СИСТЕМНІ ПЕРИФЕРІЙНІ ПРИСТРОЇ

3. 1. Загальні відомості

До системної периферії відносяться модулі, що визначають швидкодію і функціональні характеристики окремих пристроїв. Вбудовані FLASH– пам'ять і ОЗП, контролер зовнішньої шини, підсистема тактування з її схемою ФАПЧ (фазове автопідлаштування частоти), яка використовується для множення частоти сигналу зовнішнього генератора і забезпечує формування тактового сигналу процесора частотою до 72 МГц, засоби керування електроспоживанням, а також підсистемою переривань. Системні периферійні пристрої розглянемо на прикладі мікроконтролерів сімейства LPC2300.

3.2. Внутрішні шини

З точки зору програміста, пам'ять кожного з пристроїв сімейства LPC2300 є безперервним 32–бітовим адресним простором. Однак насправді всередині пристрою є кілька шин. Ці шини були розроблені компанією ARM, і в будь – якому мікроконтролері з ядром ARM міститься, щонайменше, дві шини (рисунок 3.1).

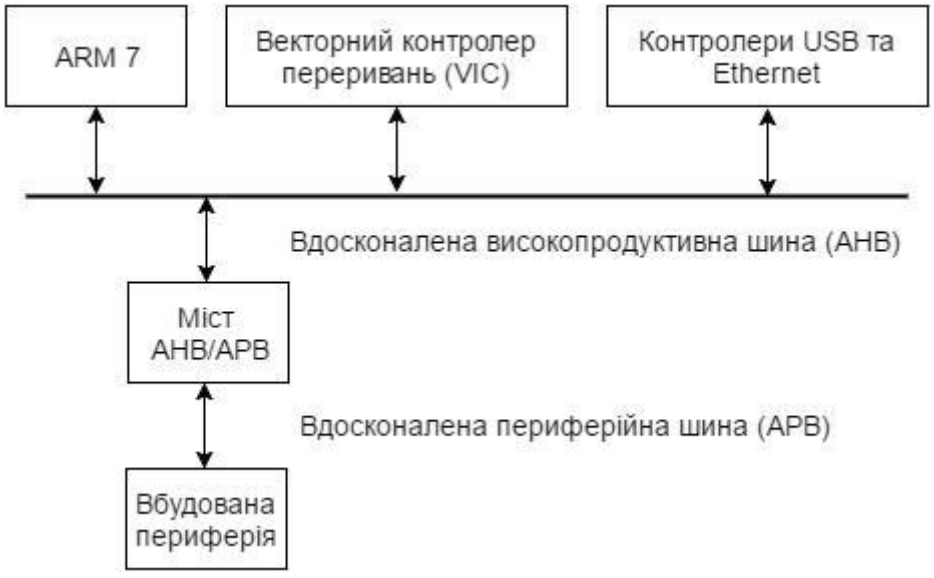

Рисунок 3.1 – Спрощена базова структура ARM7–сумісного мікроконтролера

Ядро ARM7 підключено до вдосконаленої високопродуктивної шини (Advanced High performance Bus – AHB). Як видно з назви шини, вона забезпечує найбільшу швидкість обміну між периферійними пристроями та ядром ARM7,

тому до неї підключаються найбільш швидкодіючі периферійні пристрої, такі як векторний контролер переривань, контролери USB і Ethernet.

Решта периферійних модулів підключені до другої стандартної шини – вдосконаленої периферійної шини (Advanced Peripheral Bus – APB). Сама шина підключається до шини AHB за допомогою моста, в складі якого є подільник частоти тактового сигналу. Завдяки цьому подільнику шина APB може працювати повільніше, ніж ядро ARM7 і шина AHB. Відповідно, для зменшення потужності живлення ми можемо запускати призначену для користувача периферію на частоті, яка буде нижче за частоту основного процесора.

У складі мікроконтролерів сімейства LPC2300 також є ряд швидкодіючих периферійних пристроїв, кожний з власним модулем прямого доступу до пам'яті (Direct Memory Access – DMA). Функціонування таких пристроїв вимагає частих звернень до шини, і застосування в даному випадку базової структури привело б до значного зниження загальної продуктивності мікроконтролера через постійні конфлікти між ядром ARM7 і модулями DMA. З цієї причини мікроконтролери LPC2300 мають більш складну структуру і включають в себе дві шини AHB, одну шину APB і додаткову локальну шину для зв'язку з ядром ARM7 (рисунок 3.2).

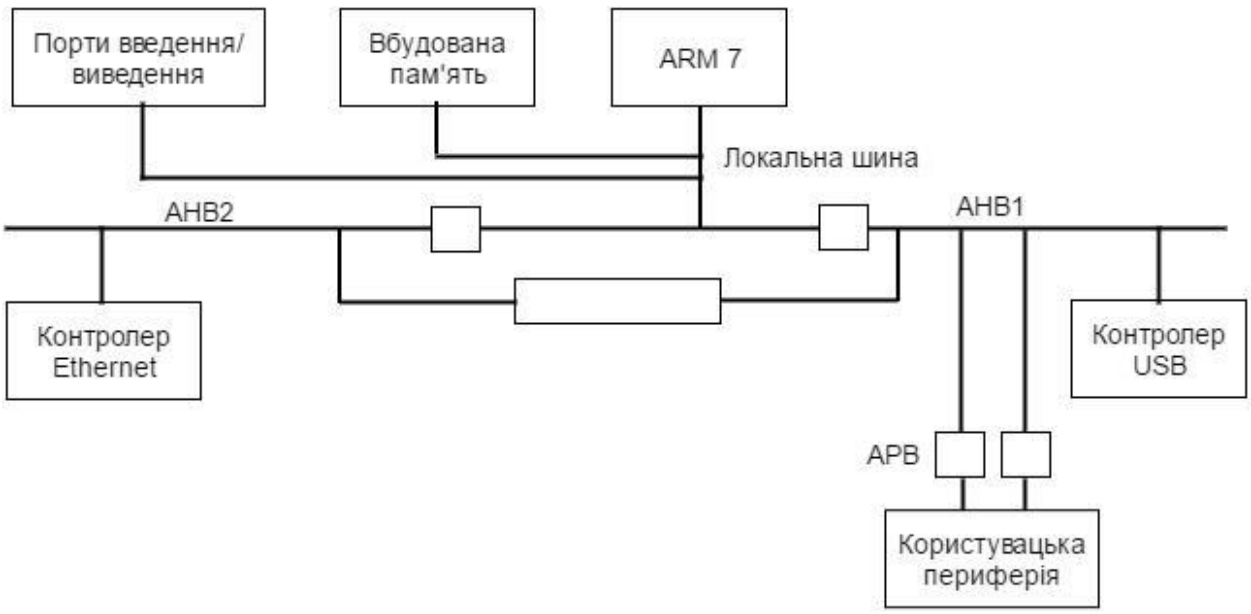

Рисунок 3.2 – Внутрішні шини мікроконтролерів LPC2300

Контролер Ethernet MAC і пов'язаний з ним модуль DMA підключені до власної високошвидкісної шини AHB. Контролер USB і його модуль DMA розміщені на другий шині AHB разом з модулем DMA загального призначення. Всі периферійні пристрої загального призначення підключені до цієї шини через міст AHB/APB.

Існує ще третя локальна шина, яка використовується для підключення вбудованих FLASH–пам'яті і ОЗП до ЦПП. Зрозуміло, пам'ять програм і даних можна було б підключити до ЦПП через шину AHB, проте в цьому випадку виконання програми періодично припинялося б через конфлікти на шині. Використання окремих локальних шин виключає можливість виникнення подібних затримок, що забезпечує більш високу продуктивність процесора. У зв'язку з тим, що операції читання/запису регістрів на локальній шині виконуються набагато швидше, компанія NXP розмістила регістри модуля GPIO на локальній шині. Таке рішення дозволило значно збільшити частоту опитування портів порівняно з попереднім сімейством, в якому ці регістри були підключені до шини APB.

3.3 Системні тактові сигнали

У мікроконтролерах LPC2300 є два основних внутрішніх тактових сигнали. Перший з них називається Cclk і використовується для тактування ЦПП ARM7 і периферійних пристроїв, які підключено до шини AHB, включаючи контролер USB, контролер Ethernet і модуль DMA загального призначення. Другий внутрішній тактовий сигнал, Pclk, служить для тактування периферійних пристроїв, які підключено до шини APB. Для формування обох зазначених сигналів може використовуватися будь–який з трьох джерел тактового сигналу – вбудований RC–генератор, зовнішній генератор і зовнішній «годинниковий» кварцовий резонатор (рисунок 3.3).

Для керування системою тактування призначено декілька регістрів, які показано на рисунку 3.4.

За замовчуванням, після скидання мікроконтролери сімейства LPC2300 працюють від внутрішнього RC–генератора з номінальною частотою 4 МГц.

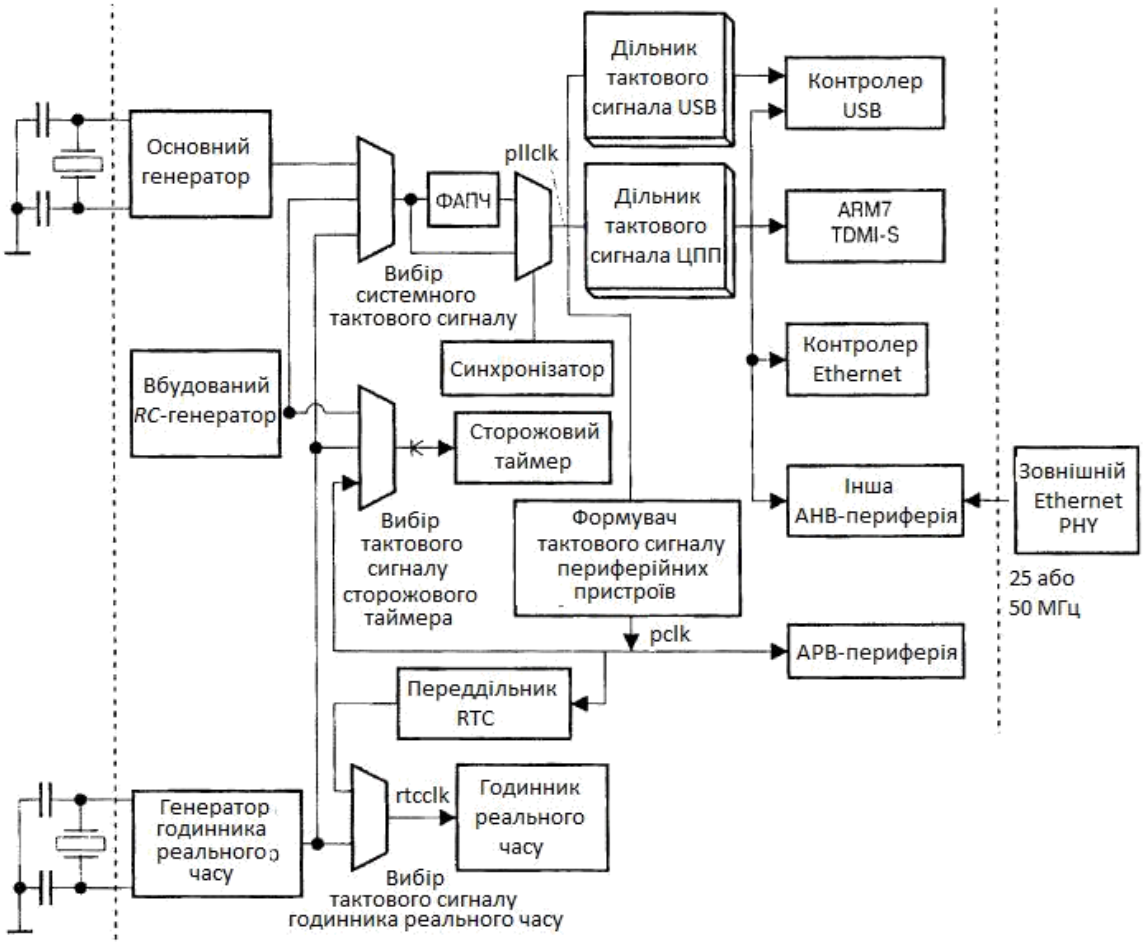

Рисунок 3.3 – Формування системних тактових сигналів

Рисунок 3.4 – Регістри керування системою тактування

Сигнал від цього генератора можна безпосередньо використати для тактування мікроконтролера, а можна подати на вхід схеми ФАПЧ. Однак цей генератор не є досить точним, щоб від нього можна було б тактувати контролер USB. Для даної мети можна використовувати тільки основний генератор із зовнішнім резонатором. Наявність RC–генератора дозволяє мікроконтролеру приступити до виконання команд перш, ніж сигнал основного генератора стане стабільним. Завдяки цьому забезпечується швидкий запуск після скидання або

швидкий вихід з режиму Power Down. Крім того, частота RC–генератора фіксована і відома завантажувачу, який завжди запускається після апаратного скидання. Після запуску користувацької програми в ній за допомогою регістра вибору джерела тактового сигналу CLKSRCSEL можна задати джерело системного тактового сигналу (рисунок 3.5). Цей регістр визначає, який з генераторів буде підключений до входу схеми ФАПЧ.

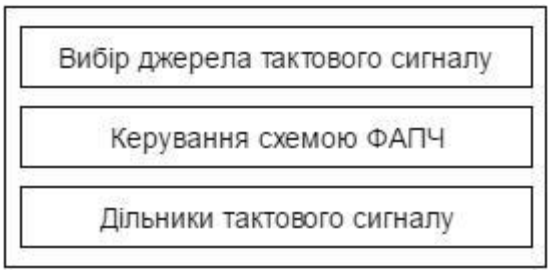

Рисунок 3.5 – Група регістрів для керування системним тактовим сигналом

Після виходу зі стану скидання мікроконтролери LPC2300 працюють від внутрішнього RC–генератора, а основний генератор при цьому вимкнений. Тому у стартовому коді необхідно спочатку дозволити роботу зовнішнього генератора, встановивши біт OSCEN регістра керування і стану системи SCS в 1 (рисунок 3.6). Також необхідно записати коректне значення в біт OSCRANGE для

вказівки робочого діапазону генератора: 1…20 МГц або 15…24 МГц.

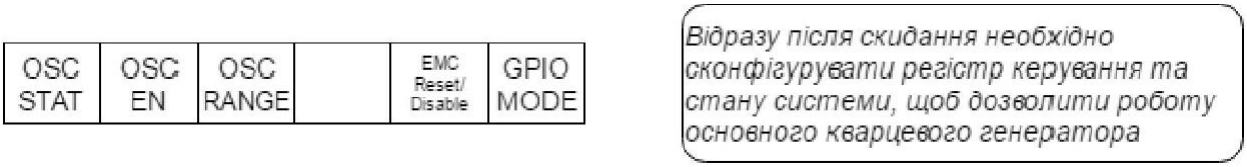

Рисунок 3.6 – Регістр керування та стану системи

Коли сигнал включеного генератора стане стабільним, біт OSCSTAT регістра встановиться в 1. Після цього можна задіяти генератор як джерело системного тактового сигналу, записавши необхідне значення в полі CLKSRC регістра CLKSRCSEL. Однак перед перемиканням джерела системного тактового сигналу треба переконатися, що схема ФАПЧ відключена. Це потрібно зробити, тому що завантажувач включає схему ФАПЧ і залишає її включеною після переходу до користувацької програми.

3.4 Тактові сигнали периферійних пристроїв

Тактові сигнали для кожного з периферійних пристроїв, підключених до шини APB, формуються з тактового сигналу ЦПП Cclk. Для керування частотами тактових сигналів призначені два регістри вибору тактового сигналу периферії PCLKSEL0 і PCLKSEL1, в яких кожному периферійному пристрою відповідає два біти. Заносячи в ці біти різні значення, можна задавати коефіцієнт ділення сигналу Cclk, який дорівнює 1, 2 або 4. За замовчуванням після скидання мікроконтролера частота тактового сигналу для всіх периферійних пристроїв становить Cclk/4, тобто вся периферія працює в чотири рази повільніше ЦПП. Це означає, що при запуску мікроконтролера його живлення мінімальне, проте для досягнення максимальної продуктивності необхідно збільшити частоту сигналу Pclk для кожного використовуваного пристрою.

3.5 Керування енергоспоживанням

У всіх мікроконтролерах, які добре спроектовані, струм живлення прямо пропорційний кількості логічних елементів та частоті їх перемикання.
Мікроконтролери сімейства LPC2300 не є винятком. Їх простота і відносно невелика кількість логічних елементів, що складають ядро ARM7, сприяють малому енергоспоживанню. У мікроконтролерах сімейства передбачено 4 режими зниженого енергоспоживання, за допомогою яких здійснюється керування двома окремими групами споживачів. Крім того, тактовий сигнал будь–якого периферійного пристрою можна відключити в будь–який момент часу. Відповідно, енергоспоживанням мікроконтролера можна керувати як в динамічному режимі, так і в статичному, відключивши при старті програми всі невикористовувані модулі.

3.5.1 Група споживачів

У мікроконтролерах LPC2300 є дві окремі групи споживачів (рисунок 3.7). В першу групу входить модуль годинника реального часу і блок СОЗП розміром 2 Кбайт з малим струмом живлення, який називається також батарейним ОЗП.
До другої групи входять ЦПП ARM7 і інші периферійні пристрої. Таке рішення дозволяє відключати більшу частину кристала LPC2300, зберігаючи при цьому критичні змінні в батарейному ОЗП.

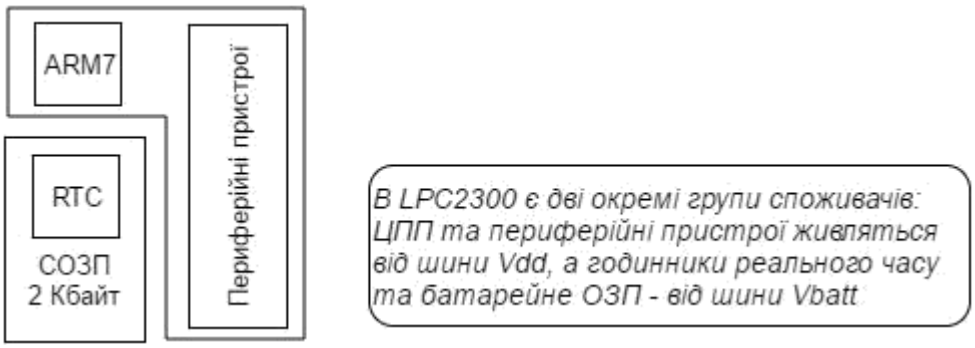

Рисунок 3.7 – Групи споживачів

3.5.2 Режими зниженого енергоживлення

Мікроконтролер може перебувати в одному з чотирьох режимів зниженого енергоспоживання – Idle, Sleep, Power Down і Deep Power Down. Для перемикання мікроконтролера в будь–який з цих режимів використовуються три молодших біти PM [2:0] регістра PCON.

3.5.2.1 Режим Idle

При запису значення 001 в біти PM [2:0] регістра PCON мікроконтролер перемикається в режим Idle. В цьому режимі зупиняється ЦПП, проте всі периферійні пристрої продовжують працювати (рисунок 3.8). Скидання або будь–яке з переривань від периферійних пристроїв призведуть до виходу процесора з сплячого режиму і відновлення виконання програми.

3.5.2.2 Режим Sleep

В цей режим мікроконтролер перемикається при запису в біти PM[2:0] регістра PCON значення 101. У режимі Sleep відключаються ЦПП та всі периферійні пристрої, за винятком годинника реального часу (рисунок 3.9). Також відключається зовнішній генератор і схема ФАПЧ.

Однак щоб забезпечити швидкий вихід мікроконтролера із сплячого режиму, FLASH–пам'ять при цьому переключається в режим Standby, а вбудований RC–генератор продовжує працювати. Вміст ОЗП та регістрів процесора також зберігається. Вихід з режиму Sleep відбувається за перериванням від годинника реального часу або за сигналом зовнішнього переривання. Зрозуміло, скидання також призведе до виходу мікроконтролера із

сплячого режиму. Після відновлення роботи мікроконтролера необхідно буде повторно дозволити роботу основного генератора і схеми ФАПЧ для виконання програми з максимальною швидкістю.

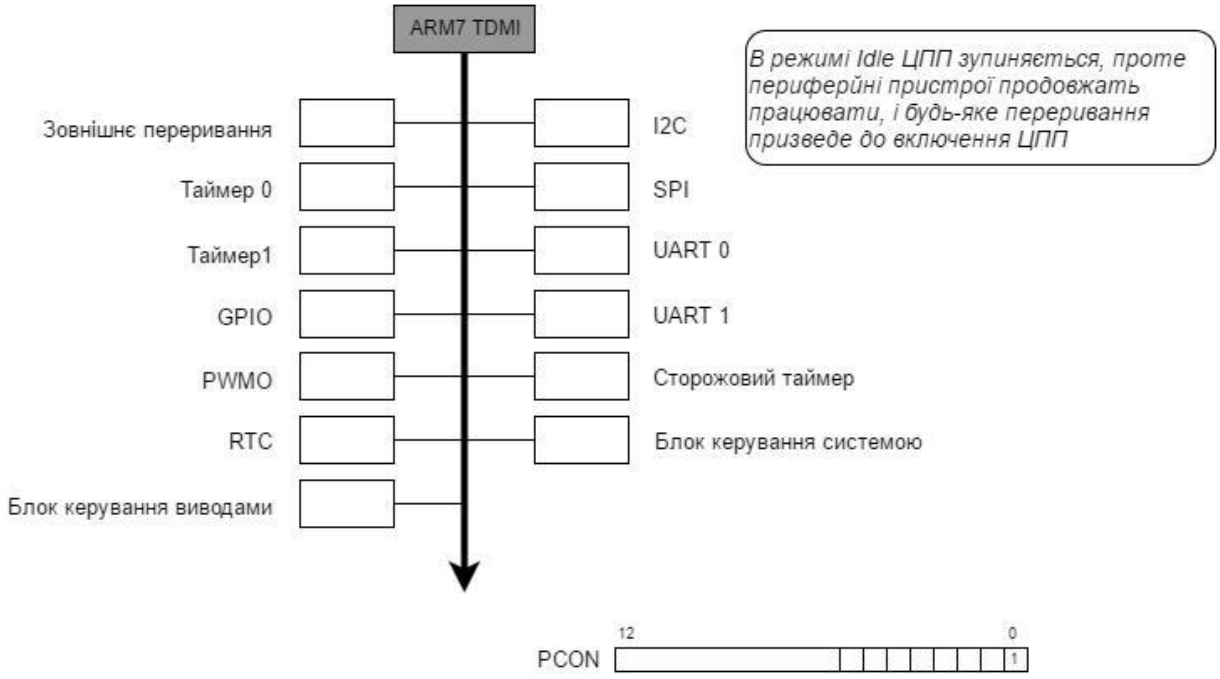

Рисунок 3.8 – Режим Idle

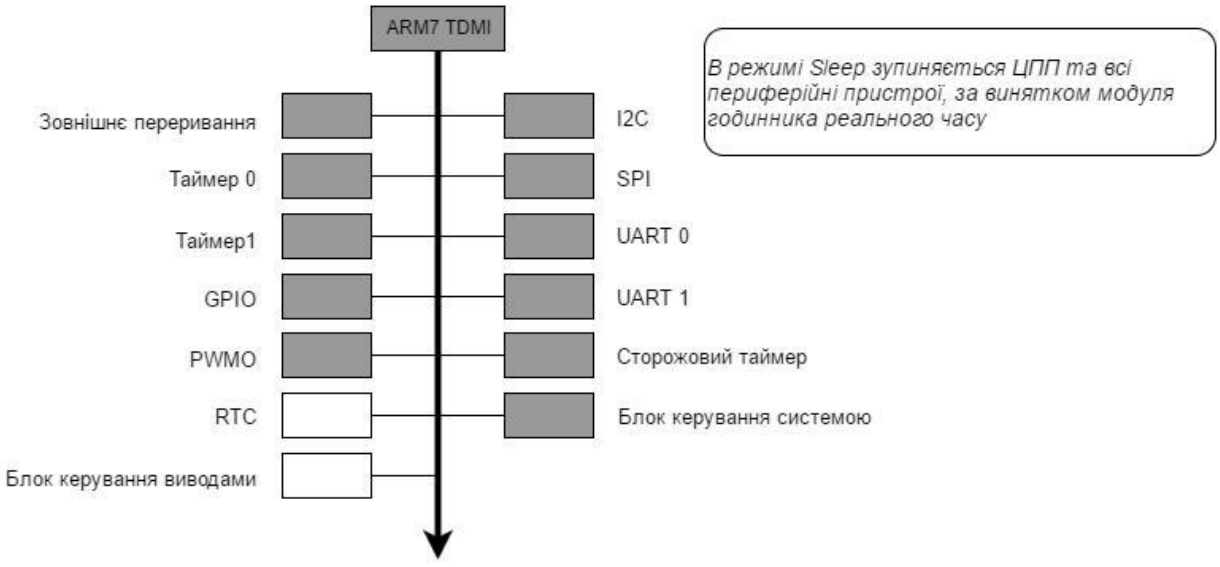

Рисунок 3.9 – Режим Sleep

3.5.2.3 Режим Power Down

Перемикання в режим Power Down здійснюється при записі в біти PM [2:0] регістра PCON значення 010. Цей режим практично ідентичний режиму Sleep, за винятком того, що FLASH–пам'ять також перемикається в режим Power Down. Тому звертатися до FLASH–пам'яті можна буде тільки через 100 мкс після виходу мікроконтролера із сплячого режиму.

3.5.2.4 Режим Deep Power Down

В режим Deep Power Down мікроконтролер перемикається при записі в біти PM [2:0] регістра PCON значення 110. У цьому режимі кристал повністю відключається від шини живлення, відповідно вміст СОЗП та регістрів процесора втрачається. Таким чином, в даному режимі досягається найменше енергоспоживання мікроконтролера. Однак якщо на вивід Vbat при цьому присутня напруга живлення, то годинник реального часу продовжить працювати, а вміст батарейного ОЗП зберігається. Вийти із зазначеного режиму мікроконтролер може або за перериванням від годинника реального часу, або в результаті скидання. Після того, як мікроконтролер вийде з режиму Deep Power Down, він продовжить працювати так само, як після апаратного скидання.

3.5.2.5 Керування енергоспоживанням периферійних пристроїв

Для включення/відключення тактових сигналів периферійних пристроїв служить регістр керування енергоспоживанням PCONP. Кожному периферійному пристрою, наявного в складі мікроконтролера, відповідає один біт даного регістра. Якщо цей біт встановлений в 1, то тактовий сигнал надходить на периферійний пристрій, а якщо скинутий в 0 – не надходить. Зазначений регістр дозволяє відключати невикористовувані периферійні пристрої, зменшуючи, таким чином, сумарне енергоспоживання мікроконтролера. Також даний регістр може використовуватися для динамічного включення/відключення периферійних пристроїв при інтелектуальному керуванні енергоспоживанням. Після скидання більшість периферійних пристроїв включено, проте найбільш складні і "ненажерливі" вимкнені. Перед

зверненням до будь–якого з регістрів цих периферійних пристроїв потрібно переконатися, що на відповідний пристрій подається тактовий сигнал.

За замовчуванням після скидання в вимкненому стані знаходяться такі периферійні пристрої:

- АЦП;
- контролери CAN;
- таймери 2 та 3;
- UART 2 та 3;
- інтерфейс I^2C;
- інтерфейс SD;
- модуль DMA загального призначення;
- контролер Ethernet;
- контролер USB.

Під час розробки програми можуть використовуватися налагоджувальні засоби, які підключаються до виділених виводів ARM7 за інтерфейсом JTAG. Тому після переведення ЦПП в режим Idle або Power Down подальше налагодження буде можливе тільки після «пробудження» ЦПП.

3.6 Організація пам’яті

3.6.1 Загальна характеристика

Пам’ять мікроконтролерів ARM7 має архітектуру Фон–Неймана, що означає, що мікропроцесори мають лінійний адресний простір [1, 4, 7]. Тобто, програми та дані зберігаються в одній пам’яті. З однієї сторони це призводить до того, що мікропроцесор не здатний відокремлювати дані від програм, а з іншої до того, що над командами, що зберігаються в пам’яті, можливо виконувати такі самі операції як і над даними (копіювання тощо).

Архітектура Фон–Неймана біль гнучка ніж Гарвардська, проте, вона має один суттєвий недолік. Через те що команди та дані зберігаються в одній пам’яті, то і доступ до них виконується через одну шину даних. Це призводить до того, що водночас процесор не може отримувати з пам’яті нові програмні інструкції та оперувати даними в пам’яті. Інакше кажучи, вузьким місцем архітектури Фон–Неймана є пропускна здібність шини між процесором і пам’яттю.

При побудові архітектури для сімейства мікропроцесорів ARM7 в основу було покладено ідею мінімізації використання пам’яті для досягнення максимальної швидкодії.

Пам’ять мікропроцесорів ARM7 може бути охарактеризована як неструктурований адресний простір розміром у 4 гігабайти.

При цьому виконується вирівнювання інформації за розміром комірки, що була обрана для її зберігання, тобто 32– бітні слова вирівнюються по 4 байти, а 16– бітні напівслова по 2 байти.

При зберіганні підтримується порядок байтів як від молодшого до старшого (так званий порядок байтів «Little–endian»), так і від старшого до молодшого (так званий порядок байтів «Big–endian»).

Увесь доступний простір пам’яті розбивається на окремі ділянки різних розмірів:

– супер секції – 16 МБ блоки пам’яті;

– секції – 1 МБ блоки пам’яті;

– великі сторінки – 64 КБ блоки;

– маленькі сторінки – 4 КБ блоки.

Подібні блоки пам’яті виділяються просторам пам’яті конкретних застосувань.

Не дивлячись на те, що пам’ять не має фізичної структуризації, вона має логічну ієрархію та поділяється на 4 рівні:

1) кеш 1– го рівня та сильнозв’язна пам’ять;
2) кеш 2– го рівня;
3) оперативна пам’ять (енергозалежна);
4) постійна пам’ять (енергонезалежна).

Розрізнення рівнів пам’яті є цілком логічним. Фізично пам’ять являє собою звичайне підключення модулів ОЗП до процесора. При цьому, кожен наступний рівень має меншу швидкодію та більш простий та дешевий у реалізації.

Шина, що знаходиться між процесором та пам’яттю, має 4 режими роботи, а саме:

1) холостий цикл;
2) послідовний цикл;
3) непослідовний цикл;
4) цикл передачі регістра сопроцесора.

В режимі холостого циклу обмін інформацією між процесором та пам’яттю не виконується.

В режимі послідовного циклу процесор виконує звернення до інформації, що знаходиться у комірках пам’яті, які фізично розташовані одна за одною або на відстані не більше 32 біти.

В режимі непослідовного циклу процесор виконує звернення до інформації, що зберігається у комірках пам’яті, які фізично розташовані на відстані більше ніж 32 біти одна від одної.

В режимі циклу передачі регістра сопроцесору процесор не виконує звернення до пам’яті, а лише передає керуючі регістри сопроцесору.

Пам’ять має логічну організацію, яку зображено на рисунку 3.10.

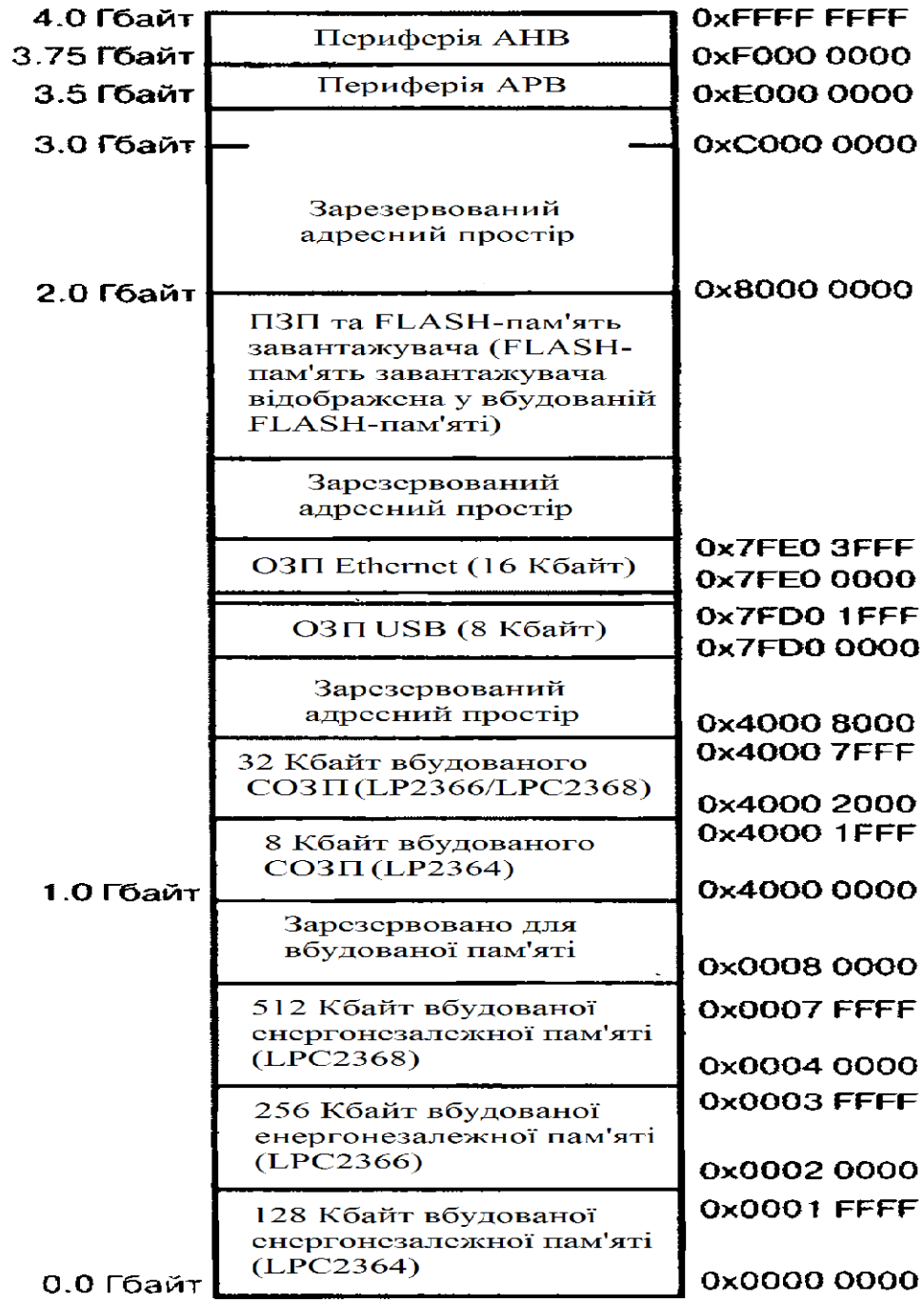

Рисунок 3.10 – Логічна організація пам’яті ARM7

Вбудована FLASH–пам'ять розташовується, починаючи з адреси 0x00000000, а користувальницький ОЗП – з адреси 0x40000000. Ділянки ОЗП, які використовуються в якості буферів модулів USB і Ethernet, розташовані починаючи з адрес 0x7FD00000 і 0x7FE0000 відповідно.

При виготовленні мікроконтролерів сімейства LPC2300 в пам’ять записується програма завантажувача FLASH–пам'яті, а також програма

керування монітором реального часу. Ці програми розміщаються в діапазоні 0x7FFFFFFF ... 0x80000000. Область адрес 0x80000000 ... 0xE0000000 зарезервована для зовнішньої пам'яті. Всі призначені для користувача периферійні пристрої, які підключено до шини APB, відображено на області пам'яті з адресами 0xE0000000 ... 0xE0200000, при цьому кожному пристрою відводиться сторінка розміром 16 Кбайт. Регістри векторного контролера переривань розміщуються в старших адресах, починаючи з адреси 0xFFFFF000.

Якщо програма користувача намагається звернутися до пам'яті, що знаходиться поза вказаними діапазонами адрес або ж до відсутньої пам'яті в межах цих адрес, процесор згенерує виняткову ситуацію «Відміна» (Abort). Цей механізм жорстко закладено у конструкцію процесора, і його не можна ні змінити, ні відключити.

Ділянки пам’яті, які виділяються для застосування користувачем поділяються на 3 типи:

1) нормальна;
2) пам'ять пристроїв;
3) упорядкована пам’ять.

Належність ділянки пам’яті до одного із цих типів визначається за допомогою встановлення відповідного атрибута. Ці атрибути взаємовиключні та не можуть бути скомбіновані.

Нормальна пам’ять використовується за замовчуванням та підтримує атомарні звернення, транзакції, тощо.

Пам’ять пристроїв використовується, коли робота з пам’яттю можлива лише у послідовній манері. Звернення до пам’яті може призвести до впливу на стан системи, тощо.

Впорядкована пам’ять використовується в тому разі, коли доступ до пам’яті обмежений, звернення до пам’яті може призвести до «побічних ефектів» (вплинути на стан системи) або при кожному звернені до однієї і тієї ж ділянки пам’яті буде отримано різне значення.

Нормальна пам’ять та пам’ять пристроїв може бути помічена як пам’ять загального користування (shared) або пам’ять приватного користування (non–shared).

Впорядкована пам’ять за замовчуванням вважається пам’яттю загального користування та це не може бути змінено. Якщо пам’ять помічена як «пам’ять загального користування», це означає що з нею працюють декілька процесорів одночасно.

Крім цього, нормальна пам’ять може бути позначена одним з атрибутів, що керують кешуванням даних:

1) некешована пам’ять;
2) кеш наскрізний;
3) кеш зі зворотним записом.

Некешована пам’ять ніяким чином не кешується і при зверненні до неї відбувається безпосереднє звернення до ділянки пам’яті, в якій зберігається оригінальне значення.

Кеш наскрізний працює таким чином, що кожен раз при зверненні до однієї і тієї ж ділянки пам’яті її значення переноситься до пам’яті наступного рівня, що є більш швидкою.

Кеш зі зворотним записом працює таким чином, що дані кешуються у найшвидшій доступній ділянці пам’яті при першому зверненні, а синхронізація значення кешу з оригінальним (що розташоване у більш повільній пам’яті) виконується лише при переміщенні кешу.

Загалом, ARM7 підтримує 3 режими роботи з пам’яттю з точки зору адресації:

– робота з усім простором пам’яті одночасно (flat addressing);
– захищений режим (Protected Memory System Architecture);
– віртуальний адресний простір (Virtual Memory System Architecture).

В режимі роботи з усім простором пам’яті одночасно процесор оперує реальними фізичними адресами усієї доступної пам’яті.

В захищеному режимі процесор також оперує фізичними адресами, але при цьому пам’ять поділяється на спеціальні ділянки, кожна з яких характеризується розміром та правами, необхідними для того, щоб працювати з нею.

В режимі роботи з віртуальним адресним простором, процесор оперує лише певними віртуальними адресами, що транслюються у реальні за допомогою модуля транслятора віртуальних адрес при зверненні до реальних комірок пам’яті. При цьому, в режимі роботи з віртуальним адресним простором також можлива перевірка прав доступу до ділянок пам’яті, але вважається що це є лише додатковою можливістю через те, що в захищеному режимі ця перевірка виконується простіше.

За замовчуванням використовується режим роботи з усім простором пам’яті одночасно. Керуванням режимом роботи з пам’яттю виконується за допомогою підключення або відключення відповідних модулів до процесора.

Деякі команди ARM (наприклад LDR, STR) для доступу до пам’яті можуть використовувати відносну регістрову адресацію зі зміщенням (рисунок 3.11).

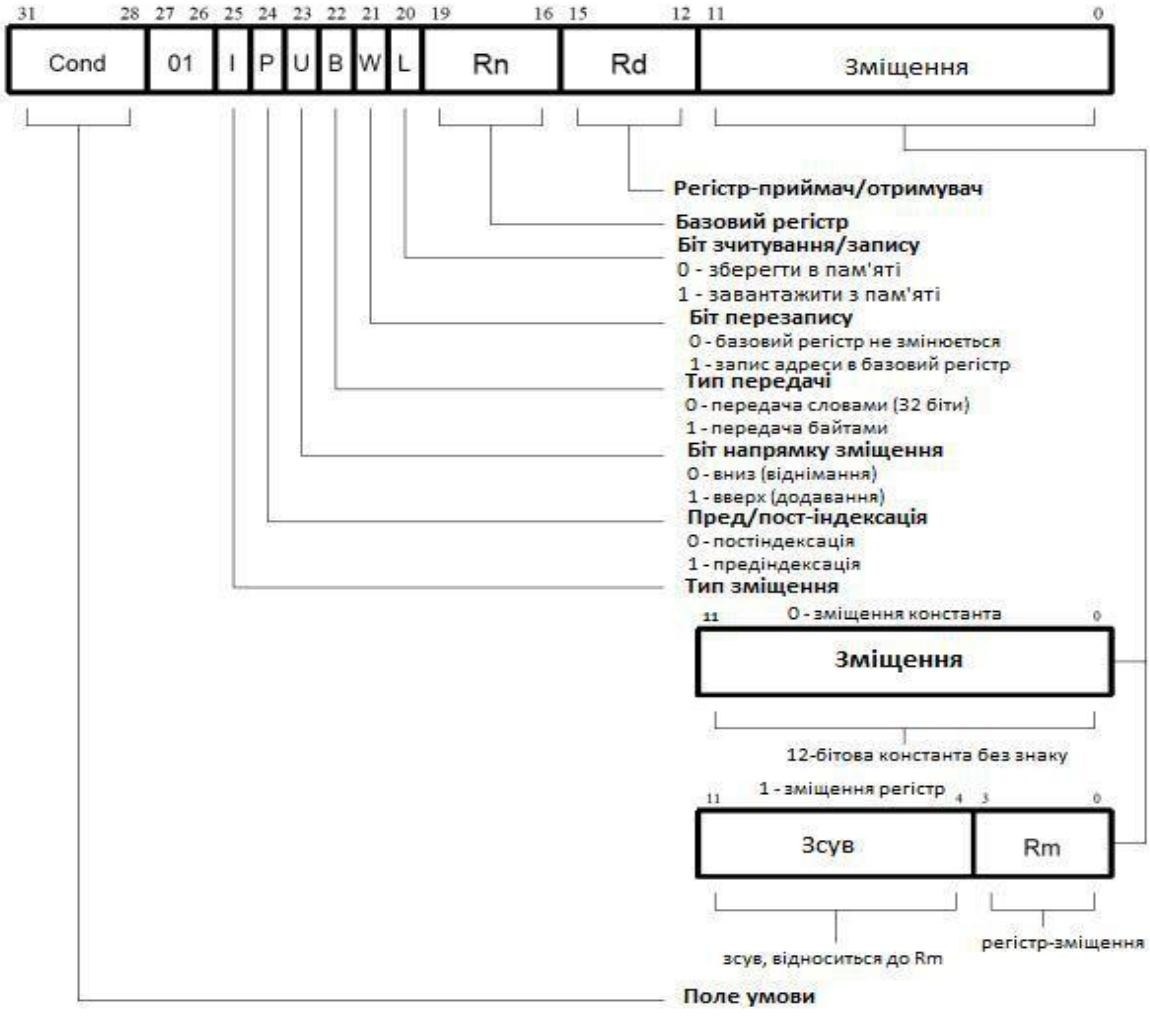

Рисунок 3.11 – Команди з відносною адресацією

Адреса комірки пам'яті, яка використовується у цих командах передачі, обчислюється за допомогою додавання або віднімання з деяким зміщенням щодо зазначеного базового регістра. Якщо потрібна автоіндексація, то результат цього обчислення може бути записаний назад в цей же базовий регістр.

Зсув відносно базового регістра може бути задано або у вигляді 12– бітної константи, зазначеної в коді команди, або у вигляді другого регістра– операнда (при виконанні команди його вміст може бути піддано логічному зсуву), а саме зміщення може бути або додано (U=1), або віднято (U=0) з базового регістра Rn. Такі модифікації зміщення можуть бути виконані або до виконання пересилання (преіндексація, P=1) або після неї (постіндексація, P=0).

Біт W дозволяє вибрати режим адресації: інкремент або декремент. Модифіковане значення базової адреси або записується назад в нього ж (W = 1), або залишається без змін (W = 0). У разі постіндексної адресації біт W втрачає сенс і тому обнуляється, в цьому випадку можна зберегти колишнє значення базової адреси за допомогою встановлення зміщення, яке дорівнює нулю. Передача даних з постіндексацією завжди модифікує вміст базової адреси. Детальніше це питання розглянуто в [4].

3.6.2 Модуль прискорення роботи пам’яті

Модуль прискорення роботи пам'яті (Memory Accelerator Module – MAM) призначено для підвищення швидкості виконання команд в мікроконтролерах сімейства LPC2300 [1, 4, 7]. Модуль MAM підключається до локальної шини ЦПП ARM7 і розміщується між ним і FLASH– пам'яттю (рисунок 3.12).

Одним з основних обмежуючих факторів, що перешкоджають створенню високопродуктивних однокристальних мікроконтролерів на базі ядра ARM7, є досить великий час доступу до вбудованої FLASH–пам'яті. Процесор ARM здатний працювати на частотах до 80 МГц, однак час доступу до вбудованої FLASH–пам'яті становить цілих 50 нс. Відповідно, вже одне тільки використання FLASH– пам'яті обмежує швидкість виконання програми на рівні 20 МГц (в чотири рази менше максимальної тактової частоти процесора).

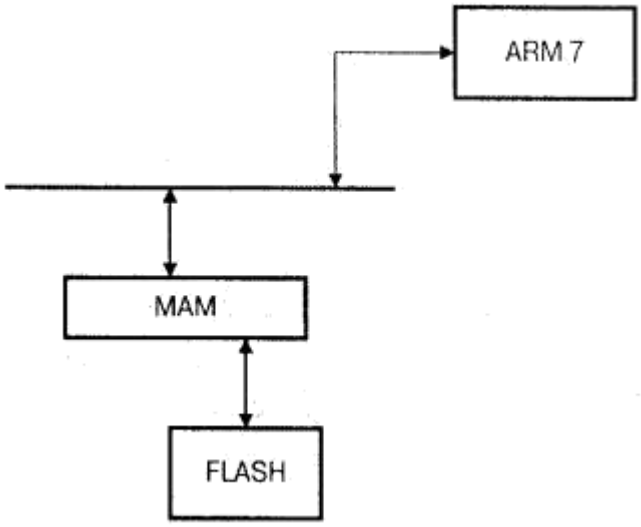

Рисунок 3.12 – Модуль прискорення роботи пам’яті

Існує кілька шляхів вирішення цієї проблеми. Найпростіший полягає в завантаженні критичних секцій програми в ОЗП і запуск їх вже звідти. Оскільки ОЗП має набагато менший час доступу, загальна продуктивність набагато збільшиться. Недолік такого рішення полягає в тому, що вбудований ОЗП є обмеженим і цінним ресурсом. Використання його для зберігання команд суттєво обмежує розмір створюваних нами програм. Інший спосіб полягає у використанні вбудованого кеша. Кешем називається область пам'яті невеликого розміру, розташована між процесором і основною пам'яттю, де зберігається вміст ділянок основної пам'яті, до яких зверталися в останній раз. У разі правильно спроектованого кеша процесор буде використовувати його, де тільки можливо, зменшуючи, таким чином, вплив повільної пам'яті. Однак повнофункціональний кеш є складним периферійним пристроєм, для реалізації якого потрібна величезна кількість логічних елементів і, відповідно, значна частина площі кристала мікроконтролера. Це суперечить основній ідеї ядра ARM7–простоті. Іншим недоліком використання повнофункціонального кеша є непередбачуваність часу виконання коду, який використовує цей кеш. Відповідно цей спосіб не можна застосовувати в додатках, які повинні бути передбачуваними і повторюваними.

Модуль MAM (рисунок 3.13) є компромісом між складністю повно функціонального кеша і простотою прямого доступу до FLASH– пам'яті.

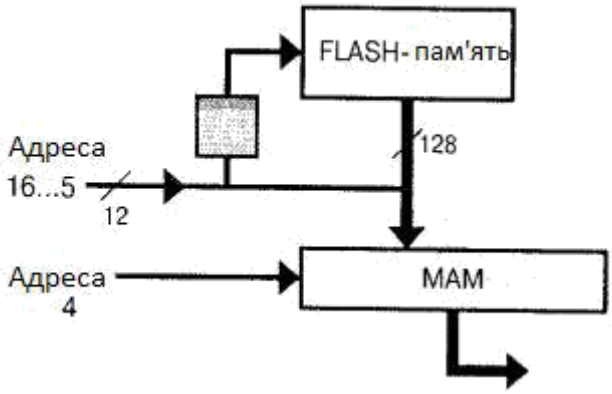

Рисунок 3.13 – Робота модуля MAM

Подібно кешу, модуль MAM намагається своєчасно забезпечити в своїй локальній пам'яті наявність такої команди ARM, яку повинен буде виконати ЦПП. Для цього FLASH–пам'ять має не 32–бітну, а 128–бітну організацію. Таким чином, при одному зверненні до FLASH–пам'яті можна завантажити в швидку локальну пам'ять модуля чотири команди ARM або вісім команд THUMB. У цій локальній пам'яті розташовані буфери попередньої вибірки, даних і переходів. Особливо добре цей метод працює при використанні команд ARM, які для «згладжування» коротких переходів можуть використовувати умовне виконання, щоб отримати максимально можливу лінійність коду. Для підтримки коротких циклів і переходів в модулі MAM передбачені спеціальні буфери переходів, що зберігають останні завантажені команди, які при необхідності можна виконати повторно.

Вся ця складна структура модуля MAM абсолютно прозора для користувача, а його конфігурація визначається двома регістрами – регістром таймінгу MAMTIM і регістром керування MAMCR. Крім того, є кілька додаткових регістрів, що містять інформацію про поточну ефективність роботи модуля. Регістр таймінгу використовується для завдання відповідності між тактовою частотою ЦПП і часом доступу до FLASH– пам'яті. Значення, яке записано в трьох молодших бітах регістра таймінгу, відповідає кількості тактів тактового сигналу ЦПП, яке потрібно модулю MAM для звернення до FLASH–пам'яті. При частоті системного тактового сигналу, меншою 20 МГц, в ці біти рекомендується заносити число 1, при частотах 20…40 МГц – 2, а при частоті понад 40 МГц – 3.

Регістр керування MAM використовується для завдання режиму роботи модуля (рисунок 3.14).

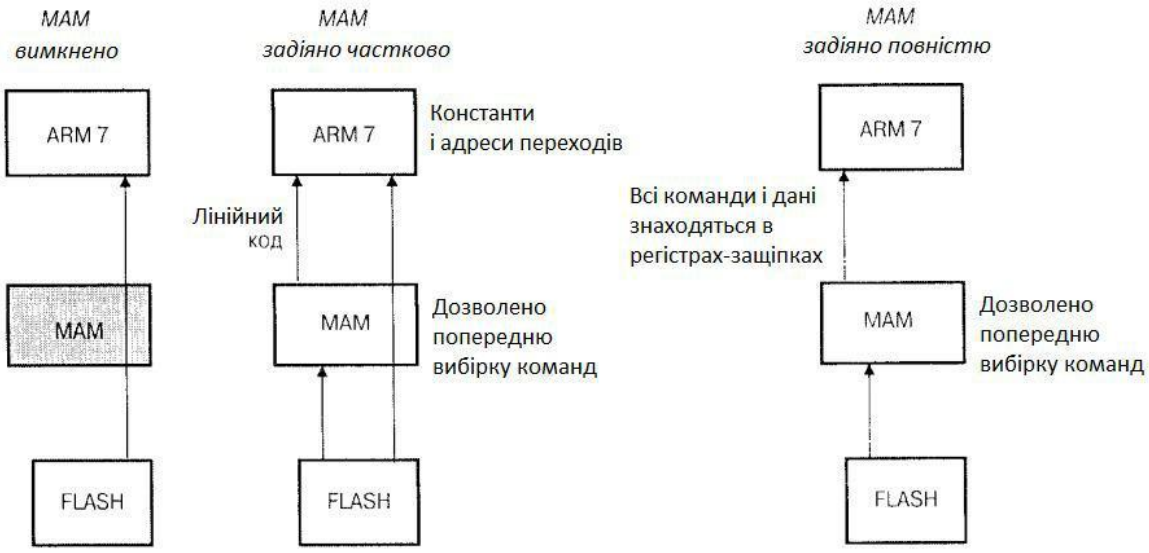

Рисунок 3.14 – Режими роботи модуля MAM

Після скидання модуль MAM вимкнений і всі команди і значення констант зчитуються безпосередньо з FLASH– пам'яті. Модуль MAM можна включити частково так, щоб лінійний код зчитувався б з модуля, а адреси переходів і значення констант – безпосередньо з FLASH– пам'яті. І нарешті, модуль MAM можна задіяти повністю. В цьому випадку всі звернення до FLASH–пам'яті будуть проводитися через нього. Сенс в наявності даних режимів полягає в тому, що, як і при кешуванні, час виконання коду з модуля MAM–величина невизначена. Тому, якщо необхідно забезпечити конкретний час виконання коду, ми можемо відключити модуль або зменшити його вплив. Проте, навіть якщо модуль MAM задіяний повністю, його вплив помітно набагато менше, ніж кешу. Зокрема, використовуючи інструментарій аналізу продуктивності (performance analysis) засобів розробки, можна передбачити продуктивність коду.

Щоб полегшити цей аналіз, а також для того, щоб оцінити ефективність роботи модуля MAM, в модулі передбачено кілька регістрів для збору статистичної інформації (рисунок 3.15), які можна використовувати для вимірювання продуктивності модуля.

Рисунок 3.15 – Регістри для оцінки ефективності роботи MAM

Основними серед цих регістрів є два лічильника, які відстежують звернення до FLASH– пам'яті і до буферів модуля MAM. Регістр керування дозволяє задати умову, за якої буде відбуватися інкрементування лічильника. Змінюючи вміст регістра керування, ми можемо розрізняти вибірку констант і вибірку команд, що дозволяє нам визначати частоту вибірок команд, частоту вибірок даних або загальну частоту вибірок. Грунтуючись на цих вимірах, можна в якійсь мірі визначити ефективність використання модуля MAM в програмі.

3.6.3 Приклад конфігурації модуля MAM

Наведений нижче приклад починає виконуватися при частоті вихідного сигналу схеми ФАПЧ, яка дорівнює 60 МГц, і вимкненому модулі MAM. Програма послідовно, з деякою затримкою, короткочасно вмикає світлодіоди. В цей же час виконується аналого–цифрове перетворення і, якщо результат перетворення перевищує 0x00000080, вмикається модуль MAM для досягнення максимальної швидкості виконання програми. Роботу модуля MAM можна помітити візуально за збільшенням частоти спалахів світлодіодів.

```
int main(void)

{

unsigned int delay;
```

```
unsigned int FLASHer = 0x00010000; //оголошуємо локальні змінні


I0DIR1 = 0x00FF0000; //всі порти – виходи
APBDIV = 0x02;
ADCR = 0x00270601; //ініціалізуємо АЦП: 10 біт, AIN0, 3 МГц
ADCR |= 0x01000000; //Запускаємо перетворення
while(1)
{
do
{
val = ADDR; //Зчитуємо регістр даних АЦП
}
while((val & 0x80000000) == 0);
val = ((val » 6) & Ox03FF);
if(val < 0x80)
{
MAMCR = 0;
MAMTIM = 0x03:
MAMCR = 0x02;
}
else
{
MAMCR = 0x0;
}
for(delay=0;delay<0xl00000; delay++) //Цикл затримки
{
}

ChangeGPIOPinState(FLASHer); //Виводимо значення в
порт FLASHer = FLASHer << 1; //Зсуваємо маску
```

```
if(FLASHer & 0x01000000)
{
FLASHer = 0x00010000; //Перевіряємо маску на переповнення
}
}
}
void ChangeGPIOPinState(unsigned int state)
{
IOCLR1 = state; //Скидаємо відповідні виводи IOSET1 =
state; //Встановлюємо відповідні виводи
}
```