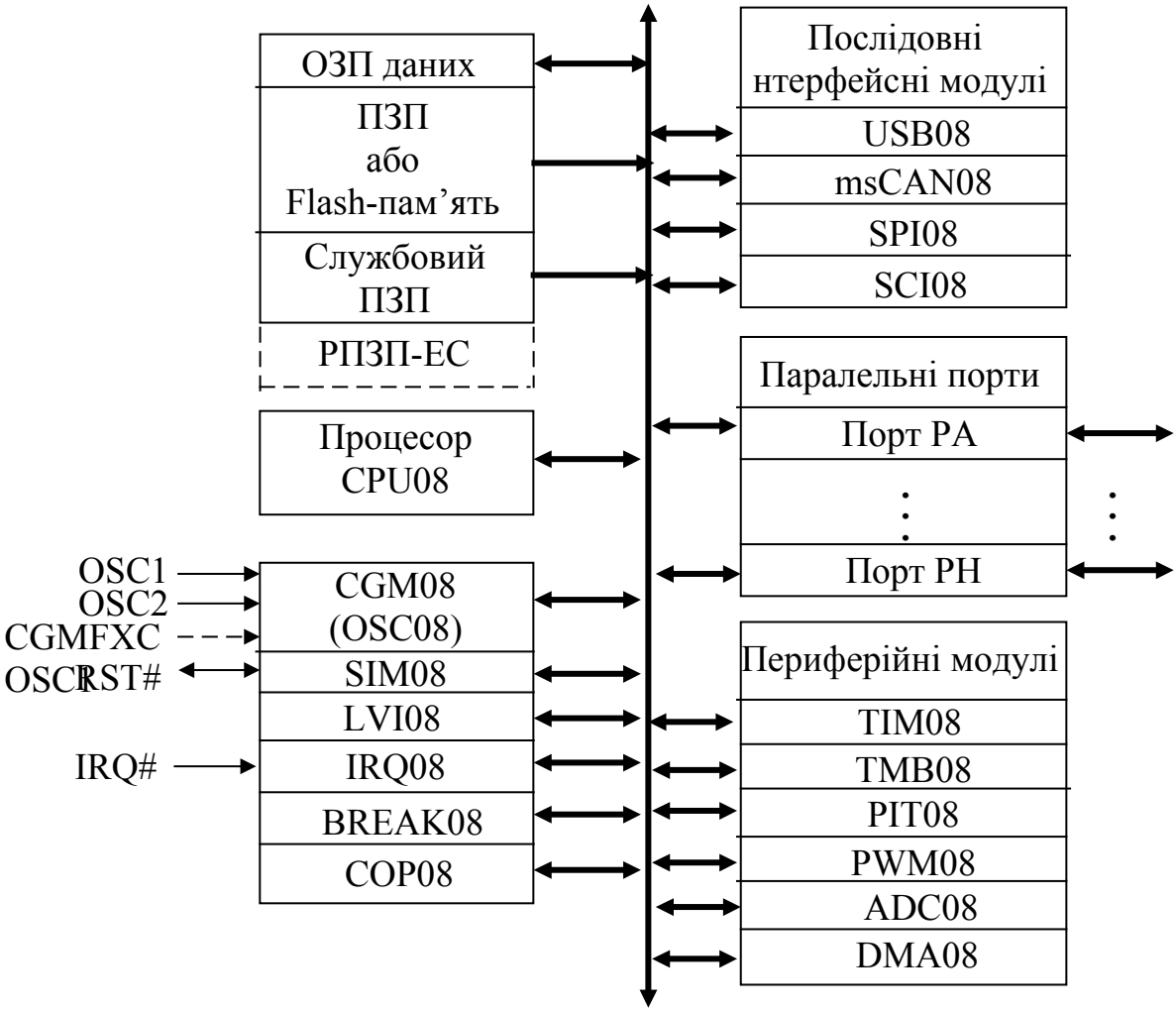


Рисунок 13.14 – Загальна структурна схема МК сімейства M68HC08/908

Конфігурування МК визначається вмістом регістрів конфігурування, котрі визначають характеристики мікроконтролера. МК підсімейства M68HC08 у своєму складі мають регістри конфігурування MOR, які є комірками ПЗП і їхнє програмування здійснюється у перебігу виготовляння мікросхеми.

Регістри конфігурування CONFIG підсімейства M68HC908 треба програмувати при кожному запускові МК, а упродовж сеансу роботи їхній вміст залишається незмінним.

Формат вмісту регістрів CONFIG1 та CONFIG2 для МК серій M68HC908JK1, JK3 та JL3 подано на рис. 13.15.

Біти регістра CONFIG1 призначено для виконування таких функцій:

– COPRS – використовується для керування роботою вартового таймера і визначає період його роботи;

– LVID – використовується для дозволу роботи модуля контроля живлення; значення LVID = 1 забороняє роботу цього модуля;

– SSREC – визначає час затримки Td при виході МК з режиму зупину; при значенні SSREC = 1 час затримки дорівнює Td = 32·Tq і Td = 4096·Tq – при встановленні SSREC = 0;

	7	6	5	4	3	2	1	0
CONFIG1 ($001F)	COPRS	Резерв	Резерв	LVID	Резерв	SSREC	STOP	COPD
CONFIG2 ($001E	IRQPUD	Резерв	Резерв	LVIT1	LVIT0	Резерв	Резерв	Резерв

Рисунок 13.15 – Формат вмісту регістрів конфігурування

– STOP – значення біта STOP = 1 дозволяє перехід МК до режиму зупину при надходженні команди STOP; значення STOP = 0 при виконуванні команди STOP здійснює перезавантаження процесора;

– COPD – використовується для керування роботою вартового таймера; значення COPD = 1 забороняє роботу цього модуля.

Біти регістра CONFIG2 також використовуються для конфігурування і мають таке призначення:

– IRQPUD – значення цього біта IRQPUD = 1 забезпечує підключення до входу IRQ# резистора, який “підтягує” потенціал цього входу до потенціалу напруги живлення;

– LVIT1-0 – визначають значення номінальної напруги живлення; значення LVIT1-0 = 00 або 01 відповідає значенню 5,0 В, а значення LVIT1-0 = 10 – 3,0 В.

Модуль формування тактових сигналів CGM08 (Clock Generator Module) призначено для формування послідовностей імпульсів, потрібних для роботи процесора та периферійних модулів.

Модуль CGM08 вміщує два генератори імпульсів: CG, котрий формує послідовність імпульсів, частота яких F_q визначається зовнішнім кварцовим резонатором, і генератор PG – частота сигналу котрого $F_p = N \cdot F_q$ визначається роботою схеми фазового автоналаштовування частоти.

Сигнал з виходу одного з генераторів після ділення його на 2, надходить на модуль системної інтеграції SIM08, де виконується формування тактових імпульсів. Потреба використовування двох генераторів зумовлена вимогами зменшення рівня високочастотних завад при роботі з високими частотами. Для пристроїв, які працюють з тактовою частотою понад 1 МГц, рекомендовано використовувати генератор PG. Сигнал на виході такого генератора формується в результаті множення відносно низькочастотного сигналу F_q і роботи схеми фазового автоналаштовування частоти PLL (Phase-Locked Loop). Схема PLL має два режими роботи:

– режим захоплення частоти, при роботі в якому після початкового завантаження забезпечується перехід до генерування сигналу в заданому частотному діапазоні;

– режим утримування. В цьому режимі схема підтримує значення вихідної частоти і компенсує можливі відхилення її значення в межах робочої смуги частот.

Керування роботою модуля CGM08 здійснюється за допомогою вмісту регістрів керування PCTL (адреса $001C), PBWC (адреса $001D), PPG (адреса $001E). Формат цих регістрів подано на рис. 13.16.

	7	6	5	4	3	2	1	0
PCTL	PLLIE	PLLF	PLLON	BCS	1	1	1	1
PBWC	AUTO	LOCK	ACQ#	XLD	0	0	0	0
PPG	MUL7	MUL6	MUL5	MUL4	VRS7	VRS6	VRS5	VRS4

Рисунок 13.16 – Формат вмісту регістрів керування модуля CGM08

Регістр PCTL – регістр керування модулем CGM08 – вміщує такі біти:

– PLLIE – біт дозволу переривання за запитом від модуля CGM08. При запуску МК набирає значення PLLIE = 0 (переривання заборонені) і може встановлюватися PLLIE = 1 лише за автоматичного керування генератором PG;

– PLLF – ознака запиту переривання від модуля CGM08. Набирає значення PLLF = 1 за встановленого біта PLLIE і автоматичного керування;

– PLLON – біт дозволу роботи (при PLLON = 1) генератора PG.При запуску МК встановлюється PLLON = 0;

– BCS – біт вибору сигналу задля формування тактових частот. Значення BCS = 0 передбачає використовування генератора CG, а значення BCS = 1 – генератора PG. При включенні МК встановлюється значення BCS = 0, при зміні цього значення – біт PLLON =1 і стає доступним лише для читання;

Регістр PBWC – регістр керування формуванням частоти має в своєму складі біти:

– AUTO – біт керування режимом роботи генератора PG. Значення біта AUTO = 1 встановлюється за автоматичного режиму та AUTO = 0 – за програмного;

– LOCK – біт точного встановлювання частоти за автоматичного режиму роботи. Значення LOCK = 1 встановлюється, якщо частота сигналів, які формуються, перебуває в межах робочої смуги частот, і LOCK = 0 – якщо частота виходить за ці межі;

– ACQ# – біт керування режимом роботи генератора. За автоматичного керування значення ACQ# =0 засвідчує, що робота відбувається в режимі захоплення частоти, а значення ACQ# = 1 – в режимі утримування. За програмного режиму значення цього біта зумовлюють режим роботи: ACQ# = 0 переводить генератор до режиму захоплення частоти, а ACQ# = 1 – до режиму утримування;

– XLD – біт контролю функціонування кварцового резонатора. При встановленні XLD = 1 через 4·N тактів зчитується його значення; якщо значення XLD = 0, то резонатор є активний і формує точне значення частоти; якщо після зчитування значення XLD = 1, то це відповідає неактивному станові резонатора.

При запуску всі біти цього регістра набирають значення 0.

Регістр PPG – регістр завдавання коефіцієнтів, які визначають часові характеристики сигналів, вміщує такі біти:

– MUL7–4 визначають значення коефіцієнта множення частоти N = 1...15;

– VRS7– 4 задають коефіцієнт L = 1…15, котрий визначає значення центральної частоти в робочій смузі частот системи PLL.

Модуль системної інтеграції SIM08 виконує початкове завантаження МК при включенні живлення та перезавантаження при надходженні зовнішнього сигналу скидання RST# або сигналу від модуля контролю функціонування COP08 або при формуванні помилкового коду команди чи звернення до неіснуючої адреси. Окрім того, цей модуль формує тактові сигнали для процесора й інших модулів, керує передаванням команд і даних внутрішньою шиною, обслуговує запити переривання, зреалізовує різні режими роботи МК. До складу модуля входять регістри, вміст котрих відповідає виконуванню перелічених функцій і формат яких подано на рис. 13.17.

	7	6	5	4	3	2	1	0
SRSR	POR	PIN	COP	ILOP	ILAD	0	LVI	0
SBSR	Резерв	РезервIE	Резерв	Резерв	Резерв	Резер.	SBSW	Резерв
SBFCR	BCFE	Резерв	Резерв	Резерв	Резерв	Резерв	Резерв	Резерв

Рисунок 13.17 – Формат вмісту регістрів керування модулем системної інтеграції

Регістр SRSR (адреса $FE01) – регістр стану модуля системної інтеграції SIM08. Його вміст дозволяє встановити причину запуску МК. Кожен з його бітів має відповідне призначення і встановлюється в 1 в разі, якщо:

– POR – відповідає запуску МК при включенні живлення;
– PIN – запуск МК за сигналом скидання RST#;
– COP – запуск за сигналом вартового таймера COP08;
– ILOD – перезавантаження при формуванні помилкового коду команди;
– ILAD – перезавантаження при зверненні до неіснуючої адреси;
– LVI – запуск за сигналом модуля контролю живлення LVI 08.

Вміст регістра стану є доступний лише для зчитування й після його виконання всі біти встановлюються в 0. При роботі програми ініціалізації вміст цього регістра слід зчитати і програма має виконати аналіз причин, що призвели до запуску чи перезавантаження.

Регістр SBSR (адреса $FE00) – регістр коригування адреси повернення з підпрограми обслуговування переривань. До його складу входить лише один біт SBSW, до якого є можливе звернення, решта бітів зарезервована задля тестування в умовах виробника. Біт SBSW встановлюється, якщо переривання в контрольній точці відбулось при виконанні команд WAIT або STOP. В цьому

разі МК виходить з режиму очікування чи зупину і розпочинає виконування підпрограми обробки переривання, в котрій до стека завантажується адреса команди, яка слідує за командою WAIT чи STOP. Тому при поверненні до основної програми, після завершення виконування підпрограми обробки переривання (команда RTI), МК не повернеться до режиму очікування чи зупину. Задля запобігання зміни режиму функціонування підпрограма обробки переривання перед виконанням команди RTI має перевірити значення цього біта, і, в разі потреби, скоригувати значення адреси повернення. Задля скидання цього біта слід записати 0 до цього біта регістра SBSR.

Регістр SBFCR (адреса $FE03) – регістр дозволу змінювання стану периферійних модулів. Також має лише один активний біт. Встановлення біта BCFE = 1 дозволяє при обробці переривань в контрольній точці змінювати стан всіх периферійних модулів. При значенні біта BCFE = 0 стан периферійних модулів в перебігу налагодження програми буде зберігатися.

Модуль керування зовнішнім перериванням IRQ08 обслуговує зовнішні запити переривання, які надходять на вхід IRQ#, відповідно до обраного режиму. До складу модуля входить регістр керування зовнішніми перериваннями ISCR (адреса $001A), формат якого подано на рис. 13.18.

7	6	5	4	3	2	1	0
0	0	0	0	IRQF	ACK	IMASK	MODE

Рисунок 13.18 – Формат вмісту регістрів керування зовнішніми перериваннями ISCR

До цього регістру входять чотири біти, які мають таке призначення:

– IRQF – ознака зовнішнього запиту переривання (є доступний лише для читання), значення IRQF = 1 встановлюється при надходженні запиту переривання на вхід IRQ#;

– ACK – біт підтвердження приймання запиту переривання (є доступний лише для запису), при запису до нього 1 відбувається скидання біта IRQF;

– IMASK – біт маски зовнішнього переривання. При встановленні IMASK = 1 надходження зовнішнього запиту не призводить до переривання виконуваної програми;

– MODE – біт визначення виду сигналу переривання; встановлення значення MODE = 1 спричинює переривання при надходженні на вхід IRQ# низького рівня сигналу.

Модуль переривання в контрольній точці BREAK08 використовується при налагодженні програмного забезпечення для реалізації режиму зупину у контрольній точці.

Переривання такого типу відбувається, якщо при налагодженні програмного забезпечення сформована процесором адреса збігається з 16-розрядною адресою контрольної точки, записаної в регістрах BRKH-BRKL модуля BREAK08**.** При цьому, замість адреси наступної команди, формується адреса програмного переривання SWI, при обслуговуванні якого зупиняється

робота таймерних модулів та вартового таймера. Виконання програмного переривання SWI є немаскованим, тому відбувається незалежно від значення біта I регістра прапорців.

Для керування модулем використовується регістр BRKSR, який входить до складу модуля. Формат регістра подано на рис. 13.19.

7	6	5	4	3	2	1	0
BRKE	BRKA	0	0	0	0	0	0

Рисунок 13.19 – Формат вмісту регістру BRKSR модуля BREAK08

До складу цього регістра входять два активних біти – BRKE та BRKA, котрим притаманні такі особливості роботи:

– BRKE – ознака дозволу переривання в контрольній точці; значення BRKE = 1 дозволяє переривання, а BRKE = 0 – забороняє;

– BRKA – ознака збігу адреси поточної команди з вмістом регістрів BRKH-BRKL$, встановлення значення BRKA = 1 призводить до переривання в контрольній точці.

При функціонуванні модуля також використовуються значення бітів BCFE та SBSW в модулі системної інтеграції.

Модуль контролю напруги живлення LVI08 використовується для контролю напруги живлення. Якщо при роботі МК напруга живлення знижується понад задане значення, то модуль контролю напруги живлення LVI08 переводить МК до початкового стану, котрий зберігається до встановлення нормального рівня напруги.

У більшості серій сімейства до складу модуля LVI08 входить регістр стану LVISR, вміст якого можна лише зчитувати. Формат регістра подано на рис. 13.20.

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

Рисунок 13.20 – Формат вмісту регістра стану LVISR

Біт LVIOUT цього регістра набирає значення LVIOUT = 1, якщо напруга живлення зменшується понад рівень 4,0...4,3 В, за номінальної напруги живлення 5 В і до рівня 2,4...2,7 В за номінального значення 3 В.

Модуль продовжує роботу в режимі очікування і відключається в режимі зупину.

Модуль контролю функціонування COP08 контролює виконування програми за допомогою вартового таймера.

Робота вартового таймера полягає в тому, що до регістру керування COPCTL (адреса $FFFF) періодично записується довільне число. Запис повинен відбуватися не рідше одного разу за час T_w. Цей час визначається значенням біта COPRS у регістрі конфігурування CONFIG1 і становить:

$T_w = 262128 \cdot T_q$ – при значенні біта COPRS = 0;

$T_w = 8176 \cdot T_q$ – при значенні біта COPRS = 1.

Якщо такий запис не буде виконуватися, то вартовий таймер виконає перезапуск МК. Такий запис зручно виконувати командою

```
STA $FFFF        ; запис вмісту акумулятора до регістра з адресою $FFFF
```

Цю команду слід долучати до тексту програми задля періодичного виконування.

Вартовий таймер продовжує свою роботу в режимі очікування, і тому, залежно від його поточного стану перед надходженням команди WAIT, може статися його незапланованe спрацьовування.

В режимі зупину вартовий таймер припиняє роботу, але зберігає значення поточного стану, тому має сенс перед виконуванням команди STOP виконувати скидання цього модуля.

До складу послідовних інтерфейсних модулів входять:

Модуль інтерфейса з послідовною шиною USB08 забезпечує обмін даним по шині USB, яка використовується в обчислювальній техніці.

Модуль послідовного інтерфейса msCAN08 забезпечує обмін даними в стандартному та розширеному форматах відповідно до специфікацій CAN 2.0 A та CAN 2.0 B.

Модуль синхронного периферійного інтерфейса SPI08 забезпечує синхронний обмін даними зі швидкістю до 4 Мбіт/с. Цей модуль використовується для організації зв’язку поміж МК та іншими пристроями на незначній відстані.

Модуль асинхронного інтерфейса зв’язку SCI08 зреалізовує стандартний асинхронний протокол обміну 8- або 9-бітними даними з одним стартовим та одним стоповим бітами зі швидкістю до 130 кбіт/с.

Блок паралельних портів вміщує від 2-х до 8-ми паралельних 8-розрядних портів (PA, PB, ..., PG, PH). Виводи деяких портів можуть використовуватися задля виконання альтернативних функцій: організації послідовного обміну даними, обміну сигналами таймерних модулів, введення аналогових сигналів тощо.

До блока периферійних модулів для різних серій можуть входити:

Таймерний модуль TIM08 призначений для формування часових інтервалів, потрібних для керування МК та іншими пристроями, які входять до складу системи, в цілому. Він побудований на базі 16-розрядного лічильника, який, в різних моделях, може мати 2, 4 чи 6 каналів, які можуть працювати в режимі захоплення чи збігу з наперед окресленим значенням часу спрацьовування. Канали мають входи сигналів захоплення IC і відповідні регістри захоплення та порівняння, а також виходи сигналів збігання. Лічильник може працювати з вхідними сигналами, які надходять від тактового генератора модуля CGM08 – режим таймера (з можливістю його зупину та запуску) або з сигналами від зовнішніх пристроїв – режим лічби зовнішніх подій. Окрім того, пари каналів таймера можуть використовуватися задля

формування сигналів широтно-імпульсної модуляції (ШІМ). Більшість моделей сімейства вміщує два незалежних таймерних модуля.

Модуль базового таймера TBM08 забезпечує періодичне формування сигналів запиту переривання при переповненні базового лічильника. Він вміщує 15-розрядний лічильник, на вхід якого надходять сигнали кварцового резонатора. В режимі очікування цей таймер продовжує функціонувати і забезпечує перехід МК до робочого режиму. В режимі зупину таймер може продовжувати працювати, якщо дозволено роботу генератора CGM08 і формування сигналу запиту переривання.

Модуль таймера періодичних переривань PIT08 використовується для періодичного формування сигналів запиту переривання. Модуль побудовано на базі 16-розрядного лічильника і забезпечує формування сигналів запиту переривань в широкому діапазоні значень періоду цих сигналів. В режимі очікування цей таймер продовжує функціонувати і забезпечує перехід МК до робочого режиму при переповненні лічильника, а в режимі зупину цей модуль відключається.

Модуль широтно-імпульсного модулятора PWM08 вміщує 12-розрядний 6-канальний ШІМ модулятор.

Модуль аналого-цифрового перетворення ADC08 використовується для аналого-цифрового перетворення вхідних аналогових сигналів. В більшості моделей зреалізовано 8-розрядне перетворення аналогового сигналу, а в деяких моделях – 10-розрядне. Кількість аналогових входів в різних моделях може становити від 4 до 15.

Модуль прямого доступу до пам’яті DMA08 забезпечує роботу зі швидкодіючими зовнішніми пристроями.

Процесорний модуль CPU08 є модифікованим варіантом процесора CPU05. Він має розширений набір команд та способів адресування і програмно цілковито є сумісний з CPU05. Регістрову модель процесора CPU08 подано на рис. 13.21.

До регістрової моделі входять:

– 8-розрядний акумулятор A;

– 16-розрядний індексний регістр H:C. Для забезпечення програмної сумісності з CPU05 індексний регістр складається з двох частин, роздільне використовування яких дозволяє забезпечувати функціонування програм МК сімейства M68HC05/705;

– 16-розрядний програмний лічильник PC;

– 16-розрядний вказівник стека SP;

– 8-розрядний регістр прапорців CCR. Вміщує такі ознаки, які формуються у процесорі CPU05. До них додано ознаку [illegible] таку пе[illegible]но = 1, як[illegible] результат має кількість розрядів більшу, ніж можли[illegible]о розмістити їх в розрядній сітці МК.

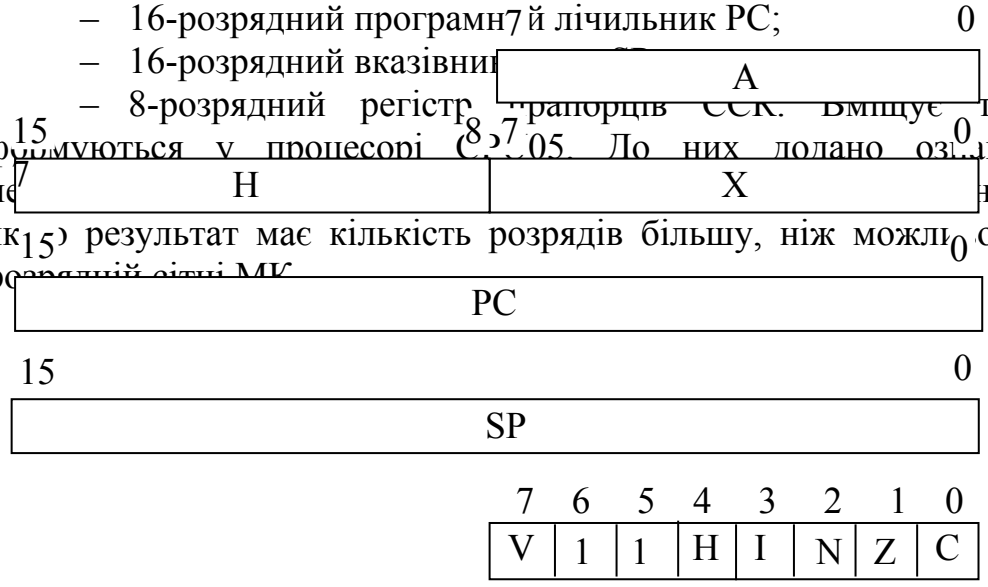


Рисунок 13.21 – Регістрова модель процесора CPU08

Обсяг адресного простору, який може адресувати МК сімейства M68HC08/908, становить 64 кбайти. Деякі МК сімейства мають менший обсяг адресного простору, тому в їхньому адресному просторі є області адрес, які не відповідають певним пристроям. При зверненні до таких адрес зреалізовується переривання, яке відповідає наявності помилки у програмі – зверненні до неіснуючої комірки. Внаслідок цього відбувається перезавантаження МК, до програмного лічильника PC автоматично завантажується адреса першої команди обробки переривання з двох останніх комірок адресного простору ($FFFE – $FFFF).

При перезавантаженні МК у вказівник стека автоматично записується адреса вершини стека ($00FF), яка забезпечує використовування в якості стека комірок пам’яті ОЗП та регістрів, адреси яких містяться у діапазоні ($0000...$00FF). При роботі користувач має змогу замінити цю адресу вершини стека на іншу в межах адресного простору ОЗП.

Сімейство MC68HC11/711

Найбільш досконалим сімейством 8-розрядних МК є MC68HC11/711. Сьогодні випускається близько 20 різних моделей МК цього сімейства. До складу структурної схеми МК входять модулі, які за функціональним призначенням є аналогічні до описаних вище. Різні моделі сімейства мають однакове процесорне ядро і відрізнюються обсягом та типом внутрішнього запам’ятовувального пристрою, складом периферійного обладнання й деякими іншими характеристиками. Особливістю цього сімейства є можливість підключення зовнішньої пам’яті обсягом від 64 кбайт до 4 Мбайт.

МК сімейства MC68HC11/711 у своєму складі мають внутрішню пам’ять програм – ПЗП – у складі MC68HC11 або РПЗП – у MC68HC711, обсягом до 32 кбайт; ОЗП даних обсягом від 192 до 1024 байт. Деякі моделі мають внутрішній РПЗП-ЕС обсягом до 540 байт.

При використовуванні МК цього сімейства можна зорганізовувати роботу в двох режимах – автономному і розширеному. При роботі в автономному режимі використовується лише внутрішня пам’ять МК, а в розширеному

дозволяється підключення зовнішньої пам’яті, робота з якою відбувається за допомогою мультиплексованої чи окремої зовнішньої шини адреси/даних. В деяких моделях передбачено розширення зовнішньої пам’яті і можливість зорганізовувати банки пам’яті.

Всі моделі мають в своєму складі 16-розрядний таймер, котрий може мати 3-4 входи сигналів захоплення частоти (IC) і 4-5 входів сигналів збігу частот. Цей таймер також використовується задля формування сигналів періодичних переривань. Окрім таймера, до складу МК входить 8-розрядний лічильник зовнішніх подій.

Деякі МК вміщують 8-розрядні широтно-імпульсні модулятори, які мають 4 входи, що вони можуть працювати в режимі 16-розрядних широтно-імпульсних модуляторів з двома входами.

МК сімейства вміщують від 4-х до 10-ми паралельних 8-розрядних паралельних портів, асинхронний і синхронний послідовні порти SCI (SCI+) і SPI.

До складу більшості моделей входить 8-розрядний АЦП з 8 аналоговими входами.

Процесорне ядро сімейства складається з процесора 68HC11, регістрову модель котрого зображено на рис. 13.22.

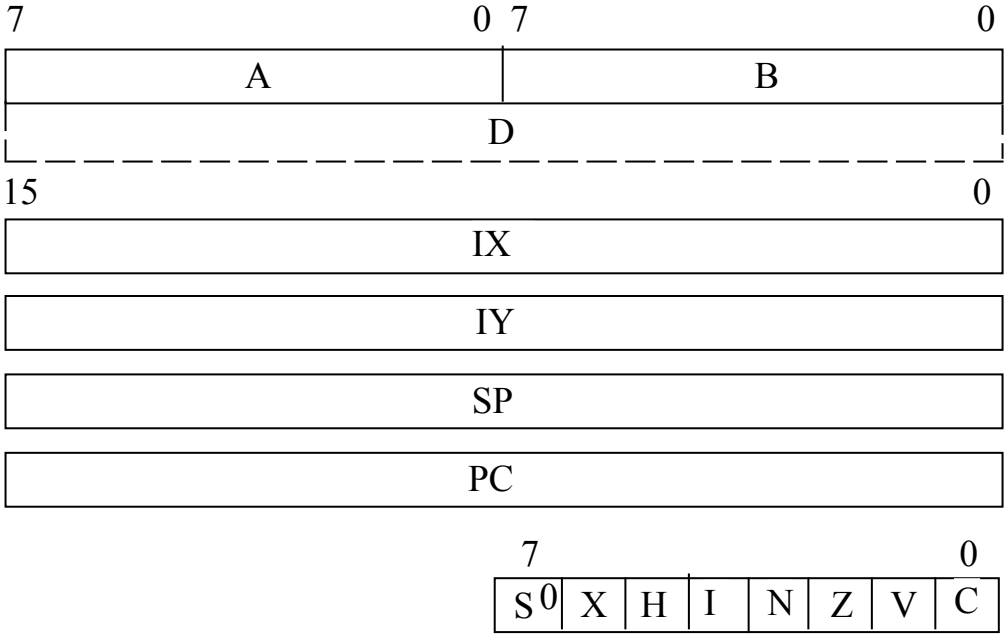


Рисунок 13.22 – Регістрова модель процесора 68HC11

До складу регістрової моделі процесора 68HC11 входять:

– два 8-розрядні акумулятори A і B, котрі при виконуванні деяких команд об’єднуються у 16-розрядний регістр D;

– два 16-розрядні індексні регістри IX і IY, котрі використовуються для формування адреси при індексному адресуванні операндів;

– 16-розрядний вказівник стека SP;

– 16-розрядний програмний лічильник PC;

– 8-розрядний регістр прапорців CCR, котрий вміщує ознаки H, I, N, Z, V і C, призначення й використовування котрих є аналогічні до 8-розрядних МК. До складу цього регістра до CPU12 додатково введено ознаку X, встановлення якої X = 1 забороняє обслуговування зовнішнього запиту переривання по входу XIRQ#, і ознаку S, яка при встановленні S = 1 забороняє переключення МК до режиму зупину при надходженні команди STOP.

Процесор виконує обробку 8- і 16-розрядних операндів і зреалізовує 108 команд. При роботі з операндами він використовує такі способи адресування як сімейство 68HC05/705; більшість команд, виконуваних МК, є аналогічні до команд сімейства 68HC05/705. Відмінності зумовлено наявністю двох акумуляторів. Додатково до системи команд введено команди ділення і розширено можливості команд бітових операцій.

МК сімейства можуть працювати в чотирьох режимах, які зумовлюються зовнішніми сигналами MODB і MODA, які приймаються при початковому завантаженні МК, а також вмістом регістру HPRIO. Режими функціонування визначаються відповідно до табл. 13.2.

Таблиця 13.2 – Режими функціонування МК сімейства MC68HC11/711

Режим	Зовнішні сигнали		Вміст регістру HPRIO				
	MODB	MODA	RBOOT	SMOD	MDA	IRV	PSEL3-0
Робочий автономний	1	0	0	0	0	0	0110
Робочий розширений	1	1	0	0	1	0	0110
Спеціальний – завантаження	0	0	1	1	0	1	0110
Спеціальний – тестування	0	1	0	1	1	1	0110

Автономний (однокристальний) робочий режим передбачає функціонування МК з використанням лише внутрішньої пам’яті, а в розширеному режимі передбачено підключення через порти B та C зовнішнього запам’ятовувального пристрою. При роботі у спеціальному режимі відбувається завантажування внутрішнього запам’ятовувального пристрою через послідовний порт SPI під керуванням спеціальної програми-монітора, яку розміщено у службовому ПЗП, або тестування МК заводом-виробником.

Режим функціонування і конфігурація МК визначаються вмістом службових регістрів HPRIO, CONFIG, OPTION та INIT, котрі входять до складу процесора. Формат вмісту цих регістрів подано на рис. 13.23. Для різних серій МК адреси цих регістрів та їхній склад не збігаються, але призначення бітів є однаковим.

HPRIO	RBOOT	SMOD	MDA	IRVNE	PSEL3	PSEL2	PSEL1	PSEL0
OPTION	ADPU	CSEL	IRQE	DLU	CME	FCME	CR1	CR0
INIT	RAM3	RAM2	RAM1	RAM0	REG3	REG2	REG1	REG0
CONFIG	0	0	0	0	NOSEC	NOCOP	ROMON	EEON

Рисунок 13.23 – Формат вмісту регістрів конфігурування
МК MC68HC11/711

До складу регістра HPRIO входять біти:

– RBOOT – біт дозволу звернення до програми-монітора, яка виконує завантажування ОЗП із зовнішнього джерела. З табл. 13.2 видно, що при роботі в спеціальному режимі завантаження значення цього біта встановлюється RBOOT = 1;

– SMOD разом з бітом MDA, відповідно до табл. 13.2, визначають режим функціонування МК після початкового завантаження;

– IRVNE – в розширеному режимі значення IRVNE = 1 дозволяє видавання даних на виводи порту С при зверненні до внутрішньої пам’яті; у автономному режимі та спеціальних режимах значення IRVNE = 1 забороняє виведення тактових сигналів на вивід Е (для зменшення рівня завад в системі);

– PSEL3-0 – визначають пріоритет обслуговування маскованих запитів переривань.

При початковому завантажуванні біти RBOOT, SMOD, MDA, IRVNE та PSEL3-0 набирають значень відповідно до табл. 13.2. Після запуску значення біта RBOOT і бітів SMOD та MDA у спеціальному режимі можна змінювати. Значення біта IRVNE можна одноразово змінювати у будь-якому режимі. До бітів PSEL3-0 можливе звернення в робочому режимі. До біта MDA також є можливе одноразове звернення в робочому режимі.

Регістр OPTION вміщує біти, які визначають функціонування окремих модулів МК:

– ADPU – значення ADPU =1 включає живлення модуля АЦП;

– CSEL – значення біта CSEL = 1 дозволяє використовування внутрішнього RC-генератора задля формування синхросигналів забезпечення програмування РПЗП-ЕС та роботу АЦП; значення CSEL = 0 відповідає використовуванню задля синхронізації тактових імпульсів МК;

– IRQE – визначає вид зовнішнього сигналу запиту переривань IRQ#; при значенні IRQE = 0 за активний вважається низький рівень вхідного сигналу IRQ#; IRQE = 1 визначає, що запит переривання здійснюється від зрізу сигналу IRQ#;

– DLU – значення DLU = 1 визначає затримку початку функціонування після виходу МК з режиму зупину на час близько 4000 тактів, при значенні біта DLU = 0 затримка становить 4 такти;

– CME – дозволяє функціонування, за CME = 1, схеми контролю тактової частоти;

– FCME – дозволяє функціонування, за FCME = 1, схеми контролю тактової частоти, за будь-якого значення біта CME; за значення FCME = 0 функціонування блока контролю тактової частоти визначається вмістом біта CME;

– CR1-0 – біти визначають значення Kw – коефіцієнта задля завдавання можливих інтервалів часу контролю для вартового таймера. Визначається відповідно до табл. 13.1.

При початковому завантаженні всі біти регістра OPTION скидаються і набирають значення 0, окрім біта DLU, який встановлюється в 1. При подальшій роботі у спеціальних режимах всі біти є доступними і в них може проводитися запис і відбуватися зчитування їхнього вмісту. В робочих режимах значення бітів ADPU, CSEL, CME може змінюватися, а значення інших бітів можна одноразово змінювати на протязі 64 тактів після запуску МК.

Вміст регістра INIT визначає розміщення в адресному просторі ОЗП даних і внутрішніх регістрів периферійних модулів. При початковому завантаженні біти RAM3-0, котрі визначають адресу ОЗП, набирають значення 0000, що забезпечує звернення до сторінки з адресою 0. Біти REG3-0 набирають значення 0001 і адресують сторінку з адресою 1, в якій розміщено адреси всіх внутрішніх регістрів периферійних модулів. При роботі в одному з робочих режимів дозволяється одноразовий запис до цього регістру протягом 64-х тактів після запуску.

Регістр CONFIG визначає конфігурацію МК і його біти мають таке изначення:

– NOSEC – захист внутрішніх ОЗП та РПЗП-ЕС від зовнішнього зчитування; значення біта NOSEC = 0 вказує на наявність такого захисту (встановлюється виробником, за замовленням користувача), значення біта NOSEC = 1 визначає відсутність такого захисту;

– NOCOP – керування роботою модуля контролю функціонування; значення NOCOP = 0 дозволяє функціонування цього модуля;

– ROMON – керування роботою внутрішнього ПЗП; значення біта ROMON = 1 дозволяє функціонування ПЗП, а значення ROMON = 0 спричинює звернення до зовнішньої пам’яті;

– EEON – керування роботою внутрішнього РПЗП-ЕС; значення біта EEON = 1 дозволяє функціонування РПЗП-ЕС, а значення EEON = 0 спричинює звернення до ОЗП або до зовнішньої пам’яті.

Регістр CONFIG програмується користувачем як РПЗП-ЕС, і тому він зберігає значення своїх бітів до перепрограмування.

13.1.2 16-розрядні мікроконтролери

Сімейство 68HC12/912

16-розрядні МК сімейства 68HC12/912 є основним промисловим стандартом 16-розрядних МК фірми Motorola. З 2002 року фірма розпочала випуск нового покоління цього сімейства – 68HCS12/912. Ці МК схарактеризовуються значним підвищенням продуктивності (тактову частоту збільшено до 40 МГц) і збільшенням обсягу внутрішньої Flash-пам'яті (до 512 кбайт).

До складу сімейства 68HC(S)12/912 входить низка МК, які відрізнюються один від одного типом та обсягом внутрішньої пам'яті, кількістю та типами периферійних пристроїв, які розміщено у мікросхемі.

В системах та пристроях ці МК можуть працювати в автономному та розширеному режимах. При роботі в автономному режимі (Single Chip) МК використовує лише внутрішню пам'ять задля зберігання програм і даних, а в розширеному режимі передбачено підключення зовнішнього запам'ятовувального пристрою і організація 8- або 16-розрядної системної шини (в більшості моделей – мультиплексованої).

МК сімейства 68HC12 мають модульну структуру, яка складається з низки стандартних функціональних блоків, взаємодія поміж якими зреалізовується по стандартизованій міжмодульній шині. Загалом до складу МК входять: 16-розрядний процесор CPU12, внутрішній оперативний запам'ятовувальний пристрій (ОЗП даних), РПЗП-ЕС, Flash-пам'ять або ПЗП, програмований маскою, модуль інтеграції LIM та набір периферійних модулів. Процесор CPU12 взаємодіє з цими модулями через їхні регістри, адреси котрих після запуску розміщуються у перших 512 комірках адресного простору.

Модуль інтеграції LIM використовується для реалізації службових функцій, до котрих належить:

– генерування потрібних тактових сигналів;

– початковий запуск і конфігурування МК;

– реалізація переривань при надходженні запитів від зовнішніх пристроїв, внутрішніх переривань процесора, періодичних переривань, а також зовнішніх запитів;

– реалізація інтерфейса із зовнішньою системною шиною у розширеному режимі;

– контроль функціонування МК за допомогою вартового таймера і моніторинга тактових імпульсів;

– налагодження програм за допомогою режиму BDM (Back-ground Debug Mode) і апаратного встановлення контрольних точок.

До складу блока периферійних модулів можуть входити:

– 8 – 12 паралельних портів задля обміну 8-розрядними даними. Виводи більшості портів можуть використовуватися для виконування інших функцій,

наприклад для передавання адреси та даних при роботі із зовнішньою пам’яттю, організації послідовних портів SCI, SPI таймера, АЦП, контролерів шин тощо;

– таймерні модулі TIM та ECT, котрі побудовано на базі 16-розрядного лічильника, а також 16-розрядний лічильник подій, який входить до складу деяких моделей;

– модулі аналого-цифрового перетворювання входять до складу всіх МК сімейства, мають 8 входів для приймання аналогових сигналів. В більшості моделей зреалізовано 10-бітове перетворювання аналогового сигналу;

– модуль формування сигналів широтно-імпульсної модуляції має чотири вихідних канали, значення парності для кожного з котрих можуть визначатися окремо;

– модулі послідовного асинхронного та синхронного інтерфейсів SCI та SPI, котрі працюють аналогічно до відповідних пристроїв 8-розрядних МК;

– спеціалізовані інтерфейсні модулі BDLC, CAN, I2C входять до складу різних моделей і зреалізовують обмін даними відповідно до мережних протоколів J1850, CAN 2.0A/B, I2C.

16-розрядний процесор CPU12, який входить до складу МК сімейства дозволяє виконувати обробку 8- і 16-розрядних операндів. Регістрова модель цього процесора є аналогічна до процесора CPU11, яку подано на рис. 13.22.

До складу регістрової моделі входять ті ж самі регістри, як у CPU11, які мають аналогічне функціональне призначення.

Програмний код CPU12 є суміщеним з кодом 68HC11 на рівні вхідного тексту.

Процесор CPU12 виконує всі способи адресування, які використовуються сімейством 68HC11/711 і, окрім того, може додатково зреалізовувати ще 7 додаткових варіантів індексного адресування.

Система команд процесора CPU12 вміщує 208 команд над операндами, які розміщуються у комірках пам’яті та регістрах, і є розширеним набором команд сімейства 68HC11/711.

Сімейство 68HC16/916

МК сімейства 68HC16/916 є подальшим розвиненням тенденцій, які було зреалізовано в 8- та 16-розрядних контролерах. Сімейство вміщує низку моделей, котрі відрізнюються типом та обсягом внутрішньої пам’яті та й номенклатурою периферійного обладнання, які входять до складу МК. У порівнянні з сімейством 68HC11/711 ці МК схарактеризовуються такими перевагами:

– розширений набір команд та способів адресування;

– введення спеціальних регістрів та команд для зреалізовування операцій цифрової обробки сигналів;

– підвищена продуктивність за рахунок збільшення тактової частоти до 25 МГц;

– збільшення до одного Мбайта обсягу адресного простору (через те що є дозволено роздільне звернення до пам’яті команд та пам’яті даних, то сумарний обсяг пам’яті може сягати двох Мбайтів);

– використовування периферійних пристроїв з розширеними функціональними можливостями;

– наявність вбудованих засобів налагодження програмного забезпечення.

МК сімейства 68HC16/916 мають модульну структуру і будуються зі стандартних функціональних модулів, до складу яких входять: процесор CPU16, модуль внутрішньої пам’яті, модуль системної інтеграції SIM або SCIM, модуль послідовного інтерфейсу QSM, багатоканальний комунікаційний інтерфейс MCCI, процесор часових інтервалів TPU, таймерні модулі GPT або CTM, аналого-цифровий перетворювач ADC. Деякі з цих модулів використовуються також у 32-розрядних МК фірми. Узагальнена структурна схема МК сімейства 68HC16/916 подана на рис. 13.24.

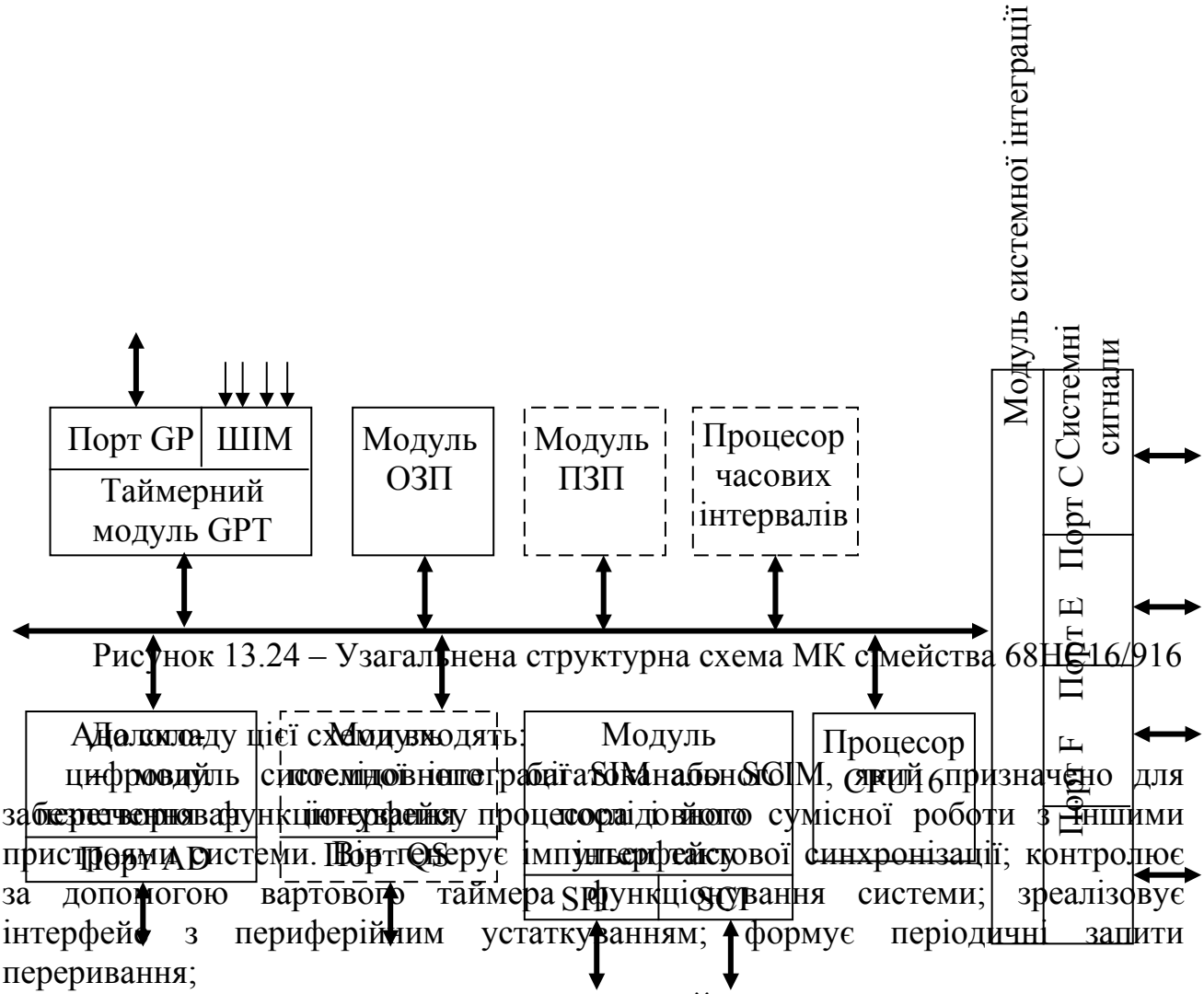


Рисунок 13.24 – Узагальнена структурна схема МК сімейства 68HC16/916

До складу цієї схеми входять: модуль системної інтеграції SIM або SCIM, який призначено для забезпечення функціонування процесора і його сумісної роботи з іншими пристроями системи. Він керує імпульсною та шинною синхронізації; контролює за допомогою вартового таймера функціонування системи; зреалізовує інтерфейс з периферійним устаткуванням; формує періодичні запити переривання;

– таймерний модуль GPT або CTM. Його побудова і функціональне призначення є аналогічні до таймерних модулівм сімейства 68HC11/711 та 68HC12/912. До складу модуля GPT входить пристрій для формування сигналів з широтно-імпульсною модуляцією;

– процесор часових інтервалів TPU, який входить до складу деяких моделей, призначений для виконування низки функцій формування часових інтервалів по 16-ти таймерних каналах без участі в них процесора;

– модуль багатоканального послідовного інтерфейсу MCCI, вміщує синхронний порт SPI та два асинхронних порти SCI. Їхнє функціонування є аналогічне до однойменних портів МК сімейства 68HC05/705 та 68HC11/711;

– модуль послідовного інтерфейсу QSM використовується в деяких моделях сімейства і вміщує один асинхронний порт SCI і модифікований варіант синхронного порту QSPI, котрий має в своєму складі буферну пам’ять, що надає змогу зорганізовувати передавання блоків даних обсягом до 32-х байт без участі процесора;

– модуль аналого-цифрового перетворювача виконує 8- або 10-бітне перетворювання сигналів, які надходять з 8-ми аналогових входів.

Процесор CPU16 виконує обробку 8-, 16- та 32-розрядних операндів і має систему команд, до складу якої входить 264 команди. Регістрову модель процесора наведено на рис. 13.25.

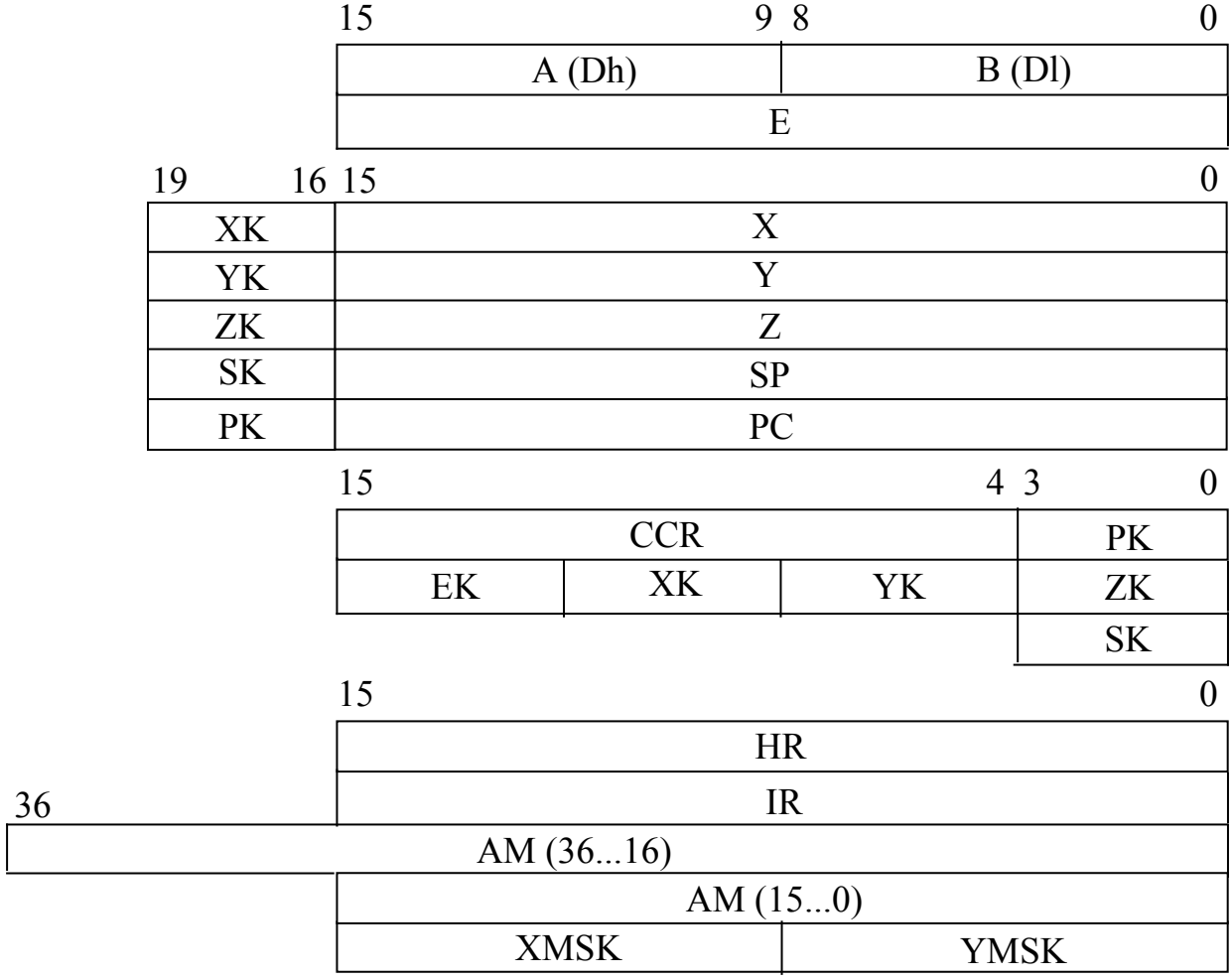


Рисунок 13.25 – Регістрова модель процесора CPU16

До регістрової моделі входять:

– два 8-розрядних акумулятора – A та B, котрі при обробці 16- та 32-розрядних операндів використовуються як один 16-розрядний акумулятор D;

– додатковий 16-розрядний акумулятор E;

– три 16-розрядних індексних регістри – X, Y, Z, котрі використовуються для адресування операндів;

– 16-розрядний вказівник стека SP;

– 16-розрядний лічильник команд PC.

Всі індексні регістри, вказівник стека і лічильник команд мають 4-розрядне розширення, відповідно XK, YK, ZK, SK, PK. Ці розширення використовуються для адресування 16-ти банків пам’яті, на котрі поділено адресний простір МК. Вибір банку здійснюється за допомогою 4-розрядного розширення адреси ЕК і значень відповідних розширень. Розширення ЕК, XK, YK, ZK поєднано в один 16-розрядний регістр К. Розширення лічильника команд РК розміщено у молодших розрядах регістра ознак CCR, а розширення вказівника стека SP – в окремому 4-розрядному регістрі SK.

До регістрової моделі також входять спеціалізовані регістри HR, IR, AM, MR, котрі використовуються при виконуванні послідовного множення-додавання дробових чисел при виконуванні цифрової обробки сигналів і два 8-розрядних регістри – XMSK та YMSK – задля зберігання масок, котрі визначають обсяг буферного модуля пам’яті при виконанні команд множення з накопиченням MAC.

Вміст регістра умов подано на рис. 13.26.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	MV	H	EV	N	Z	V	C	IP			SM	PK			

Рисунок 13.26 – Вміст регістру умов CCR

До його складу входять:

– РК – 4 розряди розширення програмного лічильника;

– SM – ознака “насичення” результату в акумуляторі AM;

– IP – маска запиту переривання;

– EV – ознака поширення результату в акумуляторі AM;

– MV – ознака переповнювання результату в акумуляторі AM;

Інші біти – S, H, N, Z, V, C – мають призначення, аналогічне до таких самих ознак процесора CPU11, 12.

Біти SM, EV, MV використовуються при виконуванні операцій цифрової обробки сигналів.

МК сімейства 68HC16/916 забезпечують конвеєризацію при виконуванні програм і черга з 6-ти команд зберігається у пристрої керування процесора. Тому поточне значення вмісту програмного лічильника разом з розширенням РК:PC, що визначає адресу наступної команди, має значення на 6 більше за значення старшого байта поточної команди.

13.1.3 32-розрядні мікроконтролери

Всі МК сімейства 683xx побудовано з використанням 32-розрядного процесорного ядра CPU32 і периферійних модулів. Ці пристрої поєднуються за допомогою стандартної міжмодульної шини. Більшість периферійних модулів є аналогічні до периферійних модулів МК сімейства 68HC16/916.

Архітектура МК 683xx базується на принципах, що їх було закладено в сімейство мікропроцесорів MC68000, що дозволяє мати обсяг адресного простору до 16 Мбайт і використовувати програмне забезпечення, яке розроблено для процесорів цього сімейства. Відповідно до особливостей архітектури, є певна різниця при виконуванні програм операційної системи і програм користувача. Тобто МК може функціонувати у двох режимах: режимі супервізора (виконування програм операційної системи) і режимі користувача (виконування програм користувача). За роботи в режимі супервізора при виконуванні програм дозволяється доступ до всіх ресурсів системи; команди, які виконуються лише в цьому режимі, називаються привілейованими. В режимі супервізора користувач має можливість звертатися до регістрів і комірок пам'яті, які входять до складу периферійних модулів. При включенні МК розпочинає функціонувати в режимі супервізора і за ініціалізації операційна система визначає режим подальшої роботи. Переведення до режиму супервізора здійснюється при встановленні відповідного біта в регістрі ознак процесора і при подальшій роботі режим не змінюється. Повернення до режиму користувача виконується лише при обробці виключних ситуацій або повторному перезапуску. Ці особливості відображено в регістровій моделі процесора CPU32.

Регістрову модель процесора CPU32 наведено на рис. 13.27.

До складу регістрової моделі входять:

– 2 групи 32-розрядних регістрів:

8 регістрів даних, які можуть працювати з даними, що є байтами, словами та довгими словами;

8 регістрів адреси, в котрих регістр A7 є дубльованим; він використовується в якості вказівника стека, для роботи в режимі супервізора має назву SSP, а при роботі в режимі користувача – USP.

– PC – 32-розрядний програмний лічильник; в процесорі CPU32 використовуються лише 24 розряди, відповідно до розрядності шини даних і обсягу адресного простору;

– SR – 16-розрядний регістр стану; складається з двох частин, кожна з котрих має розрядність: системного байта і байта користувача (регістру ознак – CCR). Регістр SR є повністю доступним в режимі супервізора. В режимі користувача доступ є можливий лише до байта користувача. Вміст регістра SR подано на рис. 13.28;

– регістри VBR, SFC, DFC є доступними лише в режимі супервізора. 32-розрядний регістр VBR вміщує базову адресу таблиці векторів виключень, завантаження цього регістра проводиться привілейованою командою MOVEC. Регістри SFC, DFC використовуються при виконуванні процесором

привілейованої команди MOVEC, котра зреалізовує пересилання даних поміж регістрами D7...0 або A7...0 і зовнішньою пам’яттю. Під час записування даних до пам’яті вміст регістра DFC використовується в якості функціонального коду FC2...0, а при завантажуванні регістрів в якості функціонального коду FC2...0 використовується вміст SFC. В такий спосіб відбувається організація додаткових банків пам’яті і розширюється загальний адресний простір.

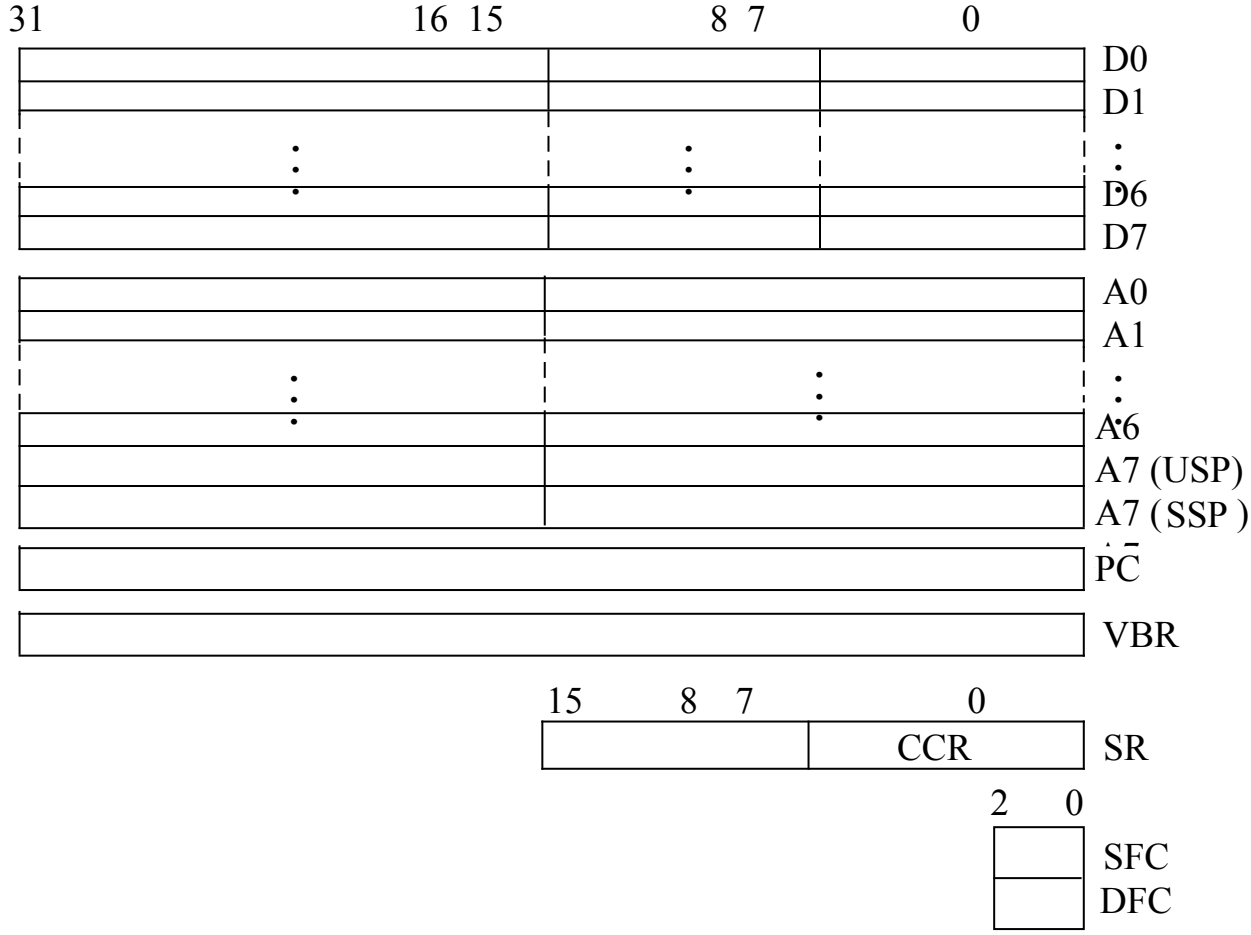


Рисунок 13.27 – Регістрова модель процесора CPU32

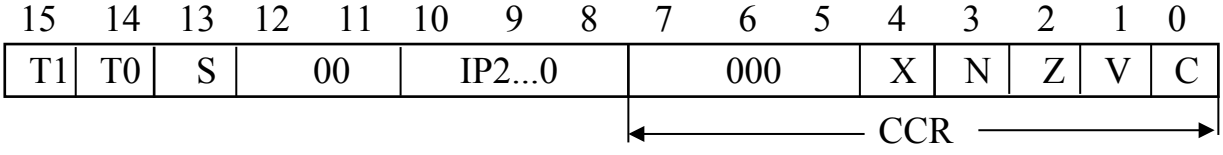


Рисунок 13.28 – Формат вмісту регістра стану SR

До регістра стану SR входять біти:

системні ознаки:

– T1 – ознака трасування по програмних переходах; значення T1 = 1 зупиняє виконування після кожної команди передавання керування (умовних, безумовних переходів);

– T0 – ознака трасування; встановлення біта T0 = 1 переключає процесор до покрокового режиму, програма зупиняється після виконання кожної команди;

– S – ознака супервізора, значення цієї ознаки S = 1 відповідає функціонуванню процесора в режимі супервізора;

– IP2-0 – маска запитів переривання, визначає рівень пріоритету обслуговування поточного запиту переривання;

ознаки користувача:

– X – ознака розширення результату, при виконуванні більшості операцій копіює значення біта C; у деяких випадках значення цього біта формується в залежності від виконуваної операції;

– N – ознака знаку; зберігає копію знакового розряду результату операції; значення ознаки N = 1 відповідає від'ємному результатові;

– Z – ознака нульового результату; при виниканні результату, який дорівнює 0, ознака набирає значення 1;

– V – ознака переповнювання; набирає значення V = 1 в разі переповнювання розрядної сітки при обробці операндів зі знаком;

– C – ознака перенесення; здійснює перенесення із старшого розряду при виконуванні арифметичних операцій та зберігає вміст розряду, який було висунуто, при виконуванні операцій зсувів; набирає значення C = 1 при виникненні перенесення із старшого розряду результату виконуваної операції.

Контрольні запитання:

1 Чим відрізнюються мікроконтролери різних моделей фірми Motorola?

2 Які пристрої можуть входити до складу мікроконтролерів?

3 Які регістри входять до складу процесорного ядра CPU05 і яке функціональне призначення кожного з них?

4 Які ознаки результату формуються при роботі МК сімейства M68HC05/705?

5 Які послідовні і паралельні порти входять до складу МК сімейства M68HC05/705?

6 Який обсяг має стек в МК сімейства M68HC05/705 і яку адресу має вказівник вершини при початковому завантаженні?

7 Яке призначення має блок конфігурації і які регістри до нього входять в МК сімейства M68HC05/705?

8 В чому полягає функціональне призначення бітів регістра MOR МК MC68HC705J1A і як відбувається змінювання їхнього вмісту?

9 Які регістри входять до складу блока конфігурування МК MC68HC705C8A?

10 Яке призначення має блок контроля функціонування МК сімейства M68HC05/705?

11 В чому полягає виконання процедури непрограмованого скидання МК сімейства M68HC05/705?

12 Як відбувається керування процесом обміну даними через асинхронний порт МК сімейства M68HC05/705?

13 Як відбувається керування процесом обміну даними через синхронний порт МК сімейства M68HC05/705?

14 Які сигнали формуються при організації синхронного обміну через порт SPI МК сімейства M68HC05/705?

15 Як побудовано багатофункціональний таймер в МК сімейства M68HC05/705?

16 Призначення бітів регістра керування-стану блока MFT?

17 Як визначаються часові інтервали встановлення ознаки періодичного переривання в блоці MFT?

18 Де зберігається адреса повернення до головної програми в період виконування підпрограми?

19 Чим відрізнюються системи синхронного та асинхронного послідовного обміну?

20 Які сигнали використовуються для організації синхронного обміну даними через порт SPI?

21 Які два генератори імпульсів входять до складу модуля формування тактових сигналів CGM 08?

22 Яке призначення має модуль системної інтеграції SIM 08?

23 Чим відрізнюються регістрові моделі процесорів CPU05 та 68HC11?

24 Які особливості побудови властиві 16-розрядним МК?

25 Які пристрої входять до складу МК сімейства 68HC16/916 і яке вони мають призначення?

26 Які регістри входять до складу регістрової моделі процесора CPU16?

27 Які регістри входять до складу регістрової моделі процесора CPU32?

28 Які МК можуть виконувати операції цифрової обробки сигналів?

Контрольні запитання підвищеної складності:

1 Як запрограмувати МК MC68HC705J1A для роботи із зовнішним генератором тактових імпульсів?

2 Запрограмуйте МК MC68HC705J1A на приймання зовнішніх запитів на переривання по каналах PA3...PA0.

3 Запрограмуйте режим непрограмного скидання у МК сімейства M68HC05/705.

4 Сформуйте керувальне слово, яке слід записати до регістру SCCR1 МК MC68HC705C8A для включення приймача при надходженні даних довжиною 1 байт зі значенням одиниці в старшому розряді і для запису значення біта контролю з прийнятого сигналу.

5 Що станеться з прийнятою інформацією, якщо в регістрі стану SCSR МК MC68HC705C8A буде записано:

TDRE	TC	RDFR	IDLE	OR	NF	RE	
0	1	1	1	1	0	0	–

6 Яким чином відбувалася робота вартового таймера МК MC68HC705J1A, якщо в регістрі керування-стану TSCR буде записано:

TOF	RTIF	TOIE	RTIE	TOFR	RTIFR	RT1	RT0
1	1	1	1	0	1	1	0

7 Назвіть причину перезавантаження МК сімейства M68HC08/908, якщо у регістрі стану LVISR буде записано:

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

8 Сформуйте керувальне слово для МК MC68HC11/711 для забезпечення:

– включення модуля АЦП та його синхронізації від генератора тактових імпульсів МК;

– переривання здійснюється при надходженні зрізу імпульса на вході IRQ#;

– дозволу функціонування схеми контролю тактової частоти;

– встановлення значення коефіцієнта $K_w = 4$.

13.2 Система команд мікроконтролерів фірми Motorola

Вхідний контроль:

1 Які особливості має мова Асемблер порівняно з мовами програмування високого рівня?

2 Які способи адресування операндів мають мікропроцесори фірми Motorola?

3 В який спосіб зберігаються слова в комірках пам’яті, які адресують мікропроцесори фірм Intel та Motorola?

4 Вміст якого регістра змінюється у командах передавання керування?

5 Для чого використовується вміст бітів регістра ознак у командах передавання керування?

Способи адресування і система команд сімейства M68HC05/705

Мікроконтролери цього сімейства, відповідно до регістрової моделі і структурної схеми МК (розд. 13.1), можуть зреалізовувати набір з 65-ти команд, які виконують обробку 8-розрядних операндів. Операнди можуть розміщуватися в регістрах A, X та пам’яті. Команди мають розмір від одного до 3-х байт: у першому байті розміщується код операції, а другий та третій

забезпечують адресування операндів, тобто можуть бути команди, які не вміщують операндів. МК зреалізовує такі способи адресування:

– регістрове – операнд розміщується в регістрі А або Х;

– непряме регістрове (індексне) – за адресу операнда слугує вміст регістру Х;

– індексне зі зміщенням – адреса операнда є сумою вмісту регістра Х та 8- або 16-розрядного зміщення (число без знаку), що задано в другому та третьому байтах команди;

– пряме – адреса операнда визначається значеннями другого та третього байтів команди;

– безпосереднє – 8-розрядний операнд (число) вказується у другому байті команди;

– відносне – використовується лише в командах розгалуження. Адреса наступної команди визначається як сума поточного вмісту програмного лічильника PC і 8-розрядного зміщення (число зі знаком), яке розміщено у другому байті команди.

При використовуванні непрямого регістрового і прямого адресування можливе є звернення лише до 256-ти початкових позицій адресного простору, у котрому розміщено всі регістри МК і значна частина ОЗП. Індексне зі зміщенням адресування при використовуванні 8-розрядного зміщення дозволяє звернення до 512-ти початкових позицій адресного простору, при використовуванні 16-розрядного зміщення – до всього адресного простору, але треба, щоби максимальне значення адреси не перевищувало $1FFF. При використовуванні відносного адресування можна звертатися до комірки, розташованої на 127 позицій вище або нижче за поточну. Способи адресування, залежно від сформованої адреси, поділяються на довгі та короткі. До коротких належать способи адресування: непряме-регістрове, пряме та індексне зі зміщенням при використовуванні 8-розрядного зміщення.

Всі команди, що входять до системи команд сімейства M68HC05/705, можна поділити на групи відповідно до операцій, які вони виконують. Тобто є команди пересилань, арифметичних та логічних операцій, зсувів, бітових операцій і встановлення ознак, керування програмою та процесором.

До **команд пересилань** належать команди, які виконують завантажування операндів з пам’яті до регістрів А, Х або завантажування комірок пам’яті з цих регістрів, пересилання операндів поміж регістрами А та Х, а також обнулення вмісту регістрів А, Х і комірок пам’яті, що адресуються короткими способами адресування. Перелік команд пересилань і способів подання операндів подано у табл. 13.3.

До **команд арифметичних операцій** належать команди, які виконують відповідні арифметичні операції над 8-розрядними операндами, один з котрих розміщується в акумуляторі А, а другий міститься у комірці пам’яті, адреса котрої визначається способом адресування в команді. Результат виконання операції неодмінно записується до акумулятора. Операції додавання та віднімання при виконуванні можуть використовувати значення біта перенесення С. До таких операцій також належать команди інкрементування та

декрементування, котрі, відповідно, зреалізовують додавання (віднімання) 1 до (від) вмісту акумулятора, індексного регістра або комірки пам’яті.

Таблиця 13.3 – Перелік команд пересиланнь

Назва	Мнемокод	Операція	Спосіб адресування
Команди завантаження регістра А	LDA #opr	#opr → A	Безпосереднє
	LDA $addr	(M) → A	Пряме коротке
	LDA $addr	(M) → A	Пряме довге
	LDA ,X	(M) → A	Непряме регістрове
	LDA $addr,X	(M) → A	Індексне зі зміщенням коротке
	LDA $addr,X	(M) → A	Індексне зі зміщенням довге
Команди завантаження регістра Х	LDX #opr	#opr → X	Безпосереднє
	LDX $addr	(M) → X	Пряме коротке
	LDX $addr	(M) → X	Пряме довге
	LDX ,X	(M) → X	Непряме регістрове
	LDX $addr,X	(M) → X	Індексне зі зміщенням коротке
	LDX $addr,X	(M) → X	Індексне зі зміщенням довге
Команди завантаження комірки пам’яті з регістра А	STA $addr	(A) → M	Пряме коротке
	STA $addr	(A) → M	Пряме довге
	STA ,X	(A) → M	Непряме регістрове
	STA $addr,X	(A) → M	Індексне зі зміщенням коротке
	STA $addr,X	(A) → M	Індексне зі зміщенням довге
Команди завантаження комірки пам’яті з регістра Х	STX $addr	(X) → M	Пряме коротке
	STX $addr	(X) → M	Пряме довге
	STX ,X	(X) → M	Непряме регістрове
	STX $addr,X	(X) → M	Індексне зі зміщенням коротке
	STX $addr,X	(X) → M	Індексне зі зміщенням довге
Пересилання вмісту регістра А в регістр Х	TAX	(A) → X	Регістрове
Пересилання вмісту регістра Х в регістр А	TXA	(X) → A	Регістрове
Обнулення вмісту регістра А	CLRA	$00 → A	Регістрове
Обнулення вмісту регістра Х	XLRX	$00 → X	Регістрове
Обнулення вмісту комірки пам’яті	CLR $addr	$00 → M	Пряме коротке
	CLR, X	$00 → M	Непряме регістрове
	CLR $addr, X	$00 → M	Індексне зі зміщенням коротке

МК також зреалізовує операцію беззнакового множення 8-розрядних даних, які розміщуються в акумуляторі A та індексному регістрі X; результат множення записується до пари регістрів X:A, отже, старший байт неодмінно записується до регістра X.

До команд арифметичних операцій також належать команди порівняння і тестування операндів. Існують дві групи таких команд: порівняння вмісту акумулятора A і вмісту комірки пам'яті та вмісту регістра X і комірки пам'яті. Команди обох груп виконуються як віднімання від вмісту відповідного регістра вмісту комірки пам'яті, при цьому прапорці змінюються відповідно до результату, а результат не формується.

Тестування виконується для операндів, які можуть розміщуватися в пам'яті, акумуляторі та індексному регістрі. Їхнє виконування полягає у відніманні числа $00 від операнда. При цьому формуються ознаки, що відповідають самому операндові.

Команди зміни знаку (формування доповнювального коду) також належать до цієї групи. В них можуть використовуватися лише короткі способи адресування операндів.

Перелік команд арифметичних операцій подано у табл. 13.4.

Таблиця 13.4 – Арифметичні команди

Назва	Мнемокод	Операція	Спосіб адресування
Додавання з перенесенням	ADC opr	(A)+(M)+C→A	Використовуються всі способи адресування і подання операндів, як показано в табл. 13.3
Додавання	ADD opr	(A)+(M)→A	
Віднімання з позикою	SBC opr	(A)–(M)–C→A	
Віднімання	SUB opr	(A)–(M)→A	
Інкрементування комірки пам'яті	INC $addr	(M)+1→M	Пряме коротке
	INC ,X		Непряме регістрове
Інкрементування регістра A	INCA	(A)+1→A	Регістрове
Інкрементування регістра X	INCX	(X)+1→X	Регістрове
Декрементування комірки пам'яті	DEC $addr	(M)–1→M	Пряме коротке
	DEC ,X		Непряме регістрове
Декрементування регістра A	DEC A	(A)–1→A	Регістрове
Декрементування регістра X	DEC X	(X)–1→X	Регістрове
Беззнакове множення	MUL	(X)×(A)→X:A	Як показано в таблиці

Закінчення табл. 13.4

Назва	Мнемокод	Операція	Спосіб адресування
Порівняння операндів	CMP opr	(A)–(M)	Використовуються всі способи адресування і подання операндів, як показано в табл. 13.3
	CPX opr	(X)–(M)	
Тестування операндів	TST $addr	(M)–$00	Пряме коротке
	TST ,X		Непряме регістрове
	TST $addr,X		Індексне зі зміщенням коротке
	TSTA	(A)–$00	Регістрове
	TSTX	(X)–$00	Регістрове

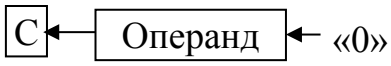
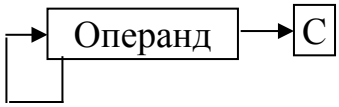
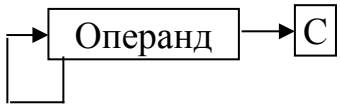
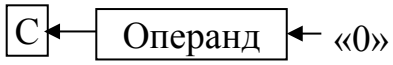
До **логічних операцій,** які виконує МК M68HC05/705, належать команди логічного множення (кон'юнкції), логічного додавання (диз'юнкції), виключного АБО та логічного інвертування, які виконуються з 8-розрядними операндами. Команда бітового тестування, котра також належить до цієї групи, виконує логічне множення операндів, але не формує результату. Перелік команд логічних операцій подано у табл. 13.5.

Таблиця 13.5 – Команди логічних операцій

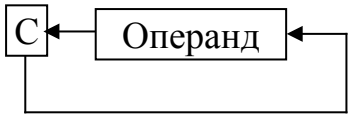
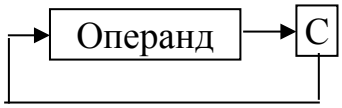
Назва	Мнемокод	Операція	Спосіб адресування
Логічне множення	AND opr	A $\wedge$(M)→A	Використовуються всі способи адресування і подання операндів, як показано в табл. 13.3
Логічне додавання	OR opr	A $\vee$ (M)→A	
Виключне АБО	EOR opr	A $\veebar$ (M)→A	
Логічна інверсія операнда	COM $addr	$FF–(M)→M	Пряме коротке
	COM ,X		Непряме регістрове
	COM $addr,X		Індексне зі зміщенням коротке
	COMA	$FF–(A)→A	Регістрове
	COMX	$FF–(X)→X	Регістрове

До групи **команд зсувів,** які зреалізовує МК M68HC05/705, належать команди арифметичних, логічних та циклічних зсувів, котрі виконуються над вмістом регістрів A, X та комірок пам'яті, що адресуються короткими способами адресування. Перелік команд зсувів та особливості їхнього виконування подано в табл. 13.6.

Таблиця 13.6 – Перелік команд зсувів

Назва	Синтаксис Асемблера	Способи адресування	Операція
Арифметичний зсув ліворуч	ASL $addr	Пряме коротке	
	ASL ,X	Непряме регістрове	
	ASL $addr,X	Індексне зі зміщенням коротке	
Арифметичний зсув ліворуч вмісту A	ASLA	Регістрове	
Арифметичний зсув ліворуч вмісту X	ASLX	Регістрове	
Арифметичний зсув праворуч	ASR $addr	Пряме коротке	
	ASR ,X	Непряме регістрове	
	ASR $addr,X	Індексне зі зміщенням коротке	
Арифметичний зсув праворуч вмісту A	ASRA	Регістрове	
Арифметичний зсув праворуч вмісту X	ASRX	Регістрове	
Логічний зсув ліворуч	LSL $addr	Пряме коротке	
	LSL, X	Непряме регістрове	
	LSL $addr, X	Індексне зі зміщенням коротке	
Логічний зсув ліворуч вмісту A	LSLA	Регістрове	
Логічний зсув ліворуч вмісту X	LSLX	Регістрове	
Логічний зсув праворуч	LSR $addr	Пряме коротке	
	LSR ,X	Непряме регістрове	
	LSR $addr,X	Індексне зі зміщенням коротке	
Логічний зсув праворуч вмісту A	LSRA	Регістрове	
Логічний зсув праворуч вмісту X	LSRX	Регістрове	

Закінчення табл. 13.6

Назва	Синтаксис Асемблера	Способи адресування	Операція
Циклічний зсув ліворуч	ROL $addr	Пряме коротке	
	ROL ,X	Непряме регістрове	
	ROL $addr,X	Індексне зі зміщенням коротке	
Циклічний зсув ліворуч вмісту A	ROLA	Регістрове	
Циклічний зсув ліворуч вмісту X	ROLX	Регістрове	
Циклічний зсув праворуч	ROR $addr	Пряме коротке	
	ROR ,X	Непряме регістрове	
	ROR $addr,X	Індексне зі зміщенням коротке	
Циклічний зсув праворуч вмісту A	RORA	Регістрове	
Циклічний зсув праворуч вмісту X	RORX	Регістрове	

Команди бітових операцій, які входять до системи команд МК M68HC05/705, встановлюють необхідне значення (0 або 1) відповідного біта b_n у 8-розрядному операнді, адреса котрого вміщується у другому байті команди (використовується лише пряме адресування). Номер біта *n* задається в команді. До цієї групи команд також належать команди встановлення необхідного значення прапорців C та I в регістрі ознак CCR. Перелік команд бітових операцій подано в табл. 13.7.

Таблиця 13.7 – Команди бітових операцій

Назва	Синтаксис Асемблера	Операція
Встановлення потрібного значення біта з номером n = (0...7)	BCLR n,$addr	$0 \rightarrow b_n$
	BSET n,$addr	$1 \rightarrow b_n$
Встановлення потрібного значення прапорця C	CLC	0→C
	SEC	1→C
Встановлення потрібного значення прапорця I	CLI	0→I
	SEI	1→I

До групи **команд керування програмою** входять команди, які зреалізовують:

– безумовний перехід у програмі;

– умовні переходи (умовні та безумовні розгалуження);
– перехід (розгалуження) до підпрограми;
– вихід з підпрограми;
– програмне переривання;
– повернення з програми обслуговування переривання.

Перелік команд цієї групи наведено у табл. 13.8.

Таблиця 13.8 – Команди керування програмою

Назва	Мнемокод	Операція
Безумовний перехід	JMP $addr	$(EA)_{корот.} \rightarrow PC$
	JMP $addr	$(EA)_{довг.} \rightarrow PC$
	JMP ,X	$(X) \rightarrow PC$
	JMP $addr,X	$(X) + \$addr_{корот.} \rightarrow PC$
	JMP $addr,X	$(X) + \$addr_{довг.} \rightarrow PC$
Умовний перехід	B_{CC} $rel8	(PC)+2+$rel8 → PC, якщо умова виконується
Безумовне розгалуження	BRA $rel8	(PC)+2+$rel8 → PC
Відсутність розгалуження	BRN $rel8	(PC)+2 → PC
Розгалуження при встановленні відповідного біта n	BRSET n,$addr,rel8	(PC)+2+rel8 → PC, якщо $b_n = 1$
	BRCLR n,$addr,rel8	(PC)+2+rel8 → PC, якщо $b_n = 0$
Перехід до підпрограми	JSR $addr	(PC)+n → PC (PCl)→SP; SP – 1→SP (PCh)→SP; SP –1→SP (EA)→PC
	JSR $addr	
	JSR ,X	
	JSR $addr,X	
	JSR $addr,X	
Розгалуження до підпрограми	BSR $rel8	(PC)+2 → PC (PCl)→SP; SP – 1→SP (PCh)→SP; SP –1→SP (PC)+2+rel8→PC
Вихід з підпрограми	RTS	SP+1→SP; (SP)→PCh SP+1→SP; (SP)→PCl
Програмне переривання	SWI	(PC)+1→PC (PCl)→SP; (SP) – 1→SP (PCh)→SP; (SP) –1→SP (X)→SP; (SP) – 1→SP (A)→SP; (SP) – 1→SP (CCR)→SP; (SP)–1→SP 1→I(CCR); Ст. байт Ve→PCh Мл. байт Ve→PCl

Закінчення табл. 13.8

Назва	Мнемокод	Операція
Вихід з програми обслуговування переривання	RTI	SP+1→SP; (SP)→CCR SP+1→SP; (SP)→A SP+1→SP; (SP)→X SP+1→SP; (SP)→PCh SP+1→SP; (SP)→PCl

Команда JMP завантажує до програмного лічильника адресу переходу, відповідно до способу адресування, що використовується.

В командах умовного переходу і розгалужень використовується лише відносний спосіб адресування. Адреса переходу формується як сума поточного вмісту програмного лічильника, збільшеного на 2 (задля звернення до наступної команди в програмі, що виконується), і 8-розрядного зміщення ($rel8) відносно початкової адреси розміщення програми, яке є числом зі знаком. Умови, що перевіряються, наведено в табл. 13.9. Команда безумовного розгалуження BRA є аналогом команди безумовного переходу з використанням відносного адресування, а команда відсутності розгалуження BRN є аналогічна до команди NOP. Вона пропускає у програмі два байти, після чого триває виконання програми. Команду BRN зручно використовувати при налагодженні програм замість команди B_{CC} для виконування програми без розгалуження.

Таблиця 13.9 – Мнемокоди і умови виконування команд умовних переходів

Мнемокод умови $_{CC}$	Умова, що перевіряється	Логічний вираз
NE EQ	Не дорівнює (ненульовий результат) Дорівнює (нульовий результат)	Z = 0 Z = 1
HI LS	Вище Нижче або дорівнює	(Z + C) = 0 (Z + C) = 1
HS LO	Вище або дорівнює (немає перенесення) Нижче (є перенесення)	C = 0 C = 1
PL MI	Додатний результат Від’ємний результат	N = 0 N = 1
HCC HCS	Немає міжтетрадного перенесення Є міжтетрадне перенесення	H = 0 H = 1
MC MS	Переривання дозволено Переривання заборонено	I = 0 I = 1
IH IL	Відсутність запиту переривання Надходження запиту переривання	IRQ# = 1 IRQ# = 0

Команди розгалуження при встановленні відповідного біта n BRSET і BRCLR перевіряють значення цього біта в операнді, визначеному за допомогою 8-розрядного прямого адресування. Зміщення в цих командах ($rel8) визначається відносно початкової адреси розміщення програми.

Команди переходу і умовного переходу до підпрограм JSR та BSR зберігають у стеку адресу команди повернення до головної програми і завантажують до програмного лічильника PC початкову адресу підпрограм, котра визначається за допомогою прямого, індексного чи індексного зі зміщенням адресування. При виконуванні команди JSR адреса, що завантажується до стека, має бути більшою за вміст регістру PC на 1, 2 або 3, залежно від способу адресування. При використовуванні індексного адресування це 1, при індексному адресуванні з 8-розрядним зміщенням чи короткому прямому – 2 і при адресуванні з 16-розрядним зміщенням і довгому прямому – 3. В команді BSR використовується відносне адресування. При поверненні до головної програми вміст регістра PC відновлюється зі стека.

Команда програмного переривання SWI до завантаження початкової адреси підпрограми обслуговування програмного переривання зберігає в стеку: значення програмного лічильника PC, акумулятора A, індексного регістра X та регістра ознак CCR. Потім встановлює значення біта дозволу переривання I = 1 і побайтно завантажує до програмного лічильника значення вектора переривань V_e. Команда повернення відновлює всі значення, які було завантажено до стека.

До **команд керування процесором** належать команди, які подано у табл. 13 10.

Таблиця 13.10 – Команди керування процесором

Назва	Мнемокод	Операція
Встановлення початкового значення вказівника стека	RSP	$00FF→SP
Відсутність операцій	NOP	(PC)+1 → PC
Перехід до режиму очікування	WAIT	Зупинення процесора 0→ I
Перехід до режиму зупину	STOP	Зупинення генератора тактових імпульсів 0→ I

Команда встановлення початкового значення вказівника стека завантажує адресу $00FF до регістра SP.

Команди WAIT та STOP переводять МК до режиму енергозберігання з можливістю повернення до нормального режиму, для чого при їхньому виконуванні встановлюються значення ознаки дозволу переривань I = 0.

Контрольні запитання:

1 Які прапорці виставляються при виконуванні арифметичних операцій?

2 Які прапорці виставляються при виконуванні логічних операцій?

3 Які способи адресування використовуються в мові Асемблер МК фірми Motorola?

4 Як поділяються способи адресування залежно від сформованої адреси? Чим вони відрізнюються?

5 За допомогою яких команд можна виконувати завантаження регістра акумулятора?

6 Які способи адресування можна використовувати задля зберігання вмісту індексного регістра X в пам'яті?

7 Чим відрізнюється використовування команд CMP та TEST?

8 Наведіть приклади виконування команд ASRA та LSRA з операндом, який дорівнює $82 на 2 розряди, і поясніть результати.

9 До якого регістра буде завантажено адресу переходу при виконуванні команди JMP ,X?

10 В який спосіб формується адреса нового вмісту програмного лічильника PC при виконуванні команди BRSET n,$addr,rel18?

Контрольні запитання підвищеної складності:

1 Які дії виконує МК при виконуванні команди програмного переривання SWI?

2 Які дії виконує МК при виконуванні команди повернення з програми обслуговування переривання RTI?

3 Які умови можуть використовуватись у команді умовного переходу B_{CC}?

4 Які команди додавання операндів у мові Асемблер МК фірми Motorola Ви знаєте? Чим вони відрізнюються одна від одної?

5 Чим різняться команди SBC і SUB? Поясніть на прикладі.

6 В яких регістрах розміщуються операнди при виконуванні команди MUL і де розміщується результат після її виконання?

13.3 Налаштовування вбудованих засобів мікроконтролерів

Вхідний контроль:

1 Які периферійні пристрої входять до складу мікроконтролера MC68HC705J1A?

2 Для чого використовується регістр MOR блока конфігурування МК MC68HC705J1A і яке призначення мають його окремі розряди?

3 Яке призначення має блок контролю функціонування МК MC68HC705J1A і які пристрої входять до його складу?

4 Яка різниця існує поміж послідовними і паралельними портами мікроконтролера?

5 Які порти входять до складу МК MC68HC705J1A?

6 Які регістри входять до складу паралельних портів МК MC68HC705J1A?

7 За допомогою яких команд можна здійснювати запис до регістра DDRA?

8 В який спосіб визначається напрямок обміну через паралельний порт МК MC68HC705J1A?

Виводи паралельних портів МК MC68HC705J1A можна використовувати для обміну будь-якими сигналами. При використовуванні окремих розрядів можна формувати на них сигнали у послідовному вигляді, тобто, МК можна використовувати в якості генератора послідовності імпульсів потрібної частоти і щілинності. Блок-схему алгоритму функціонування МК MC68HC705J1A в якості генератора імпульсів подано на рис. 13.29.

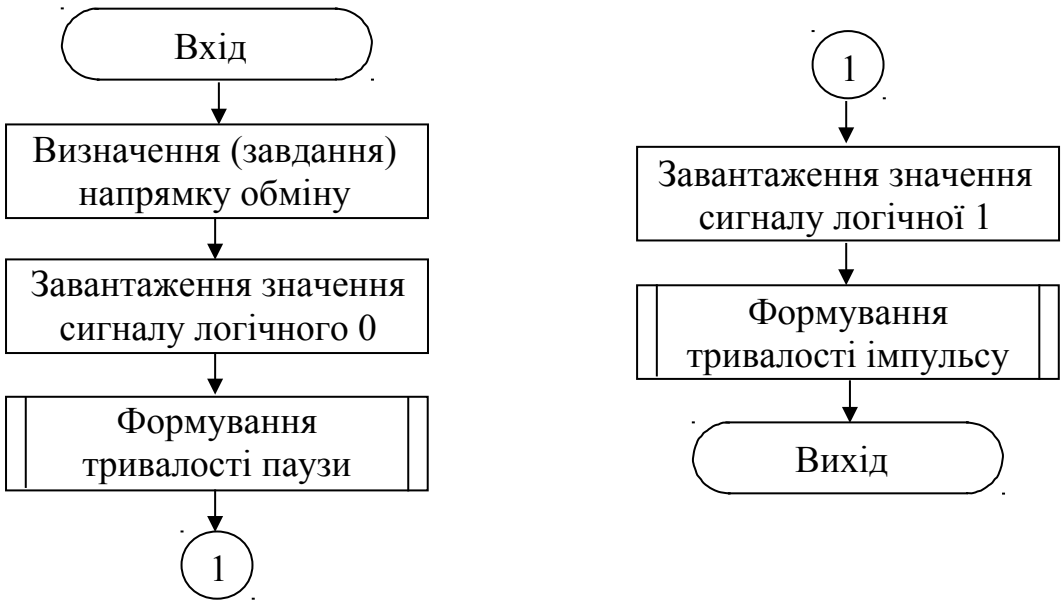


Рисунок 13.29 – Блок-схема алгоритму роботи MC68HC705J1A в якості генератора імпульсів

Визначення напрямку обміну зумовлюється встановленням відповідного біта в регістрі напрямку пересилання DDRA (адреса $0004) або DDRB (адреса $0005), як було показано в розділі 13.1.1. Вважаємо, що це будуть всі 8 виводів порту А.

Виведення здійснюється з регістра даних порту PORTA, до якого треба завантажити значення сигналу з акумулятора А. Виведення даних триватиме упродовж такого відтинку часу, допоки сигнали на виходах порту не змінять свого значення. Тому треба сформувати часовий інтервал, котрий визначатиме час формування імпульсу чи то паузи поміж імпульсами. Цей часовий інтервал формується за допомогою підпрограми формування тривалості; вважаємо, що послідовність розпочинається з паузи (рівень логічного 0). Зазвичай формування часових інтервалів може відбуватися за допомогою таймера чи то у програмний спосіб. При формуванні часового інтервалу в програмний спосіб розроблюється циклічна підпрограма, час виконування якої дорівнює заданому часовому інтервалові. При розв’язуванні нашого завдання ця підпрограма може бути побудована за алгоритмом, який подано на рис. 13.30. На цьому рисунку не зазначено блоки входу та виходу, тому що вважається, що робота цієї підпрограми здійснюється згідно з алгоритмом рис. 13.29.

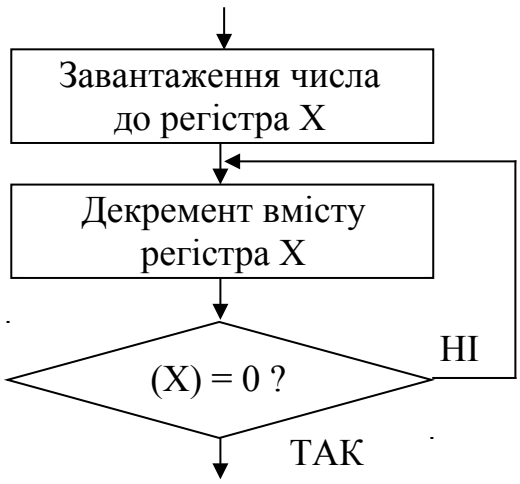


Рисунок 13.30 – Блок-схема алгоритму підпрограми часової затримки

Число, завантажуване до регістра визначає час затримки підпрограми.

Після виконання цієї підпрограми слід завантажити до регістра даних нове значення сигналу і сформувати часову затримку, що дорівнювала б часові виведення цього значення.

За алгоритмом рис. 13.29, з урахуванням алгоритму рис. 13.30, розроблюємо програму генератора імпульсної послідовності мовою Асемблер.

```
      LDA  #FF      ; Визначення напрямку обміну на виведення
      STA  $0004    ; даних каналом А
M3:   LDA  #00      ; Завантаження рівня сигналу, який
                    ; виводитиметься каналом А
      STA  $0000    ; Виведення каналом А сигналу логічного 0
      JSR  M1       ; Безумовний перехід до підпрограми затримки,
                    ; розташованої за міткою М1
      LDA  #FF      ; Встановлення рівня логічної 1 і виведення цього
      STA  $0000    ; сигналу каналом А
      JSR  M2       ; Безумовний перехід до підпрограми,
                    ; розташованої за міткою М3
      BRA  M3       ; Безумовний перехід до формування нового
                    ; періоду сигналу
M1:   LDX #8        ; Завантаження до індексного регістру Х кількості
                    ; циклів виконування підпрограми формування
                    ; сигналу логічного 0
M4:   DECX          ; Підпрограма формування часового інтервалу
                    ; паузи поміж імпульсами. Декремент вмісту
                    ; регістра Х
      BNE  M4       ; Умовний перехід до команди DECX
      RTS           ; Вихід з підпрограми формування часового
                    ; інтервалу паузи поміж імпульсами. Повернення
                    ; здійснюється до команди LDA #FF, що є
```

```
                    ; наступною за командою звернення до цієї
                    ; підпрограми
M2:  LDX #4         ; Завантаження до індексного регістра X кількості
                    ; циклів виконування підпрограми формування
                    ; сигналу логічної 1
M5:  DECX           ; Підпрограма формування одного імпульсу
                    ; з послідовності
                    ; Декремент вмісту регістру X
     BNE  M5        ; Умовний перехід до команди DECX
     RTS            ; Вихід з підпрограми формування часового
                    ; інтервалу паузи поміж імпульсами. Повернення
                    ; здійснюється до команди BRA
```

Ця програма формуватиме імпульсну послідовність, частота слідування імпульсів в котрій визначатиметься тактовою частотою МК.

Використовуючи таймер, який входить до складу МК, можна також формувати сигнали з потрібною часовою затримкою і використовувати ці сигнали задля керування потрібними об’єктами. Для формування потрібних часових інтервалів можна використовувати сигнали переривання від таймера або від периферійних пристроїв.

Припустімо, що треба розробити програму керування станом об’єкта (періодичне включення/виключеня) по переповнюванні 15-розрядного лічильника багатофункціонального таймера MFT, якщо не надходить сигнал зовнішнього переривання. В разі надходження сигналу зовнішнього переривання, який надходить на вивід 0 порту A, програма формує сигнал логічної 1 на виводі 7 порту A.

Програма, яка функціонуватиме за цим алгоритмом, матиме такий вигляд:

```
     CLRX           ; Обнулення індексного регістра X
     LDA #$80       ; Завантаження сигналу включення, який
                    ; виводитиметься у 8-му розряді порту A
     STA $0000      ; Включення зовнішнього пристрою по каналу A
     LDA  #$FE      ; Завантаження до регістра A коду, який
                    ; визначатиме виводи порта A PA7...PA1 як
                    ; виходи, а вивід PA0 – як вхід
     STA $0004      ; Завантаження коду з регістра A до регістра
                    ; напрямку пересилання DDRA
     BSET 5,$0008   ; Встановлення біта дозволу переривання
                    ; переповнювання лічильника у регістрі TSCR
     CLI            ; Скидання ознаки I в регістрі CCR (дозвіл
                    ; переривання)
M1:  JMP M1         ; Безумовний перехід на ту саму команду задля
                    ; формування затримки формування вихідного
                    ; сигналу до моменту переповнювання лічильника
                    ; у багатофункціональному таймері
```

```
M2:  BSET 3,$0008     ; Встановлення біта TOFR (біт скидання ознаки
                      ; переповнювання) у регістрі TSCR
     LDA $0000        ; Завантаження до акумулятора А вмісту регістра
                      ; даних PORTA (число #$80)
     EOR #$80         ; Обнулення вмісту акумулятора
     STA $0000        ; Відключення зовнішнього пристрою по каналу А
                      ; (виведення рівня логічного 0 каналом А)
     RTI              ; Вихід з підпрограми обслуговування
                      ; переривання (повернення до команди JMP M1)
M3:  BSET 1,$000A     ; Скидання ознаки IRQR (скидання запиту
                      ; на переривання) і скидання ознаки IRQF
                      ; (відсутність запиту на переривання) в регістрі
                      ; ISCR
     BSET 7,$0000     ; Виведення сигналу через вивід 7 PORTA
     RTI              ; Вихід з підпрограми формування сигналу на
                      ; виводі 7 PORTA
```

До складу цієї програми входять дві підпрограми, які позначено мітками M2 та M3. Підпрограма M2 призначена для формування сигналу відключення зовнішнього пристрою, який підключено до порту А. Повернення з цієї підпрограми відбувається до команди, позначеної міткою M1, яка формує часову затримку до моменту переповнювання лічильника багатофункціонального таймера MFT. Переповнювання лічильника призводить до встановлення внутрішнього переривання, яке забезпечує циклічне змінювання стану виводу 8 порту А. Для правильного функціонування програми до пам’яті слід завантажувати адреси, до яких здійснюватиметься звернення при виконуванні відповідних переривань.

```
$07F8    03   0F   03   18   03   00   03   00
```

У комірках пам’яті з адресами $07F8 та $07F9 розміщено початкову адресу підпрограми обробки переривання по переповнюванні лічильника – $030F, в комірках $07FA та $07FB – початкову адресу підпрограми обробки зовнішнього переривання – $0318 і в наступних комірках – початкову адресу розміщення всієї програми.

Звернення до підпрограми, поміченої міткою M3, здійснюється лише за наявності зовнішнього переривання.

Програма формує часові інтервали, що вони дорівнюють часові виконання 1025 та 1022 машинних тактів МК. Тривалість цих інтервалів можна змінити, встановлюючи потрібні значення RT1-0 в регістрі TSCR.

Протокол виконування програми в режимі обробки переривання по переповнюванні лічильника наведено в табл. 13.11. В протоколі не зазначено вміст регістрів, які не використовуються при виконуванні цієї програми: регістра даних порту В, регістра напрямку пересилання порту В та регістра PDRB, а також регістрів ISCR та PDRA, які не змінюють своїх значень, $80 та

$00, відповідно. В колонці CYCLES наведено кількість машинних циклів, яка відповідає виконуванню поточної команди.

Таблиця 13.11 – Протокол виконування програми періодичного включення/відключення периферійного пристрою за сигналом переповнення лічильника

Команда	A	X	PORTA	DDRA	TSCR	TIMER	CCR	CYCLES
CLRX	XX	$00	XX	$00	$03	$00	111.I…	$00
LDA #$80	$80	$00	XX	$00	$03	$00	111.I.Z.	$03
STA $0000	$80	$00	XX	$00	$03	$01	111.IN..	$05
LDA #$FE	$80	$00	XX	$00	$03	$02	111.IN..	$09
STA $0004	$FE	$00	XX	$00	$03	$02	111.IN..	$0B
Формування сигналу включення								
BSET 5,$0008	$FE	$00	$80	$FE	$03	$03	111.IN..	$0F
CLI	$FE	$00	$80	$FE	$23	$05	111.IN..	$14
JMP M1	$FE	$00	$80	$FE	$23	$05	111..N..	$16
…								
JMP M1	$FE	$00	$80	$FE	$23	$FF	111..N..	$3FD
JMP M1	$FE	$00	$80	$FE	$A3	$00	111..N..	$400
BSET 3,$0008	$FE	$00	$80	$FE	$A3	$03	111.IN..	$40D
LDA $0000	$FE	$00	$80	$FE	$23	$04	111.IN..	$412
EOR #$80	$80	$00	$80	$FE	$23	$05	111.IN..	$415
STA $0000	$00	$00	$80	$FE	$23	$05	111.I.Z.	$417
RTI	$00	$00	$00	$FE	$23	$06	111.I.Z.	$41B
Формування сигналу відключення								
JMP M1	$FE	$00	$00	$FE	$23	$09	111..N..	$424
…								
JMP M1	$FE	$00	$00	$FE	$23	$FF	111..N..	$7FC
JMP M1	$FE	$00	$00	$FE	$23	$FF	111..N..	$7FF
JMP M1	$FE	$00	$00	$FE	$A3	$00	111..N..	$802
BSET 3,$0008	$FE	$00	$00	$FE	$A3	$03	111.IN..	$807
LDA $0000	$FE	$00	$00	$FE	$23	$05	111.IN..	$814
EOR #$80	$00	$00	$00	$FE	$23	$05	111.I.Z.	$817
STA $0000	$00	$00	$00	$FE	$23	$06	111.IN..	$819
RTI	$80	$00	$80	$FE	$23	$07	111.IN..	$81D

Контрольні запитання:

1 В який спосіб можна задати час затримки, що він формується при виконуванні часової затримки?

2 В який спосіб можна здійснювати вихід з підпрограми часової затримки за різних способів її побудови?

3 Чим визначатиметься час затримки при використовуванні переривання за переповнювання лічильника?

4 Скільки розрядів має регістр-лічильник багатофункціонального таймера MFT?

Контрольні запитання підвищеної складності:

1 В чому полягає призначення бітів RT1-0 і як значення їхнього вмісту можна використовувати задля програмування вбудованих засобів?

2 Як визначити час формування МК певного інтервалу?

3 Напишіть програму керування (включення/відключення) певним об’єктом. Включення об’єкта відбувається формуванням сигналу логічної 1 на виводі 7 порту А, а відключення – формуванням сигналу логічного 0 на тому ж самому виводі. Включення відбувається при надходженні сигналу зовнішнього переривання, а відключення через певний час затримки, який формується за допомогою багатофункціонального таймера MFT.

14 RISC-ПРОЦЕСОРИ ФІРМИ MOTOROLA

Вхідний контроль:

1 Скільки байтів може займати команда мовою Асемблер CISC-процесорів?

2 Які особливості має архітектура CISC- та RISC-процесорів?

3 Які способи адресування операндів використовують 32-розрядні універсальні мікропроцесори фірми Motorola?

4 Які ознаки CISC- чи RISC-архітектури мають мікропроцесори Pentium фірми Intel?

5 Скільки регістрів входять до регістрового файла моделі користувача універсальних МП фірми Motorola?

14.1 RISC-процесори PowerPC

Назва RISC (Reduced Instruction Set Computing) означає “комп’ютер зі зменшеним набором команд”, але в сучасних розробках RISC-мікропроцесорів набір команд є значно розширений і може вміщувати команди оброблення даних з плаваючою точкою. Систему команд RISC-процесорів можна схарактеризувати як таку, яка має фіксований формат команд, що значно спрощує керувальний пристрій та зменшує обсяг мікропрограмної пам’яті і призводить до зменшення розміру кристала RISC-процесорів, зниження їхньої вартості та підвищення тактової частоти. Система команд є вільна від складних та рідко використовуваних команд і способів адресування. Введення фіксованого набору команд забезпечує високу ефективність роботи конвеєра, зменшує кількість тактів простоювання та очікування процесорів, що підвищує їхню ефективність.

Переваги та ефективність RISC-процесорів зумовили їхнє виробництво усіма провідними фірмами, які випускають мікропроцесори: Motorola, Intel, Hewlett-Packard, IBM, Sun Microsystems, MIPS.

За найоптимальніші RISC-процесори та RISC-контролери вважаються розробки консорціуму фірм Motorola, IBM, Apple Computers – MPC6XX PowerPC, а також розробки фірми Motorola – MFC5XXX ColdFire.

До основних особливостей RISC-процесорів можна віднести:

– розширений обсяг регістрової пам’яті: від 32-х до кількох сотень регістрів загального призначення;

– використовування в командах оброблення даних лише регістрового адресування; команди звернення до пам’яті використовуються лише при завантаженні та зберіганні вмісту регістрів, а також у командах керування програмою;

– відмову від апаратної підтримки складних способів адресування – з постінкрементом, предекрементом та деяких непрямих;

– фіксований формат команд, зазвичай чотири байти, замість змінного формату (від одного до 12-ти байт), що є характерним для CISC-мікропроцесорів;

– відсутність команд, які не відповідають прийнятому форматові.

За базову модель МП PowerPC можна вважати MPC604. У цій моделі, як і у сімействі MX68XXX, зберігається принцип виокремлення певних ресурсів задля розв'язування завдань супервізора та користувача. Це означає, що архітектура процесора вміщує окремі регістри, які входять і до моделі користувача і до моделі супервізора, і має привілейовані команди, які виконуються лише в режимі супервізора.

До регістрової моделі користувача (рис. 14.1), яка є спільною для всього сімейства PowerPC, входять:

– тридцять два 32-розрядні регістри загального призначення GPR31...GPR0 для зберігання цілочисельних операндів;

– тридцять два 64-розрядних регістри FPR31…FPR0 для зберігання операндів з плаваючою точкою;

– 32-розрядні регістри прапорців (умов) CR та станів (рис. 14.2, а) при обробці даних з плаваючою точкою FPSCP;

– 32-розрядні регістри XER, LR, CTR, які використовуються при обробці виключень.

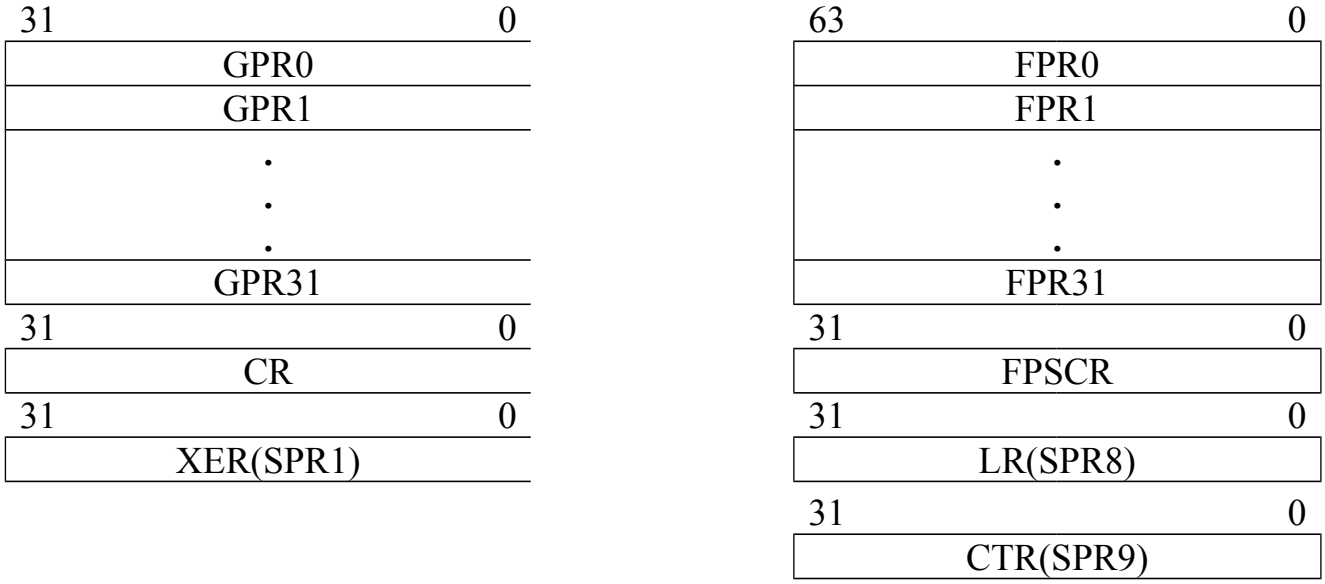
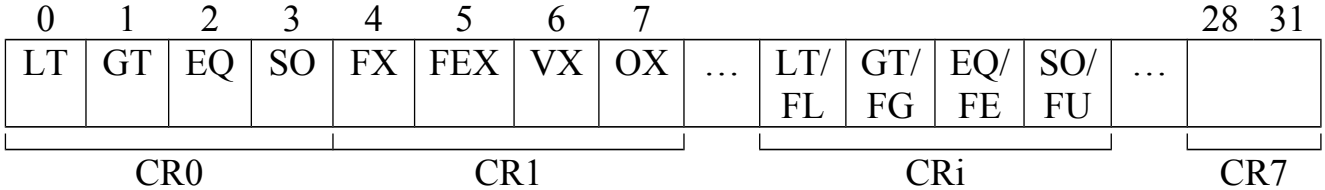


Рисунок 14.1 – Регістрова модель користувача
для мікропроцесорів PowerPC



а) CR

б) XER

Рисунок 14.2 – Формати вмісту регістрів CR та XER

Слід звернути увагу на те, що регістрова модель не вміщує ані програмного лічильника, ані вказівника стека.

Регістр прапорців CR складається з восьми 4-бітових полів CR7...CR0, молодший з яких CR0 вміщує прапорці, які формуються внаслідок цілочисельних операцій:

- LT = 1 за від'ємного результату;
- GT = 1 за додатнього ненульового результату;
- EQ = 1 за нульового результату;
- SO – набирає значення аналогічного біта у регістрі XER (рис. 14.2, б).

Поле CR1 вміщує прапорці, які встановлюються при операціях з плаваючою точкою. Вміст інших полів CR7...CR2 встановлюється за результатом операцій порівняння цілих чи дійсних, при операціях з плаваючою точкою, чисел. При порівнянні цілих чисел формуються прапорці LT (менш), GT (понад), EQ (дорівнює), SO. При порівнянні дійсних чисел встановлюються аналогічні прапорці FL (менш), FG (понад), FE (дорівнює), а також FU, який встановлюється у 1, коли один з порівнюваних операндів є не-число. Наявність кількох результатів порівняння дозволяє водночас зберігати їх і використовувати за потреби.

Нумерацію розрядів RISC-процесорів PowerPC прийнято в напрямку праворуч → ліворуч, розпочинаючи з 0-го розряду, який є старшим, тобто D0 – старший розряд даних, а D31 – молодший. Відповідно, вміст регістрів мікропроцесора подається у форматі, коли старший біт має номер 31.

Регістр XER вміщує прапорці перенесення CA, переповнення OV, які встановлюються при виконуванні арифметичних цілочисельних операцій, та прапорець загального переповнення S0, який зберігається таким, що дорівнює 1 до виконування команд завантаження до регістру нового даного.

Регістр зв'язку LR зберігає адреси команд умовного та безумовного переходів чи звернення до підпрограми.

Регістр-лічильник CTR вміщує задану кількість циклів і при виконуванні чергового циклу його вміст зменшується на 1.

На рис. 14.1 у дужках подано також номери певних поіменованих регістрів, які використовуються у певних командах звернення до цих регістрів як до службових.

Регістрова модель супервізора, окрім регістрової моделі користувача, вміщує кілька груп службових регістрів, основним з яких є регістр керування MSR, який визначає режим функціонування процесора. Режими функціонування процесора можуть бути такими:

- зі зниженим енергоспоживанням;
- з встановленням порядку big-endian або little-endian оброблення байтів при виключеннях;
- з обслуговуванням зовнішніх запитів переривань;
- режим користувача чи режим супервізора;
- режим виконування операцій над числами з плаваючою точкою;
- режим трасування;
- режим задавання базової адреси векторів переривань;

– режим дозволу трансляції адрес команд та даних за допомогою пристроїв керування пам’яттю IMMU, DMMU;

– режим з контролем продуктивності процесора тощо.

У процесорі передбачено заходи щодо контролю функціонування кешів команд та даних. Можливе є задавання режимів послідовного та паралельного виконування команд у суперскалярній структурі, статичного та динамічного передбачення подальшого перебігу програми.

Окремі службові регістри мікропроцесора визначають серійний номер та код ідентифікації процесора в мультипроцесорних системах.

Процесор вміщує таймер базового часу, який перемикається тактовою частотою.

На рис. 14.3 подано структуру МП MPC604. Процесор має суперскалярну структуру, яка складається з шести виконавчих пристроїв, які працюють паралельно:

– блок оброблення розгалужень BPU;

– два блоки для виконування одноциклових цілочисельних операцій SIU1, SIU2;

– один пристрій для виконування багатоциклових цілочисельних операцій MIU;

– пристрій оброблення даних з плаваючою точкою FPU;

– блок вибирання операндів з пам’яті LSU.

Суперскалярна структура забезпечує одночасне виконування чотирьох команд.

Через суперскалярну структуру процесора можливе звернення виконавчих пристроїв одночасно до одних і тих самих регістрів. З метою зниження помилок за спроби запису до регістру введено буферні регістри, які затримують запис доти, допоки інший пристрій не зчитає старі дані. Вони слугують задля проміжного зберігання операндів, які дублюють регістри GPR, FPR, використовувані при виконуванні поточних операцій.

Після завершення цих операцій здійснюється перезапис результатів у GPR та FPR, так званий зворотний запис.

Конвеєр виконування команд має шість основних каскадів: вибирання команди (1), дешифрування (2), розподілення команд поміж виконавчими пристроями (3), виконування команд (4), запису результатів до буфера (5) та зворотного запису результатів до регістрів SPR чи FPR. Пристрій керування одночасно вибирає з кеша і декодує чотири команди, які розподіляються по відповідних виконавчих пристроях.

Більшість команд виконуються за один такт. Ці команди зреалізовуються за допомогою SIU1 та SIU2, які здійснюють додавання, віднімання, порівняння, зсуви, логічні операції. Цілочисельне множення здійснюється за 3 або 4 такти, ділення – за 20 тактів. Ці операції виконує пристрій MIU. Більшість команд з плаваючою точкою, окрім ділення, виконується за три такти у трьох виконавчих каскадах, які складають структуру FPU. Правильна послідовність

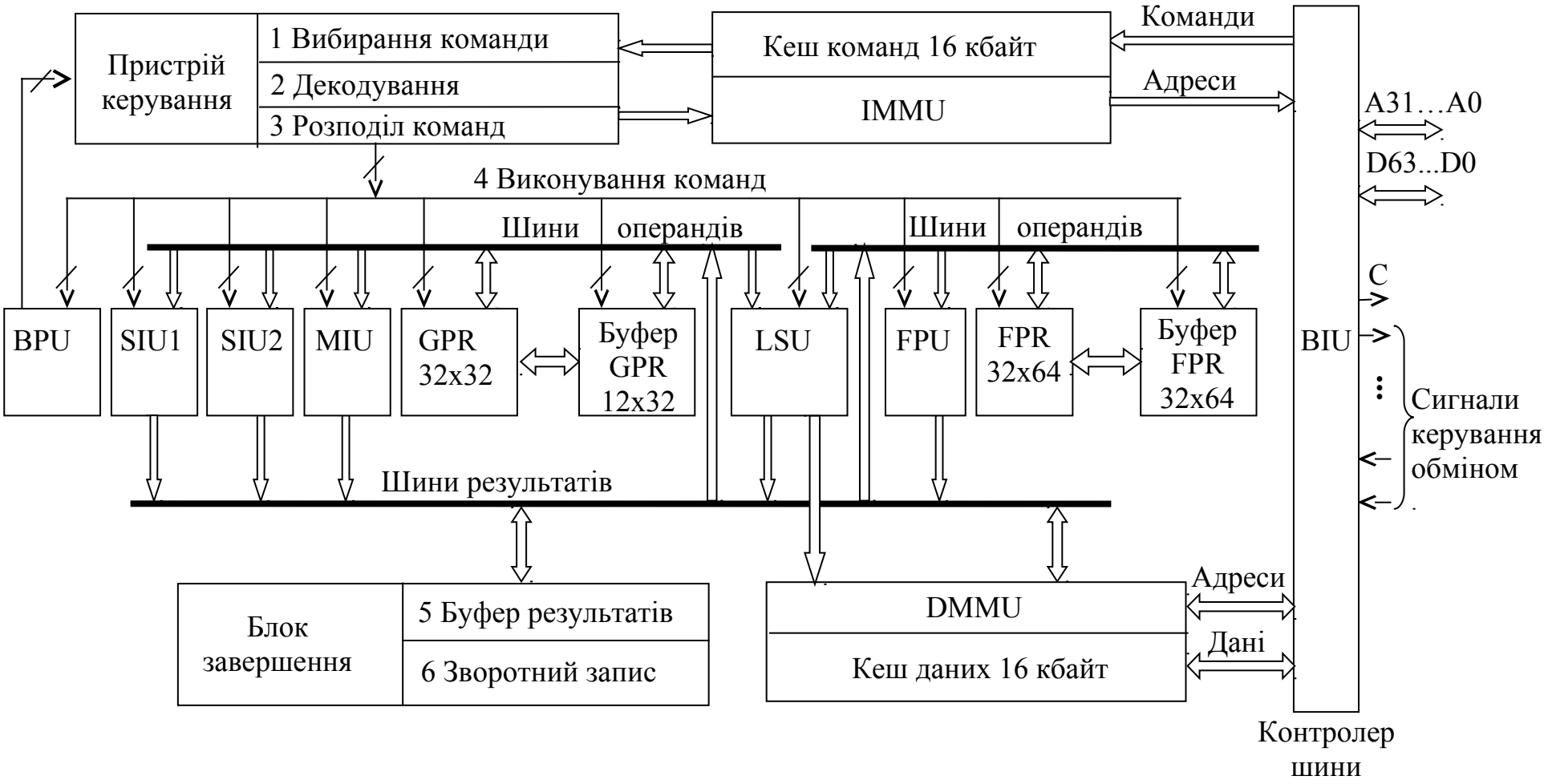


Рисунок 14.3 – Структура мікропроцесора MPC604

здобутих результатів виконування операцій збирається у спеціальному блоці завершення, який побудовано на підгрунті буферної пам'яті типу черги. Зворотний запис результатів з буфера до регістрів GPR, FPR здійснюється відповідно до порядку подавання команд.

Блок оброблення розгалужень BPU завбачає напрямок розгалуження за алгоритмами статичного та динамічного завбачання, які ґрунтуються відповідно на тексті програми або на результатах попередніх розгалужень. Алгоритми завбачання використовують для прийняття рішень кеш-адресу розгалужень BTAC та таблицю історії розгалужень BHT. Кеш BTAC зберігає 64 адреси переходів, які виконувались попередніми командами. Якщо виконувана команда переходів вміщує адресу, яка збігається з вже записаною у BTAC, то процесор вказує на розгалуження за цією адресою, тобто перехід здійснюється, а якщо ж адреси у таблиці немає, – то не здійснюється.

У разі надходження команди умовного переходу BPU аналізує таблицю BHT, яка є кеш-пам'яттю, у ній зберігаються 512 адрес попередніх умовних переходів. Для кожної адреси у таблиці є два біти, які визначають імовірність розгалуження за вказаною адресою: 00 – практично не виконується, 01 – виконується рідко, 10 – виконується часто, 11 – виконується надто часто. Значення цих бітів змінюється після кожної операції умовного переходу: якщо розгалуження відбулося за зазначеною адресою, то імовірність збільшується на 1 (максимальне значення становить 11); якщо дана адреса не використовується, то імовірність зменшується на 1 (мінімальне значення 00). Замість адрес з нульовою імовірністю вводяться нові адреси з команд умовних переходів, які надходять. Розгалуження завбачається, якщо його імовірність становить 01 або 11. Правильність завбачання перед виконуванням додатково перевіряється після визначення відповідних прапорців у регістрі CR. Завбачена команда за адресою, вказаною у команді умовного переходу, вибирається у конвеєр, тобто розгалуження здійснюється; якщо ж адресу переходу не записано в таблиці BTAC, то виконується наступна команда програми. При перевірці виконування умов переходу за прапорцями у регістрі CR, якщо умову не виконано, тобто завбачення було помилковим, процесор звільнює конвеєр від команд, які були помилково вибрані, і завантажує нові команди, розпочинаючи з правильної адреси. Помилково зроблені завбачання призводять до зниження продуктивності процесора.

Через суперскалярну структуру процесора водночас можуть виконуватись кілька команд, які змінюють значення прапорців у регістрі CR. Для розв'язування конфліктів у BTU є вісім буферних регістрів, які дублюють вміст CR і використовуються різними виконавчими пристроями. Після завершення виконування команди їхній вміст переноситься до основного регістру CR, тобто здійснюється зворотний запис.

Мікропроцесор MPC604 здійснює роздільне оброблення команд і даних за допомогою двох пристроїв керування пам'яттю – IMMU та DMMU, які можуть виконувати сегментне, сторінкове та блокове сегментування. Внутрішні кеші команд та даних мають обсяг 16 кбайт.

Інтерфейс мікропроцесора із зовнішніми пристроями забезпечується контролером шини BIU. Для кожного байта адреси та даних передаються та приймаються біти парності, за допомогою яких здійснюється контроль помилок звернення до шини. Шина адреси має 32 розряди, а шина даних – 64 розряди, що підвищує продуктивність мікропроцесора. Припустиме є підключення зовнішнього кеша 2-го рівня.

Мікропроцесор вміщує спеціалізований послідовний порт для виконування тестування за стандартом JTAG (IEEE 1149.1).

Для живлення мікропроцесора використовується напруга $V_ж = 3{,}3$ В. Зниження енергоспоживання (400 мВт) забезпечується при зупині мікропроцесора, коли припиняється виконування команд, але продовжують працювати базовий таймер, система декрементування та блок обслуговування запитів на переривання.

Контрольні запитання:

1 Які основні особливості RISC-процесорів Вам відомі?

2 Поясніть, з яких регістрів складається регістрова модель користувача процесора MPC604?

3 Які провідні фірми випускають RISC-процесори?

4 Які поля має регістр прапорців CR і в який спосіб вони використовуються?

5 Назвіть режими функціонування процесора MPC604.

6 Який обсяг регістрової пам’яті використовують сучасні RISC-процесори?

7 Яке нумерування розрядів прийнято у регістрах RISC-процесорів?

8 Які вбудовані методи захисту використовує МП PowerPC?

Контрольні запитання підвищеної складності:

1 Доведіть, що процесор MPC604 має суперскалярну структуру.

2 Як апаратно оброблюються розгалуження за алгоритмами динамічного завбачання?

14.2 RISC-процесори ColdFire

Вхідний контроль:

1 Чи є довжина команд процесора MPC604 незмінна?

2 Яку розрядність мають шини адреси та шини даних процесора MPC604?

RISC-процесори ColdFire (MCF5XXX) мають таку саму модель користувача, як і сімейство M68XXX, зреалізовують основні команди та способи адресування цього сімейства. Завдяки цьому МП MCF5XXX можуть використовувати значний обсяг програмного забезпечення, розробленого для M68XXX. Для зменшення обсягу пам’яті команд використовуються команди змінної довжини: 2, 4, 6 байт. Низка моделей вміщують на кристалі BIC

таймери, паралельні, послідовні порти, контролер переривань та інші периферійні пристрої; за це їх називають інтегрованими. МП Cold Fire слугують для побудови мікропроцесорних систем і можуть входити до складу спеціалізованих мікроконтролерів.

МП вміщують процесорне ядро CFPU з RISC-архітектурою, об'єднану кеш-пам'ять команд/даних обсягом 2 кбайти та блок зовнішнього інтерфейсу, який забезпечує зв'язок з 32-розрядною мультиплексованою системною шиною даних/адрес.

Регістрова модель процесора CFPU відрізнюється тим, що має один вказівник стека A7 і формує спільний стек для режимів користувача та супервізора (див. рис. 11.1). До регістрової моделі супервізора додатково входять регістр базової адреси таблиці векторів переривань та виключень, регістри керування кеш-пам'яттю та звернення до пам'яті.

Процесор CFPU працює у режимі користувача або супервізора, аналогічно до МП сімейства M68XXX, емулює основні команди і способи адресування цих МП. З базового набору команд CFPU не виконує деякі команди та операції з двійково-десятковими числами, що не робить систему команд недостатньо повною.

Для налагодження МПС на МП MCF5202 надаються такі можливості:

– реалізація режиму налагодження, за якого процесор працює під керуванням зовнішнього налагоджувача;

– контроль внутрішнього стану процесора при виконуванні поточної програми;

– виконування програми із зупином у контрольних точках.

На рис. 14.4 подано структуру інтегрованого мікропроцесора MCF5204, до складу якого входять:

– процесорне ядро CFPU з RISC-архітектурою ColdFire;
– кеш-пам'ять команд IC обсягом 512 байт;
– статичне ОЗП обсягом 512 байт;
– модуль системного інтерфейсу SIM-M з 8-розрядним портом A;
– блок тестування та налагодження (БТН);
– таймерний модуль;
– асинхронний послідовний інтерфейс (АПІ) типу UART.

До регістрової моделі супервізора додано 32-розрядні регістри RAMBAR та MBAR, які задають базову адресу та режим роботи внутрішнього ОЗП, регістрів різних модулів та блоків МП.

БТН зреалізовує набір режимів та варіантів тестування за стандартом JTAG (IEEE 1149.1).

Внутрішнє статичне ОЗП має обсяг 512 байт. Це ОЗП може розміщуватися, розпочинаючи з будь-якої адреси, яка може задаватися програмно. Керування ОЗП здійснюється шляхом ініціалізації регістра RAMBAR, біти якого можуть маскувати доступ до нього різних типів звернень: в режимі користувача, супервізора при записуванні та зчитуванні даних та команд.

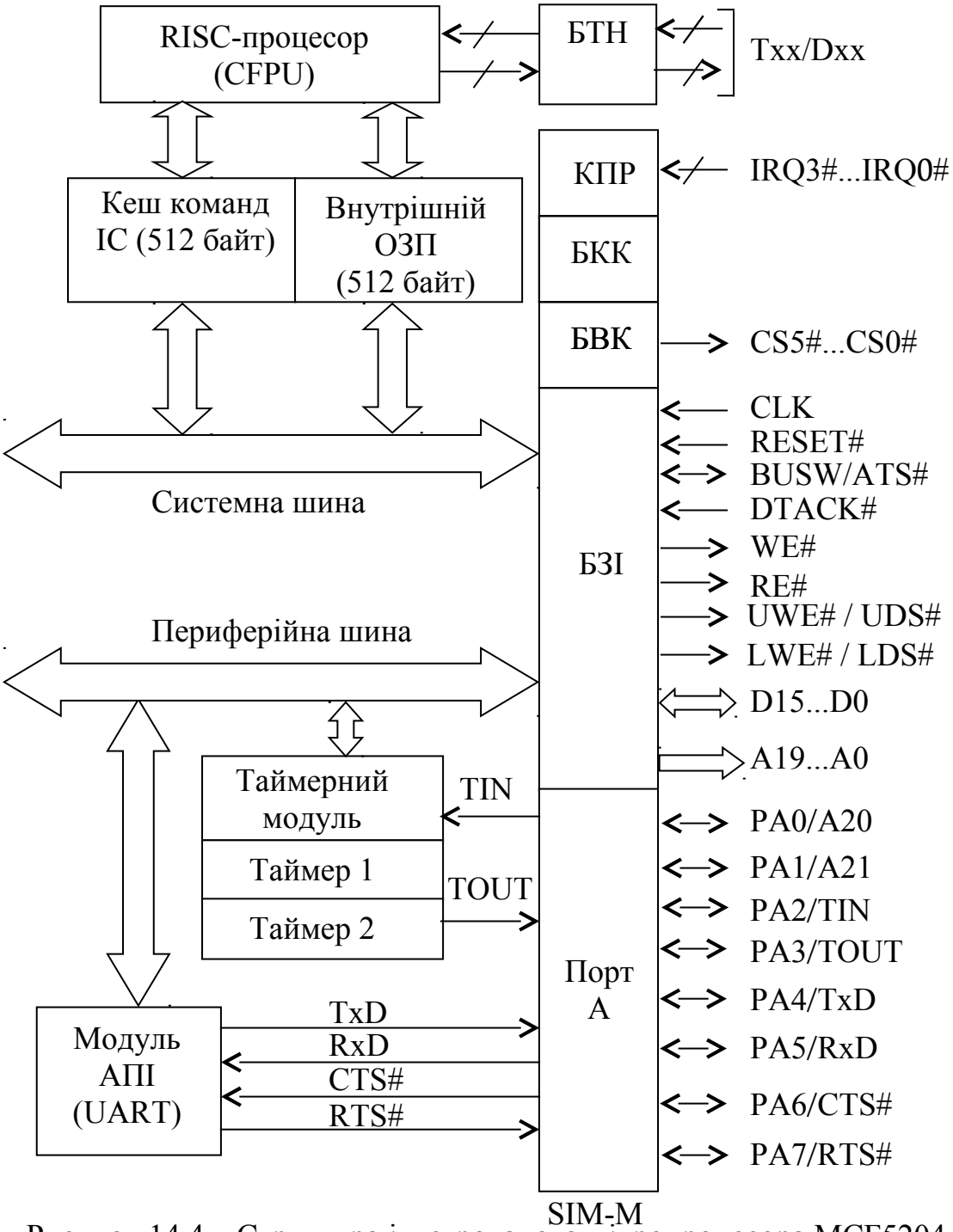


Рисунок 14.4 – Структура інтегрованого мікропроцесора MCF5204

Модуль системної інтеграції SIM-M, до якого входить блок зовнішнього інтерфейса (БЗІ), забезпечує доступ до шести різних банків даних обсягом по 4 Мбайти, для кожного з яких можна ініціалізувати власний протокол обміну.

БВК вміщує шість наборів по три регістри, які керують шістьма банками даних, призначених для зберігання даних у різних режимах роботи процесора.

Блок конфігурації та контролю (БКК) вміщує регістр MBAR, який задає базову адресу блока пам’яті, в якому зберігаються адреси службових регістрів внутрішніх модулів: SIM-M, таймерного та UART (АПІ). У регістрі MBAR є біт

достовірності та інші біти, які визначають права доступу до службових регістрів. До складу БКК входить також монітор шини та вартовий таймер. Монітор шини програмується на видавання запиту виключення “помилка звернення до шини” у разі, коли сигнал підтвердження обміну DTACK# = 0 не надходить від зовнішнього пристрою упродовж 128, 256, 512 або 1024 тактів.

Контролер переривань КПР забезпечує обробку чотирьох зовнішніх запитів переривань та чотирьох внутрішніх запитів від вартового таймера, таймерів 1 та 2, АПІ.

Порт А використовується для паралельного двоспрямованого обміну даними.

Таймерний модуль складається з 16-розрядних таймерів 1 та 2. Таймер 1 функціонує як таймер загального призначення і може використовуватись у режими вимірювання часових інтервалів або частоти сигналів, формування імпульсів заданої частоти та тривалості. Таймер 2 працює в режимі лічби внутрішніх імпульсів і може використовуватись для генерації періодичних переривань.

Асинхронний послідовний інтерфейс АПІ (UART) використовується для послідовного обміну МП із зовнішніми пристроями і в перебігу приймання контролює помилки парності, порушення кадру, переповнення, порушення зв’язку. Виявлення помилок призводить до формування запиту на переривання. Цей запит оброблюється КПР, так само як і запити після передавання чергового символу, за порожнього буфера передавача, після прийняття символу та заповнення буферної пам’яті приймача.

Основними областями використовування сімейства ColdFire є зв’язок, промислова автоматика, обчислювальна техніка, системи телекомунікацій, контрольно-вимірювальна апаратура тощо.

Контрольні запитання:

1 Якими головними рисами можна схарактеризувати RISC-процесори?

2 Які удосконалення структури порівняно з процесором MPC604 мають RISC-процесори MCF5XXX?

3 З якою метою у процесорі MCF5XXX використовуються команди змінної довжини?

4 Яку структуру має інтегрований процесор MCF5204?

5 У яких пристроях телекомунікацій використовуються RISC-процесори фірми Motorola?

Контрольні запитання підвищеної складності:

1 З якою метою в режимі супервізора можна задавати базову адресу та режим роботи внутрішнього ОЗП, регістрів різних модулів та блоків процесора MCF5XXX?

2 З якою метою у процесорі MCF5XXX використовуються 6 банків баних?

14.3 Система команд RISC-мікропроцесорів сімейства PowerPC

Вхідний контроль:

1 Які способи адресування використовують універсальні МП MC68XXX?

2 Які способи адресування використовуються при обробленні одно- та двовимірних масивів?

Задля забезпечення підвищення продуктивності мікропроцесорів PowerPC притаманні іншим мікропроцесорам способи адресування операндів застосовуються в обмеженому обсязі: регістровий, непрямий регістровий зі зміщенням, непрямий регістровий з індексуванням, відносний, абсолютний, безпосередній.

Всі команди арифметичних та логічних операцій, порівняння та зсувів виконуються лише з регістровим та безпосереднім адресуванням. Уникнення звертання до циклів звернення до шини з метою вибирання операндів підвищує продуктивність процесора. Команди мають триадресний формат, що дозволяє виключити зайві пересилання поміж регістрами. Регістри операндів називаються rA та rB, регістр розміщення результату є rD. Звернення до регістрів здійснюється за їхніми номерами GPR31…GPR0. Безпосередній операнд зі знаком SIm або без знаку UIm задається 32-розрядним словом, яке йде за кодом команди.

Команди завантажування та зберігання вмісту регістрів використовують при зверненні до пам’яті непряме регістрове адресування зі зміщенням або індексуванням. Ефективна адреса операнда EA визначається виразами:

EA = (rA!0) + d32 при адресуванні зі зміщенням;

EA = (rA!0) + (rB) при адресуванні з індексуванням;

де rA, rB – номери регістрів загального призначення GPR31...GPR1; d32 – 32-розрядне зміщення. Регістр GPRO не використовується за цих способів адресування: якщо в якості номера вказано 0, то формується адреса EA = d32 при адресуванні зі зміщенням, EA = (rB) – при адресуванні з індексуванням. Так зреалізовується пряме адресування зі зверненням за заданою адресою d32 та непряме регістрове адресування за адресою, заданою вмістом регістра rB.

У командах розгалужень використовують пряме та відносне адресування. За адресу переходу слугує вказана в команді адреса t-adr або сума (PC + t-adr).

Усі команди мають 32-розрядний формат коду операції. При використовуванні безпосереднього адресування після коду операції йде 32-розрядний операнд. У командах, які використовують непряме регістрове адресування зі зміщенням, після коду операції йде 32-розрядне зміщення, а в командах розгалужень – 32-розрядна адреса t-adr. Отже, всі команди мають 4 або 8 байт.

Команди арифметичних операцій (табл. 14.1) мають кілька модифікацій, які відрізнюються формуванням прапорців за результатами операцій. Якщо після мнемокоду операції йде символ “**.**”, то за результатами операції встановлюються відповідні значення бітів SO, EQ, GT, LT у полі CR0 регістра

CR. Якщо мнемокод команди має суфікс “O”, то за результатами її виконання встановлюються біти OV, SO у регістрі XER; біт SO дублюється у полі CR0. Команди додавання addic, addc, adde, addme, addze та віднімання subfc, subfic, subfe, subfme, subfze встановлюють прапорець перенесення CA у регістрі XER за результатом операції. Команда addi при значенні гA = 0 завантажує безпосередній операнд зі знаком SIm до регістра гD, а при гA = /0 – виконує додавання цього операнда з вмістом регістра гB. Команда безпосереднього додавання addis виконує при га = /0 додавання вмісту гA з безпосереднім операндом SIm, зсунутим на чотири розряди ліворуч, а при га = 0 – завантажує зсунутий операнд SIm до регістра гD.

Таблиця 14.1 – Команди арифметичних операцій

Синтаксис Асемблера	Операція
addi(addis) rDrA,SIm	rA|O+Slm →rD (зсув SIm)
add(add., addo, addo.) rDrA,rB	rA+rB →rD
addic(addic.) rD,rA,SIm	rA+SIm →rD, встановлення CA
addc(addc., addco, addco.) rD,rA,rB	rA+rB →rD, встановлення CA
adde(adde., addeo, addeo.) rD,rA,rB	rA+rB+CA→rD, встановлення CA
addme(addme., addmeo, addmeo.) rD,rA	rA-1+CA → rD, встановлення CA
addze(addze., addzeo, addzeo.) rD,rA	rA+CA → rD, встановлення CA
subf(subf., subfo, subfo.) rD,rA,rB	-rA+rB →rD
subfic rD,rA,Slm	-rA+Slm → rD, встановлення CA
subfc(subfc., subfco, subfco.) rD,rA,rB	-rA+rB → rD, встановлення CA
subfe(subfe., subfeo, subfeo.) rD,rA,rB	-rA+rB+CA → rD, встановлення CA
subfme(subfme., subfmeo, subfmeo) rD,rA	-rA-1+CA → rD, встановлення CA
subfze(subfze., subfzeo, subfzeo) rD,rA	-rA+CA → rD, встановлення CA
mulli rD,rA,Slm	rA*SIm → rD, знакове 16x16
mullw(mullw., mullwo, mullwo) rD,rA,rB	rA*rB → rD, знакове 16x16
mulhw(mulhw.) rD,rA,rB	rA*rB → rD, знакове 32x32
mulhwu(mulhwu.) rD,rA,rB	rA*rB → rD, беззнакове 32x32
divw(divw., divwo, divwo.) rD,rA,rB	rA/rB → rD, знакове 32:32
divwu(divwu., divwuo, divwuo.) rD,rA,rB	rA/rB → rD, беззнакове 32:32
neg(neg., nego, nego$_k$) rD,rA	-rA → rD
extsb(exstb.) rA,rS	rS(мл.байт) →rA ⎫ Розширення
extsh(exsth) rA,rS	rS(мл.п/слова) →rA ⎭ знаком

Команди множення та ділення можуть оперувати з операндами зі знаком та без знаку. При знаковому множенні 16-розрядних операндів (команди mulli, mullw) у регістрі гD розміщується 32-розрядний результат. При множенні 32-розрядних чисел зі знаком (команда mulhw) або без знаку (команда mulhwu) до регістра гD записуються старші 32 розряди результату, а до наступного регістра з номером (гB + 1) – молодші 32 розряди. При діленні 32-розрядних чисел зі знаком (команда divw) або без знаку (команда divwu) до регістра гD заноситься 32-розрядний результат.

Команда neg змінює знак операнда. Команда extsb перетворює молодший байт вмісту регістра rS шляхом розширення знаку у 32-розрядне дане, яке розміщується в rD. Команда extsh виконує аналогічне розширення знаку для 16-ти молодших розрядів (молодше полуслово) вмісту регістра rS.

Команди логічних операцій (табл. 14.2) виконуються над операндами без знаку, які є записані до регістрів rS, rB або задані безпосередньо у команді (UIm). МП зреалізовує вісім логічних операцій: “ТА”, “АБО”, “виключне АБО”, “ТА-НІ”, “АБО-НІ”, інверсію, імплікацію, еквівалентність. Команди, які мають після мнемокоду символ “**.**”, виставляють відповідні прапорці в полі CR0 регістра CR. Команди логічних операцій зі зсувом ands., oris., xoris. перед виконуванням вказаних в них операцій виконують зсув безпосереднього операнда ліворуч на 16 розрядів. До даної групи входить також команда cntizw, яка визначає кількість послідовно розташованих нулів у регістрі rS, розпочинаючи зі старшого розряду. Це число заноситься до регістра rA.

Таблиця 14.2 – Команди логічних операцій

Синтаксис Асемблера	Операція
andi.(andis.) rA,rS,UIm	rS ^ UIm →rA, (зсув UIm)
ori.(oris.) rA,rS,UIm	rS v UIm →rA, (зсув UIm)
xori.(xoris) rA,rS,UIm	rS ⊕ UIm →rA, (зсув UIm)
and(and.) rA,rS,rB	rS ^ rB →rA
or(or.) rA,rS,rB	rS v rB →rA
xor(xor.) rA,rS,rB	rS ⊕ rB →rA
and(nand.) rA,rS,rB	rS ^ rB →rA
nor(nor.) rA,rS,rB	rS v rB →rA
andc(andc.) rA,rS,rB	rS ^ rB →rA
orc(orc.) rA,rS,rB	rS v rB →rA
egv(egv.) rA,rS,rB	rS ⊕ rB →rA
cntizw(cntizw.) rA,rS	Число старших 0 в rS→rA

Команди порівняння (табл. 14.3) здійснюють віднімання операндів, які зберігаються в регістрах rA, rB або заданих безпосередньо (SIm, UIm), які можуть бути числами зі знаком або беззнаковими. За результатами віднімання встановлюються прапорці у полі CRi регістра CR. Номер поля i = 7...2 задається операндом crdD в команді Асемблера. Якщо цей операнд не є заданий або дорівнює 0, то прапорці встановлюються у полі CR0. Операнд L приймається таким, що дорівнює 0.

Команди зсувів зреалізовують багаторозрядні логічні (sew, srw), арифметичні (sraw, srawi) та циклічні (rewinm, rewnm, rewimi) зсуви вмісту регістра rS з розміщенням результату в rA. Кількість розрядів зсуву задається або безпосереднім операндом Ns, або вмістом шести молодших розрядів регістра rB. При логічних зсувах ліворуч (команда sew) або праворуч (команда srw) розряди, які звільнюються, заповнюються нулями. При арифметичному

зсуві праворуч (команди sraw, srawi) в розряди дублюється знак зсуваного операнда.

Таблиця 14.3 – Команди порівняння та зсувів

Синтаксис Асемблера	Операція
cmpi crfD,L,rA,SIm cmp crfD,L,rA,rB cmpli crfD,L,rA,UIm cmpl crfD,L,rA,rB	-rA + Sim (знакове) -rA + rB (знакове) -rA + Sim (беззнакове) -rA + rB (беззнакове)
slw(slw.) rA,rS,rB srw(srw.) rA,rS,rB srawi(srwi.) rA,rS,Ns sraw(sraw.) rA,rS,rB	[rS←s(rB)]→rS, заповнення нулями [rS→s(rB)]→rA, заповнення нулями [rS→s(Ns)]→rA, заповнення нулями [rS→s(rB)]→rA, заповнення нулями
riwinm(riwinm.) rA,rS,Ns,Mb,Me riwnm(riwnm.) rA,rS,rB,Mb,Me rlwimi(rlwimi.) rA,rS,Ns,Mb,Me	[rS←r(Ns)] ^ M→rA [rS←r(rB)] ^ M→rA ([rS←r(Ns)] ^ M) v (rA ^ M) →rA

Циклічні зсуви зреалізовуються лише ліворуч, а результат логічно множиться на маску M. Маска формується відповідно до задаваних у команді номерів початкового Mb та кінцевого Mc бітів. Біти маски з номерами від Mc до Mb мають значення 1, тому відповідні біти результату зберігаються. Решта бітів маски дорівнюють 0, і відповідні біти результату набирають такого самого значення. Здобутий внаслідок зсуву та маскування результат записується до регістра rA. При виконуванні команд rewinm, rewnm попередній вміст регістра rA не зберігається. Команда rewimi зберігає в регістрі rA значення бітів, для яких біти маски становили 0.

Команди зсувів можуть зумовлювати встановлення прапорців у полі CR0 регістра CR залежно від результату операції. Для цього після мнемокоду команди слід поставити символ “**.**”.

Команди завантаження та зберігання (табл. 14.4) завантажують до регістра GPR з номером rD байт, слово, довге слово (32 розряди) з пам’яті або записують у пам’ять байт, слово та довге слово з регістра rS. Для звернення до пам’яті використовується непряме регістрове адресування зі зміщенням або індексуванням, у цьому разі в команді зазначаються адресний rA та індексний rB регістри, при адресуванні зі зміщенням – адресний регістр rA та зміщення d32. При завантаженні байта (команди lbz та lbzx) старші розряди регістра rD заповнюються нулями. При завантаженні слова старші розряди заповнюються або нулями (команди lbhz, lnzx), або значенням старшого знакового біта b15 завантажуваного довгого слова (команди lha, lhax).

Команди завантаження-зберігання мають модифікації з суфіксом u, який надає команді додаткові функції: до регістра rA після пересилання завантажується ефективна адреса EA, використовувана в даній команді.

Таблиця 14.4 – Команди завантаження та зберігання вмісту регістра GPR

Синтаксис Асемблера	Операції
lbz(lbzu) rD,d(rA) lbzx(lbzux) rD,rA,rB lhz(lhzu) rD,d(rA) lhzx(lhzux) rD,rA,rB lha(lhau) rD,d(rA) lhax(lhaux) rD,rA,rB lwz(lwzu) rD,d(rA) lwzx(lwzux) rD,rA,rB	(EA)→rD, байт, (EA→rA) (EA)→rD, байт, (EA→rA) (EA)→rD, півслово, (EA→rA) (EA)→rD, півслово, (EA→rA) (EA)→rD, півслово, (EA→rA) (EA)→rD, півслово, (EA→rA) (EA)→rD, слово, (EA→rA) (EA)→rD, слово, (EA→rA)
stb(stbu) rS,d(rA) stbx(Stbux) rS,rA,rB sth(sthu) rS,d(rA) sthx(sthux) rS,rA,rB stw(stwu) rS,d(rB) stwx(stwux) rS,rA,rB	rS→(EA), байт, (EA→rA) rS→(EA), байт, (EA→rA) rS→(EA), півслово, (EA→rA) rS→(EA), півслово, (EA→rA) rS→(EA), слово, (EA→rA) rS→(EA), слово, (EA→rA)
lhbrx(lwbrx) rD,rA,rB sthbrx(stwbrx) rA,rB	(EA)→rD, переставлення байтів (EA)→rD, переставлення байтів
lmw rD,d(rA) stmw rS,d(rA)	(EA)→rD...GPR31, групове пересилання Rs…GPR31→(EA), групове пересилання
Lswi(lswx) rD,rA,Nb(rS) stswi(stswx) rS,rA,Nb(rB)	(EA)→rD, пересилання рядка rS→(EA), пересилання рядка

Команди lhbrz та lwbrz переставляють байти при завантаженні до регістра rD слова або довгого слова відповідно. Команди sthbrx, stwbrx виконують аналогічне переставлення байтів у разі запису з регістра rS до пам’яті слова чи подвійного слова.

Команди групового пересилання Imw, stmw завантажують або записують до пам’яті вміст групи регістрів, розпочинаючи із заданого в команді номера rD або rS до останнього регістра з номером GPR31. Завантажувані дані розташовуються в комірках пам’яті, розпочинаючи з адреси EA, яка визначається непрямим регістровим адресуванням зі зміщенням. Власне регістр rA не повинен входити до групи регістрів, які адресуються.

Команди пересилання рядків символів lswi, lswx, stswi, stswx здійснюють завантаження до регістрів або запис до пам’яті n байтів. Кількість пересиланих байтів задається зазначеним у команді операндом Nb (команди lswi, stswi) або значенням восьми старших бітів поля SS регістра XER (команди lswx, stswx). Початкова адреса пересиланих байтів є EA = (rA) для команд lswi, stswi та EA = (rA!0) + (rB) – для команд lswx, stwx. Дані у регістрах розміщуються чи вибираються, починаючи з молодшого байта регістра rD або rS. Команди спричинюють виключення, якщо rA або rB входить до числа регістрів, в які завантажується або з яких вибирається рядок символів.

Керування програмою здійснюється командами безумовних та умовних переходів, програмних переривань tw, twi, системного виклику sc та

повернення з підпрограм оброблення переривань rfi (табл. 14.5). Адреса команди переходу визначається за допомогою прямого (команди b, bl, bc, bcl) адресування.

Таблиця 14.5 – Команди керування програмою

Синтаксис Асемблера	Операція
b t-adr ba t-adr bl t-adr bla t-adr	PC + (t-adr)→PC (t-adr)→PC PC→LR, PC + (t-adr)→PC PC→LR, (t-adr)→PC
bc BO,BI,t-adr bca BO,BI,t-adr bcl BO,BI,t-adr bcla BO,BI,t-adr	PC + (t-adr)→PC, якщо BO = CRi (t-adr)→PC, якщо BC = CRi PC→LR, PC + (t-adr)→PC, якщо BO = CRi PC→LR, (t-adr)→PC, якщо BO = CRi
bclr BO,BI bclrl BO,BI bcctr BO,BI bcctrl BO,BI	LR→PC, якщо BO = CRi PC→LR, якщо BO = CRi CTR→PC, якщо BO = CRi PC→LR, CTR→PC, якщо BO = CRi
twi TO,rA,SIm tw TO,rA,rB	-rA+Slm, якщо TO, то PC, MSR→SRRO, SRR1, Ve→PC -rA+rB, якщо TO, то PC, MSR→SRRO, SRR1, Ve→PC
sc *rfi	PC, MSR→SSRO, SRR1, Ve→PC SSRO, SRR1→PC, MSR

Задане в команді число t-adr є абсолютною адресою чи зміщенням, яке визначає значення адреси переходу. За наявності у мнемокоді команди переходу суфікса l (команди bl, bla, bcl, bcla), поточне значення програмного лічильника заноситься до регістра зв’язку LR для забезпечення можливості повернення до наступної команди програми, тобто, ці команди можуть слугувати для виклику підпрограм. Команди bc, bca, bcl, bcla зреалізовують умовні переходи або виклики підпрограм. Умовою розгалуження є збіг вмісту поля CRi в регістрі умов CR з чотирма молодшими бітами заданого в команді операнда BO. Номер поля i, яке використовується для порівняння, задається операндом BI. За збігу заданого значення BO і вмісту поля CRi здійснюється перехід до команди, адреса якої визначається за допомогою абсолютного (команди bca, bcla) або відносного (команди bc, bcl) адресування. За відсутності збігу виконується наступна команда програми. Команди bcl, bcla, які зберігають у регістрі LR адресу наступної команди, слугують для реалізації умовних викликів підпрограм. Команди bclr, bclrl зреалізують умовне повернення з підпрограм, якщо задане значення BO збігається зі змістом поля CRi. У цьому разі до програмного лічильника завантажується вміст регістра зв’язку LR. Команда bcrfri зберігає в регістрі зв’язку LR поточний вміст PC задля можливості повернення з підпрограми.

При виконуванні команд bcctr, bcctri до програмного лічильника завантажується вміст регістра CTR; якщо виконується умова BO = Cri, команда bcctrd забезпечує зберігання поточного вмісту програмного лічильника в регістрі зв'язку LR.

Програміст може вказувати у програмі на підвищену імовірність розгалуження, якщо встановить у 1 п'ятий біт операнда BO. У цьому разі блок оброблення розгалужень BPU забезпечить вибирання до конвеєра наступних команд відповідно до вказаного завбачання. Такий спосіб називається статичним. За нульового значення п'ятого біта в операнді BO блок BPU здійснює динамічне завбачання за допомогою кеша адрес BTAC і таблиці історії BHT.

Команди програмних переривань tw, twi зреалізовують умовне звернення до підпрограми їхнього обслуговування. Поточний вміст програмного лічильника та регістр MSR зберігаються у регістрах SRQO та SRR1. Умовою звернення є збіг прапорців у полі CRO, які встановлюються при порівнянні вмісту регістра rA з безпосереднім операндом SIm (команда twi) або з вмістом регістра rB (команда tw), з бітами заданого в команді 4-розрядного операнда TO. Переривання зреалізовується, якщо хоча б один із встановлених у 1 прапорців у CRO збігається з відповідним бітом BO. Вміст rA, rB та операнд SIm подаються при порівнянні як число зі знаком. Команда sc здійснює виклик системного переривання з підпрограмою обслуговування. Команда rfi здійснює повернення з переривання, за якого з регістрів SRR0, SRR1 відновлюється вміст програмного лічильника та регістра керування MSR, забезпечуючи повернення до виконання перерваної програми. Ця команда виконується лише в режимі супервізора.

При виконуванні команд умовних переходів використовуються різні поля регістра CR. Операції з бітами цього регістра зреалізуються за допомогою команд, поданих у табл. 14.6. Команда mtcrd завантажує до CR вміст регістра rS, який логічно множиться на 32-бітову маску Mcr, задану в команді. Ця команда встановлює потрібні значення бітів CR або його немаскованих полів. Команда mfcr пересилає вміст CR до регістра rD. Команда mcrxr копіює чотири молодших біти з регістра виключень XER (прапорці SO, OV, CA) в полі CRi регістра CR , де номер поля і задається операндом crft. Після копіювання ознаки SO, OV, CA у регістрі XER обнулюються. Команда mcrf копіює вміст поля CRi у поле CRj, де значення i, j задаються операндами crfS та crfD. Команди логічних операцій над окремими бітами bi, bj вмісту регістра CR використовують номери i, j цих бітів, які задаються операндами crbA та crbB, а результат операції розміщується у біті bk, номер якого k визначається у біті crbD.

Команда crand виконує операцію "ТА", cror – "АБО", crxor – "виключне АБО", crnand – "ТА-НІ", cregv – еквівалентність , crande – заборона, crorc – імплікація.

Таблиця 14.6 – Команди, які змінюють вміст регістра CR

Синтаксис	Операція
mtcrf Mcr,rS	rS ^ Mcr→CR
mfcr rD	CR→rD
mcrxr crfD	XER(3-0) →CRi
mcrf crfD,crfS	Cri→CRj
crand crbD,crbA,crbB	bi ^ bj→bk
cror crbD,crbA,crbB	bi v bj→bk
crxor crbD,crbA,crbB	bi ⊕ bj→bk
cmand crbD,crbA,crbB	bi ^ bj→bk
crnor crbD,crbA,crbB	bi v bj→bk
cregv crbD,crbA,crbB	bi ⊕ bj→bk
crandc crbD,crbA,crbB	bi ^ bj→bk
crorc crbD,crbA,crbB	bi v bj→bk

Команди sync, isync, сісіо використовуються для синхронізації виконування програм кількома процесорами у багатопроцесорній системі. Команда sync зумовлює завершення виконування усіх попередніх команд, а потім дозволяє виконування наступних, тобто затримує конвеєр до моменту звільнення усіх виконавчих пристроїв. Команда isync забезпечує завершення виконування попередніх команд, після чого звільнює конвеєр від наступних команд, які до нього надійшли, і вибирає з пам’яті нову пару команд, тобто очищує конвеєр. Команда сісіо завершує виконування попередніх команд, які потребують звернення до зовнішньої пам’яті даних. Наступні звернення виконуються лише після виконання цієї команди, тобто зреалізовується затримка звернень, яка може використовуватись іншими пристроями. При виконуванні команди сісіо на зовнішніх виводах, які визначають тип циклу звернення до шини, встановлюється комбінація сигналів, яка використовується для впорядковування звернення різних пристроїв до спільної оперативної пам’яті. Команди ссіwx та ссowx забезпечують звернення мікропроцесора до спеціалізованих зовнішніх пристроїв чи розділів пам’яті при прийманні або передаванні керувальної інформації. Команди isync, сісіо та наступні виконуються мікропроцесором у режимі супервізора.

З 2002 року фірма Motorola випускає нові моделі RISC-процесорів: MPC7455, які мають суперскалярну структуру з 11-ма виконавчими механізмами. Ці моделі здатні виконувати чотири команди за один такт. Тактова частота MPC7455 сягає 1000 МГц, споживана потужність – 21,3 Вт. Напруга живлення становить 1,6 В, а схем введення-виведення – 2,5 В. Мікропроцесор має внутрішній кеш 1-го рівня (по 32 кбайт команд та даних) та 2-го рівня обсягом 256 кбайт. Мікропроцесори MPC7455, MPC740, MPC750 призначено переважно для роботи в мультипроцесорних системах, відповідно до принципу SIMD, з метою здійснювання цифрового оброблення сигналів.

Контрольні запитання:

1 Який формат команд мають RISC-процесори?

2 Як зреалізовуються алгоритми завбачання у блоці оброблення розгалужень МП MPC604?

3 Скільки команд водночас може виконувати МП MPC604?

4 Які способи адресування операндів переважно використовують RISC-процесори сімейства PowerPC?

Контрольні запитання підвищеної складності:

1 Як організовано звернення виконавчих пристроїв МП MPC604 водночас до одних і тих самих регістрів?

2 Завантажте програмно регістри GPR1, GPR2 та GPR30 відповідно даними $12345678, $ABCDH, $EF.

3 Зреалізуйте підпрограму затримки на час, який залежить від числа K, яке зберігається в регістрі GPR10.

15 АРХІТЕКТУРА ТА ПРИНЦИПИ ПОБУДОВИ ПРОЦЕСОРІВ ЦИФРОВОГО ОБРОБЛЕННЯ СИГНАЛІВ

Вхідний контроль:

1 З якою метою використовується цифрове оброблення сигналів у телекомунікаціях?

2 Який математичний апарат покладено у підгрунтя цифрового оброблення сигналів?

3 Які цифрові пристрої виконують цифрове оброблення сигналів?

4 Які особливості має гарвардська архітектура побудови МП?

5 Як подаються дані в обчислювальних системах на мікропроцесорах?

15.1 Основні напрямки цифрового оброблення сигналів (ЦОС)

До основних напрямків ЦОС належать цифрова фільтрація та спектральний аналіз. Цифрова фільтрація зреалізовується засобами пропускання цифрового сигналу через цифрові фільтри зі скінченною та нескінченною імпульсними характеристиками – СІХ та НІХ фільтри. Обидва типи фільтрів є лінійні системи з постійними параметрами, у яких вхідна x_n та вихідна y_n послідовності сигналів пов'язані відношеннями типу згортки. Відгук системи на поодинокий імпульс (імпульсна характеристика) позначимо через h_k, тоді залежність поміж відліками вихідного y_n та вхідного x_n сигналів можна описати виразом

$$y_n = \sum_{k=0}^{n} h_k x_{n-k}\ ,\ n = 0,\ 1,\ 2,\ ..., \qquad (15.1)$$

де x_n, y_n – відліки вхідного та вихідного сигналів; x_{n-k} – вхідний відлік, затриманий на k інтервалів дискретизації.

У СІХ-фільтрі відлік вихідного сигналу визначається лише значеннями вхідного сигналу, а у НІХ-фільтрі – значеннями вхідного та вихідного сигналів. Вираз, який описує НІХ-фільтр, має вигляд

$$y_n = \sum_{k=0}^{N-1} b_k x_{n-k} - \sum_{k=1}^{M-1} a_k y_{n-k}\ , \qquad (15.2)$$

де N, M є цілі константи; a_k, b_k – постійні коефіцієнти, які описують конкретний фільтр; x_n, y_n – відліки вхідного та вихідного сигналів.

СІХ-фільтр описується виразом

$$y_n = \sum_{k=0}^{N-1} b_k x_{n-k} \qquad (15.3)$$

або

$$y_n = b_0 x_n + b_1 x_{n-1} + b_2 x_{n-2} + ... + b_n x_{n-N+1}\ . \qquad (15.4)$$

Ефективна система цифрової фільтрації потребує ефективної реалізації дискретної згортки, яку можна розкласти на операції множення, накопичуючого підсумовування та операції затримки.

В області спектрального аналізу використовується пряме та обернене дискретне перетворення Фур'є (ДПФ), швидке перетворення Фур'є (ШПФ) тощо. Спектральний аналіз базується на відомих методах подання заданої функції за допомогою інших, які називаються базовими і властивості яких є відомі. Періодичну вхідну послідовність можна розкласти у ряд Фур'є:

$$x_n^p = \sum_{k=-\infty}^{\infty} X_k^p e^{jw_k n}, \qquad (15.5)$$

де X_k^p – амплітуда гармоніки, $e^{jw_k n} = \cos w_k n + j\sin w_k n$ – комплексна змінна, $w_k = 2\pi \cdot k / N$ – частота спектральної складової (гармоніки).

Ряд Фур'є можна записати також виразами

$$x_n^p = \sum_{k=0}^{N-1} X_k^p e^{j(2\pi/N)k-n} \quad \text{або} \quad x_n^p = \frac{1}{N}\sum_{k=0}^{N-1} X_k^p e^{j(2\pi/N)k-n}. \qquad (15.6)$$

Вираз (15.6) є обернене перетворення, яке описує Фур'є-образ функції.

Для обчислення коефіцієнтів ряду Фур'є використовується ДПФ:

$$X_k^p = \sum_{n=0}^{N-1} x_n^p e^{-j(2\pi/N)nk} = \sum_{n=0}^{N-1} x_n^p W^{nk}, \text{ де } W = e^{-j(2\pi/N)}. \qquad (15.7)$$

Аналіз цього виразу призводить до висновку, що основними операціями при його обчислюванні є операції комплексного множення та підсумовування.

Обчислення коефіцієнта $W_N^k = \cos[(2\pi/n)k] - j\sin[(2\pi/N)k]$ можна здійснити:

– використовуючи підпрограми або таблиці синуса та косинуса;

– у прямий табличний спосіб (вибиранням готових значень з таблиці);

– за допомогою рекурентної формули

$$W_N^k = (W_N^{k-1})W_N^L \text{ при } W_N^0 = 1;$$

– у таблично-алгоритмічний спосіб.

Трудомісткість прямого обчислення виразу (15.7) є велика і зростає зі зростанням N. Оцінити міру складності алгоритмів ШПФ, а також їхні особливості можна через аналіз обчислювальної системи.

Використовування алгоритму ШПФ з проріджуванням за частотою потребує переставлення елементів вихідної послідовності. Операція "метелик", яка визначає двоточкове дискретне перетворення ДПФ, потребує обчислення виразів

$$x = A + B \cdot W_N^k;$$

$$y = A - B \cdot W_N^k,$$

де A, B – вхідні значення; W_N^k – коефіцієнт.

Задля віднаходження вихідної послідовності у правильному порядку слід переставити вхідну послідовність в такий спосіб, що двійкові номери вихідної послідовності здобуваються додаванням 1 до старшого розряду з поширення перенесення у бік молодших розрядів (праворуч). Така операція адресування називається *біт-реверсивною*.

Задля здобуття амплітуд та фаз складових спектра (гармонік) треба виконати обчислення

$$\frac{\sqrt{X_{Rc}^2 + X_{im}^2}}{N},\ \operatorname{arctg}\frac{X_{Rc}}{X_{im}},$$

де X_{Rc}, X_{im} – дійсна та уявна частини комплексних коефіцієнтів.

Зазвичай додатково буває потрібне обчислення функцій $\log_2 x$ та 2^x.

Потрібні для цифрового оброблення сигналів операції підтримуються апаратно у процесорах цифрового оброблення сигналів. Узагальнену архітектуру процесорів DSP подано на рис. 15.1.

Контрольні запитання:

1 Які алгоритми цифрової обробки сигналів у часовій області Вам відомі?

2 Які алгоритми цифрової обробки сигналів у спектральній області Вам відомі?

3 Від яких чинників залежить трудомісткість алгоритмів перетворення Фур'є?

Контрольні запитання підвищеної складності:

1 При виконуванні ШПФ відліки біт-інверсної послідовності розташовуються у порядку слідування двійково-інверсних номерів вхідних відліків:

Вхідна послідовність		Біт-інверсна послідовність	
відлік	двійковий номер	двійковий номер	відлік
x_0	000	000	x_0
x_1	001	100	x_4
x_2	010	–	x_2
x_3	–	110	x_3
x_4	–	001	x_1
x_5	101	101	x_5
x_6	110	–	x_3
x_7	111	111	x_7

Заповніть у наведеній таблиці пропущені значення.

2 Як називається процес проріджування цифрового сигналу за часом?

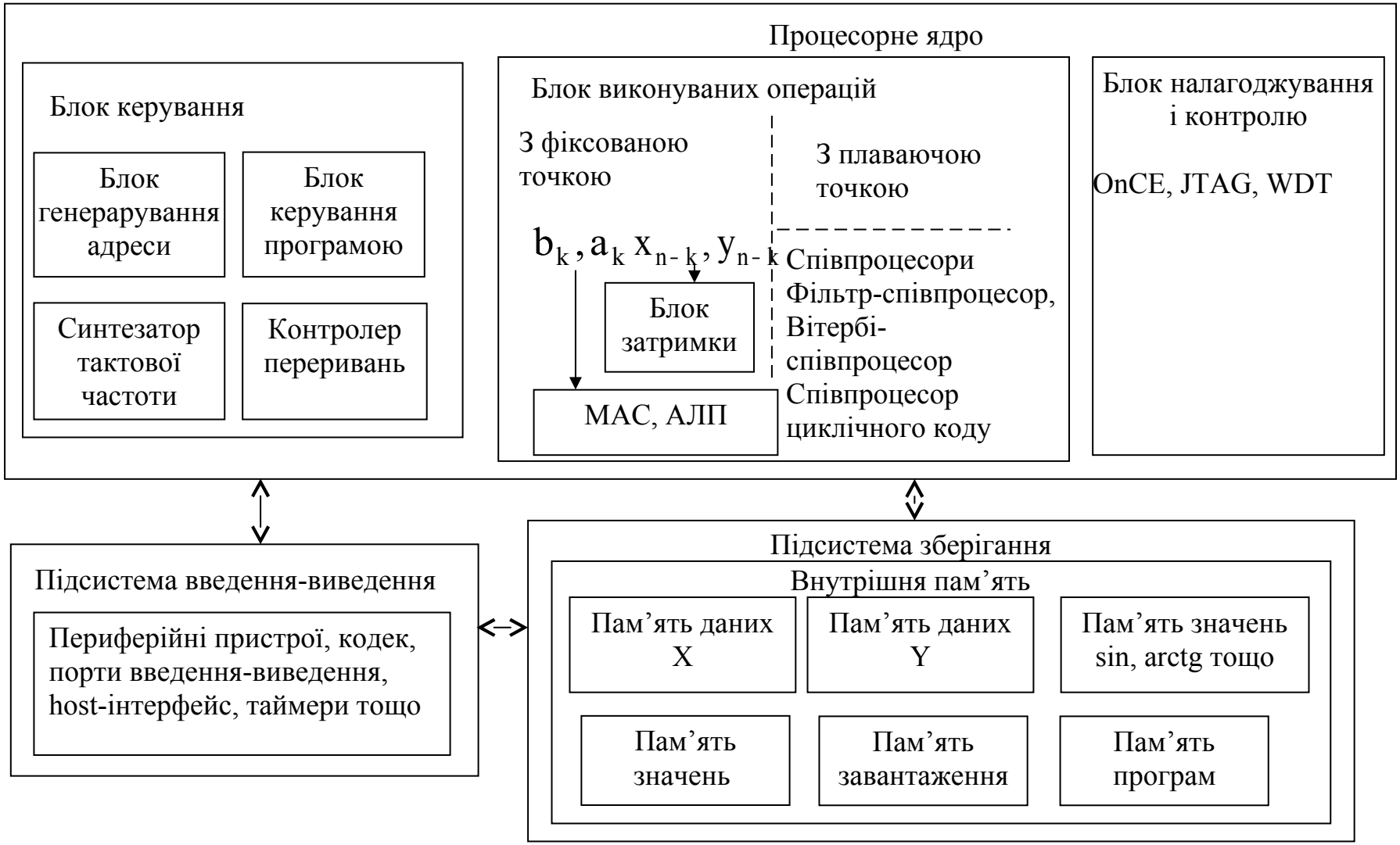


Рисунок 15.1 – Узагальнена архітектура DSP

15.2 Узагальнена архітектура процесорів сімейства DSP563XX

Вхідний контроль:

1 Які операції цифрової обробки сигналів є найпоширенішими?

2 В який спосіб сприяє гарвардська архітектура підвищенню продуктивності процесорів?

3 Поясніть, як сприяє режим ПДП підвищенню швидкості обміну даними поміж МПС та периферійними пристроями?

Процесори DSP563XX використовуються в мобільному зв'язку, цифрових системах комутації, пристроях оброблення мовного сигналу, тонального набору, цифрових телевізійних- та радіосистемах, диктофонах, локальних мережах, портативній техніці, відеотелефонах, цифрових фільтрах, аналізаторах спектру, криптографії, системах керування, навігаційному обладнанні, супутниковому зв'язку, розпізнаванні образів тощо. DSP563XX побудовані за гарвардською архітектурою і мають середню швидкодію 80 MIPS.

Периферія вміщує 8-бітний паралельний хост-інтерфейс, 32-бітний універсальний хост-інтерфейс, два розширені синхронні послідовні інтерфейси – ESSI1 та ESSI0, послідовний комунікаційний інтерфейс SCI, модуль таймера. На кристалі є вбудовані співпроцесори:

– фільтр-співпроцесор FCCP (Filter Co-Processor), який зреалізовує алгоритми фільтрації;

– Вітербі-співпроцесор VCCP (Viterbi Co-Processor), який зреалізовує алгоритм задля відновлення з максимальною вірогідністю сигналу зі спотвореннями, наприклад GSM;

– співпроцесор циклічного коду CCOP (Cyclic-code Co-Processor), який зреалізовує кодування та декодування даних, генерування коду парності та контролю.

Підсистема пам'яті – ОЗП програм, кеш інструкцій, ОЗП даних X, ОЗП даних Y – може бути сконфігурована в чотири способи і вміщує також ПЗП програм з організацією 6144x24, ПЗП даних Y – 3072x24, ПЗП, яке завантажує програми з зовнішньої пам'яті – 192x24.

Узагальнену архітектуру процесорів сімейства DSP563XX подано на рис. 15.2.

Сімейство DSP 563XX вміщує нове ядро, базоване на новітніх технологіях, які зумовили низьку вартість, низьке енергоспоживання, високу ефективність мікропроцесорів. За рахунок включення кеша команд обсягом 1Кx24 процесори допускають підключення повільної зовнішньої пам'яті при зберіганні такої самої ефективності. Продуктивність процесора лінійно залежить від частоти генератора. На кристалі BIC розміщено синтезатор частоти, порт налагоджування JTAG, потрійний таймер, host-інтерфейс (HI) для

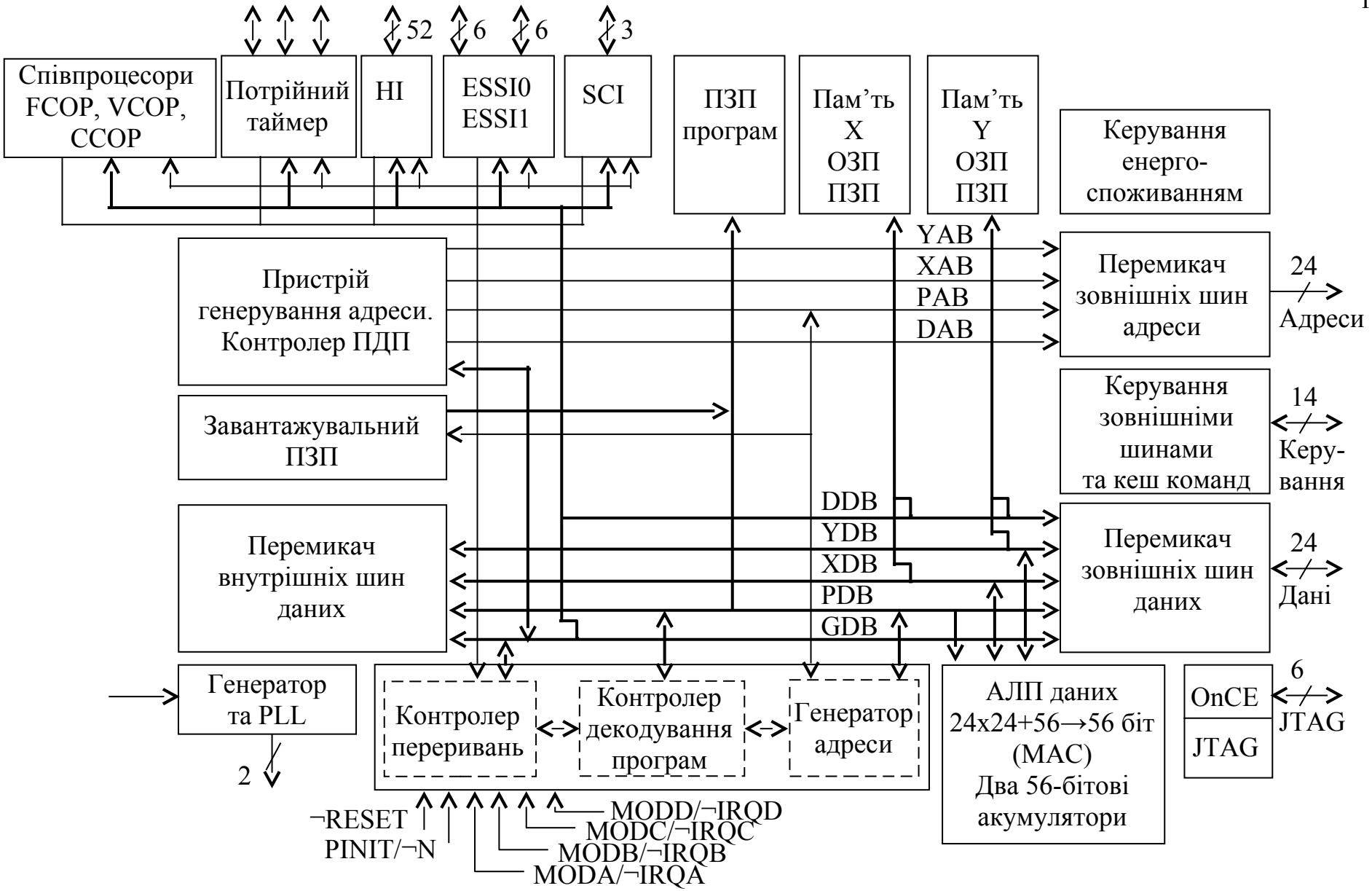


Рисунок 15.2 – Узагальнена архітектура процесорів сімейства DSP563XX

побудови багатопроцесорних систем, розширені синхронні послідовні інтерфейси ESSI0 та ESSI1, контролер кеша, послідовний комунікаційний інтерфейс SCI, пристрій керування потужністю, шестиканальний контролер ПДП (DMA Data Bus) контролер переривань для оброблення переривань у режимах MODA, MODB, MODC, MODD.

До архітектури процесорів сімейства долучено додаткову шину даних DDB (DMA Data Bus), що дозволяє за допомогою контролера ПДП передавати блоки інформації, не уповільнюючи роботу процесора.

Сімейство DSP563XX підтримує промислові стандарти щодо комп'ютерної техніки, мікропроцесорів, DSP та контролерів ПДП.

32-розрядна шина хост-інтерфейса (HI) зреалізовує три класи інтерфейсів: шини PCI, універсальну шину, порт введення-виведення загального призначення.

Сімейство DSP563XX має продуктивність 66/80/100 MIPS на частотах відповідно 66/80/100 Мгц.

Контрольні запитання:

1 Які співпроцесори вбудовуються у кристали сигнальних процесорів сімейства DSP563XX фірми Motorola**?**

2 В який спосіб пов'язані в архітектурі сигнальних процесорів наявність пам'яті програм, пам'яті X та Y ОЗП, окремих шин до цих видів пам'яті з можливістю виконувати в одній команді операції та кілька пересилань?

3 У яких областях телекомунікацій використовуються DSP?

Контрольні запитання підвищеної складності:

1 Як здійснюється тестування сигнальних процесорів відповідно до стандарту IEEEII49.1 через JTAG-порт?

2 З якою метою на кристалі DSP563XX розташовується синтезатор частоти?

15.3 Організація циклічного буфера в DSP

Вхідний контроль:

1 Наведіть приклади використовування режиму обміну ПДП поміж МПС та зовнішніми пристроями.

2 Що таке модульна арифметика?

Завдання цифрової обробки сигналів потребують реалізації потокової обробки великих обсягів даних у реальному режимі часу. Це є можливе за високої швидкодії процесора та наявності апаратних засобів інтенсивного обміну із зовнішніми пристроями.

Циклічний буфер слугує за "вікно" заданого розміру, через яке "протягується" задана кількість відліків вхідного сигналу з метою їхнього оброблення в реальному часі.

Циклічний буфер у складі DSP – це обмежена за обсягом пам'ять X або Y з M комірок, які адресуються за типом непрямого регістрового адресування. Ефективна адреса комірки обчислюється як алгебраїчна сума вмісту регістра адреси Rn, регістра зміщення Nn та регістра модифікації Mn за правилами модульної арифметики. Модульна арифметика забезпечує звернення лише до тих комірок пам'яті, які належать конкретному буферові.

Нижня межа буфера визначається вмістом регістра Rn, а верхня дорівнює (Rn + M – 1), де значення (M – 1) зберігається в регістрі модифікації Mn. Значення нижньої межі, яка називається ще базою адреси (Bs), має мати нулі у k молодших розрядах, k вибирається з умови $2^k \geq M$; база адреси є кратна до 2^k.

Для організації буфера обсягом 5 комірок пам'яті (M = 5) кількість k молодших нульових бітів у базі Bs може становити 3. База Bs може бути 101000 або 111000, тобто бути кратною до 2^k. У загальному випадку база вибирається такою, що дорівнює

$$\underbrace{XXXX...XX}_{(16-k)\text{ біт}}\underbrace{00\ldots0}_{k\text{ біт}}, \text{ де } X = 0 \text{ або } 1.$$

Верхня межа буфера дорівнює Bs + M – 1, а гранична межа – Bs + 2^k – 1.
На рис. 15.3 подано організацію циклічного буфера.

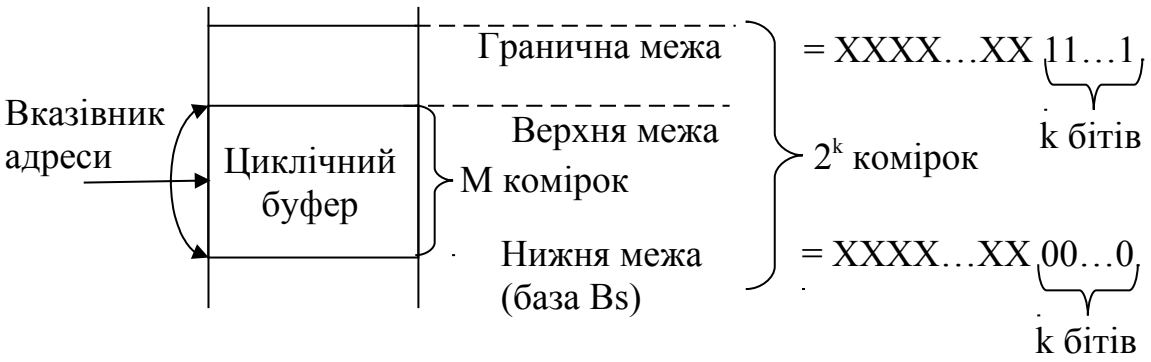


Рисунок 15.3 – Організація циклічного буфера

Початкова адреса, тобто вміст регістра Rn може встановлюватись довільно у зазначених межах.

При використовуванні регістра Nn адреса обчислюється за формулою ((Rn ± Nn)mod M. На рис. 15.4 подано виконування команди

MOVE X0, X: (RY) + N0

при роботі з циклічним буфером. Згідно з командою, вміст вхідного регістра АЛП даних X0 пересилається до комірки X-пам'яті даних за адресою, зазначеною в R0, після чого здійснюється інкрементування адреси на вміст регістра N0 з використовуванням арифметики за mod M. Всі адреси зазначено в десятковій системі числення; циклічний буфер має обсяг 90 комірок з базою Bs = 128, верхньою межею 217 та граничною межею 255.

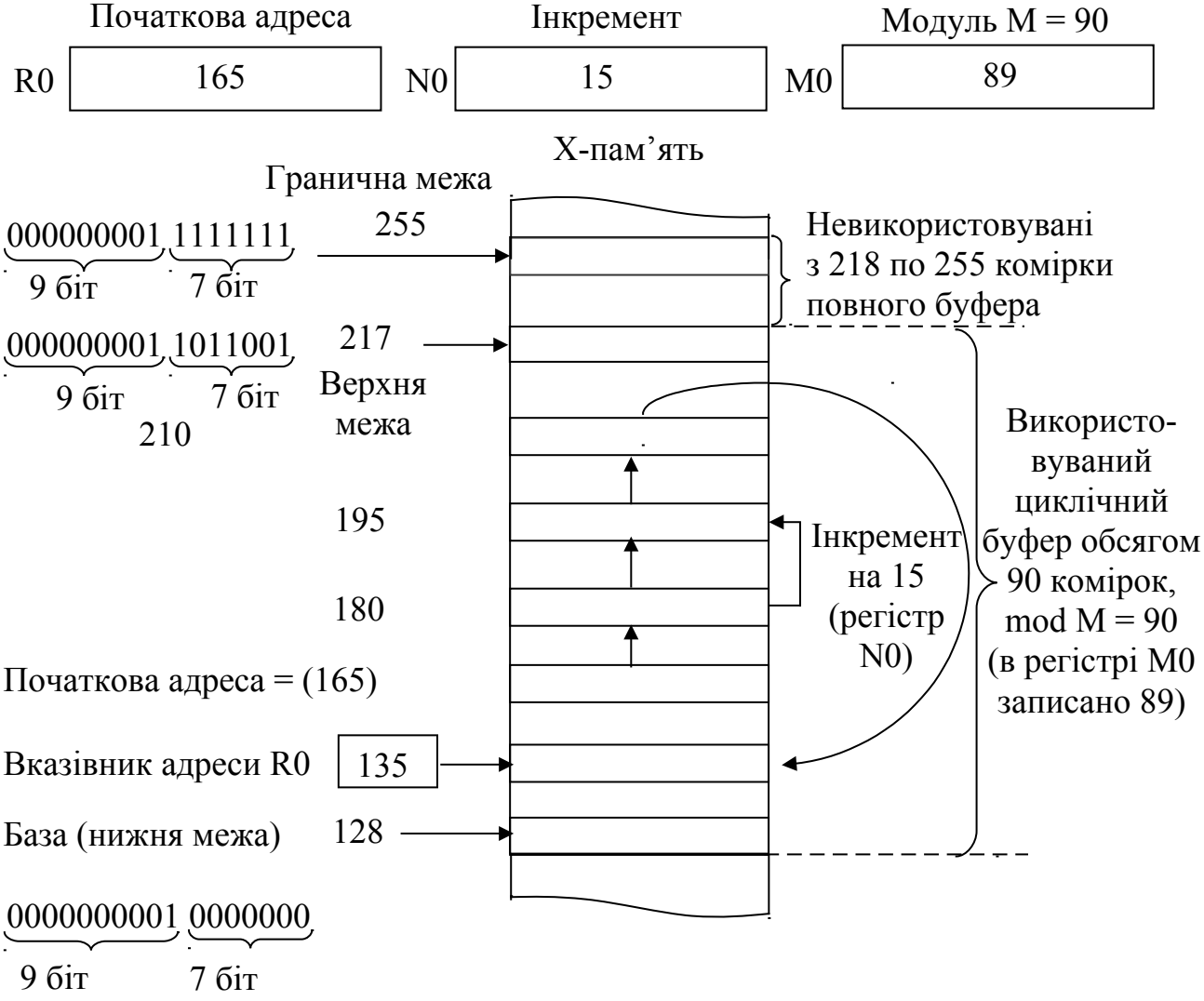


Рисунок 15.4 – Використовування арифметики за модулем M

Кількість молодших нульових бітів у базі вибрано з умови $2^k \geq 90$, тоді k = 7. Bs = 128 (крайнє 2^k). Верхня межа становить 128 + 90 = 217. Гранична межа становить $128 + 2^7 - 1 = 255$; комірки з 218 по 255 повного буфера не використовуються і можуть використовуватись для інших цілей. Після першого виконання команди початкова адреса інкрементується на 15 і вміст R0 дорівнює 180. При подальшому інкрементуванні адреси вміст R0 послідовно стає 195 та 210. При черговому інкрементуванні адреса циклічного буфера могла б стати 225 і вийти за верхню межу буфера. Модульна арифметика примушує вміст R0 залишатися усередині буфера, тобто (R0) = 225 – 90 = 135.

Контрольні запитання:

1 Як вибирається початкова адреса в межах циклічного буфера?
2 У якій пам'яті, X або Y, може бути зорганізовано циклічний буфер?
3 В який спосіб визначаються верхня та нижня межі циклічного буфера?

Контрольні запитання підвищеної складності:

1 З якою метою у DSP використовується циклічний буфер?

2 Організуйте у пам'яті циклічний буфер обсягом 7 комірок.

3 Вкажіть значення нижньої, верхньої та граничної межі.

4 Покажіть, як спрацьовує модульна арифметика при обчисленні адреси пам'яті у межах циклічного буфера?

15.4 Програмна реалізація цифрового фільтра СІХ

Вхідний контроль:

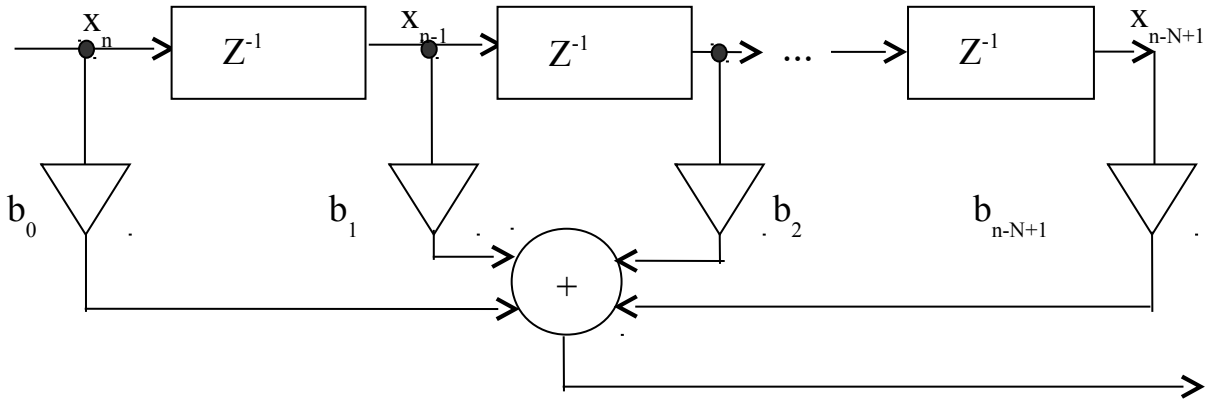
1 На якій операції цифрової обробки сигналів базується алгоритм реалізації цифрового фільтра СІХ?

2 Фільтри яких частот можна побудувати на базі цифрових СІХ-фільтрів?

Алгоритм реалізації фільтра має вигляд:

$$y_n = \sum_{k=0}^{N-1} b_k x_{n-k} \text{ , де } n = 0, 1, 2\ldots.$$

Цифровий фільтр має структуру:



Фрагмент програми, яка зреалізовує цифровий фільтр 4-го порядку:

```
1 ADDR_A EQU 5000          ; Задання адреси циклічного
                           ; буфера (ЦБ)
2 ADDR_B EQU 6000          ; Задання адреси bk
3 ADDR_C EQU 4000          ; Задання адреси xn
4 ADDR_D EQU 3000          ; Задання адреси yn
5 ADDR_E EQU 2000          ; Задання адрес N та M
6 ORG Y:ADDR_B             ; В Y-пам'яті, розпочинаючи
7 DC 0.9,0.8               ; з адреси 6000, записано
8 DC 0.7,0.6               ; вектор коефіцієнтів b
9 ORG X:ADDR_C             ; В X-пам'яті, розпочинаючи
```

```
                                  ; з адреси 4000, записано вектор
                                  ; відліків xn
10 ORG X:ADDR_E                   ; В X-пам'яті за адресою 2000
11 DC 4                           ; занесено розмір буфера N
12 ORG Y:ADDR_E                   ; В Y-пам'яті за адресою 2000
13 DC 4                           ; занесено кількість відліків M
14 ORG P:$100                     ; Об'єктний код завантаження до
                                  ; пам'яті команд,
                                  ; розпочинаючи з адреси $100
15 MOVE #ADDR_E,R6                ; R6 – вказівник на N та M
16 MOVE #ADDR_D,R5                ; R5 – вказівник на Yn
17 MOVE #ADDR_B,R4                ; R4 – вказівник на bk
18 MOVE #ADDR_C,R1                ; R1 – вказівник на вектор xn
19 MOVE #ADDR_A,R0                ; R0 – вказівник на ЦБ
20 MOVE X:(RG),A                  ; Запис N до акумулятора
21 DEC A                          ; Організація ЦБ
22 MOVE A,M0                      ; довжиною N (M0 = N – 1)
23 CLR A                          ; Обнулення циклічного
24 REP X:(R6)                     ; буфера
25 MOVE A,X(R0)+
26 MOVE M0,M4
27 MOVE X:(R1)+,X0                ; Завантаження x0→X0
28 DO Y:(R6),loop
29 CLR A X0,X(R0)- Y:(R4)+,Y0     ; xn→циклічний буфер
                                  ; bk→Y0
30 REP M0
31 MAC X0,Y0,A X: (R0)-, X0 Y: (R4)+, Y0   ; Команда MAC-операції; сума
                                  ; bk·xn-k від k = 0 до k = N – 2
32 D0 Y: (R6), loop               ; xn→ЦБ
   CLR A X0, X: (R0)- Y: (R4)+, Y0   ; bk→Y0
   REP M0
   MAC X0, Y0, A X:(R0)-,X0 Y:(R4)+,Y0   ; Сума bk·xn-k від k = 0 до k = N – 2
   MACR X0,Y0,A (R0)+             ; A – yn, R0 вказує на xn-N+1
   MOVE X:(R1)+,X0 A,Y:(R5)+      ; xn→X0, запис результату
                                  ; yn до Y-пам'яті за адресою
                                  ; від 3000
   loop                           ;
```

Команда MAC ±S1, S2, D виконує знакове множення з акумулюванням: множення двох знакових 24-бітних операндів джерела S1 та S2 та додавання або віднімання результату з/від акумулятора приймача D; результат зберігається в D.

Команда MACR ± S1, S2, D виконує знакове множення з акумулюванням та округленням: множення двох знакових 24-бітних операндів джерела S1 та S2, додавання або віднімання результату з/від акумулятора приймача D та

подальше округлення результату з використовуванням конвергентного округлення; результат зберігається в D.

Контрольні запитання:

1 Які операції виконує команда MAC?

2 З якою метою виконується обмеження результату за програмної реалізації CIX-фільтра?

3 В який спосіб у процесорі DSP563XX апаратно підтримується команда MAC X0, Y0, A X: $(R0)^-$, X0 Y: $(R4)^+$, Y0 ?

Контрольні запитання підвищеної складності:

1 Наведіть структуру цифрового фільтра 5-го порядку.

2 У яких командах наведеного вище фрагмента програми треба зробити зміни, щоби зреалізувати цифровий фільтр 5-го порядку, і які саме?

16 МПС НА МІКРОКОНТРОЛЕРАХ, МІКРОПРОЦЕСОРАХ ТА DSP

Вхідний контроль:

1 Яку архітектуру можуть мати багатопроцесорні системи?

2 Які функції має виконувати інтерфейс у багатопроцесорній системі?

3 Яку розрядність шини адреси можуть мати мікроконтролери фірми Motorola MC68HC05J та MC68HC11?

4 З якою метою реалізується клямкування адреси у МПС, якщо шина адреси/даних мікропроцесора є мультиплексована?

5 З якою метою в МПС використовується пріоритетний шифратор?

6 З якою метою в МПС використовується декодер вектора переривань?

DSP працюють переважно у складі багатопроцесорних систем, принаймні двопроцесорних, де другим процесором, так званим Host-процесором, може бути мікропроцесор, мікроконтролер, інший DSP або апаратний ПДП. Зв’язок поміж DSP та Host-процесором зреалізовується через Host-інтерфейс (HI). На прикладі Host-інтерфейсу DSP сімейства DSP56000 фірми Motorola розглянемо його структуру. HI – 8-бітний повнодуплексний, з подвійною буферізацією паралельний порт, який долучується безпосередньо до шини даних Host-процесора. HI – асинхронний інтерфейс, який вміщує два банки регістрів: один, доступний Host-процесору, і другий банк, доступний процесору DSP. HI забезпечує швидкість передавання пакетів 8 Мбайт/с, максимальна швидкість передавання даних при обробленні переривань становить 1,71 мільйон 24-розрядних слів/с.

Структуру HI подано на рис. 16.1. Процесор DSP розглядає HI як периферійний пристрій, який займає три 24-бітні слова у просторі пам’яті даних. Регістри інтерфейсу є доступні за допомогою стандартних команд процесора і способів адресування. При програмному та апаратному скиданні HI використовується задля стандартного введення-виведення. Двоспрямована шина даних H7...H0 використовується для передавання даних поміж Host-процесором та DSP. Виходи адреси HA2...HA0 забезпечують адресне вибирання регістрів HI, вони є стабільні, якщо вхід ¬HEN дозволу переривань має низький активний рівень. Вхід HR/¬W вибирає напрямок передавання даних при доступі до Host-процесора. Якщо HR/¬W = 1 та ¬HEN є активний, дані передаються з DSP до Host-процесора; якщо HR/¬W = 0, то при активному вході ¬HEN, дані передаються з Host-процесора до DSP. Вхід HR/¬W є стабільний, якщо ¬HEN активний. Вхід ¬HEN – дозвіл HOST, дозволяє передавати дані шиною даних HOST. Якщо ¬HEN не є активний, лінії шини даних перебувають у третьому стані. Вихід ¬HREQ забезпечує надходження від DSP до Host-процесора, контролера ПДП або іншого зовнішнього контролера

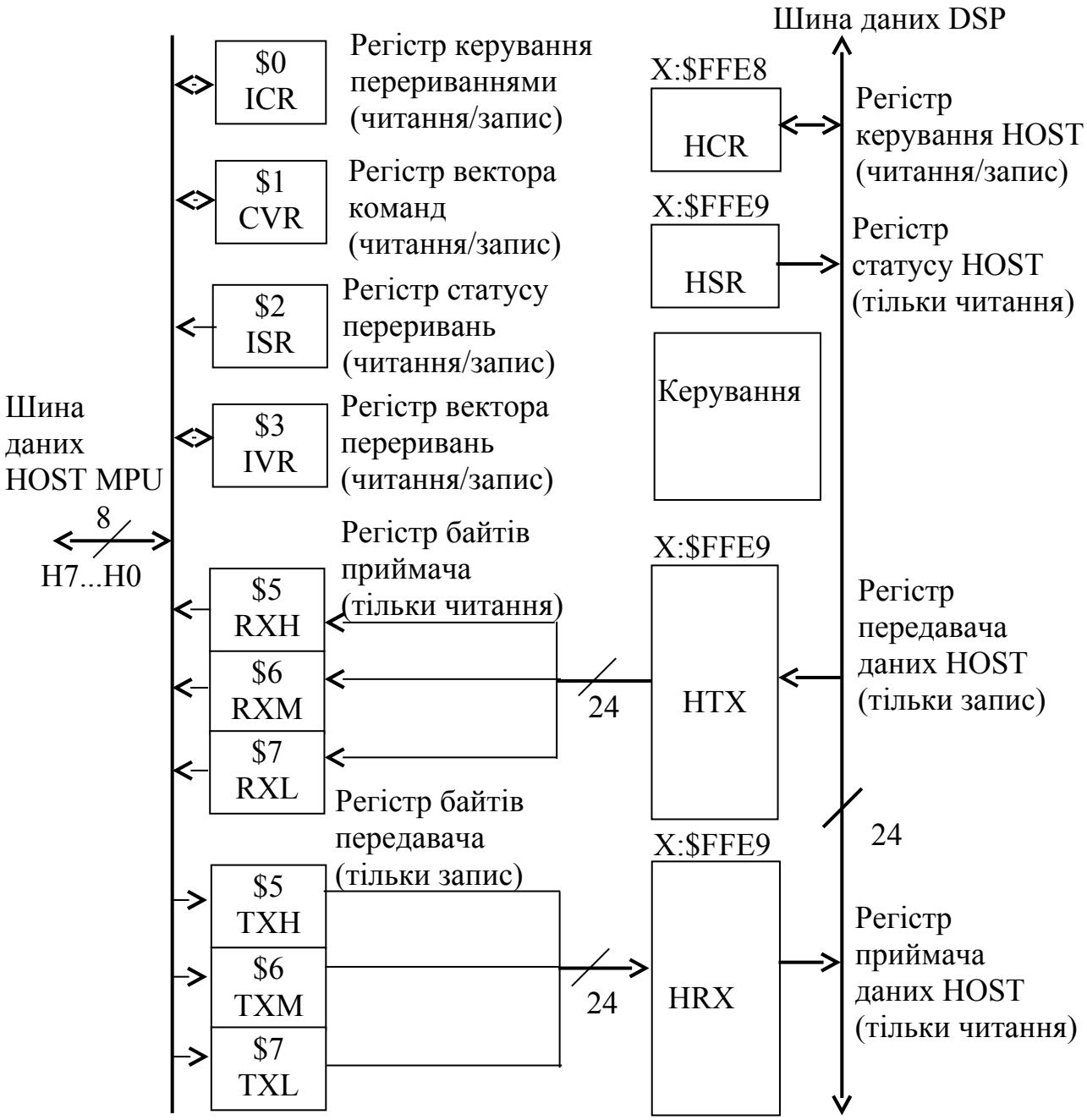


Рисунок 16.1 – Структура HI

сигналу запиту на переривання ¬IRQ. Вихід ¬HREQ може сполучуватись з контактом запиту на переривання Host-процесора, запитом передавання контролера або входом керування зовнішнього пристрою. Вихід ¬HACK – забезпечення сигналу відповіді при операціях ПДП та сигналу відповіді при перериванні для сумісності з процесорами сімейства MC68XXX. У першому випадку сигнал ¬HACK використовується для стробування даних при операціях ПДП, а у другому – для дозволу видавання на шину даних вектора переривань, якщо сигнал HREQ є активний.

На рис. 16.2 наведено двопроцесорну систему на базі DSP56000 та мікроконтролера MC68HC11 фірми Motorola, який має мультиплексовані шини адреси та даних і тому потребує клямкування адреси. Усі невикористовувані входи може бути підключено до живлення через резистор, як наприклад, ¬HACK, для запобігання виникнення помилкових сигналів.

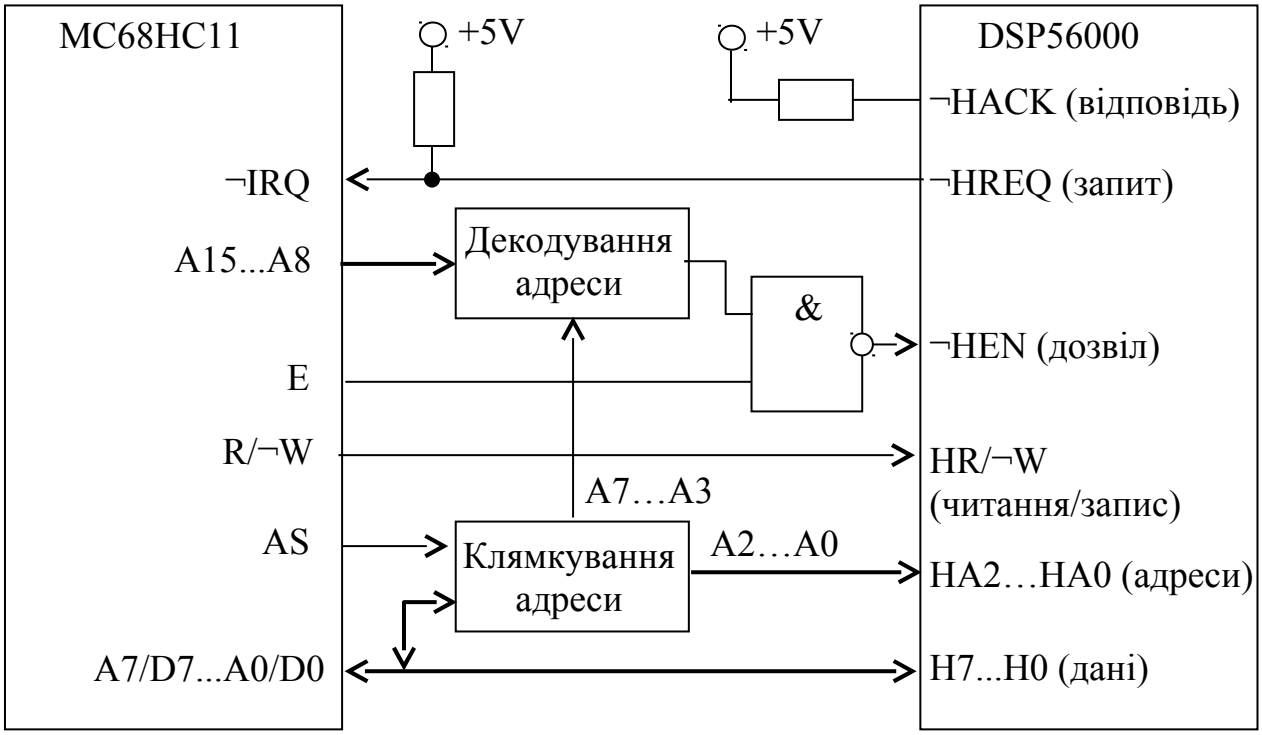


Рисунок 16.2 – Система на базі DSP та MC68HC11

На рис. 16.3 наведено двопроцесорну систему на базі DSP56000 та процесора MC68000 фірми Motorola. Процесор MC68000 може використовувати команду MOVEP зі словом, довгим словом задля передавання до DSP або читання з нього послідовності даних. При використовуванні в якості Host-процесора MC68020 або MC68030 у будь-якій команді може використовуватись динамічно змінюваний розмір шини.

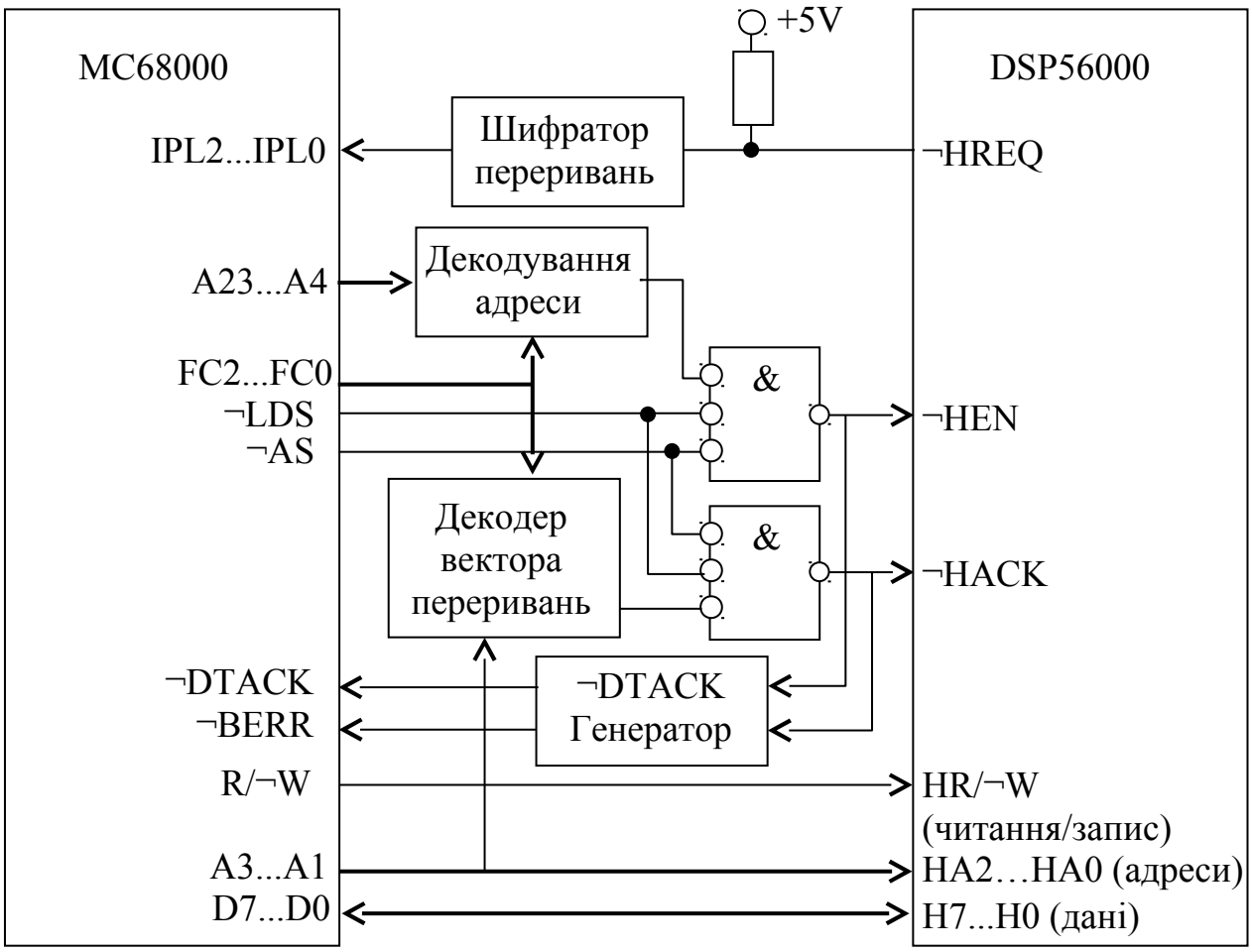


Рисунок 16.3 – Система на базі DSP та MC68000

На рис. 16.4 наведено мікропроцесорну систему з чотирьох DSP, які сполучуються за допомогою одного Host-інтерфейсу. Така МПС може виконувати до 41-го мільйона команд у секунду і може масштабуватись для підвищування продуктивності. SSI на рис. 16.4 – послідовний інтерфейс.

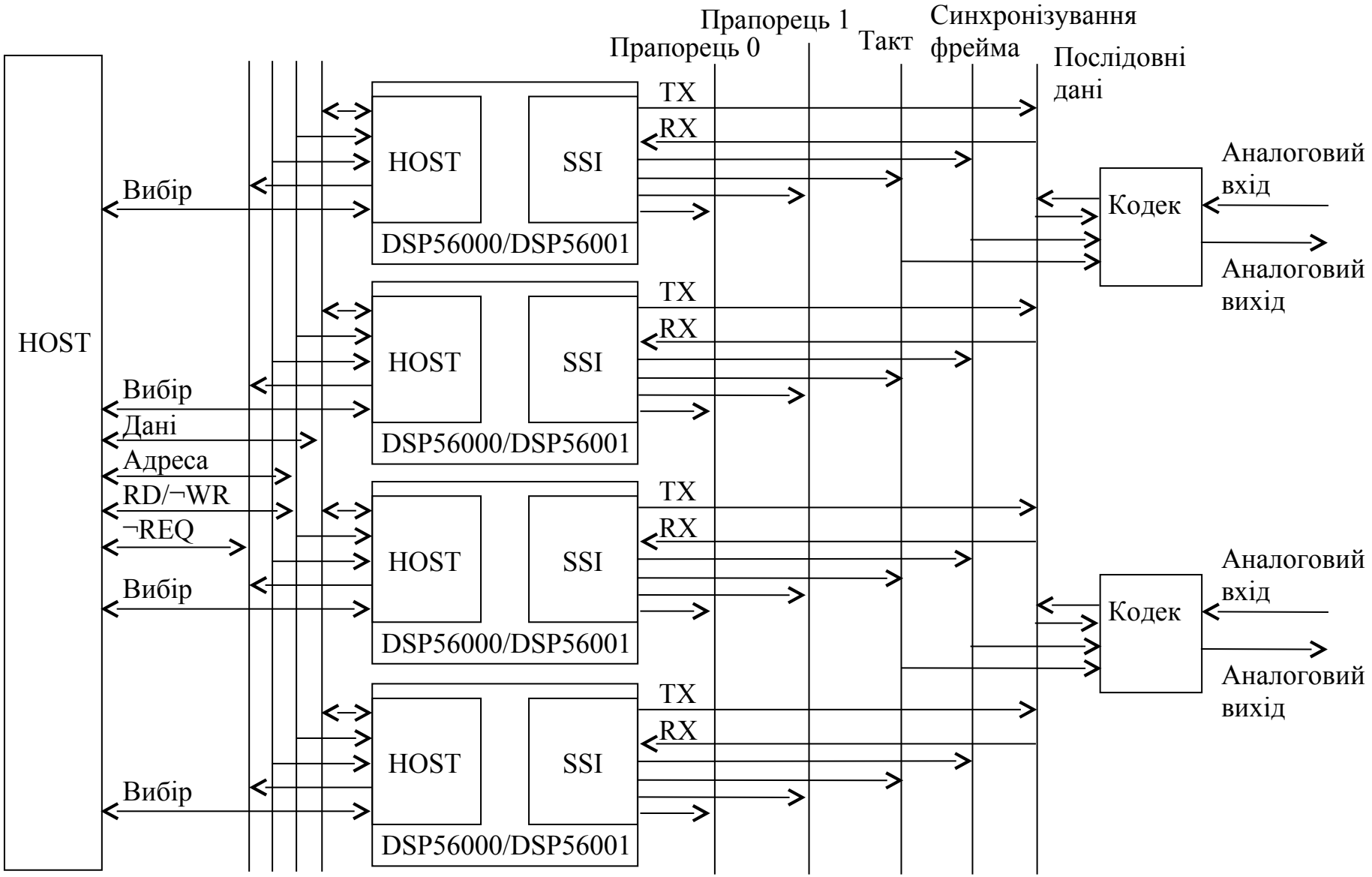


Рисунок 16.4 – Система на базі кількох DSP

Контрольні запитання:

1 Яку роль у багатопроцесорних системах відіграє мікроконтролер і яку DSP?

2 Яку роль у багатопроцесорних системах відіграє мікропроцесор і яку DSP?

3 За допомогою якого вузла DSP зреалізується долучання його до Host-процесора?

4 Які пристрої можуть використовуватись в якості Host-процесора?

Контрольні запитання підвищеної складності:

1 В який спосіб можуть обмінюватися інформацією процесори у багатопроцесорних системах?

2 В який спосіб при послідовному чи паралельному передаванні даних зреалізовується обмін поміж DSP та Host-процесором?

3 Чи є передавання даних через Host-інтерфейс: синхронне? асинхронне?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ ДО 4-го МОДУЛЯ

1 Шагурин И. И. Микропроцессоры и микроконтроллеры фирмы Motorola: Справ. пособие. – М.: Радио и связь, 1998. – 560 с.: ил.
2 Шагурин И. И. Современные микроконтроллеры и микропроцессоры Motorola: Справ. пособие. – М.: Горячая линия – Телеком, 2004. – 952 с.: ил.
3 MC68HC05 APPLICATIONS GUIDE – ©MOTOROLA INC., 1989.
4 MC68HC705J1A TECHNICAL DATE – ©MOTOROLA, 1995.
5 James M. Sibigtroth MC68HC705J1A Understanding Small Microcontrollers – ©MOTOROLA, 1995.
6 Куприянов М. С., Матюшин Б. Д. Цифровая обработка сигналов: процессоры, алгоритмы, средства проектирования. – 2-е изд., перераб. и доп. – С.Пб.: Политехника, 1999. – 592 с.: ил.
7 Солонина А. И., Уласович Д. А., Яковлев Л. А. Цифровые процессоры обработки сигналов фирмы Motorola. – С.Пб.: БХВ-Петербург, 2000. – 512 с.: ил.
8 Белами Дж. Цифровая телефония: Пер. с англ. / Под ред. А. Н. Берлина, Ю. Н. Чернышова. – М.: Око-Трендз, 2004. – 640 с.: ил.