МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
КАФЕДРА АВТОМАТИКИ ТА УПРАВЛІННЯ У ТЕХНІЧНИХ СИСТЕМАХ

Архітектура новітніх мікроконтролерів
Периферійні модулі мікроконтролерів сімейства ARM

Навчальний посібник
для студентів спеціальності 151 «Автоматизація та комп’ютерно–інтегровані технології»

Київ КПІ ім. Ігоря Сікорського 2017

`

Архітектура новітніх мікроконтролерів: Периферійні модулі мікроконтролерів сімейства ARM: Навчальний посібник для студентів спеціальності 151 «Автоматизація та комп'ютерно–інтегровані технології» кафедри Автоматики та управління у технічних системах / Автор:

А.О. Новацький– К: „КПІ ім. Ігоря Сікорського", 2017–354 с.

Відповідальний за випуск: Л.Ю. Юрчук

Рецензенти: А.Ю.Дорошенко, О.А.Чемерис

Посібнику надано гриф «Затверджено вченою радою КПІ ім. Ігоря Сікорського»

Протокол № від 2017 р

Навчальний посібник охоплює теоретичний матеріал та практичні завдання, які необхідні для вивчення базової частини дисципліни «Архітектура новітніх мікроконтролерів». Робота містить чотири розділи, в яких розглядаються наступні питання: склад та основні характеристики мікроконтролерів сімейства ARM; структура типового ARM–мікроконтролера; модуль таймерів/лічильників; паралельні порти; підсистема переривань; послідовний асинхронний інтерфейс (УАПП); організація пам'яті; послідовний синхронний інтерфейс I^2C; послідовний синхронний периферійний інтерфейс SPI; послідовний асинхронний периферійний інтерфейс CAN; аналого–цифровий перетворювач; цифро–аналоговий перетворювач; модуль ФАПЧ; годинник реального часу; модуль DMA та вартовий таймер.

В кожному розділі розглядаються основні архітектурні особливості периферійних модулів ARM – мікроконтролерів на апаратному рівні, наводяться приклади використання цих модулів.

Робота може бути корисною студентам відповідних спеціальностей при вивченні дисциплін, які пов'язані із проектуванням та використанням мікропроцесорних систем, а також при виконанні бакалаврських та магістерських робіт, курсових та дипломних проектів, в яких використовуються мікропроцесорні пристрої. Останнє було враховано при оформленні роботи, яку виконано згідно вимог до конструкторської документації.

ВСТУП 8
СПИСОК СКОРОЧЕНЬ 9
1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА МІКРОКОНТРОЛЕРІВ СІМЕЙСТВА ARM 13
1.1 RISC–архітектура мікроконтролерів 13
1.2 Загальний огляд мікроконтролерів сімейства ARM 15
1.3 Сімейство ядер ARM 19
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 21
2 СТРУКТУРА ТИПОВОГО МІКРОКОНТРОЛЕРА З ARM–ЯДРОМ22
2.1 Загальні відомості 22
2.2 Мікроконтролер LPC2378 23
2.3 Мікроконтролер STM32 32
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 36
3 СИСТЕМНІ ПЕРИФЕРІЙНІ ПРИСТРОЇ 37
3.1 Загальні відомості 37
3.2 Внутрішні шини 37
3.3 Системні тактові сигнали 39
3.4 Тактові сигнали периферійних пристроїв 42
3.5 Керування енергоспоживанням 42
3.5.1 Група споживачів 42
3.5.2 Режими зниженого енергоживлення 43
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 47
3.6 Організація пам’яті 48
3.6.1 Загальна характеристика 48
3.6.2 Модуль прискорення роботи пам’яті 54
3.6.3 Приклад конфігурації модуля MAM 58
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 61
3.7 Модуль ФАПЧ 62
3.7.1 Принцип роботи схеми ФАПЧ 62

3.7.2 Модуль ФАПЧ мікроконтролерів сім’ї LPC23XX 66
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 82
3.8 Система переривань 83
3.8.1 Загальні відомості 83
3.8.2 Блок керування виводами..... 84
3.8.3 Виводи зовнішніх переривань 84
3.8.4 Структура переривань 85
3.8.5 Невекторні переривання..... 91
3.8.6 Програмні переривання..... 91
3.8.7 Вкладені переривання 92
3.8.8 Джерела переривань..... 94
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 97
3.9 Модуль прямого доступу до пам’яті 98
3.9.1 Загальна характеристика..... 98
3.9.2 Огляд модуля DMA 99
3.9.3 Синхронізація DMA 100
3.9.4 Пересилання з пам'яті в пам'ять 100
3.9.5 Пакетна передача..... 102
3.9.6 Підтримка модулем DMA периферійних пристроїв..... 102
3.9.7 Пересилання несуміжних даних..... 103
3.9.8 Регістри керування модулем DMA 103
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 121
4 ПЕРИФЕРІЙНІ ПРИСТРОЇ ЗАГАЛЬНОГО ПРИЗНАЧЕННЯ 122
4.1 Загальні положення 122
4.2 Порти введення/виведення загального призначення..... 122
4.2.1 Загальні відомості 122
4.2.2 Швидкі регістри введення/виведення 124
4.2.3 Опис керуючих регістрів..... 126
4.2.4 Регістр значення виводів GPIO..... 127

4.2.5 Регістр встановлення GPIO 128
4.2.6 Регістр скидання GPIO 128
4.2.7 Регістр напрямку GPIO 129
4.2.8 Особливості програмування GPIO 129
4.2.9 Переривання від GPIO 129
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 131
4.3 Таймери загального призначення 132
4.3.1 Режим захоплення 132
4.3.2 Режим лічильника зовнішніх подій 133
4.3.3 Режим збігу 133
4.3.4 Опис керуючих регістрів модуля таймерів 135
4.3.5 Регістри переривань IR 138
4.3.6 Регістри керування таймерами T[0/1/2/3]CR 139
4.3.7 Регістри керування підрахунком 140
4.3.8 Регістри таймерів/лічильників T0TC … T3TC 140
4.3.9 Регістри масштабування T0PR … T3PR 142
4.3.10 Регістри лічильників масштабування T0PC … T3PC 142
4.3.11 Регістри збігу MR0 … MR3 142
4.3.12 Регістри керування збігом T[0/1/2/3]MCR 142
4.3.13 Регістри захоплення CR0 та CR1 144
4.3.14 Регістри керування захопленням T[0/1/2/3]CCR 145
4.3.15 Регістри зовнішнього збігу T[0/1/2/3]EMR 146
4.3.16 Структура модуля таймерів/лічильників 148
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 150
4.4 Модуль ШІМ 151
4.4.1 Загальна характеристика 151
4.4.2 Режим лічильника 155
4.4.3 Структурна схема модуля широтно–імпульсного модулятора 155
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 157

4.5 Сторожовий таймер 158
4.5.1 Загальні відомості 158
4.5.2 Період сторожового таймера..... 158
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 160
4.6 Модуль УАПП 161
4.6.1 Загальні відомості 161
4.6.2 Універсальний асинхронний послідовний програмований приймач–передавач (УАПП) 161
4.6.3 Пристрій перетворення рівнів (ППР) 163
4.6.4 Роз'єм RS–232C 164
4.6.5 Модуль УАПП в мікроконтролерах сімейства LPC2300 165
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 192
4.7 Інтерфейс I^2C 193
4.7.1 Загальна характеристика інтерфейсу I^2C 193
4.7.2 Типові особливості інтерфейсу I^2C мікроконтролерів сімейства LPC2300 198
4.7.3Режими роботи інтерфейсу I^2C мікроконтролерів сімейства LЗC2300 200
4.7.4 Опис виводів 228
4.7.5 Опис керуючих регістрів..... 229
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 241
4.8 Інтерфейс SPI..... 242
4.8.1 Основні положення 242
4.8.2 Передача даних в режимі ведучого 248
4.8.3 Передача даних в режимі веденого..... 249
4.8.4 Переривання SPI..... 249
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 255
4.9 Модуль аналого–цифрового перетворювача (АЦП) 256
4.9.1 Основні відомості про АЦП 256

4.9.2 Модуль АЦП сімейства мікроконтролерів LPX23XX 262
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 277
4.10 Модуль цифро–аналогового перетворювача(ЦАП) 278
4.10.1 Загальна характеристика 278
4.10.2 Розрахунок цифро–аналогового перетворювача на матриці R–2R з підсумовуванням напруг 279
4.10.3 Модуль ЦАП мікроконтролера LPC2378 281
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 285
4.11 Модуль годинника реального часу 286
4.11.1 Загальна характеристика 286
4.11.2 Регістри керування модулем RTC 290
4.11.3 Приклад програмування RTC–модуля 297
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 301
4.12 Контролер CAN – інтерфейсу 302
4.12.1 Загальні відомості 302
4.12.2 CAN–модуль ARM – мікроконтролера NXP LPC2378 309
4.12.3 Керуючі регістри контролерів CAN1 і CAN2 у складі мікроконтролера LPC2378 329
4.12.4 Регістри фільтрації повідомлень 344
ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ 351
ПРЕДМЕТНИЙ ПОКАЖЧИК 352
СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ 354

ВСТУП

Курс „Архітектура новітніх мікроконтролерів” належить до циклу дисцилін „Комп”ютерна електроніка та мікропроцесорна техніка” та є одним з основних при підготовці магістра в області комп'ютеризованих систем управління та автоматики. В ньому висвітлюються проблеми проектування, програмування і застосування мікропроцесорних систем на прикладі мікроконтролерів LPC2378 сімейства ARM7.

Архітектура ARM (Advanced RISC Machine) – 32–бітна RISC–архітектура процесорів, яка розроблена компанією ARM Limited. Процесори ARM мають низьке енергоспоживання, тому завоювали сегмент масових мобільних продуктів (стільникові телефони, кишенькові комп'ютери) і знаходять широке застосування у вбудованих системах середньої і високої продуктивності (від мережевих маршрутизаторів і точок доступу до телевізорів). Станом на 2014 р. 95% смартфонів у світі, 80% усіх цифрових камер і 35% усіх електронних пристроїв використовують ARM–технології.

В результаті вивчення дисципліни студент повинен вміти проектувати мікропроцесорні системи на сучасних мікроконтролерах, розробляти алгоритми та керуючі програми на асемблері та мовах високого рівня.

У практичному плані студент повинен навчитися проектувати та програмувати мікропроцесорні системи за модульним принципом, користуватися науково–технічною і додатковою літературою, самостійно опановувати нові питання, які відносяться до даної дисципліни. Вивчення курсу базується на знаннях, набутих у процесі вивчення дисциплін: мікропроцесорні системи, проектування мікропроцесорних систем, архітектура комп'ютерних систем та мереж, комп'ютерна електроніка, алгоритмічні мови програмування, проектування мікропроцесорних систем та мереж.

СПИСОК СКОРОЧЕНЬ

АК – аналоговий компаратор

АЦП – аналого–цифровий перетворювач

БО – безпосередній операнд

БРА – буферний регістр адреси

ГКН– генератор, який керується напругою

ГТІ – генератор тактових імпульсів

ДК – двійковий код

ЗПП – зовнішня пам’ять програм

InterIC, або I^2C–інтерфейс I^2C

КОП – код операції

МЗР– молодший значущий розряд

МК – мікроконтролер

МП – мікропроцесор

МПС – мікропроцесорна система

МПСК – мікропроцесорна система керування

МЦ – машинний цикл

ОЗП – оперативний запам’ятовуючий пристрій

ОК – об’єкт керування

ПВВ – пристрій введення/виведення

ПВЗ – пристрій вибірки і зберігання

ПЗ – програмне забеспечення

ПК – персональний комп'ютер

ПЛМ – програмована логічними матрицями

ППР – пристрій перетворення рівнів

РГ – регістр

РЗП – регістр загального призначення

РКІ – рідкокристалічний індикатор

РКС – регістр керуючого слова

РМ – резистивна матриця

РПД – резидентна пам’ять даних

`

РПН – регістр послідовного наближення

РПП – резидентна пам’ять програм

СБ – старший байт

СД – світлодіод

СЗР – старший значущий розряд

СМА – суматор адреси

СУР – схема узгодження рівнів

СФР – схема формування рівнів

США – системна шина адреси

УАПП – універсальний асинхронний послідовний програмований приймач – передавач

ФАПЧ – фазове автопідлаштовування частоти

ФД – фазовий детектор

ФНЧ – фільтр нижніх частот

ЦАП – цифро–аналоговий перетворювач

ЦОМ – цифрова обчислювальна машина

ЦПП – центральний процесорний пристрій

ШІМ – широтно–імпульсна модуляція

ШФ – шинний формувач

Access Line – тактова частота

ACP – Accelerator Coherency Port

AHB – Advanced High–Performance Bus

AMP – Asymmetric Multi–Processing

APB (Advanced Peripheral Bus) – покращена периферійна шина

ARM – Advanced RISC Machine

Block System Functions – блок системних функцій

CAN (Controller Area Network

CISC – Complex Instruction Set Computing

CLKSRCSEL– регістр вибору джерела тактового сигналу

CPSR (Current Program Status Register) – регістр поточного стану програми

DBX (Direct Bytecode eXecution) – пряме виконання байт–коду

`

DMA – Direct Memory Access – пам'ять прямого доступу

EMC – контролер зовнішньої пам'яті

FIFO – буфер передачі

FIQ – швидке переривання

GIC – Generic Interrupt Controller

GP DMA – контролер прямого доступу до пам'яті загального призначення

GPIO – General Purpose Input Output– порти введення/виведення загального призначення

ICE – In–Circuit Emulator – схемний емулятор

IRQ – переривання загального призначення

IoT – Internet of Things

Java ME – Java Mobile Edition

LER – Latch Enable Register – регістри керування защіпкою

LLPP – Low Latency Peripheral Port

LSB – Least Significant Bit – молодший значущий розряд

MAM –Memory Accelerator Module – модуль прискорення роботи пам'яті

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

MISO – Master–In, Slave–Out– вхідна лінія даних ведучого інтерфейсу і вихідна лінія даних веденого інтерфейсу

MMU (Memory Management Unit) – блок управління пам'яттю

MOSI– Master–Out, Slave–In– вихідна лінія даних ведучого інтерфейсу і вхідна лінія даних веденого інтерфейсу

MPU (Memory Protection Unit) – модуль захисту пам’яті

MSB – Most Significant Bit – старший значущий розряд

NVRAM – Non–Volatile Random–Access Memory – енергонезалежна пам'ять

PLL – Phase Locked Loop

PWM – Pulse width modulator–модуль ШІМ

RISC – Restricted (reduced) Instruction Set Computer

RTC – Real Time Clock – годинник реального часу

SCU – Snoop Control Unit

`

SIMD (Single Instruction, Multiple Data) – одиночний потік команд, множинний потік даних

SMP – Symmetric Multi–Processing

SPI – Serial Peripheral Interface – послідовний синхронний повнодуплексний стандарт передачі даних

SPSR – Saved Program Status Register – регістр зберігання стану програми

SRAM–статична оперативна пам'ять

SWI – команда програмного переривання

TCM – Tightly–Coupled Memory – тісно зв’язана пам'ять

TDMI – Thumb + Debug + Multiplier + ICE

VIC – Vector Interrupt Controller – векторний контролер переривань

`

1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА МІКРОКОНТРОЛЕРІВ СІМЕЙСТВА ARM

1.1 RISC–архітектура мікроконтролерів

У 1980 році, в університеті Берклі, був початий проект RISC (restricted (reduced) instruction set computer). Планувалося створити такий процесор, який би містив лише найнеобхідніші інструкції. Дослідження базувалися на використанні конвеєрної обробки і агресивного використання техніки регістрового вікна. У звичайному процесорі є невелика кількість регістрів, і програма може використовувати будь–який регістр в будь–який час. У процесорі, що використовує технологію регістрового вікна, дуже велика кількість регістрів (наприклад, 128), але програми можуть використовувати обмежену кількість (наприклад, тільки 8 в кожен момент часу).

Програма, обмежена лише вісьмома регістрами для кожної процедури, може виконувати дуже швидкі виклики процедур: «вікно» просто зрушується до 8–регістрового блоку потрібної процедури, а при поверненні з процедури зсувається назад до регістрів цієї процедури. У звичайному процесорі більшість процедур при виклику змушені зберігати значення деяких регістрів в стеку для того, щоб користуватися цими регістрами при виконанні процедури. При поверненні з процедури значення регістрів відновлюються зі стека.

У перших архітектурах, що зараховуються до RISC, більшість команд для спрощення декодування мають однакову довжину і схожу структуру, арифметичні операції працюють тільки з регістрами, а робота з пам'яттю йде через окремі команди завантаження (load) і збереження (store). Ці властивості і дозволили краще збалансувати етапи конвеєризації, зробивши конвеєри в RISC значно ефективнішими, дозволивши підвищити тактову частоту і ефективність суперскалярності (розпаралелювання команд між декількома виконавчими блоками).

Нерідко слова «скорочений набір команд» неправильно розуміється, як мінімізація кількості команд в системі команд. Насправді, команд у багатьох RISC–процесорів більше, ніж у CISC–процесорів. Термін «скорочений» описує

той факт, що скорочений обсяг і час роботи, що виконується кожною окремою командою – максимум один такт доступу до пам'яті – тоді як складні команди CISC–процесорів можуть вимагати для свого виконання сотень тактів доступу до пам'яті.

Характерні особливості RISC–процесорів:

– фіксована довжина машинних команд (наприклад, 32 біти) і простий формат команди;

– спеціалізовані команди для операцій з пам'яттю – читання або запису. Операції виду «прочитати–змінити–записати» відсутні. Будь–які операції «змінити» виконуються тільки над вмістом регістрів (т. зв. архітектура load–and–store);

– велика кількість регістрів загального призначення (32 і більше);

– відсутність підтримки операцій виду «змінити» над укороченими типами даних: байт або 16–бітове слово. Так, наприклад, система команд DEC Alpha містила лише операції над 64–бітними словами, і вимагала розробки і подальшого виклику процедур для виконання операцій над байтами, 16– і 32–бітними словами;

– відсутність мікропрограм всередині самого процесора. Те, що в CISC–процесорі виконується мікропрограмами, в RISC–процесорі виконується як звичайний (хоча і поміщений в спеціальне сховище) машинний код, який не відрізняється принципово від коду ядра ОС і додатків.

В даний час багато архітектур процесорів є RISC–подібними, наприклад, ARM, DEC, Alpha, SPARC, AVR, MIPS, POWER і PowerPC. Найбільш широко використовувані в настільних комп'ютерах процесори архітектури x86 раніше були CISC–процесорами, проте нові процесори, починаючи з Intel Pentium Pro, є CISC–процесорами з RISC–ядром. Вони безпосередньо перед виконанням перетворять CISC–інструкції x86–процесорів в більш простий набір внутрішніх інструкцій RISC.

Після того, як процесори архітектури x86 були переведені на суперскалярну RISC–архітектуру, можна сказати, що більшість існуючих нині процесорів засновані на архітектурі RISC.

1.2 Загальний огляд мікроконтролерів сімейства ARM

Архітектура ARM (Advanced RISC Machine) – 32–бітна RISC–архітектура процесорів. Розроблена компанією ARM Limited. Процесори ARM мають низьке енергоспоживання, тому завоювали сегмент масових мобільних продуктів (стільникові телефони, кишенькові комп'ютери) і знаходять широке застосування у вбудованих системах середньої і високої продуктивності (від мережевих маршрутизаторів і точок доступу до телевізорів). Станом на 2014 р., 95% смартфонів у світі, 80% усіх цифрових камер і 35% усіх електронних пристроїв використовують ARM технології.

Історія ARM почалася, коли компанія Acorn Computers після успіху з комп'ютером BBC Micro задумалася над переходом від відносно слабких процесорів MOS Technology 6502, яким не вистачало потужності для підтримки графічного інтерфейсу, до більш продуктивних рішень. Жодна з доступних 16–бітних архітектур не задовольняла їхніх вимог. Тому компанія вирішила розробити нову 32–бітну архітектуру на основі нової на той час архітектури RISC. Проект дістав назву Acorn RISC Machine (1983р.). Ключовим завданням було досягнення низьких затримок при обробці переривань, як у MOS Technology 6502. Архітектура доступу до пам'яті з 6502 дозволила розробникам досягнути хорошої продуктивності без використання дорогого модуля прямого доступу до пам'яті. Процесор вперше запрацював у 1985р. і був названий *ARM1*. Це був повністю функціональний 32–розрядний процесор з 26–розрядним (64 Мбайт) адресним простором. Він мав шістнадцять 32–бітних регістрів, ортогональний набір 32–бітних інструкцій, тобто при виконанні інструкції не було обмеження на використання тільки визначених для неї регістрів. Лічильник команд був обмежений 26–ма бітами, тому що верхні 6 біт 32–розрядних регістрів були зарезервовані в якості статус–міток. Тому програмний код перебував всередині перших 64 Мбайт пам'яті.

Перші серійні процесори під назвою *ARM2* стали доступні в наступному році. У ньому додали інструкції множення, апаратну підтримку співпроцесорів математики з плаваючою комою. ARM2, можливо, є найпростішими із використовуваних 32–бітних мікропроцесорів в світі, робота якого забезпечена

лише 30000 транзисторами (для порівняння, старіша на 6 років модель Motorola 68000 містила близько 70000 транзисторів). Така простота обумовлювалася відсутністю мікропрограми, яка, наприклад, у процесорі 68000 становила від 1/4 до 1/3 площі кристалу, і відсутністю кешу. Це привело до низьких витрат енергії, і той же час ARM був набагато продуктивніший, ніж Intel 80286 – до 4 MIPS (million instructions per second – мільйонів інструкцій за секунду).

Наступний процесор – *ARM3* – уже мав кеш (4 КБ). Тактова частота зросла до 25 МГц, продуктивність склала 13 MIPS.

В кінці 1980–х років компанія Apple почала працювати з Acorn Computers над новими версіями ядра ARM. В результаті співробітництва був створений *ARM6* (1992 р.). Адресну шину розширили до 32 біт, зберігаючи сумісність з попереднім 26–бітним режимом. Введені два нові режими процесора – для обробки помилок звернення до пам'яті та невизначених інструкцій. Додані два нові регістри: регістр поточного стану процесора (CPSR, Current Processor State Register) і регістр збереженого стану процесора (SPSR, Stored Processor State Register). ARM6 працював на частоті 20, 30 і 33 МГц і виконував в середньому 17, 26, і 28 MIPS відповідно. Це був також енергоефективний процесор, з низькою для того часу напругою живлення ядра 3.3В. Ядро ARM, попри всі зміни, залишилося фактично такого ж розміру: кількість транзисторів зросла всього до 35000. Це був початок для мобільних вбудованих систем: першим продуктом, який використовував ARM6, став Apple Newton MessagePad – одна з перших серій кишенькових персональних комп'ютерів.

У *ARM7* (1993р.) збільшили розмір кеш–пам'яті до 8 КБ. Також був представлений розширений набір команд – Thumb. Thumb – це інший 16–бітний набір інструкцій, що дозволяє (теоретично), щоб програми займали половину від їх розміру в пам'яті. Використовуючи всього 8 регістрів, за відсутності підтримки умовного виконання, він працює повільніше, але був відповіддю на інші вбудовані 8–бітні і 16–бітні процесори. У ARM7 удосконалили множення – з’явилися як 32–розрядні, так і 64–бітні інструкції множення і множення/накопичення. Реалізація виявилася настільки успішною, що її

використали в наступних версіях ядер – ARM8, ARM9 і StrongARM. В ARM7 вперше представили апаратне відлагодження (on–chip debugging).

ARM7TDMI є удосконаленням ядра ARM7. TDMI розшифровується, як «Thumb + Debug + Multiplier + ICE», де ICE (In–Circuit Emulator, схемний емулятор) – апаратний пристрій, який підключаються до системи через спеціальний порт, і дає можливість взяти контроль над вбудованою системою з метою відлагодження. З продуктивністю до 130 MIPS, це було одне з найбільш широко використовуваних ядер у вбудованих системах, Apple iPod, Nintendo Game Boy Advance і більшості популярних мобільних телефонів.

ARM8 (1996р.) включав модуль передбачення переходів (branch prediction) і пам'ять з подвійною пропускною здатністю. Тут з'явився п'ятиатапний конвеєр, у той час як в ARM7 тільки три. Завдяки цьому ARM8 вдвічі продуктивніший порівняно з ARM7.

У *ARM9* (1998р.) відбувся перехід з класичної архітектури фон Неймана на модифіковану Гарвардську, з відділенням шини команд від шини даних і кешу. Завдяки цьому вибірка команди і доступ до даних може відбуватися одночасно. ARM9 також використовував п'ятиетапний конвеєри, введений у ARM8.

ARM9TDMI став заміною для надзвичайно популярного ARM7TDMI. Додатки, призначені для ARM7TDMI, були приблизно на 30% швидше на ARM9TDMI.

Для *ARM10* задачею було подвоїти продуктивність ARM9. Для цього ARM працювали над двома основними аспектами: оптимізацією конвеєра і швидкістю виконання команд. Для підвищення швидкості конвеєра, етап декодування розділили на два, де на одному декодується частина інструкції, на другому читаються регістри з залишком декодованої послідовності. В ARM10 з'явилося нове ядро множення – швидкий 16 × 32 апаратний помножувач. Це дозволило виконувати одну 32–бітну операцію множення і накопичення за один такт, що є величезне поліпшення в порівнянні з 3–5 тактами на ARM9.

Архітектура, що використовується в ARM9 і ARM10, впровадила технологію Jazelle DBX (Direct Bytecode eXecution, пряме виконання байт–коду), яка дозволяє виконання Java байт–коду на апаратному рівні. Спрямована

головним чином на ринок мобільних телефонів, Jazelle давала змогу Java ME (Java Mobile Edition) додаткам і іграм працювати швидше шляхом перетворення байт–коду в машинні інструкції ARM. ARM стверджує, що приблизно 95% байт–коду в типових програмах обробляється на апаратному рівні.

У 2002р. ARM випустила ядро *ARM11*. Його архітектура реалізовує технологію SIMD (Single Instruction, Multiple Data – одиночний потік команд, множинний потік даних) – принцип комп'ютерних обчислень, що дозволяє забезпечити паралелізм на рівні даних. Інструкції SIMD розроблені для виконання повторюваних інструкцій на багатьох наборах даних, що широко використовується в аудіо – і відеокодеках, зокрема, для MPEG–4 на ринку мобільних телефонів. Додавання інструкцій SIMD подвоїла швидкість обробки MPEG–4. П’ятиетапний конвеєр замінили на восьмиетапний, підвищивши очікувану частоту до 1 ГГц. ARM11 також підтримує обмежене позачергове виконання команд, дозволяючи конвеєру продовжити виконання, якщо результат попередньої команди не потрібен наступним командам.

ARM11 був величезним успіхом, ARM–процесори стають все потужнішими, але й дорожчими. Тим не менш, деякі клієнти не були зацікавлені в дуже швидких процесорах та складній архітектурі і шукали більш «легкі» рішення, в той же час отримуючи всі переваги від останніх технологічних досліджень. ARM перелаштували свою лінію процесорів для задоволення потреб особливих сфер. Так у 2004р. було повідомлено про нове сімейство *Cortex*.

До сімейства Cortex входить 3 класи процесорів: Cortex–A, Cortex–R і Cortex–M. Cortex–A («A» – application) призначено для пристроїв, що вимагають високої продуктивності (смартфони, планшети), на яких працює багатозадачна операційна система. Cortex–R («R» – real time) розроблено для додатків реального часу, де час реакції повинен бути мінімальний. Cortex–M («M» – microcontroller) призначено для мікроконтролерів і недорогих вбудованих пристроїв.

Процесор Cortex–A призначено для прикладних систем. Це потужні чипи, які дозволяють запустити повноцінну операційну систему, високопродуктивні мультимедійні кодеки і вимогливі до ресурсів додатки. Серія Cortex–A оснащено

MMU (memory management unit, блок управління пам'яттю) – апаратний компонент, що відповідає за управління доступом до пам'яті, який запитується процесором. Додавши декілька зовнішніх компонентів, можна створювати передові платформи. На Cortex–A працюють смартфони, планшети, цифрові телевізори та інформаційно–розважальні системи, і навіть були розроблені ноутбуки на багатоядерних Cortex–A.

Процесор Cortex–M призначено для додатків до мікроконтролерів. Ці додатки не вимагають великої обчислювальної потужності, але потребують багато ліній введення і виведення, дуже малий форм–фактор, детерміновану реакцію на переривання і виключно низьке енергоспоживання. Cortex–M активно використовуються в Bluetooth–пристроях, контролерах сенсорного екрану, пристроях дистанційного керування.

1.3 Сімейство ядер ARM

Нумерація версій ARM–ядер не відповідає нумерації версій ARM–архітектури, тому часто виникає плутанина (таблиця 1.4). Версії архітектури записуються, як ARMv, а версії ядра – як ARM. Крім того, перше число у версії ядра не завжди однакове з першим числом у версії архітектури, до якої це ядро належить.

Наприклад, ядро ARM1 належить до архітектури ARMv1, а ядра ARM2 і ARM3 – до архітектури ARMv2. Хоча різниця між ARM2 і ARM3 більша, ніж між ARM2 і ARM1. Ядер ARM4 і ARM5 не існувало. Ядра ARM6 і ARM7 належать до архітектури ARMv3 і т.д. (таблиця 1.1). Після ARM11 всі ядра називають Cortex.

Таблиця 1.1 – Приналежність ядер ARM до архітектури

Архітектура	Ядро
ARMv1	ARM1
ARMv2	ARM2, ARM3
ARMv3	ARM6, ARM7
ARMv4	StrongARM, ARM7TDMI, ARM8, ARM9TDMI
ARMv5	ARM7EJ, ARM9E, ARM10E, XScale
ARMv6	ARM11
ARMv6–M	Cortex–M0, Cortex–M0+, Cortex–M1
ARMv7–A	Cortex–A5, Cortex–A7, Cortex–A8, Cortex–A9, Cortex–A12, Cortex–A15
ARMv7–R	Cortex–R4, Cortex–R5, Cortex–R7
ARMv7–M	Cortex–M3
ARMv7E–M	Cortex–M4
ARMv8–A	Cortex–A53, Cortex–A57

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Назвіть характерні особливості RISC–процесорів.
2) Які архітектури процесорів є RISC–подібними?
3) Як розуміти слова «скорочений набір команд» у RISC–процесорів?
4) Дайте визначення архітектурі ARM.
5) Де сьогодні широко використовуються ARM–контролери?
6) Коли відбувся перехід з класичної архітектури фон Неймана на модифіковану Гарвардську архітектуру ARM–контролерів?
7) Чим відрізняються Гарвардська архітектура та архітектури фон Неймана?
8) Які класи процесорів входять до сімейства Cortex ?
9) Охарактеризуйте процесор Cortex–A.
10) Охарактеризуйте процесор Cortex–R.
11) Охарактеризуйте процесор Cortex–M.
12) Чим відрізняються версії ARM–ядер від версій ARM– архітектури?
13) Назвіть відмінності між ARM7 та ARM7TDMI?

2 СТРУКТУРА ТИПОВОГО МІКРОКОНТРОЛЕРА З ARM–ЯДРОМ

2.1 Загальні відомості

Мікроконтролерне ядро ARM було розроблене однойменною англійською компанією, яка була організована в 1990 році. Назва ARM походить від "Advanced RISC Machines". Слід зауважити, що компанія спеціалізується суто на розробці мікропроцесорних ядер і периферійних блоків, при цьому не має виробничих потужностей з випуску мікроконтролерів. Компанія ARM постачає свої розробки в електронній формі, на основі якої клієнти конструюють свої власні мікроконтролери. Клієнтами компанії є понад 60 компаній–виробників напівпровідників, серед яких можна виділити таких популярних виробників на ринку напівпровідникових компонентів країн СНД, як Altera, Analog Devices, Atmel, Cirrus Logic, Fujitsu, MagnaChip (Hynix), Intel, Motorola, National Semiconductor, Philips , ST Microelectronics, Texas Instruments і т. ін.

В даний час архітектура ARM займає лідируючі позиції і охоплює значну частину ринку 32–розр. вбудованих RISC–мікропроцесорів. Поширеність даного ядра пояснюється його стандартністю, що надає можливість розробником більш гнучко використовувати, як свої, так і сторонні програмні напрацювання, як при переході на нове процесорний ARM–ядро, так і при міграціях між різними типами ARM–мікроконтролерів.

Деякі компанії використовують розроблені ARM–процесори для спеціальних застосувань, проте більшості вони потрібні для мобільних телефонів, систем управління автомобільними двигунами, лазерних принтерів PostScript та інших пристроїв масового застосування. Для всіх цих пристроїв необхідні такі якості, як висока швидкодія, помірна ціна і низьке енергоспоживання.

Фірмою розроблено цілий ряд 32–розрядних RISC–процесорів з різними можливостями і різною продуктивностю, а її процесор ARM, який розроблено ще в 1994 році, використовується до теперішнього часу.

Сама фірма визначає процесор ARM7 як універсальне ядро 32–розрядного RISC–мікропроцесора з малим енергоспоживанням, яке призначене для використання в різних замовних і спеціальних ІС. Малі розміри RISC–ядра

дозволяють успішно інтегрувати його в великі замовні схеми, які можуть містити RAM, ROM, DSP, додаткову логіку та інші елементи.

Основні характеристики ядра ARM7:

– 32–розрядний RISC–процесор (32–розрядні шини даних і адреси) з продуктивністю 17 MIPS при тактовій частоті 25 МГц (пікова продуктивність 25 MIPS);

– 32–розрядна адресація – лінійний адресний простір в 4 Гб – виключає потребу в сегментованій, розділеній на банки або оверлійній пам'яті;

– тридцять один 32–розрядний регістр загального призначення і шість регістрів стану;

– регістри адрес, запису і конвеєра;

– циклічний пристрій зі зсувом і перемножувач;

– трирівневий конвеєр (вибірка команди, її декодування і виконання);

– робочі режими Big Endian і Little Endian;

– напруга живлення 3,3 і 5 В;

– мале енергоспоживання 0,6 мА / МГц, при виготовленні за CMOS–технологією з топологічними нормами 0,8 мкм;

– повністю статична робота, що дозволяє додатково знижувати споживання за рахунок зменшення тактової частоти, що ідеально для критичних до споживання застосувань;

– швидке реагування на переривання застосувань реального масштабу часу;

– підтримка систем віртуальної пам'яті;

– проста, але потужна система команд.

В роботі, як приклад, розглядаються основні перефирійні модулі мікроконтролерів LPC2378 та STM32, які використовують ядро ARM7 та Cortex.

2.2 Мікроконтролер LPC2378

LPC2378 – один із старших контролерів свого сімейства [1, 4, 7]. Йому притаманні такі основні особливості:

– 32 бітне ядро ARM7TDMI–S;

– тактова частота до 72 МГц;
– 512 КБ Flash + 58 КБ SRAM;
– Ethernet 10/100 MAC + DMA;
– USB 2.0 в режимі Full–speed;
– двоканальний CAN 2.0B;
– контролер DMA;
– I2S, три I2C, три SPI / SSP, чотири UART;
– можлива робота з зовнішніми картами SD / MMC;
– АЦП (8 каналів, 10 біт), ЦАП (1 канал, 10 біт);
– і т. ін.

Більш повний перелік периферії наведено в документі «LPC2378 Preliminary datasheet».

Контролер зазвичай упаковується виробником в 144–вивідний корпус TQFP, зовнішній вигляд якого наведено на рисунку 2.1.

Рисунок 2.1 – Зовнішній вигляд мікроконтролера LPC2378 (вид зверху)

Зовні контролер нічим не примітний, окрім як великою кількістю виводів. Для того щоб визначити функцію того чи іншого виводу необхідно звернутися до відповідної документації. На рисунку 2.2 наведено рекомендоване електричне підключення контролера.

Для того, щоб контролер запустився, необхідно забезпечити його якісним живленням 3,3В і поставити необхідні фільтруючі конденсатори. Для внутрішніх потреб контролера (для живлення ядра) потрібно джерело напругою 1,8В, але контролери цього сімейства мають вбудований перетворювач напруги і, тому, потрібні лише конденсатори (C47…C49).

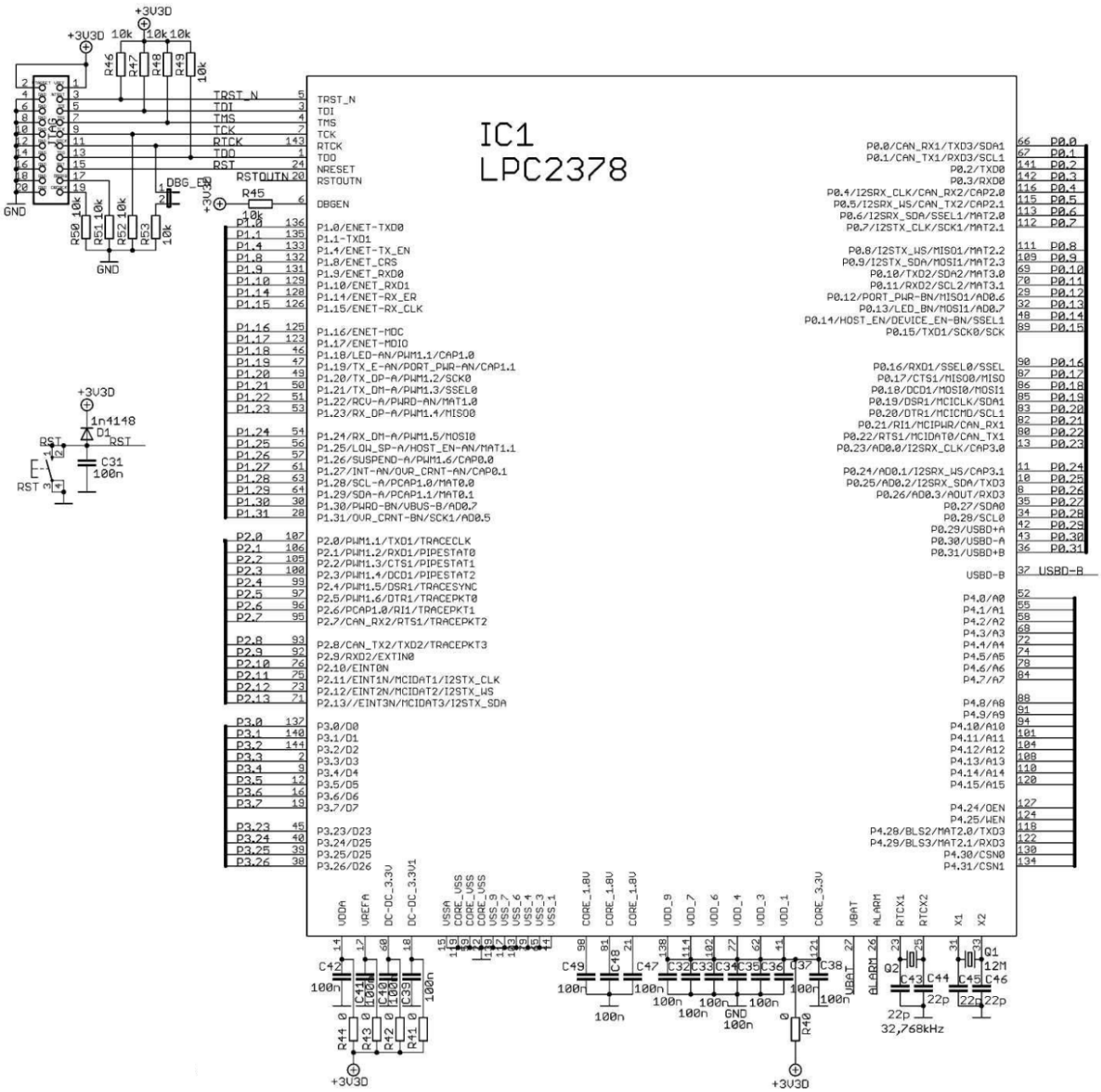


Рисунок 2.2 – Рекомендоване електричне підключення контролера

Для тактування контролера потрібно як мінімум один кварц на входи X1, X2 (основний вхід тактового генератора) або на входи R1TCX1 + R1TCX2 (вхід тактового генератора для годинника реального часу). Кварц можна і не ставити, скориставшись внутрішнім (вбудованим в чіп) генератором, але через його низьку точність, використовувати деяку периферію не вдасться (так, наприклад USB працювати не буде, оскільки йому потрібні тактові сигнали з точними інтервалами часу, які не може забезпечити мініатюрний кварц). Великий 20–контактний роз'єм (JTAG) зліва вгорі (рисунок 2.2) призначено для

налагодження та програмування контролера. При необхідності до нього підключається штекер з шлейфом, що йде від відлагоджувача JTAG. З використанням відповідного програмного забезпечення, наприклад, що входить в IDE Keil for ARM7, в контролер записується програма, яку програміст (користувач) набрав і відкомпілював. Програми створюються з використанням модифікації мови С, що включає розширення для програмування контролерів.

Нижче в таблиці 2.1 наведено порівняння мікроконтролера LPC2378 з іншими мікроконтролерами ряду LPC23xx.

Таблиця 2.1 – Характеристики мікроконтролерів LPC23xx

Table 2. LPC23xx features overview

Part	Local bus SRAM (kB)	Flash (kB)	EMC	USB/ GP SRAM (kB)	USB device	USB host/ OTG	Ethernet	Ethernet GP SRAM (kB)	CAN channels	SD/ MMC	ADC channels	GPIO pins
LPC2361	8	64	no	8	yes	yes	no	16	2	no	6	70
LPC2362	32	128	no	8	yes	yes	yes	16	2	no	6	70
LPC2364	8	128	no	8	yes	no	yes	16	2	no	6	70
LPC2365	32	256	no	8	no	no	yes	16	-	no	6	70
LPC2366	32	256	no	8	yes	no	yes	16	2	no	6	70
LPC2367	32	512	no	8	no	no	yes	16	-	yes	6	70
LPC2368	32	512	no	8	yes	no	yes	16	2	yes	6	70
LPC2377	32	512	Mini	8	no	no	yes	16	-	yes	8	104
LPC2378	32	512	Mini	8	yes	no	yes	16	2	yes	8	104
LPC2387	64	512	no	16	yes	yes	yes	16	2	yes	6	70
LPC2388	64	512	Mini	16	yes	yes	yes	16	2	yes	8	104

З таблиці видно, що контролер відрізняється наявністю USB– інтерфейсу, але не може виступати в якості хосту USB (тобто не може до себе підключати пристрої USB). Також у контролера є Ethernet–контролер, 8 аналогових входів (АЦП), і 104 виводи загального призначення. Кількості швидкої пам'яті (ОЗП) на локальній шині 32 Кб достатньо для виконання більшості завдань автоматики, так само як і постійної пам'яті Flash, об'ємом 512 Кб. При необхідності користувач може підключати зовнішні мікросхеми пам'яті і, таким чином, збільшити об`єм пам'яті.

Для того, щоб більш детально ознайомитися з контролером, розглянемо його структурну схему (рисунок 2.3).

Структура досить складна, тому її умовно розділено червоною смугою на важливі складові контролера: ядро і швидкі пристрої та іншу периферію (нижче червоної смуги). З першого погляду видно, що на контролері присутні кілька шин даних (AHB, APB, Local Bus – на рисунку без назви).

Найголовніший блок – ARM7TDMI–S – це ядро мікроконтролера, до якого відносяться всі основні функції по керуванню пам'яттю, обчисленням, обробці програми і т. ін.

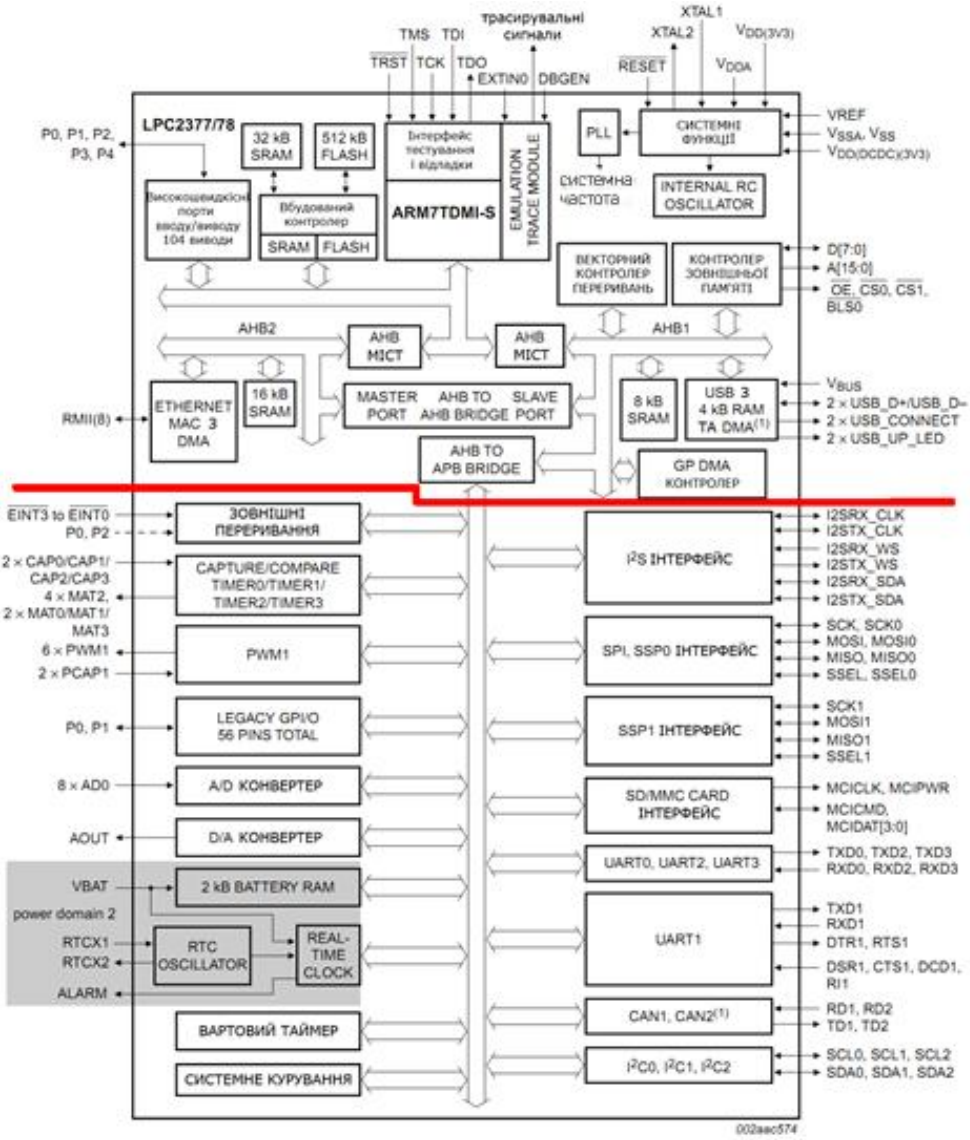


Рисунок 2.3 – Структурна схема мікроконтролера LPC2378

Блок System Functions (блок системних функцій) призначено для контролю роботи контролера, скидання (вхід RESET) та тактування (XTAL1, XTAL2). До блоку приєднано блоки вбудованого низькоточного RC–осцилятора (Internal RC

Oscillator), з якого здійснюється запуск контролера, а також фазового автопідстроювання частоти (PLL = Phase Locked Loop), що відповідає за розподіл частоти і тактування контролера. Не дивлячись на відсутність зв'язку на структурній схемі від ядра до цих блоків, існує можливість налаштовувати їх роботу з програми через регістри (не обов'язково під час запуску контролера).

На рисунку 2.4 наведено ядро та швидкі пристрої з перекладом на українську мову деяких блоків.

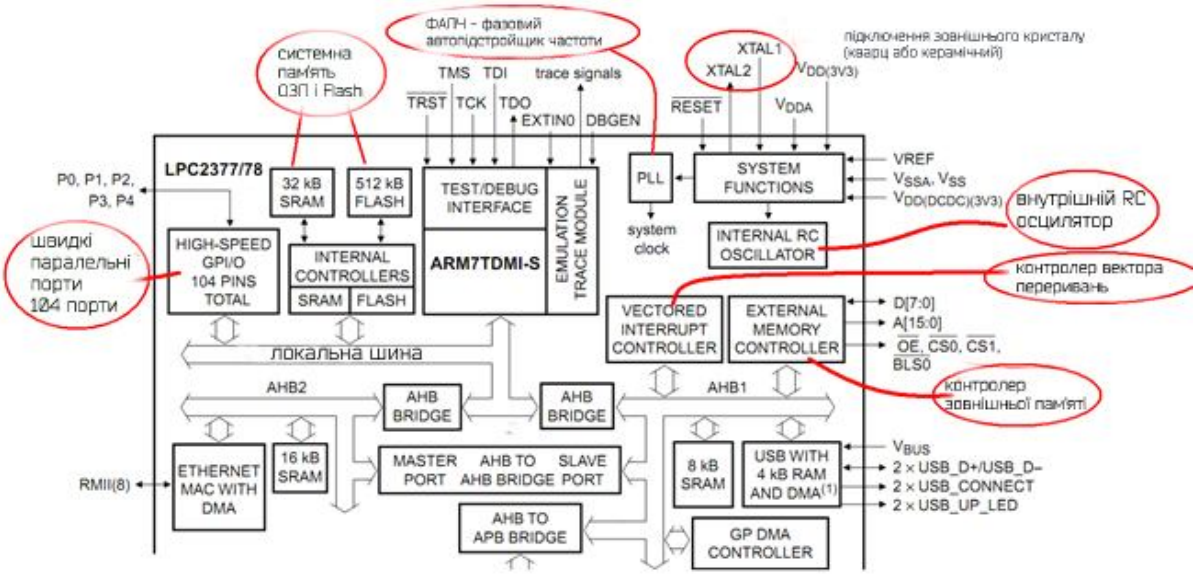


Рисунок 2.4 – Швидка периферія і ядро контролера LPC2378

Зв'язок з периферією проводиться за допомогою наступних спеціальних шин (рисунок 2.5):

– ARM7 local bus (локальна шина). Шина забезпечує швидкий доступ до пам'яті чіпа та швидкий обмін через паралельний порт (GPIO);

– AHB (Advanced High performance Bus – покращена високопродуктивна шина) – зв'язує ядро зі швидкістною периферією і зовнішнюю пам'яттю. На контролері є дві шини: AHB1 і AHB2;

– APB (Advanced Peripheral Bus–покращена периферійна шина) – зв'язує інші периферійні пристрої на чипі;

– Шина AHB1 – зв'язує ядро з:

– VIC (vector interrupt controller) – векторний контролер переривань (32 вектори) та контролер USB;

– GP DMA – контролер прямого доступу до пам'яті загального призначення (дозволяє скоротити кількість переривань при роботі з модулями, з якими проводиться інтенсивний обмін даними через буфери, наприклад Ethernet, USB і т.ін.);

– EMC – контролер зовнішньої пам'яті. Забезпечує доступ до зовнішньої пам'яті (Flash) і різної зовнішньої периферії, яка є проектована на адресний простір пам'яті.

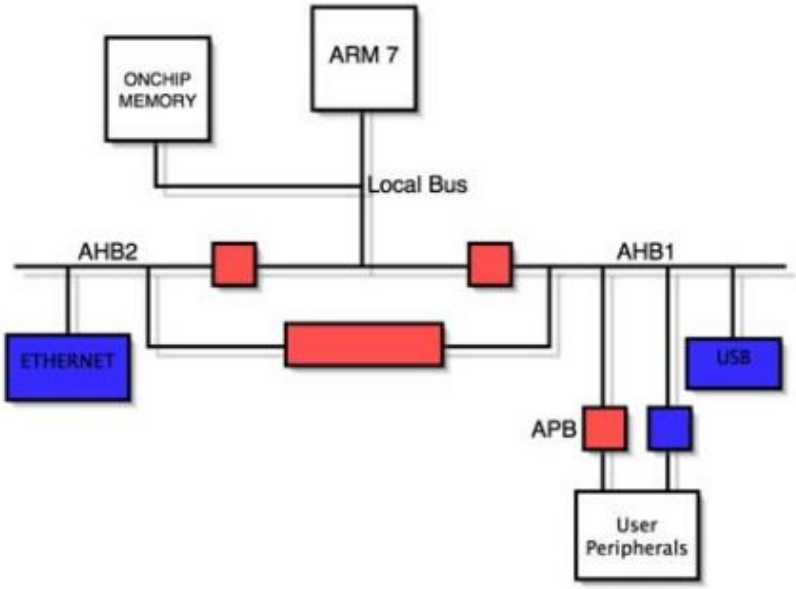


Рисунок 2.5 – Зв'язок шин даних контролера LPC2378

Шина AHB2 – зв'язує тільки ядро і Ethernet модуль (в тому числі і

внутрішню пам'ять Ethernet = 16 КБ SRAM). AHB2 виділена в окрему шину, щоб не заважати роботі іншим пристроям, тому що обсяг даних, який передається через Ethernet може бути значним і повністю завантажити периферійну високопродуктивну шину даних.

Шини AHB незалежні, але можуть бути підключені в режим Master–Slave (рисунок 2.6).

У такому разі:

– AHB2 працює в режимі Master;

– AHB1 – в режимі Slave;

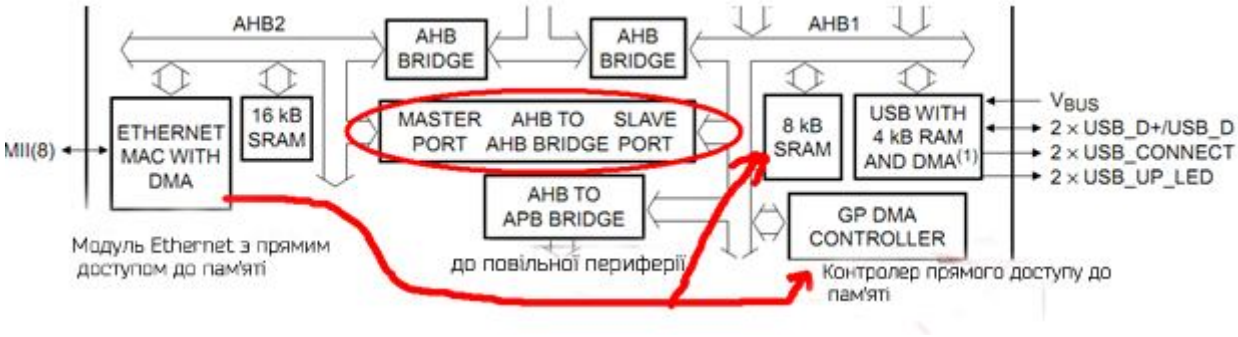


Рисунок 2.6 – З'єднання периферійних шин AHB1 і AHB2 через міст AHB–to–AHB

– AHB2 може запитувати з основної пам'яті контролера чи зовнішньої пам'яті додаткове місце під буфер Ethernet;

– адресація шиною AHB завжди 32–бітна, та прозора для програміста, тобто програміст напряму не працює з шиною;

– усім пристоям на шині AHB виділяється адресний простір нижче адреси 4.0 Гб (0xFFFF FFFF);

– кожен пристрій отримує блок 16 Кб (рисунок 2.7).

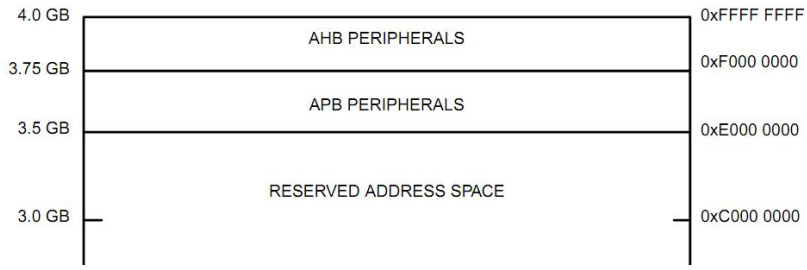


Рисунок 2.7 – Поділ на блоки адресного простору пристроїв шин AHB і APB

Також важливо зазначити наступні моменти, які викликані такою конфігурацією шин даних:

– пам’ять SRAM буфера Ethernet 16 КБ і пам'ять SRAM, яка використовується контролером GP DMA = 8 КБ, також може бути використана для зберігання даних або коду;

– пам'ять 2 Кб годинника реального часу (RTC), можна використовувати тільки для зберігання даних. Оскільки вона належить до

повільної шини, зчитувати з неї код недоцільно, тому що це уповільнило б ядро контролера, який постійно очікує передачі нової порції коду;

– пам'ять годинника реального часу (RTC) можна використовувати в ролі NVRAM (Non–volatile random–access memory – енергонезалежної пам'яті), оскільки вона живиться від батарейки і її вміст зберігається у вимкненому стані.

– шина APB (Advanced Peripheral Bus – покращена периферійна шина) – призначена для розщеплення і з'єднання безлічі більш повільних периферійних пристроїв.

APB – ще більш повільна шина, ніж AHB. Зв'язок з APB здійснюється за допомогою мосту AHB–APB. Пристрої на шині APB також адресуються блоком, який лежить вище 3.5 Гб (рисунок 2.7). Блок займає 2 Мб. Кожен пристрій в блоці резервує за собою 16 Кб адресного простору із загального блоку (послідовно).

На рисунку 2.8 наведено пристрої, які з’єднуються шиною APB.

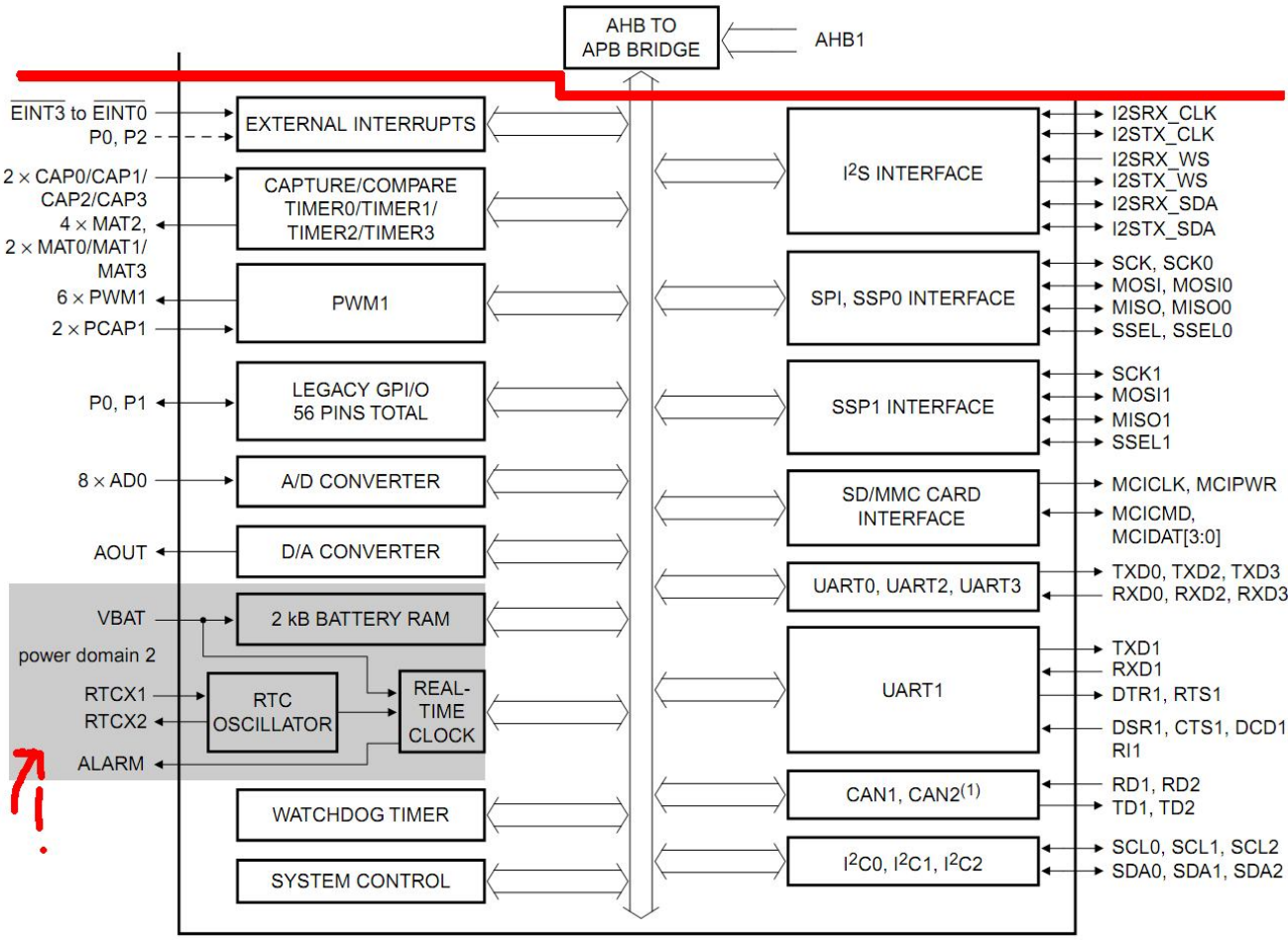


Рисунок 2.8 – Пристрої, які з’єднуються шиною APB

Кожний периферійний пристрій має по одній лінії, що сполучає його і VIC (векторний контролер переривань).

Будь–яка ніжка порту Port0 або Port2, незалежно від функцій, може бути запрограмована на генерацію переривань (сумарно 46 ніжок):

– за наростанням;
– за спаданням;
– за наростанням і спаданням вхідного сигналу.

Ці запити переривань будуть скомбіновані із запитом переривання EINT3.

2.3 Мікроконтролер STM32

Враховуючи велику кількість компаній, що виробляють мікроконтролери архітектури ARM, велику кількість цих контролерів навіть у однієї компанії, коротко зупинимося на контролерах серії STM32 компанії STMicroelectronics [1, 8].

STMicroelectronics – європейська мікроелектронна компанія, одна з найбільших, що займаються розробкою, виготовленням та продажем різних напівпровідникових електронних і мікроелектронних компонентів. Сьогодні штаб–квартира компанії знаходиться в Женеві, в той же час, її холдингова компанія STMicroelectronics NV зареєстрована в Амстердамі, однак компанія історично пов'язана з Італією та Францією, де значно взаємодіє з ринком. Компанія має представництва в США, Китаї та Японії.

Мікроконтролери STM32 використовують ядро Cortex–M3. Сімейство ARM Cortex – нове покоління процесорів, які виконані за стандартною архітектурою і відповідають різним технологічним вимогам. На відміну від інших ЦПП (цифровий процесорний пристрій) ARM, сімейство Cortex є завершеним процесорним ядром, яке об'єднує стандартне ЦПП і системну архітектуру. Сімейство Cortex доступно в трьох основних профілях: профіль A для високопродуктивних застосувань, профіль R для застосувань у реальному часі і профіль M для чутливих до вартості мікроконтролерних застосувань.

Мікроконтролери STM32 виконано на основі профілю Cortex–M3, яке спеціально розроблене для застосувань, де необхідні розвинені системні ресурси і, при цьому, мале енергоспоживання. Вони характеризуються настільки

низькою вартістю, що можуть конкурувати з традиційними 8 і 16–бітними мікроконтролерами. І хоча ЦПП ARM7 і ARM9 були з успіхом інтегровані в стандартні мікроконтролери, в них все ж таки простежується початкова орієнтованість на системи на кристалі (SoC). Це особливо помітно за способами обробки виняткових ситуацій та переривань, тому у різних виробників мікроконтролерів ці способи обробки реалізовано різним чином. Cortex–M3 є стандартизованим мікроконтролерним ядром, яке крім ЦПП, містить всі інші основні елементи мікроконтролера (в т.ч. система переривань, системний таймер SysTick, відлагоджувальна система та карта пам'яті). 4 гігабайтний адресний простір Cortex–M3 розділено на чітко розподілені області коду програми, статичного ОЗП, пристроїв введення–виведення і системних ресурсів.

На відміну від ядра ARM7, Cortex–M3 виконано за Гарвардською архітектурою і, тому, має декілька шин, що дозволяють виконувати операції паралельно. Сімейство Cortex має можливість оперувати з фрагментованими даними (unaligned data), що також відрізняє його від попередніх архітектур ARM. Цим гарантується максимальна ефективність використання внутрішнього статичного ОЗП. Сімейство Cortex також підтримує можливості встановлення і скидання біт в межах двох областей пам'яті розміром 1 Мбайт за методом bit banding. Цей метод надає ефективний доступ до регістрів і прапорців ПВВ, розташованих в області статичного ОЗП, і виключає необхідність інтеграції повнофункціонального бітового процесора.

Основою STM32 є процесор Cortex–M3. Він являє собою стандартизований мікроконтролер, який інтегрує 32–бітний ЦПП, шинну структуру, блок вкладених переривань, відлагоджувальну систему і відповідну організацію пам'яті.

До появи STM32 компанія ST вже мала у своєму асортименті 4 сімейства мікроконтролерів на основі ядер ARM7 і ARM9, однак саме у мікроконтролерів STM32 було досягнуто суттєве поліпшення співвідношення вартості і робочих характеристик. Мікроконтролери STM32, ціна яких за штуку при покупці великих кількостей становить трохи більше одного Євро, кидають серйозний виклик існуючим 8–бітним мікроконтролерам. Мікроконтролери STM32, які

спочатку випускалися в 14 різних варіантах, розділено на дві групи: Performance Line, до якої увійшли мікроконтролери з тактовою частотою ЦПП до 72 МГц, і Access Line (тактова частота до 36 МГц). Обидві групи мікроконтролерів сумісні за розташуванням виводів і програмному забезпеченню. Обсяг їх вбудованої Flash–пам'яті досягає 128 Кбайт, а статичного ОЗП – 20 Кбайт. З моменту появи перших мікроконтролерів STM32 їх асортимент був істотно розширений новими представниками з підвищеними розмірами ОЗП і Flash–пам'яті, а також з більш складними ПВВ.

Сімейство STM32 складається з двох груп. Група Performance Line працює на тактових частотах до 72 МГц й оснащена повним набором ПВВ, а група Access Line працює на частотах до 36 МГц та інтегрує обмежений набір ПВВ.

Сімейство STM32 – це не тільки мікроконтролери на ядрі Cortex–M3. Архітектура Cortex–M включає в себе також ядра Cortex–M0 і Cortex–M4.

Cortex–M0 – це Cortex–M3 з скороченим набором команд, призначено для більш дешевих і менш вимогливих з точки зору продуктивності рішень. Cortex–M0 дозволить замінити 16–бітові мікроконтролери і, в меншій мірі, 8–бітові мікроконтролери. Cortex–M4 – це Cortex–M3, збагачено новими командами для обробки даних і призначено для застосувань, що вимагають більш високої продуктивності, з більш складною обробкою сигналу (операції з плаваючою комою на апаратному рівні). Cortex–M4 можна буде використовувати в нижньому сегменті DSP–додатків.

Програмний код, що працює на ядрі Cortex–M0, буде в повному обсязі працювати і на ядрі Cortex–M3, оскільки для Cortex–M3 діють всі інструкції Cortex–M0. Програмний код, що працює на ядрі Cortex–M3, також буде працювати на Cortex–M4, оскільки для Cortex–M4 залишаються чинними всі інструкції Cortex–M3. Тобто, зробивши виріб на Cortex–M3, можна буде далі зробити його більш дешеві і прості варіанти на Cortex–M0 або більш дорогі і складні вироби на Cortex–M4 з мінімальними витратами на переробку програмного коду. Оскільки Cortex–M3 вже став світовим стандартом, і оскільки Cortex–M0 і Cortex–M4 є натуральними продовженнями Cortex–M3, нікого не здивує, якщо вони також стануть стандартами найближчим часом. Інші

виробники також активно працюють у цьому напрямку (Texas Instruments, Freescale, NXP і т.ін.)

У підсумку можна сказати, що обираючи STM32, розробник обирає досить популярний мікроконтролер на Cortex–M3, з перспективою переходу на інші ядра Cortex–M, але при цьому не закриває двері для продукції інших виробників.

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Ким було розроблено мікроконтролерне ядро ARM?

2) Від чого походить назва ARM?

3) На чому спеціалізується компанія "Advanced RISC Machines"?

4) Які компанії є клієнтами компанії ARM?

5) Які основні характеристики ядра ARM7?

6) Назвіть основні особливості мікроконтролера LPC2378.

7) Опишіть відмінності мікроконтролера LPC2378 від інших мікроконтролерів ряду LPC23xx.

8) Охарактеризуйте детальніше такі особливості мікроконтролера LPC2378 як:

   структурна схема; швидка периферія і ядро контролера; зв'язок з периферією; шина AHB1; шина AHB2; міст AHB–to–AHB.

9) Чим займається компанія STMicroelectronics?

10) На основі якого ядра виконані мікроконтролери STM32?

11) Охарактеризуйте склад та основні характеристики мікроконтролера STM32.

12) За якою архітектурою виконано мікроконтролер STM32?

13) Опишіть сімейство STM32.

3 СИСТЕМНІ ПЕРИФЕРІЙНІ ПРИСТРОЇ

3.1 Загальні відомості

До системної периферії відносяться модулі, що визначають швидкодію і функціональні характеристики окремих пристроїв [1, 4, 7]. 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).

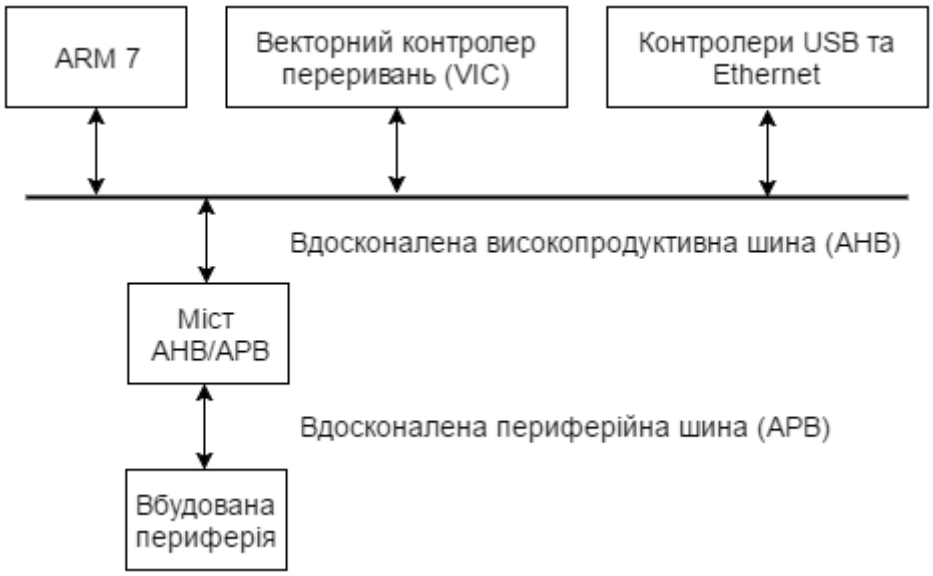


Рисунок 3.1 – Спрощена базова структура ARM7–сумісного мікроконтролера

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).

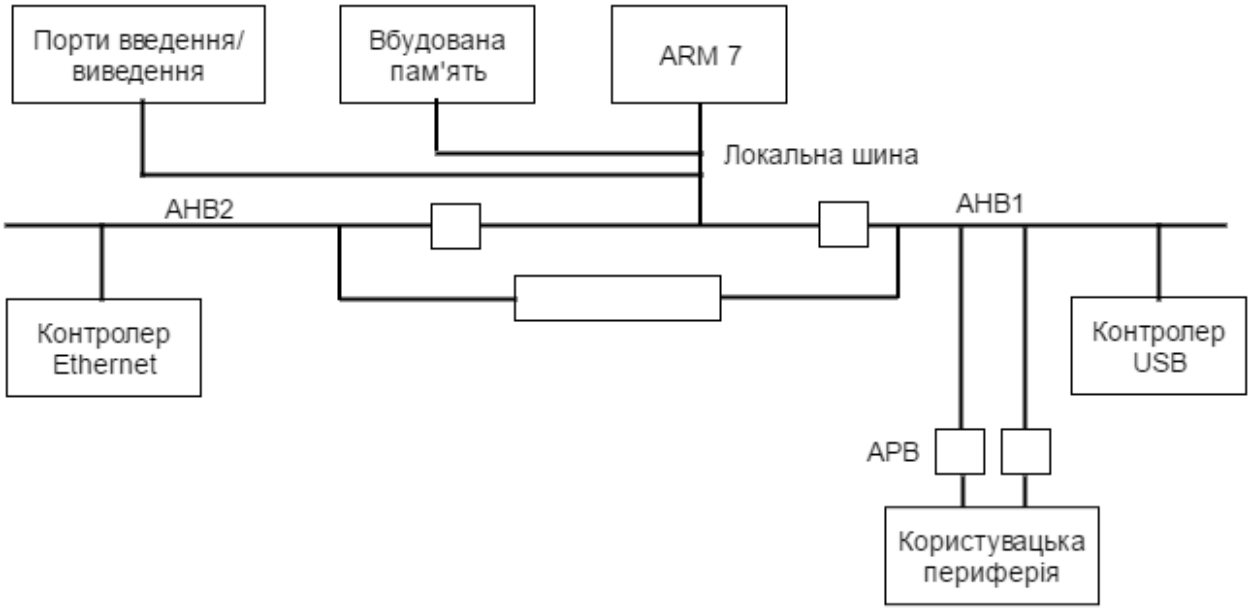


Рисунок 3.2 – Внутрішні шини мікроконтролерів LPC2300

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.

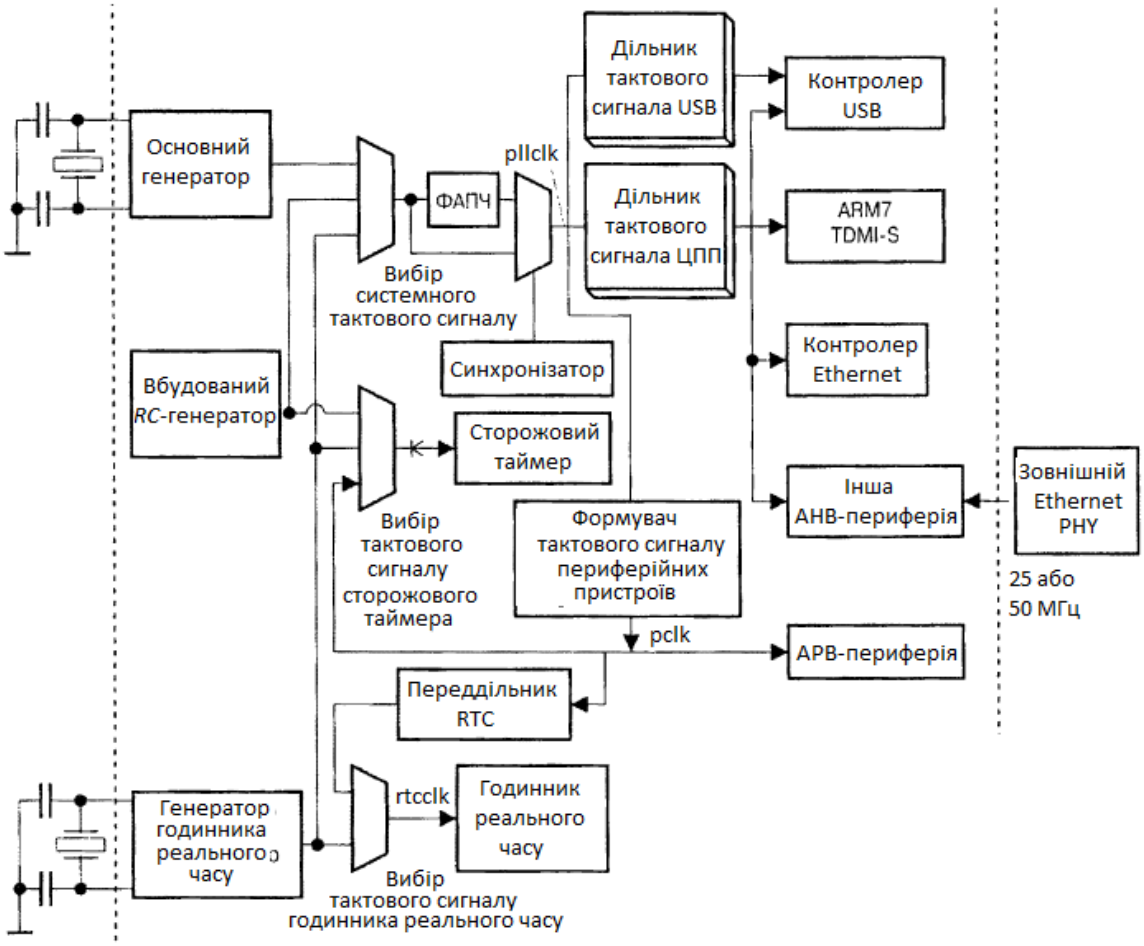


Рисунок 3.3 – Формування системних тактових сигналів

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.

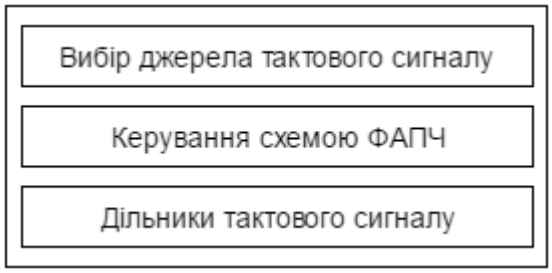


Рисунок 3.5 – Група регістрів для керування системним тактовим сигналом

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.

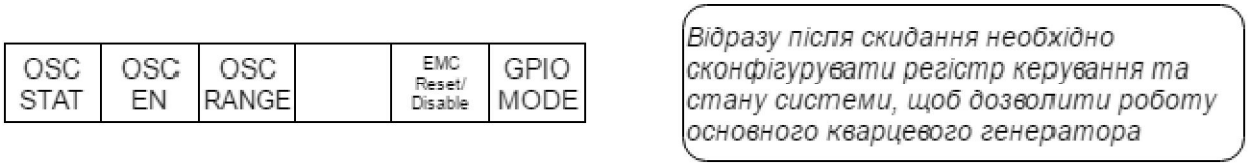


Рисунок 3.6 – Регістр керування та стану системи

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.

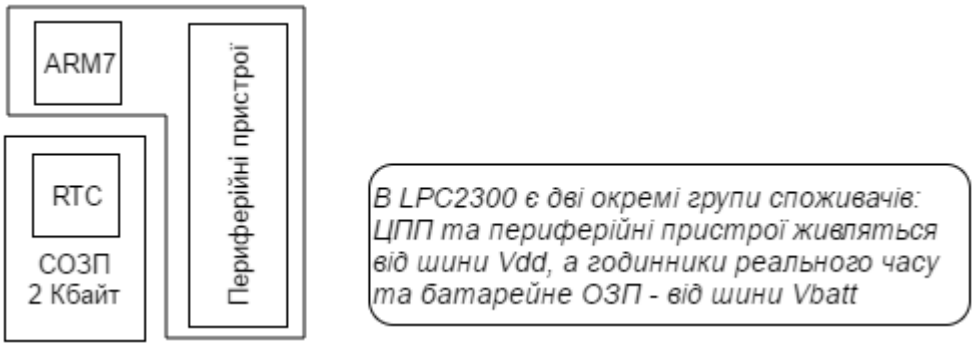


Рисунок 3.7 – Групи споживачів

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.

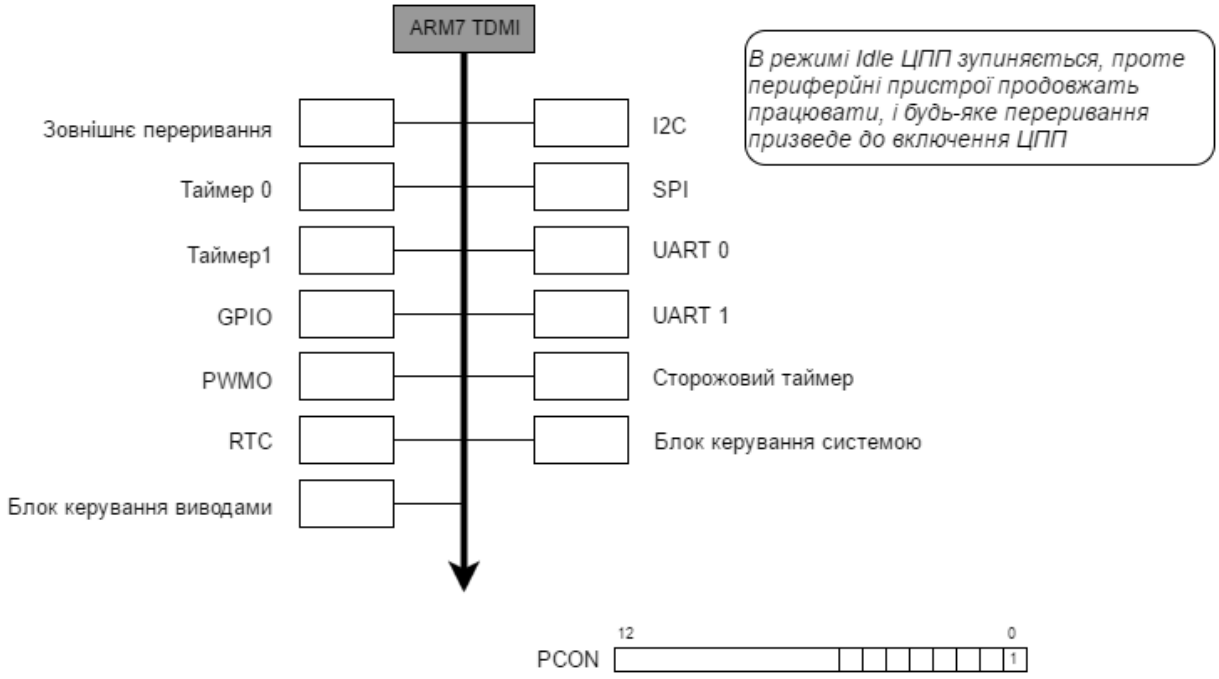


Рисунок 3.8 – Режим Idle

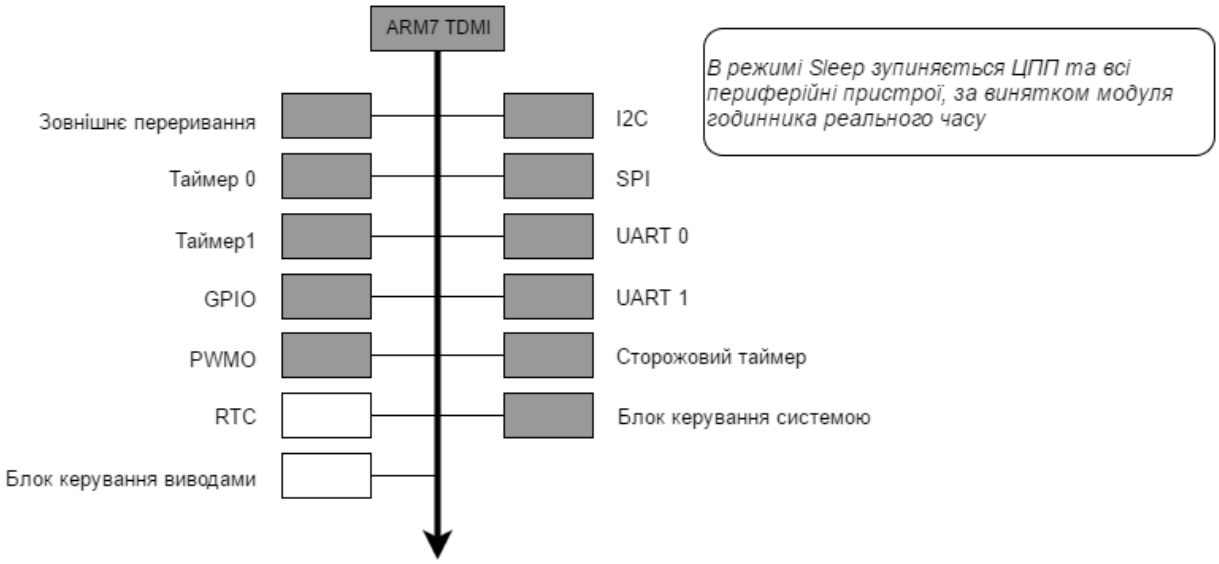


Рисунок 3.9 – Режим Sleep

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.

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Які модулі відносяться до системної периферії?
2) Що представляє собою пам'ять кожного з пристроїв сімейства LPC2300 з точки зору програміста?
3) Які шини використовуються в мікроконтролерах сімейства LPC2300 для взаємодії ядра та периферійних пристоїв?
4) Які системні тактові сигнали використовуються в мікроконтролерах сімейства LPC2300 та як вони формуються?
5) Назвіть режими зниженого енергоспоживання в мікроконтролерах сімейства LPC2300.
6) Дайте характеристику режиму зниженого енергоспоживання Idle.
7) Дайте характеристику режиму зниженого енергоспоживання Power Down.
8) Дайте характеристику режиму зниженого енергоспоживання Deep Power Down.
9) Дайте характеристику режиму зниженого енергоспоживання Sleep.
10) В якому стані знаходяться периферійні пристрої після скидання?
11) Який регістр служить для керування включенням/відключенням тактових сигналів периферійних пристроїв?
12) Які периферійні пристрої за замовчуванням після скидання знаходяться в вимкненому стані?
13) В якому режимі зниженого енергоспоживання досягається найменше енергоспоживання мікроконтролера?
14) Як може відбуватися вихід з режиму Sleep?
15) Які регістри використовуються для керування системним тактовим сигналом?
16) До якої шини підключено ядро ARM7?
17) До якої шини підключено векторний контролер переривань, контролери USB і Ethernet?
18) Як підключаються FLASH–пам'ять і ОЗП до 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.

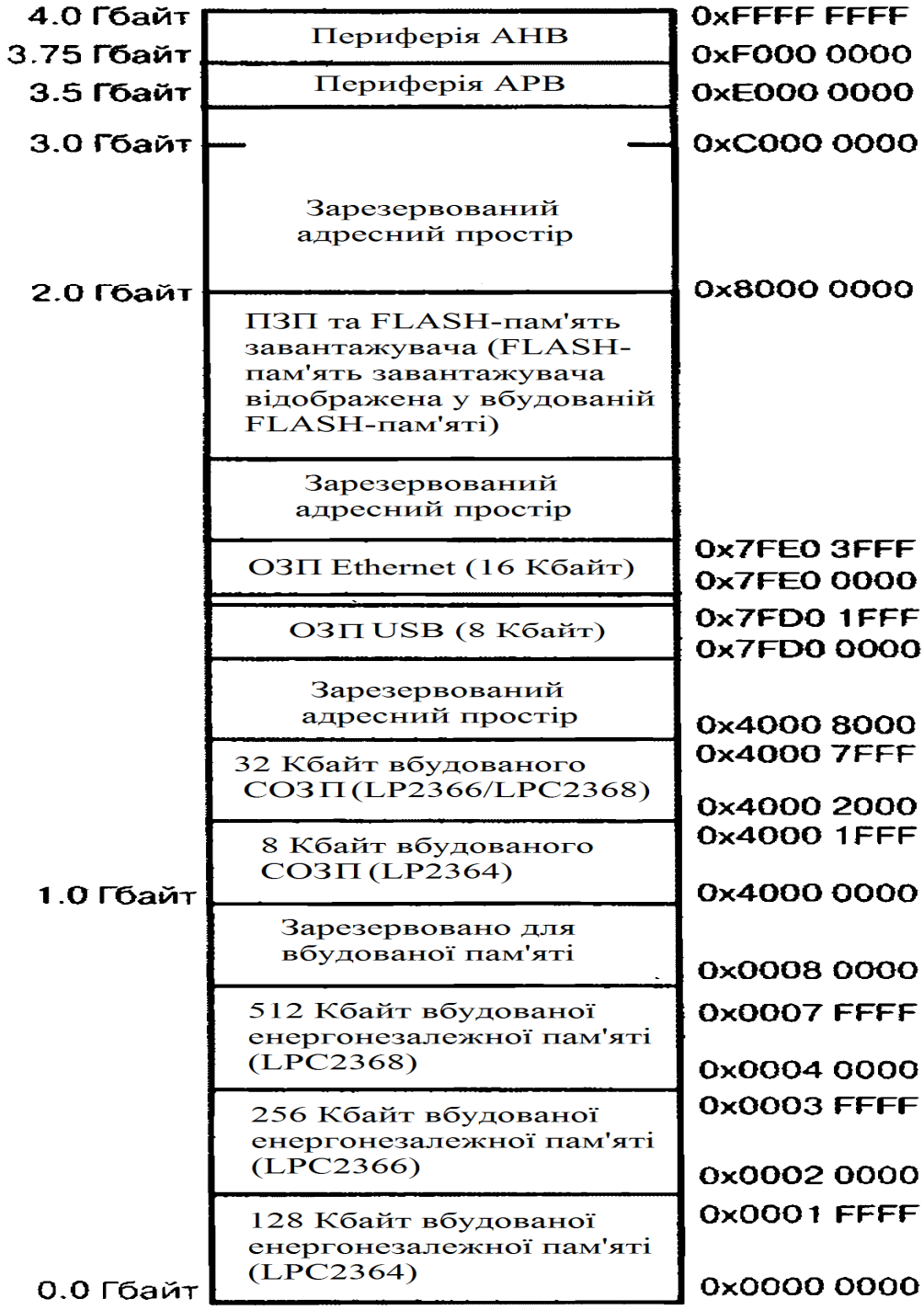


Рисунок 3.10 – Логічна організація пам’яті ARM7

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).

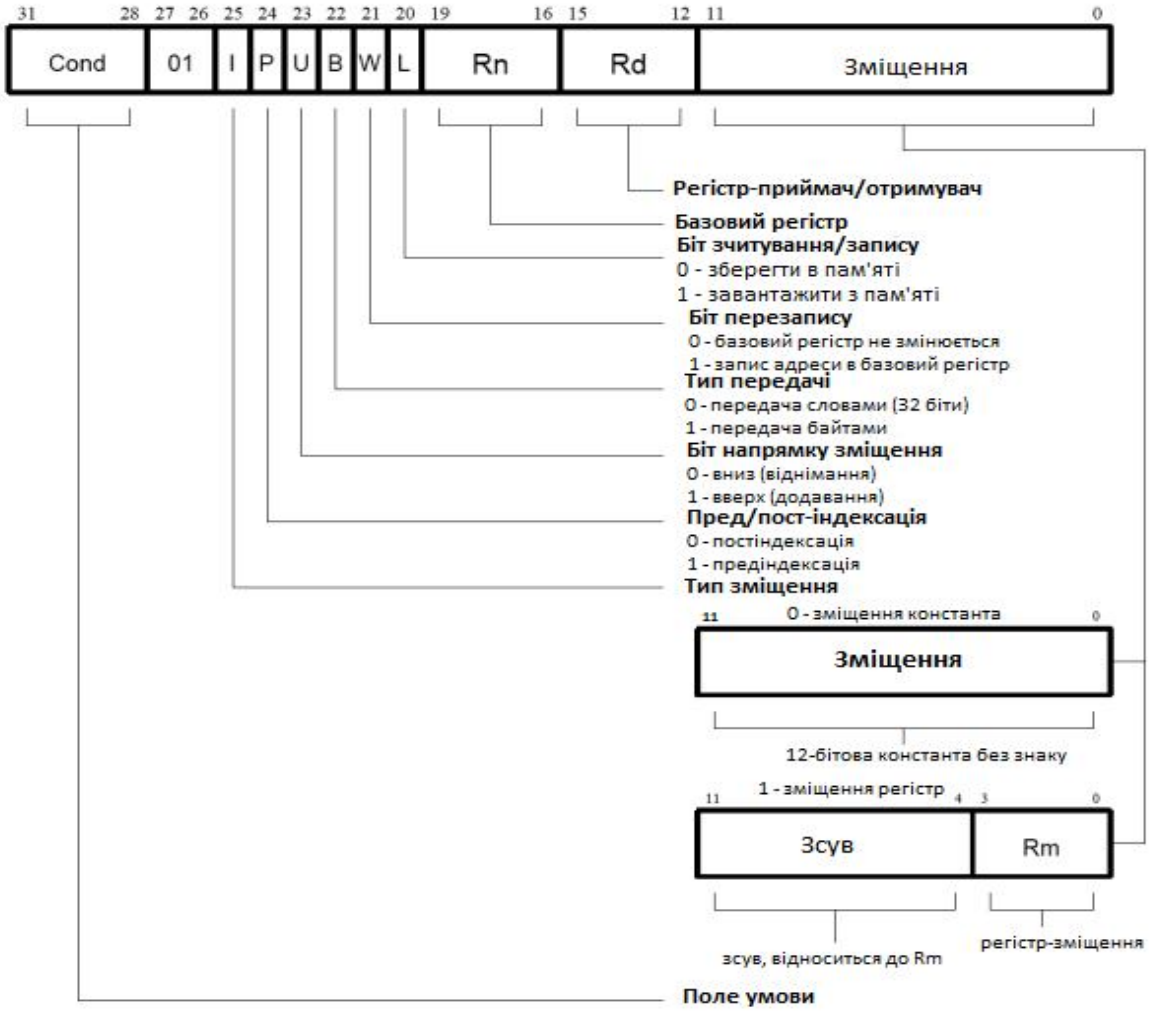


Рисунок 3.11 – Команди з відносною адресацією

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).

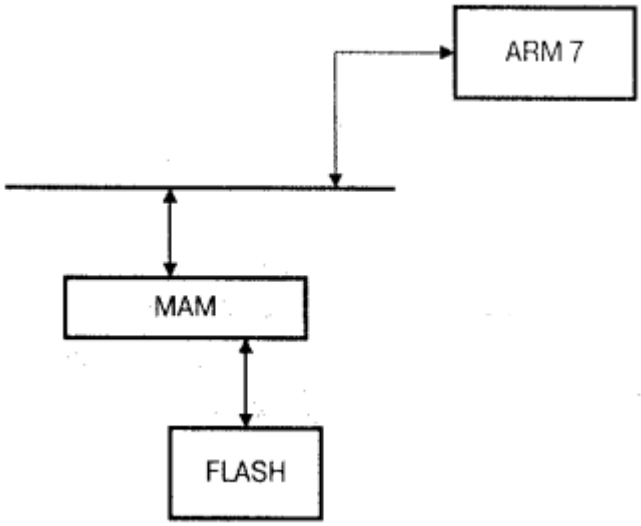


Рисунок 3.12 – Модуль прискорення роботи пам’яті

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.

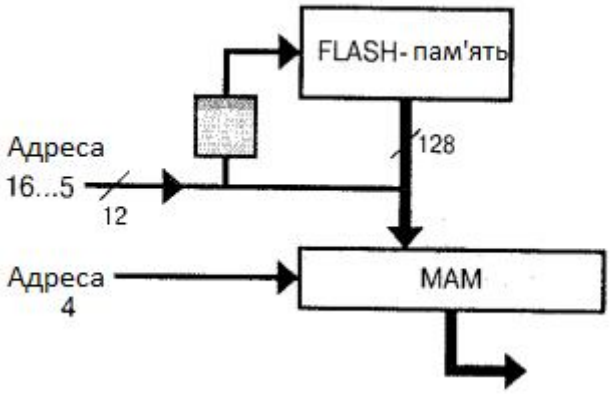


Рисунок 3.13 – Робота модуля MAM

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).

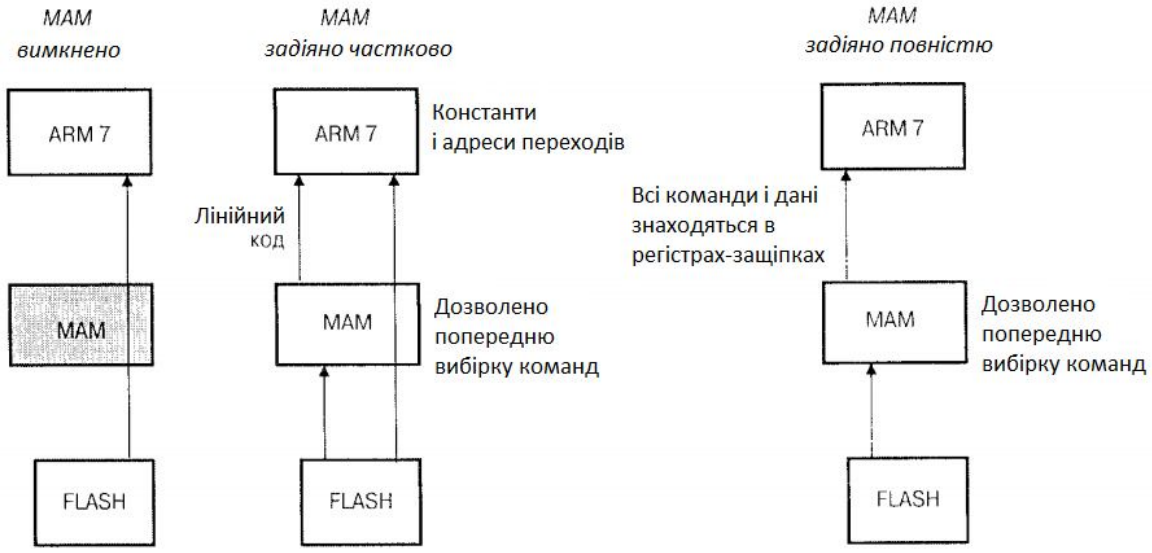


Рисунок 3.14 – Режими роботи модуля MAM

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
}
```

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Яку архітектуру має модуль пам’яті?
2) Дайте порівняння архітектури пам’яті мікроконтролерів Фон–Неймана з Гарвардською архітектурою.
3) Яка ідея була покладена в основу побудови модуля пам’яті для сімейства ARM7?
4) Скількі байт може вмістити пам’ять ARM?
5) Що представляє собою пам'ять кожного з пристроїв сімейства LPC2300 з точки зору програміста?
6) Який порядок зберігання байтів в пам’яті підтримується в сімействі ARM7?
7) Яку логічну організацію має пам’ять в сімействі ARM7?
8) З якої адреси розташовується вбудована FLASH–пам'ять?
9) З якої адреси розташовується користувальницький ОЗП?
10) В якому діапазоні розміщаються програма завантажувача FLASH–пам'яті, а також програма керування монітором реального часу?
11) На які типи поділяються ділянки пам’яті, що виділяються для застосування користувачем?
12) Які команди можуть використовуватися для доступу до пам’яті?
13) Дайте характеристику відносній регістровій адресації зі зміщенням.
14) Для чого призначено модуль прискорення роботи пам'яті (Memory Accelerator Module – MAM) ?
15) До якої шини підключається модуль MAM?
16) Опишіть призначення та роботу модуля MAM.
17) Якими регістрами визначається конфігурація модуля MAM?
18) Опишіть режими роботи модуля MAM.
19) В якому стані знаходиться модуль MAM після скидання?
20) Як можна оцінити ефективність роботи модуля MAM?

3.7 Модуль ФАПЧ

3.7.1 Принцип роботи схеми ФАПЧ

Фазове автопідлаштовування частоти (ФАПЧ) широко використовується в радіотехнічних пристроях і системах зв'язку. На основі генераторів, які керуються напругою (ГКН), і схем ФАПЧ будують синтезатори частоти, які дозволяють отримати сітку стабільних частот. Одним із застосувань синтезаторів є використання їх в якості опорних генераторів в перетворювачах радіочастотних сигналів.

Схема ФАПЧ (англійський еквівалент Phase–Locked Loop – PLL) являє собою систему з контуром зворотного зв'язку, який містить ГКН, фазовий детектор (ФД), фільтр нижніх частот (ФНЧ) і підсилювач (рисунок 3.16)

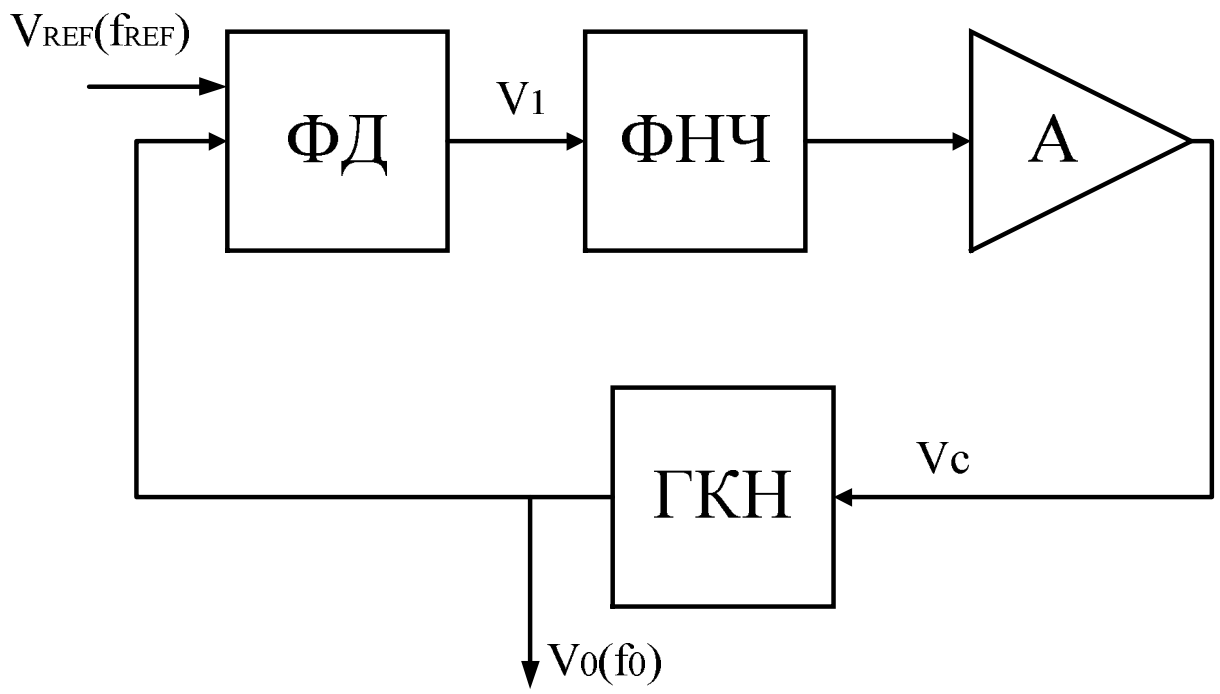


Рисунок 3.16 – Структурна схема ФАПЧ

ФД (або детектор помилки) порівнює фази сигналу опорного генератора V_{ref} і вихідного сигналу ГКН V_0 і виробляє напругу V_1, величина якої залежить від різниці фаз цих сигналів. Вихідна напруга ФД пропускається через ФНЧ для придушення високочастотних складових і їх гармонік, які присутні в опорному сигналі і на виході ГКН. Напруга з виходу ФНЧ підсилюється і використовується як керуюча для ГКН (V_c).

В основі роботи системи ФАПЧ лежить фазова синхронізація сигналів підлаштовуваного і опорного генераторів. Однак, перш ніж виникне фазова синхронізація, коло ФАПЧ треба синхронізувати за частотою. Після того, як відбудеться захоплення частоти, на виході фазового детектора з'явиться напруга, яка пропорційна різниці фаз між сигналом на виході ГКН і зовнішнім еталонним сигналом V_{ref}.

Щоб система ФАПЧ працювала належним чином, необхідно забезпечити повний замкнутий тракт петлі зворотного зв'язку, як показано на рисунку 3.16. При відсутності зовнішнього опорного сигналу або коли петля зворотного зв'язку розімкнута, ГКН працює на частоті попереднього налаштування f_0, яку називают власною частотою або частотою власних коливань. Власна частота – це вихідна частота ГКН в системі ФАПЧ з розімкненою петлею зворотного зв'язку. Власна частота ГКН визначається зовнішніми компонентами. Як вже говорилося, перш ніж система ФАПЧ увійде в режим стеження, повинно відбутися захоплення частоти. Коли на вхід системи ФАПЧ надходить зовнішній вхідний сигнал з частотою F_{ref}, компаратор порівнює частоту зовнішнього сигналу з частотою вихідного сигналу ГКН f_0. На виході фазового детектора формується напруга помилки, неузгодженості Δf, яка пропорціональна різниці фаз двох сигналів на вході. Напруга сигналу помилки через фільтр нижніх частот і підсилювач впливає на керуючий елемент ГКН. Якщо власна частота ГКН f_0 досить близька до частоти зовнішнього опорного сигналу f_{ref}, то під дією зворотного зв'язку в схемі ФАПЧ ГКН синхронізується, тобто захоплює зовнішній вхідний сигнал. Тому вихідна частота ГКН – це сума або різниця його власної частоти і різниці між зовнішньою опорною частотою і власною частотою ГКН.

По суті, система ФАПЧ має три робочих стани:

- автономний стан власних, вільних коливань;
- режим захоплення;
- режим стеження.

У автономному стані зовнішній сигнал опорної частоти відсутній або петля зворотного зв'язку розімкнена. При цьому ГКН генерує сигнал на своїй

власній частоті, яка визначається зовнішніми компонентами. У стані захоплення, система ФАПЧ повинна мати зовнішній опорний сигнал і замкнену петлю зворотного зв'язку. Зі стану захоплення система ФАПЧ з часом переходить в режим стеження частоти. У режимі спостереження частота ГКН відстежує частоту зовнішнього вхідного сигналу, тобто дорівнює їй. Коли система ФАПЧ знаходиться в стані утримання, частота ГКН слідує за змінами частоти зовнішнього опорного сигналу.

Фільтр нижніх частот пригнічує вхідні частоти f_{ref} і f_0, а також їх сумарну частоту $f_0 + f_{ref}$. Таким чином, на виході ФНЧ присутній єдиний сигнал – відносно низька різницева частота, яку називають частотою биття. Сигнал биття підсилюється, а потім надходить на вхід ГКН, змінюючи вихідну частоту ГКН пропорційно величині і полярності своєї напруги. Оскільки вихідна частота ГКН змінюється, амплітуда і частота сигналу биття також пропорційно змінюється.

Після певного числа циклів підстроювання вихідна частота ГКН стає рівною зовнішній опорній частоті і, як кажуть, відбувається захоплення частоти системою ФАПЧ. Як тільки захоплення частоти відбулося, частота биття на виході ФНЧ стає рівною 0 Гц (напруга постійного струму), а його величина і полярність будуть пропорційні різниці фаз між зовнішнім опорним сигналом і вихідним сигналом ГКН. Напруга постійного струму забезпечує вхідне зміщення для ГКН, підтримуючи режим стеження частоти зовнішнього сигналу.

Схема ФАПЧ використовується, наприклад, в якості основного елемента при побудові синтезаторів частоти, які виробляють сітку високостабільних частот, які кратні частоті опорного генератора. Для забезпечення високої стабільності використовують опорний генератор з кварцовим резонатором. Структуру такого синтезатора приведено на рисунку 3.17.

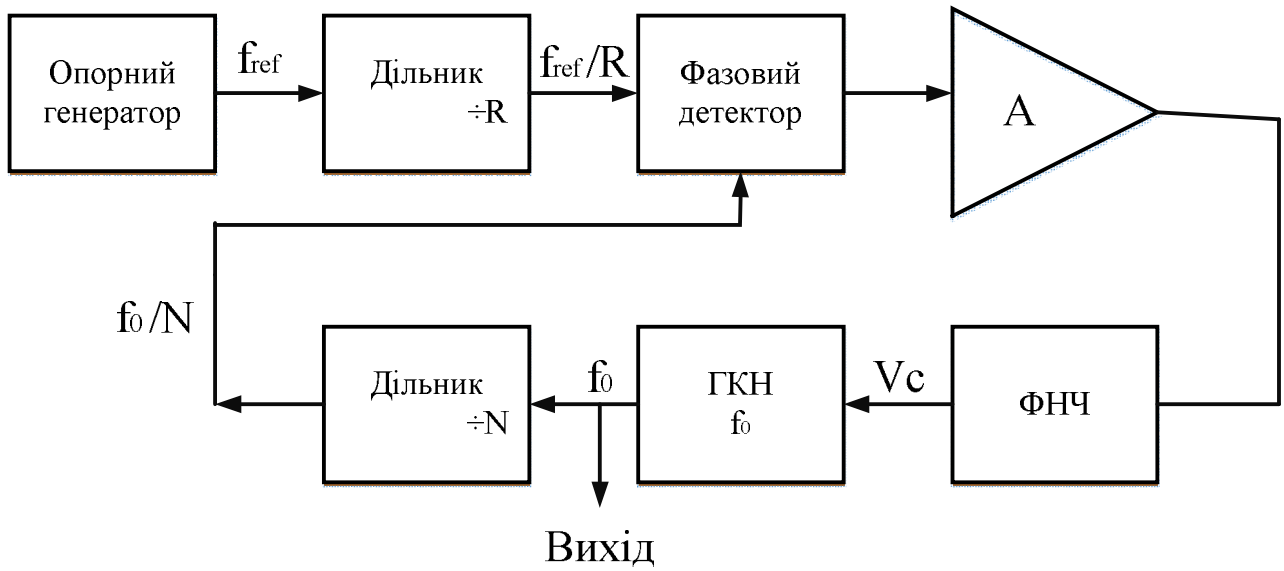


Рисунок 3.17 – Структурна схема синтезатора частоти з ФАПЧ

Частота опорного генератора f_{ref} ділиться на ціле число R за допомогою вхідного подільника. На його виході виробляється сигнал з частотою f_{ref} / R. Частота ГКН також ділиться на ціле число N за допомогою подільника в петлі ФАПЧ, приймаючи значення f_0/N. Коли схема ФАПЧ працює в режимі захоплення опорного сигналу, має місце рівність:

$$\frac{f_{ref}}{R} = \frac{f_0}{N}\,. \qquad (3.1)$$

Звідки:

$$f_0 = \frac{N}{R} f_{ref}\,. \qquad (3.2)$$

Для отримання різних коефіцієнтів ділення застосовують програмовані подільники. Тим самим забезпечується широкий спектр вихідних частот синтезатора, які кратні частоті опорного сигналу. Діапазон вихідних частот синтезатора визначається смугою захоплення ФАПЧ, а абсолютні значення меж цього діапазону залежать від власної частоти ГКН, що до якої розташовується смуга захоплення. Величина кроку зміни частоти синтезатора Δf (роздільна здатність синтезатора) в межах діапазону його перебудови залежить від величини коефіцієнта ділення подільника N і від часу захоплення ФАПЧ. При зменшенні Δf потрібно зменшення часу захоплення (часу встановлення), тобто зменшення часу переходу від одного стійкого стану з частотою f_{01} до іншого стійкого стану з частотою $f_{02} = f_{01} \pm \Delta f$.

3.7.2 Модуль ФАПЧ мікроконтролерів сім'ї LPC23XX

3.7.2.1 Загальні відомості

Модуль ФАПЧ мікроконтролерів сім'ї LPC23XX приймає на вхід тактовий сигнал з частотою в діапазоні від 32 кГц до 25 МГц [1, 4, 7]. Вхідна частота підвищується до високих частот, а потім ділиться, щоб подати потрібну тактову частоту на процесор і блок USB.

Вхід ФАПЧ, в діапазоні від 32 кГц до 25 МГц, спочатку може бути розділений на величину "N", яка знаходиться в діапазоні від 1 до 256. Це вхідне ділення забезпечує велику кількість можливостей для надання широкого діапазону вихідних частот при одній частоті вхідного сигналу.

Окрім вхідного подільника в схемі ФАПЧ стоїть мультиплікатор, який може ділити вихідну частоту схеми на величину "M" (діапазон від 1 до 32768). Результуюча частота повинна бути в діапазоні від 275 МГц до 550 МГц.

Є додаткові подільники на виході ФАПЧ, які понижують частоту до тієї, що необхідна для процесора, USB та іншої периферії. Структурну схему модуля ФАПЧ показано на рисунку 3.18.

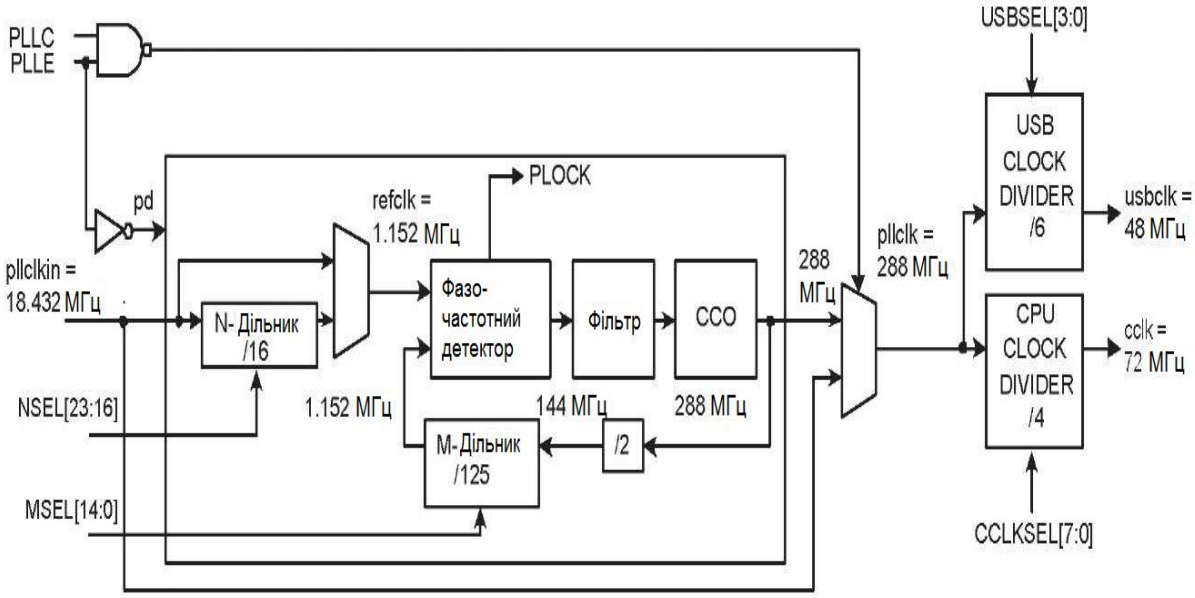


Рисунок 3.18 – Структурна схема модуля ФАПЧ

Роботу модуля ФАПЧ описано у підрозділі 3.7.1. Як приклад, в схемі на рисунку 3.18 N=16, M=125, вхідна частота дорівнює 18,432 МГц, вихідна частота – 288 МГц, коефіцієнт ділення цієї частоти для отримання частоти 48 МГц для

модуля USB дорівнює 6, а коефіцієнт ділення цієї частоти для отримання частоти 72 МГц для CPU дорівнює 4.

Активація ФАПЧ здійснюється через регістр PLLCON. Значення мультиплікатора ФАПЧ і коефіцієнта ділення дільника контролюються регістром PLLCFG. Ці два регістри захищені, щоб запобігти випадковій зміні параметрів ФАПЧ або деактивації ФАПЧ.

Оскільки всі операції чіпу, включаючи сторожовий таймер, можуть залежати від ФАПЧ, якщо він так налаштований (наприклад, коли він надає тактову частота для чіпа), то випадкова зміна налаштування ФАПЧ може призвести до несправності мікроконтролера.

Захист здійснюється подібно до сторожового таймера. Для доступу до регістрів треба згенерувати особливу послідовність, детальна інформація про яку наводиться нижче при описі регістра PLLFEED.

За замовчанням, після скидання мікроконтролера, а також в режимі Power Down система ФАПЧ відключена. Вона може бути запущена лише програмою користувача. Програма повинна сконфігурувати і активізувати ФАПЧ, дочекатися замикання петлі ФАПЧ, а потім комутувати вихід ФАПЧ в якості джерела тактової частоти мікроконтролера.

3.7.2.2 Регістри керування ФАПЧ

ФАПЧ керується регістрами, які описано у таблиці 3.1.

3.7.2.2.1 Керуючий регістр PLLCON

Регістр PLLCON, адреса: 0xE01FC080 (таблиця 3.2) містить керуючі біти, які використовуються для підключення до мікроконтролера системи ФАПЧ і дозволу її роботи.

Таблиця 3.1 – Керуючі регістри ФАПЧ

Назва	Опис	Доступ	Reset	Адреса
PLLCON	**Регістр керування ФАПЧ.** Служить для тимчасового зберігання інформації, необхідної для підключення системи ФАПЧ до мікроконтролера і дозволу її роботи. Значення, які записано в цей регістр, не задіються, поки не виконано коректну послідовність подачі (введення даних) ФАПЧ.	R/W	0	0xE01F C080
PLLCFG	**Регістр конфігурації ФАПЧ.** Служить для тимчасового зберігання інформації, необхідної для того, щоб модифікувати налаштування конфігурації ФАПЧ. Значення, які записано в цей регістр, не задіються, поки не виконано коректну послідовність подачі (введення даних) ФАПЧ.	R/W	0	0xE01F C084
PLLSTAT	**Регістр стану ФАПЧ.** Зчитування цього регістра повертає інформацію, яку записано в регістри керування і конфігурації ФАПЧ. Регістр PLLSTAT не буде достовірно відображати стан ФАПЧ у випадку, якщо запис в регістри PLLCON або PLLCFG була проведена, але коректна послідовність подачі (введення даних) ФАПЧ не була сгенерована.	RO	0	0xE01F C088

Продовження таблиці 3.1

PLLFEED	**Регістр подачі ФАПЧ.** За допомогою цього регістра дозволяється завантаження даних керування і конфігурації ФАПЧ з регістрів PLLCON і PLLCFG в тіньові регістри, які фактично забезпечують керування і конфігурацію ФАПЧ.	WO	NA	0xE01F C08C

Таблиця 3.2 – Опис бітів регістра керування PLLCON

Біт	**Символ**	**Опис**	**Reset**
0	PLLE	**Біт включення ФАПЧ.** Коли цей біт встановлено і виконано коректну послідовність подачі (введення даних), то ФАПЧ активується і петля ФАПЧ замикається на необхідній частоті.	0
1	PLLC	**Біт підключення ФАПЧ до мікроконтролера.** Коли біти PLLC і PLLE встановлено і виконано коректну послідовність подачі (введення даних), то ФАПЧ підключається до мікроконтролера і стає для нього джерелом синхросигналу.	0
31:2	–	Програмне забезпечення користувача не повинно проводити запис в зарезервовані біти. Читання зарезервованого біта повертає невизначене значення.	NA

При підключенні ФАПЧ до мікроконтролера, його центральний процесор і всі вбудовані периферійні модулі тактуються вихідними сигналами ФАПЧ. Зміни вмісту регістра PLLCON не вступають в силу, поки не буде

виконано коректну послідовність подачі (введення даних) ФАПЧ (див. опис регістра подачі ФАПЧ: PLLFEED – 0xE01FC08C).

Система ФАПЧ повинна бути налаштована і включена із замиканням петлі зворотного зв'язку до того, як ФАПЧ стане використовуватися в якості джерела синхронізації мікроконтролера. Спеціальна внутрішня схема координує операції перемикання синхронізації на ФАПЧ і назад, щоб гарантувати їх проведення без збоїв. Слід зауважити, що апаратні засоби не забезпечують замикання петлі ФАПЧ до її підключення до мікроконтролера, а також автоматичне відключення ФАПЧ від мікроконтролера при розмиканні її петлі в ході роботи. Розмикання петлі ФАПЧ може бути викликано нестабільністю частоти задаючого генератора. В цьому випадку відключення ФАПЧ від мікроконтролера не виправить ситуацію.

3.7.2.2.2 Регістр конфігурації PLLCFG

Регістр PLLCFG, адреса: 0xE01FC084 (таблиця 3.3) містить значення множника вхідного перетворювача частоти і коефіцієнта ділення програмованого дільника ФАПЧ. Зміни регістра PLLCFG не вступають в силу, поки не буде виконано коректну послідовність подачі (введення даних) ФАПЧ (див. опис регістра подачі ФАПЧ (PLLFEED – 0xE01FC08C)). Методика обчислення частоти ФАПЧ і значень множника і коефіцієнта ділення наведені в підрозділах 3.7.1 та 3.7.2.5.

3.7.2.2.3 Регістр стану ФАПЧ

Регістр PLLSTAT, адреса: 0xE01F C088 (таблиця 3.5) доступний тільки для зчитування і повертає фактичні параметри та стан ФАПЧ, які є дійсними під час цього читання. Поточне прочитане значення регістра PLLSTAT може не узгоджуватися зі значеннями, що містяться в регістрах PLLCON і PLLCFG, тому що зміни вмісту цих регістрів не вступають в силу, поки не буде подано коректну послідовність подачі (введення даних) ФАПЧ (див. опис регістра подачі ФАПЧ).

Таблиця 3.3– Опис бітів регістра конфігурації PLLCFG

Біт	Символ	Опис	Reset
14:0	MSEL	**Значення множника частоти схеми ФАПЧ.** Відповідає величині «M–1» у формулах обчислення частоти ФАПЧ. Деякі значення M при частоті осцилятора 37,68 кГц вказано у таблиці 3.4.	0
15	–	Програмне забезпечення користувача не повинно проводити запис в зарезервовані біти. Читання зарезервованого біта повертає невизначене значення.	NA
23:16	NSEL	**Значення коефіцієнта ділення попереднього дільника частоти схеми ФАПЧ.** Відповідає величині «N» у формулах обчислення частоти ФАПЧ. Діапазон значень N від 1 до 32. Більш докладно про вибір правильного значення NSEL див. підрозділ 3.7.2.5 «Обчислення частоти ФАПЧ»	0
31:24	–	Програмне забезпечення користувача не повинно проводити запис в зарезервовані біти. Читання зарезервованого біта повертає невизначене значення.	NA

Таблиця 3.4 – Значення мультиплікатора для осцилятора 37,68 кГц

Мультиплікатор (M)	**Дільник (N)**	**f_{cco}**
4272	1	279.9698
4395	1	288.0307
4578	1	300.0238
4725	1	309.6576
4807	1	315.0316
5127	1	336.0031
5188	1	340.0008
5400	1	353.8944
5493	1	359.9892
5859	1	383.9754
6042	1	395.9685
6075	1	398.1312
6104	1	400.0317
6409	1	420.0202
6592	1	432.0133
6750	1	442.3680
6836	1	448.0041
6866	1	449.9702
6958	1	455.9995
7050	1	462.0288
7324	1	479.9857
7425	1	486.6048
7690	1	503.9718
7813	1	512.0328
7935	1	520.0282
8057	1	528.0236
8100	1	530.8416

Продовження таблиці 3.4

Мультиплікатор (M)	**Дільник (N)**	**f_{cco}**
8545	2	280.0026
8789	2	287.9980
9155	2	299.9910
9613	2	314.9988
10254	2	336.0031
10376	2	340.0008
10986	2	359.9892
11719	2	384.0082
12085	2	396.0013
12207	2	399.9990
12817	2	419.9875
12817	3	279.9916
13184	2	432.0133
13184	3	288.0089
13672	2	448.0041
13733	2	450.0029
13733	3	300.0020
13916	2	455.9995
14099	2	461.9960
14420	3	315.0097
14648	2	479.9857
15381	*2*	504.0046
15381	3	336.0031
15564	3	340.0008
15625	2	512.0000
15869	2	519.9954
16113	2	527.9908
16479	3	359.9892

Продовження таблиці 3.4

Мультиплікатор (M)	**Дільник (N)**	**f_{cco}**
17578	3	383.9973
18127	3	395.9904
18311	3	400.0099
19226	3	419.9984
19775	3	431.9915
20508	3	448.0041
20599	3	449.9920
20874	3	455.9995
21149	3	462.0070
21973	3	480.0075
23071	3	503.9937
23438	3	512.0109
23804	3	520.0063
24170	3	528.0017

Таблиця 3.5 – Опис бітів регістра стану PLLSTAT

Біт	**Символ**	**Опис**	**Reset value**
14:0	MSEL	Зчитування цих біт повертає останнє значення мультиплікатора, яке записано в ФАПЧ. Це значення, яке використовується ФАПЧ на момент читання, і на 1 менше ніж у фактичного множника.	0

Продовження таблиці 3.5

Біт	Символ	Опис	Reset value
15	–	Програмне забезпечення користувача не повинно проводити запис в зарезервовані біти. Читання зарезервованого біта повертає невизначене значення .	NA
23:16	NSEL	Зчитування цього біта повертає останнє значення попереднього дільника, яке записано в ФАПЧ. Це значення, яке використовується ФАПЧ на момент читання, і на 1 менше ніж у фактичного дільника .	
24	PLLE	Зчитування цього біта повертає останнє значення біта включення, яке записано в ФАПЧ. ВИСОКИЙ рівень (тобто біт встановлений в логічну 1) означає, що система ФАПЧ на момент читання активована. НИЗЬКИЙ рівень (скидання в 0) означає, що система ФАПЧ вимкнена. Цей біт автоматично скидається, коли мікроконтролер переходить в режим Power Down .	0
25	PLLC	Зчитування цього біта повертає останнє значення біта підключення ФАПЧ до мікроконтролера, яке записано в ФАПЧ. Якщо біти PLLC і PLLE встановлені, ФАПЧ є джерелом синхронізації для мікроконтролера. Якщо один з бітів PLLC або PLLE скинутий, ФАПЧ не бере участь в синхронізації мікроконтролера і синхроімпульси надходять в обхід системи ФАПЧ безпосередньо в мікроконтролер. Цей біт автоматично скидається, коли пристрій переходить в режим Power Down .	0

Продовження таблиці 3.5

Біт	Символ	Опис	Reset value
26	PLOCK	Значення цього біта відображає поточний стан петлі ФАПЧ. Коли біт скинутий, петля ФАПЧ не замкнута. Коли біт встановлений в 1, петля ФАПЧ замкнута на необхідній частоті .	0
31:27	–	Програмне забезпечення користувача не повинно проводити запис в зарезервовані біти. Читання зарезервованого біта повертає невизначене значення.	NA

Значення біта PLOCK відображає поточний стан петлі ФАПЧ. Коли біт скинутий, петля ФАПЧ не замкнута. Коли біт встановлений в 1, петля ФАПЧ замкнута на необхідній частоті.

Значення PLOCK не може бути стійким, коли опорна частоти ФАПЧ F_{REF}, яка дорівнює вхідній частоті ФАПЧ, яку поділено на дільник попереднього значення, менше, ніж 100 кГц, або більше, ніж 20 МГц. У цих випадках схема ФАПЧ може вважатися стабільною після того, як пройде час запуску. Цей час дорівнює $500/F_{REF}$ секунд, коли F_{REF} перевищує 400 кГц і $200/F_{REF}$ секунд, коли F_{REF} менше 400 кГц.

PLOCK пов'язано з контролером переривань. Це дозволяє програмному забезпеченню включати ФАПЧ і продовжувати виконувати інші функції без необхідності чекати замикання ФАПЧ. Коли відбувається переривання, ФАПЧ може бути під'єднано, а переривання відключено.

3.7.2.2.4 Регістр введення даних

Модифіковані програмою значення регістрів PLLCON і PLLCFG набирають чинності лише після того, як буде виконано коректну послідовність введення даних («подачі») в ФАПЧ. Ця послідовність, яку потрібно записати в регістр PLLFEED, адреса: 0xE01FC08C, виглядає таким чином:

1. Записати значення 0xAA.

2. Записати значення 0x55.

Ці дві процедури запису повинні слідувати одна за одною і займати послідовні цикли шини APB. Остання вимога значить, що на час проведення операції введення даних в ФАПЧ всі переривання повинні бути заборонені. Якщо значення, які записуються в регістр подачі, некоректні або хоча б одна з вищезгаданих умов не виконується, ніякі зміни регістрів PLLCON і PLLCFG в силу не вступлять. Опис бітів регістра подачі PLLFEED представлено в таблиці 3.6.

Таблиця 3.6 – Опис бітів регістра подачі PLLFEED

Біт	**Символ**	**Опис**	**Reset**
7:0	PLLFEED	Послідовність подачі: 0xAA, 0x55)	NA

3.7.2.3 Режими роботи ФАПЧ

Можливі комбінації бітів PLLE і PLLC, що задають режими функціонування ФАПЧ, наведено в таблиці 3.7.

Таблиця 3.7 – Комбінації бітів PLLC та PLLE

PLLE	PLLC	Опис
0	0	Систему ФАПЧ вимкнено і відключено від мікроконтролера. Синхронізація мікроконтролера здійснюється безпосередньо від входу синхронізації.
0	1	Систему ФАПЧ активовано, але не підключено до мікроконтролера. ФАПЧ може бути підключено до мікроконтролера після того, як буде, встановлено біт PLOCK.

Продовження таблиці 3.7

PLLE	PLLC	Опис
1	0	Випадок, ідентичний комбінації 00. Ця комбінація запобігає можливості підключення ФАПЧ до мікроконтролера, якщо систему ФАПЧ відключена.
1	1	Систему ФАПЧ активовано і підключено до мікроконтролера в якості основного джерела синхронізації.

3.7.2.4. ФАПЧ та режим Power–down

Перехід в режим Power Down автоматично вимикає схему ФАПЧ і відключає її від мікроконтролера. «Пробудження» з режиму Power Down не веде за собою автоматичного відновлення параметрів налаштування ФАПЧ. Тому це відновлення має бути зроблено для користувача програмою. Як правило, пишеться спеціальна підпрограма, в якій щоб активізувати ФАПЧ, потрібно дочекатися замикання її петлі, біт PLOCK регістра PLLSTART дорівнює 1, а потім підключити ФАПЧ до мікроконтролера.

3.7.2.5 Обчислення частоти ФАПЧ

Формули отримання частоти схеми ФАПЧ використовують параметри, які описано в таблиці 3.8.

Вихідна частота ФАПЧ (коли ФАПЧ активовано і підключено до мікроконтролера) визначається за формулою:

$$F_{CCO} = (2 \times M \times F_{IN}) / N .$$

Вхідні сигнали і параметри ФАПЧ повинні задовольняти наступним умовам:

F_{IN} – від 32 кГц до 50 МГц ,

F_{CCO} – від 275 Мгц до 550 МГц .

ФАПЧ може бути розраховано відносно інших параметрів:

$$M = (F_{CCO} \times N) / (2 \times F_{IN});$$

$$N = (2 \times M \times F_{IN}) / F_{CCO};$$

$$F_{IN} = (F_{CCO} \times N) / (2 \times M).$$

Таблиця 3.8 – Параметри для обчислення частоти ФАПЧ

Параметр	**Опис**
F_{IN}	Частота PLLCLKIN (вхідна частота)
F_{CCO}	Вихідна частота ФАПЧ (вихід Current Controlled Oscillator)
N	Коефіцієнт попереднього дільника, задається бітом NSEL регістра PLLCFG: N=NSEL + 1. N – ціле число від 1 до 32.
M	Множник, задається бітом MSEL регістра PLLCFG: M=MSEL + 1. Не всі можливі значення підтримуються.
F_{REF}	Опорна частота ФАПЧ: $F_{REF}=F_{IN}$, яку поділено на N.

3.7.2.5.1 Дозволені значення M

На високих частотах генератора в діапазоні мегагерц, допускаються значення M від 6 до 512. Це підтримує весь корисний діапазон основного генератора і генератора IRC (внутрішній резонатор).

Для більш низьких частот, особливо коли для синхронізації ФАПЧ використовується модуль RTC, для підтримки потрібної частоти, наприклад для модулів CAN/USB, можна використати набір з 65 додаткових значень M (таблиця 3.9).

3.7.2.5.2 Процедура налаштування параметрів системи ФАПЧ

Розрахунок параметрів модуля ФАПЧ може бути виконано наступним чином:

1. Частота тактових сигналів ЦПП та USB визначається виразом:

$$Cclk = Fcco/Cclksel,$$

$$USBclk = Fcco/USBsel.$$

Тактовий сигнал для кожного з периферійних пристроїв на шині APB формується з сигналу Cclk індивідуальним програмованим подільником Cclksel, коефіцієнт ділення якого може дорівнювати: 1,2 чи 4.

Таблиця 3.9 – 65 додаткових значень мультиплікатора для використання з низькою частотою вхідного тактового сигналу

4272	4395	4578	4725	4807
5127	5188	5400	5493	5859
6042	6075	6104	6409	6592
6750	6836	6866	6958	7050
7324	7425	7690	7813	7935
8057	8100	8545	8789	9155
9613	10254	10376	10986	11719
12085	12207	12817	13184	13672
13733	13916	14099	14420	14648
15381	15564	15625	15869	16113
16479	17578	18127	18311	19226
19775	20508	20599	20874	21149
21973	23071	23438	23804	24170

Для роботи контролера USB потрібен сигнал частотою 48 МГц. Відповідно, щоб можна було використовувати цей модуль, величина Fcco повина бути кратна 48 МГц і знаходиться в діапазоні 275…550 МГц. Якщо поділити її за допомогою подільника USBSEL на відповідну величину, отримаємо потрібне значення. З сигналу Fcco за допомогою подільника CPUSEL можна, наприклад, сформувати тактовий сигнал ЦПП з частотою 60 МГц. Дане значення вибрано з міркувань забезпечення сумісності з мікроконтролерами попереднього сімейства LPC2100, максимальна тактова частота котрих була

обмежена 60 МГц. Ця ж частота може використовуватися для тактування контролера CAN. Якщо Fcco рівна 480 МГц, то при використання зовнішнього резонатора з частотою 12 МГц і N=1, величина коефіцієнта M повинна дорівнювати 20. Відповідно, величина USBsel повинна дорівнювати 10 (USBclk=48 МГц), а величина CLKSEL–8 (Cclk=60 М Гц).

2. Вибір бажаної тактової частоти процесора: Cclk може бути заснований на вимогах додатка до продуктивності процесора, а також має враховувати необхідність підтримки заданого набору стандартних швидкостей універсального асинхронного прийомопередавача (UART), якщо він використовується в додатку і т. ін. При виборі Cclk слід мати на увазі, що периферійні пристрої можуть тактуватися сигналом з частотою, меншою тактової частоти процесора. Для отримання бажаної частоти Cclk обирають значення $F_{CCO,}$ яке націло ділиться, маючи на увазі вимогу підтримки USB, і те, що менші значення F_{CCO} призводять до меншого розсіяння потужності.

3. При виборі значення для вхідної частоти ФАПЧ: F_{IN} треба врахувати, що це може бути сигнал, який отримано від основного кварцового генератора, генератора RTC, або RC–генератора. Для підтримки USB, треба використовувати основний кварцовий генератор.

4. При розрахунку значень M і N для отримання достатньо точних значень частоти Fcco треба врахувати, що потрібне значення M–1 записується у поле MSEL регістра PLLCFG, а потрібне значення N–1 – у поле NSEL регістра PLLCFG.

Загалом, краще використовувати менші значення для N, для зменшення значення мультиплікатора M, яке має бути вибрано для отримання потрібного значення Fcco. У зв'язку з труднощами у пошуку кращих значень, в деяких випадках рекомендується використовувати електронну таблицю, щоб відразу показати багато можливостей, з яких може бути вибрана найкраща.

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Який англійський еквівалент має схема ФАПЧ?
2) Опишить склад та роботу схеми ФАПЧ.
3) Які робочі стани має система ФАПЧ?
4) Опишіть структурну схему синтезатора частоти з ФАПЧ.
5) Опишіть структурну схему модуля ФАПЧ мікроконтролерів сім'ї LPC23XX.
6) Якими регістрами керується схема ФАПЧ?
7) Наведіть опис бітів регістра керування PLLCON.
8) Наведіть опис бітів регістра конфігурації PLLCFG.
9) Наведіть опис бітів регістра стану PLLSTAT.
10) Наведіть опис бітів регістра введення «подачі» PLLFEED.
11) Назвіть режими функціонування ФАПЧ.
12) Як працює схема ФАПЧ в режимі Power–down?
13) Як обчислюється вихідна частота ФАПЧ?
14) Яке значення повинна мати частота для контролера USB та як вона формується?
15) Яке значення повинна мати частота тактового сигналу ЦПП та як вона формується?
16) Яке значення F_{CCO} обирають для отримання бажаної частоти Cclk?
17) Який генератор треба використовувати для підтримки USB?
18) Що треба врахувати при розрахунку значень M і N для отримання достатньо точних значень частоти Fcco?
19) Як працює схема ФАПЧ в режимі Power–down?
20) Для чого служить регістр подачі ФАПЧ?
21) Для чого служить регістр стану ФАПЧ?
22) Для чого служить регістр конфігурації PLLCFG?
23) Якими повинні бути значення множника M?
24) Через який регістр здійснюється активація ФАПЧ?

3.8 Система переривань

3.8.1 Загальні відомості

Поняття переривання (від англ. Interrupt) з'явилося в мікропроцесорній техніці давно і з плином часу міцно в ній закріпилося. Перериванням в мікроконтролерах відведена особлива роль. Вони дозволяють мікроконтролеру швидко реагувати на певні події (від англ. Event), підвищуючи тим самим його оперативність. Переривання виникає в результаті певної очікуваної події і тимчасово припиняє виконання основного коду програми, переходячи до виконання підпрограми, яка знаходиться в обробнику переривання. Після завершення цієї підпрограми триває виконання основного коду програми.

Існує поняття зовнішніх і внутрішніх переривань. Зовнішнє переривання ініціюється зовнішньою подією, наприклад натисненням кнопки користувачем або замиканням контактів датчика. Внутрішнє переривання обумовлено деякою подією, пов'язаною, наприклад, з прийомом або передачею даних через інтерфейс, завершенням рахунку таймера і тому подібним.

Розрізняють апаратне і програмне переривання. Апаратне переривання виникає в результаті впливу зовнішніх сигналів або апаратного встановлення регістрових прапорців подій. Програмне переривання формується за допомогою коду програми шляхом програмного встановлення прапорців подій або безпосереднього виклику підпрограми обробки переривання.

Нижче ми познайомимося з двома джерелами виняткових ситуацій: переривання загального призначення (IRQ) і швидке переривання (FIQ) мікроконтролерів ARM7 [1, 4, 7]. Ці винятки використовуються для підтримки різних джерел переривань, зовнішніх по відношенню до ЦПП ARM7. У нашому випадку такими джерелами є користувальницькі периферійні пристрої. Для вивчення структури переривань сімейства LPC2300 нам буде потрібно просте джерело переривань.

В якості такого джерела можна скористатися виводами зовнішніх переривань. Цей периферійний модуль найпростіше конфігурувати. Крім того, до входу EINT1 може бути підключена кнопка, за допомогою якої ми зможемо ініціювати переривання і спостерігати реакцію у налагоджувачі.

3.8.2 Блок керування виводами

Всі контакти введення/виведення в мікроконтролерах сімейства LPC2300 мають додаткові функції і підключені до вбудованих периферійних модулів через мультиплексори, які називають блоками вибору функції виводів (pinselectblock). За допомогою цього блоку (рисунок 3.19) можна задіяти вивід в якості лінії порту введення/виведення загального призначення (GPIO) або ж «повісити» на нього одну з трьох функцій.

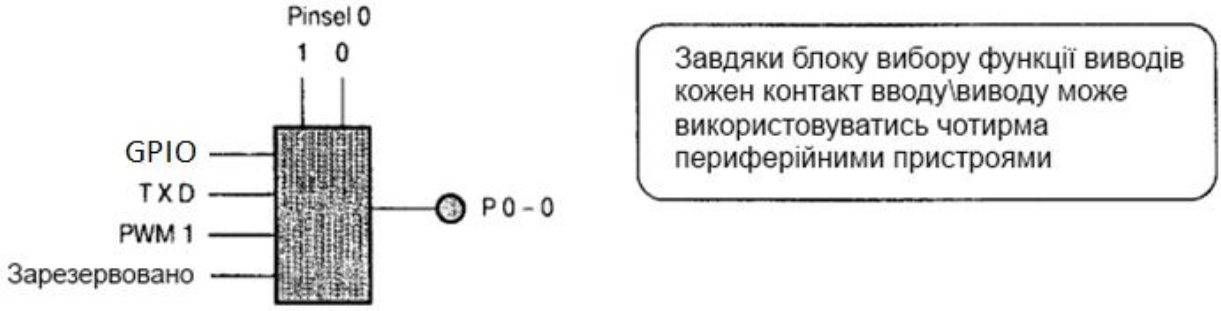


Рисунок 3.19 – Блок вибору функції виводів

При скиданні всі контакти введення/виведення задіяні в якості ліній портів введення/виведення загального призначення. Додаткові функції вибираються за допомогою регістрів PINSEL. Наприклад, лінія зовнішнього переривання EINT0 є одночасно контактом порту введення/виведення загального призначення P2.10. Утримування цієї лінії в стані НИЗЬКОГО рівня при скиданні мікроконтролера викликає запуск вбудованого обробника команд завантажувача. Тому для використання зовнішнього переривання ми повинні записати в регістр вибору функції виведення відповідне значення, що підключає вивід до схеми переривань.

3.8.3 Виводи зовнішніх переривань

Керування зовнішніми перериваннями здійснюється за допомогою чотирьох регістрів. Регістром EXMODE задається тип переривання – за фронтом або за рівнем. Якщо використовується переривання за фронтом, то в регістрі EXPOL задається активний фронт (наростаючий або спадаючий), при виявленні якого буде генеруватися переривання. Якщо використовується переривання за рівнем, то воно може виникнути лише при НИЗЬКОМУ у рівні на вході

переривання. За допомогою регістра EXWAKE можна дозволити вихід процесора з режиму Power Down за зовнішнім перериванням (рисунок 3.20). Для ініціалізації навчальної програми потрібно задати для переривання EINT0 чутливість за рівнем і підключити цей вивід до процесора за допомогою регістра PINSEL0.

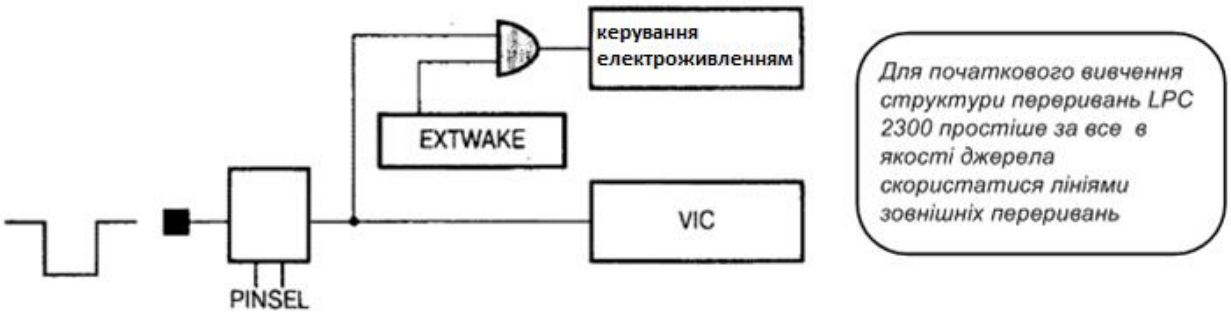


Рисунок 3.20 – Конфігурування лінії зовнішнього переривання

3.8.4 Структура переривань

ЦПП ARM7 має дві лінії зовнішніх переривань: одну для режиму оброблення запиту на швидке переривання (FIQ), а іншу для режиму обробки запиту на звичайне переривання (IRQ). В системах на базі ARM7 повинно бути тільки одне джерело переривань, що генерує переривання FIQ, щоб процесор міг якомога швидше перейти в цей режим і приступити до обробки переривання. З цього випливає, що інші джерела переривань повинні бути підключені до лінії переривання IRQ. У простих системах дані джерела можна об'єднати за допомогою багатовходового елемента «АБО». У цьому випадку при виникненні переривання процесор повинен перевірити кожний периферійний пристрій для визначення джерела переривання, що може зайняти досить багато часу. Очевидно, що нам потрібно більш складне рішення.

Для більш ефективної обробки зовнішніх переривань в мікроконтролери було додано спеціальний модуль (рисунок 3.21), який називають векторним контролером переривань (VectorInterruptController–VIC).

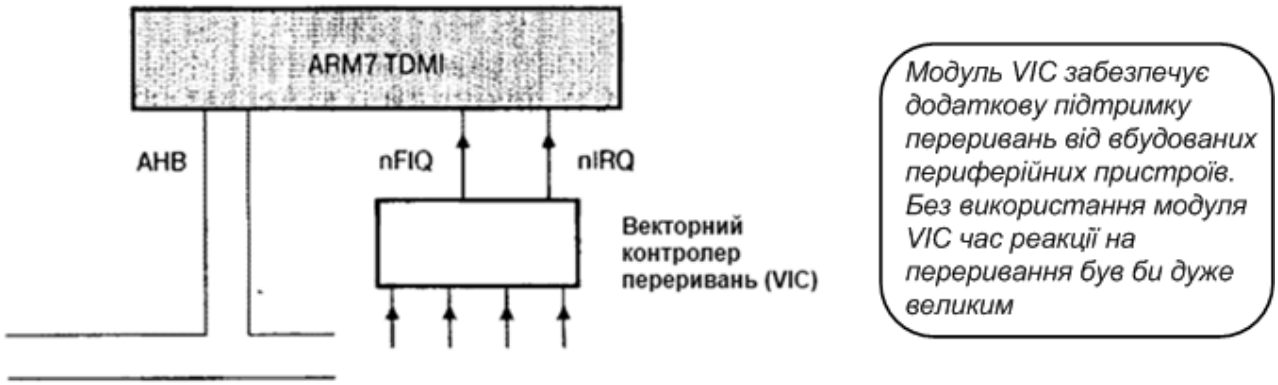


Рисунок 3.21 – Підтримка переривань за допомогою модуля VIC

Модуль VIC являє собою високо оптимізований контролер переривань. Модуль VIC використовується для обробки всіх джерел переривань від вбудованих периферійних пристроїв. Кожен з внутрішніх джерел переривань підключено до певного каналу модуля VIC. У програмі кожен з цих каналів можна підключити до ліній переривань ЦПП (FIQ і IRQ) одним із трьох способів.

Кожне переривання може оброблятись як швидке переривання FIQ, векторне переривання IRQ або ж не векторне переривання IRQ. Всі три типи переривань мають різний час відгуку. Найшвидшим є переривання FIQ, далі за ним йде векторне переривання, а найповільніше процесор реагує на невекторні переривання IRQ. Нижче розглядаються ці три методи обробки переривань. Регістри модуля VIC зображено на рисунку 3.22.

3.8.4.1 Переривання FIQ

Будь яке джерело переривання може використовуватися в якості переривання FIQ. У регістрі вибору типу переривання модуля VIC (регістр VICIntSelect) кожному джерелу переривань відповідає свій біт. При встановленні цього біта обраний канал контролера підключається до лінії переривання FIQ. В ідеальному випадку у нас має бути тільки одне переривання FIQ.

Адреса	Тип	Розмір	Значення скидання	Назва
VIC base +0x000	Read	32	0x00000000	VICIRQSTATUS
VIC base +0x004	Read	32	0x00000000	VICFIQSTATUS
VIC base +0x008	Read	32	-	VICRAWINTR
VIC base +0x00C	Read/write	32	0x00000000	VICINTSELECT
VIC base +0x0010	Read/write	32	0x00000000	VICINTENABLE
VIC base +0x014	Write	32	-	VICINTENCLEAR
VIC base +0x018	Read/write	32	0x00000000	VICSOFTINT
VIC base +0x01C	Write	32	-	VICSOFTINTCLEAR
VIC base +0x020	Read/write	1	0x0	VICPROTECTION
VIC base +0x024	Read/write	16	0xFFFF	VICSWPRIORITY MASK
VIC base +0x028	Read/write	4	0xF	VICPRIORITYDAISY
VIC base +0x100	Read/write	32	0x00000000	VICVECTADDR0
VIC base +0x104	Read/write	32	0x00000000	VICVECTADDR1
	● ● ●			
VIC base +0x17C	Read/write	32	0x00000000	VICVECTADDR31
VIC base +0x200	Read/write	4	0xF	VICVECTPRIORITY0
VIC base +0x204	Read/write	4	0xF	VICVECTPRIORITY1
	● ● ●			
VIC base +0x27C	Read/write	4	0xF	VICVECTPRIORITY31
VIC base +0xF00	Read/write	32	0x00000000	VICADDRESS
VIC base +0xFE0	Read	8	0x92	VICPERIPHID0
VIC base +0xFE4	Read	8	0x11	VICPERIPHID1

Рисунок 3.22 – Адреси регістрів модуля VIC

Тим не менш, при одночасному встановленні декількох бітів вказаного регістра буде дозволено кілька джерел переривань FIQ. У цьому випадку за допомогою регістра стану FIQ модуля VIC (VICFIQStatus) можна буде при вході

в обробник визначити конкретне джерело переривання і перейти до необхідної секції обробника.

Очевидно, що наявність декількох джерел переривань FIQ уповільнює перехід до процедури обробки переривань. Після призначення джерела переривання FIQ, переривання можна дозволити, використовуючи регістр дозволу переривань модуля VIC (VICIntEnable). Крім того, необхідно сконфігурувати всі периферійні пристрої, що генерують переривання, а також дозволити їм генерацію цих переривань. При генерації переривання FIQ процесор переходить в режим FIQ, а програма переходить за адресою 0x0000001C – вектору переривання FIQ. Для обслуговування переривання треба розмістити в цьому місці команду переходу до потрібної процедури обробки переривання.

3.8.4.1.1 Вихід із переривання FIQ

Оголошення Сі–функції в якості оброблювача переривання FIQ вказує компілятору використовувати коректну команду повернення для відновлення виконання фонового коду з того місця, де його виконання було перерване. Однак перш ніж вийти з процедури обробки переривання, потрібно переконатися, що прапорці переривань всіх периферійних пристроїв скинуто (рисунок 3.23).

Рисунок 3.23 – Скидання прапорця переривання

В іншому випадку переривання буде генеруватися постійно, доки відповідний прапорець не буде скинуто. Для скидання прапорця треба записати у відповідний йому біт регістра не 0, а 1.

3.8.4.2 Векторні переривання IRQ

Якщо одне з джерел переривань визначено як переривання FIQ, то інші джерела мають бути підключені до лінії IRQ. Для швидкої та ефективної обробки цих переривань в модулі VIC мається апаратно реалізована

програмована таблиця, в якій зберігаються адреси всіх підпрограм обробки переривань. У модулі VIC є 32 слоти для векторної адресації переривань. Для кожного слоту передбачено два регістри – регістр адреси вектора (VICVectAddr) і регістр керування вектором (VICVectCntl).

Регістри пріоритету вектора (VICVectPrioityxx) дозволяють призначати кожному слоту переривань свій пріоритет. У загальній складності підтримується 16 рівнів пріоритету, від 15 (найменший пріоритет) до 0 (найбільший пріоритет). Після скидання всі слоти переривань мають мінімальний рівень пріоритету, який може бути збільшений користувачем. Якщо двом слотам буде призначений один і той же пріоритет, то при виникненні переривань від обох джерел першим буде обслугований слот з найменшим номером.

Другим регістром для керування слотом в модулі VIC є регістр адреси вектора (VICVectAddr). Як випливає з його назви, в цьому регістрі повинна зберігатися адреса відповідної Сі–функції, яка буде запускатися при виникненні переривання, пов'язаного з даним слотом. Насправді, при генерації векторного переривання канал переривання підключається до заданого йому слоту, а адреса підпрограми обробки переривання, що знаходиться в регістрі адреси вектора даного слоту, завантажується в окремий регістр адреси вектора (рисунок 3.24). Таким чином, при генерації переривання, яке сконфігуровано як векторне, адреса його оброблювача буде завантажена в фіксовану комірку пам'яті, яка називається регістром адреси вектора (VICVectAddrxx).

Поки модуль VIC виконує всі ці дії, процесор ARM7 перемикається в режим IRQ і переходить за адресою вектора переривання IRQ (0x00000018). Щоб виконати відповідну підпрограму обробки переривання, необхідно завантажити в PC адресу з регістра адреси вектора модуля VIC. Наступна команда асемблера виконує цю операцію за один такт:

```
LDR        PC, [PC, #–0x0120].
```

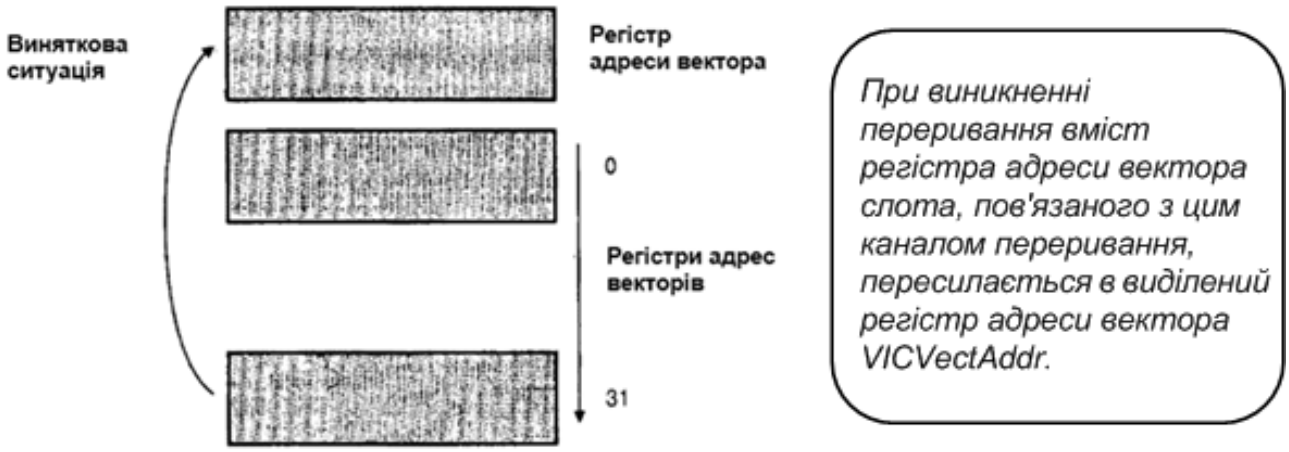


Рисунок 3.24 – Отримання адреси векторного переривання IRQ

Оскільки ми вже знаходимося в режимі IRQ, то нам відомо, що поточна адреса дорівнює 0x00000018 + 8 (з урахуванням конвеєра). При відніманні з цього значення числа 0x0120 відбудеться перехід через верхню межу 32–бітного адресного простору, в результаті чого вказана команда завантажить в лічильник команд вміст комірки пам'яті з адресою 0xFFFFFF00 (вміст регістра адреси вектора), рисунок 3.25.

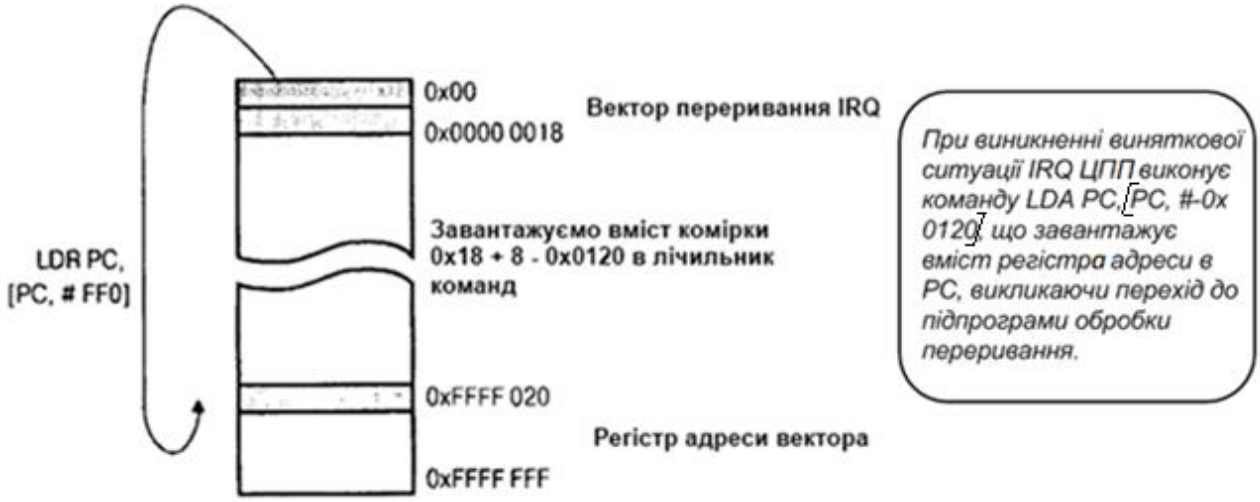


Рисунок 3.25 – Обробка векторного переривання IRQ

Як і у випадку переривання FIQ, потрібно переконатися в тому, що всі прапорці переривань периферійного пристрою, які генерують запит на переривання, скинуті. Крім того, в кінці обробника переривання потрібно виконати холосту операцію запису в регістр адреси вектора VICVectAddr

(рисунок 3.26). Ця операція вказує модулю VIC на закінчення обробки переривання і дозволяє оброблення відкладених переривань IRQ.

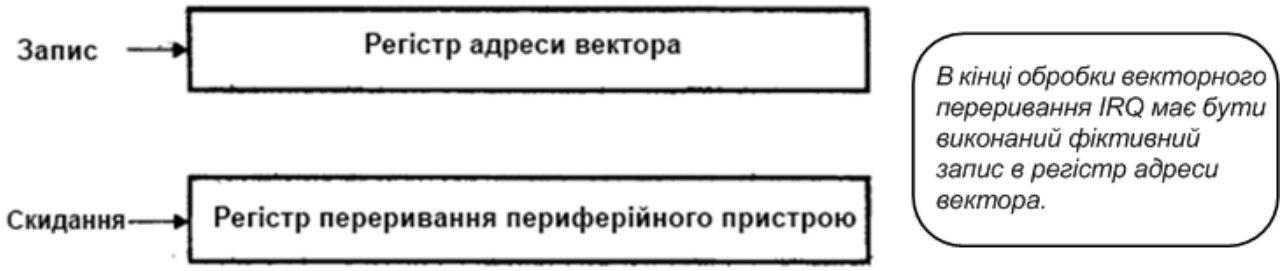


Рисунок 3.26 – Закінчення обробки векторного переривання IRQ

3.8.5 Невекторні переривання

Модуль VIC може обробляти до 32 векторних переривань IRQ і принаймні одне швидке переривання FIQ. Якщо в мікроконтролері є більше 33 джерел переривань, то решта може обслуговуватися як невекторні переривання. Всі джерела невекторних переривань обслуговуються однією єдиною підпрограмою обробки переривань. Адреса цієї підпрограми зберігається в додатковому регістрі адреси вектора, що називається регістром адреси вектора за замовчуванням (VICDefVectAddr). Якщо в модулі VIC переривання дозволено і при цьому не сконфігуровано як переривання FIQ, то воно буде сприйнято як невекторне переривання. При генерації такого переривання адреса із регістра VICDefVectAddr перезавантажується в регістр адреси вектора VICVectAddr, що викликає перехід процесора до цієї підпрограми. При вході в цю підпрограму ЦПП повинен прочитати регістр статусу IRQ, щоб визначити, яке із джерел невекторних переривань згенерувало його.

3.8.6 Програмні переривання

Модуль VIC надає можливість прикладному ПЗ генерувати переривання по кожному з каналів, використовуючи регістри програмного переривання модуля (рисунок 3.27). Ці регістри не мають ніякого відношення до команди програмного переривання (SWI), однак дозволяють перевірити джерела переривань (при включенні живлення або ж при налагодженні під час розробки програми).

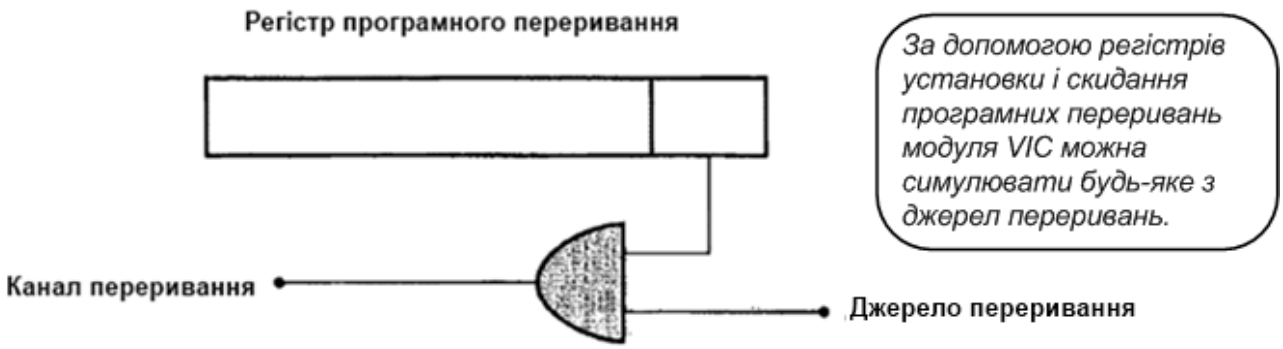


Рисунок 3.27 – Регістр програмного переривання

Крім усього іншого, в модулі VIC передбачено спеціальний захищений режим, що запобігає доступ до будь якого регістра модуля в режимі User. Якщо прикладній програмі потрібен доступ до модуля VIC, то вона повинна перемкнутися в привілейований режим. Для цього можна використовувати переривання FIQ або IRQ, а також команду SWI.

Типові значення затримок для різних переривань наведено у [4]. У разі невекторних переривань затримка складається з часу реакції на векторне переривання і часу, необхідного для зчитування регістра стану IRQ і визначення джерела.

3.8.7 Вкладені переривання

На апаратному рівні ЦПП ARM7 і модуль VIC не підтримують вкладені переривання. Якщо у вашій програмі вимагається можливість переривати виконання підпрограми обробки переривань, це можна зробити програмно за допомогою пари макросів (рисунок 3.28). Перш ніж перейти до розгляду вкладених переривань, треба звернути увагу на те, що при обробці процесором ARM7 зовнішнього переривання, переривання IRQ забороняються. Крім того, при вході в Сі–функцію, оголошену як обробник переривання IRQ, регістр LR_isr зберігається в стеку.

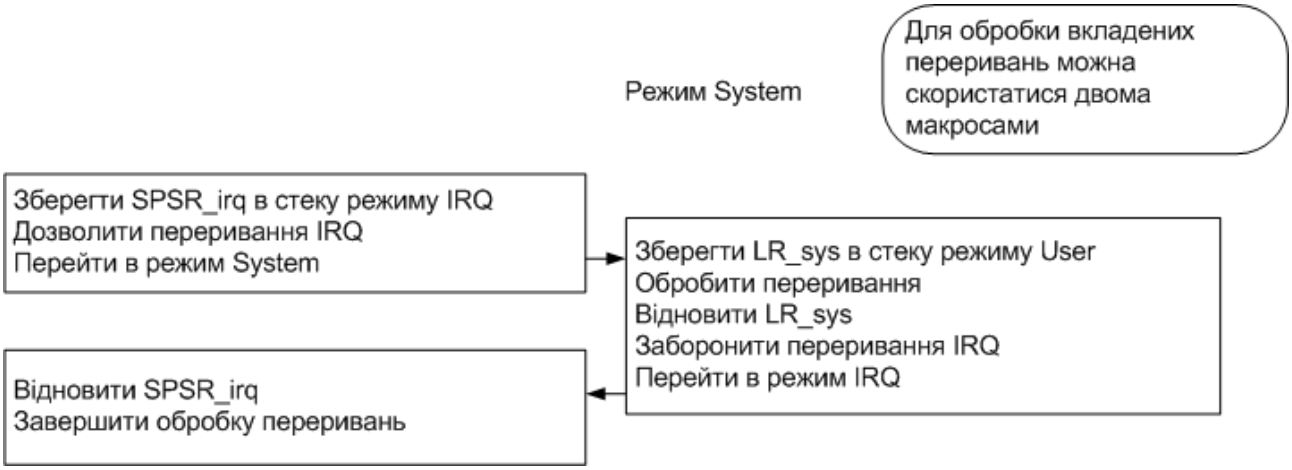


Рисунок 3.28 – Обробка вкладених переривань

Після того як процесор увійшов в підпрограму обробки переривання IRQ, нам потрібно виконати ряд команд для включення підтримки вкладених переривань. Перш за все, необхідно зберегти регістр SPSR_irq в стеку. Це дозволить нам коректно відновити стан регістра CPSR при поверненні в режим User. Потім треба дозволити переривання IRQ і перейти в режим System (цей режим аналогічний режиму User, проте в ньому можна використовувати команди MSR та MRS). У режимі System необхідно зберегти новий регістр зв'язку, тому що в ньому може зберігатися значення, яке використовується в фоні (в режимі User). Тому даний регістр заноситься в стек режиму System. Після виконання цих операцій ми можемо приступити до виконання підпрограми обробки переривання. Після завершення обробки запускається макрос, що виконує зворотні операції. Цей макрос відновлює стан регістра зв'язку, забороняє переривання IRQ, перемикає процесор в режим IRQ, відновлює регістр SPRS_irq, після чого обробка переривань вважається завершеною. Код описаних макросів наведено нижче.

```
#define IENABLE
        __asm { MRS         LR, SPSR }
        __asm { STMFD  SP!, {LR} }
        __asm { MSR         CPSR_c, #0x1F }
        __asm { STMFD  SP’, {LR} }

#define IDISABLE
```

```
__asm { LDMFD  SR!, {LR} }
__asm { MSR            CPSR_c, #0x92 }
__asm { LDMFD  SP!, {LR} }
__asm { MSR            SPSR_cxsf, LR }
```

Цей код складається з восьми команд (32 байти коду ARM), а загальний час виконання обох макросів становить 230 нс. Описана методика дозволяє реалізувати обробку відповідного переривання іншим перериванням. Якщо вам необхідно задати пріоритети вкладених переривань, то в макросі слід передбачити блокування переривань з низьким пріоритетом шляхом заборони в модулі VIC джерел переривань з більш низьким пріоритетом.

3.8.8 Джерела переривань

В таблиці 3.10 наведено список джерел переривань для окремих периферійних пристроїв.

Таблиця 3.10 – Джерела переривань

Блок	Прапорці	Канал VIC
WDT	Переривання вартового таймера (WDINT)	0
–	Зарезервовано для програмних переривань	1
Ядро ARM	Вбудований ICE DbgCommRx	2
Ядро ARM	Вбудований ICE DbgCommTx	3
TIMER0	Співпадіння 0..3 (MR0, MR1, MR2, MR3) Захоплення 0…3 (CR0, CR1, CR2, CR3)	4
TIMER1	Співпадіння 0..3 (MR0, MR1, MR2, MR3) Захоплення 0…3 (CR0, CR1, CR2, CR3)	5
UART0	Стан лінії Rx (RLS) Регістр передачі пустий (THRE) Прийняття даних доступу (RDA) Індикатор тайм–ауту (CTI)	6

Продовження таблиці 3.10

Блок	Прапорці	Канал VIC
UART1	Стан лінії Rx (RLS) Регістр передачі пустий (THRE) Прийняття даних доступу (RDA) Індикатор тайм–ауту (CTI) Переривання за станом модему	7
PWM0	Співпадіння 0...6 (MR0, MR1, MR2, MR3, MR4, MR5, MR6)	8
I2C	SI (зміна стану)	9
SPI0	Прапорець переривання від SPI0 (SPIF) Режим помилки (MODF)	10
SPI1	Прапорець переривання від SPI1(SPIF) Режим помилки (MODF)	11
PLL	Петлю ФАПЧ замкнуто (PLOCK)	12
RTC	Лічильник інкрементування (RTCCIF) Тривога (RTCALF)	13
Керування системою	Зовнішнє переривання 0 (EINT0)	14
Керування системою	Зовнішнє переривання 1 (EINT1)	15
Керування системою	Зовнішнє переривання 2 (EINT2)	16
Керування системою	Зовнішнє переривання 3 (EINT3)	17
A/D	АЦП	18

Продовження таблиці 3.10

Блок	Прапорці	Канал VIC
CAN	CAN1 Tx	20
	CAN2 Tx	21
	CAN3 Tx (лише у LPC2194/2292/2294 інакше зарезервовано)	22
	CAN4 Tx (лише у LPC2194/2292/2294 інакше зарезервовано)	23
	Зарезервовано	24,25
	CAN1 Rx	26
	CAN2 Rx	27
CAN	CAN3 Rx (лише у LPC2194/2292/2294 інакше зарезервовано)	28
	CAN3 Rx (лише у LPC2194/2292/2294 інакше зарезервовано)	29
	Зарезервовано	30,31

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Яку роль виконують переривання в мікропроцесорних системах?
2) Що відбувається в роботі мікропроцесорної системи при виникненні переривання?
3) Назвіть два види переривань в мікропроцесорних системах.
4) Чим відрізняються апаратне та програмне переривання?
5) Чим відрізняються переривання загального призначення (IRQ) і швидке переривання (FIQ)?
6) Поясніть призначення блоку вибору функції виводів.
7) За допомогою яких регістрів здійснюється керування зовнішніми перериваннями?
8) Який модуль в мікроконтролерах ARM7 використовується для більш ефективної обробки зовнішніх переривань?
9) Для обробки яких джерел переривань використовується модуль VIC?
10) Як в мікроконтролері програмується переривання FIQ?
11) Як виконується вихід із переривання FIQ?
12) Опишіть особливості обробки векторних переривань IRQ.
13) Яку адресу має вектор переривання IRQ?
14) Як виконується закінчення обробки векторного переривання IRQ?
15) Які переривання можуть обслуговуватися як невекторні?
16) Якою підпрограмою обслуговуються джерела невекторних переривань?
17) Як модуль VIC надає можливість прикладному ПЗ генерувати переривання?
18) Як в мікроконтролері підтримується обробка вкладених переривань?
19) Назвіть джерела переривань від окремих периферійних пристроїв.
20) Де знаходиться підпрограма, яка викликається за перериванням?
21) Наведіть приклади подій, які можуть ініціювати зовнішнє переривання.

3.9 Модуль прямого доступу до пам’яті

3.9.1 Загальна характеристика

Контролер прямого доступу до пам'яті (Direct Memory Access – DMA) є представником сімейства модулів PrimeCell™, розроблених компанією ARM, і оптимізований для роботи спільно з шинною структурою ядра ARM [1, 4, 7]. Контролер DMA загального призначення підключено до шини AHB через два порти – ведений, через який ЦПП може звертатися до регістрів контролера, і ведучий, за допомогою якого механізм прямого доступу до пам'яті здійснює захоплення шини, а також виконує її арбітраж з ЦПП ARM7 і спеціалізованими модулями DMA, контролерами USB і Ethernet.

У складі контролера DMA є два незалежних модуля прямого доступу до пам'яті, кожен з яких може бути налаштований для здійснення пересилання даних з пам'яті в пам'ять, з пам'яті до периферійного пристрою, з периферійного пристрою в пам'ять, а також з одного периферійного пристрою в інший (рисунок 3.29). По закінченню пересилання контролер DMA може згенерувати переривання, при цьому сигнали переривань від усіх модулів DMA об'єднуються за схемою АБО і підключаються до одного – єдиного каналу переривань модуля VIC.

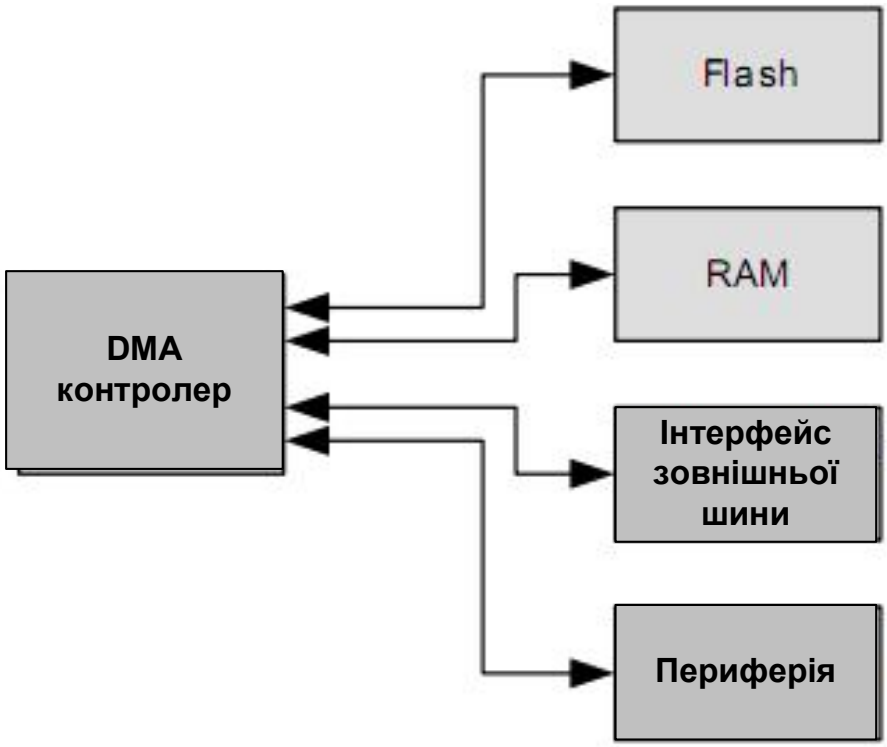


Рисунок 3.29 – Види пересилання даних контролером DMA

3.9.2 Огляд модуля DMA

Для того, щоб вивчити роботу модуля DMA, розглянемо, для початку, найпростішу операцію – пересилання даних з пам'яті в пам'ять. Для виконання цієї операції модуль DMA спочатку виграє арбітраж на шині AHB, після чого копіює дані з джерела у свій внутрішній FIFO–буфер. Потім вміст внутрішнього буфера записується в задану область пам'яті. Зчитування і запис модулем DMA даних може здійснюватися як у вигляді одноразових пересилань, так і у вигляді пакетів з декількох пересилань. В останньому випадку модуль DMA може блокувати внутрішні шини до завершення кожного етапу пересилання. Також модуль DMA може зчитувати і записувати блоки даних різного розміру. Це означає, зокрема, що в процесі операції пересилання може здійснюватися упакування і розпакування даних. Наприклад, можна зчитати з пам'яті чотири 32 – бітних слова даних, а потім пересилати їх в буфер передавача UART по 16 біт. Після того, як модуль DMA виграє арбітраж на шині і буде готовий до пересилання даних, він дозволить керувати процесами зчитування і запису цих даних. Однак у разі пересилання даних між пам’яттю і периферійними пристроями у якості контролера обміну може виступити саме периферійний пристрій, що дозволяє запис або зчитування даних по мірі готовності. Крім того, модуль DMA підтримує пересилання несуміжних даних. Для кожного з модулів ви можете у вигляді пов'язаного списку задати послідовність пересилань, які згодом будуть автоматично виконуватися один за одним. Це дозволяє зчитувати дані з несуміжних областей пам'яті, а потім і передавати їх у вигляді єдиного блоку в пам'ять або периферійний пристрій. Аналогічно можна запрограмувати модуль DMA таким чином, щоб він «розкидав» вміст єдиного блоку пам'яті за різними адресами.

Всього у контролері DMA є 24 регістри, які можна розбити на три групи: 14 регістрів стану і конфігурації та по 5 регістрів керування кожного з модулів. Загальні регістри стану і конфігурації (рисунок 3.30) відповідають, в основному, за дозвіл роботи контролера DMA і керування індивідуальними перериваннями модулів.

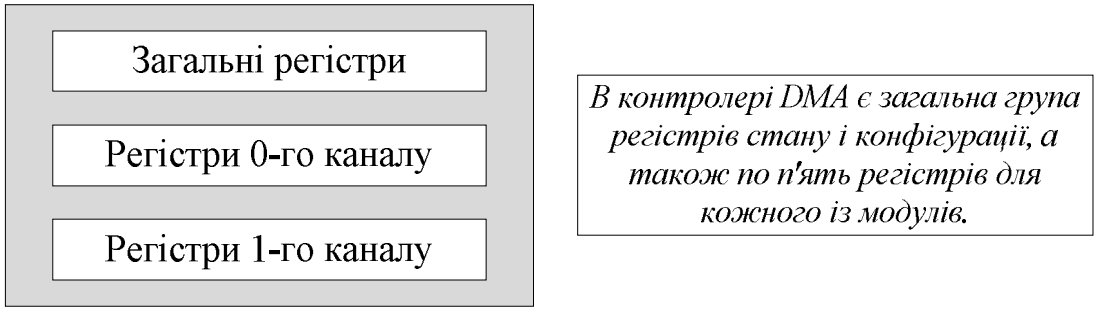


Рисунок 3.30 – Групи регістрів контролера DMA

Дозвіл роботи контролера DMA здійснюється встановленням біта E регістра конфігурації DMACConfiguration. Кожний з модулів DMA має два переривання – переривання terminal count, що генерується при завершенні пересилання, і переривання помилки, яке генерується при виявленні модулем помилки на шині. Дозволи для обох джерел переривань здійснюються в регістрі конфігурації каналу DMACCOConfiguration і DMACCIConfiguration. Крім того, для кожного джерела переривання мається по парі регістрів стану: регістр стану переривання і регістр необробленого стану переривання. Останній регістр відображає стан усіх прапорців переривань, незалежно від того, було відповідне переривання дозволено чи ні, тоді як звичайний регістр стану переривання відображає стан тільки дозволених переривань DMA. Для ініціювання одиночного або групового пересилання за DMA, у разі, якщо модуль DMA виступає в ролі контролера потоку, використовуються регістри програмного запиту на одиночне (DMACSoftSReq) і пакетне пересилання (DMACSoftBReq) відповідно.

3.9.3 Синхронізація DMA

Модулі DMA можуть здійснювати пересилання через всі внутрішні шини мікроконтролерів LPC2300/2400. Якщо ці шини працюють з різними швидкостями, необхідно в регістрі синхронізації DMACSync встановити в 1 біти синхронізації для різних запитів DMA. Це дозволить уникнути виникнення конфліктів на шинах, але й збільшить час відгуку модулів.

3.9.4 Пересилання з пам'яті в пам'ять

Після дозволу роботи модуля DMA і конфігурування переривань можна приступати до конфігурування власне регістрів каналів. При здійсненні пересилання даних з пам'яті в пам'ять початкові адреси джерела і приймача

заносяться в однойменні регістри (DMACCxSrcAddr і DMACCxDcstAddr). Регістр елемента зв’язаного списку використовується при пересиланні несуміжних даних і для одиночного пересилання повинен бути скинутий в 0.

У регістрі керування (рисунок 3.31) можна задати розмір пересилання, у разі якщо роль контролера потоку грає модуль DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCIE	PROT 2	PROT 1	PROT 0	DI	SI	0	0	DWIDTH			SWIDTH			DBSize[2:1]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB Size0	DBSize[2:1]			TransferSize[11:0]											
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Рисунок 3.31 – Регістр керування модуля DMA

Якщо ж в якості контролера потоку виступає периферійний пристрій, то в цьому полі регістра повинен бути записаний 0. У процесі пересилання вміст даного поля декрементується, однак, щоб прочитати його і отримати осмислене значення, вам необхідно спочатку заборонити роботу модуля DMA. Поля розрядності джерела та приймача дозволяють задати розмір слів, що зчитуються і записуються модулем DMA. При цьому можна задавати різні значення вказаних полів. Упакування та розпакування даних буде здійснюватися модулями автоматично. При необхідності адреси джерела і приймача можуть інкрементуватися після кожного пересилання, для чого слід встановити біти SI і DI відповідно. Завдяки цьому ви зможете переслати блок даних з однієї безперервної області пам'яті в іншу або ж скопіювати блок даних в якусь область пам'яті, наприклад, групу регістрів периферійного пристрою. У цьому ж регістрі можна задати різні опції захисту. Встановивши біт PROT0, можна заборонити на час пересилання звернення до регістрів DMA з коду, що виконується в режимі User. Біт PROT1 дозволяє включити буферизацію. При цьому пересилання даних буде здійснюватися з максимально можливою швидкістю, проте може відбутися порушення однорідності даних, оскільки буферизовані дані повинні бути записані в ОЗП. Якщо встановити біт дозволу переривання термінального рахунку, то в кінці кожного DMA – пересилання буде генеруватися переривання, яке повідомить ЦПП про те, що модуль DMA готовий до подальшого

використання. Останнє поле регістра керування дозволяє задати розмір пакетного пересилання.

3.9.5 Пакетна передача

Кожен з модулів DMA здатний зчитувати одиночне слово і передавати його за заданою адресою. Перш ніж модуль DMA зможе виступити в якості майстра шини, він повинен на час цих операцій відібрати шину у ЦПП і інших модулів DMA. Сконфігурувавши відповідним чином поля розміру пакета джерела і приймача в регістрі керування, можна здійснювати пересилання декількох слів даних в пакетному режимі. При цьому кожен з модулів DMA дозволяє здійснювати в пакетному режимі до 256 пересилань. Встановивши біт L регістра конфігурації каналу (DMACCxConfiguration), можна попередити звільнення шини модулем DMA до завершення пересилання.

3.9.6 Підтримка модулем DMA периферійних пристроїв

У ролі контролера потоку для кожного з модулів DMA можуть виступати периферійні пристрої.

Необхідно дозволити використання модуля DMA конкретним периферійним пристроєм і записати в поля пристрою – джерела і пристрою – приймача регістра конфігурації каналу DMA (DMACCxConfiguration) значення, відповідні використовуваному пристрою (рисунок 3.32).

Після дозволу роботи модуля DMA він буде керуватися відповідним периферійним пристроєм.

Рисунок 3.32 – Регістр конфігурації модуля DMA

3.9.7 Пересилання несуміжних даних

Обидва модулі DMA підтримують пересилання несуміжних даних. Цей механізм дозволяє запрограмувати виконання декількох DMA – пересилань в заданій послідовності, в результаті чого можна буде зібрати дані, які розташовано в різних областях пам'яті, і помістити їх в одну безперервну область. Точно також за допомогою декількох DMA – пересилань, що виконуються автоматично, можна «розкидати» дані з однієї області пам'яті за різними адресами. Області пам'яті в ОЗП задаються у вигляді «елементів» списку, що представляють собою запис, в якому міститься адреса джерела, адреса приймача, адреса наступного елемента списку і керуюче слово. Початкова адреса такого елемента заноситься в регістр елемента зв’язаного списку модуля DMA, і після закінчення поточного пересилання елемент, на який посилається поточний, буде автоматично завантажений у регістр керування каналу DMA. При цьому регістр елемента зв’язаного списку буде вказувати на наступний елемент. За рахунок такого механізму зв’язаних списків ми можемо зв'язати кілька DMA – пересилань. Для останнього пересилання в ланцюжку необхідно в регістрі керування дозволити переривання DMA з тим, щоб після його завершення було згенеровано переривання і можна було б ініціалізувати нові DMA – пересилання.

3.9.8 Регістри керування модулем DMA

Регістри керування модулем DMA наведено у таблиці 3.11.

Таблиця 3.11 – Регістри керування модулем DMA

Назва	Опис	Доступ	Початкове значення	Адреса
Загальні регістри				
DMACIntStatus	Interrupt Status Register	RO	0x0	0xFFE0 4000
DMACIntTCStatus	Interrupt Terminal Count Status Register	RO	0x0	0xFFE0 4004
DMACIntTCClear	Interrupt Terminal Count Clear Register	WO	–	0xFFE0 4008
DMACIntErrorStatus	Interrupt Error Status Register	RO	0x0	0xFFE0 400C

Продовження таблиці 3.11

Назва	Опис	Доступ	Початкове значення	Адреса
DMACIntErrClr	Interrupt Error Clear Register	WO	–	0xFFE0 4010
DMACRawIntTCStatus	Raw Interrupt Terminal Count Status Register	RO	–	0xFFE0 4014
DMACRawIntErrorStatus	Raw Error Interrupt Status Register	RO	–	0xFFE0 4018
DMACEnbldChns	Enabled Channel Register	RO	0x0	0xFFE0 401C
DMACSoftBReq	Software Burst Request Register	R/W	0x0000	0xFFE0 4020
DMACSoftSReq	Software Single Request Register	R/W	0x0000	0xFFE0 4024
DMACSoftLBReq	Software Last Burst Request Register	R/W	0x0000	0xFFE0 4028
DMACSoftLSReq	Software Last Single Request Register	R/W	0x0000	0xFFE0 402C
DMACConfiguration	Configuration Register	R/W	0x0000 0000	0xFFE0 4030
DMACSync	Synchronization Register	R/W	0x0000	0xFFE0 4034
Регістри Channel 0				
DMACC0SrcAddr	Channel 0 Source Address Register	R/W	0x0000 0000	0xFFE0 4100
DMACC0DestAddr	Channel 0 Destination Address Register	R/W	0x0000 0000	0xFFE0 4104
DMACC0LLI	Channel 0 Linked List Item Register	R/W	0x0000 0000	0xFFE0 4108
DMACC0Control	Channel 0 Control Register	R/W	0x0000 0000	0xFFE0 410C
DMACC0Configuration	Channel 0 Configuration Register	R/W	0x00000 [1]	0xFFE0 4110
Регістри Channel 1				
DMACC1SrcAddr	Channel 1 Source Address Register	R/W	0x0000 0000	0xFFE0 4120
DMACC1DestAddr	Channel 1 Destination Address Register	R/W	0x0000 0000	0xFFE0 4124
DMACC1LLI	Channel 1 Linked List Item Register	R/W	0x0000 0000	0xFFE0 4128
DMACC1Control	Channel 1 Control Register	R/W	0x0000 0000	0xFFE0 412C
DMACC1Configuration	Channel 1 Configuration Register	R/W	0x00000 [1]	0xFFE0 4130

1 – біт [17] тільки для читання

3.9.8.1 Загальні регістри

3.9.8.1.1 Регістр статусу переривань (Interrupt Status Register, DMACIntStatus – 0xFFE0 4000)

Регістр DMACIntStatus (таблиця 3.12) доступний тільки для читання і відображає стан переривань після маскування. Високий рівень позначає активний запит конкретного DMA каналу на переривання. Запит може бути викликаний або через помилку, або через запити на переривання термінального рахунку.

Таблиця 3.12 – Опис розрядів регістра DMACIntStatus

Біт	Символ	Опис	Початкове значення
0	IntStatus0	Стан переривань 0 каналу після маскування	0
1	IntStatus1	Стан переривань 1 каналу після маскування	0
31:2	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A

3.9.8.1.2 Регістр статусу переривань термінального рахунку (Interrupt Terminal Count Status Register, DMACIntTCStatus – 0xFFE0 4004)

Регістр DMACIntTCStatus (таблиця 3.13) доступний тільки для читання і відображає стан термінального рахунку після маскування.

Таблиця 3.13 – Опис розрядів регістра DMACIntTCStatus

Біт	Символ	Опис	Початкове значення
0	IntTCStatus0	Стан запиту термінального рахунку на переривання 0 каналу	0
1	IntTCStatus1	Стан запиту термінального рахунку на переривання 1 каналу	0
31:2	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A

3.9.8.1.3 Регістр скидання переривання термінального рахунку (Interrupt Terminal Count Clear Register, DMACIntClear – 0xFFE0 4008)

Регістр DMACIntTCClear (таблиця 3.14) доступний тільки для запису та скидає запит термінального рахунку на переривання. При записі до регістра, кожен біт даних високого рівня скидає відповідний біт регістра статусу, біти низького рівня не впливають на відповідні біти регістра.

Таблиця 3.14 – Опис розрядів регістра DMACIntTCClear

Біт	Символ	Опис	Початкове значення
0	IntTCClear0	Запис 1 скидає запит термінального рахунку на переривання для 0 каналу	–
1	IntTCClear1	Запис 1 скидає запит термінального рахунку на переривання для 1 каналу	–
31:2	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A

3.9.8.1.4 Регістр статусу переривання за помилкою (Interrupt Error Status Register, DMACIntErrorStatus – 0xFFE0 400C)

Регістр DMACIntErrorStatus (таблиця 3.15) доступний для читання і відображає стан помилки переривання після маскування.

Таблиця 3.15 – Опис розрядів регістра DMACIntErrorStatus

Біт	Символ	Опис	Початкове значення
0	IntErrorStatus0	Стан переривання за помилкою для каналу 0	0x0
1	IntErrorStatus1	Стан переривання за помилкою для каналу 1	0x0
31:2	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A

3.9.8.1.5 Регістр скидання переривання за помилкою (Interrupt Error Clear Register, DMACIntErrClr – 0xFFE0 4010)

Регістр DMACIntErrClr (таблиця 3.16) доступний тільки для запису і скидає запити на переривання «помилки». При запису до регістра, кожен біт даних високого рівня скидає відповідний біт регістра статусу, біти низького рівня не впливають на відповідні біти регістра.

Таблиця 3.16 – Опис розрядів регістра DMACIntErrClr

Біт	Символ	Опис	Початкове значення
0	IntErrClr0	Запис 1 скидає запит на переривання «помилки» для 0 каналу (IntErrorStatus0)	–
1	IntErrClr1	Запис 1 скидає запит на переривання «помилки» для 1 каналу (IntErrorStatus1)	–
31:2	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A

3.9.8.1.6 Регістр статусу необробленого переривання термінального рахунку (Raw Interrupt Terminal Count Status Register, DMACRawIntTCStatus – 0xFFE0 4014)

Регістр DMACRawIntTCStatus (таблиця 3.17) доступний тільки для читання і відображає стан каналу, який надіслав запит на закінчення передачі (переривання термінального рахунку) до маскування. Високий рівень біта відображає активне переривання термінального рахунку до маскування.

Таблиця 3.17 – Опис розрядів регістра DMACRawIntTCStatus

Біт	Символ	Опис	Початкове значення
0	RawIntTCStatus0	Стан переривання термінального рахунку для каналу 0 до маскування.	–
1	RawIntTCStatus1	Стан переривання термінального рахунку для каналу 1 до маскування.	–
31:2	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A

3.9.8.1.7 Регістр статусу необробленого переривання за помилкою (Raw Error Interrupt Status Register, DMACRawIntErrorStatus – 0xFFE0 4018)

Регістр DMACRawIntErrorStatus (таблиця 3.18) доступний тільки для читання і відображає стан каналу, який надіслав запит на переривання «помилки» до маскування. Високий рівень біта відображає активний запит на переривання «помилки» до маскування.

Таблиця 3.18 – Опис розрядів регістра DMACRawIntErrorStatus

Біт	Символ	Опис	Початкове значення
0	RawIntErrorStatus0	Стан переривання «помилки» для каналу 0 до маскування.	–
1	RawIntErrorStatus1	Стан переривання «помилки» для каналу 1 до маскування.	–
31:2	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A

3.9.8.1.8 Регістр увімкнення каналу (Enabled Channel Register, DMACEnbldChns – 0xFFE0 401C)

Регістр DMACEnbldChns (таблиця 3.19) доступний тільки для читання і відображає увімкнені канали, відповідно до біта Enable у регістрі DMACCxConfiguration. Високій рівень біта означає увімкнений DMA – канал. Біт скидається за закінченням DMA – передачі.

Таблиця 3.19 – Опис розрядів регістра DMACEnbldChns

Біт	Символ	Опис	Початкове значення
0	EnabledChannels0	Стан каналу 0	0
1	EnabledChannels1	Стан каналу 1	0
31:2	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A

3.9.8.1.9 Регістр запиту програми на пакетну передачу (Software Burst Request Register, DMACSoftBReq – 0xFFE0 4020)

Регістр DMACSoftBReq (таблиця 3.20) доступний для читання/запису та вмикає пакетні запити DMA, що створюються програмним забезпеченням. Запит DMA може бути згенерований для кожного джерела записом 1 до відповідного біта регістра. Біт регістра скидається за завершенням транзакції. Запис 0 до регістра нічого не змінює. Читання регістра показує джерела, які потребують пакетні передачі DMA. Запит може бути надісланий або з периферійного пристрою, або регістра.

Таблиця 3.20 – Опис розрядів регістра DMACSoftBReq

Біт	Символ	Опис	Початкове значення
0	SoftBReqSSP0Tx	Запит програми на пакетну передачу для SSP0 Tx	0
1	SoftBReqSSP0Rx	Запит програми на пакетну передачу для SSP0 Rx	0
2	SoftBReqSSP1Tx	Запит програми на пакетну передачу для SSP1 Tx	0
3	SoftBReqSSP1Rx	Запит програми на пакетну передачу для SSP1 Rx	0

Продовження таблиці 3.20

Біт	Символ	Опис	Початкове значення
4	SoftBReqSDMMC	Запит програми на пакетну передачу для SD/MMC	0
5	–	Зарезервований, запис до біта неможливий.	N/A
6	SoftBReqI2S1	Запит програми на пакетну передачу для I^2S1	0
31:7	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A

3.9.8.1.10 Регістр запиту програми на одиночну передачу (Software Single Request Register, DMACSoftSReq – 0xFFE0 4024)

Регістр DMACSoftSReq (таблиця 3.21) доступний для читання/запису і вмикає одиночні DMA – запити для програм. DMA – запит може бути згенерований для кожного джерела записом 1 до відповідного біта регістра. Біт регістра скидається за завершенням транзакції. Запис 0 до регістра нічого не змінює. Читання регістра повертає джерела, що потребують одиночних DMA передач.

Таблиця 3.21 – Опис розрядів регістра DMACSoftSReq

Біт	Символ	Опис	Початкове значення
0	SoftReqSSP0Tx	Запит програми на одиночну передачу для SSP0 Tx	0
1	SoftReqSSP0Rx	Запит програми на одиночну передачу для SSP0 Rx	0
2	SoftReqSSP1Tx	Запит програми на одиночну передачу для SSP1 Tx	0
3	SoftReqSSP1Rx	Запит програми на одиночну передачу для SSP1 Rx	0
4	SoftReqSDMMC	Запит програми на одиночну передачу для SD/MMC	0
31:5	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A

3.9.8.1.11 Регістр запиту програми останньої пакетної передачі (Software Last Burst Request Register, DMACSoftLBreq – 0xFFE0 4028)

Регістр DMACSoftLBReq (таблиця 3.22) доступний для запису/читання та надає можливість програмі на запит передачі останнього пакету DMA. DMA –

запит може бути згенерований для кожного джерела записом 1 до відповідного біта регістра. Біт регістра скидається за завершенням транзакції. Запис 0 до регістра нічого не змінює. Читання регістра повертає джерела, що потребують останню пакетну DMA – передачу.

Таблиця 3.22 – Опис розрядів регістра DMACSoftLBReq

Біт	Символ	Опис	Початкове значення
3:0	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A
4	SoftLBReqSDMMC	Запит програми на останню пакетну передачу для SD/MMC	0
31:5	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A

3.9.8.1.12 Регістр запиту програми останньої одиночної передачі (Software Last Single Request Register, DMACSoftLSReq – 0xFFE0 402C)

Регістр DMACSoftLSReq (таблиця 3.23) доступний для запису/читання та надає можливість програмі на запит останньої одиночної DMA – передачі. DMA – запит може бути згенерований для кожного джерела записом 1 до відповідного біта регістра. Біт регістра скидається за завершенням транзакції. Запис 0 до регістра нічого не змінює. Читання регістра повертає джерела, що потребують останньої одиночної DMA – передачі.

Таблиця 3.23 – Опис розрядів регістра DMACSoftLSReq

Біт	Символ	Опис	Початкове значення
3:0	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A
4	SoftLSReqSDMMC	Запит програми на останню одиночну передачу для SD/MMC	0
31:5	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A

3.9.8.1.13 Регістр конфігурації (Configuration Register, DMACConfiguration – 0xFFE0 4030)

Регістр DMACConfiguration (таблиця 3.24) доступний для читання/запису та призначений для конфігурації роботи DMA. Порядок байтів AHB – майстра може бути змінений записом відповідного значення до біта M даного регістра.

Таблиця 3.24 – Опис розрядів регістра DMACConfiguration

Біт	Символ		Опис	Початкове значення
0	E		Стан DMA:	0
		0	Відключений. DMA в режимі зниженого споживання енергії	
		1	Увімкнений	
1	M		Порядок байтів AHB – майстра:	0
		0	Режим Little – endian	
		1	Режим Big – endian	
31:2	–	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A

3.9.8.1.14 Регістр синхронізації (Synchronization Register, DMACSync – 0xFFE0 4034)

Регістр DMACSync (таблиця 3.25) доступний для читання/запису і вмикає або відключає логіку синхронізації для сигналів – запитів DMA. DMA сигнали – запити складаються з DMACBREQ [15:0], DMACSREQ [15:0], DMACLBREQ [15:0], та DMACLSREQ [15:0]. Скинутий біт вмикає логіку синхронізації для заданої групи DMA – запитів. Встановлений біт відключає логіку синхронізації для заданої групи DMA – запитів. Якщо регістр скинутий в 0, логіка синхронізації увімкнена.

3.9.8.2 Канальні регістри

Канальні регістри використовуються для програмування двох DMA – каналів. Ці регістри включають:

– два регістри DMACCxSrcAddr;

– два регістри DMACCxDestAddr;

– два регістри DMACCxLLI;

– два регістри DMACCxControl;

– два регістри DMACCxConfiguration, де x = 0/1.

При виконанні scatter/gather – операцій DMA перші 4 регістри оновлюються автоматично

Таблиця 3.25 – Опис розрядів регістра DMACSync

Біт	Символ	Опис	Початкове значення
15:0	DMACSync	Увімкнення або відключення логіки синхронізації DMA. Скинутий біт вмикає логіку синхронізації для DMACBREQ [15:0], DMACSREQ [15:0], DMACLBREQ [15:0], та DMACLSREQ [15:0] сигналів – запитів. Встановлений біт вимикає логіку синхронізації для заданої групи DMA– запитів.	0x0000
1:16	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A

.

3.9.8.2.1 Канальні регістри адреси джерела (Channel Source Address Registers, DMACC0SrcAddr – 0xFFE0 4100 та DMACC1SrcAddr – 0xFFE0 4120)

Два регістри DMACCxSrcAddr (таблиця 3.26) доступні для читання/запису зберігають поточну адресу джерела даних для передачі, яку вирівняно побайтово. Кожен з регістрів програмується напряму за допомогою програми перед тим, як увімкнути канал. Коли DMA – канал увімкнений, цей регістр оновлюється:

– при інкременті адреси джерела,

– при переході за елементами зв’язаного списку, коли завершується передача даних.

Коли канал активний, читання регістра не дасть корисної інформації. Значення може змінитися за завершенням читання регістра програмою. Регістр призначений для читання тільки тоді, коли канал неактивний: значення регістра вказуватиме на адресу останнього прочитаного елемента.

Примітка: Адреси джерела та приймача мають бути вирівняно за розміром джерела та приймача.

Таблиця 3.26 – Опис розрядів регістрів DMACCxSrcAddr

Біт	Символ	Опис	Початкове значення
31:0	SrcAddr	Адреса джерела DMA	0x0000 0000

3.9.8.2.2 Канальні регістри адреси приймача (Channel Destination Address Registers, DMACC0DestAddr – 0xFFE0 4104 та DMACC1DestAddr – 0xFFE0 4124)

Два регістри DMACCxDestAddr (таблиця 3.27) доступні для читання/запису і зберігають поточну адресу приймача даних, яку вирівняно побайтово. Кожен з регістрів програмується напряму за допомогою програми перед тим, як увімкнути канал. Коли DMA – канал увімкнений, регістр оновлюється з інкрементом адреси приймача та при переході за елементами зв'язаного списку, коли завершується передача даних. Коли канал активний, читання регістра не дасть корисної інформації. Значення може змінитися за завершенням читання регістра програмою. Регістр призначений для читання тільки тоді, коли канал неактивний: значення регістра вказуватиме на адресу приймача останнього прочитаного елемента.

Таблиця 3.27 – Опис розрядів регістрів DMACCxDestAddr

Біт	Символ	Опис	Початкове значення
31:0	DestAddr	Адреса приймача DMA	0x0000 0000

3.9.8.2.3 Канальні регістри елемента зв'язаного списку (Channel Linked List Item Registers, DMACC0LLI – 0xFFE0 4108 та DMACC1LLI – 0xFFE0 4128)

Два регістри DMACCxLLI (таблиця 3.28) доступні для читання/запису і зберігають вирівняну за словом адресу наступного елементу зв'язаного списку (Linked List Item, LLI). Якщо LLI дорівнює 0, тоді поточний LLI є останнім у ланцюгу, і DMA – канал відключається за завершенням всіх пов'язаних зі зв'язаним списком DMA – передач.

Примітка: Програмування даного регістра тоді, коли увімкнений DMA – канал, може призвести до невизначених наслідків.

Таблиця 3.28 – Опис розрядів регістрів DMACCxLLI

Біт	Символ	Опис	Початкове значення
0	Reserved	Зарезервований, зчитувати як 0, не змінювати.	N/A
1	R	Зарезервований, записувати як 0, маскований при читанні.	0
31:2	LLI	Елемент зв'язаного списку. Біти [31:2] адреси вказують на наступний LLI. Біти [1:0] адреси 0.	0

3.9.8.2.4 Регістри керування каналом (Channel Control Registers, DMACC0Control – 0xFFE0 410C та DMACC0Control – 0xFFE0 412C)

Два регістри DMACCxControl (таблиця 3.29) доступні для читання/запису і зберігають керуючу інформацію DMA – каналу, таку як розмір передачі, розмір пакету, ширина передачі.

Таблиця 3.29 – Опис розрядів регістрів DMACCxControl

Біт	Символ	Опис	Початкове значення
11:0	TransferSize	Розмір передачі. Запис встановлює розмір передачі у випадку, коли DMA виступає в ролі контролера потоку. Читання даного поля повертає кількість завершених передач у шині приймача. Поле призначене для читання, коли канал відключений.	0
14:12	SBSize	Розмір пакету джерела. Відображає кількість передач, які разом складають запит на пакетну передачу джерела.	0
17:15	DBsize	Розмір пакету приймача. Відображає кількість передач, які разом складають запит на пакетну передачу приймача. Значення має відповідати розміру пакету периферійного пристрою приймача, або, якщо приймачем є пам'ять, граничному розміру пам'яті.	0
20:18	SWidth	Ширина передачі джерела. Передачі ширше за розмір шини AHB – майстра не підтримуються. Ширина джерела та приймача можуть відрізнятися одне від одного. Обладнання автоматично пакує та розпаковує дані так, як необхідно.	0

Продовження таблиці3.29

Біт	Символ	Опис	Початкове значення
23:21	DWidth	Ширина передачі приймача. Передачі ширше за розмір шини AHB – майстра не підтримуються. Ширина джерела та приймача можуть відрізнятися одне від одного. Обладнання автоматично пакує та розпаковує дані так, як необхідно.	0
25:24	–	Зарезервовані, запис до бітів неможливий. Результат читання бітів невизначений.	N/A
26	SI	Інкремент джерела. Якщо встановлений, адреса джерела буде інкрементована після кожної передачі.	0
27	DI	Інкремент приймача. Якщо встановлений, адреса приймача буде інкрементована після кожної передачі.	0
30:28	Prot	Біти захисту та доступу	0
31	TCIE	Біт увімкнення переривання термінального рахунку. Контролює, чи повинний поточний LLI викликати переривання термінального рахунку.	0

Кожен з регістрів програмується напряму за допомогою програми перед тим, як увімкнути канал. Якщо канал увімкнений, регістр оновлюється при переході до наступного елементу зв’язаного списку за завершенням передачі повного пакету даних. Коли канал активний, читання регістра не дасть корисної інформації. Значення може змінитися за завершенням читання регістра програмою. Регістр призначений для читання тільки тоді, коли канал неактивний.

Вплив значення 3 – бітних полів DBSize та SBSize на відповідні розміри пакетів наведено в таблиці 3.30.

Таблиця 3.30 – Вплив значення 3 – бітних полів DBSize та SBSize на відповідні розміри пакетів

Значення в бітах	Розмір пакету
000	1
001	4
010	8
011	16
100	32
101	64
110	128
111	256

Вплив значення 3 – бітних полів SWidth та DWidth на відповідну ширину передач наведено в таблиці 3.31.

Таблиця 3.31 – Вплив значення 3 – бітних полів SWidth та DWidth на відповідну ширину передач

Значення в бітах	Ширина передачі
000	Байт (8 біт)
001	Півслова (16 біт)
010	Слово (32 біт)
011 або 1xx	Зарезервовано

3.9.8.2.5 Захист та доступ до інформації

AHB надає доступ до інформації джерелу та приймачу тоді, коли проходить передача. Інформація про передачу надається при програмуванні DMA – каналу (біт Prot регістра DMACCxControl, а також біт Lock регістра DMACCxConfiguration) (таблиця 3.32). Ці біти програмуються у програмі, інформація може бути використана в разі необхідності.

Таблиця 3.32 – Опис бітів захисту та доступу

Біт DMACCxControl	Значення	Опис	Початкове значення
28		Привілейований або користувацький режим. Даний біт контролює сигнал AHB HPROT[1]. Відображає режим доступу	0
	0	Користувацький режим	
	1	Привілейований режим	
29		Відображає, чи є доступ буферізованим. Може бути використаний для повідомлення AMBA – мосту, що читання може бути завершене без очікувань. Контролює сигнал AHB HPROT[2].	0
	0	Не буферізований	
	1	Буферізований	
30		Відображає, чи є доступ кешованим. Контролює сигнал AHB HPROT[3]. Може бути використаний для повідомлення AMBA – мосту, що якщо зчитано перший пакет розміру 8, міст може передати повний пакет розміру 8 на шину приймача, аніж передаватиме транзакції одну за раз.	0
	0	Не кешований	
	1	Кешований	

3.9.8.2.6 Регістри конфігурації каналу (Channel Configuration Registers, DMACC0Configuration – 0xFFE0 4110 та DMACC1Configuration – 0xFFE0 4130)

Два регістри DMACCxConfiguration (таблиця 3.33) доступні для читання/запису, за виключенням 17 біта, що доступний тільки для читання. Використовується для конфігурації DMA – каналу. Регістри не оновлюються при запиті нового LLI.

Таблиця 3.33 – Опис розрядів регістрів DMACCxConfiguration

Біт	Символ	Значення	Опис	Початкове значення
0	E (Enable)		Стан каналу також може бути зчитаний у регістрі DMACEnbldChns. Канал вмикається встановленням даного біта. Вимикається канал скиданням біта. Це призведе до закінчення поточної передачі та втрати даних у FIFO відповідного каналу. Перезапуск каналу встановленням біта Enable може призвести до неочікуваних результатів. Канал також вимикається тоді, коли досягнуто останній елемент LLI, або якщо виникла помилка.	0
		0	Відключений	
		1	Увімкнений	
4:1	SrcPeripheral		Джерело запитів периферії	0
		0000	SSP0 Tx	
		0001	SSP0 Rx	
		0010	SSP1 Tx	
		0011	SSP1 Rx	
		0100	SD/MMC	
		0101	I2S channel 0	
		0110	I2S channel 1	
		0111 або 1xxx	Зарезервовані значення.	
5	–	–	Зарезервовані значення.	N/A

Продовження таблиці 3.33

Біт	Символ	Значення	Опис	Початкове значення
9:6	DestPeripheral		Ці біти визначають приймач DMA – запитів. Поле ігнорується у випадку, якщо приймачем передачі є пам'ять. Див. опис значень SrcPeripheral.	0
10	–	–	Зарезервовані значення.	N/A
13:11	FlowCntrl		Контроль потоку та тип передачі. В якості контролера потоку можуть виступати: DMA, джерело, приймач. Типи передач наведено в таблиці 3.34.	0
14	IE (Interrupt Error)		Маска переривання «помилки». Якщо біт скинутий, він маскує переривання «помилки» відповідного каналу.	0
15	ITC		Маска переривання термінального рахунку. Якщо біт скинутий, він маскує переривання термінального рахунку відповідного каналу.	0
16	L (Lock)		Блокування. Якщо біт встановлений, він дозволяє захищені передачі.	0
17	A (Active)		Біт може бути використаний для коректного завершення роботи каналу DMA. Запис до біта нічого не змінює.	
		0	У FIFO черзі каналу даних немає	
		1	У FIFO черзі каналу є дані	
18	H (Halt)			
		0	Увімкнути DMA – запити.	
		1	Ігнорувати DMA – запити джерела.	
31:19	–		Зарезервовані значення.	

Вплив 3 – бітних значень поля FlowCntrl на вибір контролера потоку та типу передачі наведено в таблиці 3.34.

Таблиця 3.34 – Вплив значень поля FlowCntrl на вибір контролера потоку та типу передачі

Значення в бтах	Тип передачі	Контролер
000	Пам’ять – пам’ять	DMA
001	Пам’ять – периферійний пристрій	DMA
010	Периферійний пристрій – пам’ять	DMA
011	Периферійний пристрій джерела – периферійний пристрій приймача	DMA
100	Периферійний пристрій джерела – периферійний пристрій приймача	Периферійний пристрій приймача
101	Пам’ять – периферійний пристрій	Периферійний пристрій
110	Периферійний пристрій – пам’ять	Периферійний пристрій
111	Периферійний пристрій джерела – периферійний пристрій приймача	Периферійний пристрій джерела

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Опишіть призначення модуля прямого доступу до пам'яті в мікропроцесорних системах.
2) Як контролер DMA підключається до окремих вузлів мікроконтролера?
3) Скільки модулів DMA має контролер DMA?
4) Як контролер DMA підключається до шини AHB?
5) Які види пересилань даних виконуються контролером DMA?
6) Як в контролері DMA відбувається пересилання даних з пам'яті в пам'ять?
7) На які групи розбито регістри керування контролером DMA?
8) Як здійснюється дозвіл роботи контролера DMA?
9) Які переривання має кожний з модулів DMA?
10) Які регістри призначені для програмування переривань від модулів DMA?
11) Як виконується синхронізація модулів DMA?
12) Як програмується пересилання даних з пам'яті в пам'ять?
13) Як здійснюється пересилання даних в пакетному режимі?
14) Як програмується використання периферійних пристроїв в якості контролера потоку в модулях DMA?
15) Як в модулях DMA підтримується пересилання несуміжних даних?
16) Назвіть регістри, які призначено для керування модулем DMA.
17) Опишіть призначення окремих розрядів регістра статусу переривань.
18) Опишіть призначення окремих розрядів регістрів статусу, скидання та статусу необробленого переривання термінального рахунку.
19) Опишіть призначення окремих розрядів регістрів статусу, скидання та статусу необробленого переривання за помилкою.
20) Опишіть призначення окремих розрядів регістрів запиту програми на окремі види передачі.
21) Опишіть призначення окремих розрядів регістра увімкнення каналу.
22) Опишіть призначення окремих розрядів канальних регістрів.

4 ПЕРИФЕРІЙНІ ПРИСТРОЇ ЗАГАЛЬНОГО ПРИЗНАЧЕННЯ

4.1 Загальні положення

Цей розділ присвячено периферійним пристроям, які є в складі мік–роконтроллеров сімейства [1, 4, 7]. Процес конфігурації кожною пристрою і його використання демонструється на прикладах. Після того, як ви розберетеся з роботою периферійних пристроїв, тексти прикладів можна буде використовувати в якості основи для написання власних драйверів низького рівня.

4.2 Порти введення/виведення загального призначення

4.2.1 Загальні відомості

Мікроконтролери LPC23xx можуть мати до п'яти портів введення/виведення загального призначення GPIO (General Purpose Input Output), кожен з яких містить 32 лінії введення/виведення [1, 4, 7]. Таким чином, в нашому розпорядженні є до 160 виводів. Щоб зберегти сумісність з моделями LPC21xx попереднього сімейства, для керування портами PORT0 та PORT1 використовуються звичайні керуючі регістри, які підключено до шини APB. Однак при використанні цих регістрів зміна стану портів відбувається досить повільно. Тому в мікроконтролерах LPC23xx з’явився другий набір швидких регістрів керування портами введення/виведення (Fast GPIO) (рисунок 4.1), який розміщено на локальній шині.

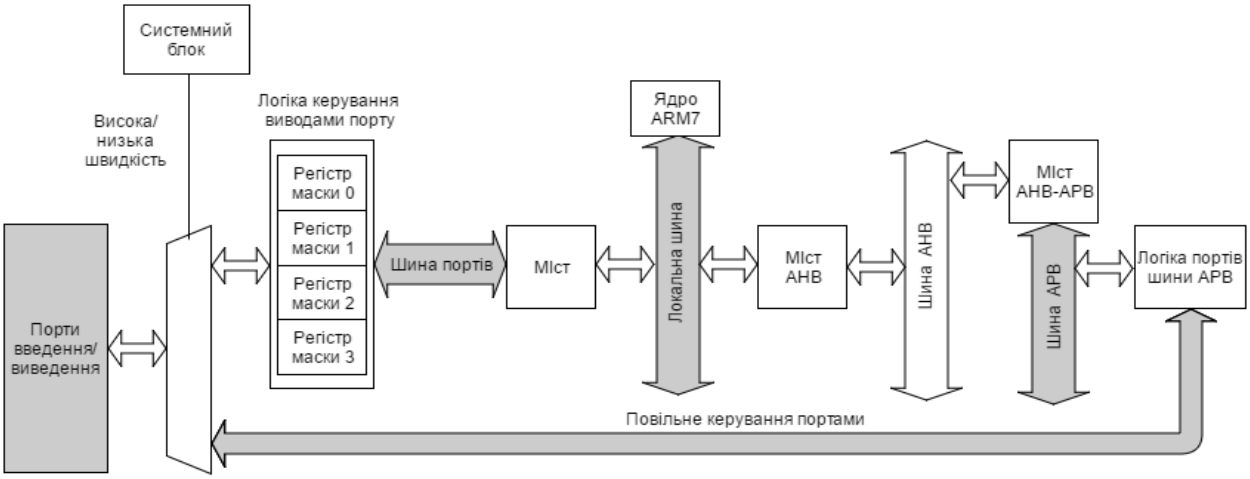


Рисунок 4.1 – Швидкі регістри GPIO

Рекомендується використовувати саме ці регістри та ігнорувати успадковані регістри GPIO у всіх випадках, за винятком налаштування портів існуючого коду. Окрім цього, порти PORT0 та PORT1 можуть генерувати переривання при появі на заданому виводі контролера зростаючого чи спадаючого фронту сигналу.

Контакти пристрою, які не підключені до конкретної периферійної функції контролюються регістрами введення/виведення загального призначення GPIO. Контакти можуть бути динамічно сконфігуровані як входи або виходи. Окремі регістри дозволяють встановлювати або скидати будь–яку кількість виходів одночасно. Мікроконтролери сімейства LPC2377/78 використовують прискорені функції GPIO:

- регістри GPIO переміщують до локальної шини ARM, що забезпечує досить швидке введення/виведення;
- регістри–маски дозволяють обробляти набори бітів порту у якості групи, залишаючи інші біти без змін;
- усі регістри GPIO адресуються напівслівно та побайтово;
- повне значення порту можна записати в одній інструкції.

Окрім цього, будь–який контакт порту P0 або порту P2 може бути запрограмований для генерації переривання за переднім, заднім фронтом або одним з них. Визначення перепаду асинхронне, тому він може працювати, коли зупинений тактовий генератор, наприклад у режимі зниженого енергоспоживання. Включення переривання може бути використано для пробудження чіпу з режиму Power–Down.

Для ліній введення/виведення загального призначення (GPIO) мікроконтролерів сімейства LPC2300 характерні наступні особливості:

- керування напрямом (вхід це чи вихід) кожної лінії індивідуально;
- розділене керування встановленням та скиданням вихідного рівня лінії;
- усі лінії, за замовчуванням, після скидання є лініями GPIO, які налаштовано як входи.

Лінії GPIO можуть використовуватися для наступних додатків:

- введення/виведення загального призначення;

– керування світлодіодами чи іншими індикаторами;
– керування зовнішніми пристроями, які підключено до мікроконтролера;
– опитування цифрових входів.

4.2.2 Швидкі регістри введення/виведення

У мікроконтролерах попереднього сімейства LPC2100 регістри керування GPIO розміщувались на шині APB разом з регістрами усіх інших периферійних пристроїв. Однак, при роботі з цими регістрами дуже багато часу витрачалося на пересилання даних по шинам AHB та APB. Для зміни стану виводу було потрібно 14 тактів, тому під час роботи на частоті 60 МГц максимальна частота переключення виводів порту складала всього 4,3 МГц. В новіших версіях пристроїв LPC213x компанія NXP додала нову групу регістрів, які підключено до локальної шини ARM7 (рисунок 4.1). Ці регістри дозволяють перемикати вивід порту всього за два такти (максимальна частота перемикання – 30 МГц). Окрім швидких регістрів GPIO, було додано ще регістр маски, що спрощує маніпулювання бітами кожного порту.

У сімействі LPC2300 керуючі регістри портів введення/виведення підключені до локальної шини. Таке рішення дозволило значно прискорити керування виводами портів. З міркувань сумісності залишені також «класичні» регістри, за допомогою яких можна керувати портами PORT0 і PORT1.

При скиданні блок керування виводами конфігурує усі виводи периферійних пристроїв як входи портів введення/виведення загального призначення (GPIO). Керування виводами GPIO здійснюється чотирма регістрами (рисунок 4.2).

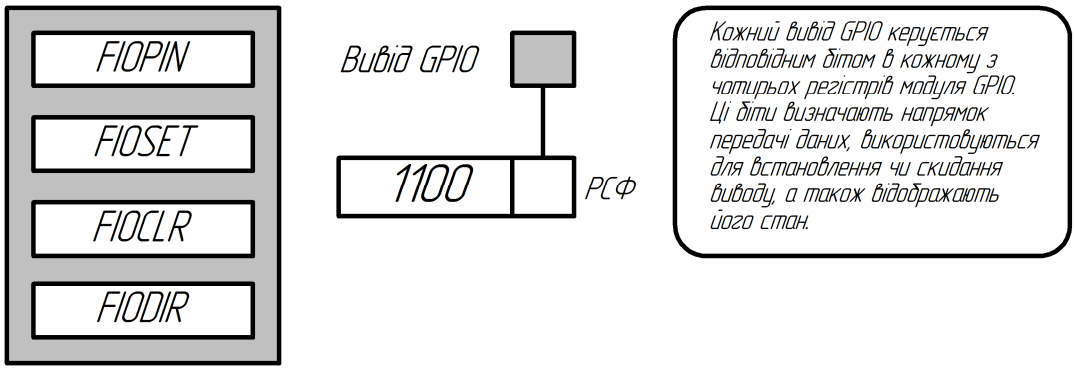


Рисунок 4.2 – Керування виводами GPIO

Біти регістра FIODIR дозволяють індивідуально конфігурувати кожний вивід у якості входу (0) чи виходу (1). Якщо вивід є виходом, його станом можна керувати за допомогою регістрів FIOSET та FIOCLR. Запис 1 в біти цих регістрів приводить до встановлення (FIOSET) чи скидання (FIOCLR) відповідного виводу. Стан виводу GPIO можна визначити в будь–який момент. Для цього треба прочитати вміст регістра FIOPIN. Регістр FIOMASK використовується для маскування окремих бітів регістрів FIOSET, FIOCLR та FIOPIN. Якщо біт регістра маски скинути в 0, то відповідні біти регістрів FIOSET, FIOCLR та FIOPIN можуть бути зчитані або доступні для запису. Якщо встановлений біт регістра маски – 1, то запис у відповідні біти регістрів неможливий, а читання з них не відображатиметься у регістрі FIOPIN. Таке маскування допомагає прискорити маніпулювання лініями введення/виведення на низькому рівні.

Нижче наведена програма, яка викликає миготіння світлодіоду.

```
#include <LPC23xx.h>

void main()
{
int flasher;
int delay;
FIO2DIR = 0x000000FF;
FIO2MASK = 0xFFFFFF00;
while(1)
    {
    FIO2CLR = 0x000000FF;

    for(flasher =1;flasher<0x00000100;flasher = flasher <<1)
    {
       FIO2SET = flasher; //FIO2PIN = flasher;
        for(delay = 0;delay<0xA0000;delay++) //simple     delay
loop
        {
          ;
        }
    }
    }
//return (0);
}
```

В таблиці 4.1 наведено опис виводів портів введення/виведення.

Таблиця 4.1 – Опис виводів портів введення/виведення

Найменування	Тип	Опис
P0.0…P0.31 P1.0…P1.31 P2.0…P2.31 P3.0…P3.31 P4.0…P4.31	Введення/виведення	Входи/виходи загального призначення (GPIO)

4.2.3 Опис керуючих регістрів

Для програмування GPIO використовуються 4 регістри (таблиця 4.2).

Таблиця 4.2 – Керуючі регістри GPIO

Адреса	Ім'я	Опис виводів	Доступ
0xE0028000	IOPIN	**Регістр значення GPIO порту 0.** Поточний стан виводів завжди може бути прочитаний через цей регістр незалежно від режиму та напрямку виводу.	R/W
0xE0028004	IOSET	**Регістр встановлення GPIO порту 0.** Цей регістр керує станом вихідних виводів разом з регістром IOCLR. Запис логічної одиниці у будь–який біт приведе до появи потенціалу високого рівня на відповідному виводі. Запис логічного нуля ефекту не має.	R/W
0xE0028008	IODIR	**Регістр напрямку GPIO порту 0.** Цей регістр задає напрямок роботи (введення чи виведення) індивідуально для кожного виводу.	R/W
0xE002800C	IOCLR	**Регістр скидання GPIO порту 0.** Цей регістр керує станом вихідних виводів разом з регістром IOSET. Запис логічної одиниці у будь–який біт приведе до появи потенціалу низького рівня на відповідному виводі та скидання цього біта в регістрі IOSET. Запис логічного нуля ефекту не має.	W

4.2.4 Регістр значення виводів GPIO

Цей регістр відображає значення сигналів, які подано на вхідні виводи. Значення регістра відображає будь–які зміни зовнішніх сигналів, але тільки на тих виводах мікроконтролера, які були сконфігуровані як лінії GPIO. Для виводів портів, які сконфігуровано не як лінії GPIO (а, наприклад, для виконання інших функцій), контроль з використанням регістра IOPIN не дасть достовірного результату.

Вибір для виводу порту будь–якої однієї функції повністю виключає всі інші функції, які можуть бути доступні на тому ж самому виводі. Часткове виключення з цього правила мають лише входи АЦП. Незалежно від функції, яка вибрана для виводу порту, до якого підключений вхід АЦП, цей вхід АЦП у будь–який момент може бути прочитаний мікроконтролером, та зміна вхідної напруги на ньому буде відображена в результаті аналого–цифрового перетворення.

Однак, коректне читання результату АЦП може бути виконано лише у випадку, коли для даного виводу обрана аналогова функція (при цьому буде активована відповідна схема інтерфейсу між виводом та модулем АЦП). У всіх інших випадках, підключення до даного виводу цифрових логічних схем, необхідних для активізації якоїсь іншої функції, приведе до некоректного проведення аналого–цифрового перетворення.

Зауваження: під час запису у цей регістр, значення передається у вихідний регістр, минаючи IOSET та IOCLR. Потрібно уникати використання цієї особливості через те, що при цьому немає можливості індивідуально керувати виводами.

Таблиця 4.3–Опис та адреси регістрів IOPIN (IO0PIN – 0xE0028000, IO1PIN – 0xE0028010, IO2PIN – 0xE0028020, IO3PIN – 0xE0028030)

IOPIN	Опис	Після скидання
31:0	Значення виводів GPIO. Біт 0 відповідає P0.0, …, біт 31 відповідає P0.31	Не визначено

4.2.5 Регістр встановлення GPIO

Цей регістр використовується для подання на вихідні виводи GPIO сигналу високого рівня. Запис логічної 1 у відповідний біт приведе до появи на відповідному виводі потенціалу високого рівня. Запис логічного 0 не змінить значення біта та потенціалу на виході. Якщо вивід налаштований як вхід чи виконує альтернативну функцію, то запис 1 в його біт не дасть ніякого ефекту.

Читання з IOSET поверне значення вихідного регістра GPIO, значення у якому сформовано попередніми записами в IOSET та IOCLR (чи IOPIN, що дуже не бажано). Значення цього регістра не відображає стан входів.

Таблиця 4.4 – Опис та адреси регістрів IOSET (IO0SET – 0xE0028004, IO1SET – 0xE0028014, IO2SET – 0xE0028024, IO3SET – 0xE0028034)

IOSET	Опис	Після скидання
31:0	Значення встановлює біти GPIO. Біт 0 відповідає P0.0, …, біт 31 відповідає P0.31	0

4.2.6 Регістр скидання GPIO

Цей регістр призначено для подання на вихідні виводи GPIO сигналу низького рівня. Запис логічної 1 у відповідний біт приведе до появи на відповідному виводі потенціалу низького рівня та очищення цього біта в регістрі IOSET. Запис логічного 0 не змінить значення біта на виході. Якщо вивід налаштований як вхід чи виконує альтернативну функцію, то запис 1 в його біт не дасть ніякого ефекту. Значення цього регістра не відображає стан входів.

Таблиця 4.5 – Опис та адреси регістрів IOCLR (IO0CLR – 0xE002800C, IO1CLR – 0xE002801C, IO2CLR – 0xE002802C, IO3CLR – 0xE002803C)

IOCLR	Опис	Після скидання
31:0	Значення скидає біти GPIO. Біт 0 відповідає P0.0, …, біт 31 відповідає P0.31	0

4.2.7 Регістр напрямку GPIO

Цей регістр використовується для керування напрямком роботи виводів, які використовують GPIO. Значення біта напрямку для того чи іншого виводу задає напрямок передавання даних через цей вивід.

Таблиця 4.6 – Опис та адреси регістрів IODIR (IO0DIR – 0xE0028008, IO1DIR – 0xE0028018, IO2DIR – 0xE0028028, IO3DIR – 0xE0028038)

IODIR	Опис	Після скидання
31:0	Задає напрямок виводів GPIO (0 – вхід, 1– вихід). Біт 0 відповідає P0.0, …, біт 31 відповідає P0.31	0

4.2.8 Особливості програмування GPIO

Якщо для виводу відповідний біт було встановлено і в регістрі IOSET і в регістрі IOCLR, то рівень на виході встановиться у відповідності з останнім записом. Так, при послідовності:

IO0SET=0x00000080,

IO0CLR=0x00000080,

на виході P0.7 буде низький рівень, тому що запис у IO0CLR був після запису в IO0SET.

Додатки, які потребують одночасної зміни всіх ліній паралельного порту, можуть використовувати прямий доступ до відповідного регістра значень виводів порту GPIO (IOPIN). Припустимо, що якщо виводи P0.8 та P0.15 сконфігуровано як виходи, запис в регістр IO0PIN:

IO0PIN=0x0000C700

дасть той самий ефект, що і послідовність команд

IO0SET=0x0000C700,

IO0CLR=0x00003800.

Варіант з записом в регістри IO0SET та IO0CLR буде виконуватись довше у порівнянні з єдиною командою запису в регістр IO0PIN.

4.2.9 Переривання від GPIO

Окрім роботи за своїм основним призначенням, порти GPIO0 та GPIO2 можуть генерувати переривання при зміні стану будь–якого з виводів порту.

Два регістри дозволу переривання (на кожний порт) дозволяють генерувати переривання за зростаючим чи спадаючим фронтом індивідуально для кожного з виводів порту (рисунок 4.3).

Обидві лінії переривань від портів введення/виведення з'єднані в одну лінію EINT3, яка під'єднана до виділеного слоту модуля VIC. При генерації переривання у обробнику потрібно перевірити вміст регістрів статусу, щоб визначити, який з виводів згенерував переривання і, якщо це потрібно, який фронт було зафіксовано – зростаючий чи спадаючий. Як і для інших переривань від периферійних пристроїв, перед виходом з обробника потрібно скинути прапорці переривання.

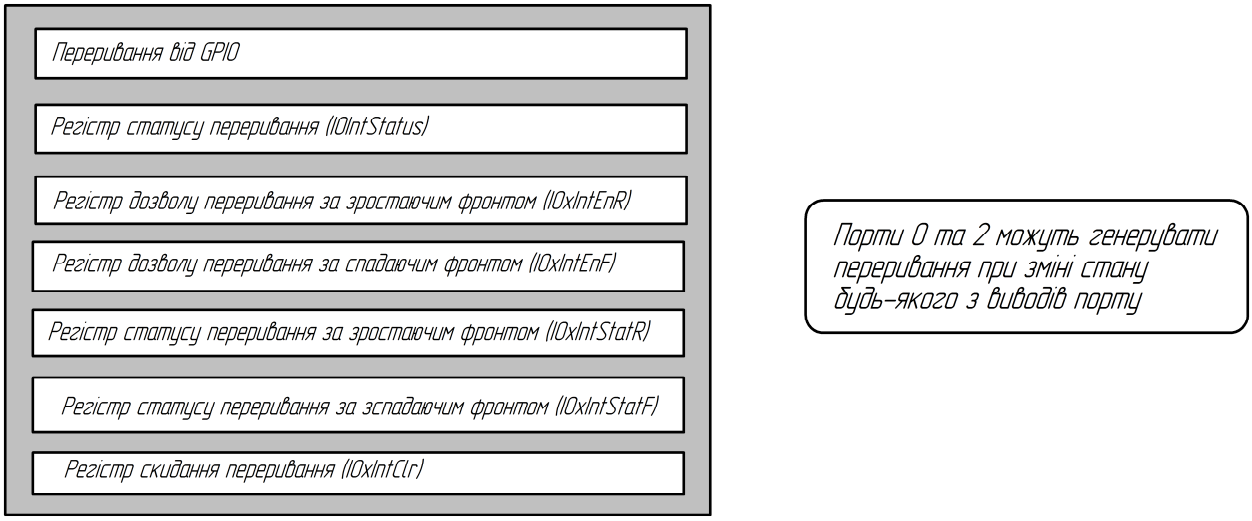


Рисунок 4.3 – Регістри керування перериваннями GPIO

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Скільки портів введення/виведення мають мікроконтролери сім`ї LPC23xx?

2) Скільки ліній введення/виведення має кожен порт?

3) Які керуючі регістри використовуються для керування портами PORT0 та PORT1 та до якої шини їх підключено?

4) Який другий набір регістрів керування портами введення/виведення з'явився в мікроконтролерах LPC23xx та на якій шині його розміщено?

5) Який фронт сигналу на заданому виводі портів PORT0 та PORT2 може генерувати переривання?

6) Які прискорені функції GPIO використовують мікроконтролери сімейства?

7) Які особливості характерні для ліній введення/виведення загального призначення (GPIO) мікроконтролерів сімейства LPC2300?

8) Наведіть приклади додатків, для яких можуть використовуватися лінії GPIO?

9) З якою частотою можуть перемикати вивід порту швидкі регістри?

10) Назвіть регістри, які можуть використовуватись для керування виводами GPIO.

11) Який регістр призначено для програмування кожного виводу у якості входу чи виходу?

12) За допомогою яких регістрів можна керувати станом виводу, який є виходом?

13) Який регістр призначено для визначення стану виводу?

14) Для чого може використовуватися регістр FIOMASK?

15) Назвіть регістри, які можуть використовуються для програмування GPIO.

16) Дайте опис бітів регістрів IOPIN.

17) Дайте опис бітів регістрів встановлення та скидання.

18) Дайте опис бітів регістрів напрямку.

19) Наведіть особливості програмування GPIO.

20) Наведіть особливості генерації переривань від GPIO.

4.3 Таймери загального призначення

4.3.1 Загальна характеристика

У складі мікроконтролерів сімейства LPC2300 є чотири таймери загального призначення [1, 4, 7]. Всі ці таймери мають однакову структуру і функціональні можливості. Таймери побудовані на базі 32–бітного лічильника, який об'єднано з 32–бітним переддільником (рисунок 4.4). В якості тактового сигналу у всіх таймерах використовується тактовий сигнал шини APB (Pclk).

Рисунок 4.4 – Спрощена структура таймера загального призначення

Швидкість лічби таймера залежить від значення, що знаходиться в регістрі переддільника. Лічильний регістр переддільника буде інкрементуватися за кожним імпульсом сигналу Pclk до тих пір, поки не досягне значення, що знаходиться в регістрі переддільника. У цей момент вміст лічильного регістра таймера збільшується на 1, лічильний регістр переддільника скидається в нуль і знову починається лічба. Регістр керування таймером містить всього два біти, які використовуються для включення/відключення таймера і його скидання.

4.3.2 Режим захоплення

Поряд з основним лічильником кожен таймер має до чотирьох каналів захоплення (capture). Вони дозволяють запам'ятовувати значення лічильного регістра таймера за зміною зовнішнього сигналу (рисунок 4.5).

Кожному каналу захоплення відповідає один з виводів мікроконтролера, використання якого може бути дозволено блоком керування виводів. Вміст регістра керування каналом захоплення визначає, поява якого із фронтів сигналу (наростаючого, спадаючого або обох) ініціює захоплення.

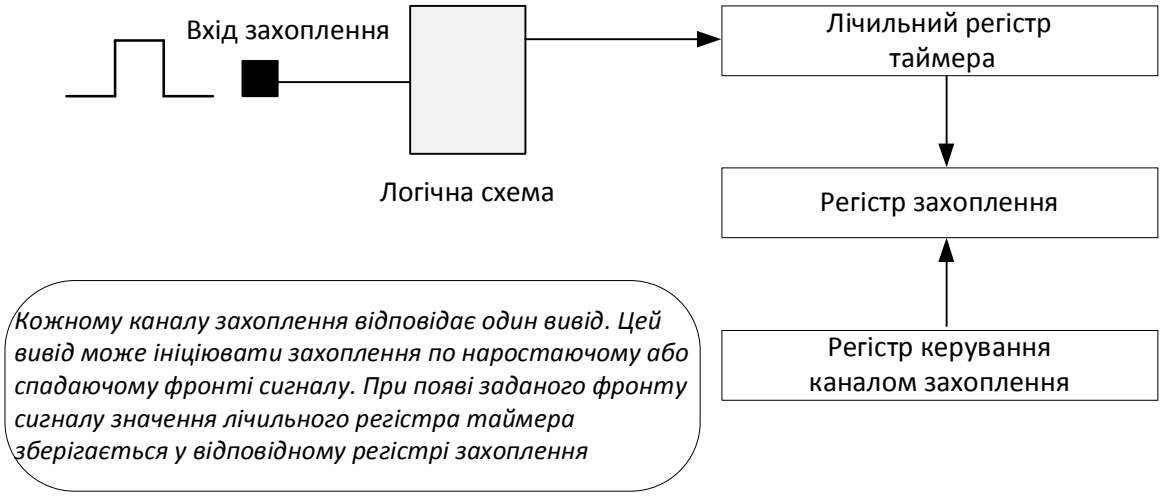


Рисунок 4.5–Режим захоплення

При настанні цієї події поточне значення лічильного регістра таймера переписується до відповідного регістра захоплення і, якщо це необхідно, генерується переривання.

4.3.3 Режим лічильника зовнішніх подій

Регістр керування лічбою визначає, яким чином буде використовуватися таймер – як лічильник зовнішніх подій, або як звичайний таймер. За допомогою даного регістра можна змінити тактовий сигнал таймера з Pclk на сигнал від зовнішнього джерела, що подається на вхід захоплення даного таймера. При цьому лічильний регістр таймера може інкрементуватися за наростаючим, спадаючим або навіть за обома фронтами зовнішнього сигналу.

4.3.4 Режим збігу

Кожен таймер має до чотирьох каналів збігу (співпадіння, match). З кожним з цих каналів пов'язаний регістр збігу, що містить 32–бітне число. Лічильний регістр таймера безперервно порівнюється з вмістом даного регістра (рисунок 4.6). При співпадінні їх значень виникає подія «співпадіння», яка може впливати на таймер (скинути, зупинити або згенерувати переривання), а також

може впливати на стан певного виводу мікроконтролера (встановлення, скидання або перемикання).

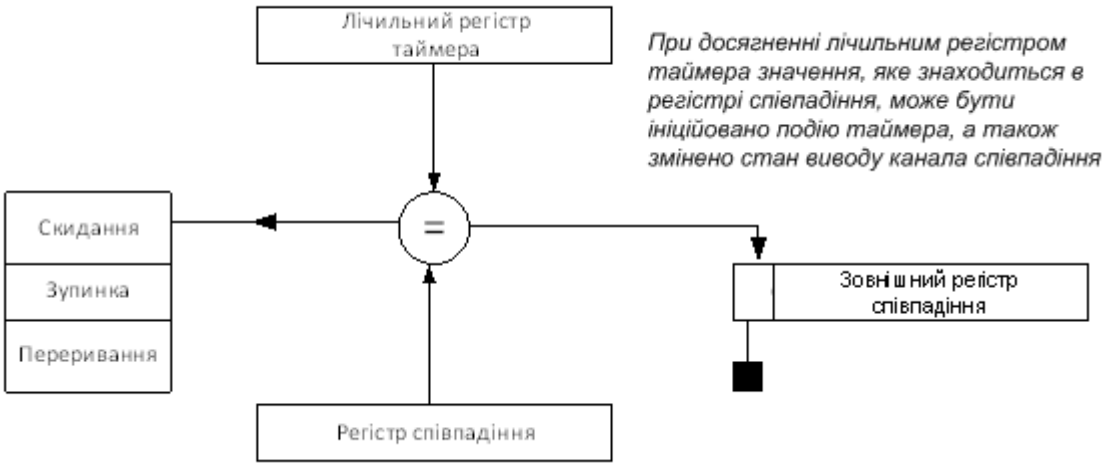


Рисунок 4.6 – Режим збігу

Щоб скористатися цією можливістю таймера, треба завантажити в регістр співпадіння потрібне значення. Подальша конфігурація блоку співпадіння здійснюється за допомогою регістра керування блоком співпадіння (Match Control Register). У цьому регістрі кожному каналу відповідає набір бітів, що використовуються для дозволу виконання наступних дій за подією збігу: генерація переривання від таймера, скидання таймера, зупинка таймера. Причому допускається будь–яка комбінація зазначених дій. Крім того, кожному каналу співпадіння відповідає певний вивід мікроконтролера, стан якого може змінюватися при співпадінні. Як і в разі каналів захоплення, для використання зазначеної можливості спочатку потрібно за допомогою блоку керування виводами підключити вивід до каналу співпадіння. Стан цих виводів визначається станом чотирьох молодших бітів зовнішнього регістра співпадіння EMR (рисунок 4.7).

Регістр EMR містить 4–бітове поле, загальне для всіх каналів. Стан бітів цього поля визначає поведінку виводів збігу при виникненні події. Крім того, в регістрі є біти, змінюючи які можна безпосередньо змінювати стан виводу кожного з каналів.

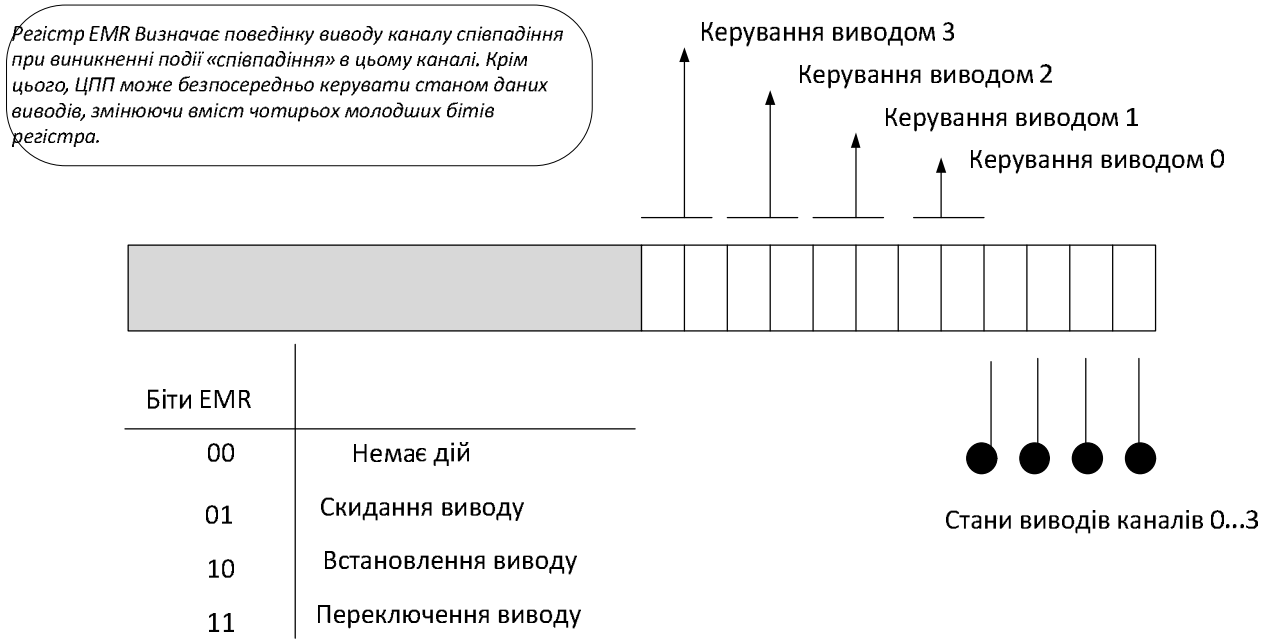


Рисунок 4.7 – Зовнішній регістр збігу

4.3.5 Опис керуючих регістрів модуля таймерів

Кожен таймер/лічильник містить керуючі регістри, які показано в таблиці 4.7 ("значення скидання" відноситься до даних, що зберігаються тільки у використовуваних біта та не включає зарезервований зміст бітів). Більш докладний опис регістрів наводиться нижче.

Таблиця 4.7 – Керуючі регістри таймерів/лічильників 0…3

Біти	Опис	Доступ	Значення скидання	Регістр TIMERn/ ім’я та адреса
IR	**Регістр переривань.** IR може бути записаний, щоб очистити переривання. IR може бути прочитаний, щоб визначити, які з восьми можливих джерел переривань знаходяться в очікуванні.	R/W	0	T0IR – 0xE000 4000 T1IR – 0xE000 8000 T2IR – 0xE007 0000 T3IR – 0xE007 4000

Продовження таблиці 4.7

Біти	Опис	Доступ	Значення скидання	Регістр TIMERn/ ім'я та адреса
TCR	**Регістр керування таймером.** TCR використовується для керування функціями таймера/лічильника. Таймер/лічильник може бути відключений або скинутий через TCR.	R/W	0	T0TCR – 0xE000 4004 T1TCR – 0xE000 8004 T2TCR – 0xE007 0004 T3TCR – 0xE007 4004
TC	**Таймер/лічильник.** 32–х бітний TC збільшується кожні PR + 1 циклів P_{CLK}. TC керуються через TCR.	R/W	0	T0TC – 0xE000 4008 T1TC – 0xE000 8008 T2TC – 0xE007 0008 T3TC – 0xE007 4008
PR	**Регістр масштабування.** Якщо лічильник масштабування (нижче) дорівнює цьому значенню, то в наступному такті інкрементується TC та очищається PC.	R/W	0	T0PR – 0xE000 400C T1PR – 0xE000 800C T2PR – 0xE007 000C T3PR – 0xE007 400C
PC	**Лічильник масштабування.** 32–х бітний PC являє собою лічильник, який збільшується до значення, що зберігається в PR. Коли досягається значення в PR, TC збільшується, а PC буде очищений.	R/W	0	T0PC – 0xE000 4010 T1PC – 0xE000 8010 T2PC – 0xE007 0010 T3PC – 0xE007 4010
MCR	**Регістр керування збігом.** MCR використовується для генерації переривання, скидання PC, зупинки PC та TC при виявленні збігу.	R/W	0	T0MCR – 0xE000 4014 T1MCR – 0xE000 8014 T2MCR – 0xE007 0014 T3MCR – 0xE007 4014
MR0	**Регістр збігу 0.** Значення регістра збігу безперервно порівнюється зі значенням таймера/лічильника. Коли два значення рівні, то це через MCR може скидати TC, зупиняти TC і PC, і/або генерувати переривання.	R/W	0	T0MR0 – 0xE000 4018 T1MR0 – 0xE000 8018 T2MR0 – 0xE007 0018 T3MR0 – 0xE007 4018

Продовження таблиці 4.7

Біти	Опис	Доступ	Значення скидання	Регістр TIMERn/ ім'я та адреса
MR1	**Регістр збігу 1.** Дивись опис MR0.	R/W	0	T0MR1 – 0xE000 401C T1MR1 – 0xE000 801C T2MR1 – 0xE007 001C T3MR1 – 0xE007 401C
MR2	**Регістр збігу 2.** Дивись опис MR0.	R/W	0	T0MR2 – 0xE000 4020 T1MR2 – 0xE000 8020 T2MR2 – 0xE007 0020 T3MR2 – 0xE007 4020
MR3	**Регістр збігу 3.** Дивись опис MR0.	R/W	0	T0MR3 – 0xE000 4024 T1MR3 – 0xE000 8024 T2MR3 – 0xE007 0024 T3MR3 – 0xE007 4024
CCR	**Регістр керування захопленням.** CCR програмує сигнали на входах захоплення, які використовуються для завантаження регістрів захоплення, а також генерацію переривання, коли подія захоплення відбувається.	R/W	0	T0CCR – 0xE000 4028 T1CCR – 0xE000 8028 T2CCR – 0xE007 0028 T3CCR – 0xE007 4028
CR0	**Регістр захоплення 0.** CR0 завантажується значенням TC, коли відбувається подія захоплення на вході CAPn.0 (CAP0.0, CAP1.0, CAP2.0 або CAP3.0).	RO	0	T0CR0 – 0xE000 402C T1CR0 – 0xE000 802C T2CR0 – 0xE007 002C T3CR0 – 0xE007 402C

Продовження таблиці 4.7

Біти	Опис	Доступ	Значення скидання	Регістр TIMERn/ ім'я та адреса
CR1	**Регістр захоплення 1.** CR1 завантажується значенням TC, коли відбувається подія захоплення на вході CAPn.0 (CAP1.0, CAP1.1, CAP2.1 або CAP3.1).	RO	0	T0CR1 – 0xE000 4030 T1CR1 – 0xE000 8030 T2CR1 – 0xE007 0030 T3CR1 – 0xE007 4030
EMR	**Зовнішній регістр збігу.** EMR керує сигналами на зовнішніх виводах MATn.0…3 при збігу (n відповідає таймерам від 0 до 3).	R/W	0	T0EMR – 0xE000 403C T1EMR – 0xE000 803C T2EMR – 0xE007 003C T3EMR – 0xE007 403C
CTCR	**Регістр керування підрахунком.** CTCR вибирає між режимом таймера і режимом лічильника зовнішніх подій, та в режимі лічильника вибирає активний рівень сигналу для підрахунку.	R/W	0	T0CTCR – 0xE000 4070 T1CTCR – 0xE000 8070 T2CTCR – 0xE007 0070 T3CTCR – 0xE007 4070

4.3.6 Регістри переривань IR

Регістри переривань мають адреси, які наведено у таблиці 4.8, та складаються з прапорців переривань при настанні подій збігу та захоплення. Якщо переривання генерується, то відповідний біт в IR буде мати одиничне значення. В іншому випадку, буде мати нульове значення. Записом логічної до відповідного IR–біта останній буде скинутий.

Таблиця 4.8 – Опис бітів регістрів переривань

Біт	Символ	Опис	Значення скидання
0	MR0 переривання	Прапорець переривання збігу для каналу 0	0
1	MR1 переривання	Прапорець переривання збігу для каналу 1	0
2	MR2 переривання	Прапорець переривання збігу для каналу 2	0
3	MR3 переривання	Прапорець переривання збігу для каналу 3	0
4	CR0 переривання	Прапорець переривання події захоплення для каналу 0	0
5	CR1 переривання	Прапорець переривання події захоплення для каналу 1	0
6	–	Зарезервований	0
7	–	Зарезервований	0

4.3.7 Регістри керування таймерами T[0/1/2/3]CR

Регістри керування таймерами T[0/1/2/3]CR мають адреси: 0xE000 4004, 0xE000 8004, 0xE007 0004, 0xE007 4004) та використовується для керування роботою таймерів/лічильників.

Нижче наведено опис бітів регістрів керування таймерами T[0/1/2/3]CR (таблиця 4.9).

Таблиця 4.9 – Опис бітів регістрів керування таймерами

Біт	Символ	Опис	Значення скидання
0	Дозвіл лічильника	Коли одиниця, таймер/лічильник і лічильник масштабування включені для підрахунку. При нулі, лічильники відключені.	0

Продовження таблиці 4.9

Біт	Символ	Опис	Значення скидання
1	Скидання лічильника	Коли одиниця, таймер/лічильник і лічильник масштабування синхронно скидаються на наступній додатній границі P_{CLK}. Лічильники залишаються скинутими поки TCR [1] не повернеться до нуля.	0
7:2	–	Зарезервовані. Користувацькі програми не повинні записувати до зарезервованих бітів. Значення зчитане із зарезервованого біта не визначене.	NA

4.3.8 Регістри керування підрахунком

Регістри керування підрахунком T[0/1/2/3]CTCR мають адреси: 0xE000 4070, 0xE000 8070, 0xE007 0070, 0xE007 4070) та вибирають між режимом таймера і режимом лічильника зовнішніх подій, та в режимі лічильника вибирають активний рівень сигналу для підрахунку .

Нижче наведено опис бітів регістрів керування підрахунком CTCR (таблиця 4.10).

4.3.9 Регістри таймерів/лічильників T0TC … T3TC

32–розрядні регістри таймерів/лічильників T0TC … T3TC мають адреси: 0xE000 4008, 0xE000 8008, 0xE007 0008, 0xE007 4008 та збільшуються, коли лічильник переддільника досягає кінцевого значення. Якщо немає скидання до досягнення свого верхньої межі, таймер/лічильник буде підраховувати через значення 0xFFFF FFFF, а потім повернеться назад до значення 0x0000 0000. Ця подія не викликає переривання, але регістр збігу може бути використаний для виявлення переповнення при необхідності.

Таблиця 4.10 – Опис бітів регістрів керування підрахунком T[0/1/2/3]CTCR

Біт	Символ	Значення	Опис	Значення скидання
1:0	Режим лічильника/таймера		Це поле вибирає який фронт P_{CLK} може збільшувати лічильник масштабування (PC), або коли очищається PC та інкрементується таймер/лічильник (TC). Режим таймера: TC збільшується, коли лічильник масштабування збігається з регістром масштабування.	00
		00	Режим таймера: кожен наростаючий фронт P_{CLK}	
		01	Режим лічильника: TC збільшується за наростаючим фронтом сигналу на вході CAP, який обрано бітами 3:2.	
		10	Режим лічильника: TC збільшується за спадаючим фронтом сигналу на вході CAP, який обрано бітами 3:2.	
		11	Режим лічильника: TC збільшується за обома фронтами сигналу на вході CAP, який обрано бітами 3:2.	
3:2	Вибір входу для підрахунку		Коли біти 1:0 в цьому регістрі не дорівнюють 00, ці біти визначають, який вивід CAP обирається для підрахунку:	00
		00	CAPn.0 для TIMERn	
		01	CAPn.1 для TIMERn Примітка: Якщо режим лічильника обраний для конкретного входу CAPn в TnCTCR, 3 біти для цього входу в регістрі керування захопленням (TnCCR) повинні бути запрограмовані як 000. Однак, захоплення і/або переривання можуть бути обрані для іншого входу CAPn в цьому таймері.	
		10	Зарезервований	
		11	Зарезервований	
7:4	–	–	Зарезервовані. Користувацькі програми не повинні записувати до зарезервованих бітів. Значення зчитане із зарезервованого біта не визначене.	NA

4.3.10 Регістри масштабування T0PR … T3PR

32–розрядні регістри масштабування T0PR … T3PR мають адреси: , 0xE000 400C, 0xE000 800C, 0xE007 000C, 0xE007 400C та визначають максимальне значення для лічильника масштабування.

4.3.11 Регістри лічильників масштабування T0PC … T3PC

32–розрядні лічильники масштабування T0PC…T3PC мають адреси: 0xE000 4010, 0xE000 8010, 0xE007 0010, 0xE007 4010 та керують зміною коефіцієнта ділення частоти P_{CLK}, перш ніж вона застосовується до таймера/лічильника. Це дозволяє керувати роздільною здатністю таймера в порівнянні з максимальним часом до переповнення таймера. Лічильник масштабування збільшується при кожному такті P_{CLK}. Коли він досягає значення, що зберігається в регістрі масштабування, таймер/лічильник збільшується на одиницю і лічильник масштабування скидається при наступному такті P_{CLK}. Це призводить до того, що таймер/лічильник інкрементується за кожним імпульсом P_{CLK} при PR = 0, інкрементується кожні 2 такти P_{CLK} при PR = 1 і т.д.

4.3.12 Регістри збігу MR0 … MR3

Регістри збігу мають адреси, які наведено в таблиці 4.7. Значення відповідного регістра збігу безперервно порівнюється зі значенням таймера/лічильника. Коли ці два значення рівні, то відбуваються дії, які керуються значеннями в регістрі MCR. Можливі дії: генерація переривання, скидання таймера/лічильника або зупинки таймера.

4.3.13 Регістри керування збігом T[0/1/2/3]MCR

Регістри керування збігом мають адреси: 0xE000 4014, 0xE000 8014, 0xE007 0014, 0xE007 4014 та використовується для керування відповідною операцією, коли значення одного з регістрів збігу відповідає значенню таймера/лічильника. Опис кожного з бітів регістрів наведено в таблиці 4.11.

Таблиця 4.11 – Опис бітів регістрів керування збігом T[0/1/2/3]MCR

Біт	Символ	Значення	Опис	Значення скидання
0	MR0I	1	Переривання за MR0: переривання генерується, коли MR0 збігається зі значенням в TC.	0
		0	Це переривання заборонено.	
1	MR0R	1	Скидання за MR0: TC буде скинуто, якщо MR0 збігається з ним.	0
		0	Функція відключена.	
2	MR0S	1	Зупинка за MR0: TC і PC будуть зупинені і TCR[0] буде скинуто в 0, якщо MR0 збігається з TC.	0
		0	Функція відключена.	
3	MR1I	1	Переривання за MR1: переривання генерується, коли MR1 збігається зі значенням в TC.	0
		0	Це переривання заборонено.	
4	MR1R	1	Скидання за MR1: TC буде скинуто, якщо MR1 збігається з ним.	0
		0	Функція відключена.	
5	MR1S	1	Зупинка за MR1: TC і PC будуть зупинені і TCR[0] буде скинуто в 0, якщо MR1 збігається з TC.	0
		0	Функція відключена.	

Продовження таблиці 4.11

Біт	Символ	Значення	Опис	Значення скидання
6	MR2I	1	Переривання за MR2: переривання генерується, коли MR2 збігається зі значенням в TC.	0
		0	Це переривання заборонено.	
7	MR2R	1	Скидання за MR2: TC буде скинуто, якщо MR2 збігається з ним.	0
		0	Функція відключена.	
8	MR2S	1	Зупинка за MR2: TC і PC будуть зупинені і TCR[0] буде скинуто в 0, якщо MR2 збігається з TC.	0
		0	Функція відключена.	
9	MR3I	1	Переривання за MR3: переривання генерується, коли MR3 збігається зі значенням в TC.	0
		0	Це переривання заборонено.	
10	MR3R	1	Скидання за MR3: TC буде скинуто, якщо MR3 збігається з ним.	0
		0	Функція відключена.	
11	MR3S	1	Зупинка за MR3: TC і PC будуть зупинені і TCR[0] буде скинуто в 0, якщо MR3 збігається з TC.	0
		0	Функція відключена.	
15:12	–		Зарезервовані. Користувацькі програми не повинні записувати до зарезервованих бітів. Значення зчитане із зарезервованого біта не визначене.	NA

4.3.14 Регістри захоплення CR0 та CR1

Кожен регістр захоплення CR0 та CR1 має адреси, які наведено в таблиці 4.7, та пов'язаний із відповідним виводом пристрою і може бути завантажений значенням із таймера/лічильника, коли відбувається певна подія на цьому виводі. Налаштування в регістрі керування захопленням визначає, чи включена функція захоплення, і подія захоплення відбувається за переднім фронтом, заднім фронтом або за обома фронтами сигналу на відповідному виводі.

4.3.15 Регістри керування захопленням T[0/1/2/3]CCR

Регістри керування захоплення T[0/1/2/3]CCR мають адреси: 0xE000 4028, 0xE000 8028, 0xE007 0028, 0xE007 4028 та використовується для керування одного з двох регістрів захоплення, які завантажуються значенням із таймера/лічильника, коли відбувається подія захоплення і генерується переривання. Встановлення обох бітів для зростаючого та спадаючого фронтів одночасно є дійсною конфігурацією, в результаті чого подія захоплення відбувається для обох фронтів. У наведеному нижче описі, "n" представляє номер таймера, від 0 до 3.

Примітка: Якщо режим лічильника обраний для конкретного входу CAP в CTCR, 3 біти для цього входу в цьому регістрі повинні бути запрограмовані як 000, але захоплення і/або переривання можуть бути обрані для іншого входу CAP.

Нижче наведено опис бітів регістра керування захопленням CCR (таблиця 4.12).

Таблиця 4.12– Опис бітів регістрів керування захопленням T[0/1/2/3]CCR

Біт	Символ	Значення	Опис	Значення скидання
0	CAP0RE	1	Захоплення CAPn.0 за переднім фронтом: послідовність із 0 в 1 на CAPn.0 змусить CR0 бути завантаженим вмістом TC.	0
		0	Функція відключена.	
1	CAP0FE	1	Захоплення CAPn.0 за заднім фронтом: послідовність із 1 в 0 на CAPn.0 змусить CR0 бути завантаженим вмістом TC.	0
		0	Функція відключена.	
2	CAP0I	1	Переривання CAPn.0 за подією: завантаження CR0 завдяки події CAPn.0 буде генерувати переривання.	0
		0	Функція відключена.	

Продовження таблиці 4.12

Біт	Символ	Значення	Опис	Значення скидання
3	CAP1RE	1	Захоплення CAPn.1 за переднім фронтом: послідовність із 0 в 1 на CAPn.1 змусить CR1 бути завантаженим вмістом TC.	0
		0	Функція відключена.	
4	CAP1FE	1	Захоплення CAPn.1 за заднім фронтом: послідовність із 1 в 0 на CAPn.1 змусить CR1 бути завантаженим вмістом TC.	0
		0	Функція відключена.	
5	CAP1I	1	Переривання CAPn.1 за подією: завантаження CR1 завдяки події CAPn.1 буде генерувати переривання.	0
		0	Функція відключена.	
15:6	–		Зарезервовані. Користувацькі програми не повинні записувати до зарезервованих бітів. Значення зчитане із зарезервованого біта не визначене.	NA

4.3.16 Регістри зовнішнього збігу T[0/1/2/3]EMR

Регістри зовнішнього збігу T[0/1/2/3]EMR мають адреси: 0xE000 403C, 0xE000 803C, 0xE007 003C, 0xE007 403C та керують сигналами на зовнішніх виводах MATn.0…3 при збігу. У наведеному нижче описі, "n" представляє номер таймера, від 0 до 3, а " m " представляє номер збігу, від 0 до 3.

Нижче наведено опис бітів регістрів зовнішнього збігу EMR (таблиця 4.13).

Таблиця 4.13 – Опис бітів регістрів зовнішнього збігу T[0/1/2/3]EMR

Біт	Символ	Опис	Значення скидання
0	EM0	**Зовнішній збіг 0.** Коли відбувається збіг між TC і MR0, цей біт відображає зміну сигналу на зовнішньому виводі: на низький рівень, на високий рівень, або нічого не робити, в залежності від бітів 5:4 цього регістра. Сигнал на виводі MATn.0 формується в додатній логіці: 0 = низький рівень, 1 = високий рівень.	0

Продовження таблиці 4.13

Біт	Символ	Опис	Значення скидання
1	EM1	**Зовнішній збіг 1.** Коли відбувається збіг між TC і MR0, цей біт відображає зміну сигналу на зовнішньому виводі: на низький рівень, на високий рівень, або нічого не робити, в залежності від бітів 7:6 цього регістра. Сигнал на виводі MATn.0 формується в додатній логіці: 0 = низький рівень, 1 = високий рівень.	0
2	EM2	**Зовнішній збіг 2.** Коли відбувається збіг між TC і MR0, цей біт відображає зміну сигналу на зовнішньому виводі: на низький рівень, на високий рівень, або нічого не робити, в залежності від бітів 9:8 цього регістра. Сигнал на виводі MATn.0 формується в додатній логіці: 0 = низький рівень, 1 = високий рівень.	0
3	EM3	**Зовнішній збіг 3.** Коли відбувається збіг між TC і MR0, цей біт відображає зміну сигналу на зовнішньому виводі: на низький рівень, на високий рівень, або нічого не робити, в залежності від бітів 11:10 цього регістра. Сигнал на виводі MATn.0 формується в додатній логіці: 0 = низький рівень, 1 = високий рівень.	0
5:4	EMC0	**Зовнішнє керування збігом 0.** Визначає функції зовнішнього збігу 0. Таблиця 4.14 показує кодування цих бітів.	00
7:6	EMC1	**Зовнішнє керування збігом 1.** Визначає функції зовнішнього збігу 1. Таблиця 4.14 показує кодування цих бітів.	00
9:8	EMC2	**Зовнішнє керування збігом 2.** Визначає функції зовнішнього збігу 2. Таблиця 4.14 показує кодування цих бітів.	00
11:10	EMC3	**Зовнішнє керування збігом 3.** Визначає функції зовнішнього збігу 3. Таблиця 4.14 показує кодування цих бітів.	00
15:12	–	**Зарезервовані.** Користувацькі програми не повинні записувати до зарезервованих бітів. Значення зчитане із зарезервованого біту не визначене.	NA

Нижче наведено функції бітів керування зовнішнього збігу ЕМС (таблиця 4.14).

Таблиця 4.14– Функції бітів керування зовнішнім збігом ЕМС

EMR[11:10], EMR[9:8], EMR[7:6] або EMR[5:4]	Функція
00	Нічого не робиться.
01	Очистити відповідний біт зовнішнього збігу/вихід в 0 (MATn.m на виводі низький рівень, якщо під'єднаний).
10	Встановити відповідний біт зовнішнього збігу/вихід в 1 (MATn.m на виводі високий рівень, якщо під'єднаний).
11	Перемикання відповідного зовнішнього збігу/виходу.

4.3.17 Структура модуля таймерів/лічильників

Структуру модуля таймерів/лічильників наведено на рисунку 4.8.

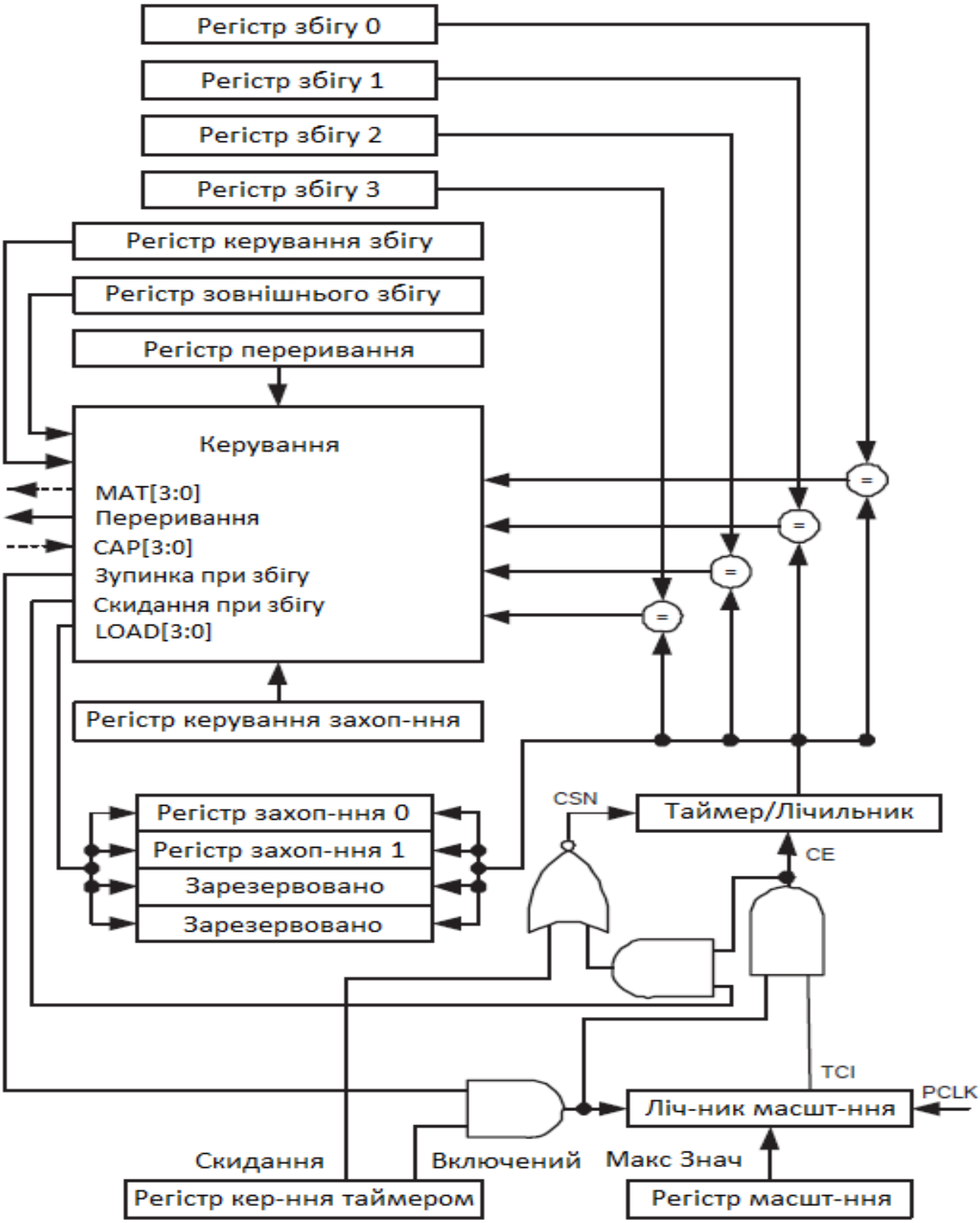


Рисунок 4.8 – Структурна схема модуля таймерів/лічильників

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Скільки таймерів є у складі мікроконтролерів сімейства LPC2300?
2) Опишіть спрощену структуру таймера загального призначення.
3) Від чого залежить швидкість лічби таймера?
4) Скільки каналів захоплення має кожен таймер?
5) Опишіть роботу таймера в режимі захоплення.
6) Опишіть роботу таймера в режимі лічильника зовнішніх подій.
7) Опишіть роботу таймера в режимі збігу.
8) Скільки каналів збігу має кожен таймер?
9) Назвіть та опишіть керуючі регістри модуля таймерів.
10) Дайте опис бітів регістрів переривань.
11) Дайте опис бітів регістрів керування таймерами.
12) Дайте опис бітів регістрів керування підрахунком.
13) Дайте опис бітів регістрів таймерів/лічильників.
14) Дайте опис бітів регістрів масштабування та лічильників масштабування.
15) Дайте опис бітів регістрів збігу та керування збігом.
16) Дайте опис бітів регістрів захоплення та керування захопленням.
17) Дайте опис бітів регістрів зовнішнього збігу.
18) Опишіть структуру модуля таймерів/лічильників.
19) За яким фронтом зовнішнього сигналу може відбуватися подія захоплення?
20) Як може змінюватися сигнал на відповідному виході при настанні події збігу?
21) Які прапорці переривань можуть використовуватись при програмуванні роботи таймерів?

4.4 Модуль ШІМ

4.4.1 Загальна характеристика

На перший погляд, модуль ШІМ здається набагато складнішим, ніж таймери загального призначення. Насправді він представляє собою той же таймер, який оснащено додатковими блоками [1, 4, 7]. Модуль ШІМ (рисунок 4.9) забезпечує формування до шести ШІМ–сигналів з керуванням за одним фронтом чи до трьох сигналів – з керуванням за обома фронтами.

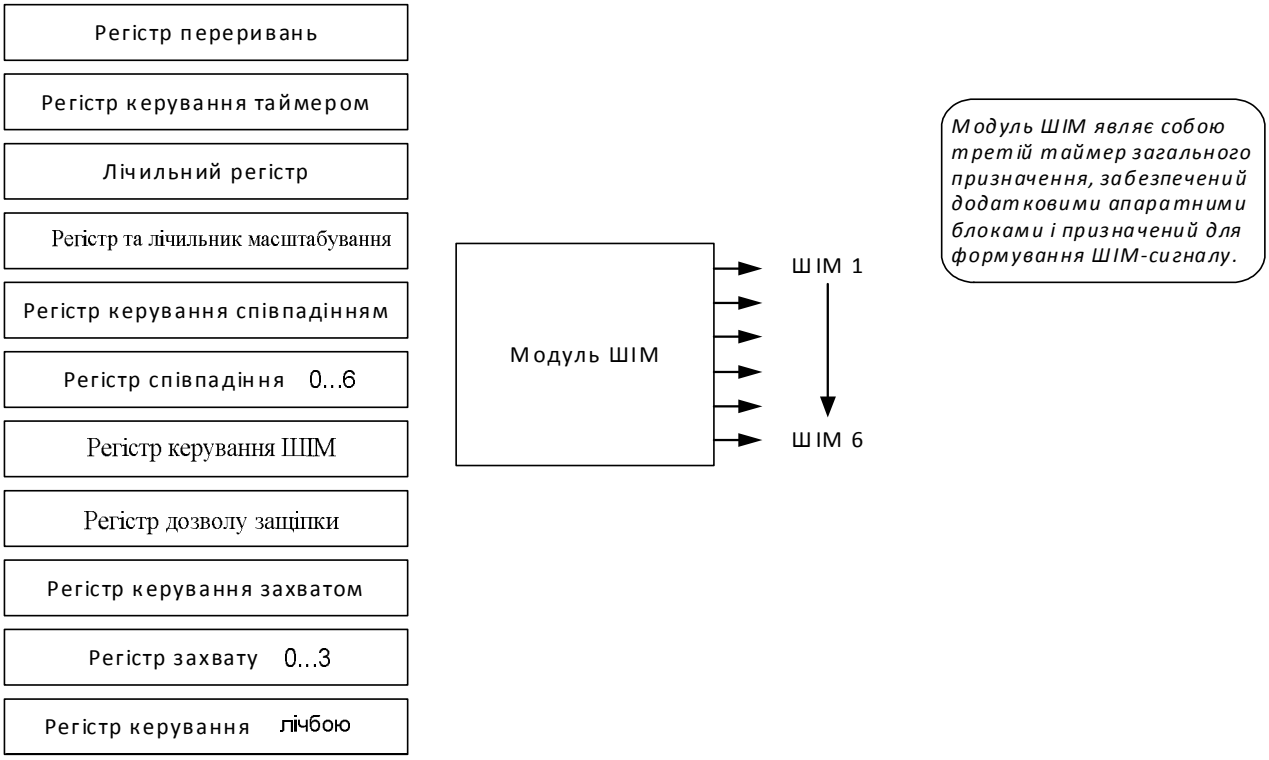


Рисунок 4.9 – Модуль ШІМ

У таймерах загального призначення при записі нового числа в регістр збігу це число починає діяти відразу ж. Якщо не передбачити спеціальних заходів в програмі, це може викликати спотворення ШІМ–сигналу. При одночасному оновленні декількох каналів нові значення будуть вступати в дію в різні моменти формованого імпульсу, що може привести до непередбачуваних результатів. Щоб уникнути подібних ситуацій, в модулі ШІМ передбачений спеціальний механізм защіпок (рисунок 4.10), який дозволяє змінювати значення ШІМ «на льоту», при цьому нові значення будуть вступати в силу одночасно і тільки на початку нового періоду.

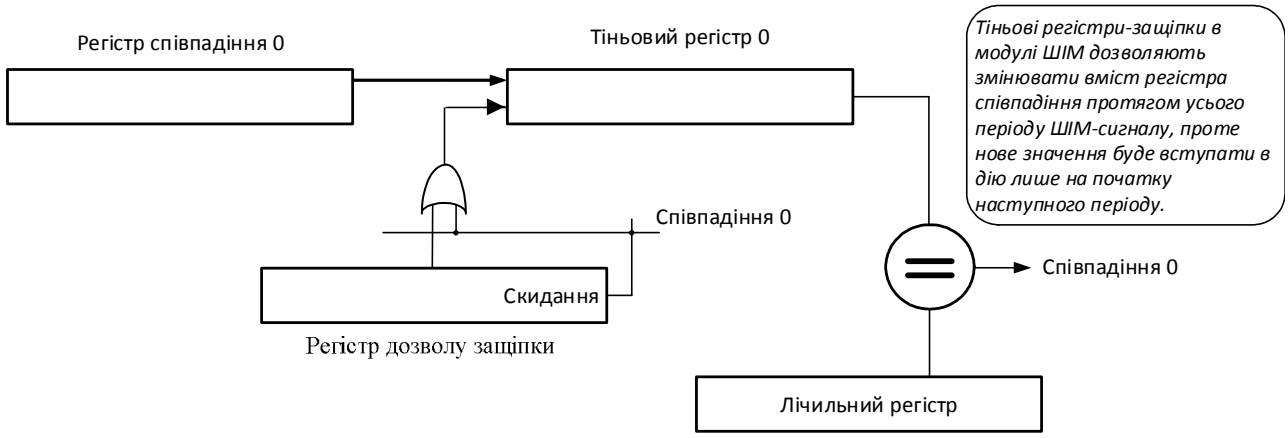


Рисунок 4.10 – Регістр – защіпка

Вміст будь–якого з регістрів збігу може бути змінено в будь–який момент часу, однак він не буде використовуватися до тих пір, поки не буде встановлений відповідний цьому каналу біт в регістрі дозволу защіпки (Latch Enable Register – LER).

Після встановлення зазначеного біта вміст регістра співпадіння буде скопійовано в тіньовий регістр на початку наступного періоду. Таким чином, гарантується, що всі зміни регістрів будуть здійснюватися одночасно на початку періоду. Крім цих тіньових регістрів–защіпок, канали збігу модуля ШІМ функціонують точно так само, як і в таймерах.

Другою відмінністю модуля ШІМ від звичайних таймерів є схема керування виводами мікроконтролера. Канали співпадіння здійснюють керування виводами не безпосередньо, а через RS– тригери (рисунок 4.11).

Така схема з RS– тригерів і мультиплексорів дозволяє формувати сигнали, які керуються за одним чи за обома фронтами. Стан мультиплексорів визначається регістром керування PWMPCR. За допомогою цих мультиплексорів можна задавати одну з двох конфігурацій вихідних каскадів модуля.

Перша конфігурація відповідає модуляції з керуванням за одним фронтом (рисунок 4.12).

У цій конфігурації вихід 0– го каналу збігу підключається до входу S всіх тригерів, а виходи інших каналів – до входів R відповідних тригерів. Таким чином, 0– й канал визначає тривалість періоду ШІМ– сигналів, загальну для всіх

каналів. В кінці періоду цей канал скидає лічильний регістр модуля і встановлює свій вихід в 1.

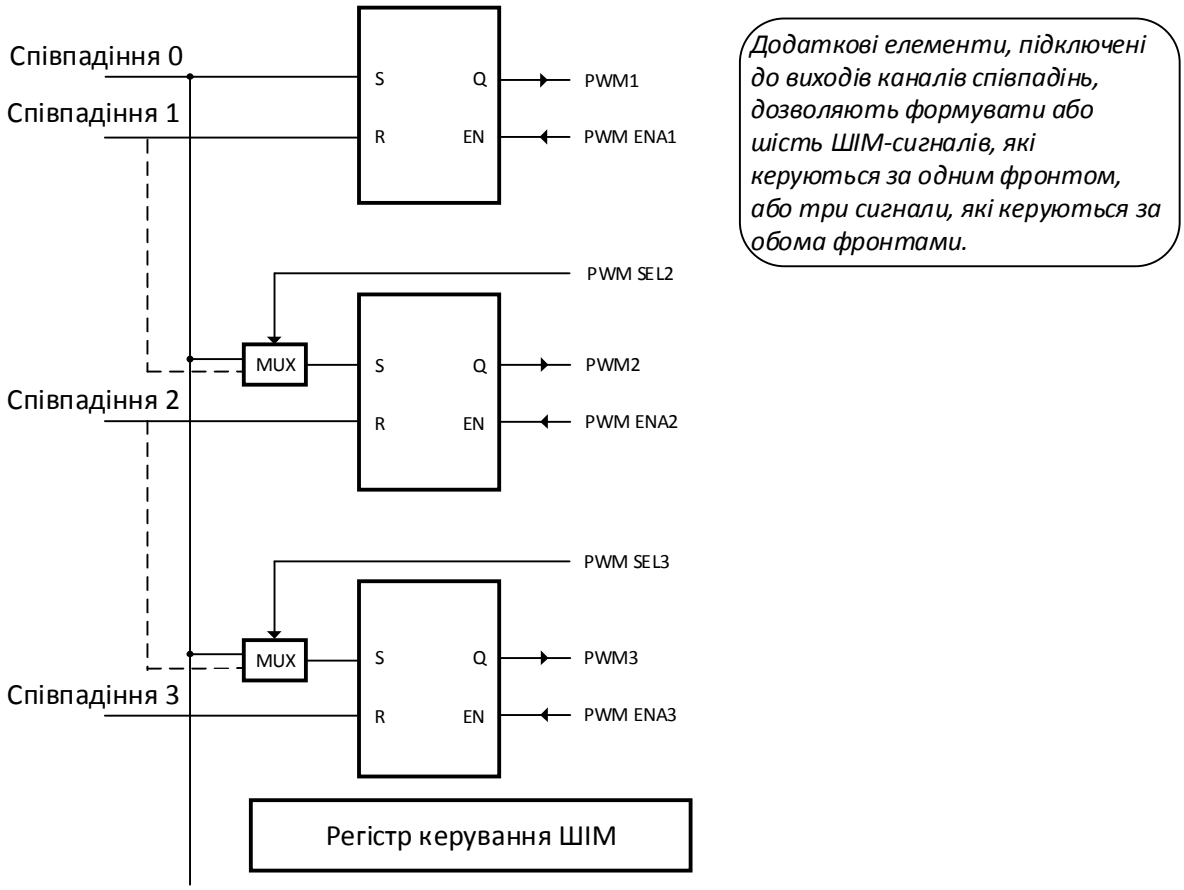


Рисунок 4.11 – Керування виводами в модулі ШІМ

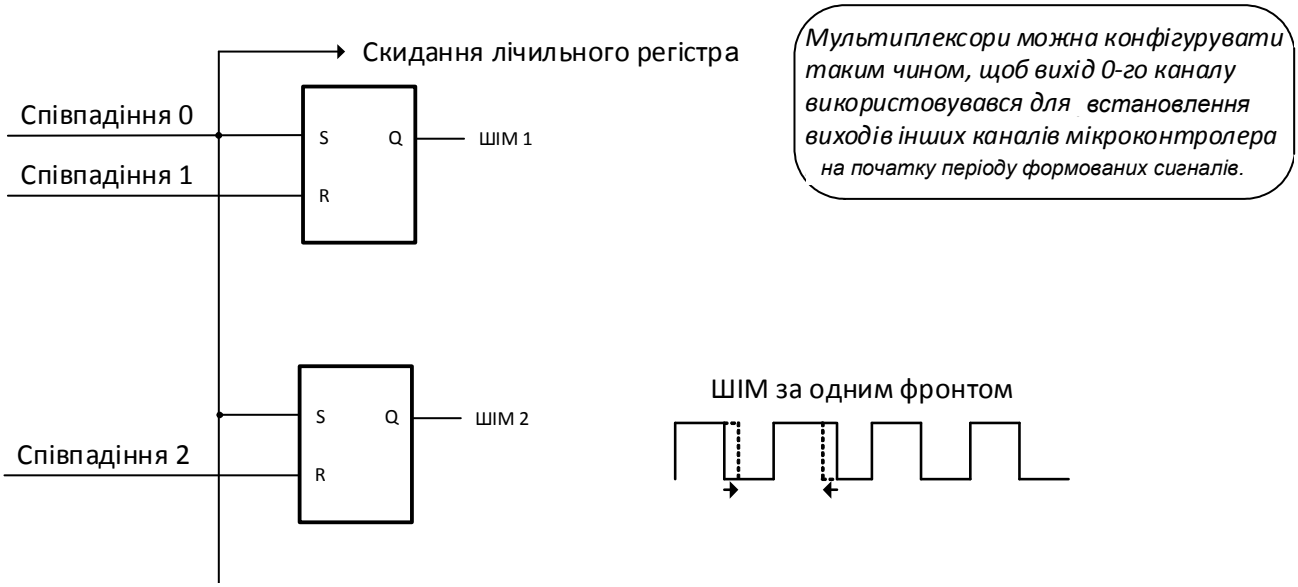


Рисунок 4.12 – Модуляція за одним фронтом

В результаті всі тригери встановлюються на початку періоду ШІМ–сигналу. При цьому на їх виходах Q з'являється напруга ВИСОКОГО рівня, а на виводах мікроконтролера формується наростаючий фронт. Модуляція сигналу здійснюється іншими каналами співпадіння, при цьому кожен канал співпадіння відповідає одному каналу ШІМ. При збігу одного з каналів відповідний тригер скидається і на виході каналу ШІМ формується спадаючий фронт. Таким чином, модуляція здійснюється зміною вмісту регістра необхідного каналу збігу.

Друга конфігурація мультиплексорів відповідає модуляції ШІМ– сигналів з керуванням за обома фронтами (рисунок 4.13). У цій конфігурації 0– й канал збігу використовується виключно для скидання лічильного регістра модуля в кінці кожного періоду ШІМ– сигналу, а входи R і S тригерів підключені до виходів відповідних каналів збігу. На початку періоду на виході каналу ШІМ присутня напруга НИЗЬКОГО рівня. Наростаючий фронт імпульсу формується каналом збігу, підключеним до входу S тригеру, а спадаючий фронт – каналом, підключеним до входу R.

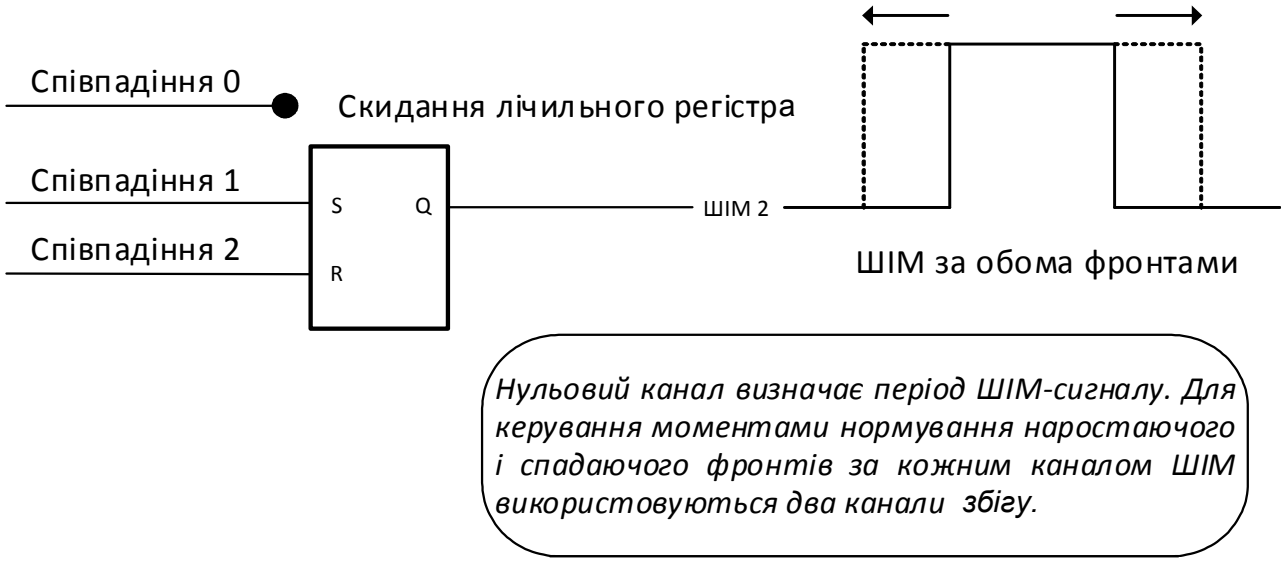


Рисунок 4.13 – Модуляція за обома фронтами

Зверніть увагу: регістр PWMPCR використовується для того, щоб гарантувати встановлення виходів каналів збігу при настанні відповідних подій. Якщо в цьому регістрі буде записано некоректне значення, модуль ШІМ працювати не буде. Крім того, на відміну від базового таймера, для роботи модуля ШІМ не потрібно ніяких переривань.

4.4.2 Режим лічильника

Як і таймери загального призначення, модуль ШІМ теж може працювати в режимі лічильника зовнішніх подій. Для керування режимом використовується регістр керування лічильником. За допомогою цього регістра можна змінити тактовий сигнал модуля з Pclk на сигнал від зовнішнього джерела, що подається на вхід захоплення модуля. При цьому лічильний регістр може інкрементуватися за наростаючим, спадаючим або навіть за обома фронтами зовнішнього сигналу.

4.4.3 Структурна схема модуля широтно–імпульсного модулятора

На рисунку 4.14 наведено структурну схему модуля ШІМ. Частини, які були додані до стандартного блоку таймера (рисунок 4.12), наведено на правій стороні і у верхній частині рисунка 4.14.

Модуль ШІМ (Pulse width modulator–PWM) заснований на стандартному блоці таймера і успадковує всі його функції, хоча окрема функція ШІМ закріплена на LPC2300. Таймер призначений для підрахунку періодів тактової частоти периферії (P_{CLK}) і при необхідності повинен генерувати переривання або виконувати інші дії при настанні певних значень таймера на підставі семи регістрів збігу. Функція ШІМ додається до цих функцій і заснована на регістрах подій збігу. Здатність окремо керувати зростанням і спаданням фронтів вихідних імпульсів дозволяє PWM використовуватися для значної кількості застосувань. Наприклад, для керування багатофазним двигуном, як правило, потрібно три ШІМ, які не перекривають один одного, з індивідуальним керуванням ширини імпульсів і позицій всіх трьох каналів. Два регістри збігу можуть бути використані для забезпечення одного фронту керування виходу ШІМ. Один регістр збігу (PWMMR0) керує шпаруватістю сигналу ШІМ, шляхом скидання лічильника при збігу. Інший регістр збігу керує положенням фронту PWM. Додаткові сигнали ШІМ вимагають тільки один зареєстрований збіг, так як частота повторення однакова для всіх виходів ШІМ. Декілька поодиноких фронтів керуються ШІМ–виходами, які повинні мати наростаючий фронт на початку кожного циклу ШІМ, коли відбувається збіг PWMMR0. Три регістри збігу можуть бути використані для забезпечення виведення ШІМ за обома фронтами. Знову ж, регістр збігу PWMMR0 контролює частоту ШІМ. Інші

регістри збігу керують двома сигналами PWM. Спеціальні регістри забезпечують керування за подвійними фронтами, як за переднім так і за заднім фронтом. Це дозволяє працювати як із додатними так і з від`ємними імпульсами.

Опис керуючих регістрів модуля ШІМ наведено у [4, 8].

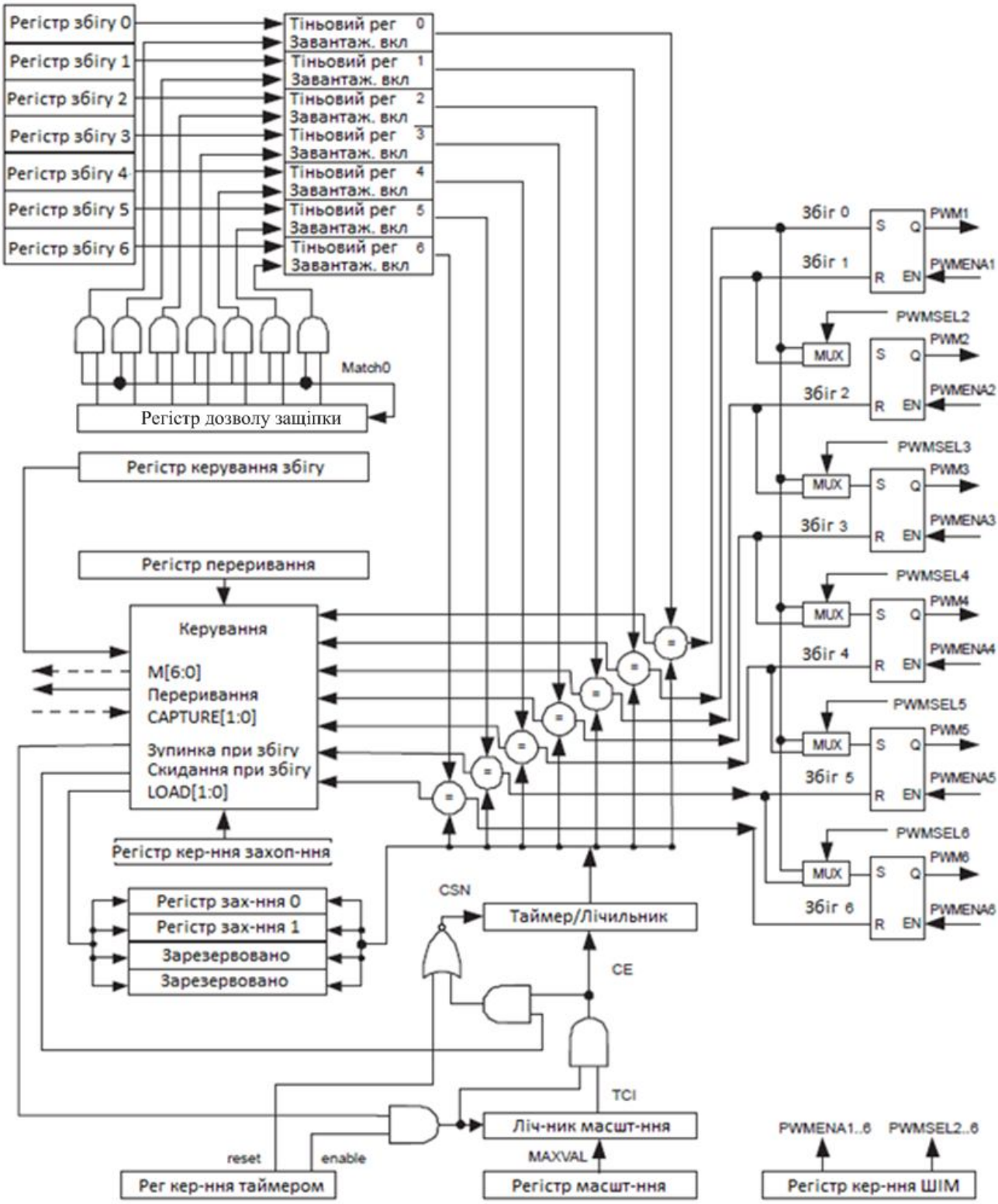


Рисунок 4.14 – Структурна схема модуля ШІМ

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Дайте визначення широтно–імпульсній модуляції.
2) Як розраховується шпаруватість ШІМ–сигналу?
3) Які ШІМ–сигнали може формувати модуль ШІМ?
4) Який таймер загального призначення виконує функцію модуля ШІМ?
5) Який механізм введено в модуль ШІМ з метою недопущення спотворення ШІМ–сигналу?
6) В які моменти періоду формування ШІМ–сигналу вступають в силу нові значення при оновленні відповідного каналу?
7) Який регістр та як використовується при реалізації механізму недопущення спотворення ШІМ–сигналу?
8) Які відмінності існують у функціонуванні каналів збігу модуля ШІМ крім наявності тіньових регістрів–защіпок?
9) Чим відрізняється схема керування виводами модуля ШІМ від звичайних таймерів?
10) Поясніть роботу схеми керування виводами в модулі ШІМ.
11) Поясніть особливості широтно–імпульсної модуляції за одним фронтом.
12) Поясніть особливості широтно–імпульсної модуляції за обома фронтами.
13) Які регістри використовуються для керуванням модулем ШІМ?
14) Поясніть використання модуля ШІМ в режимі лічильника.
15) Яка тактова частота використовується модулем ШІМ?
16) Скільки регістрів збігу має модуль ШІМ?
17) Як можуть змінюватись фронти вихідних імпульсів в модулі ШІМ при настанні події збігу?
18) Який регістр використовується для керування шпаруватістю?
19) Який регістр використовується для керування положенням фронту PWM?
20) Який регістр використовується для керування частотою ШІМ–сигналу?

4.5 Сторожовий таймер

4.5.1 Загальні відомості

Як і більшість інших мікроконтролерів, мікроконтролери сімейства LPC2300 мають в своєму складі сторожовий таймер, який використовується для встановлення контролю над програмою в разі її фатального збою [1, 4, 7]. Як тактовий сигнал сторожового таймера може використовуватися сигнал від вбудованого RC–генератора, сигнал від генератора модуля RTC або системний тактовий сигнал Pclk.

4.5.2 Період сторожового таймера

Сторожовий таймер має чотири керуючих регістри. Період тайм–ауту сторожового таймера визначається вмістом регістра постійної таймера WDTC. Значення періоду тайм–ауту визначається за формулою:

Wdperiod = Pclk x WDTC x 4.

Вбудований сторожовий таймер може викликати скидання процесора або згенерувати переривання. При скиданні за сторожовим таймером встановлюється спеціальний прапорець, так що програма може блокувати виконання «м'якого» скидання.

Мінімальне значення регістра WDTC дорівнює 256, а максимальне – 232. Таким чином, при тактовій частоті процесора 60 МГц мінімальний період сторожового таймера дорівнює 17, 066 мкс, а максимальний – трохи менше за 5 хвилин.

Після запису значення постійної сторожового таймера можна приступати до конфігурації його режиму роботи. Регістр режиму сторожового таймера WDMOD (рисунок 4.15) містить три керуючих біти, два з яких визначають, чи буде сторожовий таймер генерувати переривання і сигнал скидання процесора, а третій використовується для дозволу роботи сторожового таймера.

Крім того, в регістрі керування є два прапорці. Один з них, WDTOF, встановлюється при тайм–ауті сторожового таймера і скидається тільки після зовнішнього апаратного скидання.

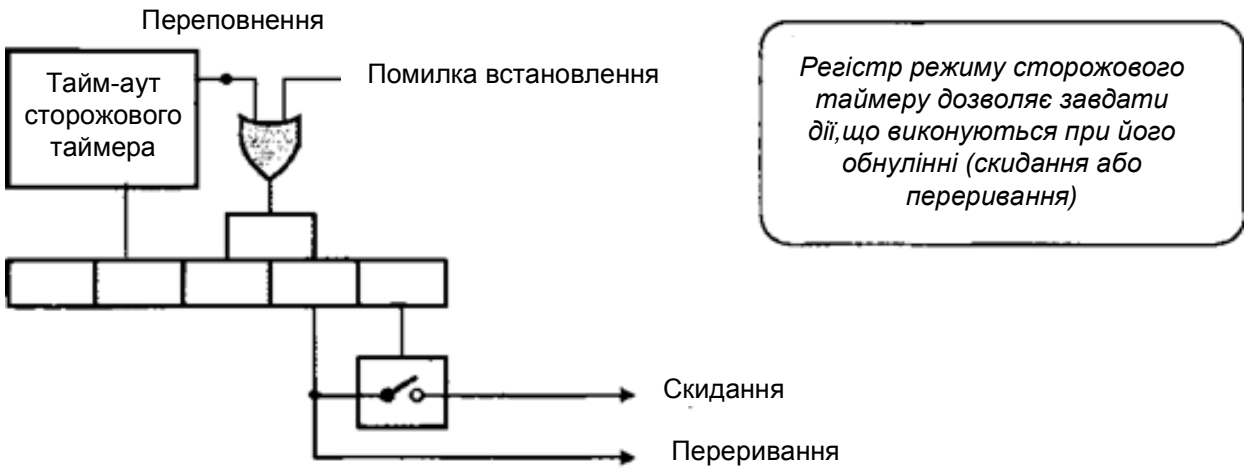


Рисунок 4.15 –Регістр режиму сторожового таймера

Це дозволяє в стартовому коді програми визначити, що стало причиною скидання – включення живлення або помилка в програмі. Другий прапорець є прапорцем переривання сторожового таймера. Даний прапорець доступний тільки для читання, а для скидання його необхідно прочитати. Якщо ви збираєтеся налагоджувати програму при включеному сторожовому таймері, не дозволяйте скидання від нього, оскільки це призведе до відключення JTAG–відлагоджувача при тайм–ауті сторожового таймера.

Після завдання періоду тайм–ауту і режиму роботи сторожового таймера його можна включити, записавши в регістр запуску WDFEED певну послідовність чисел, аналогічну послідовності, яка необхідна для включення схеми ФАПЧ. Для запуску сторожового таймера треба спочатку записати число 0AA, а потім число 0x55. Якщо ця послідовність буде порушена, відбудеться помилка запуску сторожового таймера, яка викличе тайм–аут сторожового таймера, що супроводжується відповідною генерацією переривання/скидання. Також необхідно відзначити, що хоча сторожовий таймер може бути дозволений в регістрі режиму, він не почне працювати до першого запису необхідної послідовності в регістр запуску. Щоб уникнути спрацьовування сторожового таймера після того, як він буде запущений, його необхідно періодично перевстановити, записуючи ту ж послідовність чисел.

Останнім регістром сторожового таймера є регістр значення WDTV, який дозволяє дізнатися поточне значення сторожового таймера.

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Поясніть призначення сторожового таймера.
2) Який сигнал може використовуватися в якості тактового для сторожового таймера?
3) Скільки керуючих регістрів має сторожовий таймер?
4) Як обчислюється період тайм–ауту сторожового таймера?
5) Які дії може викликати сторожовий таймер після закінчення тайм–ауту?
6) Чому дорівнює мінімальний та максимальний період сторожового таймера при тактовій частоті процесора 60 МГц?
7) Опишіть призначення регістра постійної таймера WDTC.
8) Опишіть призначення регістра режиму сторожового таймера WDMOD.
9) Як в стартовому коді програми визначити, що стало причиною скидання – включення живлення або помилка в програмі?
10) Як можна скинути прапорець переривання сторожового таймера?
11) Як можна запобігти відключенню JTAG–відлагоджувача при тайм–ауті сторожового таймера?
12) Який регістр використовується для включення сторожового таймера?
13) Які дії треба виконати при програмування запуску сторожового таймера?
14) Що станеться з роботою сторожового таймера в разі порушення відповідної послідовності запису у регістр WDFEED?
15) Коли сторожовий таймер починає працювати після його дозволу у регістрі режиму?
16) Що треба зробити для уникнення спрацьовування сторожового таймера після того, як він буде запущений?
17) Опишіть призначення регістра WDTV.

4.6 Модуль УАПП

4.6.1 Загальні відомості

Обмін інформацією між мікропроцесором/мікроконтролером та модемом може здійснюватися через інтерфейс RS–232 (рисунок 4.16), який включає:

– УАПП – універсальний асинхронний послідовний програмований приймач передавач;

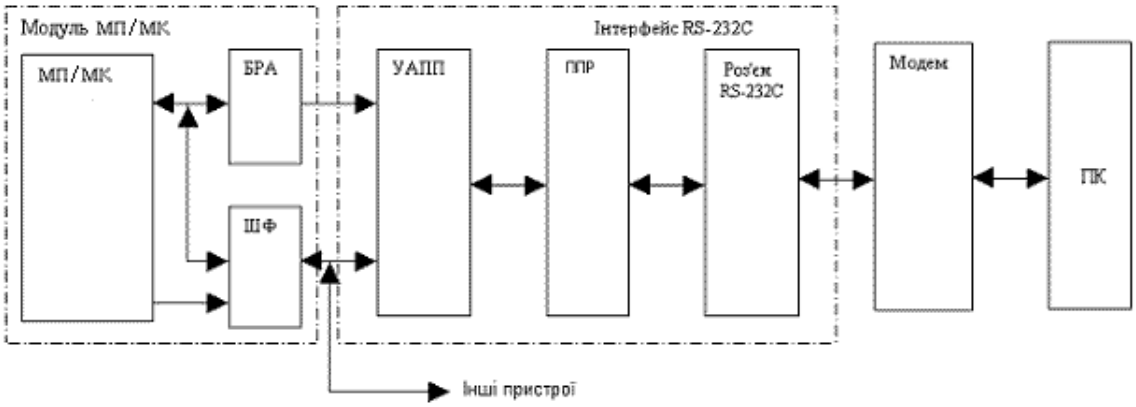
– ППР – пристрій перетворення рівнів;

– роз'єм RS–232C.

Крім інтерфейсу RS–232C схема сполучення містить:

– БРА – буферний регістр адреси;

– ШФ – шинний формувач.



МП - мікропроцесор; БРА – буферний регістр адреси
ШФ – шинний формувач; УАПП – універсальний асинхронний послідовний програмований приймач – передавач;
ППР – пристрій перетворення рівнів; ПК - персональний комп'ютер
МК- мікроконтролер

Рисунок 4.16 – Структурна схема сполучення МП/МК з модемом за допомогою інтерфейсу RS – 232C

4.6.2 Універсальний асинхронний послідовний програмований приймач–передавач (УАПП)

УАПП, наприклад, мікросхема TL16C450 (рисунок 4.17) перетворює дані з паралельного формату в послідовний при передачі (виведенні) з мікропроцесора і з послідовного формату в паралельний при прийомі (введенні) у мікропроцесор.

Формат даних, які передаються у канал зв'язку в послідовному асинхронному форматі, представлений на рисунку 4.18.

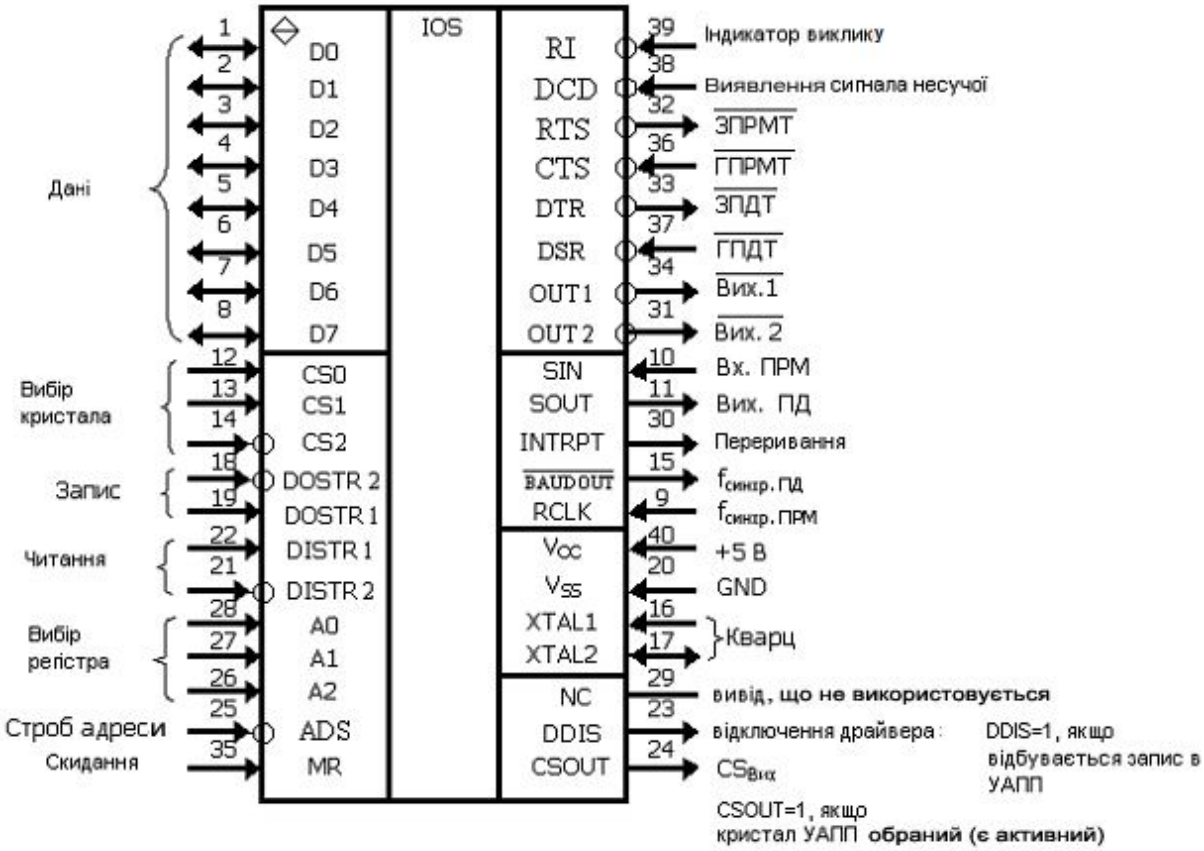


Рисунок 4.17 – Умовне позначення ІМС TL16C450 (УАПП–UART)

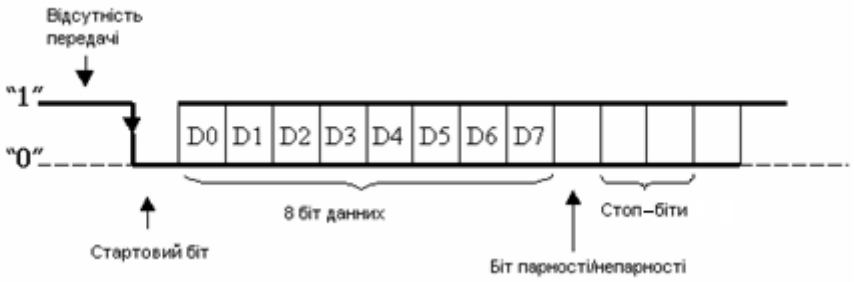


Рисунок 4.18 – Формат даних інтерфейсу RS–232C

Власне дані (5, 6, 7 чи 8 біт) супроводжуються стартовим бітом, бітом парності/непарності (якщо такий контроль програмно передбачений) і стоповим одиничним сигналом, що включає 1; 1,5 чи 2 стоп–біти. Одержавши стартовий біт, приймач вибирає з лінії біти даних через визначені інтервали часу. Дуже важливо, щоб тактові частоти приймача і передавача були однаковими (допустима розбіжність – не більше 10 %).

Швидкість передачі за RS–232C, наприклад, може вибиратися з ряду: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 біт/с (бод).

4.6.3 Пристрій перетворення рівнів (ППР)

Усі сигнали RS–232C передаються/приймаються спеціально обраними рівнями, що забезпечують високу завадостійкість зв'язку (рисунок 4.19). Слід зазначити, що дані передаються/приймаються в інверсному вигляді: логічній одиниці відповідає низький рівень, а логічному нулю – високий рівень.

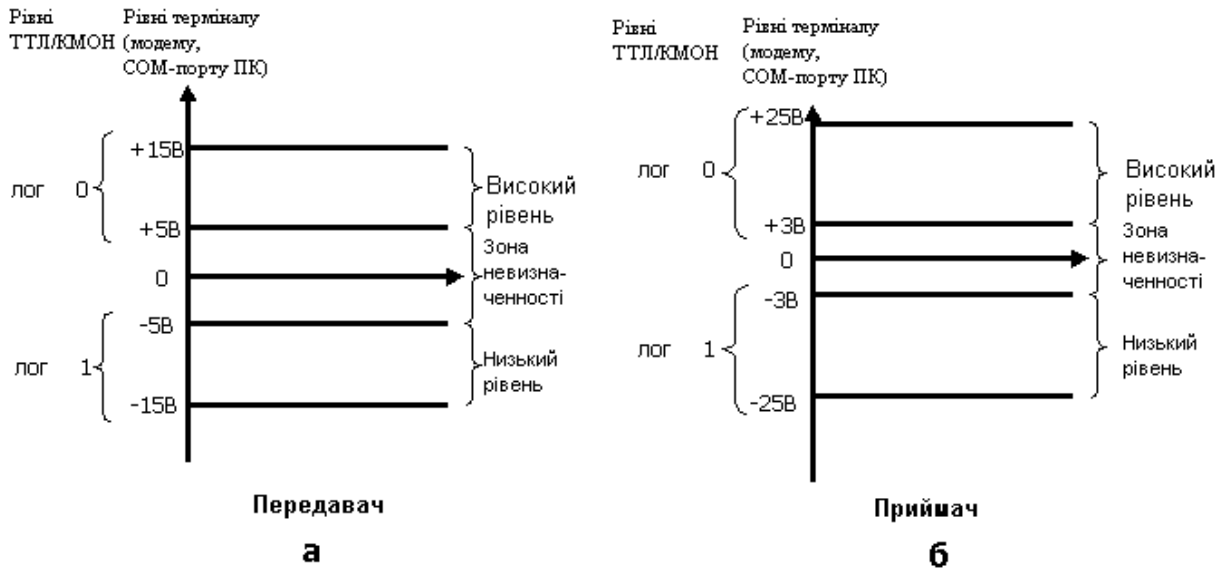


Рисунок 4.19 – Рівні сигналів RS – 232C: а – на передаючій, б – на приймальній кінцях лінії зв’язку

Як видно з рисунка 4.19 під час передачі логічного нуля на виході інтерфейсу формується високий рівень напруги в діапазоні: +5В…+15В, під час передачі логічної одиниці – низький рівень напруги в діапазоні: –5В…–15В.

Під час прийому на вхід інтерфейсу надходить високий рівень напруги в діапазоні: +3В…+25В, що несе інформацію про логічний 0, чи низький рівень напруги в діапазоні: –3В…–25В, що відображає логічну одиницю.

Таким чином, для узгодження ТТЛ/КМОН–рівнів сигналів, що діють у мікропроцесорній системі, з рівнями сигналів послідовного інтерфейсу, що передаються у лінію зв'язку або приймаються з лінії зв'язку використовують пристрої перетворення рівнів (ППР).

Існують різні варіанти схемної реалізації ППР, одним із яких є застосування мікросхеми фірми MAXIM: MAX232A. Дана мікросхема (рисунок 4.20) вимагає одне джерело живлення +5В та ряд додаткових елементів –

конденсаторів C1, C2, … , C5, що не є надмірною ціною за переваги її застосування.

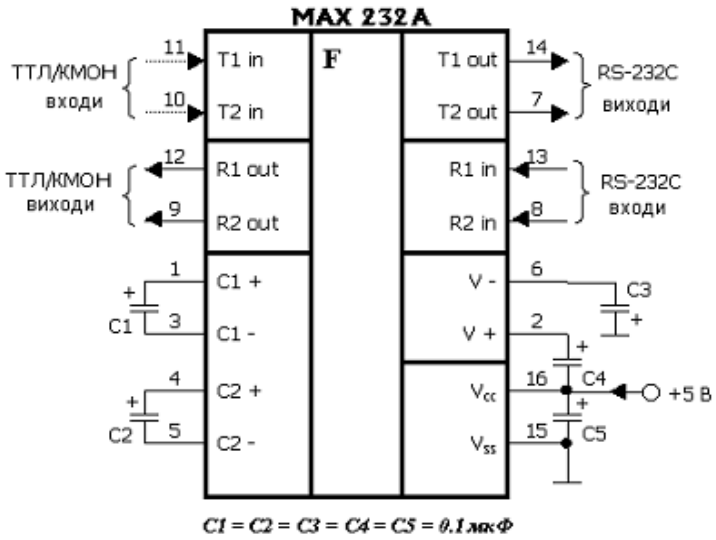


Рисунок 4.20 – Позначення і особливості підключення мікросхеми MAX232A

4.6.4 Роз’єм RS–232C

Для зв'язку інтерфейсу RS–232C із зовнішнім терміналом (модемом) або ПК може використовуватися 25– чи 9–контактні роз’єми (рисунок 4.21).

Роз’єм RS–232C

модем (термінал, ПК)

Контакт	Ланцюг
1	DCD (виявлення несучої)
2	RxD (вхід приймача)
3	TxD (вихід передавача)
4	DTR (запит передавача термінала)
5	SG (сигнальна земля)
6	DSR (готовність передавача термінала)
7	RTS (запит приймача термінала)
8	CTS (готовність приймача термінала)
9	RI (індикатор виклику)

ППР, УАПП, МП

Рисунок 4.21 – 9-контактний роз'єм RS-232C

Призначення основних контактів наступне:

SG – сигнальне заземлення, нульовий провід;

TxD – дані, що передаються мікропроцесором у послідовному коді (від'ємна логіка)

RxD – дані, що приймаються мікропроцесором у послідовному коді (від'ємна логіка);

DCD – виявлення несучої даних (детектування сигналу, що приймається МП–м від модему);

DTR – запит передавача термінала (модему або ПК);

DSR – готовність передавача термінала (модему або ПК);

RTS – запит приймача термінала (модему або ПК);

CTS – готовність приймача термінала (модему або ПК);

RI – індикатор виклику. Говорить про прийом модемом сигналу виклику від телефонної мережі.

4.6.5 Модуль УАПП в мікроконтролерах сімейства LPC2300

4.6.5.1 Загальна характеристика

В даний час у більшості мікроконтролерах сімейства LPC2300 є по чотири модулі універсального асинхронного приймача–передавача (UART): UART0, UART1, UART2, UART3 [1, 4, 7]. Всі модулі використовуються однаковим чином, якщо не рахувати того, що в модулі UART1 реалізовано повний модемний інтерфейс, а модуль UART3 підтримує передачу за протоколом IrDA. Всі модулі відповідають специфікації «Промислового стандарту 550» та мають вбудований контролер швидкості передачі і 16–байтні буфери (FIFO) прийому і передачі (рисунок 4.22). Крім цього, за допомогою модуля UART3 можна реалізувати не тільки провідний зв'язок за протоколом RS–232, а також безпровідний за стандартом IrDA.

Спрощену архітектуру модуля UART0 наведено на рисунку 4.23.

4.6.5.2 Налаштування швидкості обміну

Користувач може задавати швидкість обміну для любого з модулей UART безпосередньо в програмі чи використавши функцію автоматичного визначення швидкості. В останньому випадку модуль буде сконфігуровано автоматично.

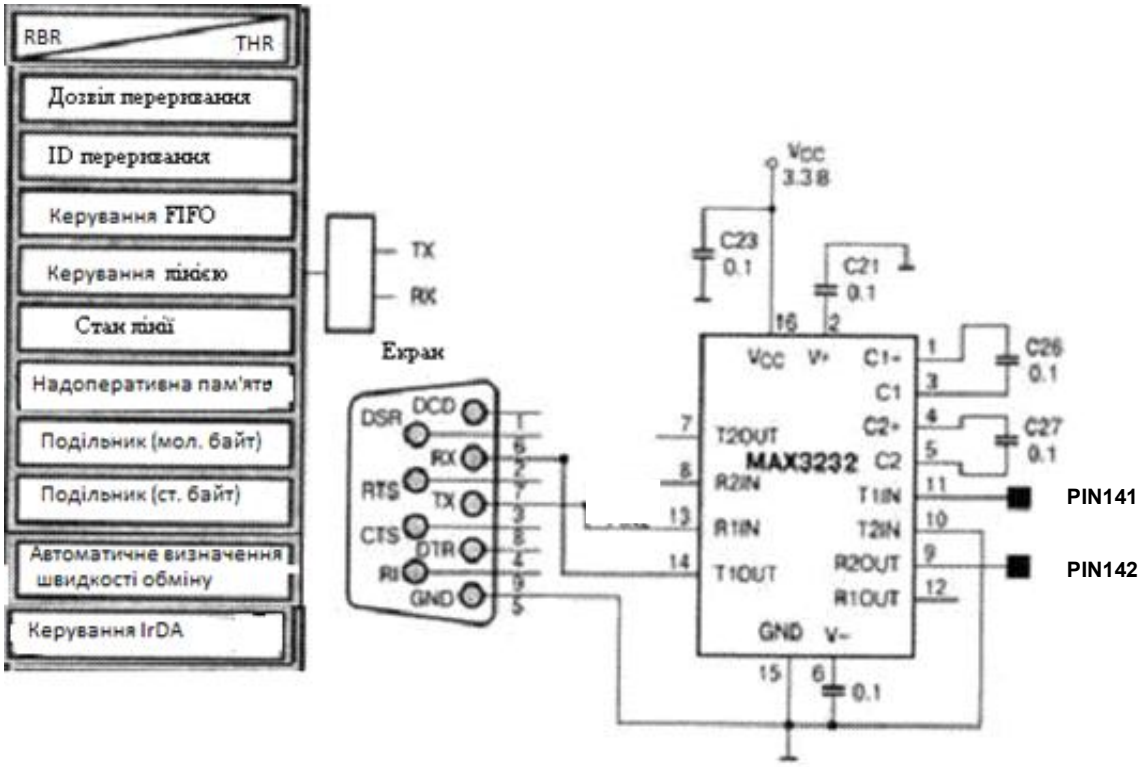


Рисунок 4.22 – Керуючі регістри модуля UART та зв'язок модуля з модемом

Ініціалізація модуля UART проводиться таким чином:

```
voidinit_serial (void)      {      /* Ініціалізація послідовного інтерфейсу */
PINSEL = 0x00050000;               /*Дозволяємо  RxD та TxD/
U0LCR = 0x00000083;                /*8 біт, без контролю парності, 1 стоп–біт*/
U0DLL = 0x00000002;                /*Швидкість обміну 9600 бод при частоті VPB 30 МГц *//*
U0LCR = 0x00000003; }              /*DLAB=0*
```

Спочатку програмується блок керування функцією виводів для відключення виводів мікроконтролера від портів введення/виведення і підключення їх до модуля UART. Потім в регістрі керування лінією UART (LCR) задається формат передачі даних (рисунок 4.24).

У нашому прикладі встановлюється наступний формат даних, які передаються: 8–бітове слово, один стоп–біт, контроль парності вимкнено. У регістрі LCR є додатковий біт (DLAB), який керує доступом до регістра–защіпки подільника контролера швидкості передачі. Для програмування контролера цей біт потрібно встановити в 1.

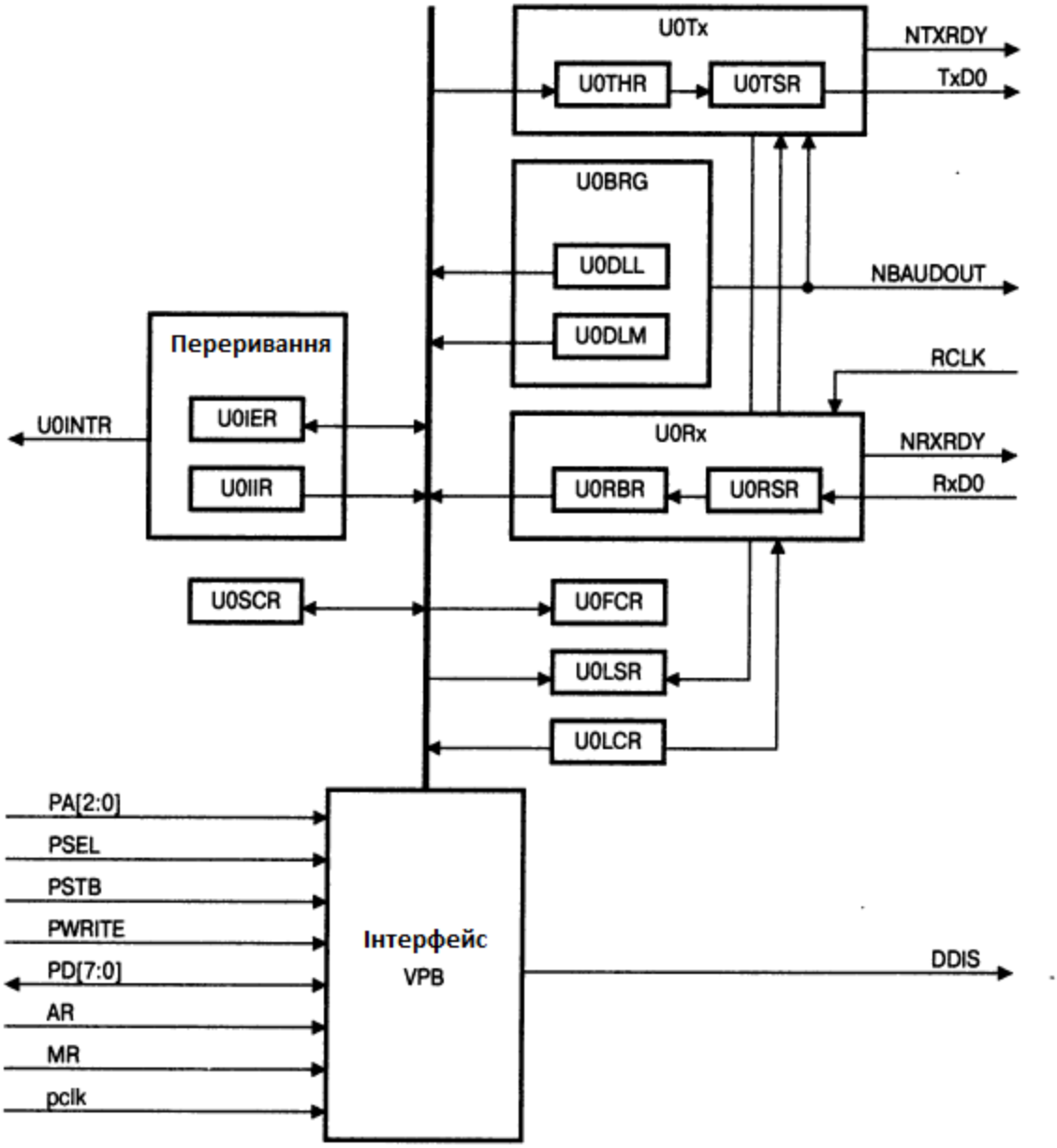


Рисунок 4.23 – Спрощена архітектура модуля UART0

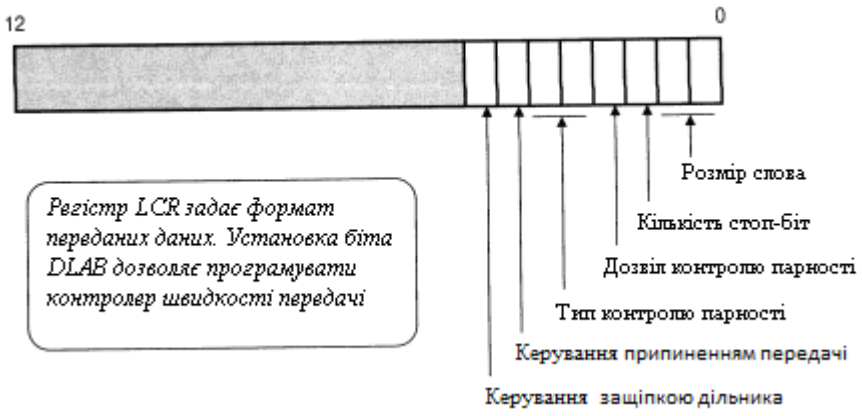


Рисунок 4.24 – Регістр керування лінією UART

Контролер швидкості передачі являє собою 16–бітний попередній подільник, що формує з сигналу Pclk тактовий сигнал модуля UART, частота якого повинна бути в 16 разів більше необхідної швидкості передачі. Таким чином, формула для обчислення коефіцієнта ділення має вигляд:

Кділ = Pclk/16*BAUD, де BAUD – необхідна швидкість передачі.

Наприклад, для програмування BAUD = 9600 біт/с при Pclk=30 М Гц

Кділ = 30000000/16 х 9600 ≈ 194, або 0хС2.

Дійсна швидкість передачі при такому значенні дільника дорівнює 9665 бод. Часто немає можливості встановити точне значення необхідної швидкості передачі, однак модуль UART буде працювати навіть при 5–відсотковому відхиленні тривалості бітового інтервалу від номінальної. Завдяки цьому у вас є певна свобода при конфігуруванні UART, якщо частоту Pclk необхідно підбирати для отримання точних часових характеристик інших периферійних пристроїв, наприклад для тактової синхронізації модуля CAN. Коефіцієнт ділення подільника зберігається в двох регістрах–защіпках: DLM і DLL. Перші вісім біт обох регістрів використовуються для зберігання відповідно старшого і молодшого байтів коефіцієнта ділення, як показано на рисунку 4.25.

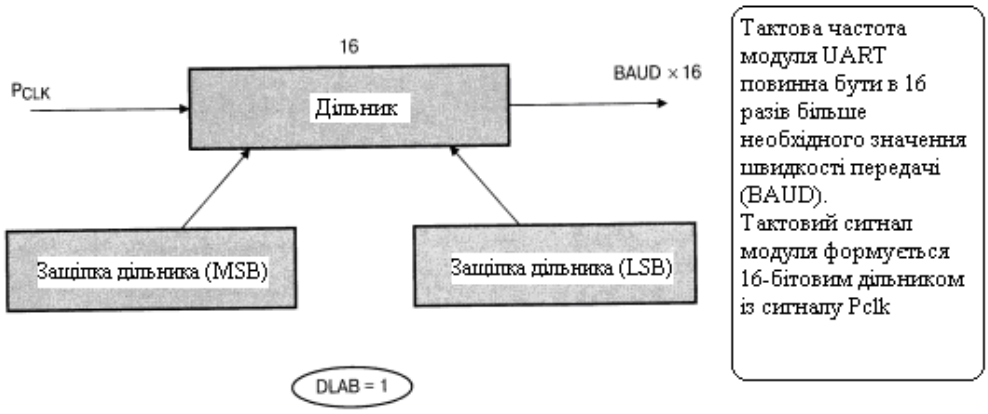


Рисунок 4.25 – Формування швидкості передачі даних UART

Програмування швидкості обміну виконується при одиничному значенні біта DLAB регістра LCR, після чого цей біт треба скинути в нуль, щоб захистити регістр U0DLL та U0DLM від випадкової зміни.

4.6.5.2 Автовизначення швидкості обміну

Будь–який з модулей UART мікроконтролера LPC23xx можна сконфігурувати для автоматичного визначення швидкості передачі пакету даних, який приймається. Після визначення швидкості в регістри–защіпки подільника заносяться коректні значення, які забезпечують роботу модуля на даній швидкості. Керування функцією автоматичного визначення швидкості здійснюється за допомогою однойменного регістра UxACR (рисунок 4.26)

31 ... 10	9	8	7	...	3	2	1	0
	Скидання переривання за тайм-аутом автовизначення швидкості (ABTOIntClr)	**Скидання переривання за завершенням автовизначення швидкості (ABEOIntClr)**				**Автоматичний перезапуск (AutoRestart)**	**Режим (Mode)**	**Запуск (Start)**

Рисунок 4.26 – Регістр автоматичного визначення швидкості обміну

Після встановлення біта Start регістра автоматичного визначення швидкості обміну модуль UART переходить до очікування символу «A» (0x16) чи «a» (0x41). Під час прийому символу за допомогою вбудованого таймера визначається інтервал між двома фронтами символу, на основі чого обчислюється швидкість обміну. Блок автоматичного визначення швидкості може працювати в 2 режимах. В режимі 1 (біт Mode встановлено) швидкість обміну визначається за тривалістю старт–біта, а режим 0 (біт Mode скинуто) – за загальною тривалістю старт–біта та 0–го біта слова даних. Обидва режими повністю взаємозамінні, тим не менше режим 0 більше підходить для роботи при високих швидкостях обміну, а режим 1 – при низьких.

Якщо потрібний символ не буде передано, то блок автоматичного визначення швидкості зафіксує тайм–аут. Якщо біт AutoRestart встановлено, то модуль UART буде старатись визначити швидкість обміну за наступним прийнятим символом. Якщо потрібно, модуль UART може згенерувати переривання при успішному визначені швидкості і при виявлені тайм–ауту.

4.6.5.3 Передача даних

Після ініціалізації модуля UART дані можна передавати, записуючи їх в регістр зберігання передавача (THR). Аналогічно, дані можна приймати, зчитуючи їх з регістра буфера приймача (RBR). Насправді обидва регістри мають однакову адресу в пам'яті. Запис за цією адресою записує символ в FIFO–буфер передачі, а читання за цією адресою завантажує символ з FIFO–буфера прийому. Нижче наведено приклади підпрограм, які здійснюють передачу і прийом символів в/з UART.

```
Int putchar (int ch)        /* Записує символ в послідовний порт */
{
if {ch == '\n')
{
while (!(U0LSR & 0x20));
U0THR = CR;                 /* Виводимо CR */
}
while (!(U0LSQ & 0x20)) ;
return {U0THR = ch);
}
Int getchar (void)  /* Читає символ з послідовного порту */
while (!(U0LSR & 0x01));
return (U0RBR);
}
```

Функції putchar() и getchar() призначено для читання/запису одного символу з/в UART. Ці низькорівневі функції використовуються функціями бібліотеки STDIO компілятора Keil, такими як printf () та scanf (). Так що, якщо ви хочете перенаправити стандартне введення/виведення з UART, скажімо, на клавіатуру і ЖК–дисплей, перепишіть ці функції для підтримки прийому і передачі одиночних символів з/в необхідного пристрою введення/виведення. Обидві функції putchar () та igetchar () зчитують регістр стану лінії (LSR) для

виявлення помилок обміну за UART, а також для контролю стану FIFO–буферів прийому і передачі (рисунок 4.27).

Регістр LSR містить набір прапорців, які відображають стан UART. Їх стан можна відслідковувати в циклі або ж зчитувати у підпрограмі обробки переривання від UART.

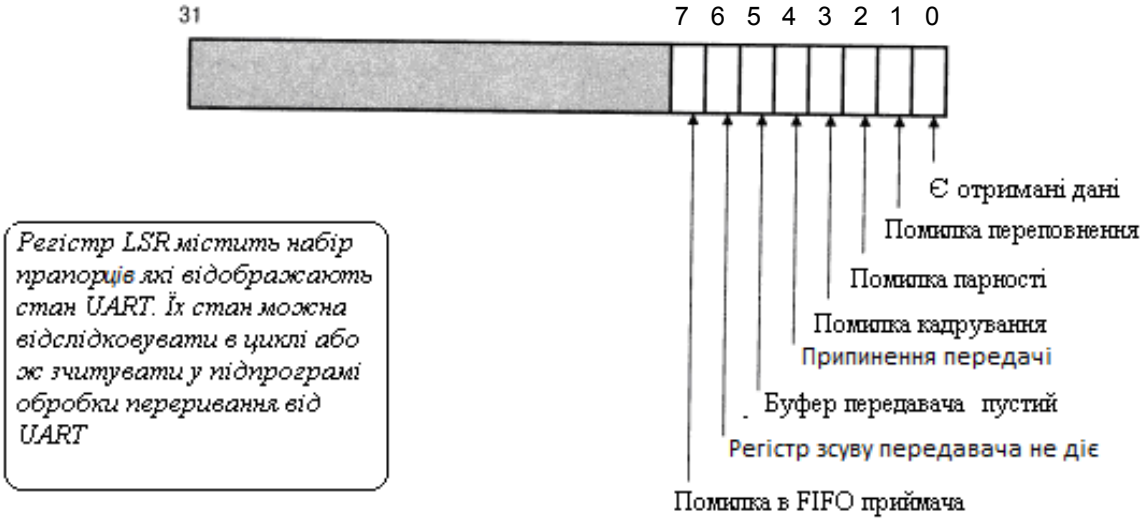


Рисунок 4.27 – Регістр стану лінії UART

У контролері переривань модулю UART виділено один канал, при цьому переривання може генеруватися трьома джерелами. По–перше, переривання UART може генеруватися при зміні стану каналу прийому. Тобто, переривання буде згенеровано при виникненні будь–якої помилки, для визначення якої можна прочитати регістр LSR. Решта два джерела переривань – це переривання приймача і передавача. Переривання за прийомом ініціюється символами, які поміщаються в FIFO–буфер прийому (рисунок 4.28). Глибина буфера, при якій буде генеруватися переривання, задається в регістрі керування FIFO.

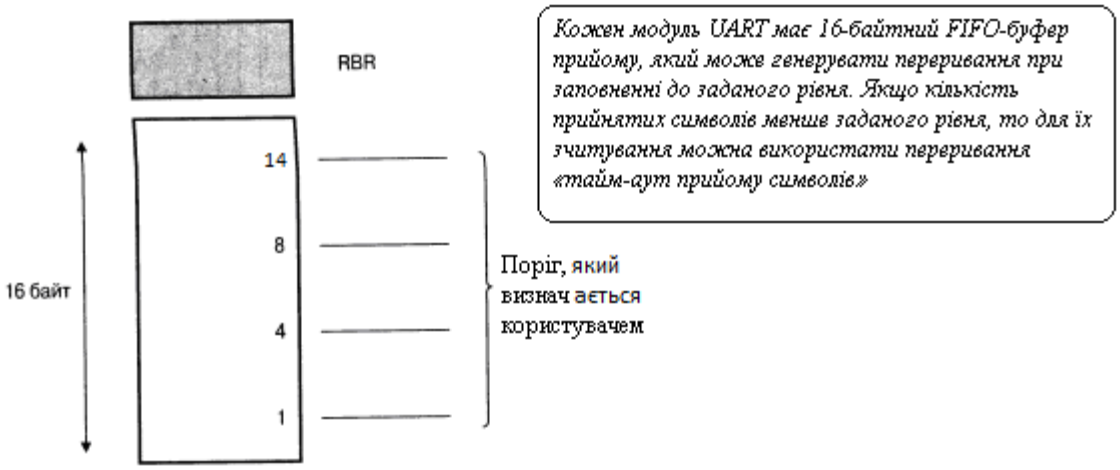


Рисунок 4.28 – FIF0–буфер прийому

Кожен модуль UART має 16–байтний FIFO–буфер прийому, який може генерувати переривання при заповненні до заданого рівня. Якщо кількість прийнятих символів менше заданого рівня, то для їх зчитування можна використати переривання «тайм–аут прийому символів».

Переривання за прийомом може генеруватися після прийому 1, 4, 8 або 14 символів. Нехай задана генерація переривання при появі в буфері восьми символів. Тоді при прийомі 34–байтного пакета буде згенеровано чотири переривання, і два символи залишаться в FIFO–буфері. Ці символи викличуть переривання «тайм–аут прийому символів» (CTI).

Переривання CTI виникає, якщо в FIFO–буфері є один або більше символів і стан буфера не змінюється протягом інтервалу, який дорівнює часу передачі 3,5 ... 4,5 символів.

FIFO–буфер передачі (рисунок 4.29) генерує переривання при спустошенні регістра зберігання передавача, а також при спустошенні регістра зсуву передавача.

Як і FIFO–буфер прийому, FIFO–буфер передачі має глибину 16 байт. FIFO–буфер передачі може генерувати переривання при спустошенні та при завершенні передачі.

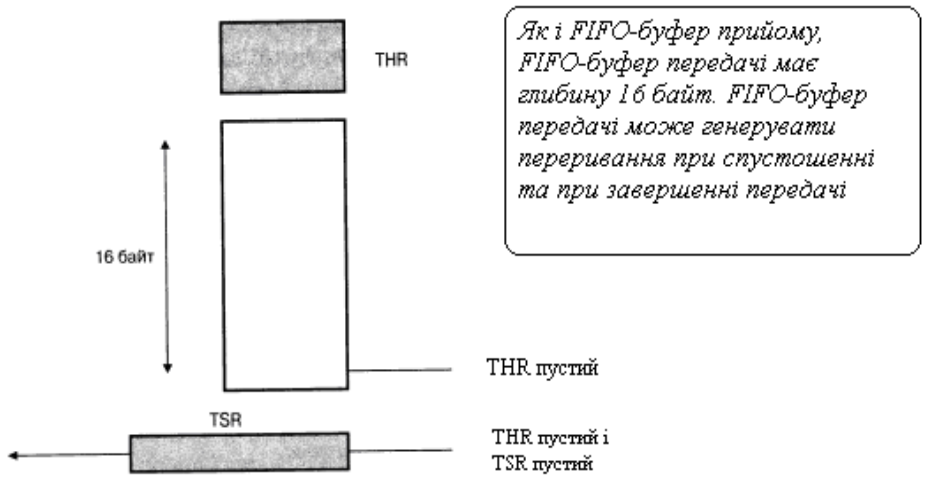


Рисунок 4.29– FIFO–буфер передачі

Модуль UART1 має таку ж базову структуру, що і UART0, однак до того ж він дозволяє керувати модемом (рисунок 4.30).

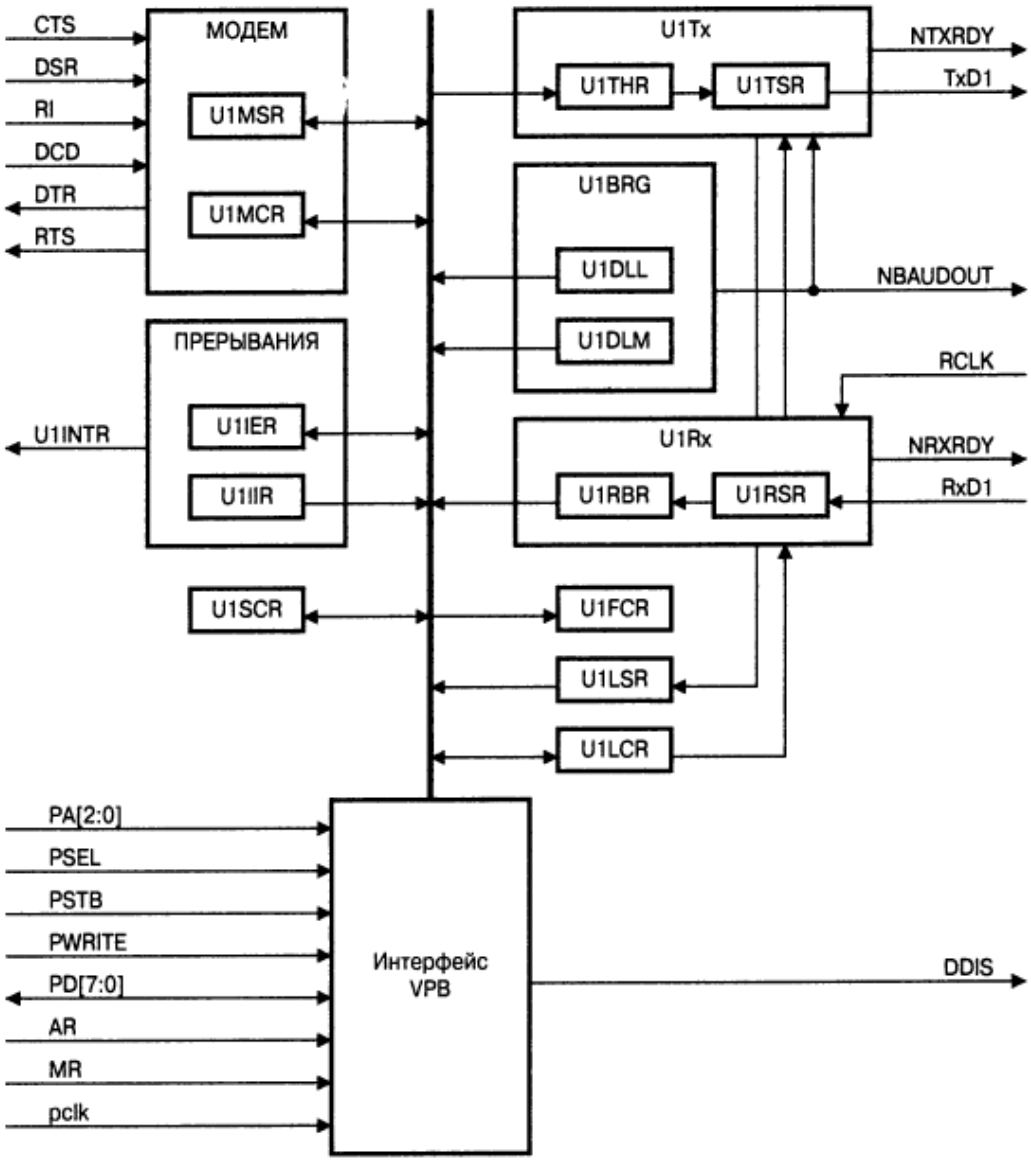


Рисунок 4.30 – Спрощена архітектура модуля UART1

Для підтримки повного модемного інтерфейсу (CTS, DCD, DSR, DTR, RI, RTS) використовуються додаткові виводи мікроконтролера, а в модулі є два додаткових регістри (регістр керування модемом і регістр стану модему) і додаткове джерело переривання, що генерує переривання при зміні стану модему (рисунок 4.31).

Ці додаткові можливості забезпечують оптимальне підключення до модему з генерацією переривання при кожній зміні регістра стану модему.

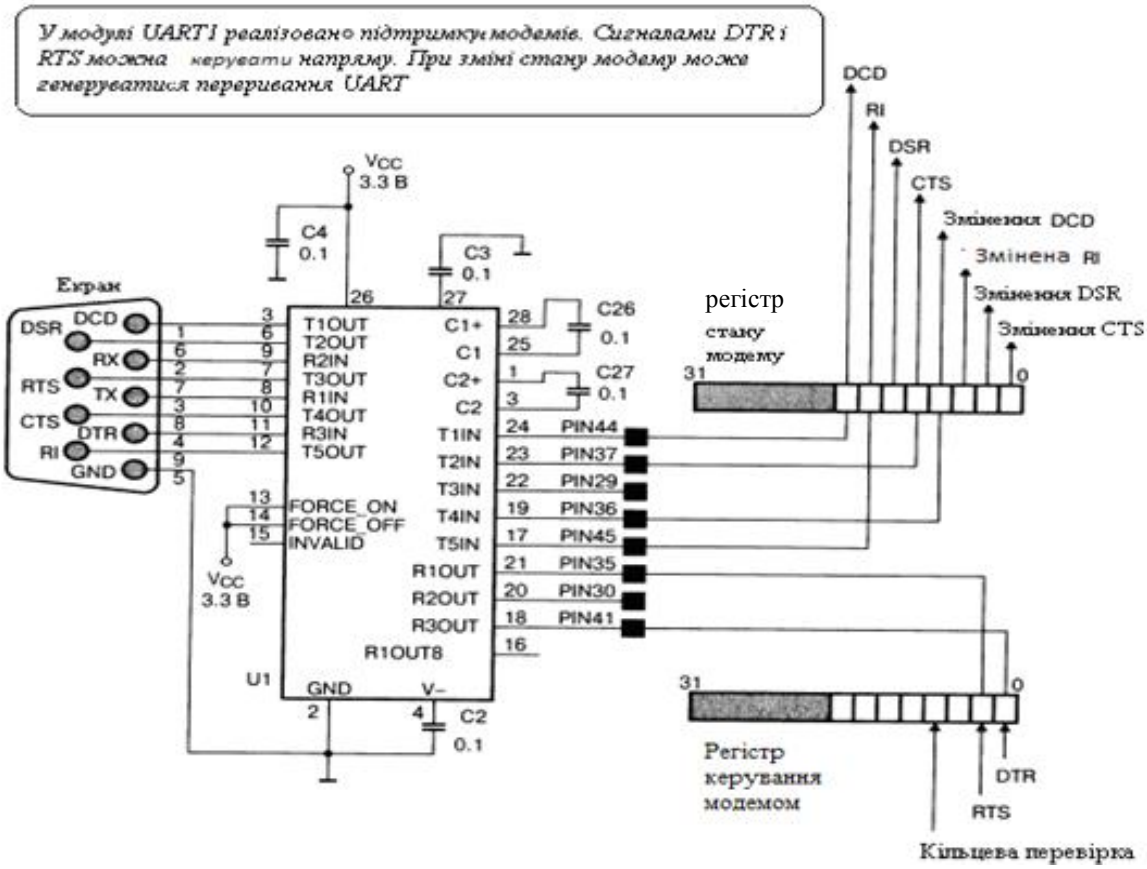


Рисунок 4.31 – Керування модемом модуля UART1

4.6.5.4 Організація обміну за протоколом IrDA

В модулі UART3 присутній додатковий керуючий регістр, який дозволяє керувати формуванням імпульсів на виході передавача в відповідності з інфрачервоним стандартом IrDA (рисунок 4.32).

Регістр керування IrDA використовується для керування модулем UART3 в відповідності до стандартів IrDA. При цьому є можливість інвертування імпульсів, за допомогою яких передаються дані. Тривалість імпульсів можна задавати, як стандартну (3/швидкість обміну), або кратну періоду сигнал Pclk.

31 6	5 3	2	1	0
	Дільник	Дозвіл імпульсів фіксованої тривалості	Інверсний режим	Дозвіл IrDA

Рисунок 4.32 – Регістр керування IrDA

4.6.5.5 Опис керуючих регістрів UART

Карту регістрів UART0 представлено в таблиці 4.15.

Таблиця 4.15 – Карта регістрів UART0

Назва	**Опис**	**Доступ**	**Значення після скидання**	**Адреса**
U0RBR	Регістр буфера прийому	RO	NA	0xE000C000 DLAB = 0
U0THR	Регістр буфера передачі	WO	NA	0xE000C000 DLAB = 0
U0IER	Регістр дозволу переривання	R/W	0	0xE000C004 DLAB = 0
U0IIR	Регістр ідентифікації переривання	RO	0x01	0xE000C008
U0FCR	Регістр керування буфером FIFO	WO	0	0xE000C008
U0LCR	Регістр керування лінією	R/W	0	0xE000C00C
U0LSR	Регістр стану лінії	RO	0x06	0xE000C014
U0SCR	Робочий регістр клавіатури	R/W	0	0xE000C01C
U0DDL	Молодший байт коефіцієнта подільника	R/W	0x01	0xE000C000 DLAB = 1
U0DLM	Старший байт коефіцієнта подільника	R/W	0	0xE000C004 DLAB = 1

4.6.5.6 Регістр U0RBR

UART0_RBR

Опис	Доступ	Скинуте значення	Адреса
Приймальний буферний регістр	RO	невизначене	$0\times$E000C000

U0RBR є вищим байтом UART0 Rx FIFO. Вищий байт з Rx FIFO містить "найстаріший" отриманий символ і може бути прочитаний через інтерфейс шини. LSB (біт 0) представляє "найстаріший" отриманий біт. Якщо отриманий символ менше 8 біт, то не використовувані MSB заповнюються нулями.

4.6.5.7 Регістр U0DLL

UART0_DLL

Опис	Доступ	Скинуте значення	Адреса
Защіпка подільника (LSB)	R/W	0×01	$0\times$E000C000

Опис бітів регістра U0DLL представлено в таблиці 4.16

Таблиця 4.16– Опис бітів регістра U0DLL

U0DLL	Функції	Опис	Скинуте значення
7:0	Регістр LSB подільника	Регістр, що містить молодший байт подільника, який разом з регістром U0DLM задає швидкість обміну через UART0	0x01

4.6.5.8 Регістр U0DLM

UART0_DLM

Опис	Доступ	Скинуте значення	Адреса
Защіпка подільника MSB	R/W	0	0×E000C004

Опис бітів регістра U0DLM предсталено в таблиці 4.17

Таблиця 4.17 – Опис бітів регістра U0DLM

U0DLM	Функції	Опис	Скинуте значення
7:0	Регістр MSB подільника	Регістр, що містить старший байт подільника, який разом з регістром U0DLL задає швидкість обміну через UART0	0

16–розрядний подільник є частиною генератора–контролера швидкості передачі UART і зберігає значення, яке використовується для формування тактового сигналу модуля UART з сигналу Pclk, який повинен бути в 16 разів більше потрібної швидкості обміну BAUD. Регістри U0DLL і U0DLM утворюють 16 бітний подільник, де U0DLL містить нижні 8 біт, а U0DLM верхні 8 біт дільника.

4.6.5.9 Регістр U0IER

UART0_IER

Опис	Доступ	Скинуте значення	Адреса
Регістр дозволу переривання	R/W	0	0×E000C004

U0IER дозволяє переривання UART0. Опис бітів регістра U0IER описано в таблиці 4.18.

Таблиця 4.18 – Опис бітів регістра U0IER

U0IER	Функції	Опис	Скинуте значення
0	Дозвіл переривання "прийняті дані доступні" (RBR)	0: заборонити переривання. 1: дозволити переривання за наявністю прийнятих даних	0
1	Дозвіл переривання "буфер регістра передавача порожній"	0: заборонити переривання. 1: дозволити переривання, коли регістр зберігання передавача порожній	0
2	Дозвіл переривання "за станом лінії прийому Rx"	0: заборонити переривання за станом лінії прийому. 1: дозволити переривання за станом лінії прийому	0
	Зарезервовані	Програмне забезпечення користувача не повинно записувати данні у зарезервовані біти	N/A

4.6.5.10 Регістр U0IIR

UART0_IIR

Опис	Доступ	Скинуте значення	Адреса
Регістр ідентифікації переривання	RO	0x01	0×E000C008

U0IIR ідентифікує переривання UART0. Опис бітів регістра U0IIR предсавлено в таблиці 4.19

Таблиця 4.19 – Опис бітів регістра U0IIR

U0IIR	Функції	Опис	Скинуте значення
0	Наявність перери-вання	0: наявність переривання. 1: відсутність переривання.	1
3:1	ID пере-ривання	011: переривання за станом лінії прийому. 010: прийняті дані доступні". 110: тайм–аут індикації прийнятого сим-волу(CTI) 001: буфер передавача порожній. Всі інші варіанти зарезервовані.	0
5:4	Зарезерво-вані	Програмне забезпечення користувача не повинно записувати дані у зарезервовані біти	N/A
7:6	Дозвіл бу-фера FIFO	Ці біти еквівалентні біту U0FCR[0]	0

4.6.5.11 Регістр U0FCR

UART0_FCR

Опис	Доступ	Скинуте значен-ня	Адреса
FIFO регістр ке-рування	R/W	0	$0\times$E000C008

U0FCR керує діями UART0 Rx та Tx FIFO. Опис бітів регістра U0FCR представлено в таблиці 4.20.

Таблиця 4.20 – Опис бітів регістра U0FCR

U0FCR	Функції	Опис	Скинуте значення
0	Дозвіл буфера FIFO	0: забороняє приймальний та передавальний буфери FIFO. 1: дозволяє приймальний та передавальний буфери FIFO.	0
1	Скидання приймального буфера FIFO	Запис логічної 1 в цей біт очищає всі байти в приймальному буфері FIFO UART0 і скидає логічну схему вказівника	0
2	Скидання передавального буфера FIFO	Запис логічної 1 в цей біт очищає всі байти в передавальному буфері FIFO UART0 і скидає логічну схему вказівника	0
5:3	Зарезервовані	Програмне забезпечення користувача не повинно записувати данні у зарезервовані біти	N/A
7:6	Вибір рівня спрацьовування тригера приймаль ного буфера FIFO	00: рівень тригера 0 (за замовчуванням прийнято 1 символ). 01: рівень тригера 1 (за замовчуванням прийнято 4 символи). 10: рівень тригера 2 (за замовчуванням прийнято 8 символів). 11: рівень тригера 3 (за замовчуванням прийнято 14 символів). Ці біти визначають, скільки символів буде записано в приймальний буфер FIFO UART0 перед тим, я буде активовано переривання.	0

4.6.5.12 Регістр U0LCR

UART0_LCR

Опис	Доступ	Скинуте значення	Адреса
Регістр керування лінією	R/W	0	0×E000C00C

U0LCR визначає формат інформаційних символів, які передаються або отримуються. Опис бітів регістра U0LCR представлено в таблиці 4.21.

Таблиця 4.21 – Опис бітів регістра U0LCR

UOLCR	**Функції**	**Опис**	**Скинуте значення**
1:0	Вибір довжини слова	00: довжина слова – 5 біт. 01: довжина слова – 6 біт. 10: довжина слова – 7 біт. 11: довжина слова – 8 біт.	0
2	Вибір кількості стопових бітів	0: 1 стоповий біт 1: 2 стопових біта (1,5, якщо U0LCR[1:0] = 00)	0
3	Дозвіл контролю парності	0: відключення генерації контролю парності та його перевірки 1: дозвіл генерації контролю парності та його перевірки	0
5:4	Вибір режиму контролю парності	00: доповнення та перевірка на непарність 01: доповнення та перевірка на парність 10: доповнення до непарності перевірка на парність 11: доповнення до парності перевірка на непарністі (див. таблицю 4.22)	0
6	Керування припиненням передачі	0: заборонити припинення передачі, 1: дозволити припинення передачі	0
7	Доступ до регістра подільника	0: заборонити доступ до регістра подільника 1: дозволити доступ до регістра подільника	0

Вплив розрядів D5, D5 на вид перевірки при прийомі описано в таблиці 4.22

Таблиця 4.22 – Вплив розрядів D5, D5 на вид перевірки при прийомі

D3	D4	D5	Доповнення при передачі до:	Перевірка при прийомі на:
0	*	*	Ні	Ні
1	0	0	Непарного числа	Непарне число
1	1	0	Парного числа	Парне число
1	0	1	Непарного числа	Парне число
1	1	1	Парного числа	Непарне число

Коли біт припинення передачі дорівнює логічній 1, то послідовний вихід (SOUT) встановлюється в стан пропуску (логічного 0) і, незалежно від інших сигналів керування передачею, знаходиться в цьому стані до скидання біта 6 в початкове нульове значення. Ця особливість дозволяє МП–ру керувати процесом підключення терміналу (модему) до системи зв'язку.

4.6.5.13 Регістр U0LSR

UART0_LSR

Опис	Доступ	Скинуте значення	Адреса
Регістр стану лінії	R0	0×60	$0\times$E000C014

UOLSR – регістр використовується тільки для читання та забезпечує інформацію про стан UART0 Rx та Tx блоків. Біти регістра U0LSR описано в таблиці 4.23.

Модуль UART2 та UART3 мікроконтролера LPC2378 програмуються аналогічно.

Таблиця 4.23 – Опис бітів регістра U0LSR

UOLSR	Функції	Опис	Скинуте значення
0	Дані прийма-мача готові	0: U0RBR пустий. 1: U0RBR містить достовірні дані. Скидання біта U0LSR[0] в стан логічного 0 проводиться або при читанні мікропроцесором даних з буферного регістра приймача, або при програмному запису логічного 0 в цей біт.	0
1	Помилка переповнення	0: стан помилки переповнення неактивний. 1: стан помилки переповнення активний. Біт U0LSR[1] встановлюється, якщо на приймач U0RSR UART0 поступив новий символ, а приймальний буфер UART0 повний. Біт скидається при зчитуванні регістра U0LSR. При цьому відбувається втрата попереднього прийнятого символу.	0
2	Помилка парності	0: стан помилки парності неактивний. 1: стан помилки парності активний. Біт U0LSR[2] встановлюється, якщо біт парності отриманого символу знаходиться в неправильному стані. Біт скидається при зчитуванні регістра U0LSR.	0
3	Помилка кадрової синхронізації	0: стан помилки кадрової синхронізації неактивний. 1: стан помилки кадрової синхронізації активний. Біт U0LSR[3] встановлюється, якщо стоп–біт отриманого символу дорівнює логічному 0. Біт скидається при зчитуванні регістра U0LSR.	0
4	Припинення передачі (BI)	0: стан припинення передачі неактивний. 1: стан припинення передачі активний. Біт U0LSR[4] встановлюється в стан логічної 1, якщо вхід даних, що приймаються, утримується в стані пропуску (логічний 0) впродовж часу, що перевищує передачу: старт–біта, бітів даних, паритету і стоп–біта/стоп–бітів. Біт скидається при зчитуванні регістра U0LSR.	0

Продовження таблиці 4.23

4	Припинення передачі (BI)	0: стан припинення передачі неактивний. 1: стан припинення передачі активний. Біт U0LSR[4] встановлюється в стан логічної 1, якщо вхід даних, що приймаються, утримується в стані пропуску (логічний 0) впродовж часу, що перевищує передачу: старт–біта, бітів даних, паритету і стоп–біта/стоп–бітів. Біт скидається при зчитуванні регістра U0LSR.	0
5	Буфер передавача пустий (THRE)	0: U0THR містить достовірні данні. 1: U0THR пустий. Біт U0LSR[5] встановлюється, якщо регістр U0THR UART0 пустий і скидається при записі даних в регістр U0THR .	1
6	Регістр зсуву передавача не діє (TEMT)	Цей біт є індикатором того, що регістр зсуву передавача не діє (не працює). Він скидається в стан логічного 0 при записі даних з регістра зберігання передавача в регістр зсуву передавача. D6 може бути доступний тільки при читанні. Біт скидається при записі даних в регістр U0THR	1
7	Помилка приймального буфера FIFO (RxFE)	0: U0RBR не містить символи з помилками прийому через UART0 або біт U0FCR[0] = 0. 1: UART0 U0RBR містить символи с помилками прийому через UART0. Біт U0LSR[7] встановлюється, якщо символ з помилкою прийому поміщається в регістр U0RBR . Біт скидається при зчитуванні регістра U0LSR.	0

Нижче розглядаються відмінності, які мають керуючі регістри модуля UART1. Карта регістра UART1 представлена в таблиці 4.24.

Таблиця 4.24 – Керуючі регістри модуля UART1

Назва	**Опис**	**Доступ**	**Значення після скидання**	**Адреса**
U1RBR	Регістр буфера прийому	RO	NA	0xE0010000 DLAB = 0
U1THR	Регістр буфера передачі	WO	NA	0xE0010000 DLAB = 0

Продовження таблиці 4.24

U1IER	Регістр дозволу переривання	R/W	0	0xE0010004 DLAB = 0
U1IIR	Регістр ідентифікації переривання	RO	0x01	0xE0010008
U1FCR	Регістр керування буфером FIFO	WO	0	0xE0010008
U1LCR	Регістр керування лінією	R/W	0	0xE001000C
U1MCR	Регістр керування модемом	R/W	0	0xE0010010
U1LSR	Регістр стану лінії	RO	0x06	0xE0010014
U1MSR	Регістр стану модему	RO	0	0xE0010018
U1SCR	Робочий регістр клавіатури	R/W	0	0xE001001C
U1DDL	Молодший байт коефіцієнту подільника	R/W	0x01	0xE0010000 DLAB = 1
U1DLM	Старший байт коефіцієнту подільника	R/W	0	0xE0010004 DLAB = 1

4.6.5.14 Регістр U1MCR

UART1_MCR

Опис	**Доступ**	**Скинуте значення**	**Адреса**
Регістр керування модемом	R/W	0	0xE0010010

U1MCR – регістр керування модемом UART1. Біти регістра U1MCR описано в таблиці 4.25

Таблиця 4.25– Опис бітів регістра U1MCR

U1MCR	Функції	Опис
0	Керування станом виходу DTR (запит передавача терміналу – ЗПДТ)	Керує станом виходу DTR. Читання цього біта повертає значення логічного 0, якщо активовано режим модемного шлейфу (петлі), в якому відбувається взаємне “замикання” сигналів модемного інтерфейсу (виходи замикаються на виходи)
1	Керування станом виходу RTS (запит приймача терміналу – ЗПРМТ)	Керує станом виходу RTS. Читання цього біта повертає значення логічного 0, якщо активовано режим модемного шлейфу (петлі), в якому відбувається взаємне “замикання” сигналів модемного інтерфейсу (виходи замикаються на входи)
3:2	Зарезервовані	Програмне забезпечення користувача не повинно записувати данні у зарезервовані біти
4	Вибір режиму “шлейфу” (петлі)	0: заборонити режим модемного “шлейфу”. 1: дозволити режим модемного “шлейфу”. В режимі модемного “шлейфу” відбувається діагностичний тест логічних схем нтерфейсу. При цьому, дані з виходу передавача подаються в послідовному вигляді на вхід приймача. В цьому режимі вхід RxD1 не задіяний, а вихід TxD1 утримується в стані 1. Чотири керуючих сигнали від модема (CTS, DSR, RI, DCD) відключаються від виводів мікроконтролера. Керуючі сигнали для модема (RTS, DTR) підключаються до відповідних входів від модема. В результаті такої комутації, значення старших 4 бітів регістра U1MSR

Продовження таблиці 4.25

		визначається значеннями молодших чотирьох бітів U1MCR.
7:5	Зарезервовані	Програмне забезпечення користувача не повинно записувати дані у зарезервовані біти

4.6.5.15 Регістр U1MSR

UART1_MSR

Опис	Доступ	Скинуте значення	Адреса
Регістр стану модему	R0	0	0xE0010018

U1MSR – регістр тільки для читання, який забезпечує інформацію про стан модему UART1. Біти регістра U1MSR описано в таблиці 4.26

Таблиця 4.26– Опис бітів регістра U1MSR

U1MSR	Функції	Опис	Скинуте значення
0	Зміна сигналу на вході CTS	0: на вході CTS від модему не було виявлено змін. 1: на вході CTS від модему було виявлено зміни. Біт скидається при читанні регістра U1MSR	0

Продовження таблиці 4.26

1	Зміна сигналу на вході DSR	0: на вході DSR від модему не було виявлено змін. 1: на вході DSR від модему було виявлено зміни. Біт скидається при читанні регістра U1MSR	0
2	Зміна сигналу на вході RI	0: на вході RI від модему не було виявлено змін. 1: на вході RI від модему було виявлено зміни. Біт скидається при читанні регістра U1MSR	0
3	Зміна сигналу на вході DCD	0: на вході DCD від модему не було виявлено змін. 1: на вході DCD від модему було виявлено зміни. Біт скидається при читанні регістра U1MSR	0
4	Готовність приймача терміналу (ГПРМТ) CTS	Відображає значення вхідного сигналу CTS. В режимі модемного шлейфу значення цього біта визначає стан біта U1MCR[1]	0
5	Готовність передавача терміналу (ГПДТ) DSR	Відображає значення вхідного сигналу DSR. В режимі модемного шлейфу значення цього біта визначає стан біта U1MCR[0]	0

Продовження таблиці 4.26

6	RI	Відображає значення вхідного сигналу RI. В режимі модемного шлейфу значення цього біта визначає стан біта U1MCR[2]	0
7	DCD	Відображає значення вхідного сигналу DCD. В режимі модемного шлейфу значення цього біта визначає стан біта U1MCR[3]	0

4.6.5.16 Налаштування швидкості обміну інформації між МК–м ARM7 та інтерфейсом УАПП

Розглянемо приклад написання мовою Асемблер фрагменту програми, який забезпечує потрібну швидкість обміну інформацією між МК–м ARM7 та інтерфейсом УАПП TL16C450/550; розрахувати значення коефіцієнта ділення подільника інтерфейсу та відносну похибку отриманої швидкості від заданої.

Вихідні дані: швидкість обміну Vпд = 9600 бод; частота тактового генератора шини VPB мікроконтролера: P_{CLK}=30 МГц.

Величина швидкості передачі даних Vпд визначається виразом:

$$V_{пд} = P_{CLK}/(k_{діл}*16),$$

де $k_{діл}$ – значення коефіцієнта ділення, яке повинно програмно завантажуватися у двобайтний регістр подільника УАПП, P_{CLK} – тактовий сигнал шини VPB мікроконтролера [4].

При Vпд = 9600 бод та P_{CLK} = 30 МГц $k_{діл}$=195=0x00C3, тоді

Vпд. реальна = 9615,39 бод.

Відносна похибка

$$\delta_{відн} = \frac{V_{пд\,реальна} - V_{пд}}{V_{пд}} * 100\% = \frac{9615{,}39 - 9600}{9600} * 100\% = 0.16\%.$$

Для програмування заданої швидкості необхідно завантажити:

молодший байт $k_{діл} = C3h$ в молодший байт подільника (U0DLL), а старший байт $k_{діл} = 00h$ в старший байт подільника (U0DLM) модуля УАПП МК–ра.

Для написання фрагменту програми потрібно знати адресу регістра подільника УАПП, та керуючого регістра U0LCR, які дорівнюють:

0xE000C00C – для регістра U0LCR;

0xE000C000 – для регістра U0DLL;

0xE000C004 – для регістра U0DLM.

Команди мікроконтролера ARM7, які використовують безпосередній операнд, мають декілька особливостей. Якщо довжина операнда дорівнює не більше восьми біт, то цей операнд без проблем компілюється і займає молодший байт машинного коду команди. Довжина безпосереднього операнда може складати до 32–х розрядів, але інформативна частина цього операнда повинна займати дві сусідні тетради (1 байт). Цей 32–х розрядний операнд при компіляції займає 12 бітів машинного коду команди, з яких вісім молодших – це інформативна частина, а наступні 4 розряди – це значення циклічного зсуву (максимальне значення – 16). При виконанні такої команди процесор циклічно зсуває інформативний байт вправо на величину, яку записано у 4–х розрядний зсув та помножену на 2 (максимальне значення: 16·2=32).

В результаті отримується початковий 32–х розрядний безпосередній операнд.

Для звернення до регістрів подільника U0DLL та U0DLM треба встановити біт DLAB у 1. DLAB – це старший розряд регістра керування лінією U0LCR.

Для встановлення DLAB в 1 програма містить 6 команд:

```
MOV R1, #0xE0000000   ;  R1 ← #0xE0000000,
MOV R2, #0x0000C000   ; R2 ← #0x0000C000,
ADD R1, R1, R2        ; R1 ← R1+R2,
ADD R1, R1, #0x0C     ; R1 ← R1+0x0C,
MOV R2, #0x80         ; R2 ← #0x80,
STRB R2, [R1]         ; M (R1)←R2.
```

В останній команді вміст регістра R2, який дорівнює 0x80, завантажується у пам’ять з адресою регістра U0LCR, яка дорівнює #0xE000C00C. Після цього наступні одинадцять команд завантажують в регістр подільника потрібний коефіцієнт ділення.

Завантаження молодшого байта:

регістр U0DLL ← МБ $k_{діл} = C3h$;

```
MOV R1, #0xE0000000        ; R1 ← #0xE0000000,
MOV R2, #0x0000C000        ; R2 ← #0x0000C000,
ADR R1, R1, R2             ; R1← R1+R2,
MOV R2, #0xC3              ; R2 ← 0xC3,
STRB R2, [R1]              ; M(R1) ← R2.
```

Завантаження старшого байта:

регістр U0DLM ← СБ $k_{діл} = 00h$;

```
MOV R1,#0xE0000000         ; R1 ← 0xE0000000,
MOV R2,#0x0000C000         ; R2 ← 0x0000C000,
ADD R1, R1, R2             ; R1 ← R1+R2,
ADD R1, R1, #0x04          ; R1 ← R1+0x04,
MOV R2, #0x00              ; R2 ← 0x00,
STRB R2, [R1]              ; M(R1)←R2.
```

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Опишіть структурну схему сполучення МП/МК з модемом за допомогою інтерфейсу RS – 232C.
2) Скільки модулів УАПП мають мікроконтролери сімейства LPC2300?
3) Які особливості мають модулі UART1 та UART3?
4) Назвіть основні керуючі регістри модуля UART.
5) Як виконується ініціалізація модуля UART?
6) Як формується тактовий сигнал модуля UART?
7) Як здійснюється керування функцією автоматичного визначення швидкості передачі пакету даних, який приймається?
8) В який регістр модуля UART записуються дані для передачі?
9) Прапорці якого регістра відображають стан модуля UART?
10) Як програмується потрібна швидкість обміну в модулі UART?
11) Які переривання може генерувати модуль UART?
12) Поясніть особливості організації обміну за протоколом IrDA.
13) Опишіть призначення бітів регістра U0RBR.
14) Опишіть призначення бітів регістрів U0DLL та U0DLM.
15) Опишіть призначення бітів регістрів U0IER та U0IIR.
16) Опишіть призначення бітів регістрів U0LCR та U0LSR.
17) Опишіть призначення бітів регістрів U1MCR та U1MSR.
18) Поясніть призначення основних контактів роз’єму RS–232C.
19) При якому значенні біта DLAB регістра LCR виконується програмування швидкості обміну?
20) Опишіть призначення та програмування FIF0–буферів передачі та прийому.
21) Які помилки прийому може виявляти модуль UART?
22) Як біти керуванням модемом можуть використовуватися для синхронізації обміну між мікропроцесором та терміналом (модемом)?

4.7 Інтерфейс I^2C

4.7.1 Загальна характеристика інтерфейсу I^2C

Інтерфейс I^2C (InterIC, або IIC) є двопровідним послідовним синхронним інтерфейсом, який розроблено фірмою “Philips Corporation”, і призначено для зв'язку між інтегральними мікросхемами або модулями. Існує ціла група I^2C–сумісних пристроїв для різних додатків: цифро–аналогові та аналого–цифрові перетворювачі, мікросхеми пам'яті, мікроконтролери, що містять модуль I^2C, і т. ін.

Шина інтерфейсу I^2C складається із двох ліній:

– двонаправленої лінії даних (SDA);
– лінії тактових (синхро) імпульсів (SCL).

На рисунку 4.33 наведена структурна схема типової мережі, що використовує для обміну даними інтерфейс (шину) I^2C.

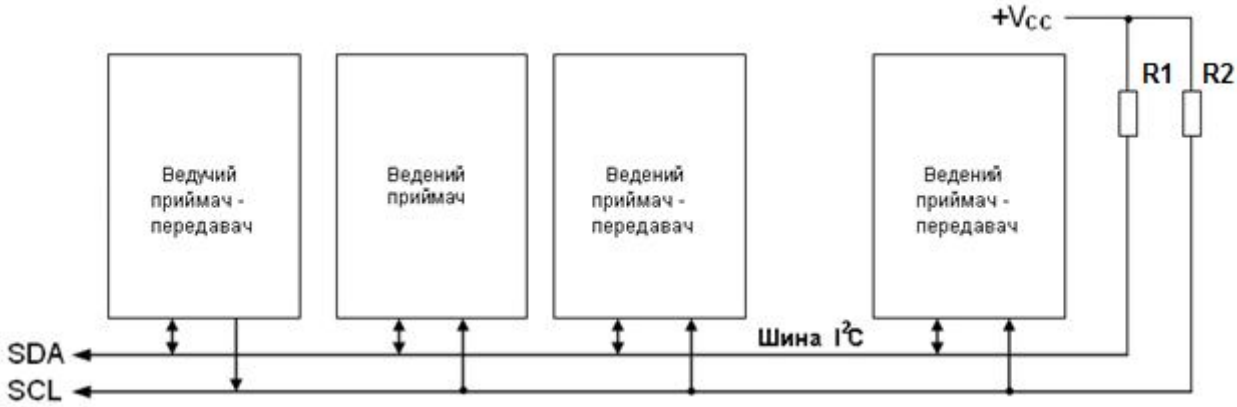


Рисунок 4.33 – Структурна схема мережі з інтерфейсом I^2C

Лінії шини SDA і SCL з'єднані з додатним полюсом напруги живлення ($+V$cc) через підтягуючі резистори $R1$ та $R2$.

Передавач генерує і передає повідомлення, а приймач його приймає.

Один із двох пристроїв, які беруть участь у обміні, є ведучим (master), а інший – веденим (slave). Ведучий пристрій керує роботою шини та формує тактові сигнали (синхросигнали) SCL.

Кожен пристрій, який використовує для обміну інтерфейс I^2C, має свою адресу. Коли ведучий пристрій бажає ініціювати обмін даними, він передає на лінію SDA адресу пристрою, з яким буде виконуватися обмін (передача/прийом).

Всі ведені пристрої слідкують за адресою, яка виставляється на шину, і порівнюють її із власною адресою. Після адреси ведучий передає біт напрямку $R/\overline{W}$, який визначає, чи буде ведучий читати дані від веденого ($R/\overline{W} = 1$), чи буде передавати дані веденому ($R/\overline{W} = 0$). Ведений приймач після одержання адреси або даних видає на шину SDA біт підтвердження (логічний нуль). Інтерфейс I^2C може використовувати два формати адреси:

– 7–ми бітну адресу;
– 10–ти бітну адресу.

У багатьох мікроконтролерах сімейства ARM використовується 7–ми бітна адреса.

На рисунку 4.34 наведений формат 7–ми бітної адреси.

S	A6	A5	A4	A3	A2	A1	A0	$R/\overline{W}$	$\overline{ACK}$

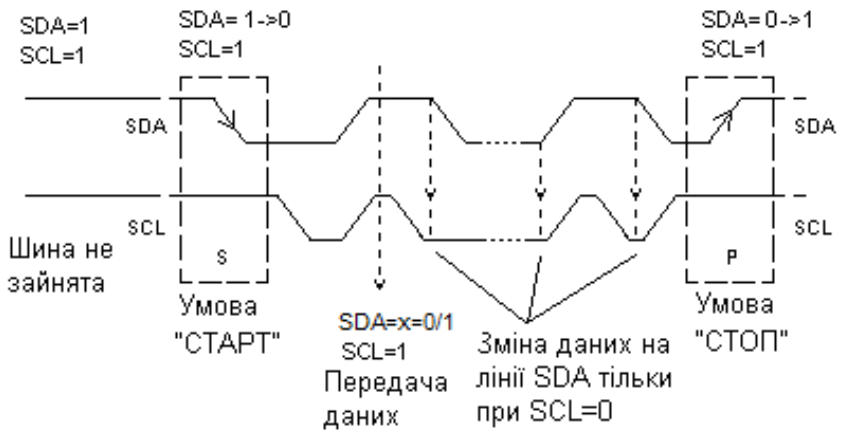
Рисунок 4.34 – Формат 7–бітної адреси

(*S* – старт, $R/\overline{W}$ – біт «читання/$\overline{запис}$», $\overline{ACK}$ – підтвердження)

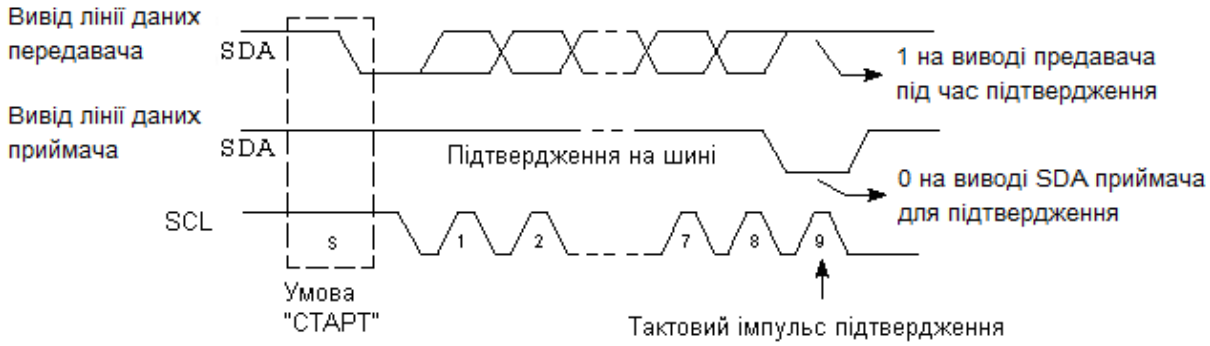
На рисунку 4.35 наведені часові діаграми, які відображають стани на шині (рисунок 4.35, а), а також пояснюють формування сигналу підтвердження (рисунок 4.35, б).

Наведені діаграми відображають наступні коректні стани сигналів на шині під час обміну даними:

– шина не зайнята: на обох лініях одиниці (SDA = SCL = 1);
– початок обміну даними: зміна сигналу на лінії даних SDA з одиниці в нуль при одиничному значенні сигналу на лінії SCL, що визначає умову початку обміну (умова «СТАРТ» – S);
– припинення передачі: зміна сигналу на лінії даних з нуля в одиницю при одиничному значенні сигналу на лінії SCL, що визначає умову закінчення обміну (умова «СТОП» – P);



а)



б)

Рисунок 4.35 – Стани на шині: старт – S, стоп – P, а) – передача даних; б) – формування сигналу підтвердження

– коректність даних: при одиничному значенні сигналу на лінії SCL стан лінії даних не повинен змінюватися, щоб не сформувати невірну умову СТАРТ або СТОП. Дані можна змінювати, якщо на лінії SCL присутній низький рівень сигналу (логічний нуль). На один біт інформації на лінії SDA міститься один тактовий імпульс на лінії SCL. Кожний цикл обміну даними починається умовою «СТАРТ» і закінчується умовою «СТОП». Кількість інформаційних бітів даних, що передаються між цими станами, необмежена. Дані передаються байтами. Приймач підтверджує одержання чергового байта, посилаючи біт підтвердження (логічний нуль) після прийому кожного байта;
– біт підтвердження: передається після прийому кожного байта даних або адреси. Активний передавач (ведений або ведучий) після передачі чергового байта формує на лінії SDA сигнал високого рівня. Ведучий

пристрій (приймач або передавач) формує на лінії SCL тактовий імпульс, а приймач (ведучий або ведений) видає на лінію даних SDA сигнал підтвердження низького рівня. Приймач, що генерує біт підтвердження, підключає лінію SDA до низького рівня і утримує її в цьому стані доти, доки тактовий імпульс лінії SCL не переключиться у стан низького рівня. Для припинення обміну приймач повинен залишити останній прийнятий байт без підтвердження, що автоматично викликає формування активним передавачем умови «СТОП».

Можливі чотири режими (типу) обміну даними для інтерфейсу I^2C.

1. Ведучий передавач: на вихід SDA передавача виводяться дані, а на лінію SCL видаються синхроімпульси. Перший переданий байт містить адресу веденого приймача (7 біт) і біт напрямку обміну даними $R/\overline{W} = 0$, що говорить про те, що буде проводитися запис (передача). Дані передаються послідовно по 8 біт. Після передачі чергового байта (адреси або даних), ведучий передавач очікує від веденого приймача біт підтвердження $\overline{ACK}$. Для завдання початку і кінця сеансу обміну даними ведучий передавач формує умови «СТАРТ» і «СТОП».

2. Ведучий приймач: спочатку сеансу обміну ведучий приймач передає на лінію SDA адресу веденого передавача (7 біт) і біт напрямку обміну $R/\overline{W} = 1$, що говорить про те, що ведучий буде здійснювати прийом. Ведучий приймач формує імпульси синхронізації, які передаються лінією SCL. Після прийому адреси ведений передавач виставляє на лінію SDA сигнал підтвердження $\overline{ACK}$, а потім передає дані. Дані від веденого передавача передаються послідовно по 8 біт лінією SDA. Після прийому чергового байта ведучий приймач виставляє на лінію SDA сигнал підтвердження $\overline{ACK}$. Умови СТАРТ і СТОП формуються ведучим пристроєм для вказівки початку і кінця сеансу обміну послідовними даними.

3. Ведений приймач: ведучий передавач видає на лінію SDA адресу веденого приймача й біт напрямку $R/\overline{W} = 0$, що говорить про те, що буде виконуватися запис (передача). На лінії SCL ведучий передавач видає синхроімпульси. Після одержання адреси ведений приймач передає сигнал

підтвердження $\overline{ACK}$, після чого ведучий передавач послідовно передає на лінію SDA дані. Ведений приймач після прийому чергового байта даних передає сигнал підтвердження $\overline{ACK}$, що надходить до ведучого передавача лінією SDA. Умови СТАРТ і СТОП формуються ведучим передавачем.

4. Ведений передавач: перший байт на шині SDA (адреса) приймається і обробляється веденим передавачем так само, як і в режимі веденого приймача. При цьому біт напрямку $R/\overline{W} = 1$, що говорить про те, що ведучий буде здійснювати прийом. Дані послідовно передаються лінією SDA від веденого передавача у той час, як синхроімпульси передаються лінією SCL від ведучого приймача. Після передачі кожного байта ведений передавач аналізує наявність на лінії SDA біта підтвердження $\overline{ACK}$, який передає ведучий приймач. Умови СТАРТ і СТОП формує ведучий приймач.

У підпорядкованому (веденому) режимі апаратні засоби інтерфейсу I^2C здійснюють пошук своєї власної підпорядкованої адреси або адреси загального виклику. Якщо отримується одна із цих адрес, відбувається запит на переривання і виконуються відповідні дії. Якщо модуль I^2C хоче захопити шину й стати ведучим, то він чекає, поки шина звільниться (SDA = SCL = 1). Можливе функціонування в якості веденого при цьому не переривається.

Два і більше пристроїв в I^2C можуть спробувати стати ведучими і одночасно згенерувати умову «СТАРТ». У цьому випадку здійснюється арбітраж шини в моменти, коли шина SCL перебуває у високому стані. Якщо один ведучий передає на лінію даних низький рівень, а інший – високий, то останній відключається від лінії, тому що стан шини SDA (низький) не відповідає високому стану внутрішньої шини даних пристрою, який бажає стати ведучим.

Якщо арбітраж шини загублений у головному (ведучому) режимі, то відповідний пристрій I^2C переключається у підпорядкований режим і може розпізнавати свою власну підпорядковану адресу.

На рисунку 4.36 наведені часові діаграми, які відображають обмін даними шиною I^2C.

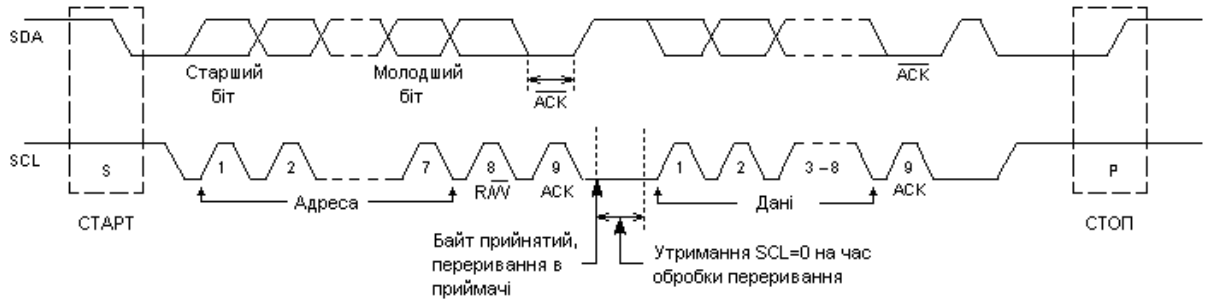


Рисунок 4.36 – Приклад обміну даними шиною I^2C

4.7.2 Типові особливості інтерфейсу I^2C мікроконтролерів сімейства LPC2300

Інтерфейс I^2C мікроконтролерів сімейства LPC2300 [1, 4, 7] має такі особливості:

- шинний інтерфейс стандарту I^2C з можливістю конфігурування як ведучого (Master), веденого (Slave) або ведучого/веденого;
- програмоване тактування, що дозволяє керувати швидкістю обміну;
- двонаправлена передача даних між ведучим і веденим;
- в режимі Multi–Master можливе одночасне підключення до шини I^2C декількох ведучих пристроїв, при цьому інтерфейс I^2C автоматично здійснює арбітраж з метою передачі послідовних даних без спотворень;
- шина I^2C може використовуватися для випробувальних і діагностичних цілей;
- основним функціональним призначенням інтерфейсу I^2C є зв'язок із зовнішніми периферійними пристроями з послідовним доступом, наприклад:
- пристроями пам'яті з послідовним доступом;
- рідкокристалічними індикаторами (РКІ) з вбудованими контролерами;
- таймером–календарем реального часу;
- програмованими логічними матрицями (ПЛМ) і т. ін.

Для ініціалізації інтерфейсу I^2C в програмі повинні бути присутніми наступні рядки:

```
VICVectCntl1   = 0x00000029;        // Обираємо слот для даного
                                        // переривання
VICVectAddr1   = (unsigned)I2CISR;  // Передаємо адресу обробника в
```

```
                                        // модуль VIC
VICIntEnable   = 0x00000200;        // Дозволяємо переривання

PINSEL0        = 0x50;              // Підключаємо виводи
                                        // мікроконтролера       до
                                    модуля
                                        // I2C
I2SCLH         = 0x08;              // Встановлюємо частоту
                                        // тактового        сигналу
                                    57.6 кГц
I2SCLL         = 0x08;
```

Щоб модуль I^2C реагував на всі події, що виникають на шині, він повинен бути відповідним чином сконфігурований. При цьому ми отримуємо периферійний пристрій, керований, в основному, за перериваннями. Отже, перше, що ми повинні зробити, це сконфігурувати векторний контролер переривань VIC (vectored interrupt controller) для роботи за перериванням від модуля I^2C. Потім необхідно сконфігурувати блок керування виводами, щоб підключити лінії даних і тактового сигналу модуля I^2C до виводів мікроконтролера. На завершення ми повинні задати швидкість передачі, програмуючи регістри I2SCLH і I2SCLL. В обох регістрах для зберігання використовуються тільки молодші 16 біт. Частота передачі бітів визначається за формулою [4, 7]:

$$\text{Bit Freq} = \text{Pclk} / (2 * (\text{I2SCLH} + \text{I2SCLL})), \text{ де}$$

Pclk — частота периферійного тактового сигналу.

У наведеному вище прикладі ініціалізації інтерфейсу схема ФАПЧ відключена, а частота резонатора дорівнює 14,7456 МГц. Тому частота передачі дорівнює:

$$\text{Bit Freq} = (14745600/8) /(2 * (8+8)) = 57600 \text{ біт/сек.}$$

Після конфігурування мікроконтролер може здійснювати обмін даними з іншими пристроями, зчитуючи і видаючи на шину дані в якості ведучого або приймаючи дані та відповідаючи на запити іншого ведучого пристрою.

4.7.3 Режими роботи інтерфейсу I^2C мікроконтролерів сімейства L3C2300

4.7.3.1 Режим ведучого передавача

У режимі ведучого передавача дані передаються від ведучого до веденого. Для ініціалізації цього режиму, в регістр керування I^2C (I2CONSET) необхідно заздалегідь записати значення, які наведено в таблиці 4.27.

Таблиця 4.27 – Ініціалізація регістра I2CONSET для режиму ведучого

Номер біта	Назва біта	Значення біта
0	–	–
1	–	–
2	AA	0
3	SI	0
4	STO	0
5	STA	0
6	I2EN	1
7	–	—

Біт (прапорець) I2EN регістра I2CONSET повинен бути встановлений, щоб дозволити роботу інтерфейсу I^2C. Біт (прапорець) AA, навпаки, слід скинути, щоб інтерфейс I^2C мікроконтролера не видавав в лінію біт «підтвердження» (ACK) у разі, коли ведучим на шині є інший пристрій. Таким чином, функціонування в режимі веденого блокується, і інтерфейс I^2C мікроконтролера буде функціонувати в режимі ведучого. Біти (прапорці) STA, STO і SI повинні бути скинуті. Біт SI скидається шляхом запису логічної 1 в біт SIC регістра I2CONCLR.

Перший байт (його передає ведучий), містить адресу веденого приймача (7 бітів) і біт напрямку подальшої передачі даних ($R/\overline{W}$). У режимі ведучого передавача цей біт повинен бути скинутий (містить значення логічного 0), що відповідає напрямку «запис». Таким чином, перший байт, що передається, містить адресу веденого і біт запису (W). Потім ведучий побайтно передає дані. Після передачі кожного з байтів даних, ведучий приймає від веденого біт

«підтвердження» (АСК). Умови START і STOP генеруються ведучим, щоб вказати початок і кінець послідовної передачі.

Коли вбудований програмований драйвер інтерфейсу I^2C встановлює біт STA, відбувається перехід в режим ведучого передавача. Тепер, якщо шина вільна (тобто інші підключені до шини пристрої не встановлюють на ній напругу НИЗЬКОГО рівня), логічна схема I^2C генерує і передає в лінію умову START. Після цього, мікроконтролер встановлює біт SI регістра I2CONSET і в регістрі стану інтерфейсу I^2C (I2STAT) повертає код стану (результат операції), який дорівнює 08h. Цей код стану призначений для використання вектором і підпрограмою обробки переривання, яка спочатку повинна завантажити адресу веденого і біт запису в регістр даних інтерфейсу I^2C (I2DAT), а потім скинути біт SI. Як уже згадувалося вище, біт SI скидається шляхом запису логічної 1 в біт SIC регістра I2CONCLR.

Біт SI встановлюється знову після того, як адреса веденого і біт R/W передані веденому, і від нього отримано біт «підтвердження» (АСК). У регістрі стану I2STAT знову повертається відповідний код стану. Дії, які мають бути виконані програмою користувача у відповідності зі значенням цього коду, перераховані в таблиці 4.28.

Після переведення модуля I^2C в режим ведучого, ми можемо почати передачу записуваних даних таким чином:

```
void I2CTransferByte(unsigned Addr, unsigned Data)
{
     I2ADB     = Addr           // Адреса та дані повинні
                                     // зберігатися в глобальних
                                     // змінних, щоб їх можна було
                                     // використовувати у перериван-
                                ні
     I2DAT     = Data;
     I2CONCLR  = 0x000000FF;    // Скидаємо   усі   біти   керування
                                // шиною I²C
     I2CONSET  = 0x00000040;    // Дозволяємо інтерфейс I²C
     I2CONSET  = 0x00000020;    // Формуємо стан СТАРТ
}
```

Таблиця 4.28 – Коди стану для режиму I^2C ведучого передавача

Код стану (I2STAT)	**Поточний стан шини I^2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами I2CONSET**				
			STA	STO	SI	AA	
08h	Умова START була передана в лінію	Завантажити адресу веденого + W	X	0	0	X	Адреса веденого + W буде передана в лінію; біт ACK буде прийнятий з лінії
10h	Умова повторний START була передана в лінію	Завантажити адресу веденого + W	X	0	0	X	Як і в попередньому випадку, адреса веденого + W буде передана в лінію
		Завантажити адресу веденого + R	X	0	0	X	I^2C перейде в режим ведучого приймача
18h	Адреса веденого + W була передана в лінію; біт ACK (підтвердження веденого) був прийнятий з лінії	Завантажити байт даних	0	0	0	X	Байт даних буде передано; біт ACK буде прийнято з лінії
		Немає операцій з I2DAT	1	0	0	X	Буде передана умова повторний START
		Немає операцій з I2DAT	0	1	0	X	Буде передана умова STOP; прапорець STO буде скинуто
		Немає операцій з I2DAT	1	1	0	X	Буде передана умова STOP, після якої буде передана умова START; прапорець STO буде скинуто

Продовження таблиці 4.28

Код стану (I2STAT)	**Поточний стан шини I2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами I2CONSET**				
			STA	STO	SI	AA	
20h	Адреса веденого + W була передана; біт NOT ACK було прийнято	Завантажити байт даних	0	0	0	X	Байт даних буде передано; біт ACK буде прийнято з лінії
		Немає операцій з I2DAT	1	0	0	X	Буде передана умова повторний START
		Немає операцій з I2DAT	0	1	0	X	Буде передана умова STOP; прапорець STO буде скинуто
		Немає операцій з I2DAT	1	1	0	X	Буде передана умова STOP, після якої буде передана умова START; прапорець STO буде скинуто
28h	Байт даних з I2CDAT передано; біт ACK було прийнято	Завантажити байт даних	0	0	0	X	Байт даних буде передано; біт ACK буде прийнято з лінії
		Немає операцій з I2DAT	1	0	0	X	Буде передана умова повторний START
		Немає операцій з I2DAT	0	1	0	X	Буде передана умова STOP; прапорець STO буде скинуто
		Немає операцій з I2DAT	1	1	0	X	Буде передана умова STOP, після якої буде передана умова START; прапорець STO буде скинуто

Продовження таблиці 4.28

Код стану (I2STAT)	**Поточний стан шини I^2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами I2CONSET**				
			STA	**STO**	**SI**	**AA**	
30h	Байт даних з I2CDAT передано; біт NOT ACK було прийнято	Завантажити байт даних	0	0	0	X	Байт даних буде передано; біт ACK буде прийнято з лінії
		Немає операцій з I2DAT	1	0	0	X	Буде передана умова повторний START
		Немає операцій з I2DAT	0	1	0	X	Буде передана умова STOP; прапорець STO буде скинуто
		Немає операцій з I2DAT	1	1	0	X	Буде передана умова STOP, після якої буде передана умова START; прапорець STO буде скинуто
38h	Арбітраж втрачений при передачі адреси веденого + R/W або байта даних	Немає операцій з I2DAT	0	0	0	X	Шина I2C буде «відпущена» без звернення до адресованого веденого
		Немає операцій з I2DAT	1	0	0	X	Умова START буде передана в лінію, коли шина

Адреса веденого пристрою і дані, які посилаються, зберігаються в глобальних змінних, щоб їх можна було використовувати в підпрограмі обробки переривання. У бітах 7...1 байта адреси знаходиться власне 7–бітна адреса веденого пристрою, а молодший біт вказує тип транзакції (1 – запис, 0 – читання). Формат обміну по шині I^2C для режиму ведучого передавача проілюстровано на рисунку 4.37. Після того як на шині буде сформовано стан

СТАРТ, генерується переривання і з регістра стану модуля I^2C можна зчитати код результату операції.

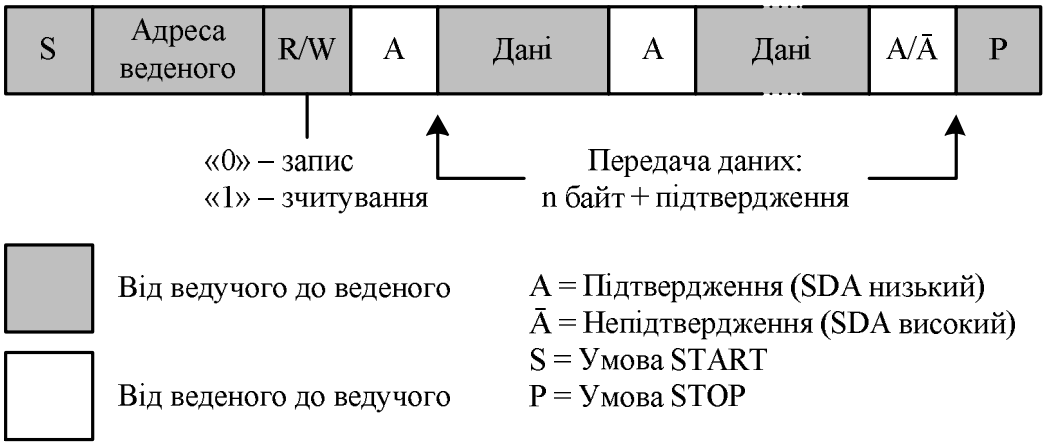


Рисунок 4.37– Формат обміну шиною I^2C для режиму ведучого передавача

При успішному формуванні стану СТАРТ цей код буде дорівнювати 0x08. Зчитавши такий код, прикладна програма повинна завантажити в регістр I2CDATA адресу веденого пристрою з керуючим бітом $R/\overline{W} = 0$. Ці дані будуть видані на шину і підтверджені веденим. При отриманні підтвердження генерується нове переривання, а в регістрі стану буде знаходитися (при успішній передачі) число 0x18. Це означає, що ведений пристрій адресований і готовий до прийому даних. Тепер ми можемо записувати в регістр I2CDATA дані. При записі чергового байта в регістр, він передається по шині I^2C і від веденого пристрою приймається підтвердження. Після прийому сигналу від веденого генерується переривання і в регістрі стану формується число 0x28 (при успішній передачі) або 0x20 (при неуспішній). В останньому випадку байт даних необхідно передати повторно. Таким чином, після передачі кожного байта і генерації переривання можна перевіряти код стану і передавати наступний байт. Після передачі всіх даних можна сформувати на шині стан СТОП, записавши відповідне значення в регістр керування модуля I^2C, і транзакція буде завершена. Неважко побачити, що підпрограма обробки переривання модуля I^2C являє собою кінцевий автомат, який при кожному перериванні перевіряє вміст регістра стану та виконує необхідні дії. Такий автомат легко реалізується за допомогою оператора switch().

```
void I2CISR (void)              // Підпрограма обробки
                                // переривання I2C
{
   switch (I2STAT)              // Зчитуємо код стану і
                                // виконуємо наступну дію
      case (0x08):              // START
         I2CONCLR               = 0x20;   // Скидаємо   прапорець
    стану
                                // СТАРТ
         I2DAT = I2CAddress;    // Посилаємо адресу і біт запису
         break;

      case (0x18):              // SLA+W, ACK
         I2DAT                  = I2Cdata;     // Пишемо  дані  в
    регістр
                                // передачі
         break;

      case (0x20):              // SLA+W, NACK
         I2DAT                  = I2CAddress;  // Повторно  поси-
    лаємо адресу і
                                // біт запису
         break;

      case (0x28):              // Дані, ACK
         I2CONSET               = 0x10;   // Формуємо СТОП
         break;

      default:
         break;
}
   I2CONCLR   = 0x08;           // Скидаємо прапорець
                                // переривання I2C
   VICVectAddr                  = 0x00000000;  // Скидаємо  пере-
ривання в VIC
}
```

У наведеному прикладі передається тільки один байт, однак цей приклад легко доопрацювати для передачі декількох байтів. Також можна додати додаткові оператори вибору для підтримки запитів даних від інших ведучих.

4.7.3.2 Режим ведучого приймача

У режимі ведучого приймача здійснюється прийом даних від веденого передавача. Передача ініціалізується тим же самим способом, що і в режимі ведучого передавача. Після генерації ведучим умови START, підпрограма обробки переривання повинна завантажити адресу веденого і біт напряму даних в регістр даних I^2C (I2DAT), а потім скинути прапорець SI в регістрі I2CONSET.

Біт напрямку передачі даних (R/W) повинен бути встановлений в логічну 1, що відповідає зчитуванню даних. Таким чином, перший переданий байт повинен містити адресу веденого і біт зчитування (R).

Після передачі ведучим адреси веденого і біта напряму даних і отримання від веденого біта «підтвердження» (АСК), мікроконтролер встановлює біт SI і повертає відповідний код стану в регістрі стану I^2C (I2STAT). Дії, які повинні бути зроблені для користувача програмним забезпеченням у відповідності зі значенням цього коду стану, перечислені в таблиці 4.29.

Формат обміну шиною I^2C для режиму ведучого приймача зображено на рисунку 4.38. На рисунку 4.39 показаний формат обміну шиною I^2C при перемиканні інтерфейсу I^2C мікроконтролера з режиму ведучого приймача в режим ведучого передавача після генерації умови повторний START.

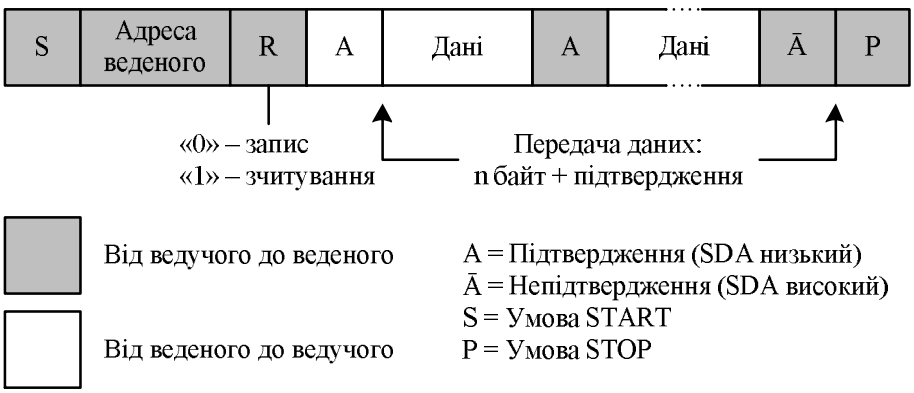


Рисунок 4.38 – Формат обміну по шині I^2C для режиму ведучого приймача

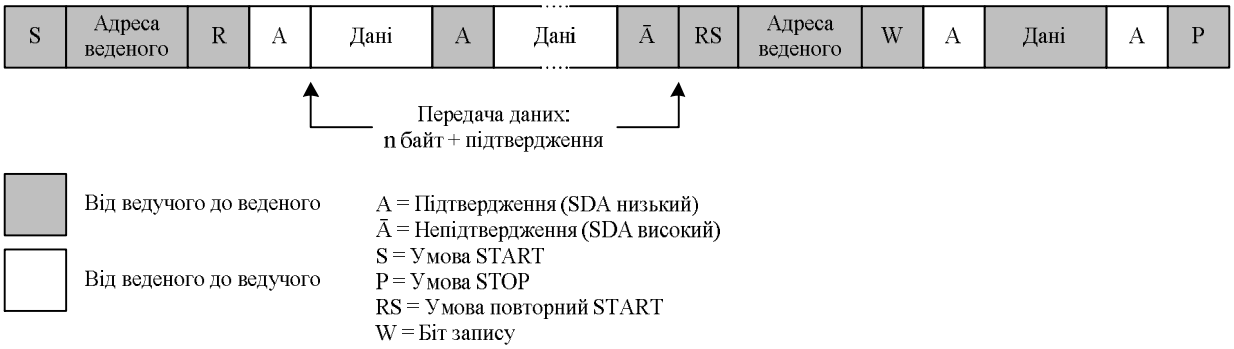


Рисунок 4.39 – Формат обміну шиною I^2C для режиму ведучого приймача, а потім для режиму ведучого передавача

Таблиця 4.29 – Коди стану для режиму I^2C ведучого приймача

Код стану (I2STAT)	**Поточний стан шини I^2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами I2CONSET**				
			STA	**STO**	**SI**	**AA**	
08h	Передана умова START	Завантажити адресу веденого +R	X	0	0	X	Адреса веденого + R буде передана в лінію; біт АСК буде прийнято з лінії
10h	Передана умова повторний START	Завантажити адресу веденого +R	X	0	0	X	Як і в попередньому випадку, адреса веденого +R буде передана в лінію
		Завантажити адресу веденого +W	X	0	0	X	I2C перейде в режим ведучого передавача
38h	Арбітраж втрачено під час прийому біта NOT АСК	Немає операцій з I2DAT	0	0	0	X	Шина I2C буде «відпущена»; I2C перейде в режим веденого
		Немає операцій з I2DAT	1	0	0	X	Умова START буде передана в лінію, коли шина звільниться
40h	Адреса веденого + R була передана в лінію; біт АСК було прийнято	Немає операцій з I2DAT	0	0	0	0	Буде принято байт даних з лінії і повернено (передано в лінію) біт NOT АСК
		Немає операцій з I2DAT	0	0	0	1	Буде принято байт даних з лінії і повернено (передано в в лінію) біт АСК

Продовження таблиці 4.29

Код стану (I2STAT)	**Поточний стан шини I^2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами**				
			STA	**STO**	**SI**	**AA**	
48h	Адреса веденого + R була передана в лінію; біт NOT ACK було прийнято	Немає операцій з I2DAT	1	0	0	X	Умова повторний START буде передана в лінію
		Немає операцій з I2DAT	0	1	0	X	Умова STOP буде передана в лінію; прапорець STO буде скинуто
		Немає операцій з I2DAT	1	1	0	X	Умова STOP буде передана в лінію, після якої буде передана умова START; прапорець STO буде скинуто
50h	Байт даних було прийнято; біт ACK було передано в лінію	Прочитати байт даних	0	0	0	0	Буде прийнято байт даних з лінії і повернено (передано в лінію) біт
		Прочитати байт даних	0	0	0	1	Буде принято байт даних з лінії і повернено (передано в лінію) біт ACK
58h	Байт даних було прийнято; біт NOT ACK було передано в лінію	Прочитати байт даних	1	0	0	X	Умова повторний START буде передана в
		Прочитати байт даних	0	1	0	X	Умова STOP буде передана в лінію; прапорець STO буде скинуто
		Прочитати байт даних	1	1	0	X	Умова STOP буде передана в лінію, після якої буде передана умова START; прапорець STO буде скинуто

Для запуску транзакції читання можна використовувати ту ж функцію I2CISR**()**, додавши в обробник переривання нові варіанти оператора switch():

```
case (0x40):        // SLA+R, ACK
I2CONSET   = 0x04;  // Дозволяємо підтвердження для байтів
                    // даних
break;

case (0x48.):       // SLA+R, NACK
I2CONSET   = 0x20;  // Знову формуємо стан СТАРТ
break;

case (0x50):        // Дані прийняті, ACK
message    = I2DAT;
I2CONSET   = 0x10;  // Формуємо стан СТОП
Lock       = 0;     // Індикуємо закінчення активності на шині
break;

case (0x58);        // Дані прийняті, NACK
I2CONSET   = 0x20;  // Знову формуємо стан СТАРТ
break;
```

4.7.3.3 Режим веденого приймача

У режимі веденого (Slave) приймача мікроконтролер одержує байти даних від ведучого (Master) передавача. Перехід в режим веденого приймача здійснюється наступним чином: для користувача програма ініціалізує регістр адреси веденого (I2ADR), записуючи в нього задане значення адреси, а також ініціалізує регістр керування I^2C (I2CONSET), записуючи в нього значення, які вказані в таблиці 4.30.

Біт I2EN (прапорець дозволу) регістра I2CONSET повинен бути встановлений, щоб дозволити роботу інтерфейсу I^2C в мікроконтролері. Встановлення в логічну 1 біта AA (прапорець підтвердження) вказує інтерфейсу I^2C мікроконтролера, що він повинен після отримання з лінії власної адреси веденого або адреси загального виклику видавати в лінію біт «підтвердження» (ACK). Біти (прапорці) STA, STO і SI повинні бути скинуті.

Після ініціалізації регістрів I2ADR і I2CONSET, інтерфейс I2C веденого очікує звернення від зовнішнього (ведучого) пристрою за його (веденого) індивідуальною адресою веденого або мережевою адресою загального виклику.

Таблиця 4.30 – Ініціалізація регістра I2CONSET для режиму веденого

Номер біта	Назва біта	Значення біта
0	–	–
1	–	–
2	AA	1
3	SI	0
4	STO	0
5	STA	0
6	I2EN	1
7	–	–

Характер цього звернення (запис або читання) ідентифікується веденим за значенням біта напрямку даних в посилці, що надійшла від ведучого. Якщо біт напрямку скинутий (НИЗЬКИЙ рівень відповідає режиму запису в лінію), значить, ведучий пристрій має намір передавати дані, а інтерфейс I^2C мікроконтролера переходить в режим веденого приймача.

Після отримання адреси веденого і біта напрямку даних, мікроконтролер у режимі веденого приймача встановлює біт SI і повертає відповідний код стану в регістр стану (I2STAT). Дії, які повинні бути зроблені для користувача програмним забезпеченням у відповідності зі значенням цього коду стану, перераховані в таблиці 4.31.

Формат обміну шиною I^2C для режиму веденого приймача показаний на рисунку 4.40.

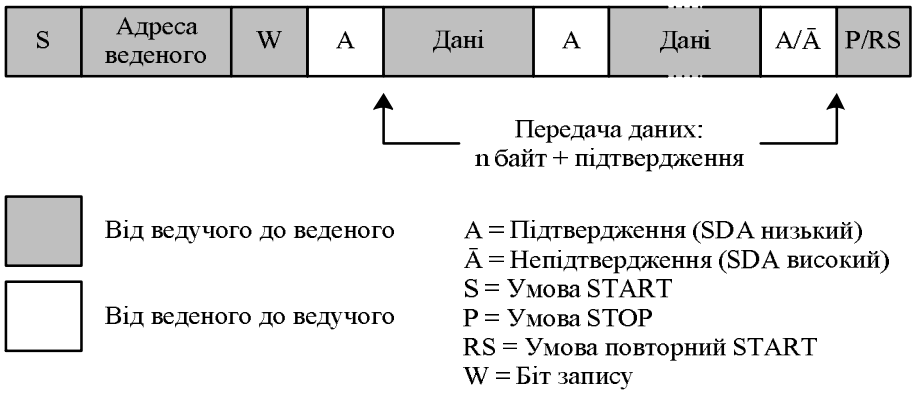


Рисунок 4.40 – Формат обміну шиною I^2C для режиму веденого приймача

Таблиця 4.31 – Коди стану для режиму I^2C веденого приймача

Код стану (I2STAT)	**Поточний стан шини I^2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами**				
			STA	**STO**	**SI**	**AA**	
60h	Індивідуальна адреса веденого +W була отримана з лінії; біт АСК було повернено (передано в лінію)	Немає операцій з I2DAT	X	0	0	0	Буде прийнято байт даних з лінії і повернено (передано в лінію) біт NOT АСК
		Немає операцій з I2DAT	X	0	0	1	Буде прийнято байт даних з лінії і повернено (передано в лінію) біт АСК
68h	Арбітраж було втрачено під час отримання від ведучого адреси веденого +R/W; індивідуальна адреса веденого +W була отримана; біт АСК було повернуто	Немає операцій з I2DAT	X	0	0	0	Буде прийнято байт даних з лінії і повернено (передано в лінію) біт NOT АСК
		Немає операцій з I2DAT	X	0	0	1	Буде прийнято байт даних з лінії і повернено (передано в лінію) біт АСК
70h	Адреса загального виклику (00h) була отримана; біт АСК було повернуто	Немає операцій з I2DAT	X	0	0	0	Буде прийнято байт даних з лінії і повернено (передано в лінію) біт NOT АСК
		Немає операцій з I2DAT	X	0	0	1	Буде прийнято байт даних з лінії і повернено (передано в лінію) біт АСК

Продовження таблиці 4.31

Код стану (I2STAT)	**Поточний стан шини I2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами**				
			STA	**STO**	**SI**	**AA**	
78h	Арбітраж було втрачено під час отримання від ведучого адреси веденого +R/W; адреса загального виклику була отримана; біт АСК було повернуто	Немає операцій з I2DAT	X	0	0	0	Буде прийнято байт даних з лінії і повернено (передано в лінію) біт NOT АСК
		Немає операцій з I2DAT	X	0	0	1	Буде прийнято байт даних з лінії і повернено (передано в лінію) біт АСК
80h	Попередньо адресована індивідуальна адреса веденого була отримана; байт даних було отримано; біт АСК було повернуто	Прочитати байт даних	X	0	0	0	Буде прийнято байт даних з лінії і повернено (передано в лінію) біт NOT АСК
		Прочитати байт даних	X	0	0	1	Буде прийнято байт даних з лінії і повернено (передано в лінію) біт АСК

Продовження таблиці 4.31

Код стану (I2STAT)	**Поточний стан шини I2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами**				
			STA	**STO**	**SI**	**AA**	
88h	Попередньо адресована індивідуальна адреса веденого була отримана; байт даних було отримано; біт NOT ACK було повернуто	Прочитати байт даних	0	0	0	0	Перемикання в режим, в якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого або адреса загального виклику не будуть
		Прочитати байт даних	0	0	0	1	Перемикання в режим, в якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого буде розпізнаватись; адреса загального виклику буде розпізнаватись, якщо
		Прочитати байт даних	1	0	0	0	Перемикання в режим, в якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого або адреса загального виклику не будуть розпізнаватись; умова START буде передана в лінію, коли шина звільниться

Продовження таблиці 4.31

Код стану (I2STAT)	**Поточний стан шини I2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами I2CONSET**				
			STA	**STO**	**SI**	**AA**	
		Прочитати байт даних	1	0	0	1	Перемикання в режим, в якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого буде розпізнаватись; адреса загального виклику буде розпізнаватись, якщо I2ADR[0] = 1; умова START буде передана в лінію, коли шина звільниться
90h	Попередньо адресована адреса загального виклику буде отримана; байт даних було отримано; біт АСК було повернуто	Прочитати байт даних	X	0	0	0	Байт даних буде отримано і буде повернуто біт NOT АСК
		Прочитати байт даних	X	0	0	1	Байт даних буде отримано і буде повернуто біт АСК

Продовження таблиці 4.31

Код стану (I2STAT)	**Поточний стан шини I2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами I2CONSET**				
			STA	**STO**	**SI**	**AA**	
98h	Попередньо адресована адреса загального виклику буде отримана; байт даних було отримано; біт NOT ACK було повернуто	Прочитати байт даних	0	0	0	0	Перемикання в режим, у якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого або адреса загального виклику не будуть розпізнаватись
		Прочитати байт даних	0	0	0	1	Перемикання в режим, у якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого буде розпізнаватись; адреса загального виклику буде розпізнаватись, якщо I2ADR[0] = 1
		Прочитати байт даних	1	0	0	0	Перемикання в режим, у якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого або адреса загального виклику не будуть розпізнаватись; умова START буде передана в лінію, коли шина звільниться

Продовження таблиці 4.31

Код стану (I2STAT)	**Поточний стан шини I2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами I2CONSET**				
			STA	**STO**	**SI**	**AA**	
		Прочитати байт даних	1	0	0	1	Перемикання в режим, у якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого буде розпізнаватись; адреса загального виклику буде розпізнаватись, якщо I2ADR[0] = 1; умова START буде передана в лінію, коли шина
A0h	Під час роботи I2C в режимі веденого приймача або веденого передавача, була отримана умова STOP або умова повторний START	Немає операцій з I2DAT	0	0	0	0	Перемикання в режим, у якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого або адреса загального виклику не будуть розпізнаватись

Продовження таблиці 4.31

Код стану (I2STAT)	**Поточний стан шини I2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами I2CONSET**				
			STA	**STO**	**SI**	**AA**	
A0h	Під час роботи I2C в режимі веденого приймача або веденого передавача, була отримана умова STOP або умова повторний START	Немає операцій з I2DAT	0	0	0	1	Перемикання в режим, у якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого буде розпізнаватись; адреса загального виклику буде розпізнаватись, якщо I2ADR[0] = 1
		Немає операцій з I2DAT	1	0	0	0	Перемикання в режим, у якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого або адреса загального виклику не будуть розпізнаватись; умова START буде передана в лінію, коли шина

Продовження таблиці 4.31

Код стану (I2STAT)	Поточний стан шини I2C і апаратної логіки інтерфейсу	Відповідь користувальницького програмного забезпечення					Наступна дія апаратної логіки інтерфейсу
		Операції з I2DAT	Операції з бітами I2CONSET				
			STA	STO	SI	AA	
		Немає операцій з I2DAT	1	0	0	1	Перемикання в режим, у якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого буде розпізнаватись; адреса загального виклику буде розпізнаватись, якщо I2ADR[0] = 1; умова START буде передана в

4.7.3.4 Режим веденого передавача

У режимі веденого передавача перший байт приймається від ведучого і обробляється аналогічно режиму веденого приймача, за винятком того, що встановлено біт напряму даних (ВИСОКИЙ рівень відповідає читанню даних з лінії) сигналізує про намір ведучого приймати дані від веденого. По лінії SDA в послідовному вигляді ведений передає ведучому дані, в той час як по лінії SCL від ведучого до веденого надходить послідовність синхроімпульсів. Умови START і STOP розпізнаються веденим, як початок і кінець послідовної передачі.

Як і в режимі веденого приймача, в режимі веденого передавача апаратні засоби інтерфейсу I^2C виконують порівняння прийнятої від ведучого адреси веденого зі своєю індивідуальною адресою веденого і адресою загального виклику. Якщо хоча б одна із цих адрес збігається з отриманим від ведучого, генерується переривання мікроконтролера за цією подією.

У таблиці 4.32 перераховані коди станів, що повертаються мікроконтролером в регістрі стану (I2STAT) для режиму веденого передавача, і

дії, які повинні бути зроблені програмою користувача у відповідності зі значенням цього коду.

Таблиця 4.32 – Коди стану для режиму I^2C веденого передавача

Код стану (I2STAT)	**Поточний стан шини I2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами**				
			STA	**STO**	**SI**	**AA**	
A8h	Індивідуальна мережева адреса веденого +R була отримана з лінії; біт АСК було повернуто (передано в лінію)	Завантажити байт даних	X	0	0	0	Останній байт даних буде передано в лінію; біт АСК буде прийнято з лінії
		Завантажити байт даних	X	0	0	1	Останній байт даних буде передано і біт АСК буде прийнято з лінії
B0h	Арбітраж було втрачено під час проходження адреси веденого +R/W от ведучого; індивідуальна адреса веденого +R була отримана; біт АСК було повернуто	Завантажити байт даних	X	0	0	0	Останній байт даних буде передано в лінію; біт АСК буде прийнято з лінії
		Завантажити байт даних	X	0	0	1	Останній байт даних буде передано і біт АСК буде прийнято з лінії

Продовження таблиці 4.32

Код стану (I2STAT)	**Поточний стан шини I2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами**				
			STA	**STO**	**SI**	**AA**	
B8h	Байт даних з регістра I2DAT було передано в лінію; біт АСК було прийнято з лінії	Завантажити байт даних	X	0	0	0	Останній байт даних буде передано в лінію; біт АСК буде прийнято з лінії
		Завантажити байт даних					Останній байт даних буде передано і біт АСК буде прийнято з лінії
C0h	Байт даних з регістра I2DAT було передано в лінію; біт NOT АСК було прийнято	Немає операцій з I2DAT	0	0	0	0	Перемикання в режим, у якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого або адреса загального виклику не будуть розпізнаватись
		Немає операцій з I2DAT	0	0	0	1	Перемикання в режим, у якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого буде розпізнаватись; адреса загального виклику буде розпізнаватись,якщо I2ADR[0] = 1

Продовження таблиці 4.32

Код стану (I2STAT)	**Поточний стан шини I2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами**				
			STA	**STO**	**SI**	**AA**	
		Немає операцій з I2DAT	1	0	0	0	Перемикання в режим, у якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого або адреса загального виклику не будуть розпізнаватись; умова START буде передана в лінію, коли шина звільниться
		Немає операцій з I2DAT	1	0	0	1	Перемикання в режим, у якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого буде розпізнаватись; адреса загального виклику буде розпізнаватись, якщо I2ADR[0] = 1; умова START буде передана в лінію, коли шина звільниться

Продовження таблиці 4.32

Код стану (I2STAT)	**Поточний стан шини I2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами**				
			STA	**STO**	**SI**	**AA**	
C8h	Останній байт з регістра I2DAT було передано в лінію (AA = 0); біт АСК було прийнято з лінії	Немає операцій з I2DAT	0	0	0	0	Перемикання в режим, у якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого або адреса загального виклику не будуть розпізнаватись
		Немає операцій з I2DAT	0	0	0	1	Перемикання в режим, у якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого буде розпізнаватись; адреса загального виклику буде розпізнаватись, якщо I2ADR[0] = 1
		Немає операцій з I2DAT	1	0	0	0	Перемикання в режим, у якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого або адреса загального виклику не будуть розпізнаватись; умова START буде передана в лінію, коли шина звільниться

Продовження таблиці 4.32

Код стану (I2STAT)	**Поточний стан шини I2C і апаратної логіки**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами**				
			STA	**STO**	**SI**	**AA**	
		Немає операцій з I2DAT	1	0	0	1	Перемикання в режим, у якому попередня адресація веденого відсутня; індивідуальна мережева адреса веденого буде розпізнаватись; адреса загального виклику буде розпізнаватись, якщо I2ADR[0] = 1; умова START

Якщо від керуючої програми мікроконтролера надходить команда перевести мікроконтролер з режиму веденого в режим ведучого, апаратні засоби інтерфейсу I^2C чекають звільнення шини, щоб не перервати транзакції, які можливо здійснюються веденим. Якщо в режимі ведучого втрачається шинний арбітраж, інтерфейс I^2C мікроконтролера негайно перемикається в режим веденого і може виявити свою індивідуальну адресу веденого в тій же самій послідовній передачі.

У таблиці 4.33 наведені значення кодів стану інтерфейсу I^2C, які повертаються в регістрі стану (I2STAT) у разі помилок шини або інших некоректних ситуацій.

Таблиця 4.33 – Коди інших станів інтерфейсу I^2C

Код стану (I2STAT)	**Поточний стан шини I2C і апаратної логіки інтерфейсу**	**Відповідь користувальницького програмного забезпечення**					**Наступна дія апаратної логіки інтерфейсу**
		Операції з I2DAT	**Операції з бітами**				
			STA	**STO**	**SI**	**AA**	
F8h	Невизначений стан, при якому ніяка інформація недоступна; SI = 0	Немає операцій з I2DAT	Немає операцій з I2CONSET				Очікування чи продовження поточної передачі
00h	Помилка на шині у режимі ведучого або вибір режима веденого через некоректні умови START або STOP. Стан 00h може також настати у разі взаємного накладення спроб інтерфейсу I2C перейти в невизначений стан	Немає операцій з I2DAT	0	1	0	X	У режимах ведучого або адресованого веденого задіяні тільки внутрішні апаратні засоби I2C. У будь–якому випадку, шина I2C буде «відпущена» і інтерфейс I2C переключиться в режим веденого без попередньої адресації; біт STO буде скинуто

Формат обміну шиною I^2C для режиму веденого передавача показано на рисунку 4.41.

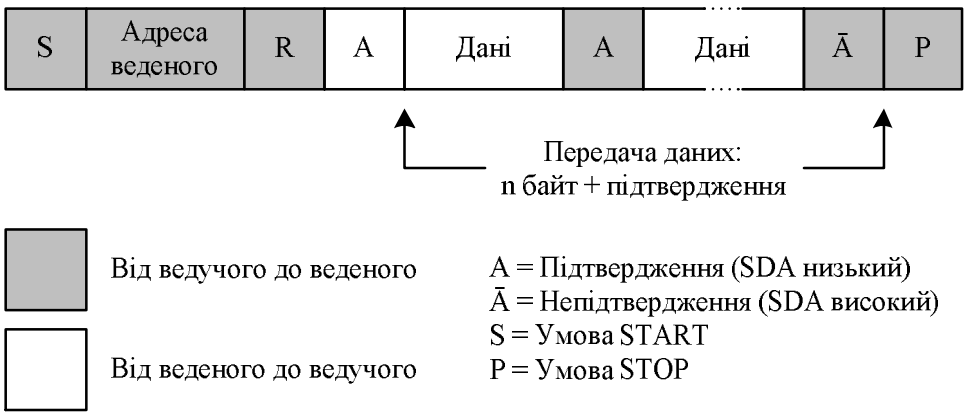


Рисунок 4.41– Формат обміну шиною I^2C для режиму веденого передавача

4.7.3.5 Взаємодія прикладної програми з модулем I^2C

Взаємодія прикладної програми з модулем I^2C базується на використанні переривання від модуля, яке виникає після кожної події, що відбулася на шині (прийом байта, формування станів СТАРТ/СТОП і т. ін.). Відповідно, під час передачі даних шиною програма може виконувати інші задачі. Якщо переривання від модуля I^2C з якихось причин використовувати не можна, його можна заборонити. У цьому випадку програма повинна буде постійно слідкувати за станом прапорця SI для реагування на події, які відбуваються на шині.

Встановлення прапорця SI регістра I2CONSET означає, що модуль I^2C закінчив виконання чергової операції і очікує реакції програми.

У старших п'яти розрядах регістра I2STAT при цьому формується відповідне значення, яке характеризує поточний стан шини I^2C. Відповідно, програма повинна проаналізувати це значення і задати подальше поводження модуля I^2C, маніпулюючи вмістом регістрів I2CONSET і I2DAT.

На рисунку 4.42 показано взаємодію прикладної програми з модулем I^2C на найпростішому прикладі передачі одного байта даних від ведучого до веденого.

Рисунок 4.42 – Приклад взаємодії програми з модулем I^2C

Передача даних відбувається у наступній послідовності:

– першою операцією при передачі даних шиною I^2C є формування стану СТАРТ. Для цього варто записати певне значення в регістр I2CONSET, відповідно до якого модуль I^2C сформує на шині стан СТАРТ. При цьому для скидання прапорця SI розряд SIC регістра I2CONCLR повинен бути встановлений в «1». Формування стану СТАРТ почнеться відразу ж після скидання прапорця SI;
– після формування стану СТАРТ встановлюється прапорець SI. Число, яке перебуває у регістрі стану I2STAT, відображає результат виконання цієї операції;
– необхідно впевнитися в успішному формуванні стану СТАРТ, перевіривши вміст регістра I2STAT. Якщо код стану відповідає очікуваному, варто завантажити в регістр I2DAT вміст пакета SLA+$\overline{W}$ і сформувати в регістрі I2CONSET команду на передачу пакета (не забуваючи скинути при цьому прапорець SI). Передача адресного пакета почнеться відразу ж після скидання прапорця;
– по закінченні передачі адресного пакета встановлюється прапорець SI. Код стану, що перебуває в регістрі I2STAT, говорить про успішну передачу пакета, а також про одержання підтвердження від веденого пристрою;
– необхідно впевнитися в успішній передачі пакета та одержанні підтвердження, перевіривши вміст регістра I2STAT. Якщо код стану відповідає очікуваному, слід завантажити дані в регістр I2DAT, а потім сформувати в регістрі I2CONSET команду на передачу пакета (не забуваючи скинути при цьому прапорець SI). Передача пакета даних почнеться відразу ж після скидання прапорця;
– по закінченні передачі пакета даних встановлюється прапорець SI. Код стану, що перебуває в регістрі I2STAT, говорить про успішну передачу пакета, а також про одержання підтвердження від веденого пристрою;
– необхідно впевнитися в успішній передачі пакета та одержанні підтвердження, перевіривши вміст регістра I2STAT. Якщо код стану відповідає очікуваному, слід записати в регістр I2CONSET значення (не

забуваючи скинути при цьому прапорець SI), відповідно до якого модуль I^2C сформує на шині стан СТОП або ПОВСТАРТ. Формування стану СТОП або ПОВСТАРТ почнеться відразу ж після скидання прапорця SI.

Зі сказаного видно, що будь–який етап взаємодії прикладної програми з модулем I^2C складається із наступних трьох частин:

1) Після завершення модулем виконання чергової операції, він встановлює прапорець SI регістра I2CONSET, записує у регістр I2STAT код стану і очікує реакції програми. Доки прапорець встановлено в «1», на лінії SCL утримується НИЗЬКИЙ рівень.

2) Після встановлення прапорця SI користувач повинен занести в регістри модуля значення, які відповідають наступному етапу обміну.

3) Після оновлення вмісту регістрів модуля, користувач повинен сформувати в регістрі I2CONSET команду для виконання наступного етапу обміну. При завантаженні в регістр нового значення необхідно скинути прапорець SI записом логічної 1 в біт SIC регістра I2CONCLR. Після скидання прапорця модуль почне виконання операції, обумовленої вмістом регістра I2CONSET.

4.7.4 Опис виводів

Опис виводів інтерфейсу I^2C мікроконтролера LPC2300 приведено в таблиці 4.34.

Таблиця 4.34 – Опис виводів інтерфейсу I^2C

Назва виводу	Тип	Опис
SDA0, SDA1, SDA2	I/O	Послідовні дані. Вхід і вихід даних I^2C. Сполучена з цим ланцюгом лінія мікроконтролера, у відповідності зі специфікаціями I^2C, має вихід з відкритим колектором

Продовження таблиці 4.34

SCL0, SCL1, SCL2	I/O	Послідовний синхросигнал. Вхід і вихід синхросигналу I^2C. Сполучена з цим ланцюгом лінія мікроконтролера, у відповідності зі специфікаціями I2C, має вихід з відкритим колектором

4.7.5 Опис керуючих регістрів

Інтерфейс I^2C мікроконтролерів сімейства LPC2300 включає в себе сім керуючих регістрів, які перераховано в таблиці 4.35.

Регістру керування відповідають два окремих регістри, які призначено для встановлення і скидання його керуючих бітів (I2CONSET і I2CONCLR). Швидкість передачі даних також визначається вмістом двух регістрів (I2CLH і I2CLL). Регістр стану I2STAT повертає коди стану, що відповідають різним подіям на шині. Регістр даних I2DAT використовується для запису кожного байта, який передається, або для зчитування кожного байта, який приймається. Коли мікроконтроллер працює в режимі веденого, його мережева адреса визначається вмістом регістра адреси I2ADDR.

Примітка. У мікроконтролерах LPC2300, що мають три інтерфейси I^2C (I^2C0, I^2C1, I^2C2), спеціальні регістри, що відносяться до інтерфейсу I^2C0, мають адреси, що збігаються з адресами спеціальних регістрів інтерфейсу I^2C в мікроконтролерах LPC2300 з одним інтерфейсом I^2C.

4.7.5.1 Регістр керування встановленням

Регістр керування встановленням має адреси: для I^2C0: I2CONSET або I2C0CONSET – 0xE001 C000, для I2C1: I2C1CONSET – 0xE005 C000, для I2C2: I2C2CONSET – 0xE008 0000.

Бітова структура регістра I2CONSET наведена в таблиці 4.36.

Таблиця 4.35 – Карта регістрів інтерфейсу I^2C

Назва	Опис	Доступ	Значення після скидання[1)]	Адреса для I^2C0	Адреса для I^2C1	Адреса для I^2C2
I2CONSET	Регістр керування встановленням	Читання/запис	0	0xE001C000	0xE005C000	0xE0080000
I2STAT	Регістр стану	Тільки читання	0xF8	0xE001C004	0xE005C004	0xE0080004
I2DAT	Регистр даних	Читання/запис	0	0xE001C008	0xE005C008	0xE0080008
I2ADR	Регістр адреси веденого	Читання/запис	0	0xE001C00C	0xE005C00C	0xE008000C
I2SCLH	Регістр старшого напівслова швидкості передачі даних SCL	Читання/запис	0x04	0xE001C010	0xE005C010	0xE0080010
I2SCLL	Регістр молодшого напівслова швидкості передачі даних SCL	Читання/запис	0x04	0xE001C014	0xE005C014	0xE0080014
I2CONCLR	Регістр керування скиданням	Читання/запис	NA	0xE001C018	0xE005C018	0xE0080018

[1)] Значення відноситься до даних, що містяться тільки в використовуваних бітах, і не включає вміст зарезервованих бітів.

Таблиця 4.36 – Біти регістра керування I2CONSET

Біти I2CONSET	Назва	Опис	Значення після скидання
0	Зарезервовано	Користувальницьке програмне забезпечення не повинно проводити запис в зарезервовані біти. Читання зарезервованого біта повертає невизначене значення: NA (Not Applicable)	NA
1	Зарезервовано	Користувальницьке програмне забезпечення не повинно проводити запис в зарезервовані біти. Читання зарезервованого біта повертає невизначене значення: NA	NA
2	AA	Прапорець видачі біта «підтвердження»	0
3	SI	Прапорець переривання від I^2C	0
4	STO	Прапорець STOP	0
5	STA	Прапорець START	0
6	I2EN	Біт дозволу I^2C	0
7	Зарезервовано	Користувальницьке програмне забезпечення не повинно проводити запис в зарезервовані біти. Читання зарезервованого біта повертає невизначене значення: NA	NA

Нижче наведено пояснення окремих бітів регістра керування встановленням.

Біт AA – прапорець видачі біта «підтвердження». Якщо прапорець встановлений, мікроконтролер повертає в лінію SDA біт «підтвердження» (НИЗЬКИЙ рівень на лінії SDA), синхронізований відповідним «підтвердженню» тактовим імпульсом на лінії SCL. Це відбувається у наступних ситуаціях:

1. По шині SDA була прийнята адреса веденого, що співпадає з адресою, яка зберігається в регістрі адреси веденого I^2C.

2. По шині SDA була прийнята адреса загального виклику, в той час як біт

загального виклику (GC) в регістрі I2ADR встановлений в 1.

3. По шині SDA був прийнятий байт даних, а інтерфейс I^2C знаходиться в режимі ведучого приймача.

4. По шині SDA був прийнятий байт даних, а інтерфейс I^2C знаходиться в режимі веденого приймача, до якого адресується звернення.

Біт AA може бути скинутий шляхом запису логічної одиниці в біт AAC регістра I2CONCLR. Коли біт AA скинутий, мікроконтролер повертає в лінію SDA біт «непідтвердження» (ВИСОКИЙ рівень на лінії SDA), синхронізований відповідним «підтвердженню» тактовим імпульсом на лінії SCL. Це відбувається у наступних ситуаціях:

1. По шині SDA був прийнятий байт даних, а інтерфейс I^2C знаходиться в режимі ведучого приймача.

2. По шині SDA був прийнятий байт даних, а інтерфейс I^2C знаходиться в режимі адресації веденого приймача.

Біт SI – прапорець переривання від I^2C. Цей біт встановлюється, коли настає одне з 25 можливих активних станів інтерфейсу I^2C. Якщо SI встановлено, низький рівень тактового сигналу на лінії SCL розтягується, а послідовний обмін буде призупинено. Коли рівень SCL високий, він не залежить від стану прапорця SI.

Біт SI може бути скинутий шляхом запису логічної 1 в біт SIC регістра I2CONCLR.

Біт STO – прапорець умови STOP. Встановлення цього біта змушує інтерфейс I^2C генерувати на шині умову STOP в режимі ведучого або вийти зі стану помилки в режимі веденого. Коли біт STO встановлюється в режимі ведучого, на шині I^2C генерується умова STOP. Коли апаратні засоби шини виявляють умову STOP, біт STO автоматично скидається.

У режимі веденого, встановлення цього біта виводить інтерфейс I^2C зі стану помилки. У цьому випадку, умова STOP на шині не генерується, а апаратні засоби інтерфейсу I^2C поводяться так, ніби умова STOP була отримана. Інтерфейс I^2C перемикається в режим веденого приймача, до якого не адресувалося звернення.

Потім прапорець STO автоматично скидається апаратними засобами шини.

Біт STA – прапорець умови START. Встановлення цього біта змушує інтерфейс I^2C перейти в режим ведучого і згенерувати на шині I^2C умову START або повторну умову START, якщо інтерфейс I^2C вже знаходиться в режимі ведучого.

У разі, коли біт STA встановлюється в режимі веденого I^2C, здійснюється перехід в режим ведучого, виконується перевірка шини і, якщо шина вільна, на ній генерується умова START. Якщо шина не вільна, то реалізуюча інтерфейс I^2C логічна схема очікує умови STOP (яка звільнить шину), а потім після закінчення затримки, що дорівнює половині періоду внутрішнього тактового генератора, генерує умову START. Якщо інтерфейс I^2C вже знаходиться в режимі ведучого, і прийняв або передав дані, то встановлення біта STA викличе генерацію умови «повторний START».

Біт STA може бути встановлений в будь–який час, включаючи моменти, коли інтерфейс I^2C знаходиться в режимі веденого, до якого адресується звернення.

Біт STA може бути скинутий шляхом запису логічної 1 в біт STAC регістра I2CONCLR. Коли біт STA скинутий, умови START або повторний START не можуть бути згенеровані.

У ситуації, коли встановлені обидва біти STA і STO, на шині I^2C буде згенеровано стан STOP, якщо інтерфейс знаходиться в режимі ведучого, а після цього буде згенеровано умову START. Якщо інтерфейс I^2C знаходиться в режимі веденого, то буде згенерована внутрішня умова, відповідна умові STOP, однак умова STOP на шину I^2C передана не буде.

I2EN – біт дозволу інтерфейсу I^2C. Коли біт I2EN встановлений, робота інтерфейсу I^2C мікроконтролера дозволена. Біт I2EN може бути скинутий шляхом запису логічної 1 в біт I2ENC регістра I2CONCLR. Коли біт I2EN скинутий, функція інтерфейсу I^2C мікроконтролера заблокована.

4.7.5.2 Регістр керування скиданням

Регістр керування скиданням має адреси: для I^2C0: I2CONCLR або I^2C0 CONCLR – 0xE001 C018, для I^2C1: I2C1CONCLR – 0xE005 C018, для I2C2:

I2C2CONCLR – 0xE008 0018.

Бітова структура регістра наведена в таблиці 4.37.

Таблиця 4.37 – Біти регістра керування скиданням

Біти I2CONCLR	Назва	Опис	Значення після скидання
0	Зарезервовано	Користувальницьке програмне забезпечення не має проводити запис в зарезервовані біти. Читання зарезервованого біта повертає невизначене значення: NA	NA
1	Зарезервовано	Користувальницьке програмне забезпечення не має проводити запис в зарезервовані біти. Читання зарезервованого біта повертає невизначене значення: NA	NA
2	AAC	Біт скидання прапорця видачі біта «підтвердження». Запис логічної 1 в цей біт скидає прапорець AA (очищає біт) в регістрі I2CONSET. Запис логічного 0 не призводить до яких–небудь	NA
3	SIC	Біт скидання прапорця переривання від I^2C. Запис логічної 1 в цей біт скидає прапорець SI в регістрі I2CONSET. Запис логічного 0 не призводить до яких–небудь наслідків	NA
4	Зарезервовано	Користувальницьке програмне забезпечення не має проводити запис в зарезервовані біти. Читання зарезервованого біта повертає невизначене значення	NA
5	STAC	Біт скидання прапорця START. Запис логічної 1 в цей біт скидає прапорець STA в регістрі I2CONSET. Запис логічного 0 не призводить до яких–небудь наслідків	NA

Продовження таблиці 4.37

Біти I2CONCLR	Назва	Опис	Значення після скидання
6	I2ENC	Біт відключення інтерфейсу I^2C. Запис логічної 1 в цей біт скидає прапорець I2EN в регістрі I2CONSET. Запис логічного 0 не призводить до яких–небудь наслідків	NA
7	Зарезервовано	Користувальницьке програмне забезпечення не має проводити запис в зарезервовані біти. Читання зарезервованого біта повертає невизначене значення: NA	NA

4.7.5.3 Регістр стану

Регістр стану I^2C має адреси: для I^2C0: I2STAT або I2C0STAT – 0xE001 C004, для I^2C1: I2C1STAT – 0xE005 C004, для I^2C2: I2C2STAT – 0xE008 0004.

Цей регістр доступний тільки для читання. Він містить код стану інтерфейса I^2C. Молодші три біти цього регістра завжди дорівнюють 0. Всього наявні 26 можливих кодів стану, 25 з них активні. Код стану F8h означає, що стан інтерфейса I^2C не змінювався і біт (прапорець) SI регістра I2CONSET не встановлено (тобто його значення дорівнює 0). Кожен із 25 значень кодів стану, що залишились, відповідають певному стану інтерфейса I^2C. Коли наступає будь–який з цих станів, автоматично встановлюється біт SI регістра I2CONSET.

Бітова структура регістра I2STAT наведена у таблиці 4.38.

Таблиця 4.38 – Біти регістра стану I^2C

Біти I2STAT	Назва	Опис	Значення після скидання
2:0	Стан	Ці біти завжди дорівнюють 0	0
7:3	Стан	Біти (код) стану	0x1F

4.7.5.4 Регістр даних

Регістр даних має адреси: для I^2C0:I2CDAT або I2C0DAT – 0xE001 C008, для I^2C1: I2C1DAT – 0xE005 C008, для I^2C2: I2C2DAT – 0xE008 0008.

Цей регістр містить дані, які будуть передані, або які були тільки що отримані через I^2C. Центральний процесор може проводити запис і читання регістра даних у будь–який час, крім моменту, коли проводиться прийом або передача байта шляхом апаратного зсуву вмісту регістра. Звернення до регістра даних можливо тільки в тому випадку, коли встановлено біт (прапорець) SI регістра I2CONSET. Дані в регістрі I2DAT залишаються стійкими до тих пір, поки залишається встановленим біт SI. Дані в I2DAT завжди зсуваються справа наліво: першим передається MSB (біт 7), а після завершення прийому байта даних перший прийнятий біт цього байта даних розташовується в MSB регістра I2DAT. Бітовая структура регистра I2DAT наведена в таблиці 4.39.

Таблиця 4.39– Біти регістра даних I^2C

Біти I2DAT	Назва	Опис	Значення після скидання
7:0	Дані	Передані/прийняті біти даних	0

4.7.5.5 Регістр адреси веденого

Регістр адреси веденого має адреси: для I^2C0: I2ADR або I2C0ADR – 0xE001 C00C, для I^2C1: I2C1ADR – 0xE005 C00C, для I^2C2: I2C2ADR – 0xE008 000C.

Цей доступний для читання і запису регістр використовується тільки тоді, коли інтерфейс I^2C працює в режимі веденого. У режимі ведучого, вміст регістра I2ADR не має ніякого значення. Молодший біт (GC) регістра I2ADR є бітом дозволу розпізнавання логікою I^2C адреси загального виклику (00h). Коли цей біт встановлено, адреса загального виклику (00h) буде розпізнаватися інтерфейсом I^2C.

Бітова структура регістра I2ADR наведена в таблиці 4.40.

Таблиця 4.40– Біти регістра адреси веденого I^2C

Біти I2ADR	Назва	Опис	Значення після скидання
0	GC	Біт дозволу розпізнавання загального виклику	0
7:1	Адреса	Адреса веденого в режимі веденого	0

4.7.5.6 Регістри програмування швидкості обміну I2SCL

Регістри програмування швидкості обміну I2SCL мають адреси: для I^2C0: I2SCLH або I2C0SCLH – 0xE001 C010, для I^2C1: I2C1SCLH – 0xE005 C010, для I^2C2: I2C2SCLH – 0xE008 0010, для I^2C0:I2SCLL або I2C0SCLL – 0xE001 C014, для I^2C1:I2C1SCLL – 0xE005 C014, для I^2C2:I2C2SCLL – 0xE008 0014.

Бітова структура регістрів I2SCLH, I2SCLL наведена відповідно у таблиці 4.41 і таблиці 4.42.

Таблиця 4.41– Біти регістра старшого доданку програмування швидкості обміну

Біти I2SCLH	Назва	Опис	Значення після скидання
15:0	Лічильні біти	Лічильні біти старшого доданку вибору періоду SCL	0x0004

Таблиця 4.42–Біти регістра молодшого доданку програмування швидкості обміну

Біти I2SCLL	Назва	Опис	Значення після скидання
15:0	Лічильні біти	Лічильні біти молодшого доданку вибору періоду SCL	0x0004

Програма–користувач може вибирати бажану швидкість передачі даних через інтерфейс I^2C шляхом встановлення значень регістрів I2SCLH і I2SCLL. Вміст регістра I2SCLH являє собою один 16–бітний доданок числа, яке визначає тривалість (в циклах Pclk) періоду синхроімпульсів на лінії SCL інтерфейсу I^2C,

а вміст регістра I2SCLL–інший доданок. Частота синхроімпульсів SCL обчислюється у відповідності з наступною формулою [4, 7]:

Частота передачі бітів = Pclk / (2 * (I2SCLH + I2SCLL)),

де (I2SCLH + I2SCLL) – значення суми вмісту регістрів I2SCLH та I2SCLL.

Значення I2SCLH і I2SCLL не обов'язково повинні бути однаковими. Вбудоване програмне забезпечення може задавати різну тривалість робочого циклу SCL, поміщаючи в I2SCLH і I2SCLL відповідні числа. Однак необхідно завжди стежити за тим, щоб частота передачі даних через інтерфейс I^2C лежала в діапазоні 0...400 кГц. Крім того, на значення регістрів I2SCLH і I2SCLL накладається таке обмеження: кожне з чисел, що містяться в них, повинно бути більшим або рівним чотирьом (0004h).

Можливі значення частоти передачі бітів для інтерфейсу I^2C, в залежності від вмісту регістрів I2SCLH, I2SCLL і значення дільника VPB наведені в таблицях 4.43, 4.44, 4.45.

Таблиця 4.43 – Вибір швидкості обміну через інтерфейс I^2C для значення дільника VPB = 1

I2SCLL + I2SCLH	Частота передачі бітів [кГц] при P_{CLK} (МГц)			
	P_{CLK}=16 МГц	P_{CLK} = 20 МГц	P_{CLK} = 40 МГц	P_{CLK} = 60 МГц
8	–	–	–	–
10	–	–	–	–
25	–	–	–	–
50	320	400	–	–
75	213,333	266,667	–	–
100	160	200	400	–
160	100	125	250	375
200	80	100	200	300
320	50	62,5	125	187,5
400	40	50	100	150
510	31,373	39,216	78,431	117,647
800	20	25	50	75
1280	12,5	15,625	31,25	46,875

Таблиця 4.44 – Вибір швидкості обміну через інтерфейс I^2C для значення дільника VPB = 2

I2SCLL + I2SCLH	Частота передачі бітів [кГц] при PCLK (МГц)			
	PCLK=16 МГц	PCLK = 20 МГц	PCLK = 40 МГц	PCLK = 60 МГц
8	–	–	–	–
10	–	–	–	–
25	320	400	–	–
50	160	200	400	–
75	106,667	133,333	266,667	400
100	80	100	200	300
160	50	62,5	125	187,5
200	40	50	100	150
320	25	31,25	62,5	93,75
400	20	25	50	75
510	15,686	19,608	39,216	58,824
800	10	12,5	25	37,5
1280	6,25	7,813	15,625	23,438

Таблиця 4.45 – Вибір швидкості обміну через інтерфейс I^2C для значення дільника VPB = 4

I2SCLL+ I2SCLH	Частота передачі бітів [кГц] при P_{CLK} (МГц)			
	PCLK=16 МГц	PCLK= 20 МГц	PCLK= 40 МГц	PCLK = 60 МГц
8	500	–	–	–
10	400	–	–	–
25	160	200	400	–
50	80	100	200	300
75	53,333	66,667	133,333	200
100	40	50	100	150
160	25	31,25	62,5	93,75
200	20	25	50	75
320	12,5	15,625	31,25	46,875
400	10	12,5	25	37,5
510	7,843	9,804	19,608	29,412
800	5	6,25	12,5	18,75
1280	3,125	3,906	7,813	11,719

На рисунку 4.43 наведена спрощена структурна схема модуля I^2C мікроконтролерів сімейства LPC2300.

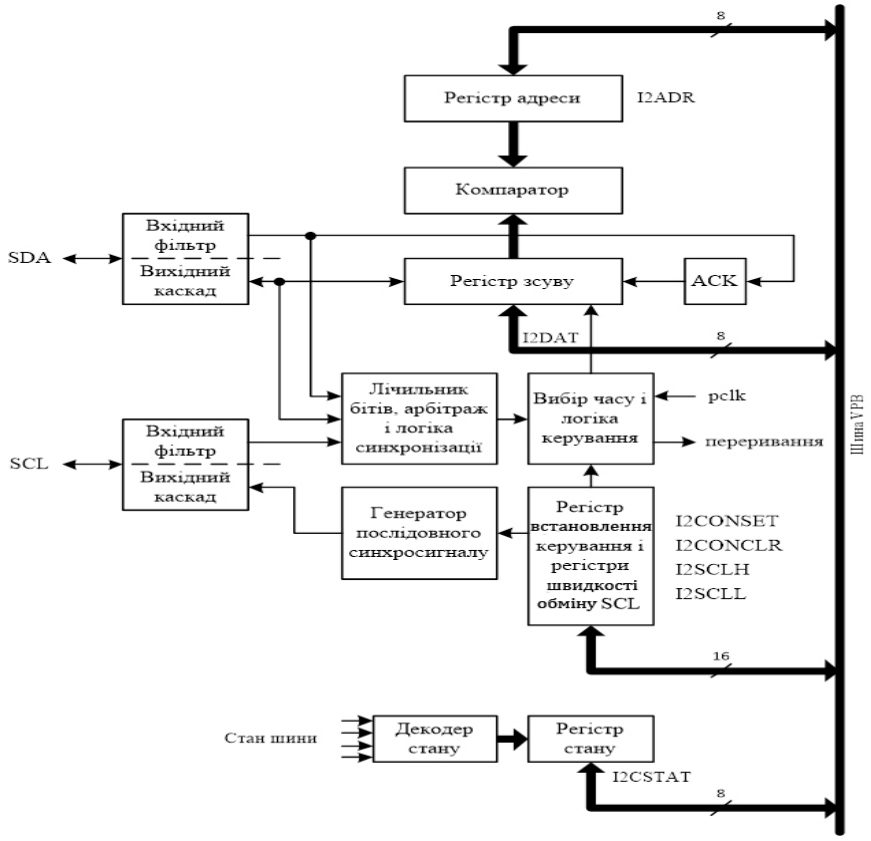


Рисунок 4.43 – Структурна схема модуля I^2C

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Назвіть основні характерні особливості інтерфейсу I^2C.
2) Наведіть приклади I^2C–сумісних пристроїв.
3) З яких ліній складається шина інтерфейсу I^2C?
4) Наведіть та опишіть структурну схему мережі з інтерфейсом I^2C.
5) Назвіть коректні стани сигналів на шині під час обміну даними.
6) Назвіть та опишіть чотири режими (типи) обміну даними.
7) Наведіть та опишіть часові діаграми, які відображають обмін даними шиною I^2C.
8) Опишіть типові особливості інтерфейсу I^2C мікроконтролерів сімейства LPC2300
9) Опишіть роботу інтерфейсу I^2C мікроконтролерів сімейства L3C2300 в режимі ведучого передавача.
10) Опишіть роботу інтерфейсу I^2C мікроконтролерів сімейства L3C2300 в режимі ведучого приймача.
11) Опишіть роботу інтерфейсу I^2C мікроконтролерів сімейства L3C2300 в режимі веденого приймача.
12) Опишіть роботу інтерфейсу I^2C мікроконтролерів сімейства L3C2300 в режимі веденого передавача.
13) Опишіть взаємодію прикладної програми з модулем I^2C.
14) Назвіть керуючі регістри модуля I^2C.
15) Опишіть біти регістр керування встановленням.
16) Опишіть біти регістра стану.
17) Опишіть біти регістра даних.
18) Опишіть біти регістра адреси веденого.
19) Опишіть біти регістрів програмування швидкості обміну.
20) Опишіть біти регістра керування скиданням.
21) Опишіть структурну схему модуля I^2C мікроконтролерів сімейства LPC2300.

4.8 Інтерфейс SPI

4.8.1 Основні положення

Модуль інтерфейсу SPI (рисунок 4.44) є простим периферійним «виконавчим пристроєм», що може видавати та читати дані на/з шини SPI, однак недостатньо інтелектуальним, щоб керувати шиною [1, 4, 7]. Після ініціалізації інтерфейсу програма повинна сама керувати процесом передачі даних шиною.

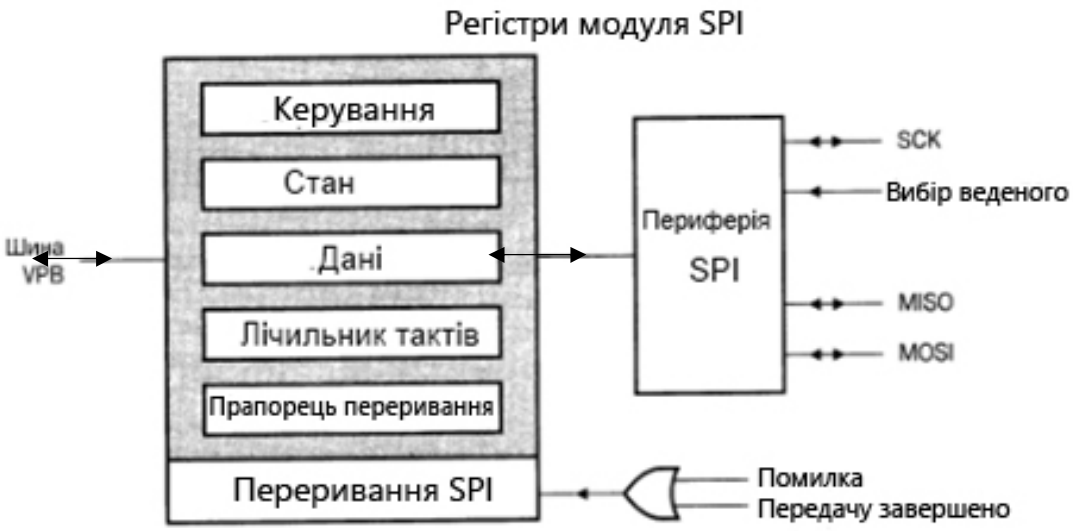


Рисунок 4.44 – Модуль інтерфейсу SPI

SPI (англ. Serial Peripheral Interface) – послідовний синхронний повнодуплексний стандарт передачі даних, який розроблено фірмою Motorola для забезпечення простого сполучення мікроконтролерів та периферії. SPI також називають чотирьохпровідним інтерфейсом. На відміну від стандартного асинхронного послідовного порту, SPI є синхронним інтерфейсом, в якому кожна передача синхронізована з тактовим сигналом, що генерується ведучим пристроєм (мікроконтролером). Периферійний пристрій синхронізує отримання бітової послідовності з тактовим сигналом. До одного послідовного периферійного інтерфейсу ведучого мікроконтролера можна під'єднати декілька ведених. Головний пристрій вибирає ведений пристрій, активуючи сигнал «вибір кристалу» (англ. chip select) на потрібній мікросхемі. Інші пристрої, які не вибрані ведучим, не беруть участі в передачі по SPI.

Інтерфейс SPI має чотири сигнальні лінії: MOSI, MISO, SCK та (NSS) (рисунки 4.45, 4.46).

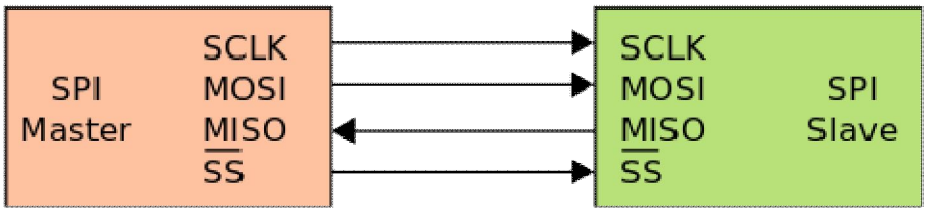


Рисунок 4.45 – SPI–шина: один ведучий та один ведений

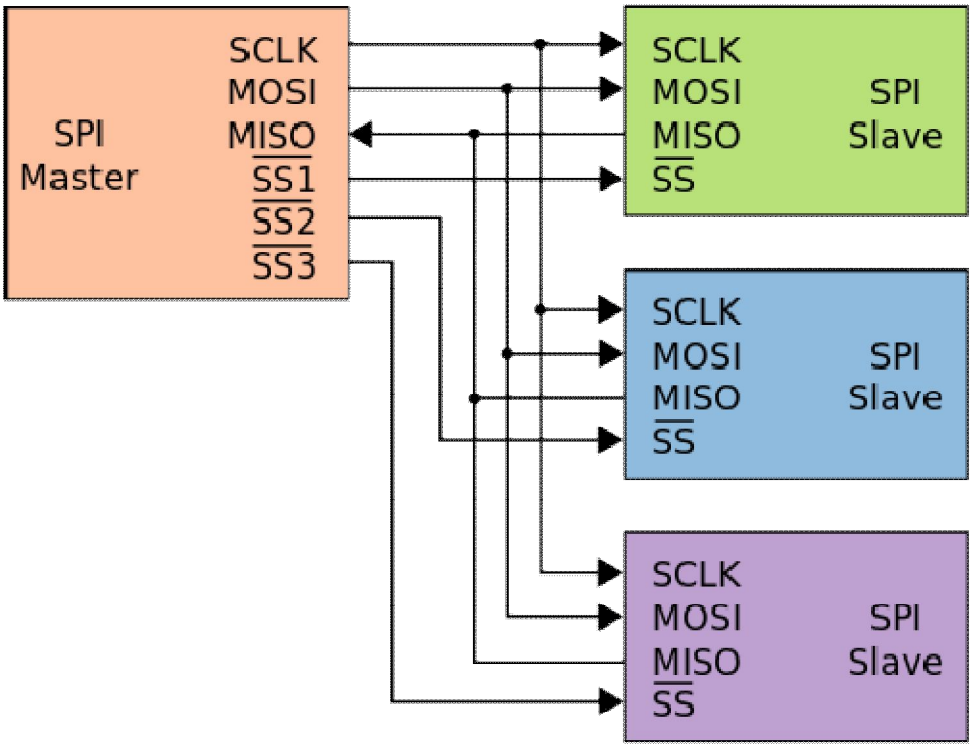


Рисунок 4.46 – SPI–шина: один ведучий, три незалежні (паралельні) ведені

Лінія MOSI (Master–Out, Slave–In) – вихідна лінія даних ведучого інтерфейсу і вхідна лінія даних веденого інтерфейсу. З назви випливає, що лінію призначено для передачі даних від ведучого (Master) інтерфейсу (або вузла мережі) до веденого (Slave) інтерфейсу (або вузла мережі).

Лінія MISO (Master–In, Slave–Out) – вхідна лінія даних ведучого інтерфейсу і вихідна лінія даних веденого інтерфейсу. Лінію призначено для передачі даних від веденого інтерфейсу до ведучого. Дані передаються байтами, починаючи зі старшого біта. Слід пам'ятати, що вихід MISO веденого інтерфейсу

знаходиться у високоімпедансному стані, якщо ведений інтерфейс не обрано за лінією NSS.

Лінію $\overline{SS}$ (NSS) (Slave Select) – лінію вибірки веденого, призначено для вибірки ведучим веденого інтерфейсу низьким логічним потенціалом.

Лінія SCLK (Serial Clock) – вихідна лінія тактових імпульсів ведучого вузла і вхідна лінія тактових імпульсів веденого вузла. Лінія SCLK використовується для синхронізації передачі даних між ведучим і веденим інтерфейсами за лініями MOSI та MISO.

У режимі ведучого на виводі синхронізації формується тактовий сигнал, що забезпечує передачу даних зі швидкістю до 400 Кбіт / с, а в режимі веденого на цей вивід надходить зовнішній тактовий сигнал від ведучого пристрою. Шина SPI розроблялася виключно для високошвидкісної послідовної передачі даних і, на відміну від шини I^2C, в ній не реалізовано ніякої схеми адресації.

Опис виводів інтерфейсу SPI наведено в таблиці 4.46.

Таблиця 4.46 – Опис виводів інтерфейсу SPI

Назва виводу	Опис виводу	Тип виводу SPI	
		Ведучий	Ведений
MISO	Вхід ведучого – вихід веденого	Вхід	Вихід
MOSI	Вихід ведучого – вхід веденого	Вихід	Вхід
SCLK	Послідовний синхросигнал	Вихід	Вхід
NSS ($\overline{SS}$)	Вибір веденого	Вихід	Вхід

Керування модулем SPI здійснюється за допомогою п’яти регістрів. Регістр лічильника тактів визначає швидкість передачі даних шиною. Тактовий сигнал шини SPI отримується простим діленням сигналу P_{clk} на число, що знаходиться в регістрі лічильника тактів. Мінімальне значення, яке зберігається в цьому регістрі дорівнює 8. Регістр керування призначений для визначення конфігурації шини SPI. Через примітивний характер процесу передачі шиною SPI і широкого діапазону пристроїв, в специфікації інтерфейсу передбачено можливість функціонування ліній синхронізації в декількох різних режимах.

Можливі чотири комбінації фази (CPHA) і полярності (CPOL) сигналу SCLK по відношенню до сигналів даних. Режими роботи визначаються комбінацією біт CPHA і CPOL (рисунок 4.47):

– CPOL = 0 – сигнал синхронізації починається з низького рівня;
– CPOL = 1 – сигнал синхронізації починається з високого рівня;
– CPHA = 0 – вибірка даних проводиться за переднім фронтом сигналу синхронізації;
– CPHA = 1 – вибірка даних проводиться за заднім фронтом сигналу синхронізації.

Часові діаграми роботи SPI–інтерфейсу показано на рисунку 4.47.

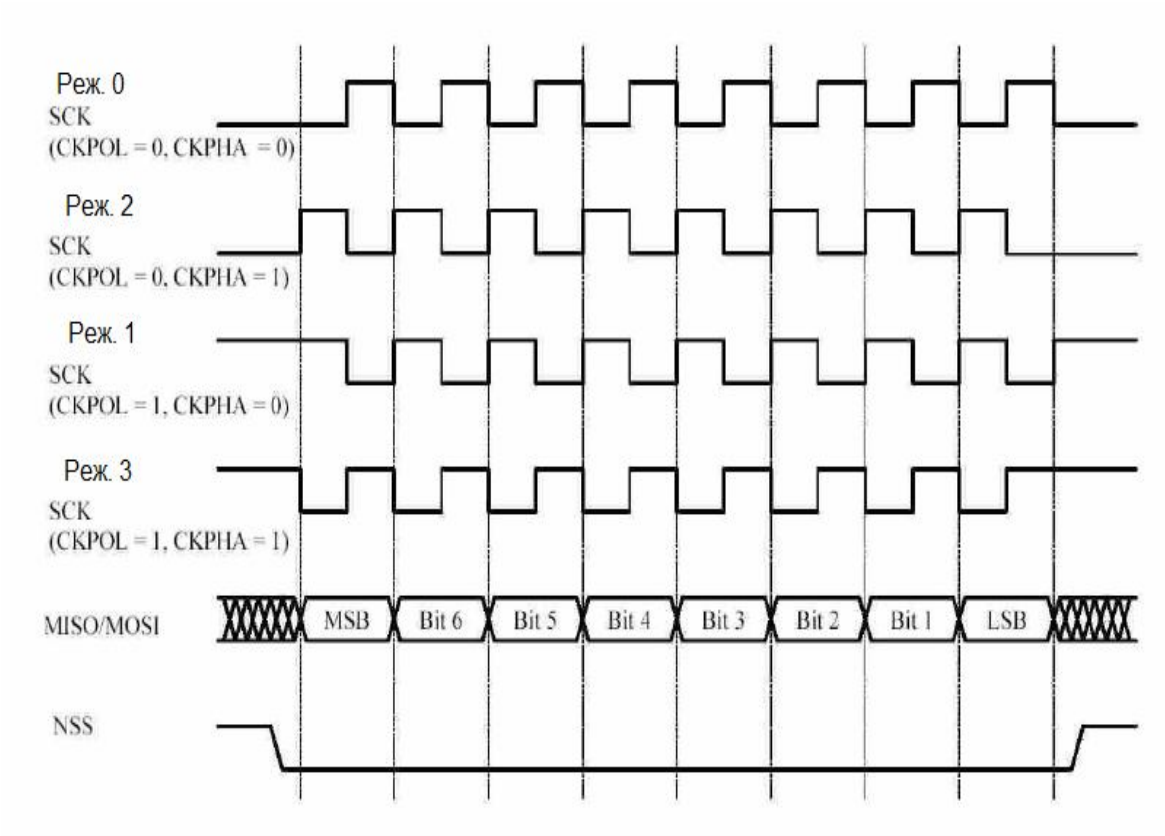


Рисунок 4.47– Часові діаграми роботи SPI–інтерфейсу

Для позначення режимів роботи інтерфейсу SPI прийнято наступну угоду:

– режим 0 (CPOL = 0, CPHA = 0);
– режим 1 (CPOL = 1, CPHA = 0);
– режим 2 (CPOL = 0, CPHA = 1);
– режим 3 (CPOL = 1, CPHA = 1).

У мережі на базі SPI–інтерфейсів одночасно тільки один інтерфейс може бути ведучим. Інтерфейс налаштовується на режим ведучого встановленням прапорця MSTR регістра S0SPCR. Якщо інтерфейс налаштовано на режим ведучого, то запис байта даних у регістр даних S0SPDR приводить до початку передачі. Ведучий інтерфейс негайно побітно зсуває дані і видає їх на лінію MOSI в супроводі тактових імпульсів на лінії SCLK. Після завершення передачі встановлюється прапорець SPIF регістра S0SPSR. Якщо дозволено переривання, то виконується відповідна підпрограма обробки цього переривання. Крім того, інтерфейс може бути запрограмований на видачу від одного до п’ятнадцяти бітів (таблиця 4.49) для здійснення зв'язку з SPI–приладами, що мають різну довжину слова. Довжину передачі (кількість переданих бітів) може бути задано бітами BitEnable та BITS в регістрі керування S0SPCR. З'єднання двох мікроконтролерів (ведучий–ведений) за інтерфейсом SPI, приведено на рисунку 4.48.

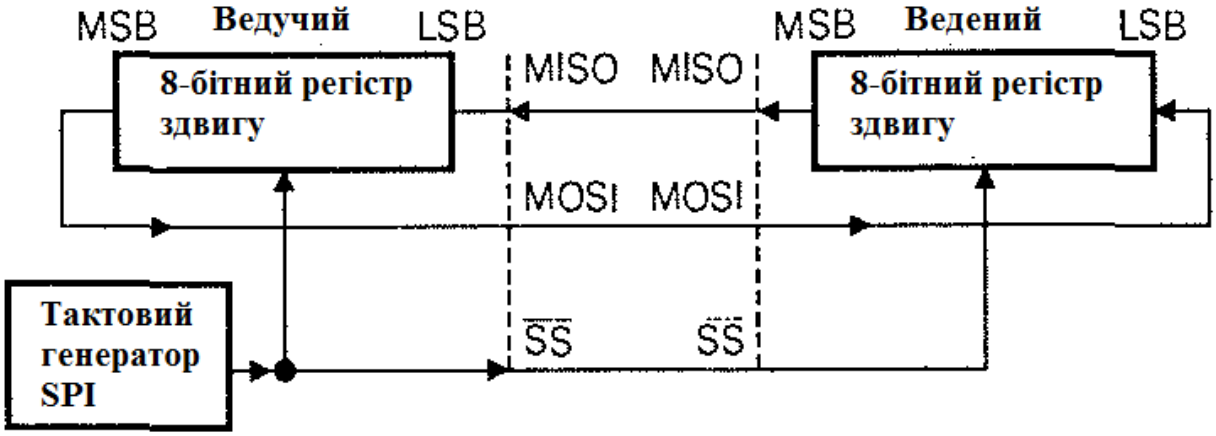


Рисунок 4.48– Схема з’єднання вузлів SPI двох різних пристроїв

Інтерфейс може працювати в дуплексному режимі. Це означає, що можлива одночасна передача даних за лінією MOSI від ведучого до веденого, і лінією MISO від веденого до ведучого. Дані, які отримано від веденого інтерфейсу, замінюють дані в регістрі даних ведучого інтерфейсу. Цей регістр двічі буферизований на введення, але не на виведення. Тобто якщо в регістр даних S0SPDR здійснюється спроба запису даних під час передачі попереднього байта, встановлюється прапорець помилки запису WCOL регістра S0SPSR і спроба запису ігнорується.

Таким чином, поточна передача даних триває безперервно. Зчитування з регістра даних S0SPDR призводить до зчитування приймального буфера. Якщо прийом не закінчено, встановлюється прапорець ROVR регістра S0SPSR. Нові дані не передаються в регістр зчитування, доки попередній прийнятий байт не буде прочитано. Очевидно, що при затримці зчитування прийнятих байтів може відбутися втрата даних. Якщо SPI–інтерфейс не налаштовано, як ведучий (Master), він буде працювати в режимі веденого (Slave).

Крім того, підтримується режим мережі з багатьма ведучими. Прапорець помилки режиму MODF регістра S0SPSR встановлюється в логічну одиницю, якщо інтерфейс визначено як ведучий (MSTR = 1) і вивід NSS переведено в низький логічний рівень, тобто SPI–інтерфейс намагаються використовувати в якості веденого.

Можливі чотири комбінації фаз тактових імпульсів і їх полярностей залежно від комбінації керуючих бітів у регістрі керування S0SPCR. Біт CKPHA вибирає одну з двох фаз тактових імпульсів, тобто фронт, за яким здійснюється запис даних. Інший біт CKPOL визначає активну полярність сигналу синхронізації (високий чи низький рівень). Очевидно, що і ведучий, і ведений вузли повинні мати однакові налаштування фази і полярності. Ще одна важлива особливість налаштування полягає в тому, що інтерфейс SPI повинен бути заборонений шляхом скидання BitEnable регістра S0SPCR на час налаштування фази і полярності тактових імпульсів.

Перед входами восьмирозрядних регістрів зсуву ведучого та веденого мікроконтролерів (рисунок 4.48) знаходяться два синхронних тригери (буфери) (на рисунку не показані), в які першим перепадом сигналу на лінії SCLK записується значення сигналу, яке присутнє на їх інформаційному вході.

Другим перепадом імпульсів на лінії SCLK відбувається зсув інформації вліво відповідно до рисунку 4.48. При цьому стан буферів переписується в молодші розряди регістрів зсуву, а черговий вихідний біт з регістрів зсуву виставляється на лінії MOSI/MISO.

Вказаний зсув у часі між моментом видачі чергового розряду в лінію зв'язку між мікроконтролерами і моментом фіксації цього біта в буфері дозволяє компенсувати часові затримки при передачі сигналів між ведучим та веденим.

Далі аналогічно здійснюється обмін між ведучим та веденим всіма наступними бітами. З кожним непарним/парним перепадом сигналу SCLK відбувається фіксація чергового біта в буфері, а з кожним парним/непарним – зсув інформації вліво.

Крім того, можна задати порядок передачі байтів, оскільки першим може передаватися як старший, так і молодший біт байта (рисунок 4.49).

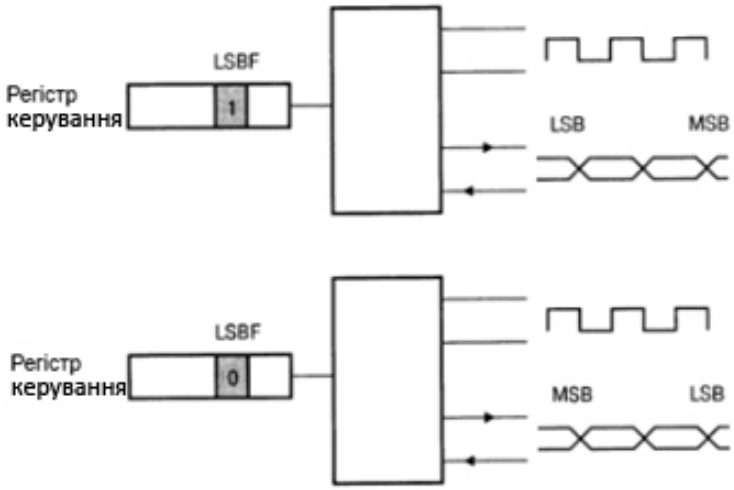


Рисунок 4.49– Прямий та зворотний порядок передачі байта

Кожній з вказаних опцій конфігурації відповідає окремий біт регістра керування, і потрібно запрограмувати ці біти так, щоб це відповідало характеристикам SPI – пристроїв, з якими потрібно обмінюватися даними.

4.8.2 Передача даних в режимі ведучого

Наступна послідовність дій описує, як слід керувати передачею даних з блоку SPI, коли він буде налаштований, як ведучий (керуючий).

Цей процес передбачає, що будь–яка попередня передача даних вже завершена:

1. Встановити необхідне значення в регістрі лічильника тактів SPI.

2. Встановити необхідні налаштування в регістрі керування SPI.

3. Записати дані, що будуть передаватися, в регістр даних SPI. Цей запис ініціює передачу даних шиною SPI.

4. Почекати, поки біт SPIF в регістрі стану SPI стане дорівнювати 1. Біт SPIF буде встановлено в 1 після останнього циклу передачі даних по SPI.

5. Прочитати регістр стану SPI.

6. Прочитати отримані дані з регістра даних SPI (за бажанням).

7. Перейдіть до кроку 3, якщо треба передати більше даних.

4.8.3 Передача даних в режимі веденого

Наступна послідовність дій описує, як слід керувати передачею даних з блоку SPI, коли він буде налаштований, як ведений. Цей процес передбачає, що будь–яка попередня передача даних вже завершена:

1. Встановити необхідні значення в регістрі керування SPI.

2. Записати дані, що потрібно передати в регістр даних SPI (за бажанням). Зауважимо, що тільки це може бути зроблено, коли в режимі веденого не виконується прийом шиною SPI.

3. Почекати, поки біт SPIF в регістрі стану SPI стане дорівнювати 1. Біт SPIF буде встановлено в 1 після останнього циклу передачі даних по SPI.

4. Прочитати регістр стану SPI.

5. Прочитати отримані дані з регістра даних SPI (за бажанням).

6. Перейдіть до кроку 2, якщо треба передати більше даних.

4.8.4 Переривання SPI

4.8.4.1 Переривання помилки переповнення

Переривання SPI виникає при встановленні прапорця SPIF при SPIE=1 та можливе при виникненні наведених нижче помилок. Переривання помилки переповнення виникає при спробі передачі даних до блоку SPI в ситуації, коли внутрішній буфер містить дані, які не були прочитані процесором. Буфер читання містить достовірні дані, на що вказує значення SPIF біта в регістрі стану. Якщо значення біта SPIF дорівнює 1 (тобто буфер повний), усі нові дані будуть втрачені, і біт переповнення (ROVR) в регістрі стану буде дорівнювати 1.

4.8.4.2 Переривання помилки запису

Між блоком інтерфейсу шини SPI і внутрішнім регістром зсуву відсутній буфер передачі (запису). В результаті, не можна записувати дані до регістра даних SPI під час передачі даних. Запис неможливий починаючи з моменту початку передачі даних шиною SPI до моменту, коли передача закінчиться і біт SPIF дорівнюватиме 1.

При спробі запису в цей проміжок часу даних до регістра даних, усі дані для передачі будуть втрачені і біт помилки запису (WCOL) буде змінений на 1.

4.8.4.2 Переривання помилки режиму

Якщо сигнал NSS на вході $\overline{SS}$ активний, коли блок SPI працює в режимі ведучого, це вказує на те, що інший блок SPI, що працює в режимі ведучого обрав даний блок веденим. Такий процес викликає помилку режиму. При виникненні цієї помилки встановлюється в 1 біт помилки режиму (MODF) в регістрі стану, всі передачі перериваються і блок SPI переходить в режим веденого.

4.8.5 Опис регістрів SPI

Карту регістрів модуля SPI наведено в таблиці 4.47.

Таблиця 4.47– Карта регістрів модуля SPI

Ім'я	Опис	Доступ	Значення за замовченням	Адреса
S0SPCR	Регістр керування	Запис/зчитування	0x00	0xE002000
S0SPSR	Регістр стану	Зчитування	0x00	0xE002004
S0SPDR	Регістр даних	Запис/зчитування	0x00	0xE002008
S0SPCCR	Регістр лічильника тактів	Запис/зчитування	0x00	0xE00200C
S0SPINT	Регістр прапорців переривань	Запис/зчитування	0x00	0xE00201C

Формат та опис регістра керування S0SPCR наведено в таблиці 4.48.

Таблиця 4.48– Регістр керування S0SPCR

Біт	Символ	Опис	Значення по замовченню
1:0	–	Зарезервовані	–
2	BitEnable	0 – передача заборонена 1 – SPI–контролер передає та приймає кількість біт, що вказана у бітах 11:8. Відповідна кількість біт наведена в таблиці 4.49	0
3	CPHA	Визначає фазу сигналу	0
4	CPOL	Визначає полярність сигналу	0
5	MSTR	0 – працює в режимі веденого 1 – працює в режимі ведучого	0
6	LSBF	Визначає з якого біта (старшого – 0 чи молодшого – 1) починати передачу	0
7	SPIE	Дозвіл переривань	0
11:8	BITS	Кількість біт, що передаються	0
15:12	–	Зарезервовані	–

Кількість біт, що будуть передаватись та прийматись SPI–контролером, відповідно до значень в бітах 11:8 наведено в таблиці 4.49.

Таблиця 4.49– Кількість біт, яку передає та приймає SPI–контролер

11р	10р	9р	8р	Число біт
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Формат та опис регістра S0SPSR наведено в таблиці 4.50.

Таблиця 4.50– Регістр стану S0SPSR

Біт	Символ	Опис	Значення за замовченням
2:0	–	Зарезервовані	–
3	ABRT	Біт аварійного припинення роботи веденого. Коли біт встановлений, то це вказує на те, що ведений пристрій аварійно завершив роботу. Очищається після читання регістра S0SPSR.	0
4	MODF	Біт помилки режиму. Коли біт встановлений, то це вказує на те, що сталася помилка режиму. Цей біт скидається шляхом читання регістра S0SPSR і подальшого запису в регістр керування SPI.	0
5	ROVR	Біт переповнення при читанні. Коли біт встановлений, це вказує, що сталося переповнення при читанні. Цей біт скидається шляхом читання регістра S0SPSR.	0
6	WCOL	Біт колізії запису. Коли біт встановлений, це вказує, що сталася колізія записи. Цей біт скидається шляхом читання регістра S0SPSR і подальшого звернення до регістра даних SPI.	0
7	SPIF	Прапорець закінчення передачі через SPI. Коли прапорець встановлений, це вказує на закінчення передачі даних через SPI. Коли пристрій є ведучим, цей прапорець встановлюється в кінці останнього циклу передачі. Коли пристрій є веденим, цей прапорець встановлюється за перепадом SCLK, що здійснює вибірку останнього біта даних. Цей прапорець скидається шляхом першого читання регістра S0SPSR і подальшого звернення до регістра даних SPI.	0

Формат та опис регістра прапорців переривання S0SPINT наведено в таблиці 4.51.

Таблиця 4.51– Регістр прапорців переривань SPI

Біт	Символ	Опис	Значення за замовченням
0	SPI Interrupt Flag (переривання від SPI)	Прапорець переривання SPI. Встановлюється інтерфейсом при виникненні переривання. Встановлюється при SPIF = 1 і при одній з трьох умов: ROVR=1, WCOL=1, або MODF=1. Прапорець скидається шляхом запису в нього 1.	0
7:1	–	Зарезервовані	–

Формат та опис регістра S0SPDR наведено в таблиці 4.52.

Таблиця 4.52– Регістр даних S0SPDR

Біт	Символ	Опис	Значення за замовченням
7: 0	DataLow	Дані двонаправленого порту	0x00
15: 8	DataHigh	Якщо біт 2 SPCR 1 і біти 11: 8 відмінні від 1000, то деякі або всі ці біти містять додаткові біти передачі і прийому. Якщо вибираються менш, ніж16 бітів, то інші біти заповнюються нулями	0x00

Формат та опис регістра S0SPCCR наведено в таблиці 4.53

Таблиця 4.53– Регістр лічильника тактів S0SPCCR

Біт	Символ	Опис	Значення за замовченням
7: 0	Лічильник	Встановлення кількості тактів	0x00

Цей регістр визначає частоту SCLK ведучого модуля. Регістр програмує частоту (швидкість) обміну модулем SPI. У ведучому режимі, цей регістр має бути парним числом: більшим або дорівнювати 8. Порушення цього може привести до непередбачуваного. Частота (швидкість) SPI0 – SCLK може розраховуватися за формулою: $F_{SCLK} = P_{CLK}$ SPI / значення S0SPCCR. У режимі веденого, тактова частота не повинна перевищувати 1/8 від частоти ведучого SPI.

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) На яку відстань і з якою швидкістю можна здійснювати обмін даними через інтерфейс SPI?
2) У яких режимах може працювати мікроконтролер при обміні даними інтерфейсом SPI?
3) Скільки та які виводи мікроконтролера використовує модуль SPI?
4) Який регістр призначений для керування модулем SPI?
5) Який регістр використовується для контролю стану модуля?
6) Як задати режим роботи мікроконтролера в режимі «MASTER»?
7) Як визначити кінець передачі байта?
8) За якої умови, при передачі даних від ведучого до веденого, можлива передача у зворотному напрямі?
9) Що означає те, що у модулі реалізована одинарна буферизація при передачі та подвійна при прийомі?
10) Чи можна до інтерфейсу підключати декілька периферійних пристроїв?
11) Назвіть кількість режимів передачі даних та принцип їх роботи.
12) Що є джерелом тактового сигналу при роботі модуля?
13) Наведіть опис бітів регістра керування.
14) Наведіть опис бітів регістра стану.
15) Наведіть опис бітів регістра прапорців переривання.
16) Опишіть передачу даних в режимі ведучого.
17) Опишіть передачу даних в режимі веденого.
18) Опишіть переривання від модуля SPI.
19) Опишіть часові діаграми роботи SPI–інтерфейсу.
20) Опишіть схему з’єднання вузлів SPI двох різних пристроїв.
21) Як компенсуються часові затримки при передачі сигналів між двома пристроями?
22) Наведіть та опишіть приклад мережі SPI.
23) За якою формулою розраховується тактова частота модуля SPI?

4.9 Модуль аналого–цифрового перетворювача (АЦП)

4.9.1 Основні відомості про АЦП

4.9.1.1 Призначення та види АЦП

Аналого–цифрові перетворювачі (АЦП) – це пристрої, що перетворюють вхідні аналогові сигнали у відповідні їм цифрові сигнали, придатні для роботи з ЦОМ і іншими цифровими пристроями. АЦП широко застосовуються в пристроях дискретної автоматики, цифрових системах керування для перетворення аналогових сигналів від датчиків у цифрову форму, у системах відображення інформації для цифрової індикації, у системах передачі даних і багатьох інших областях техніки.

Різні за фізичною природою сигнали, що знімаються з датчиків, і що характеризують контрольований процес, спочатку перетворюються в електричний сигнал, а потім вже за допомогою перетворювачів “напруга–код” – у цифрові. На вході АЦП, як правило, є напруга, яка постійно чи повільно змінюється, а з виходу знімаються дані в паралельному двійковому коді.

4.9.1.2 Розрахунок АЦП

В АЦП здійснюється квантування (дискретизація) за рівнем і часом (рисунок 4.50). На вхід перетворювача надходить аналогова напруга U_{BX}, що перетворюється в дискретну величину, яка визначається у фіксовані моменти часу найближчим до безперервної (аналогової) величини рівнем квантування.

На виході АЦП кожному дискретному значенню відповідає комбінація двійкового коду, число розрядів якого позначимо буквою N_P. Величина N_P залежить від числа дискретних значень $N_д$ на виході АЦП, включаючи нульове. Вибір N_P робиться відповідно до співвідношення:

$$2^{N_p} \geq N_д. \tag{4.1}$$

Число дискретних значень (рівнів квантування) залежить від похибки квантування за рівнем.

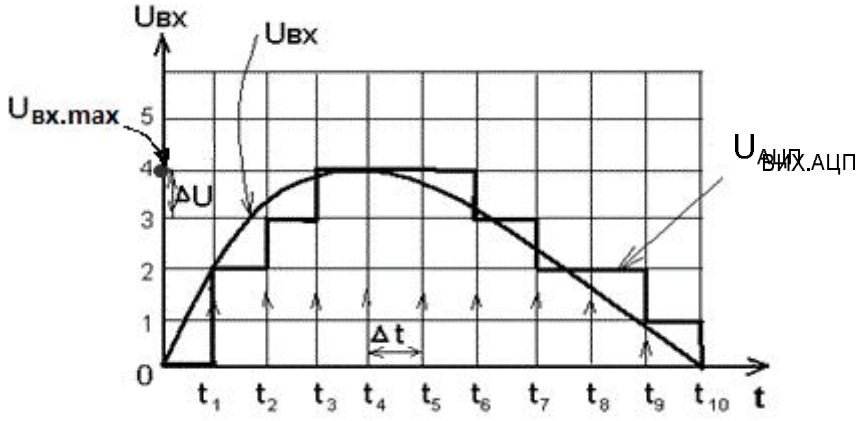


Рисунок 4.50– Квантування (дискретизація) аналогової величини за рівнем і часом

Абсолютна похибка квантування за рівнем

$$\delta_{АБС} \le \frac{\Delta U}{2}, \tag{4.2}$$

де ΔU – величина кроку квантування за рівнем, що дорівнює

$$\Delta U = \frac{U_{BX\,MAX} - U_{BX\,MIN}}{N_Д - 1}. \tag{4.3}$$

З наведеного співвідношення (4.2) випливає, що максимальна абсолютна похибка дорівнює половині кроку квантування за рівнем. Відносна похибка квантування за рівнем

$$\delta_{ВІД} \le \frac{\delta_{АБС} \cdot 100\%}{U_{BX.MAX} - U_{BX.MIN}} = \frac{\Delta U \cdot 100\%}{\{2 \cdot (U_{BX.MAX} - U_{BX.MIN})\}} =$$

$$= \frac{(U_{BX.MAX} - U_{BX.MIN}) \cdot 100\%}{\{(N_Д - 1) \cdot 2 \cdot (U_{BX.MAX} - U_{BX.MIN})\}} = \frac{50}{N_Д - 1}[\%]. \tag{4.4}$$

У наведеній формулі з $N_Д$ віднімається одиниця, тому що одним з дискретних значень є нульове. Звідси необхідне число дискретних значень, що відображує нашу безперервну функцію з заданою точністю, визначається як

$$N_Д \ge \frac{50}{\delta_{ВІД}} + 1. \tag{4.5}$$

Наприклад, при $\delta_{ВІД} \le 0{,}2\%$ $N_Д$ повинно бути не менше 251. Приймаючи $N_Д$=256 визначаємо, що число розрядів $N_Р$ у цьому випадку повинно бути 8

(2^8=256). Якщо вхідна безперервна величина змінюється, наприклад, у діапазоні від 0 до 2,55 В, то величина кроку квантування за рівнем при $N_Д$=256 дорівнює ΔU=10 мВ; $\delta_{АБС}$ ≤5 мВ; $\delta_{ВІД}$ ≤ 50/255 < 0,2%.

При проектуванні АЦП важливе значення має вибір величини кроку квантування за часом Δt=T. Значення T визначає необхідну швидкодію АЦП і каналу обробки інформації, та за теоремою Котельникова повинне відповідати виразу:

$$\Delta t = T \le \frac{1}{2 f_{MAX}}, \tag{4.6}$$

де f_{MAX} – частота вищої гармоніки спектру вхідного аналогового сигналу АЦП.

Фізично цей вираз варто трактувати в такий спосіб: на один період максимальної гармоніки вхідного аналогового сигналу при переході від аналогової до дискретної величини необхідно взяти не менш двох відліків.

4.9.1.3 Пристрій вибірки і зберігання (ПВЗ)

При аналого–цифровому перетворенні швидко змінюваних сигналів виникають динамічні похибки, що визначаються, по–перше, частотою і часом перетворення, а, по–друге, – апертурною похибкою.

Похибка, що виникає через невідповідність вхідного сигналу перетвореному цифровому значенню, називається апертурною похибкою АЦП. Ця невідповідність виникає, якщо зміна вхідного сигналу під час перетворення еквівалентна більш ніж одиниці молодшого значущого розряду (МЗР). МЗР в зарубіжній літературі позначається LSB (Least Significant Bit – молодший значущий розряд), аналогічно, MSB (Most Significant Bit – старший значущий розряд) відповідає позначенню СЗР.

У цьому випадку, при швидко змінюваному у часі вхідному сигналі створюється невизначеність у тім, яким у дійсності було миттєве значення вхідного сигналу в момент вибірки.

Час між моментом фіксації миттєвого значення вхідного сигналу (моментом відліку) і моментом одержання його цифрового еквівалента називається апертурним часом.

Апертурна похибка визначається збільшенням змінюваного у часі вхідного сигналу АЦП за час перетворення. Якщо прийняти, що для N_P – розрядного АЦП апертурна похибка не повинна перевищувати кроку квантування за рівнем ΔU_{BX} (рисунок 4.51), то між частотою сигналу f, апертурним часом t_A й апертурною похибкою має місце співвідношення [6]:

$$\Delta U_{BX} = \frac{U_m}{2^{Np} - 1} \geq U_m \cdot 2\pi f \cdot t_A. \tag{4.7}$$

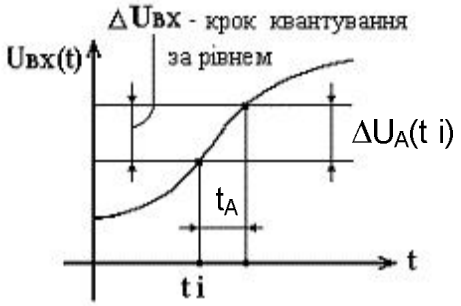


Рисунок 4.51– Визначення апертурної похибки біля *i*–ї точки відліку

Розділивши ліву і праву частини нерівності (4.7) на U_m отримаємо:

$$\frac{1}{2^{Np} - 1} \geq 2\pi f \cdot t_A. \tag{4.8}$$

Наприклад, якщо N_P = 8, а час перетворення АЦП $t_{ПРТ}$ = 7,5 мкс, то для того, щоб апертурна похибка не перевищувала одиниці молодшого значущого розряду двійкового коду на виході АЦП, частота вхідного сигналу не повинна перевищувати 83 Гц.

Для зменшення апертурної похибки АЦП звичайно використовуються пристрої вибірки і зберігання (ПВЗ), що включаються між входом АЦП і виходом джерела аналогового сигналу.

ПВЗ призначений для запам'ятовування миттєвого значення вхідного аналогового сигналу в момент вибірки і підтримки цього значення на постійному рівні під час перетворення інформації в АЦП. Подібний пристрій необхідно застосовувати в тих випадках, коли за час перетворення інформації в АЦП зміна його вхідного аналогового сигналу еквівалентна дискретній зміні вихідного сигналу більш ніж на одиницю молодшого значущого розряду (МЗР). У якості ПВЗ може бути, наприклад, використана мікросхема К1100СК2. На рисунку 4.52

зображено позначення цієї мікросхеми на електричних схемах і пояснюється, яким чином ПВЗ пов'язаний з іншими частинами мікропроцесорної системи керування (МПСК). Тривалість імпульсу запису інформації у ПВЗ (імпульсу вибірки) $t_{ЗАП}$ ($t_В$) при значенні ємності збереження $C_{ЗБ}$=1нф дорівнює 5 мкс.

Сучасні мікросхеми АЦП та модулі АЦП у складі мікроконтролерів мають у своєму складі ПВЗ.

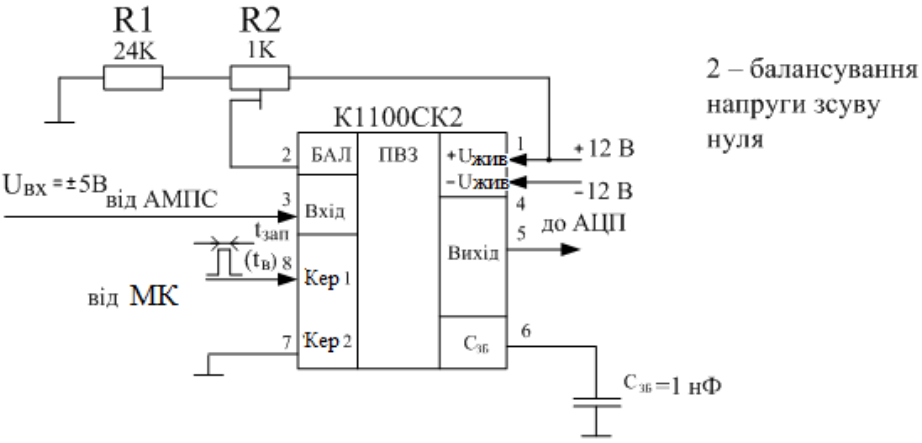


Рисунок 4.52 – Схема включення ПВЗ

4.9.1.4 АЦП послідовного наближення

Різним методам побудови АЦП відповідають пристрої, що розрізняються за точністю, швидкодією, завадостійкістю, складністю реалізації і т. ін. Одним з найбільш розповсюджених є метод послідовного наближення, який застосовується в АЦП, орієнтованих на використання в мікропроцесорних системах (МПС), наприклад, К1113 ПВ1; К572 ПВ3 і т. ін. [2, 5]. На рисунку 4.53 приведена спрощена структурна схема АЦП послідовного наближення.

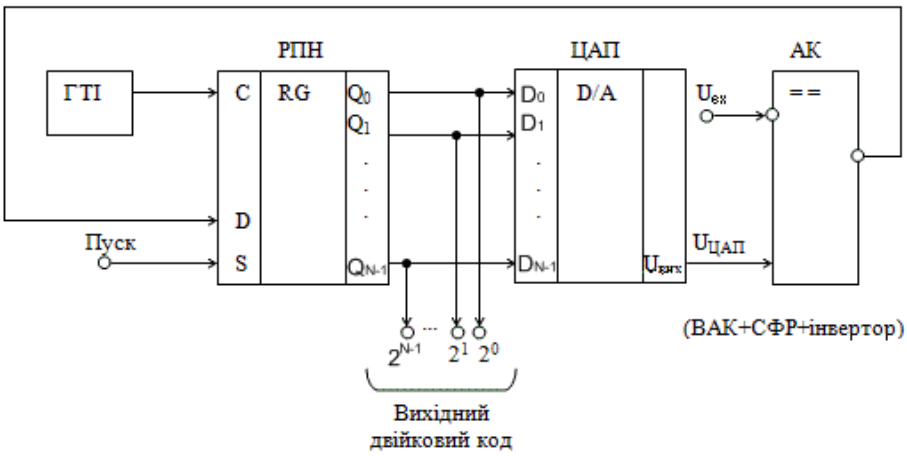


Рисунок 4.53– Структура АЦП послідовного наближення

АЦП містить регістр послідовного наближення (РПН), цифро–аналоговий перетворювач (ЦАП), аналоговий компаратор (АК) і генератор тактових імпульсів (ГТІ). Після надходження імпульсу ПУСК на виході старшого (N–1)–го розряду регістра послідовного наближення (РПН) з'являється напруга логічної 1, а на інших його виходах – логічні нулі. На виході цифро–аналогового перетворювача (ЦАП) формується напруга $U_{ЦАП}\approx 0{,}5\cdot U_{BX.MAX}$, що на входах аналогового компаратора порівнюється з вхідною аналоговою напругою U_{BX}. Аналоговий компаратор включає власне аналоговий компаратор (ВАК) на мікросхемі операційного підсилювача (ІМС ОП), схему формування рівнів (СФР), що перетворює різнополярні імпульси в цифровий сигнал, та інвертор.

Якщо вхідна напруга U_{BX} більше напруги, що знімається з виходу ЦАП, то на виході ВАК з'являється від'ємний імпульс. СФР перетворює його в нульовий цифровий сигнал. При цьому з виходу інвертора АК знімається логічна одиниця, що подається на вхід D РПН. При надходженні на вхід C РПН наступного імпульсу від ГТІ зберігається логічна 1 у старшому (N–1)–ому розряді і з'являється 1 у (N–2) розряді. Якщо при першому порівнянні $U_{BX}<U_{ЦАП}$, то з виходу АК знімається лог–й 0. В цьому випадку черговим імпульсом на синхровході вміст старшого (N–1) розряду РПН обнуляється, а в (N–2)–й записується одиниця. Якщо після другого порівняння на виходах двох старших розрядів РПН містяться дві одиниці (при першому порівнянні $U_{BX}>U_{ЦАП}$), то вихідний сигнал ЦАП: $U_{ЦАП}\approx(0{,}5+0{,}25)U_{BX.MAX}$. Якщо після першого порівняння на виході (N–1) розряду буде нуль, а на виході (N–2) розряду – одиниця, то вихідний сигнал ЦАП: $U_{ЦАП}\approx 0{,}25*U_{BX.MAX}$. На компараторі U_{BX} знову порівнюється з цією напругою і т. д. Так встановлюються всі розряди на виході РПН до самого молодшого. Після виконання останнього N_P–го порівняння, де N_P – число розрядів вихідного коду АЦП, цикл формування вихідного коду закінчується. Стан виходів РПН відповідає цифровому еквіваленту вхідної напруги. Якщо, наприклад, $U_{BX} = U_{BX.MAX}$, то комбінація вихідного коду дорівнює 111...11 (всі одиниці). У розглянутому АЦП час перетворення $t_{ПРТ}$ постійний і визначається числом розрядів N_P вихідного двійкового коду і тактовою частотою $f_{ГТІ}=1/T_{ГТІ}$; $t_{ПРТ}\approx N_P*T_{ГТІ}$. Розглянуті АЦП мають досить

високу швидкодію при відносно простій структурі, тому знаходять широке застосування, наприклад, входять до складу більшості сучасних мікроконтролерів.

4.9.2 Модуль АЦП сімейства мікроконтролерів LPC 23XX

4.9.2.1 Загальні особливості модуля АЦП сімейства LPC23XX

Модуль аналого–цифрового перетворювача (АЦП), який входить до складу мікроконтролерів ARM7 сімейства LPC 23XX, являє собою 10–бітний АЦП послідовного наближення з часом перетворення порядку 2,44 мкс (частота вибірок трохи менше 410 тис. вибірок/с) [1, 4, 8]. В LPC23XX, кількість каналів АЦП (рисунок 4.54) залежить від моделі:

- LPC2361/62/64/65/66/67/68 і LPC2387: 6 каналів;
- LPC2377/78 і LPC2388: 8 каналів.

Особливості АЦП у складі моделей мікроконтролерів сімейства LPC23XX:

– режим відключення живлення;

– діапазон вимірювання від 0 до 3.3 В (Vref);

– burst–режим перетворення для одного або декількох входів;

– додатковий запуск перетворення від сигналу на зовнішніх виводах;

– додатковий запуск перетворення від події «Збіг» таймера 0 або таймера 1;

– індивідуальні регістри для кожного A/D каналу для зменшення часу обробки переривання;

– час перетворення порядку 2,44 мкс;

– вхідний мультиплексор на 6 або 8 входів в залежності від моделі.

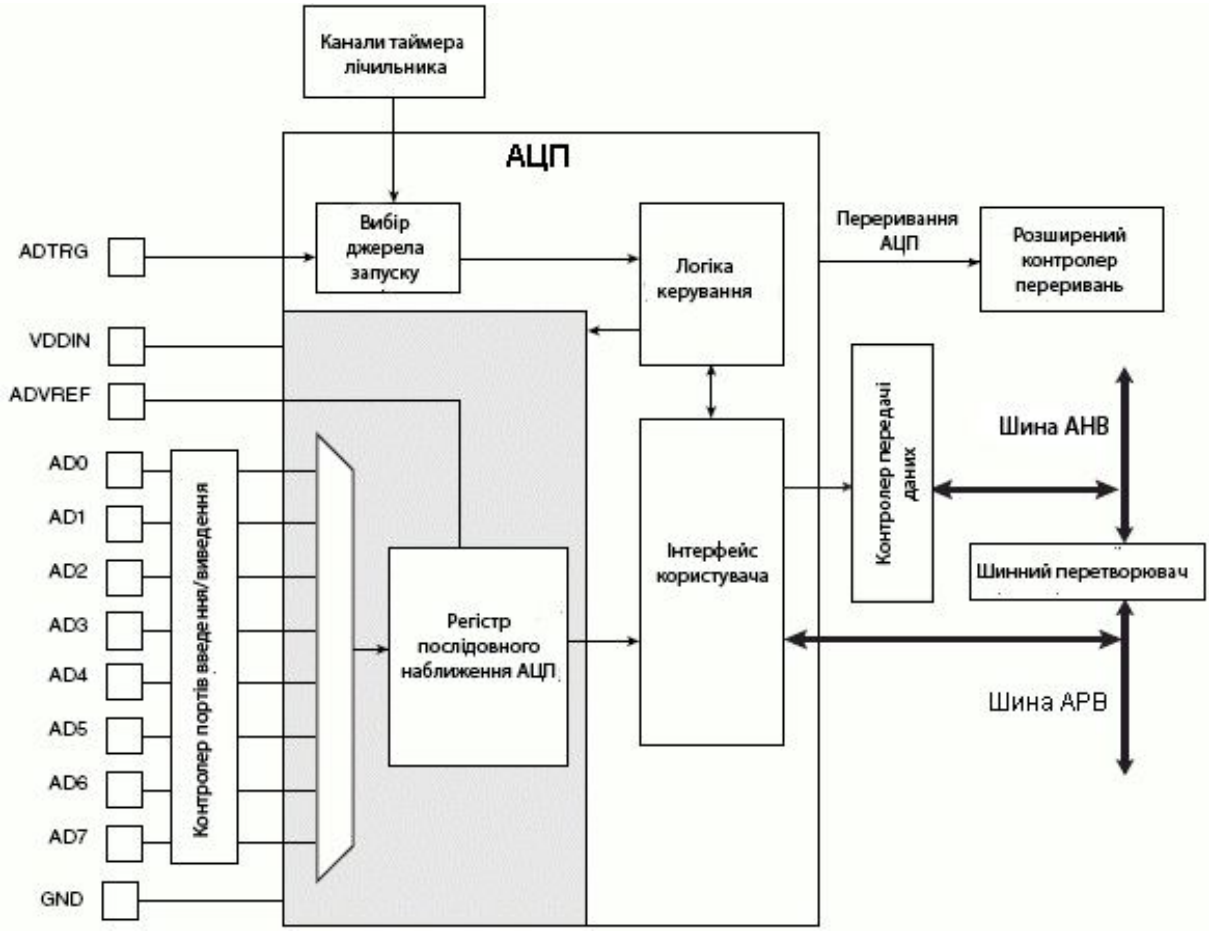


Рисунок 4.54 – Спрощена структура АЦП

4.9.2.2 Функціонування модуля АЦП

Регістр керування АЦП ADCR (рисунок 4.55) визначає конфігурацію модуля і керує запуском процесу перетворення. Насамперед при конфігуруванні АЦП визначається тактова частота модуля. Як і в інших периферійних пристроях, тактовий сигнал модуля АЦП формується з сигналу Pclk. Pclk–це тактовий сигнал шини VPB [4, 8]. Цей сигнал формується після проходження тактового сигналу ЦПП ARM7: Cclk через подільник VPB, який може знижувати частоту Cclk в 1, 2 або 4 рази. Вміст регістра керування подільником VPB можна змінювати з програми в будь–який час. Після скидання:

$$Pclk = \frac{Cclk}{4}. \quad (4.9)$$

Сигнал Pclk слід поділити таким чином, щоб отримати сигнал з частотою не більше 4.5 МГц. Якщо точне значення отримати не можна, то необхідно вибрати значення, найбільш близьке до 4.5 МГц, але не перевищувати його.

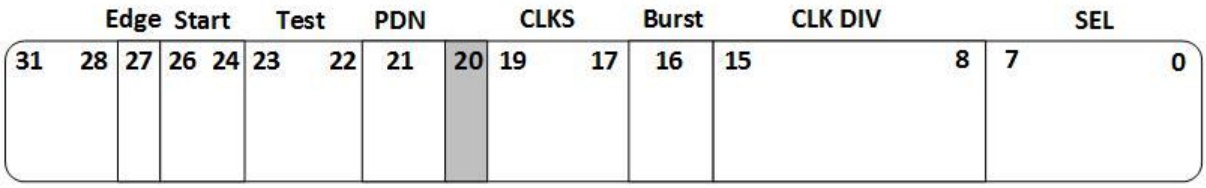


Рисунок 4.55 – Регістр керування АЦП

Сигнал Pclk ділиться на величину, що дорівнює значенню, яке записано в полі CLKDIV плюс один. Таким чином, частота модуля АЦП визначається з виразу:

$$Adclk = \frac{Pclk}{CLKDIV+1}. \quad (4.10)$$

Крім можливості відключення тактового сигналу від модуля АЦП в регістрі відключення периферійних пристроїв (PCONP), для економії живлення ми можемо повністю вимкнути даний модуль. При цьому знизиться енергоспоживання мікроконтролера і рівень завад, які створюються модулем. Після скидання мікроконтролера модуль АЦП вимкнений, тому, крім завдання частоти тактового сигналу, його необхідно включити. Для цього слід встановити біт PDN регістра керування в одиницю. На відміну від інших периферійних пристроїв АЦП може перетворювати сигнал з виводів мікроконтролера навіть тоді, коли вони налаштовані як лінії портів введення/виведення. Однак, підключивши ці виводи безпосередньо до АЦП, можна значно підвищити точність перетворення.

Перед запуском перетворення необхідно задати розрядність (розмірність) результату, записавши відповідне значення в поле CLKS регістра керування. Хоча АЦП є 10–бітовим, розрядність результату можна задати різною, аж до трьох бітів. Взагалі, розрядність дорівнює кількості тактів тактового сигналу, що приходить на одне перетворення, плюс один. Тобто для отримання 10–бітного результату необхідно 11 тактів, а для отримання 3–бітного – 4 такти. Після завдання розрядності можна починати перетворення. Модуль АЦП може працювати в двох режимах: апаратному і програмному. У першому треба вибрати необхідні канали, після чого запустити перетворення. Перетворення буде по черзі виконуватися в усіх обраних каналах до тих пір, поки не буде зупинено. Після завершення кожного перетворення результат можна зчитати, як

з глобального регістра даних АЦП ADDR, так і з регістрів даних ADDR0 ... ADDR7, що відповідає поточному каналу рисунок 4.56.

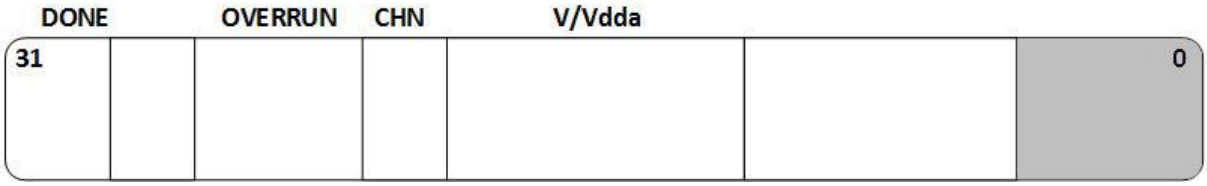


Рисунок 4.56 – Регістр даних АЦП

При завершенні перетворення встановлюється прапорець DONE регістра даних і може генеруватися переривання. Якщо Vref = V_{DDA} результат перетворення зберігається в поле V/ V_{DDA}, у вигляді відношення напруги на вході аналогового каналу до напруги V_{DDA}. Разом з результатом в поле CHN зберігається номер каналу, в якому було виконано перетворення. Якщо ж до моменту завершення перетворення результат попереднього перетворення не був оброблений, то він буде перезаписаний, і встановиться, прапорець OVERRUN. При використанні декількох каналів АЦП значення прапорців DONE і OVERRUN зручніше зчитувати з регістра стану АЦП, що містить ці прапорці для всіх каналів.

Другим режимом роботи модуля АЦП є програмний режим. В даному режимі номер потрібного каналу заноситься в поле SEL регістра керування, а перетворення з цього каналу запускається програмно, шляхом запису в поле START значення 0x01. Потім буде виконано одне перетворення, результат якого зберігається в регістрі даних ADDR. Закінчення перетворення можна визначити за перериванням або за встановленням прапорця DONE регістра ADDR. Крім того, перетворення в програмному режимі може бути запущено при виникненні події «збіг» таймера/лічильника 0, або таймера/лічильника 1, а також при виявленні заданого фронту сигналу на виводах P2.10 (пін76) або P1.27 (пін61) мікроконтролера (таблиця 4.54). Активний фронт визначається станом біта EDGE регістра ADCR.

Таблиця 4.54 – Умови запуску перетворювача

Поле START	Запуск перетворювача		Поле START	Запуск перетворювача
000	Зупинено		100	MAT0.0 (пін 63)
001	Запуск		101	MAT0.1 (пін 64)
010	P2.10 (пін76)		110	MAT1.0 (пін 51)
011	P1.27 (пін61)		111	MAT1.1 (пін 56)

4.9.2.3 Опис керуючих регістрів

Базова адреса АЦП 0xE0034000. A/D перетворювач включає в себе керуючі регістри, які показано в таблиці 4.55.

Таблиця 4.55 – Керуючі регістри АЦП

Назва	Опис	Доступ	Значення скидання	Адреса
AD0CR	Регістр керування. AD0CR регістр для вибору режиму роботи.	R/W	0x0000 0001	0xE003 4000
AD0GDR	Глобальний регістр даних. Містить результат останнього A/D перетворення.	R/W	NA	0xE003 4004
AD0STAT	Регістр статусу. Цей регістр містить DONE і OVERRUN прапорці для всіх A/D каналів, а також A/D прапорець переривання.	RO	0	0xE003 4030
AD0INTEN	Регістр дозволу переривання, містить біти дозволу переривань для кожного A/D каналу.	R/W	0x0000 0001	0xE003 400C
AD0DR0	Регістр даних каналу 0. Цей регістр містить результати останніх перетворень, які завершено на каналі 0.	R/W	NA	0xE003 4010
AD0DR1	Регістр даних каналу 1. Цей регістр містить результати останніх перетворень, які завершено на каналі 1.	R/W	NA	0xE003 4014

Продовження таблиці 4.55

Назва	Опис	Доступ	Значення скидання	Адреса
AD0DR2	Регістр даних каналу 2. Цей регістр містить результати останніх перетворень, які завершено на каналі 2.	R/W	NA	0xE003 4018
AD0DR3	Регістр даних каналу 3. Цей регістр містить результати останніх перетворень, які завершено на каналі 3.	R/W	NA	0xE003 401C
AD0DR4	Регістр даних каналу 4. Цей регістр містить результати останніх перетворень, які завершено на каналі 4.	R/W	NA	0xE003 4020
AD0DR5	Регістр даних каналу 5. Цей регістр містить результати	R/W	NA	0xE003 4024
	останніх перетворень, які завершено на 5 каналі.			
AD0DR6	Регістр даних каналу 6. Цей регістр містить результати останніх перетворень, які завершено на каналі 6.	R/W	NA	0xE003 4028
AD0DR7	Регістр даних каналу 7. Цей регістр містить результати останніх перетворень, які завершено на каналі 7.	R/W	NA	0xE003 402C

4.9.2.3.1 Регістр керування AD0CR

У таблиці 4.56 наведено регістр, що використовуються для керування модулем АЦП. Регістр керування включає біти для вибору каналу перетворення, тактування, режимів та початку перетворення.

4.9.2.3.2 Таблиця 4.56 –Регістр керування AD0CR

Біт	Символ	Знач.	Опис	Скид.
7:0	SEL	–	Вибирає один з AD0.7:0 контактів для перетворення. Біт 0 вибирає пін AD0.0, а біт 7 вибирає пін AD0.7. У програмному режимі тільки один з цих бітів повинен бути встановлений в 1.	0x01
15:8	CLKDIV	–	Частота шини APB (Pclk) ділиться на це значення плюс один, щоб використовуватись, як тактування для A/D перетворювача, який повинен працювати на частоті, що менше або дорівнює 4,5 МГц. Як правило, програмне забезпечення повинне програмувати найменше в цій області, що не перевищує частоту 4,5 МГц, але в деяких випадках (наприклад, високий опір аналогового джерела) використання малої частоти може бути небажаним.	0
16	BURST	0	Перетворення виконується у програмному режимі і вимагає 11 тактів.	0
		1	АЦП виконує перетворення зі швидкістю, яку задано бітами CLKS, скануючи (при необхідності) піни, які обрано бітами SEL. Перше перетворення після початку відповідає найменшому номеру входу з одиницею в полі SEL, а потім переходить до наступного з одиницею у полі SEL. Повторне перетворення може бути припинено скиданням цього біта, але перетворення, що виконується, коли цей біт очищений, буде завершено. Важливо: START–біти мають бути 000 при BURST = 1 або перетворення не почнеться.	
19:17	CLKS		Це поле вибирає кількість тактів для кожного перетворення в пакетному режимі, між 11 тактами (10 бітів) і 4 тактами (3 біти).	000
			Число тактів/розрядність результату	
		000	11 тактів / 10 біт	
		001	10 тактів / 9 біт	
		010	9 тактів / 8 біт	
		011	8 тактів / 7 біт	
		100	7 тактів / 6 біт	
		101	6 тактів / 5 біт	
		110	5 тактів / 4 біти	
		111	4 тактів / 3 біти	

Продовження таблиці 4.56

Біт	Символ	Знач.	Опис	Скид.
20	–	–	Зарезервовано, користувачі програмного забезпечення не повинні писати в зарезервовані біти. Значення із зарезервованого біта не визначене.	NA
21	PDN	1	A / D перетворювач ввімкнено.	0
		0	A / D перетворювач вимкнено.	
23:22	TEST		Зарезервовано, користувачі програмного забеспечення не повинні писати в зарезервовані біти. Значення із зарезервованого біта не визначене.	0
26:24	START		Коли біт BURST біт дорівнює 0 і прапорець PDN встановлено в одиницю:	NA
		000	Немає старту (це значення використовується разом з PDN, який скинуто в 0).	
		001	Початок перетворення.	
		010	Початок перетворення, коли зміна яка обрана бітом 27, відбувається на P2.10.	
		011	Початок перетворення, коли зміна яка обрана бітом 27, відбувається на P1.27.	
		100	Початок перетворення, коли зміна яка обрана бітом 27, відбувається на MAT0.0.	
		101	Початок перетворення, коли зміна яка обрана бітом 27, відбувається на MAT0.1.	
		110	Початок перетворення, коли зміна яка обрана бітом 27, відбувається на MAT1.0.	
		111	Початок перетворення, коли зміна яка обрана бітом, 27 відбувається на MAT1.1.	
27	EDGE		Цей біт використовується тільки тоді, коли поле START поле містить: 010…111. У цих випадках:	0
		1	Початок перетворення за заднім фронтом обраного сигналу.	
		0	Початок перетворення за переднім фронтом обраного сигналу.	
31:28	–	–	Зарезервовано, користувачі програмного забезпе–чення не повинні писати в зарезервовані біти. Значення із зарезервованого біта не визначене.	NA

4.9.2.3.3 Глобальний регістр даних AD0GDR

Глобальний регістр даних містить результат останнього A/D перетворення. Він включає в себе дані, DONE і Overrun (переповнення) прапорці, та номер A/D каналу, до якого відносяться дані.

Таблиця 4.57 – A/D глобальний регістр даних

Біт	Символ	Опис	Скид.
5:0	–	Ці біти завжди читаються як нулі. Вони забезпечують сумісність для майбутніх A/D перетворювачів з більш високою роздільної здатністю.	0
15:6	V/V_{REF}	Коли біт DONE встановлено в 1, це поле містить двійкову послідовність, що представляє напругу на піні Ain, який обрано значенням SEL, яку поділено на напругу на виводі V_{DDA} ($V_{DDA}=V_{REF}$). Нуль в полі означає, що напруга на контактах Ain була меншою, рівною або близькою до VSSA, а 0x3FF показує, що напруга на Ain була близькою, рівною або більшою, ніж V_{DDA}.	X
23:16	–	Ці біти завжди читаються як нулі. Вони можуть бути використані для розширення поля CHN. Дозволяють накопичення послідовних A/D значень без маски AND, до 256 значень без переповнення поля CHN.	0
26:24	CHN	Ці біти містять номер каналу, з якого отримано перетворення.	X
29:27	–	Ці біти завжди читаються як нулі. Вони можуть бути використані для розширення поля CHN. В майбутньому послужать для розширення A/D перетворювача, який	0
		може конвертувати більше каналів.	
30	OVERRUN	Цей біт дорівнює 1 в пакетному режимі, якщо результати одного або декількох перетворень було втрачено і перезаписано до перетворення, яке записує результат в V/VREF біти. Цей біт очищається при читанні з цього регістра.	0
31	DONE	Цей біт встановлюється в 1, коли перетворення завершилося. Він очищається при зчитуванні з регістра і при записі в ADCR. Якщо в ADCR записано значення в той час, як перетворення ще не завершено, то DONE скидається 0 і починається нове перетворення.	0

4.9.2.3.4 Регістр статусу AD0STAT

Регістр статусу дозволяє перевірити стан усіх A/D каналів одночасно. DONE і OVERRUN прапорці, які присутні в регістрах AD0DRn для кожного A/D каналу, також відображається в AD0STAT. Також в AD0STAT знаходиться прапорець переривання (логічне АБО всіх DONE прапорців).

Таблиця 4.58 – A/D регістр статусу

Біт	Символ	Опис	Скид.
7:0	Done 7:0	Ці біти відображають DONE–прапорці стану для кожного A/D каналу, які з'являються в регістрах даних.	0
15:8	Overrun7:0	Ці біти відображають OVERRUN–прапорці стану для кожного A/D каналу, які з'являється в регістрах даних. Читання ADSTAT дозволяє перевірити стан усіх A/D каналів одночасно	0
16	ADINT	Цей біт відображає наявність A/D переривання.	0
31:17	–	Не використовується. Завжди скинутий в 0.	0

4.9.2.3.5 Регістр дозволу переривання AD0INTEN

Цей регістр дозволяє контролювати, які переривання A/D канали генеруватимуть при завершенні перетворення. Наприклад, це може бути потрібно при використовуванні деяких A/D каналів для моніторингу датчиків при безперервному виконанні перетворень на них. Останні результати зчитуються прикладною програмою в будь–який момент, коли вони необхідні. У цьому випадку не бажано використовувати переривання в кінці кожного перетворення для деяких A/D каналів.

Таблиця 4.59 – A/D регістр дозволу переривання

Біт	Символ	Опис	Скид.
7:0	ADINTEN 7:0	Ці біти дозволяють керувати A/D каналами які генеруватимуть переривання за завершенням перетворення. Коли біт встановлено в 1, завершення перетворення на A/D каналі буде генерувати переривання.	0x00
8	ADGINTEN	Якщо біт встановлено в 1 то при встановленні прапорця DONE регістра AD0GDR генерується переривання. Коли 0, тільки окремі з A/D каналів, які дозволені за ADINTEN 7:0, можуть генерувати переривання.	1
31:9	–	Не використовується. Завжди встановлені в 0.	0

4.9.2.3.6 Регістри даних AD0DR0 …AD0DR7

Регістри даних містять результат, коли A/D перетворення завершено, а також прапорці, які вказують, коли перетворення завершено, та якщо перетворення відбулося з переповненням.

Таблиця 4.60 – A/D регістри даних

Біт	Символ	Опис	Скид.
5:0	–	Не використовується, завжди 0.	0
15:6	V/V_{REF}	Коли DONE 1, це поле містить двійкову послідовність, що представляє напругу на контакті Ain, яку поділену на напругу на виводі Vref. Нуль в полі означає, що напруга на контактних Ain була меншою, рівною або близькою до 0, в той час як 0x3FF показує, що напруга на Ain була близькою, рівною або більшою, ніж на Vref.	NA
29:16	–	Не використовується. Завжди встановлені в 0.	0
30	OVERRUN	Цей біт дорівнює 1 в пакетному режимі, якщо результати одного або декількох перетворень втрачено і перезаписано до завершення перетворення, яке записує результат в V/V_{REF} біти. Цей біт очищається при читанні регістра даних.	0
31	DONE	Цей біт встановлюється в 1, коли A/D перетворення завершується. Він очищується, коли цей регістр читається.	0

4.9.2.4 Результат перетворення АЦП

Основна синхронізація для A/D перетворювача забезпечується тактовим сигналом шини VPB Pclk. В кожен перетворювач входить програмований подільник для масштабування цієї частоти на 4,5 МГц (максимум), необхідних для процесу послідовного наближення. Точне перетворення вимагає 11 тактів цієї частоти. Характеристика перетворення АЦП зображення на рисунку 4.57: (1) – реальна; (2) – ідеальна, V_{DDA}=Vref.

Результат перетворення визначається виразом: $ADC = \frac{1024U_{IN}}{U_{REF}}$,

де U_{IN} – значення вхідної напруги в мілівольтах, а U_{REF} – величина опорної напруги. На рисунку 4.57 представлено функцію перетворення АЦП в. Код

0x000 відповідає рівню аналогової землі, а 0x3FF – рівню напруги ДОН минус 1 крок квантування за напругою.

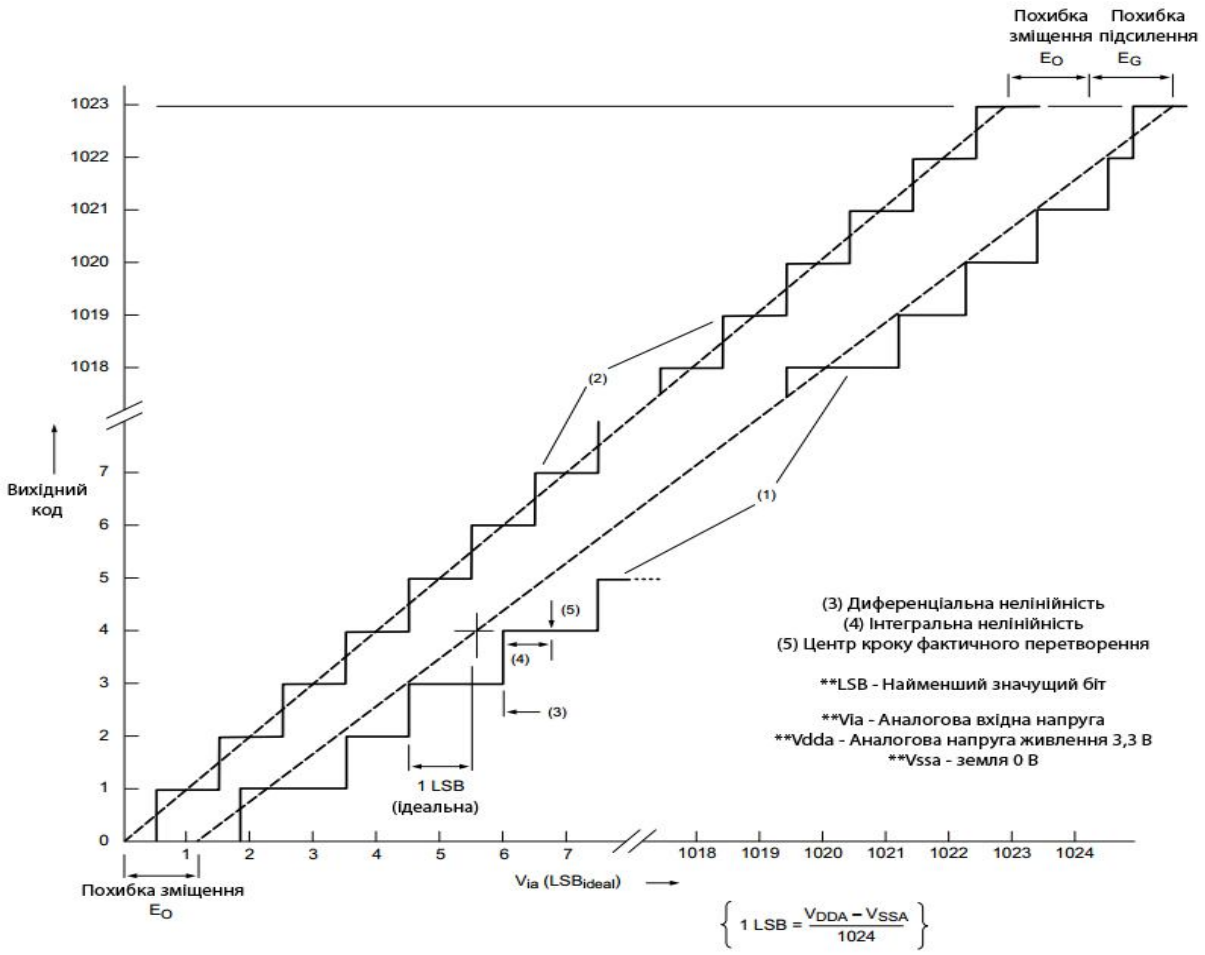


Рисунок 4.57 – Характеристика перетворення АЦП

Таблиця 4.61 відображає зв'язок між вхідним сигналом й вихідними кодами.

Приклад: Нехай напруга на одному з входів 1000 мВ, напруга ДОН дорівнює 2,56 В, тоді:

КодАЦП = 1024 * 1000 / 2560 = 400 = 0x190.

Таблиця 4.61 – Зв'язок між вхідним сигналом та вихідними кодами

$U_{АЦПn}$*	Зчитаний код	Відповідне десяткове значення
$U_{АЦПm} + U_{ДОН}$	0x3FF	1023
$U_{АЦПm} + 0.999\ U_{ДОН}$	0x3FF	1023
$U_{АЦПm} + 0.998\ U_{ДОН}$	0x3FE	1022
…	...	...
$U_{АЦПm} + 0.001\ U_{ДОН}$	0x001	1
$U_{АЦПm}$	0x000	0

$U_{АЦПm}$ – вхідна напруга, яка дорівнює нулю, $U_{АЦПn}$ – поточне значення вхідної напруги.

4.9.2.5 Опис контактів

В таблиці 4.62 подано короткий опис кожного з контактів АЦП.

Таблиця 4.62 – Опис контактів АЦП

Контакти	**Тип**	**Опис**
AD0[5:0] або AD0[7:0]	Вхід	**Аналогові входи.** A/D перетворювач може виміряти напругу на будь–якому з цих входів.
V_{REF}	Опорна напруга	**Опорна напруга.** Цей контакт забезпечує рівень опорної напруги на A/D перетворювачі.
V_{DDA} V_{SSA}	Живлення	**Аналогове живлення і заземлення**. Вони повинні бути номінально ті ж, що і напруга, V_{DD} (3.3 В) і V_{SSA} відповідно, але повинні бути ізольовані, щоб мінімізувати шуми і помилки. *Зауваження:* Якщо АЦП не використовується, V_{DDA} і V_{REF} контакти мають бути підключені до джерела живлення, і пін V_{SSA} повинен бути заземлений. Ці контакти не повинні бути залишені плаваючими.

4.9.2.6 Інтерфейс для виводів АЦП

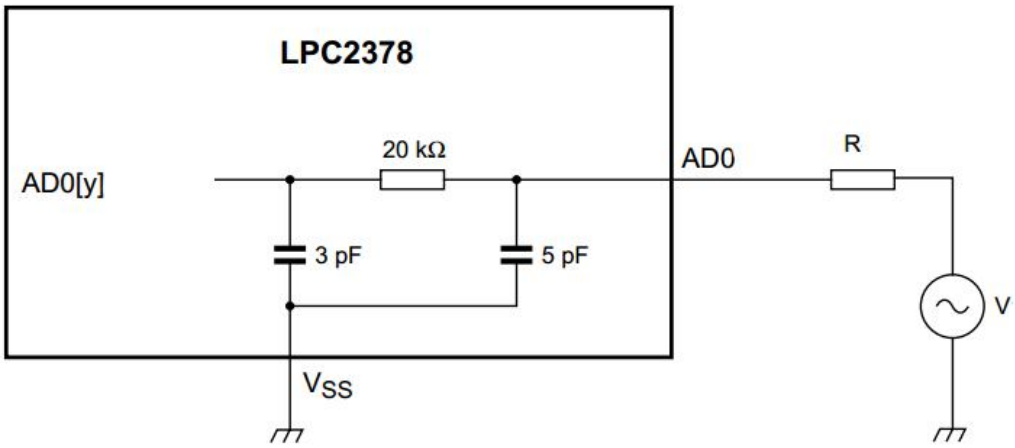


Рисунок 4.58 – Інтерфейс для виводів АЦП

4.9.2.7 Програмування АЦП

Для більш точного розуміння програмування АЦП мікроконтролерів ARM7 сімейства 23XX, наведемо приклади програм в різних режимах роботи.

Даний фрагмент демонструє використання модуля АЦП в апаратному режимі:

```
  int main (void)
  {
APBDIV = 0x00000002;                //Pclk = 30 МГц
IODIR1 = 0x00FF0000;                //P1.16...P1.P23 - виходи
ADCR   = 0x00270607;                //Ініціалізуємо АЦП: 10 біт,
                                    AIN0, 3МГц


VICVectCntl0 = 0x00000032;          //Вибираємо слот для даного
                                    переривання
VICVectAddr0 = (unsigned)AD_ISR;    //Передаємо адресу обробника
                                    в модуль VIC
VICIntEnable = 0x00040000;          //Дозволяємо переривання

while(1)
{
;
}
  }

  void AD_ISR(void)
  {
unsigned val, chan;
static unsigned result[4];
val = ADDR;                         //Зчитуємо регістр даних
val = ((val >> 6) & 0x03FF)<<16;    //Виділяємо результат
chan = ((ADCR >> 0x18) & 0x07);
result[chan] = val;
```

```
}
```

Найпростіший варіант використання модуля АЦП в програмному режимі:

```
APBDIV = 0x00000002;            //Pclk = 30 МГц
IODIR1 = 0x00FF0000;            //P1.16...P1.P23 - виходи
ADCR   = 0x00270601;            //Ініціалізуємо АЦП: 10 біт,
                          AIN0, 3МГц

ADCR   |= 0x01000000;           //Запускаємо перетворення
while(1)
{
     do
     {
     val = ADDR;                // Зчитуємо регістр даних АЦП
}-
     while((val & 0x80000000) == 0); //Очікуємо завершення
                                 перетворення
}
```

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Опишіть призначення та принцип роботи АЦП.
2) Як розраховується абсолютна та відносна похибки АЦП?
3) Як робиться вибір величини кроку квантування за часом АЦП?
4) Для чого в модулях АЦП використовується пристрій вибірки –зберігання?
5) Поясніть роботу АЦП послідовного наближення.
6) Поясніть особливості АЦП у складі моделей мікроконтролерів сімейства LPC23XX:
7) Як формується тактовий сигнал модуля АЦП?
8) Які дії треба виконати у програмі керування АЦП перед його запуском?
9) Як можна змінювати розрядність результату перетворення?
10) Що відбувається при завершенні перетворення?
11) Де та в якому вигляді зберігається результат перетворення?
12) Поясніть режими функціонування АЦП.
13) Назвіть та поясніть регістри керування АЦП.
14) Опишіть біти регістра керування модулем.
15) Опишіть біти глобального регістра даних.
16) Опишіть біти регістра статусу.
17) Опишіть біти регістра дозволу переривання.
18) Опишіть біти регістрів даних.
19) Яким виразом визначається результат перетворення АЦП?
20) Наведіть та поясніть характеристику перетворення АЦП.
21) Назвіть основні параметри АЦП.
22) Назвіть та поясніть способи запуску АЦП.
23) Поясніть як саме відбувається перетворення вхідного сигналу у двійковий код.
24) Яку роль виконує мультиплексор в схемі АЦП?
25) Наведіть короткий опис кожного з контактів АЦП.

4.10 Модуль цифро–аналогового перетворювача(ЦАП)

4.10.1 Загальна характеристика

Нижче буде розглянуто принцип роботи та розрахунку ЦАП на основі резистивної матриці R–2R з підсумовуванням напруг, який використовується в модулі ЦАП мікроконтролерів сімействах Cygnal. ЦАП з підсумовуванням напруг використовує режим роботи підсумовуючого елемента, близький до холостого ходу (операційний підсилювач підсумовує напруги, рисунок 4.59).

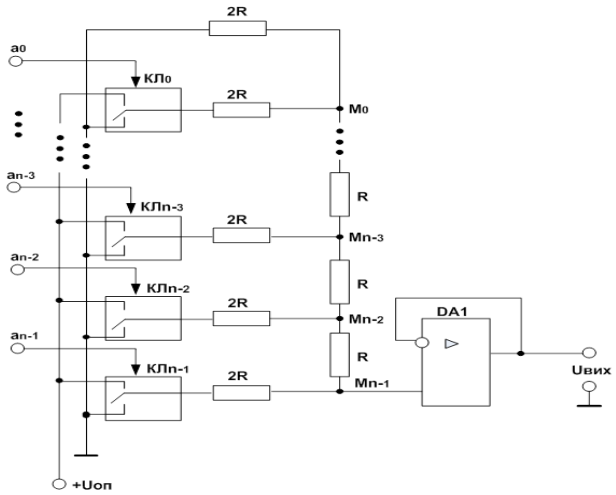


Рисунок 4.59 – n–розрядний ЦАП з матрицею R–2R та з підсумовуванням напруг

ЦАП, з підсумовуванням напруг, використовує зворотне включення входу і виходу матриці R–2R. На входи $a_0, a_1, a_2 ... a_{n-1}$ надходять цифрові сигнали, які відповідають значенню i–го розряду вхідного двійкового коду (i=0,1,…, n–1). Якщо на вході i–го розряду присутня логічна одиниця, то відповідний ключ $Кл_i$ переключається у верхнє положення та опорна напруга $U_{ОП}$ через резистори матриці R–2R з визначеним коефіцієнтом ділення подається на вхід операційного підсилювача (ОП) DA1, який не інвертує, де відбувається підсумовування напруг.

Якщо на вхід i–го розряду надходить логічний нуль, то ключ переключається в нижнє положення, і дана гілка матриці R–2R підключається до спільної шини.

Оскільки матриця резисторів є лінійним ланцюгом, її роботу можна проаналізувати методом суперпозиції, тобто внесок у вихідну напругу від кожного

джерела (розряду) розрахувати незалежно один від одного. Внески від кожного розряду підсумовуються на вході ОП, який не інвертує, і на виході отримуємо результат у вигляді напруги.

4.10.2 Розрахунок цифро–аналогового перетворювача на матриці R–2R з підсумовуванням напруг

Розглянемо роботу ЦАП, якщо в старшому розряді вхідного ДК присутня логічна одиниця, а в інших розрядах – логічні нулі. Отже, ключ $Кл_{n-1}$ знаходиться у верхньому положенні і підключає гілку резистивної матриці (РМ) з резистором 2R до джерела опорної напруги $U_{ОП}$. Інші ключі знаходяться в нижньому положенні і підключають інші гілки РМ (резистори 2R) до спільної шини. Еквівалентну схему ЦАП для цього випадку наведено на рисунку 4.60, а.

Очевидно, що еквівалентний опір РМ вище вузла $М_{n-1}$ дорівнює 2R. Так як вхідний опір ОП великий і останній працює в режимі, близькому до холостого ходу, то струм, який створюється джерелом $U_{ОП}$, протікає через два однакових резистори 2R, які утворюють дільник напруги $U_{ОП}$. У цьому випадку напруга на виході дільника визначається з виразу:

$$U_{діл} = U_{Н} = \frac{U_{ОП} \cdot 2R}{2R + 2R} = \frac{U_{ОП}}{2}. \qquad (4.11)$$

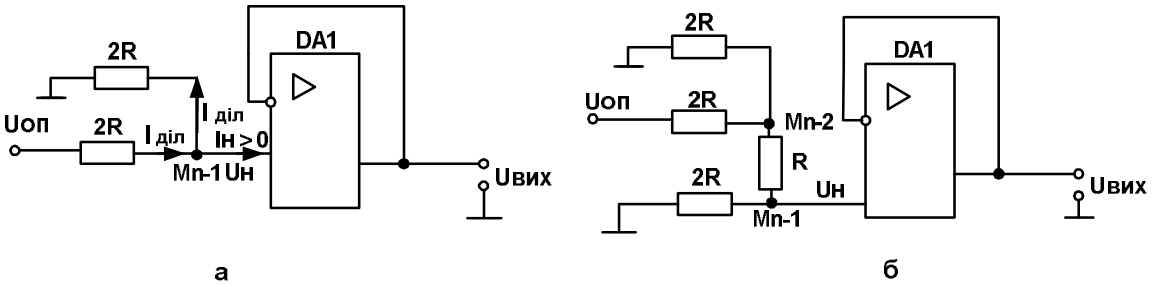


Рисунок 4.60 – Еквівалентні схеми ЦАП: а – при перетворенні коду 100…0В; б – при перетворенні коду 010…0В

Розглянемо роботу ЦАП, якщо на вхід схеми надходить комбінація ДК: 010…0В. У цьому випадку ключ $Кл_{n-2}$ увімкнений у верхнє положення, а інші

ключі – у нижнє. Еквівалентна схема ЦАП прийме вигляд, який представлено на рисунку 4.60, б.

Розглядаючи резистори R і 2R, які розташовані нижче вузла M_{n-2}, як включені послідовно $(R_{\text{ВХ.DA1}} \to \infty)$, замінюємо їх еквівалентним опором:

$$\mathrm{R} + 2\mathrm{R} = 3\mathrm{R}. \tag{4.12}$$

Тоді напруга в точці M_{n-2} визначається виразом:

$$U_{M_{n-2}} = \frac{U_{\text{оп}} \cdot 2\mathrm{R} \| 3\mathrm{R}}{2\mathrm{R} + 2\mathrm{R} \| 3\mathrm{R}} = \frac{U_{\text{оп}} \cdot 6/5 \cdot R}{2\mathrm{R} + 6/5 \cdot R} = \frac{U_{\text{оп}} \cdot 3}{8}. \tag{4.13}$$

Знаючи напругу в точці M_{n-2}, можна визначити сигнал у вузлі M_{n-1}

$$U_{M_{n-1}} = U_{\text{н}} = \frac{U_{M_{n-2}} \cdot 2\mathrm{R}}{\mathrm{R} + 2\mathrm{R}} = \frac{U_{\text{оп}}}{4}. \tag{4.14}$$

Аналогічним чином можна довести, що при подачі на вхід ЦАП ДК, який дорівнює 001…0В, напруга на вході ОП, який не інвертує, буде дорівнювати:

$$U_{\text{Н}} = \frac{U_{\text{ОП}}}{8}. \tag{4.15}$$

І, нарешті, при надходженні коду: 00...01 В напруга

$$U_{\text{Н}} = \frac{U_{\text{ОП}}}{2^n}. \tag{4.16}$$

Оскільки коефіцієнт передачі розглянутого підсумовуючого операційного підсилювача $K_{\text{U.IMC ОП}} = 1$, то вираз для визначення сумарної вихідної напруги від дії одиниць у всіх розрядах вхідного ДК прийме вигляд:

$$U_{\text{ВИХ max}} = U_{\text{ОП}}\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots + \frac{1}{2^n}\right) = \frac{\mathrm{U}_{\text{ОП}}}{2^n}\sum_{i=0}^{n-1} 2^i. \tag{4.17}$$

Якщо позначити значення і–х розрядів вхідного ДК як a_i, де a_i дорівнює 0 чи 1, то останній вираз перетвориться до вигляду:

$$U_{\text{ВИХ}} = \frac{\mathrm{U}_{\text{ОП}}}{2^n}\sum_{i=0}^{n-1} a_i \cdot 2^i. \tag{4.18}$$

Співмножник $\sum_{i=0}^{n-1} a_i \cdot 2^i$ є десятковим еквівалентом вхідного двійкового коду (представляє десяткове значення вхідного цифрового коду).

Розглянутий перетворювач називають таким, що помножує, тому що вихідна напруга пропорційна добутку значення опорного сигналу $U_{ОП}$ на значення вхідного цифрового коду.

Коефіцієнт передачі, тобто розрахункове збільшення вихідної напруги при зміні вхідного коду на одиницю молодшого розряду (ціна молодшого значущого розряду (МЗР)) складає:

$$K_{ЦАП} = \frac{U_{ОП}}{2^n} \left[\frac{В}{МЗР}\right]. \qquad (4.19)$$

Для мікроконтролера LPC2378 вирази (4.17), (4.18) та (4.19) мають вигляд :

$$U_{вих.max} = \frac{U_{оп}}{2^{12}} \sum_{i=0}^{11} 2^i = \frac{U_{оп}}{2^{12}} \cdot 1023\,, \qquad (4.20)$$

$$U_{вих} = \frac{U_{оп}}{2^{12}} \sum_{i=0}^{11} a_i 2^i\,, \qquad (4.21)$$

$$K_{ЦАП} = \frac{U_{оп}}{2^{12}} \left[\frac{В}{МЗР}\right]. \qquad (4.22)$$

4.10.3 Модуль ЦАП мікроконтролера LPC2378

4.10.3.1 Загальна характеристика

У мікроконтролері моделі LPC2378 реалізовано 10–бітний ЦАП. Це дуже простий у використанні модуль, що керується лише одним регістром [1, 4, 8].

Модуль ЦАП активується записом 10b в біти 20:21 регістра PINSEL1 і перемиканням виводу P0.26 з порту введення/виведення на функцію A_{OUT}. Майте на увазі, що АЦП також використовує цей вивід.

Після дозволу роботи модуля, перетворення може бути почате записом десятирозрядного двійкового числа в поле VALUE регістра керування DACR (рисунок 4.61).

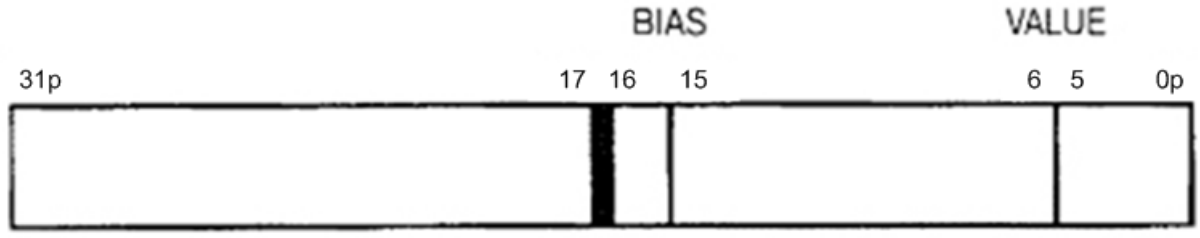


Рисунок 4.61 – Регістр керування ЦАП

Час перетворення залежить від значення, що знаходиться в біті BIAS цього регістра. Якщо цей біт встановлено, то перетворення займе 2.5 мкс. Якщо біт BIAS скинуто, то час перетворення зменшиться до 1,0 мкс. При цьому сумарний час перетворення також залежить від зовнішнього імпедансу.

Вбудований модуль ЦАП мікроконтролерів сімейства LPC23xx має такі особливості:

- 10 бітний цифро–аналоговий перетворювач;
- архітектура комутованої резистивної матриці;
- буферизований вихід;
- режим енергозбереження;
- можливість вибору швидкості перетворення та споживаної потужності.

Зауваження: коли ЦАП не використовується, виводи V_{REF} (джерело опорного напруги) та V_{DDA} повинні бути підключені до живлення, а вивід V_{SSA} повинен бути заземлений. Ці виводи не повинні бути залишені плаваючими.

Режим енергозбереження може бути активований записом значення 1b в регістр PCON.

4.10.3.2 Опис виводів модуля ЦАП

Опис виводів модуля ЦАП мікроконтролерів LPC23xx наведено в таблиці 4.63.

Таблиця 4.63 – Опис виводів модуля ЦАП

Назва виводу	**Тип**	**Опис**
A_{OUT}	Вихід	**Аналоговий вихід.** Після того, як в регістр DACR записано нове значення та закінчився обраний час перетворення, напруга на цьому виводі визначається за формулою (4.21)
V_{REF}	Лінія опорної напруги	**Опорна напруга.** Цей вивід вказує на рівень опорної напруги для ЦАП
V_{DDA}, V_{SSA}	Лінії живлення	**Аналогова напруга живлення і аналогова земля.** Хоча напруги на цих виводах номінально дорівнюють, відповідно, напругам V_{DDA}, V_{SSA}, але вони повинні бути гальванічно–розв'язані від останніх в цілях мінімізації шумів.

4.10.3.3 Опис керуючих регістрів ЦАП

Модуль ЦАП мікроконтролерів сімейства LPC23xx обслуговується тільки одним спеціальним керуючим регістром – DACR (регістр D/A–перетворювача, адреса 0xE006 C000).

Опис окремих бітів регістра DACR наведено в таблиці 4.64. Цей регістр, доступний для читання і запису, містить цифрове значення, яке буде перетворено в аналогову напругу, а також біт, який задає час перетворення ЦАП, в залежності від споживаної потужності. Біти 5:0 цього регістра зарезервовані для модулів ЦАП з більш високою роздільною здатністю.

Таблиця 4.64 – Опис окремих бітів регістра D/A– перетворювача (DACR – 0xE006 C000)

Біти регістра DACR	Назва	Опис	Значення після скидання
5:0	Зарезервовані	Програмне забезпечення користувача не повинно проводити запис в зарезервовані біти. Читання зарезервованого біта повертає невизначене значення	0
15:6	VALUE	Після закінчення обраного часу перетворення після того, як в регістр DACR записано нове значення, напруга на аналоговому виході A_{OUT} визначається за формулою: $(VALUE /1024)*V_{REF}$	0
16	BIAS	Якщо цей біт скинуто в 0, то час перетворення ЦАП складає максимум 1 мкс, а максимальний струм споживання ЦАП складає 700 мкА. Якщо цей біт встановлено в 1, то час перетворення ЦАП складає 2.5 мкс, а максимальний струм споживання становить 350 мкА	0
31:17	Зарезервовані	Програмне забезпечення користувача не повинно проводити запис в зарезервовані біти. Читання зарезервованого біта повертає невизначене значення	0

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Опишіть призначення модуля цифро–аналогового перетворювача.
2) Опишіть принцип роботи ЦАП на основі резистивної матриці R–2R з підсумовуванням напруг.
3) Які сигнали поступають на вхід схеми ЦАП?
4) Чим визначається положення електронних ключів в схемі ЦАП на основі резистивної матриці R–2R з підсумовуванням напруг?
5) Яким методом можна проаналізувати роботу ЦАП?
6) Як розраховується вихідна напруга та максимальна вихідна напруга схеми ЦАП на основі резистивної матриці R–2R з підсумовуванням напруг?
7) Що відображає коефіцієнт передачі схеми ЦАП?
8) Чому дорівнює коефіцієнт передачі схеми ЦАП на основі резистивної матриці R–2R з підсумовуванням напруг?
9) Чому ЦАП на основі резистивної матриці R–2R з підсумовуванням напруг називають таким, що помножує?
10) Яку функцію виконує операційний підсилювач у схемі ЦАП?
11) Чому дорівнює коефіцієнт підсилення операційного підсилювача?
12) Дайте загальну характеристику модуля ЦАП мікроконтролера LPC2378.
13) Як активується модуль ЦАП мікроконтролера LPC2378?
14) Як може бути почате перетворення після дозволу роботи модуля?
15) Від чого залежить час перетворення модуля ЦАП?
16) Як може бути активований режим енергозбереження модуля ЦАП?
17) Дайте опис виводів модуля ЦАП.
18) За якою формулою розраховується напруга на виході модуля?
19) Дайте опис окремих бітів регістра керування DACR.

4.11 Модуль годинника реального часу

4.11.1 Загальна характеристика

Годинники реального часу (Real Time Clock – RTC) широко використовуються в промисловості, сільському господарстві, спеціальних областях техніки, побуті, тощо.

Для реалізації RTC часто використовуються мікроконтролери, які містять таймери, які окрім роботи в основних стандартних режимах, спроможні працювати у асинхронному режимі [1, 4, 8]. Задавачем частоти для цих таймерів є, як правило, кварцовий резонатор частотою 32768 Гц, який підключається до окремих виводів мікроконтролера.

Модуль годинника реального часу являє собою годинник с календарем, розрахований на роботу до 2099 року (з урахуванням високосних років). Модуль RTC здатний працювати від системного тактового сигналу Pclk, або від зовнішнього резонатора з частотою 32,768 кГц. Окрім того, до складу модуля RTC входить НОЗП об’ємом 2 Кб (так званий батарейний ОЗП). Модуль RTC та батарейний ОЗП складають окрему групу споживачів, які під'єднано до окремої шини живлення. Так, за наявності на вході Vbat мікроконтролера напруги 3.3 В, ці вузли продовжують роботу навіть при переході мікроконтролера в режим Power Down. Як модуль RTC, так і батарейний ОЗП мають надзвичайно мале споживання та здатні працювати від батарей. Таким чином, на базі модуля RTC можна побудувати «вічний» годинник з календарем. А навіть якщо в цьому не має потреби, модуль RTC можна використовувати як генератор, що задає, та для періодичної генерації переривань, без необхідності підключення окремого зовнішнього генератора.

Окрім підтримки функції годинника та календаря, модуль RTC може мати набір сигнальних (alarm) регістрів, які можна використовувати для сповіщення конкретної дати та часу, або про наявність в лічильному регістрі конкретного значення (рисунок 4.62).

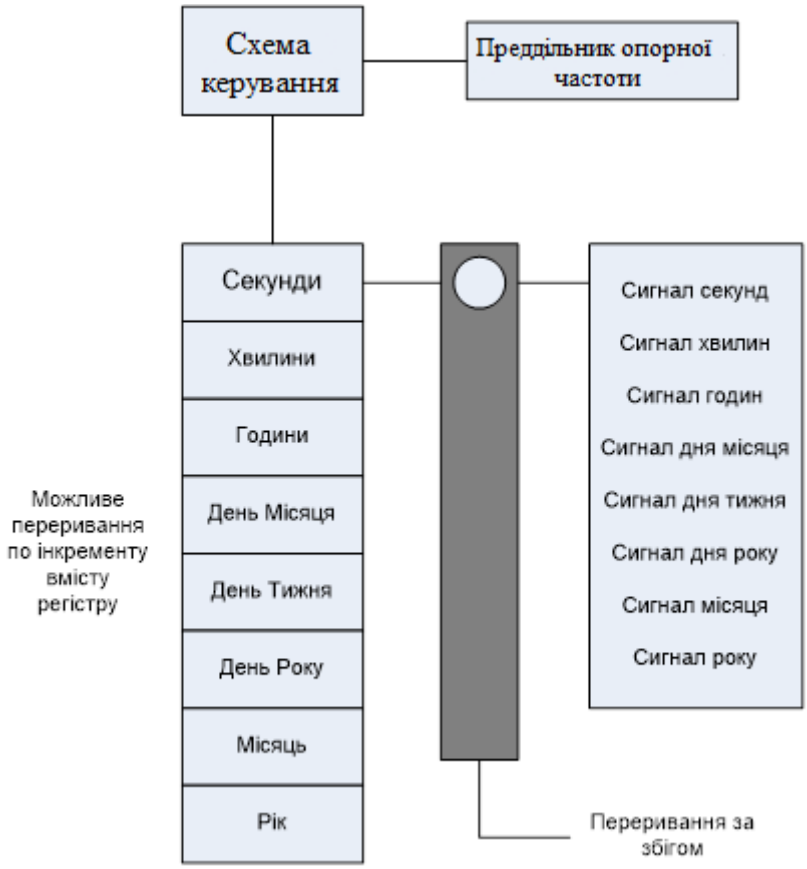


Рисунок 4.62 – Модуль RTC

Спрощену структуру модуля RTC наведено на рисунку 4.63.

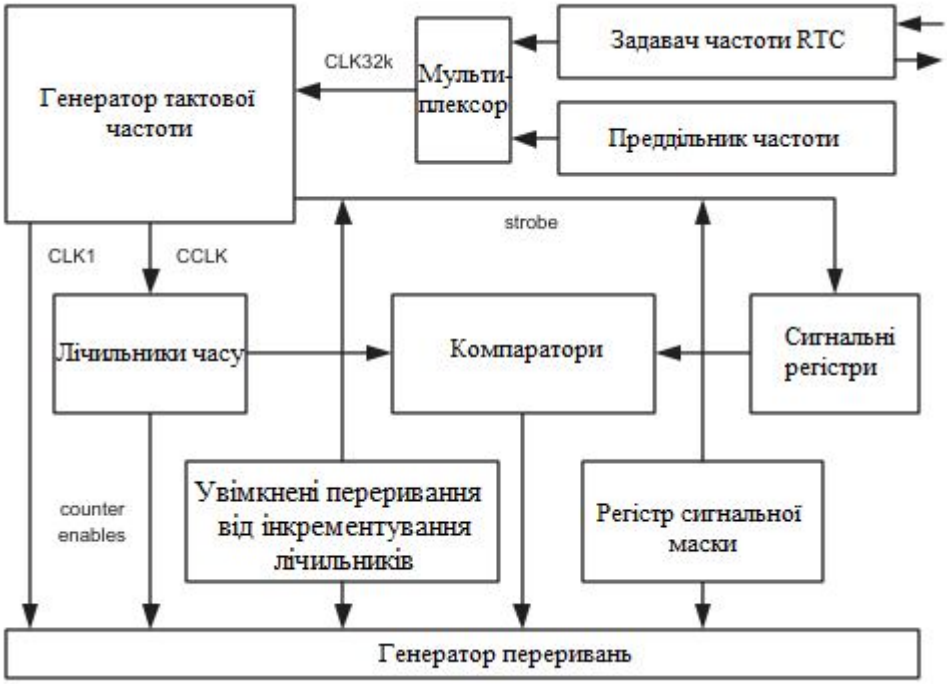


Рисунок 4.63 – Спрощена структура модуля RTC

В якості джерела опорної частоти модуль RTC може використовувати або зовнішній резонатор з частотою 32,768 кГц, або системний тактовий сигнал Pclk,

який також розрахований на стандартну частоту резонатора 32,768 кГц. Найбільш простий спосіб отримати такий сигнал – підключенням зовнішнього «годинникового» резонатора до виводів RTCX1 та RTCX2 мікроконтролера. Щоб використовувати цей резонатор в якості джерела опорної частоти, необхідно встановити біт CLKSRC в регістрі керування модулем. Якщо ж вічний календар не потрібен, модуль RTC можна тактувати сигналом Pclk, скинувши біт CLKSRC. Для того, щоб забезпечити формування сигналу з частотою 32,768 кГц, сигнал Pclk проходить крізь програмований подільник опорної частоти, що дозволяє точно ділити сигнал Pclk будь–якої частоти для формування вихідного сигналу з потрібною частотою (рисунок 4.64).

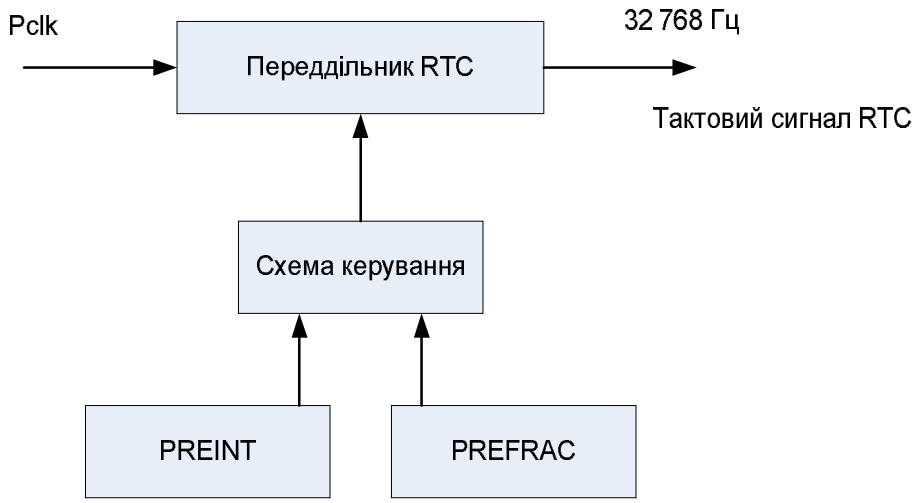


Рисунок 4.64 – Формування тактового сигналу RTC

Щоб забезпечити можливість формування тактового сигналу RTC необхідної частоти з будь–якого сигналу Pclk, преддільник модуля має більш складну схему, ніж преддільники таймерів загального призначення. Робота преддільника програмується двома регістрами: PREINT та PREFRAC. Як можна здогадатися з їх назв, в цих регістрах зберігається ціла та дробова частини коефіцієнта ділення. Значення, що записуються в регістри, обчислюються за формулами [1, 4, 8]:

$$PREINT = (int)\frac{P_{CLK}}{32768} - 1, \quad (4.23)$$

$$PREFRAC = P_{CLK} - 32768 \times (PREINT + 1). \quad (4.24)$$

Таким чином для сигналу Pclk з частотою 30 МГц:

$$PREINT = (int)\frac{30000000}{32768} - 1 = 914.$$

Тоді

$$PREFRAC = 30000000 - 32768 \cdot (914 + 1) = 17280.$$

Після завантаження цих значень до регістрів преддільника RTC модуль буде готовий до роботи. Для запуску лічильників модуля потрібно лише дозволити формування тактового сигналу в регістрі керування CCR.

PREINT = 0x00000392;

PREFRAC = 0x00004380;

CCR = 0x00000001.

Модуль має 8 лічильних регістрів (по одному на кожний часовий параметр), значення яких можна прочитати в будь–який момент часу. Також передбачено три групових регістри, що відображають ті ж самі значення, що дозволяє отримати всю інформацію про даний момент часу всього за три операції зчитування (рисунок 4.65).

Рисунок 4.65 – Групові регістри RTC

Групові регістри дозволяють отримати всю інформацію про даний момент часу (від годин до календаря), яка об’єднана у три слова даних.

RTC також може формувати сповіщення у вигляді переривань. При цьому доступні два механізми переривань. Можна запрограмувати модуль RTC таким чином, щоб переривання генерувалося за інкрементом одного з лічильних регістрів. Тобто, можливо генерувати переривання раз в секунду, при оновленні лічильника секунд, або раз на рік, при інкрементуванні лічильника років. Регістр керування перериванням за інкрементом (CIIR) дозволяє переривання такого типу для кожного з восьми лічильних регістрів.

Другим джерелом переривань є сигнальні регістри. Кожен лічильний регістр має відповідний сигнальний регістр. Якщо один з сигнальних регістрів

дозволено, то він порівнюється з лічильним регістром. Коли значення всіх дозволених сигнальних регістрів співпадуть з вмістом лічильних регістрів, буде згенеровано переривання. Тобто, можна встановити «будильник» на будь–який час до 2099 року з точністю до однієї секунди. Які з сигнальних регістрів будуть використані для порівняння, визначає регістр маски (ARM). Так як переривання від модуля RTC може бути згенеровано як за інкрементом, так і за збігом, в оброблювачі переривань потрібно розрізняти ці події. Для цього передбачено регістр типу переривання (ILR), що містить у собі три прапорці, кожен з яких встановлюється при настанні відповідної події. Для скидання переривання ці прапорці потрібно обнуляти.

4.11.2 Регістри керування модулем RTC

В таблиці 4.65 наведено карту регістрів керування модулем RTC.

Таблиця 4.65 – Карта регістрів модуля RTC

Назва	**Розмір [біт]**	**Опис**	**Доступ**	**Адреса**
Змішана група регістрів				
ILR	3	Регістр місцеположення переривань	R/W	0xE0024000
CTC	15	Лічильник імпульсів тактової частоти	RO	0xE0024004
CCR	4	Регістр керування тактуванням	R/W	0xE0024008
CIIR	8	Регістр переривань від інкрементування лічильників	R/W	0xE002400C
AMR	8	Регістр сигнальної маски	R/W	0xE0024010
CTIME0	32	Регістр консолідованого часу 0	RO	0xE0024014
CTIME1	32	Регістр консолідованого часу 1	RO	0xE0024018
CTIME2	32	Регістр консолідованого часу 2	RO	0xE002401C
CISS	8	Регістр вибору маски інкрементування лічильників	R/W	0xE0024040
Група лічильників часу				
SEC	6	Регістр секунд	R/W	0xE0024020
MIN	6	Регістр хвилин	R/W	0xE0024024
HOUR	5	Регістр годин	R/W	0xE0024028
DOM	5	Регістр днів місяця	R/W	0xE002402C
DOW	3	Регістр днів тижня	R/W	0xE0024030
DOY	9	Регістр днів року	R/W	0xE0024034
MONTH	4	Регістр місяця	R/W	0xE0024038
YEAR	12	Регістр року	R/W	0xE002403C
Група сигнальних регістрів				
ALSEC	6	Сигнальне значення секунд	R/W	0xE0024060
ALMIN	6	Сигнальне значення хвилин	R/W	0xE0024064

Продовження таблиці4.65

Назва	**Розмір [біт]**	**Опис**	**Доступ**	**Адреса**
ALHOUR	5	Сигнальне значення годин	R/W	0xE0024068
ALDOM	5	Сигнальне значення днів місяця	R/W	0xE002406C
ALDOW	3	Сигнальне значення днів тижня	R/W	0xE0024070
ALDOY	9	Сигнальне значення днів року	R/W	0xE0024074
ALMON	4	Сигнальне значення місяців	R/W	0xE0024078
ALYEAR	12	Сигнальне значення років	R/W	0xE002407C
Регістри керування преддільником				
PREINT	13	Значення преддільника (ціла частина)	R/W	0xE0024080
PREFRAC	13	Значення преддільника (дробова частина)	R/W	0xE0024084

4.11.2.1 Змішана група регістрів

Регістр місцеположення переривань (ILR) – 3–бітний регістр, який визначає, який блок згенерував переривання (таблиця 4.66). Запис логічної одиниці в біт регістра скидає переривання, що відповідає цьому біту. Запис логічного нуля не буде мати жодного ефекту. Це дає можливість користувацькій програмі зчитувати цей регістр та знову записувати в нього прочитане значення, щоб скинути лише те переривання, яке було виявлено при зчитуванні.

Таблиця 4.66 – Біти регістра місцеположення переривань

Біт	Назва біта	Опис
0	RTCCIF	Якщо цей біт встановлено, то блок переривань від інкрементування одного з лічильників згенерував переривання. Запис логічної одиниці в цей біт скидає переривання, яке було згенероване від інкрементування лічильника.
1	RTCALF	Якщо цей біт встановлено, один з сигнальних регістрів згенерував переривання. Запис логічної одиниці в цей біт скидає це переривання.
2	RTSSF	Якщо цей біт встановлено, дозволено переривання від лічильника субсекундного інкременту. Частота переривань задається регістром вибору маски інкрементування лічильників (CISS).

Лічильник імпульсів тактової частоти (CTC) доступний тільки для читання (таблиця 4.67). Він може бути обнулений через регістр керування тактуванням. Регістр CTC складається з бітів лічильника преддільника тактової частоти.

Таблиця 4.67– Біти лічильника тактової частоти

Біт	Назва біта	Опис
0	Зарезервовано	Користувацьке програмне забезпечення не повинно виконувати запис в зарезервовані біти. Зчитування зарезервованого біта повертає невизначене значення.
15:1	Лічильник тактової частоти	До подачі імпульсу на лічильник секунд CTC відраховується 32768 імпульсів за секунду. Внаслідок наявності преддільника RTC, ці 32768 збільшень можуть мати тривалість неоднорідну за часом.

Регістр керування тактуванням (CCR) – 4–бітовий регістр, який керує операціями схеми ділення тактової частоти (таблиця 4.68).

Таблиця 4.68 – Біти регістра керування тактуванням

Біт	Назва біта	Опис
0	CLKEN	Дозвіл тактування. Коли цей біт встановлено, дозволено роботу лічильників часу. Коли біт скинуто, вони заблоковані таким чином, щоб їх можна було ініціалізувати програмно.
1	CTCRST	Скидання CTC. Якщо цей біт встановлено всі компоненти лічильника CTC скинуті. Компоненти залишаються в такому стані доки біт CCR не буде скинуто.
3:2	CTTEST	Дозвіл тестування. В нормальному режимі функціонування модуля RTC, ці біти завжди мають бути скинуті
4	CLKSRC	Якщо цей біт скинуто, лічильник імпульсів тактової частоти отримує тактові імпульси від преддільника RTC. Якщо цей біт встановлено, лічильник імпульсів тактової частоти отримує тактові імпульси з частотою 32768 Гц від генератора, який зв’язаний з виводами RTCX1, RTCX2.

Регістр переривання від інкрементування лічильників (CIIR) (таблиця 4.69) дозволяє генерувати переривання кожен раз, коли будь–який лічильник часу модуля RTC інкрементується. Це переривання залишається активним, доки не буде скинуто шляхом запису логічної одиниці у відповідний біт регістра ILR.

Таблиця 4.69 – Біти регістра переривання від інкрементування лічильників

Біт	Назва біта	Опис
0	IMSEC	Коли цей біт встановлено, це означає, що дозволено переривання від інкрементування регістра секунд
1	IMMIN	Коли цей біт встановлено, це означає, що дозволено переривання від інкрементування регістра хвилин
2	IMHOUR	Коли цей біт встановлено, це означає, що дозволено переривання від інкрементування регістра годин
3	IMDOM	Коли цей біт встановлено, це означає, що дозволено переривання від інкрементування регістра днів місяця
4	IMDOW	Коли цей біт встановлено, це означає, що дозволено переривання від інкрементування регістра днів тижня
5	IMDOY	Коли цей біт встановлено, це означає, що дозволено переривання від інкрементування регістра днів року
6	IMMON	Коли цей біт встановлено, це означає, що дозволено переривання від інкрементування регістра місяців
7	IMYEAR	Коли цей біт встановлено, це означає, що дозволено переривання від інкрементування регістра років

Регістр вибору маски інкрементування лічильників (CISS) – це регістр, який дозволяє отримувати переривання центрального процесора з мілісекундним інтервалом від RTC (таблиця 4.70). Це може дозволити звільнити один з таймерів загального призначення або підтримати режим низького споживання енергії процесором між періодичними перериванням.

Таблиця 4.70 – Біти регістра вибору маски інкрементування лічильників

Біт	Назва біта	Значення	Опис
2:0	SubSecSel		SubSecSel–Second Select. За допомогою цього поля можна обрати лічильник для субсекундного переривання.
		000	Переривання генерується кожні 16 відліків годинника. При частоті 32,768 кГц, затримка між перериваннями дорівнює приблизно 488 мкс.
		001	Переривання генерується кожні 32 відліки годинника. При частоті 32,768 кГц, затримка між перериваннями дорівнює приблизно 977 мкс.
		010	Переривання генерується кожні 64 відліки годинника. При частоті 32,768 кГц, затримка між перериваннями дорівнює приблизно 1,95 мс.
		011	Переривання генерується кожні 128 відліків годинника. При частоті 32,768 кГц, затримка між перериваннями дорівнює приблизно 3,9 мс.

Продовження таблиці 4.70

Біт	Назва біта	Значення	Опис
		100	Переривання генерується кожні 256 відліків годинника. При частоті 32,768 кГц, затримка між перериваннями дорівнює приблизно 7,8 мс.
		101	Переривання генерується кожні 512 відліків годинника. При частоті 32,768 кГц, затримка між перериваннями дорівнює приблизно 15,6 мс.
		110	Переривання генерується кожні 1024 відліків годинника. При частоті 32,768 кГц, затримка між перериваннями дорівнює приблизно 31,25 мс.
		111	Переривання генерується кожні 2048 відліків годинника. При частоті 32,768 кГц, затримка між перериваннями дорівнює приблизно 62,5 мс.
6:3			Резервовані біти
7	SubSecEna		Увімкнення або відключення переривань
		0	Відключено
		1	Увімкнено

Регістр сигнальної маски AMR (таблиця 4.71) дозволяє користувацькій програмі маскувати один з сигнальних регістрів Для реалізації сигнальної функції, кожному незамаскованому сигнальному регістру відповідає лічильник часу, який генерує переривання. Переривання буде сгенеровано тільки в тому випадку, коли результат порівняння буде спочатку (одноразово) змінюватися від розбіжності до збіжності. Переривання скидається, коли у відповідний біт регістра місцеположення переривання (ILR) записується логічна одиниця. Якщо всі біти маски встановлено, сигнал тривоги заблоковано.

Таблиця 4.71 – Біти регістра сигнальної маски

Біт	Назва біта	Опис
0	AMRSEC	Коли цей біт встановлено, значення регістра секунд не використовується для порівняння при виробленні сигналу тривоги.
1	AMRMIN	Коли цей біт встановлено, значення регістра хвилин не використовується для порівняння при виробленні сигналу тривоги.
2	AMRHOUR	Коли цей біт встановлено, значення регістра годин не використовується для порівняння при виробленні сигналу тривоги.
3	AMRDOM	Коли цей біт встановлено, значення регістра днів місяця не використовується для порівняння при виробленні сигналу тривоги.

Продовження таблиці 4.71

Біт	Назва біта	Опис
4	AMRDOW	Коли цей біт встановлено, значення регістра днів тижня не використовується для порівняння при виробленні сигналу тривоги.
5	AMRDOY	Коли цей біт встановлено, значення регістра днів року не використовується для порівняння при виробленні сигналу тривоги.
6	AMRMON	Коли цей біт встановлено, значення регістра місяців не використовується для порівняння при виробленні сигналу тривоги.
7	AMRYEAR	Коли цей біт встановлено, значення регістра років не використовується для порівняння при виробленні сигналу тривоги.

Регістр консолідованого часу 0 (CTIME0) містить значення секунд, хвилин, годин та дня тижня (таблиця 4.72).

Таблиця 4.72 – Біти регістра консолідованого часу 0

Біт	Назва біта	Опис
5:0	Seconds	Значення секунд від 0 до 59
7:6	–	Зарезервовані значення
13:8	Minutes	Значення хвилин від 0 до 59
15:14	–	Зарезервовані значення
20:16	Hours	Значення годин від 0 до 23
23:21	–	Зарезервовані значення
26:24	Day of Week	Значення дня тижня від 0 до 6
31:27	–	Зарезервовані значення

Регістр консолідованого часу 1 (CTIME1) містить значення дня місяця, місяця та року (таблиця 4.73).

Таблиця 4.73 – Біти регістра консолідованого часу 1

Біт	Назва біта	Опис
4:0	Day of Month	Значення дня місця в проміжку від 1 до 28, 29, 30 або 31 (в залежності від місяця та чи є рік високосним)
7:5	–	Зарезервовані значення
11:8	Month	Значення місяця від 1 до 12
15:12	–	Зарезервовані значення
27:16	Year	Значення року від 0 до 4095
31:28	–	Зарезервовані значення

Регістр консолідованого часу 2 (CTIME2) містить значення дня року (таблиця 4.74).

Таблиця 4.74 – Біти регістра консолідованого часу 2

Біт	Назва біта	Опис
11:0	Day of Year	Значення дня року від 1 до 365 (366 для високосного року)
31:12	–	Зарезервовані значення

4.11.2.2 Група лічильників часу

Група лічильників часу складається з восьми лічильників, які описано в таблицях 4.75, 4.76. Зчитування або запис в ці лічильники може здійснюватися з використанням адрес, які приведено в таблиці 4.76.

Таблиця 4.75– Характеристика регістрів лічильників часу

Лічильник	Розмір	Тактується від	Мінімальне значення	Максимальне значення
Секунди	6	Clk 1	0	59
Хвилини	6	Секунд	0	59
Години	5	Хвилин	0	23
Дні місяця	5	Годин	1	28, 29, 30 або 31
Дні тижня	3	Годин	0	6
Дні року	9	Годин	1	365 або 366
Місяці	4	Днів місяця	1	12
Роки	12	Місяців або днів року	0	4095

Таблиця 4.76– Група регістрів лічильників часу

Адреса	Назва	Розмір	Опис	Доступ
0xE0024020	SEC	6	Значення секунд лежить в діапазоні 0…59	R/W
0xE0024024	MIN	6	Значення хвилин лежить в діапазоні 0…59	R/W
0xE0024028	HOUR	5	Значення годин лежить в діапазоні 0…23	R/W
0xE002402C	DOM	5	Значення днів місяця лежить в діапазоні 1…28,29,30,31	R/W
0xE0024030	DOW	3	Значення днів тижня лежить в діапазоні 0…6	R/W
0xE0024034	DOY	9	Значення днів року лежить в діапазоні 1…365, або 366	R/W
0xE0024038	MONTH	4	Значення місяців лежить в діапазоні 1…12	R/W
0xE002403C	YEAR	12	Значення років лежить в діапазоні 0…4095	R/W

4.11.2.3 Група сигнальних регістрів

Сигнальні регістри описано в таблиці 4.77. Значення цих регістрів порівнюються із значеннями лічильників часу. При збігу значення одного з незамаскованих сигнальних регістрів зі значенням відповідного лічильника часу буде сгенеровано переривання. Переривання буде скинуто за записом логічної одиниці у відповідний біт регістра місцеположення переривання ILR[1].

Таблиця 4.77– Група сигнальних регістрів модуля RTC

Адреса	Назва	Розмір	Опис	Доступ
0xE0024060	ALSEC	6	Сигнальне значення для секунд	R/W
0xE0024064	ALMIN	6	Сигнальне значення для хвилин	R/W
0xE0024068	ALHOUR	5	Сигнальне значення для годин	R/W
0xE002406C	ALDOM	5	Сигнальне значення для днів місяця	R/W
0xE0024070	ALDOW	3	Сигнальне значення для днів тижня	R/W
0xE0024074	ALDOY	9	Сигнальне значення для днів року	R/W
0xE0024078	ALMON	4	Сигнальне значення для місяців	R/W
0xE002407C	ALYEAR	12	Сигнальне значення для років	R/W

4.11.2.4 Група регістрів керування преддільником

Опис цих регістрів наведено в кінці таблиці 4.65, а використання відображають формули 4.23 та 4.24.

4.11.3 Приклад програмування RTC–модуля

4.11.3.1 Схема алгоритму роботи

Схема алгоритму роботи наведено на рисунку 4.66.

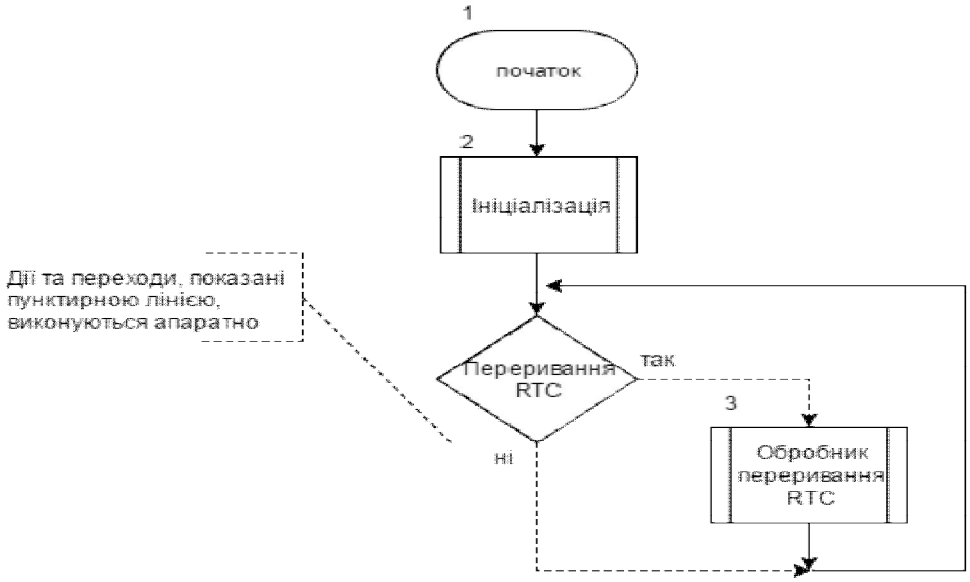


Рисунок 4.66 – Схема алгоритму роботи

Схему алгоритму ініціалізації наведено на рисунку 4.67.

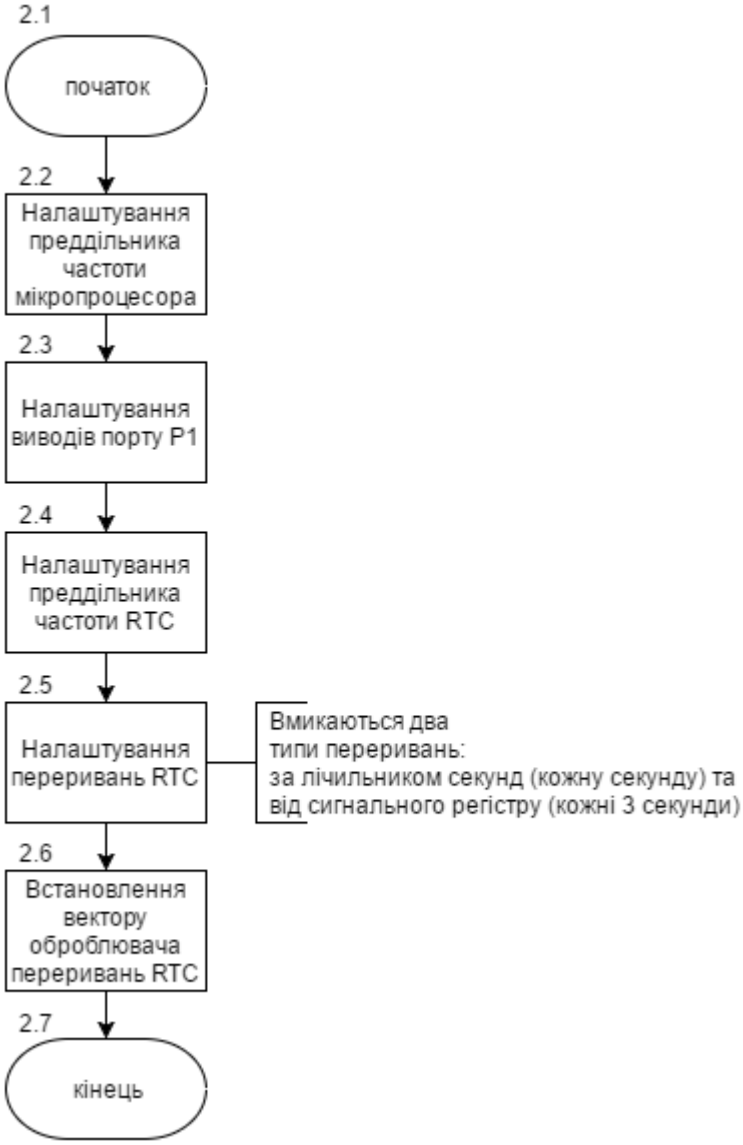


Рисунок 4.67 – Схема алгоритму ініціалізації

Схему алгоритму обробки переривання від RTC–модуля наведено на рисунку 4.68.

4.11.3.2 Програма на мові С

```
// 2
int main(void)
{
    // 2.2
    APBDIV   = 0x00000002;    // Встановлення преддільника частоти мікропроцесора
Cclk(=60) / 2 = 30 МГц
    // 2.3
    IODIR1  = 0x00FF0000;   // Виводи P1.[23– 16] –  виходи
    IOSET1  = 0x00020000;   // P1.17 встановити високий рівень
```

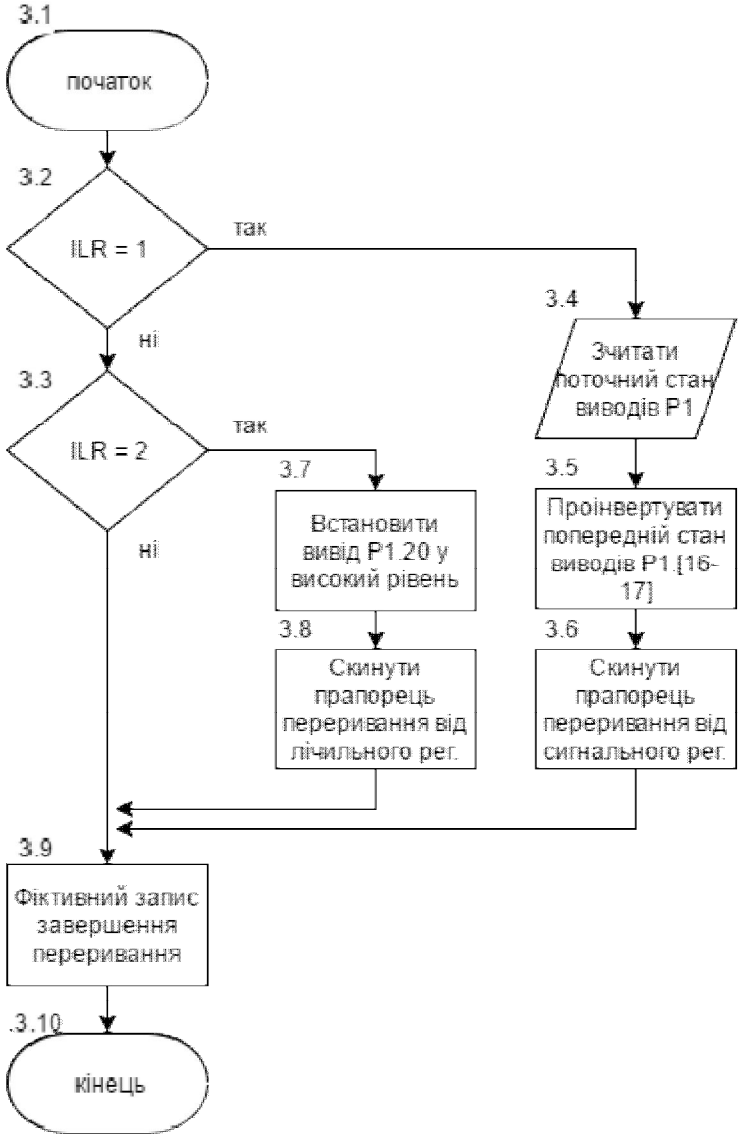


Рисунок 4.68 – Схема алгоритму обробки переривання

```
    // 2.4
    PREINT  = 0x00000392;   // Конфігуруємо преддільник RTC для Pclk = 30 МГц
```

```
        PREFRAC = 0x00004380;
        // 2.5
        CIIR    = 0x00000001;   // Дозволяємо переривання від лічильника секунд
        ALSEC   = 0x00000003;   // Встановлюємо сигнал на 3 с
        AMR     = 0x000000FE;   // Дозволяємо сигнал секунд
        CCR     = 0x00000001;   // Запускаємо RTC
        // 2.6
        VICVectAddr13 = (unsigned)RTC_isr;  // Задаємо адресу оброблювача переривання
від RTC
        VICVectCntl13 = 0x0000002D;         // Задаємо канал
        VICIntEnable  = 0x00002000;         // Дозволяємо переривання від RTC

        while(1);
    }
    // 3
    void RTC_isr(void)
    {
        unsigned led;
        // 3.2
        if(ILR & 0x00000001)             // Перевіряємо на переривання від лічильного
регістра (спрацьовує кожну секунду)
        {
            // 3.4
            led     = IOPIN1;           // Зчитуємо поточний стан виводів
            // 3.5
            IOCLR1  = led & 0x00030000; // Вимикаємо LED (P1.[16…17])
            IOSET1  = ~led & 0x00030000;// Вмикаємо LED "idle"
            // 3.6
            ILR     = 0x00000001;       // Скидаємо прапорець переривання
        }
        // 3.3
        if(ILR & 0x00000002)             // Перевіряємо на переривання від сигнальних
регістрів (спрацьовує кожні 3 секунди)
        {
            // 3.7
            IOSET1 = 0x00100000;    // Вмикаємо LED на P1.20
            // 3.8
            ILR    = 0x00000002;    // Скидаємо прапорець переривання
        }
        // 3.9
        VICVectAddr = 0x00000000;   // Фіктивний запис для індикації завершення пере-
ривання
    }
```

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Опишіть призначення годинника реального часу (Real Time Clock – RTC).
2) Які мікроконтролери, як правило, використовуються для реалізації RTC?
3) Чому дорівнює частота кварцового резонатора, який використовується в модулі RTC?
4) До яких виводів мікроконтролера підключається зовнішній «годинниковий» резонатор?
5) Наведіть та опишіть спрощену структуру модуля RTC.
6) Як відбувається формування тактового сигналу RTC з сигналу Pclk?
7) Якими регістрами програмується робота преддільника при формуванні тактового сигналу RTC з сигналу Pclk?
8) За якими формулами обчислюються значення регістрів PREINT та PREFRAC?
9) Скільки лічільних регістрів має модуль RTC та яку функцію вони виконують?
10) Скільки групових регістрів має модуль RTC та яку функцію вони виконують?
11) Назвіть джерела та опишіть механізми формування та обробки переривань.
12) Назвіть та опишіть керуючі регістри модуля RTC.
13) Опишіть призначення бітів змішаної групи регістрів.
14) Опишіть призначення бітів групи регістрів лічильників часу.
15) Опишіть призначення бітів групи сигнальних регістрів.
16) Наведіть та поясніть схему алгоритму роботи модулю.
17) Наведіть та поясніть схему алгоритму ініціалізації модулю.
18) Наведіть та поясніть схему алгоритму обробки переривання.
19) Яку функцію виконує НОЗП у складі модуля RTC?
20) Які дії треба виконати для запуску лічильників модуля RTC?
21) Що станеться коли значення всіх дозволених сигнальних регістрів співпадуть з вмістом лічильних регістрів?

4.12 Контролер CAN – інтерфейсу

4.12.1 Загальні відомості

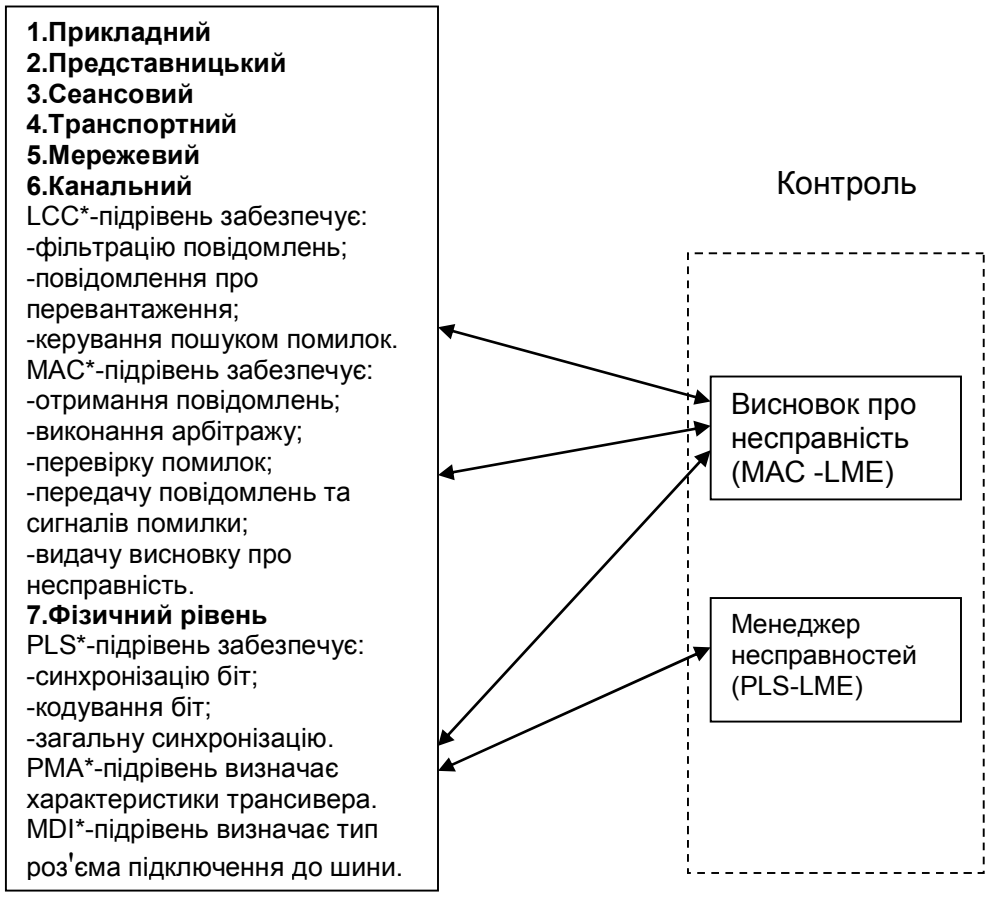
У мікроконтролерах сімейства LPC2300 є 2 незалежних контролери інтерфейсу CAN [1, 4, 8]. Незважаючи на те що спочатку протокол CAN був розроблений для об'єднання в мережу пристроїв автомобільної електроніки, в даний час він є одним з найбільш поширених мережевих протоколів для вбудованих систем і широко використовується в розподілених системах керування. Нижче, як приклад, розглядається CAN – протокол і CAN – модуль, який реалізовано в сімействі LPC2300.

Протокол CAN (Controller Area Network) було розроблено фірмою Robert Bosch GmbH в 1982 році для потреб автомобільної промисловості. За наступні 24 роки протокол CAN став стандартом в створенні бортових мереж автомобілів. Крім того, він набув широкого поширення в областях, які не мають ніякого відношення до автомобільної промисловості, однак вимагають об'єднання в мережу вбудованих пристроїв. У протоколу CAN багато особливостей, привабливих для розробників вбудованих систем. Він дозволяє створювати дешеві та прості у реалізації однорангові мережі з розвиненими засобами контролю помилок і швидкістю передачі до 1 Мбіт / с. Всі пакети в протоколі CAN досить короткі і можуть містити не більше восьми байтів даних. Ця обставина робить протокол CAN придатним для реалізації невеликих вбудованих мереж, які потребують надійної передачі критичних даних між вузлами.

4.12.1.1 Семирівнева модель OSI

Протокол CAN реалізує другий рівень (канальний) моделі осі (рисунок 4.69), тобто здійснює формування пакетів повідомлень, обмеження поширення помилок (error containment), підтвердження прийому і арбітраж.

У протоколі CAN відсутнє чітке визначення фізичного рівня, тому для передачі CAN – повідомлень можуть використовуватися різні фізичні середовища. Проте, найбільш поширеним середовищем передачі є “вита” пара, для якої випускаються стандартні мікросхеми драйверів лінії.



*Примітка: LLC (logical link control) – підрівень контролю логічного з'єднання, описує верхні компоненти канального рівня ISO/OSI; MAC (medium access control) – підрівень контролю доступу до комунікаційного середовища, представляє нижні компоненти канального рівня ISO/OSI; PLS (physical signaling) – підрівень фізичного рівня. Приймає і передає в схему трансивера бітовий потік. Виконує кодування/декодування, контролює синхронізацію; PMA (physical medium attachment) підрівень фізичного рівня, що відповідає за керування передачею, прийомом, виявлення колізій, синхронізацію і корекцію розсинхронізації. MDI (medium dependent interface) – підрівень, що визначає вимоги до роз'ємів, кабелів та термінаторів.

Рисунок 4.69 – Семирівнева модель OSI

Інші рівні моделі ISO абсолютно не визначені, тому прикладна програма працює безпосередньо з регістрами модуля CAN. Взагалі кажучи, модуль CAN можна використовувати так само, як і загальнопоширений UART, не реалізуючи дорогий і складний стек протоколів. Оскільки протокол CAN застосовується і в області автоматизації промислового виробництва, існує певна кількість специфікацій ПЗ, що визначають, яким чином повідомлення CAN використовуються для передачі даних між обладнання різних виробників. Найбільш поширеними стандартами прикладного рівня є CANopen і DeviceNet. Єдине призначення цих стандартів – це забезпечення взаємодії між обладнанням різних виробників. Якщо розробляється власна закрита система, то немає необхідності використовувати ці протоколи прикладного рівня, і ви вільно

можете реалізувати свій внутрішній протокол, як це робить більшість розробників.

4.12.1.2 Структура вузла мережі CAN

Типовий вузол мережі CAN зображено на рисунку 4.70. Кожен вузол складається з мікроконтролера і спеціалізованого контролера CAN. Цей контролер може бути розташований на одному кристалі з мікроконтролером (як в мікроконтролерах сімейства LPC2300), а може бути і окремої мікросхемою. Через драйвер, контролер CAN підключається до лінії на “витій” парі, на кінцях якої підключено узгоджуючі резистори–термінатори, опором 120 Ом. Найбільш поширеною помилкою, яку здійснюють розробники при побудові мережі CAN, є відсутність цих узгоджувальних резисторів, без яких мережа є непрацездатною.

Одна з важливих особливостей конструкції вузла мережі CAN полягає в тому, що контролер CAN має незалежні канали прийому та передачі даних від / до драйверу лінії. Так що, одночасно з видачею даних на шину, вузол може зчитувати їх назад. На цій особливості засновано арбітраж повідомлень і виявлення ряду помилок.

Логічні рівні у “витій” парі формуються наступним чином. Логічній одиниці, або вільній шині, відповідає подача на обидва дроти напруги, яка дорівнює половині різниці напруг між 0 і Vcc. Логічному нулю відповідає подача на дроти лінії диференціальної напруги V_{diff} (рисунок. 4.71).

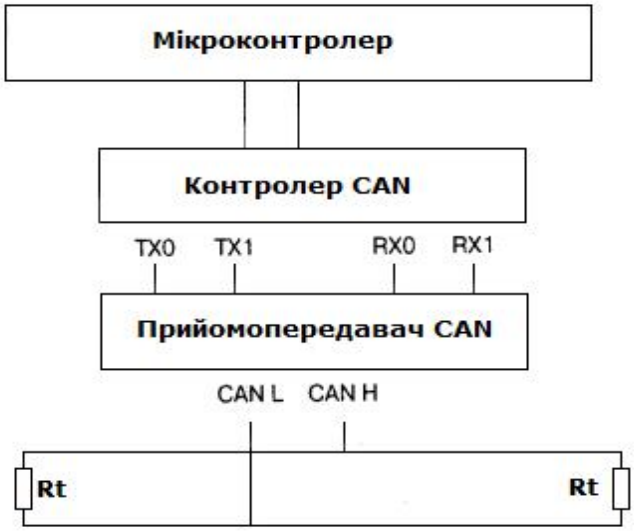


Рисунок 4.70 – Структура CAN–вузла

У термінології протоколу CAN, логічна одиниця називається рецесивним бітом, а логічний нуль називається домінантним бітом. У всіх випадках домінантний біт буде мати завжди більший пріоритет, ніж рецесивний.

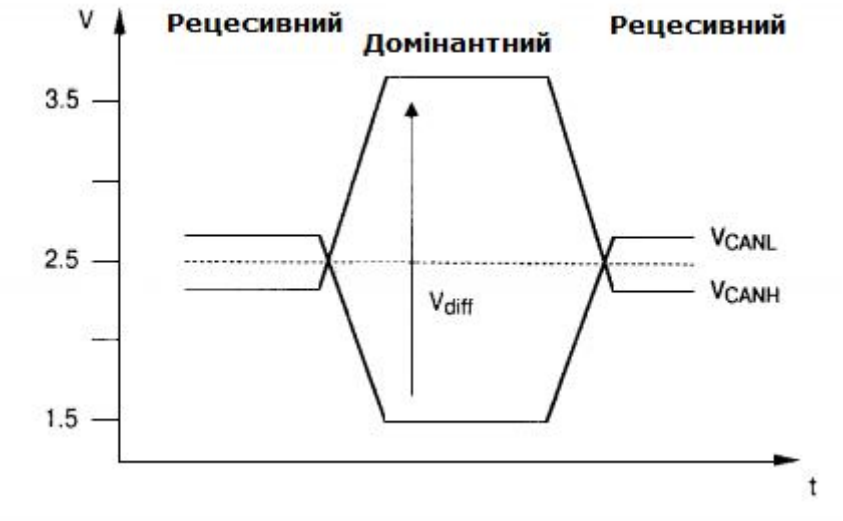


Рисунок 4.71 – Сигнали фізичного рівня CAN

Тобто, якщо десять вузлів запишуть на шину рецесивний біт, а один – домінантний, то назад усіма вузлами буде прочитано домінантний біт. Шина CAN дозволяє передавати дані зі швидкістю до 1 Мбіт/с. Зазвичай така швидкість досягається при довжині кабелю не більше 40 м. Знижуючи швидкість передачі, можна реалізовувати більш протяжні мережі. На практиці з використанням стандартних драйверів можна реалізувати мережу довжиною 1500 м із швидкістю передачі 10 Кбіт/с.

4.12.1.3 Об’єкти повідомлень CAN

У специфікації шини CAN визначено два об'єкти повідомлень, які можуть формуватися прикладним ПЗ. Об'єкт повідомлення використовується для передачі даних по мережі. Формат пакета повідомлення показано на рисунку 4.72.

Пакет повідомлення формується контролером CAN. Прикладне ПЗ надає тільки байти даних, ідентифікатор повідомлення і значення біта RTR.

Рисунок 4.72– Формат пакета повідомлень CAN

Пакет повідомлення починається з домінантного біта, що є ознакою початку кадру (Start Of Frame – SOF). Потім йде 11 – бітний або 29 – бітний ідентифікатор повідомлення. Ідентифікатор повідомлення використовується для ідентифікації даних, які передаються в цьому пакеті. Протокол CAN є протоколом типу «виробник/споживач». Кожне конкретне повідомлення, яке вироблено будь – яким унікальним вузлом, може бути одночасно споживано будь – якою кількістю вузлів мережі. Можна здійснювати і обмін типу «точка – точка», якщо зробити, щоб цей ідентифікатор приймався тільки одним вузлом мережі. Після цього повідомлення може бути надіслано вузлом – виробником конкретному вузлу – споживачу в мережі. У пакеті повідомлення біт RTR завжди скинуто. В полі DLC (Data Length Code – код довжини даних) міститься число від 0 до 8, що показує кількість байтів даних, що передаються в пакеті.

Таким чином, незважаючи на те що в повідомленні можна надсилати 8 байт даних, є можливість зменшення довжини пакета повідомлення для збереження пропускної здатності шини CAN. Слідом за байтами даних передається 15 – бітна послідовність циклічного надлишкового коду. Цей код забезпечує виявлення і корекцію помилок в бітах повідомлення від початку кадру до початку поля CRC. Після поля CRC розташовується поле підтвердження ACK. Передавальний вузол очікує, що приймаючий вузол виставить підтвердження в цьому біті під час передачі пакета. На практиці передавач посилає рецесивний біт, а всі вузли, які отримали повідомлення до цього поля, видають на шину домінантний біт, генеруючи, таким чином, підтвердження. Тобто передавачу все рівно, скільки вузлів підтвердили прийом повідомлення – 1 або 100. Тому при розробці програм прикладного рівня слід

враховувати, що дані підтвердження можна розглядати тільки як попереднє, а не повне підтвердження того, що повідомлення дійшло до всіх адресатів. Після поля підтвердження ACK формується поле кінця кадру (EndOfFrame – EOF).

Шина CAN може працювати в режимі master/slave. Вузол мережі CAN може послати по мережі запит на віддалену передачу, передавши пакет повідомлення, що не містить даних, але з встановленим прапорцем RTR (кадр віддаленого запиту). Цей кадр запитує передачу пакета повідомлення з ідентифікатором, який відповідає ідентифікатору в запиті (рисунок 4.73). При прийомі кадру віддаленого запиту вузол, який формує потрібне повідомлення, передає відповідний кадр повідомлення.

Рисунок 4.73– Запит на віддалену передачу

Як вже було зазначено, ідентифікатор повідомлення в протоколі CAN є 11 – бітним або 29 – бітним. Взагалі кажучи, існує дві версії протоколу CAN, що відрізняються тільки довжиною поля ідентифікатора (таблиця 4.78).

Таблиця 4.78 – Версії CAN – протоколу

Протокол	Ідентифікатор
2.0A	11 – бітний
2.0B Passive	11 – бітний
2.0B Active	29 – бітний

На одній шині можна використовувати обидві версії протоколу. Єдине, що не можна робити, – це посилати повідомлення з 29 – бітовим ідентифікатором пристроям стандарту 2.0A (таблиця 4.79).

Таблиця 4.79 – Можливий розмір ідентифікатора для різних версій CAN – протоколу

Протокол	Кадр з 11 – бітним ідентиф.	Кадр з 29 – бітним ідентиф.
Модуль 2.0A	Tx/Rx OK	Помилка на лінії
Модуль 2.0B Passive	Tx/Rx OK	Ігнорується
Модуль 2.0B Active	Tx/Rx OK	Tx/Rx OK

4.12.1.4 Арбітраж на шині CAN

Якщо планується передача повідомлення, і шина вільна, то повідомлення буде передано і його зможе прийняти будь – який зацікавлений в цьому повідомленні вузол. Якщо передача повідомлення запланована, а шина активна, то перш ніж приступити до передачі повідомлення, необхідно дочекатися звільнення шини. Якщо заплановано передачу декількох повідомлень, то при звільненні шини вони почнуть передаватися одночасно, синхронізуючись за ознакою початку кадру. У цьому випадку на шині почнеться процес арбітражу, завдання якого – визначити, яке повідомлень захопить шину і буде передано.

Арбітраж повідомлень на шині CAN здійснюється методом, який називається «неруйнівний побітовий арбітраж». На рисунку 4.74 зображено три повідомлення, які мають бути надіслані.

Після звільнення шини і синхронізації кадрів повідомлень за старт – бітом, на шину починають видаватися всі три ідентифікатори. При передачі перших двох бітів всі три вузли виставляють на шину однакові логічні рівні і, відповідно, зчитують ті ж значення, тому вони все продовжують передачу.

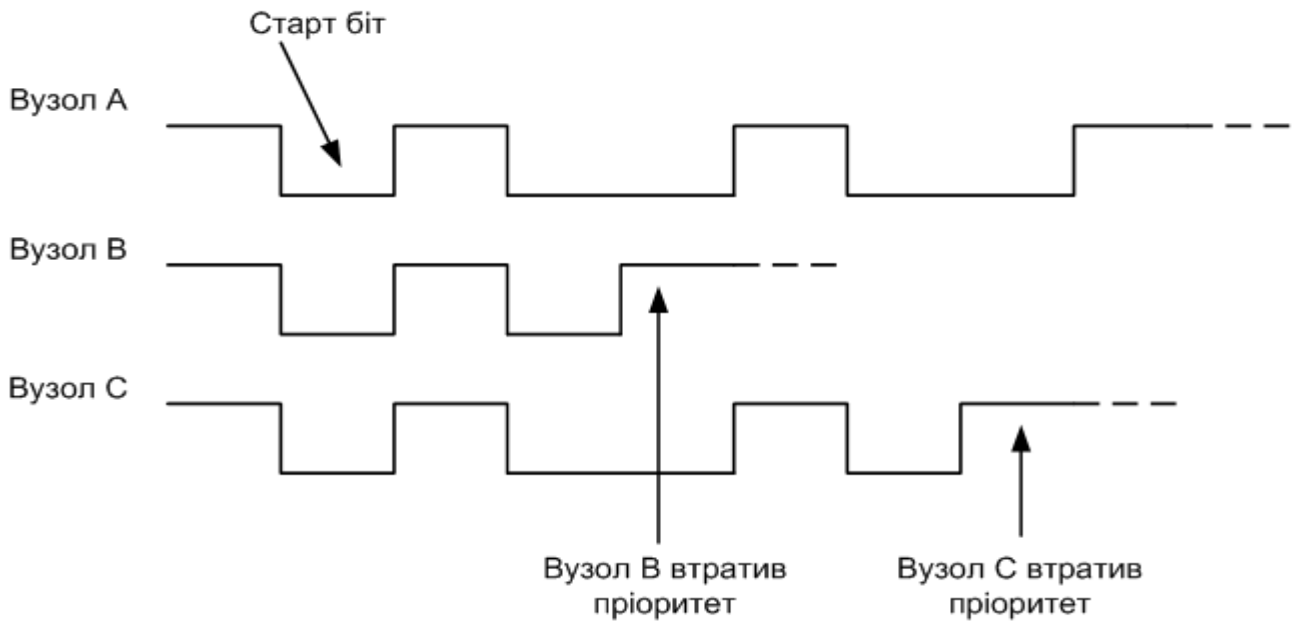


Рисунок 4.74 – Арбітраж на шині CAN

Однак при передачі чергового біта, вузли A і C виставляють на шину домінантний біт, а вузол B – рецесивний. Тобто вузол B, що виставив рецесивний біт, зчитує домінантний. У цьому випадку він звільняє шину і починає стежити за її станом. Вузли A і C продовжують передачу до тих пір, поки вузол C не видасть рецесивний біт, а вузол A – домінантний. При цьому вузол C припиняє передачу і починає стежити за станом шини. З цього моменту шина захоплюється вузлом A, який передає своє повідомлення. Після того як вузол A завершить передачу, починають передачу вузли B і C, причому вузол C захоплює шину і передає своє повідомлення. Останнім передає своє повідомлення вузол B. Якби вузол A знову захотів передати повідомлення, він би знову захопив шину, незважаючи на те, що вузли B і C чекають її звільнення. Таким чином, першим по шині CAN передається повідомлення з найменшим ідентифікатором.

4.12.2 CAN–модуль ARM–мікроконтролера NXP LPC2378

На рисунку 2.3, як приклад, зображено структурну схему мікроконтролера NXP LPC2378, який базується на ядрі ARM7TDMI – S, що дозволяє використовувати сучасну периферію, в тому числі стандарт передачі даних CAN.

У мікроконтролері LPC2378 є два вбудованих контролери CAN1 і CAN2. Кожен контролер має власні регістри стану і регістри керування, три буфери

передачі і два буфери прийому. Кожен з CAN – контролерів може працювати в напівдуплексному режимі.

На рисунку 4.75 наведено спрощену структуру CAN – модуля.

CAN – інтерфейс включає:

- APB Interface – забезпечує доступ до усіх регістрів CAN;
- Acceptance Filter –приймальний фільтр;
- Vectored Interrupt Controller (VIC) – векторний контролер переривань;
- CAN Transceiver – CAN – трансивер;
- Common Status Registers – регістр загального статусу.

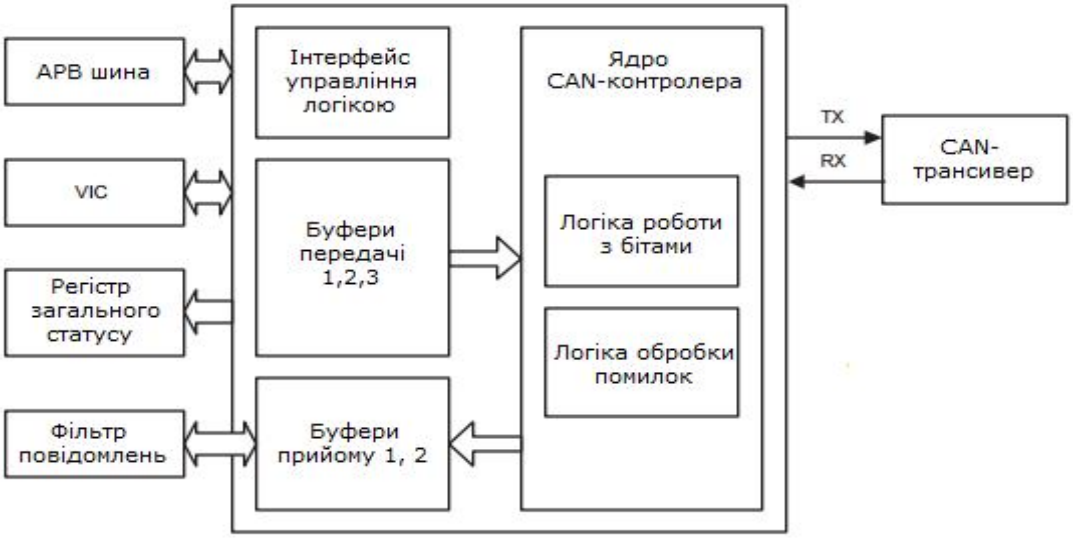


Рисунок 4.75 – Структура контролера CAN

Кожен CAN–контролер складається з CAN–ядра, ОЗП–повідомлень, кінцевого автомата обробки повідомлень і регістрів керування. CAN–контролер LPC2378 являє собою контролер CAN–протоколу і не має драйверів фізичного рівня (трансиверів). Типову схему конфігурації CAN–мережі наведено на рисунку 4.76.

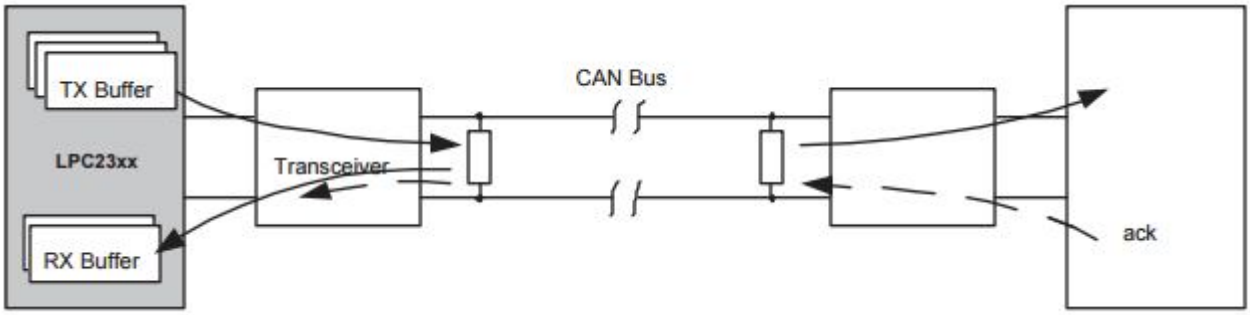


Рисунок 4.76 – Типова схема конфігурації CAN–мережі

CAN–контролер LPC2378 забезпечує асинхронний послідовний обмін даними зі швидкістю до 1Мбіт/сек., хоча ця швидкість може бути обмежена фізичним середовищем, яке обрано для передачі даних CAN–шиною.

Для ефективного керування прийомом інформації за CAN – протоколом в LPC2378 передбачено приймальний фільтр повідомлень (Acceptance Filter), який може бути налаштовано на прийом як конкретних ідентифікаторів, так і цілих діапазонів ідентифікаторів. Фільтр дозволяє спростити програмну частину проекту і прискорити роботу всієї системи.

Кожен CAN – контролер може бути налаштовано на різну частоту сигналу синхронізації приймача – передавача завдяки вбудованому подільнику частоти шини APB. Є можливість генерувати переривання за такими подіями, як: прийом повідомлення, завершення передачі повідомлення, виявлення помилки, перезапис буфера приймача, невдача при арбітражі, тощо. Всього передбачено 11 варіантів переривань.

4.12.2.1 Особливості CAN – модуля мікроконтролера LCP2378

Загальні властивості модуля CAN в LPC2378:

- сумісність зі специфікацією CAN 2.0B, ISO 11898 – 1;
- пріоритет доступу до шини визначається ідентифікатором повідомлення (11 – розрядна або 29 – розрядна версії);
- гарантований час затримки для повідомлень з високим пріоритетом.
- програмована швидкість передачі даних (до 1 Мбіт/с);
- Multicast передача повідомлень (пакети надсилаються певній підмножині адресатів) і Broadcast передача повідомлень (передача повідомлення всім приймачам мережі);
- кількість байт даних від 0 до 8;
- потужна обробка помилок.

Особливості модуля CAN в LPC2378:

- 2 CAN–контролери;
- підтримка 11 – бітного та 29 – бітного ідентифікатора;
- подвійний буфер прийому і потрійний буфер передачі;

– швидкий апаратно реалізований алгоритм підтримки пошуку великої кількості CAN – ідентифікаторів;
– глобальний приймальний фільтр розпізнає ідентифікатори розміром 11 – біт або 29 – біт для всіх повідомлень CAN – шини;
– приймальний фільтр може працювати в FullCAN – режимі.

4.12.2.2 Тактова синхронізація

На відміну від багатьох інших протоколів послідовної передачі даних, в CAN – модулі є подільники, які слугують для формування квантів часу, тобто часових слотів. Для передачі одного біта повідомлення використовується декількох таких квантів (рисунок 4.77)

Інтервал передачі біта складається з трьох сегментів. Першим є сегмент Sync, тривалість якого фіксована і дорівнює одному кванту часу. Наступні два сегменти називаються Tsegl і Tseg2. Кількість квантів в кожному із сегментів визначається користувачем.

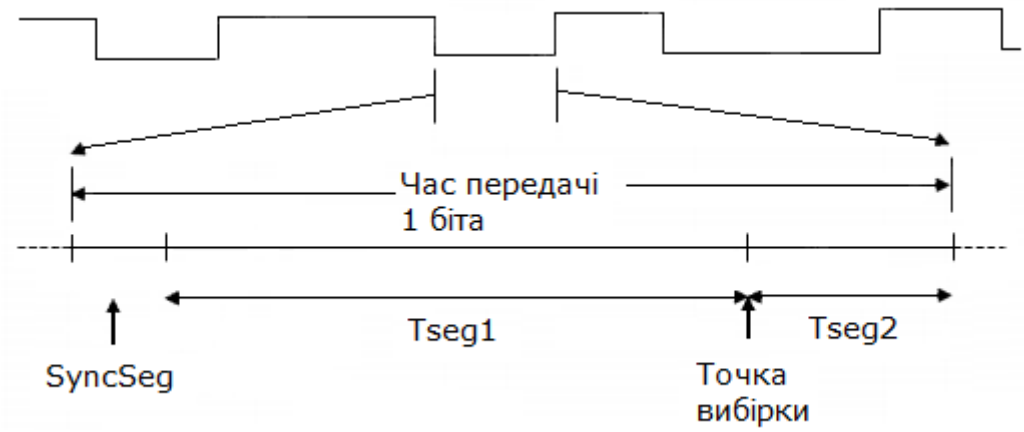


Рисунок 4.77 – Інтервал передачі біта

Найменша кількість квантів часу, складових інтервалу передачі біта, дорівнює 3, а найбільша – 17. Точка вибірки біта пристроєм розташована в кінці сегмента Tsegl. При зміні співвідношення між Tsegl і Tseg2 точка вибірки зміщується. Ця особливість дозволяє підлаштовувати протокол CAN під характеристики конкретного каналу передачі. Якщо ви працюєте на довгих лініях, точку вибірки можна зсунути назад. Якщо використовується генератор з великим дрейфом частоти, точку вибірки бажано зсунути вперед. Приймаючий

пристрій може підлаштовувати свою бітову швидкість передачі для синхронізації з передавальним пристроєм. Значення, на яке може підлаштовуватися кожен біт, називається шириною стрибка синхронізації (Synchronization Jump Width – SJW) і може бути програмно встановлено в межах від 1 до 4 квантів часу. Це значення також визначається користувачем.

Для обчислення інтервалу передачі біта використовується наступна формула:

$$Bit\ Rate\ = \frac{Pclk}{BRP * (1\ +\ Tseg1\ +\ Tseg2)}, \qquad (4.25)$$

де BRP (Band Rate Prescaler) – значення предільника генератора швидкості передачі.

У цьому виразі міститься багато невідомих. Припустимо, ми хочемо отримати швидкість бітової передачі, яка дорівнює 125 Кбіт/с при Pclk = 60 МГц, і положенні точки вибірки близько 70%. Тепер нам залишається обчислити значення BRP.

Загальна кількість квантів часу в інтервалі передачі одного біта 1 + Tsegl + Tscg2. Назвемо цю величину QUANTA і перепишемо рівняння щодо значення предільника:

$$BRP\ = \frac{Pclk}{Bit\ Rate * \ QUANTA}. \qquad (4.26)$$

Підставляючи відомі значення:

$$\mathrm{B}RP\ = \frac{60\text{МГц}}{125\text{К} * \ QUANTA}\ .$$

Тепер згадуємо, що в інтервалі передачі біта може міститися від 3 до 17 квантів часу. Тому нам треба підібрати таке ціле значення змінної QUANTA між 3 і 17, щоб отримати ціле значення BRP.

Якщо QUANTA = 16, тоді BRP = 30.

$$QUANTA = 16 = \ (1\ +\ Tseg1\ +\ Tseg2).$$

Тепер ми можемо підібрати співвідношення між Tseg1 і Tseg2, яке дасть нам бажане положення точки вибірки SP (Sample Point):

$$SP = \frac{QUANTA\ x\ 70}{100}.$$

У нашому випадку 16 х 0.7 = 11.2. Цьому числу відповідає Tsegl = 10, Tseg2 = 5, а дійсний стан точки вибірки становитиме 68,8%.

Величину ширини стрибка синхронізації SJW можна обчислити, використовуючи наступні емпіричні формули [4, 8]:

Якщо $Tseg2 \geq 5$, то $SJW = 4$. (4.27)

Якщо $Tseg2 < 5$, то $SJW = (Tseg2 - 1)$. (4.28)

В нашому випадку SJW = 4.

4.12.2.3 Обмеження розповсюдження помилок

У специфікації CAN – протоколу визначено п'ять методів обмеження поширення помилок, які реалізовано на апаратному рівні. При виявленні будь – якої помилки передавальний пристрій повторює посилку повідомлення, тому ЦП не потрібно втручатися до тих пір, доки на шині не виникне груба помилка. Передбачено три методи виявлення помилок на рівні пакетів (контроль формату, CRC і підтвердження) і два методи на рівні бітів (контроль бітів і бітстаффінг). Для реалізації цих методів використовується кілька полів, що додаються до основного повідомлення. При прийомі здійснюється перевірка, чи всі поля присутні в повідомленні (рисунок 4.78). Якщо не всі поля присутні, то повідомлення ігнорується і генерується кадр помилки. Це гарантує отримання повного повідомлення, що має коректний формат.

Стандартний кадр даних												
1	11/29	1	1	1	4	0-64 (0-8 байт)	15	1	1	1	7	3
SOF	ID	RTR	IDE	*	DLC	Поле даних	CRC	Роздільник CRC	ACK	Роздільник ACK	Кінець кадру	Пауза
		Д	Д	Д				Р		Р	Р	

Д - домінантний рівень
Р - рецесивний рівень
* - зарезервований біт

Рисунок 4.78 – Формат кадру CAN – повідомлення

Кожне повідомлення має бути підтверджено вставкою домінантного біта в полі підтвердження ACK. В полі ACK передавальний вузол посилає рецесивний біт. Приймач, який отримав повідомлення правильно, повідомляє про це передавача, посилаючи домінантний біт в полі ACK. Якщо підтвердження не прийнято, вузол, який передає, буде передавати повідомлення до тих пір, поки не отримає підтвердження.

Також пакет CAN – повідомлення містить 15 – бітне значення CRC, яке автоматично генерується передавачем і перевіряється приймачем. За допомогою CRC – коду можна виявити і виправити помилку в бітах повідомлення від початку кадру до початку поля CRC. Якщо CRC невірний і повідомлення ігнорується, то передається кадр помилки.

Послідовність контролю кадру, яка отримується за правилом надлишкового циклічного коду, найкраще підходить для кадрів із числом бітів менше ніж 127 біт. При обчисленні CRC береться початковий поліном, що визначається як поліном, коефіцієнти якого задані послідовністю біт, які складаються з полів: "початок кадру", "поле арбітражу", "поле контролю", "поле даних" (якщо є) і, для 15 молодших коефіцієнтів, 0. Цей поліном ділиться на поліном:

$$X^{15} + X^{14} + X^{10} + X^{8} + X^{7} + X^{4} + X^{3} + 1 \ .$$

Залишок від цього поліноміального ділення і є послідовність CRC, яка передається шиною.

Після того як вузол виграє арбітраж, він почне передачу свого повідомлення по шині. Як і під час арбітражу, CAN – контролер зчитує кожен біт, що видається їм на шину. Оскільки вузол вже виграв арбітраж, більше ніхто не повинен передавати дані по шині, тому значення кожного виданого на шину біта має відповідати значенню, прочитаному з шини. Якщо прочитано невірне значення, то передавач генерує кадр помилки і знову ставить повідомлення в чергу. Дане повідомлення буде послано в наступному слоті повідомлень, проте при цьому воно повинно пройти через процес арбітражу з іншими запланованими повідомленнями.

Звідси випливає одне із золотих правил розробки CAN – мереж – кожен ідентифікатор повинен бути унікальним. Тобто повідомлення з одним і тим же ідентифікатором не повинні посилатися двома різними вузлами. У цьому випадку можливе виникнення ситуації, при якій два повідомлення з однаковими ідентифікаторами почнуть передаватися одночасно, обидва будуть боротися за контроль над шиною і обидва виграють арбітраж, оскільки мають однакові ідентифікатори. Після виграшу арбітражу обидва повідомлення почнуть виставляти свої дані на шину. У якийсь момент дані виявляться різними, що призведе до виникнення помилки контролю біта. Обидва повідомлення будуть знову поставлені в чергу передачі, виграють арбітраж і знову викличуть помилку. Ця «циклічна ситуація» може повністю заблокувати мережу.

На рівні бітів в протоколі CAN реалізовано також метод вставки додаткового біта (бітстаффінг). Після кожної послідовності з п'яти бітів однакової полярності вставляється біт протилежної полярності (рисунок 4.79). Виключенням з цього правила являється кадр помилки, який представляє собою послідовність з шести домінантних бітів.

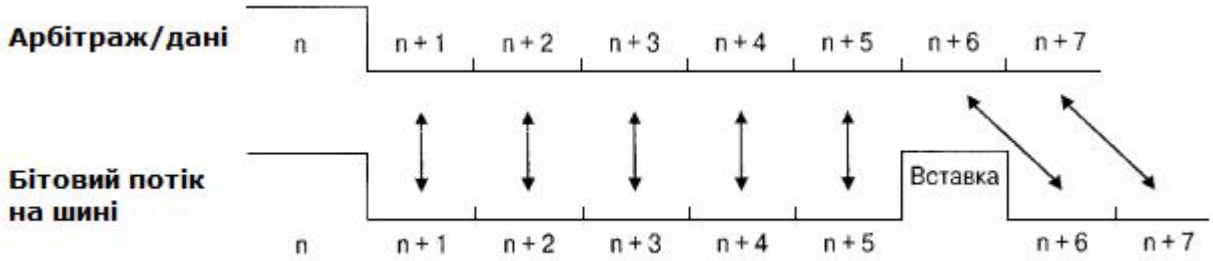


Рисунок 4.79 – Бітстаффінг

Цей метод дозволяє запобігти появі на шині постійних рівнів і забезпечує наявність в потоці бітів достатньої кількості переходів, що використовуються для ресинхронізації. Кадр помилки в протоколі CAN є простою послідовністю з шести домінантних бітів. Це дозволяє будь – якому контролеру CAN формувати на шині повідомлення про помилку відразу ж після її виявлення, не чекаючи кінця повідомлення. У кожному CAN – контролері є два лічильника помилок (рисунок 4.80).

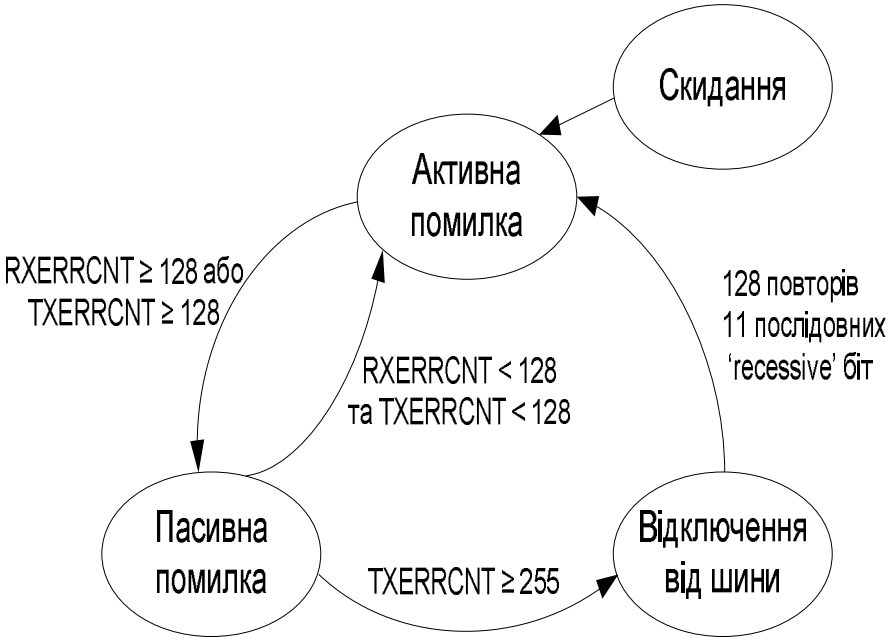


Рисунок 4.80 – Стан лічильників помилок

Цими лічильниками є лічильник помилок прийому і лічильник помилок передачі. Зміна стану зазначених лічильників відбувається при прийомі або передачі кадру помилки. Коли будь – який лічильник досягає значення 128, контролер CAN переходить зі стану ErrorActive (активна помилка) в стан ErrorPassive (пасивна помилка). В цьому режимі він продовжує відгукуватися на кадри помилки, однак при генерації кадру помилки він замість домінантних бітів виставляє на шину рецесивні. Якщо лічильник помилок передачі досягає значення 255, то контролер CAN переходить в режим bus – off (відключення від шини) і більше не приймає участі в обміні по шині. Для відновлення обміну необхідне втручання процесора, який повторно ініціалізує контролер і підключає його назад до шини. Обидва ці механізми призначено для того, щоб вузол при виникненні несправності не заблокував шину безперервної генерацією кадрів помилки.

CAN – контролер, який реалізовано в мікроконтролерах сімейства LPC2300, має кілька механізмів виявлення помилок. Перш за все, з глобального регістра стану CANGSR можна прочитати поточний стан лічильників помилок прийому і передачі.

Також в цьому регістрі є два прапорці помилок. Прапорець стану шини BS (таблиця 4.87) встановлюється при досягненні максимальної кількості помилок і

відключенні контролера CAN від шини. Другий прапорець стану помилки ES (таблиця 4.87), встановлюється при досягненні лічильниками помилок порогового значення. Це значення є довільною величиною, що записується в регістр CANEWL. За замовчуванням в даному регістрі знаходиться число 96. Як і регістри тактової синхронізації, регістр CANEWL можна змінювати тільки при знаходженні контролера в стані скидання. Крім того, регістр стану переривань CANICR надає діагностичну інформації для керування подіями на шині CAN.

У контролері CAN є такі джерела переривань:

1. Переривання закінчення передачі (по одному для кожного буфера).
2. Переривання закінчення прийому.
3. Попередження про помилку (Error Warning).
4. Переповнення даних (Data Overrun).
5. Пробудження (Wake Up).
6. Пасивна помилка (Error Passive).
7. Втрата арбітражу (Arbitration Lost).
8. Помилка на шині (Bus Error).
9. Готовність ідентифікатора (ID Ready).

4.12.2.4 Передача повідомлень CAN

В мікроконтролері кожен контролер CAN має декілька регістрів стану і регістр керування, а також три буфери передачі і два буфери прийому (таблиця 4.80).

Таблиця 4.80 – Регістри контролера CAN

Назва регістра	**Призначення**	**Доступ**
CANxMOD	Регістр керування режимом роботи	R/W
CANxCMR	Регістр керування	W
CANxGSR	Глобальний регістр стану	R
CANxICR	Регістр стану переривання	R
CANxIER	Регістр дозволу переривання	R/W
CANxBTR	Регістр тактової синхронізації	R/W
CANxEWL	Регістр порогового значення кількості помилок	R/W
CANxSR	Регістр стану	R

Щоб конфігурувати контролер, ми повинні занести потрібні значення в регістр тактової синхронізації, але цей регістр є захищеним, тобто в нього можна записувати лише тільки тоді, коли CAN – контролер находиться в стані скидання. Для переводу контролера в цей стан призначено 0 – й біт регістра режиму роботи (регістр CANMOD). Необхідно відзначити, що в регістр тактової синхронізації заносяться числа, які на одиницю менше обчислених значень. Таким чином, гарантується, що жоден з часових сегментів не буде дорівнювати нулю.

Після того, як CAN – контролер буде ініціалізовано, можна записувати дані в буфер передачі та передавати повідомлення. Кожний буфер передачі контролера може зберігати до 4–х 32 – розрядних слів. Два слова використовується для зберігання 8 байт інформації, а одне слово – для зберігання ідентифікатора повідомлення. Останніми для конфігурування є регістри інформації про кадр, який передається (таблиця 4.81).

В регістрі CANTFI зберігається значення поля DLC і прапорця RTR. Крім того, в ньому є прапорець формату кадру (FF), який визначає тип ідентифікатора повідомлення (11 або 29 – біт). Оскільки в наявності є три буфери, можна задавати внутрішній пріоритет кожного з них.

Таблиця 4.81 – Інформація про кадр, який передається

Назва регістра	Призначення	Доступ
CANxTFIn (n=1,2,3)	Інформація про кадр в буфері n, який передається	R/W
CANxTID (n=1,2,3)	Ідентифікатор кадру, який передається, в буфері n	R/W
CANxTDAn (n=1,2,3)	Байти 1...4, які передаються в буфері n	R/W
CANxTDBn (n=1,2,3)	Байти 5...8, які передаються в буфері n	R/W

Якщо планується передача даних одночасно з декількох буферів, то контролер CAN буде використовувати внутрішній арбітраж для визначення повідомлення, яке передається першим. Арбітраж може реалізовуватися двома способами. Якщо біт TPM регістра керування CANMOD скинуто, то для арбітражу буде використовуватися значення, що знаходиться в полі PRIO регістра інформації про кадр і першим буде передаватися вміст буфера, який має найменше значення поля PRIO. Після заповнення буфера даних, можна почати передачу інформації, встановивши біт TR (запит на передачу) регістра CANCMR.

На рисунку 4.81 наведено базовий алгоритм передачі інформації.

4.12.2.5 Прийом повідомлень

Після ініціалізації контролер CAN може приймати повідомлення в свій буфер прийому, структура якого схожа на структуру буфера передачі.

Регістр стану кадру, який приймається, CANRFS містить ту саму інформацію, що і регістр стану кадру, який передається. Але в регістрі CANRFS э два додаткових елементи: поле індексу ідентифікатора і прапорець BP (таблиця 4.95).

Регістри, які містять інформацію про повідомлення, які приймаються, описано в таблиці 4.82.

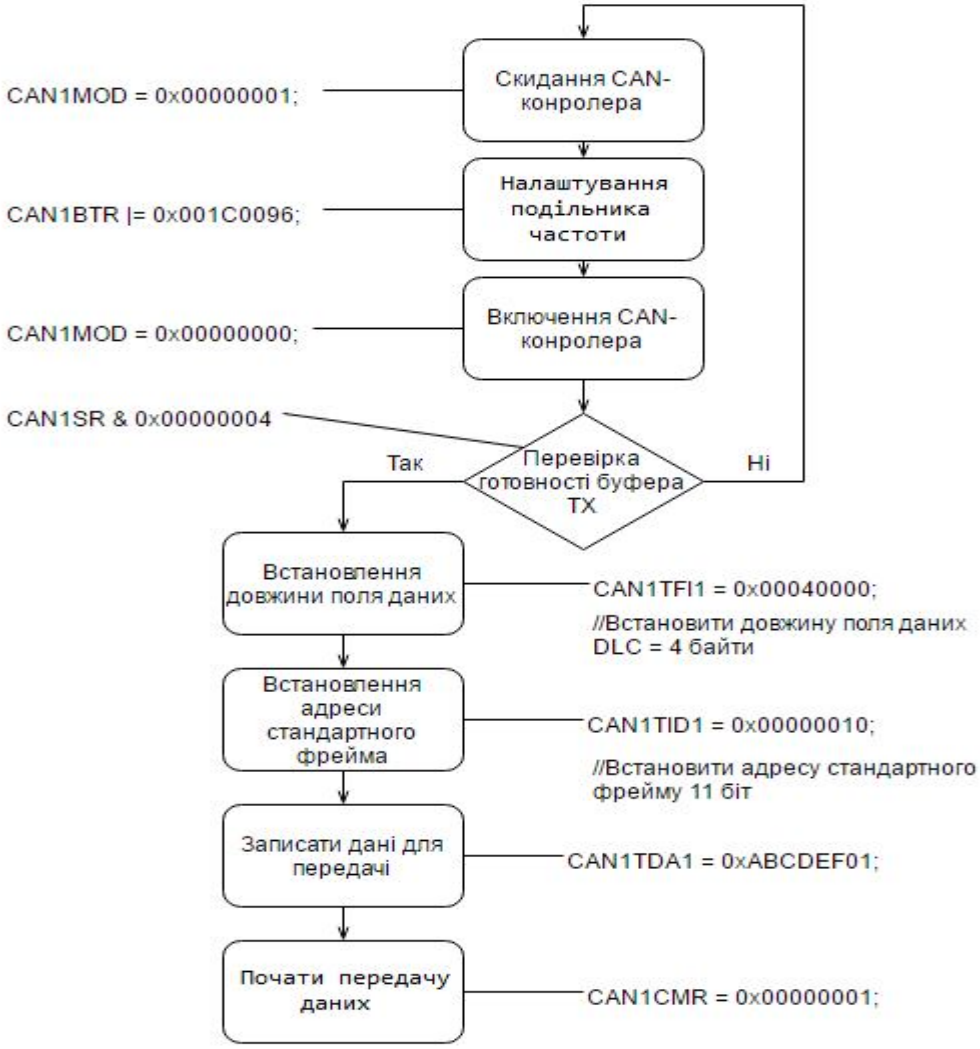


Рисунок 4.81 – Базовий алгоритм передачі інформації

Таблиця 4.82 – Інформація про кадр, який приймається

Назва регістра	Призначення	Доступ
CANxRFS	Статус кадру, який прийнято	R/W
CANxRID	Ідентифікатор повідомлення, яке прийнято	R/W
CANxRDA	Байти 1...4 повідомлення, яке прийнято	R/W
CANxRDB	Байти 5...8 повідомлення, яке прийнято	R/W

На рисунку 4.82 наведено базовий алгоритм прийому інформації. Під час надходження повідомлення спрацьовує відповідне переривання. В обробнику даного переривання можна прочитати повідомлення з одного із вхідних буферів.

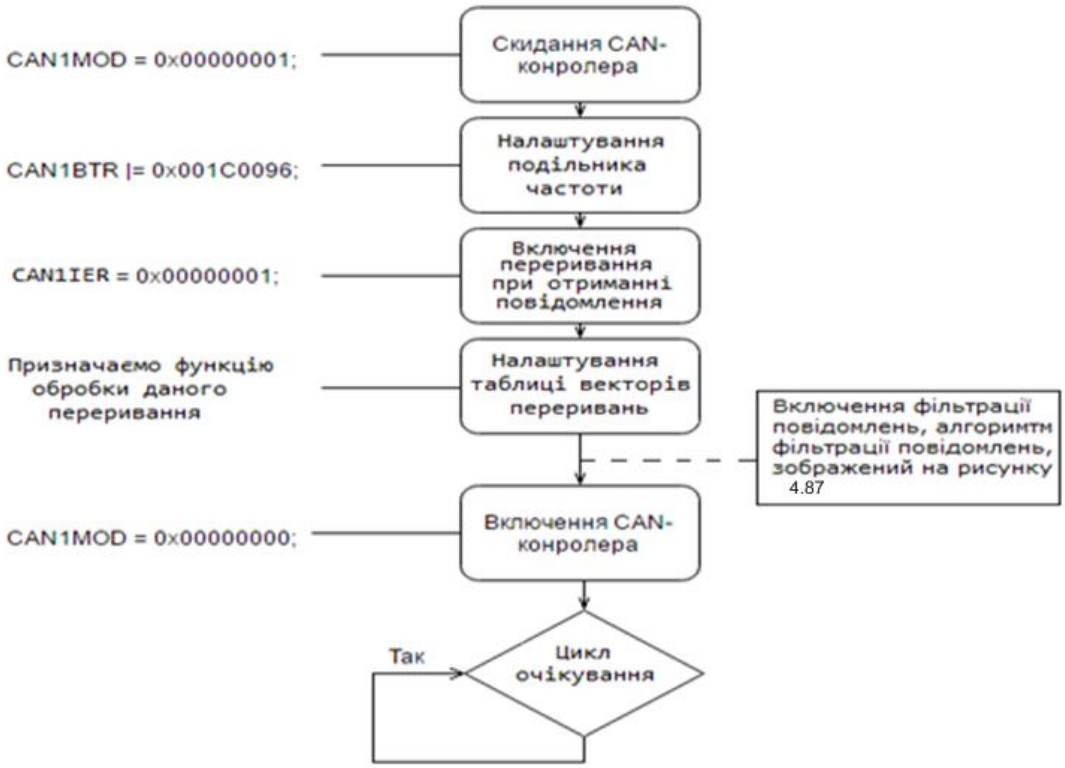


Рисунок 4.82 – Базовий алгоритм прийому інформації

4.12.2.6 Фільтрація повідомлень

Незважаючи на те що програма без фільтрації повідомлень буде працювати, в ній є два недоліки. По – перше, CAN – модуль буде приймати всі повідомлення, що передаються шиною.

У разі повністю завантаженої CAN – шини – це означає, що повідомлення будуть прийматися кожні 72 мкс. Так як в мікроконтролерах сімейства є 2 контролери CAN, ЦПП доведеться витратити багато часу тільки для керування шинами CAN. По – друге, після прийому повідомлення контролер CAN повинен прочитати і декодувати ідентифікатор повідомлення, щоб вирішити, що з ним робити. Для подолання цих проблем CAN – контролери сімейства LPC2300 мають розвинену схему фільтрації повідомлень. Фільтр повідомлень використовується як екран між контролером і CAN – шиною. Його можна запрограмувати на пропуск або блокування повідомлень з певними ідентифікаторами до того, як вони надійдуть в контролер CAN для обробки. Це запобігає завантаженню в буфер прийому контролера небажаних повідомлень і, відповідно, значно зменшує витрати ресурсів ЦПП. У складі фільтра повідомлень є блок ОЗП об'ємом 2Кбайт (512 х 32), який може

використовуватися в якості таблиці ідентифікаторів (рисунок 4.83). Завдяки наявності такої таблиці в буфер прийому контролера CAN можуть проходити як окремі повідомлення, так і повідомлення в заданих діапазонах.

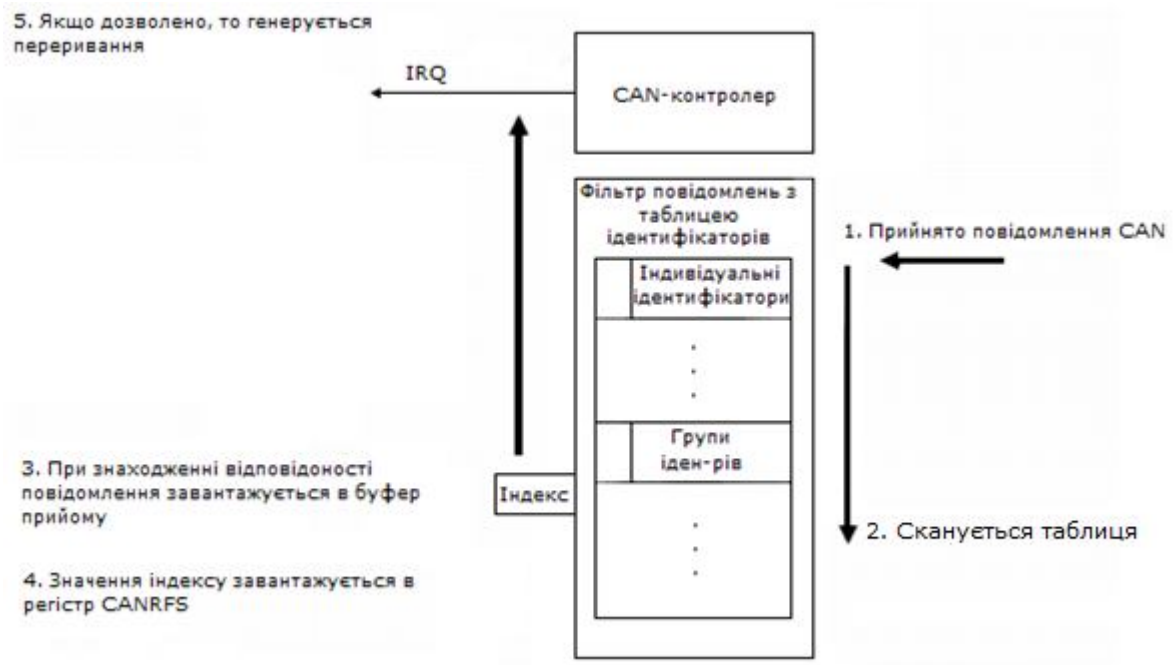


Рисунок 4.83– Фільтрація повідомлень

Після проходження повідомлення через фільтр йому присвоюється певний індекс – ціле число, яке відповідає зміщенню ідентифікатора повідомлення в таблиці фільтра повідомлення. Значення індексу зберігається в регістрі стану прийнятого кадру. Таким чином, щоб визначити, яке саме повідомлення було прийнято, замість декодування самого ідентифікатора набагато швидше і простіше використовувати його індекс.

Фільтр повідомлень, крім того, може працювати в режимі FullCAN. У цьому режимі ОЗП контролера може використовуватися в якості додаткових буферів прийому, в яких будуть зберігатися отримані дані. ЦПП буде читати цю інформацію по мірі необхідності.

4.12.2.7 Конфігурування фільтра повідомлень

У складі фільтра повідомлень наявна ОЗП пам'ять, об'ємом 2Кб, яка використовується в якості таблиці ідентифікаторів. Фільтр виконує сканування ОЗП при прийомі будь – якого повідомлення на предмет відповідності

ідентифікатором адресі або діапазону адрес і визначає, чи дозволений його прийом.

Перш ніж приступити до конфігурації фільтра повідомлень, його необхідно зупинити. Ця операція виконується встановленням в регістрі CAN_AFMR біта AccOff, при умові, що біт AccBP=0. Якщо при такій конфігурації запустити CAN – контролер, він буде приймати всі повідомлення, що з'являються на шині.

Якщо прийом дозволений, повідомлення передається на вхідні буфери контролерів CAN, які встановлюють прапорці прийому та генерують переривання. Після заборони фільтрації можна почати конфігурування фільтра, який складається з чотирьох різних таблиць:

- таблиці фільтрації за індивідуальними базовими 11 – бітними ідентифікаторами;
- таблиці фільтрації за групами базових11 – бітних ідентифікаторів;
- таблиці фільтрації за індивідуальними розширеними 29 – бітними ідентифікаторами;
- таблиці фільтрації за групами розширених 29 – бітних ідентифікаторів.

ОЗП відображено в карту пам'яті мікроконтролера, починаючи з адреси 0xE0038000. Таблиці можуть мати різний розмір, тому передбачено опис відповідних таблиць через регістри початкової і кінцевої адреси. Під час налаштування фільтра використовуються регістри, які наведено в таблиці 4.83.

Таблиці повинні розташовуватися в пам'яті послідовно. Адресу останньої комірки ОЗП необхідно записати в регістр адреси кінця таблиць CAN_EOT (ENDofTable). Після цього необхідно включити фільтр повідомлень скиданням біта AccOff та встановленням біта AccBP в регістрі CAN_AFMR.

Розглянемо формати таблиць фільтрації.

Таблиця індивідуальних стандартних ідентифікаторів (рисунок 4.84) дозволяє визначити окремі 11 – бітні ідентифікатори, які будуть пропускатися фільтром повідомлень. Кожен елемент таблиці займає два байти. 11 перших бітів містять ідентифікатор, наступним йде біт, за допомогою, якого можна динамічно

дозволити/заборонити даний елемент фільтра. Старші три біти встановлюють відповідність між даним елементом фільтра і конкретним CAN – контролером.

Таблиця 4.83 – Регістри налаштування фільтра

Назва регістра	Призначення
CAN_AFMR	Регістр керування фільтром повідомлень
CAN_SFF_SA	Регістр початкової адреси таблиці фільтрації за індивідуальними базовими ідентифікаторами
CAN_SFF_GRP_SA	Регістр початкової адреси таблиці фільтрації за групою базових ідентифікаторів
CAN_EFF_SA	Регістр початкової адреси таблиці фільтрації за індивідуальними розширеними ідентифікаторами
CAN_EFF_GRP_SA	Регістр початкової адреси таблиці фільтрації за групою розширених ідентифікаторів
CAN_EOT	Регістр кінцевої адреси таблиці фільтрації

Рисунок 4.84 – Формат таблиці індивідуальних стандартних ідентифікаторів

У таблиці груп стандартних ідентифікаторів (рисунок 4.85) застосовується той же формат, що в таблиці індивідуальних стандартних ідентифікаторів, але в кожному елементі таблиці використовується два визначення, що встановлюють нижню і верхню межу діапазону ідентифікаторів повідомлень, що пропускаються фільтром.

31 29	28 27	26 16	15 13	12 11	10 0
№ контролера	Заборонити	Невикор.	Найменший ідентифікатор групи	№ контролера	Заборонити	Невикор.	Найбільший ідентифікатор групи

Рисунок 4.85 – Формат таблиці груп стандартних ідентифікаторів

У таблиці індивідуальних розширених ідентифікаторів для кожного елемента таблиці використовується 4 байти, як показано на рисунку 4.86. Перші 29 біт визначають ідентифікатор повідомлення, що пропускається через фільтр, а старші 3 біти встановлюють відповідність між даним елементом фільтра і конкретним CAN – контролером. У кожному елементі таблиці груп розширених ідентифікаторів використовується два слова, що визначають початкове і кінцеве значення ідентифікаторів, аналогічно таблиці груп стандартних ідентифікаторів (рисунок 4.85).

31 29	28 0
№ контролера	Ідентифікатор

Рисунок 4.86– Формат таблиці індивідуальних розширених ідентифікаторів

На рисунку 4.87 наведено базовий алгоритм фільтрації повідомлень.

4.12.2.8 Режим FULLCAN

CAN – контролери, які реалізовано в мікроконтролерах LPC2300 можуть також працювати в більш розвиненому режимі FullCAN, в якому частина ОЗП фільтрів прийому використовується під буфери прийому. У цьому режимі створюється додаткова таблиця ідентифікаторів FullCANID, а частина ОЗП фільтра повідомлень виділяється під окремі буфери (рисунок 4.88). Таблиця FullCANID розташовується на початку блоку ОЗП фільтра повідомлень, а закінчується за адресою, яку визначено в регістрі стандартного формату кадру.

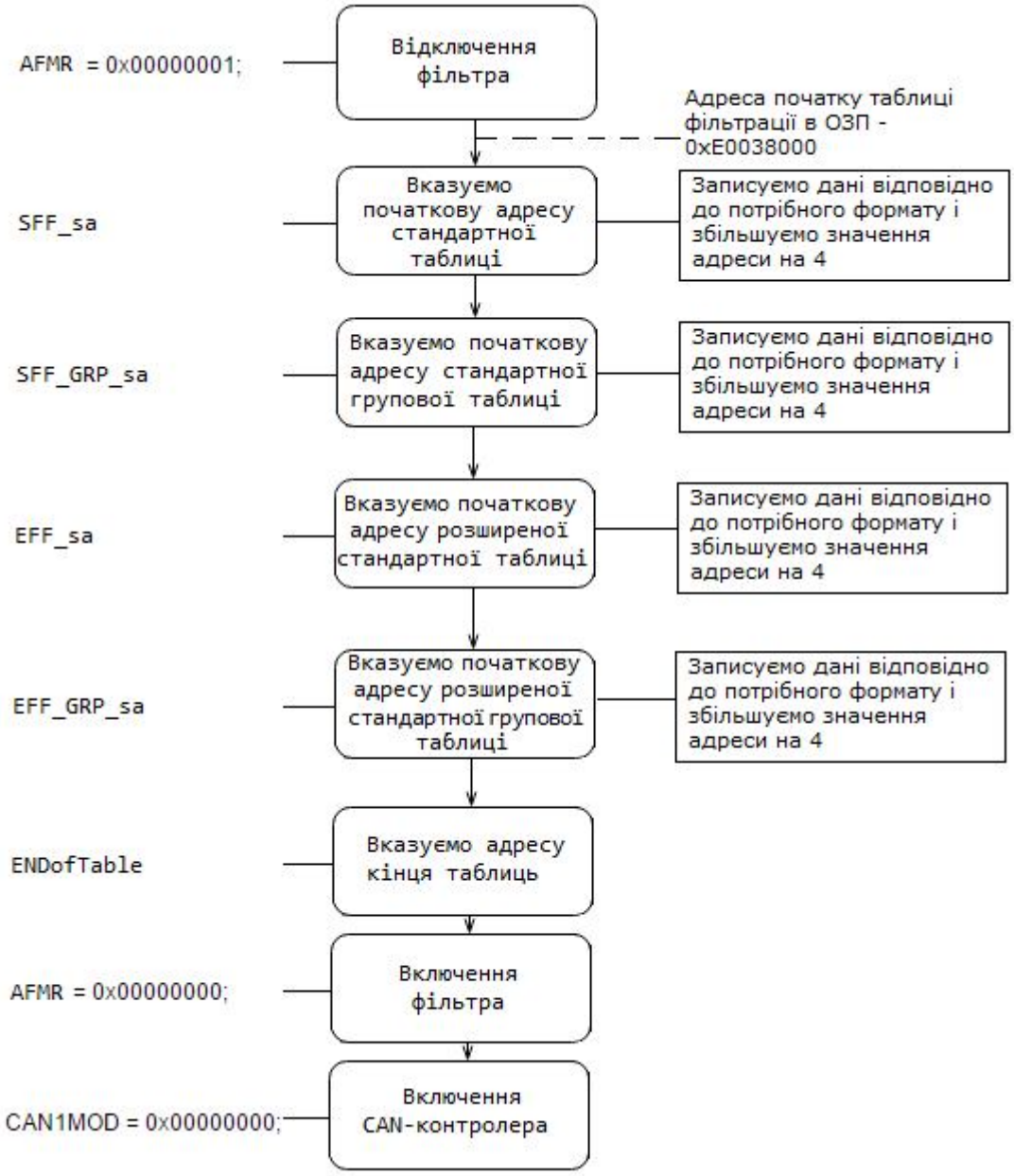


Рисунок 4.87 – Базовий алгоритм фільтрації повідомлень

Під час отримання повідомлення, ідентифікатор якого є в таблиці FullCANID, повідомлення автоматично зберігається в окремому буфері. По суті це означає, що ви можете створити область «віртуальної» пам'яті, загальної для всієї мережі. Вміст цієї пам'яті буде оновлюватися при кожній передачі коректного повідомлення CAN у всіх вузлах, яким потрібні дані зазначеного повідомлення.

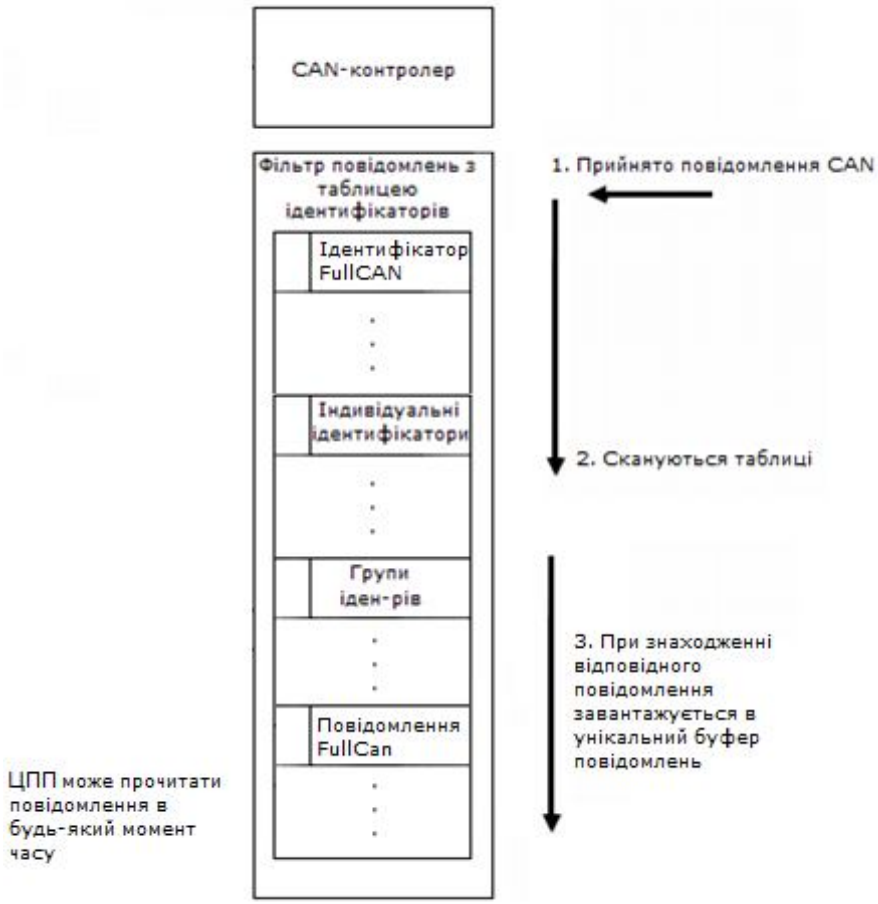


Рисунок 4.88 – Режим роботи FullCan

У режимі FullCAN фільтр повідомлення конфігурується так само, як і в базовому режимі. Включається режим FullCAN встановленням біта eFCAN в регістрі керування фільтром повідомлень AFMR. Однак перед цим треба переконатися, що в ОЗП фільтра досить місця для буферів прийому FullCAN.

Спочатку визначимо обсяг пам'яті, який необхідно для розміщення таблиці FullCAN:

$$SFF_sa = 2 * (\text{к} - \text{сть повідомлень } FullCAN),$$

при цьому значення SFF_sa повинно бути кратне чотирьом.

Потім треба визначити обсяг пам'яті, який необхідно для розміщення буферів прийому FullCAN:

EoT ≤0x800 – (6 x SFF_sa).

В результаті при прийомі повідомлення, ідентифікатор якого міститься в таблиці фільтра, воно буде прочитано з приймального буфера контролера CAN і збережено в ОЗП фільтра за адресою:

$$EoT + (\text{індекс в таблиці } FullCAN \text{ x } 12).$$

Повідомлення зберігається в 12 – байтному буфері (рисунок 4.89).

FF (Формат кадру)	RTR		SEM (семафор)		DLC		ID

Рисунок 4.89 – Буфер прийому FullCAN

Поле семафора (2 – біти) використовується для того, щоб запобігти читання процесором буфера повідомлень під час оновлення даних CAN – контролером. При отриманні нового повідомлення, яке відповідає вимогам фільтра, CAN – контролер спочатку зберігає стан кадру і заносить в семафор значення, сигналізуючи оновлення буфера (01). Дані повідомлення заносяться CAN – контролером в буфер повідомлення. Після запису всіх байтів повідомлення CAN – контролер заносить в семафор значення, сигналізуючи завершення оновлення буфера (11). Процесор, перш ніж прочитати дані повідомлення, повинен перевірити значення семафора. Коли ЦПП починає читання з FullCAN – буфера, ЦПП повинен спочатку скинути обидва біти семафора, а потім прочитати нове повідомлення. Після зчитування даних з буферу прийому слід перед передачею даних в програмі повторно перевірити стан бітів семафора. Якщо вони не дорівнюють нулю, значить, під час читання даних з буфера було прийнято нове повідомлення. Це означає, що попередні дані могли бути пошкоджені новим повідомленням і їх слід ігнорувати, а потім повторно прочитати буфер повідомлення, як описано вище.

4.12.3 Керуючі регістри контролерів CAN1 і CAN2 у складі мікроконтролера LPC2378

4.12.3.1 Регістр CANMOD

CANMOD це регістр керування режимом роботи, таблиця 4.84.

Адреса:

CAN1MOD – 0xE004 4000;

CAN2MOD – 0xE004 8000.

Таблиця 4.84 – Опис розрядів регістра CANMOD

Біт	Назва	Функція
0	RM	Reset Mode – режим перезавантаження. 0: CAN – контролер працює, деякі регістри недоступні для запису; 1: включення режиму перезавантаження, CAN – контролер недоступний, в регістри, доступні для запису, можна записувати.
1	LOW	Listen Only Mode – режим прослуховування. 0: CAN – контролер підтверджує, що повідомлення було успішно прийнято; 1: включення режиму прослуховування, контролер не дає підтверджень того, що повідомлення було успішно прийнято. Повідомлення не можуть посилатися, контролер знаходиться в стані пасивної помилки.
2	STM	Self Test Mode – режим самостійного тестування. 0: передане повідомлення повинно бути підтверджено, щоб передача вважалася успішною; 1: включення режиму самостійного тестування – CAN – контролер буде вважати передане повідомлення успішним, якщо повідомлення навіть не було підтверджено. Цей режим можна використовувати разом з встановленням біта SRR в регістрі CANCMR.

Продовження таблиці 4.84

Біт	Назва	Функція
3	TPM	0: черговість трьох буферів передачі залежить від їх CAN – ідентифікаторів; 1: черговість трьох буферів передачі залежить від відповідних полів (PRIO в регістрі CANTFI).
4	SM	Sleep Mode – режим сну. 0: нормальний режим роботи; 1: включення режиму сну, коли немає запитів на переривання і немає активності на шині.
5	RPM	Reverse Polarity Mode – режим зміни полярності сигналу. 0: на виводі CAN – шини (RX і TX) під домінантним сигналом мається на увазі логічний нуль; 1: включення режиму зміни полярності сигналу – під домінантним сигналом мається на увазі логічна одиниця.
6	–	Зарезервовані біти.
7	TM	Test Mode – режим тестування. 0: нормальний режим роботи; 1: включення режиму тестування, Rx замикається на Tx.

4.12.3.2 Регістр CANCMR

CANCMR це регістр керування, таблиця 4.85.

Адреса:

CAN1CMR – 0xE004 4004;

CAN2CMR – 0xE004 8004.

Таблиця 4.85 – Опис розрядів регістра CANCMR

Біт	Назва	Функція
0	TR	Transmission Request – запит передачі. 0: немає запиту на передачу; 1: повідомлення, вже записано в регістри CANTFI/CANTID/CANTDA/CANTDB і поміщається в чергу на передачу.
1	AT	Abort Transmission – припинення передачі. 0: не переривати передачу; 1: невиконаний запит на передачу скасовується.

Продовження таблиці 4.85

Біт	Назва	Функція
2	RRB	Release Receive Buffer – очищення буферів прийому. 0: не очищати буфери прийому; 1: регістри CANRFS/CANRID/CANRDA/CANRDB очищаються, і в них може бути записано наступний прийнятий кадр, якщо наступний прийнятий кадр не доступний, то виконання цієї команди очищає біт RBS в регістрі CANSR.
3	CDO	Clear Data Overrun – очищення переповнення даних. 0: не очищати переповнені дані; 1: очищення біта DOS в регістрі CANSR.
4	SRR	Self Reception Request – автоматичний запит прийому. 0: відключити автоматичний запит прийому; 1: повідомлення, яке уже записано в регістри CANTFI/CANTID/CANTDA/CANTDB, поміщається в чергу на передачу. На відміну від біта TR, в даному випадку приймач є доступним під час передачі і може приймати повідомлення.
5	STB1	1: вибір першого буфера для передачі (Select Tx Buffer1).
6	STB2	1: вибір другого буфера для передачі (Select Tx Buffer2).
7	STB3	1: вибір третього буфера для передачі (Select Tx Buffer3).

4.12.3.3 Регістр CANGSR

CANGSR це глобальний регістр стану, таблиця 4.86.

Адреса:

CAN1GSR – 0xE004 4008;

CAN2GSR – 0xE004 8008.

Таблиця 4.86 – Опис розрядів регістра CANGSR

Біт	Назва	Функція
0	RBS	Receive Buffer Status – статус буфера прийому. 0: немає доступних повідомлень; 1: прийняте повідомлення є в регістрах CANRFS/CANRID/CANRDA/CANRDB. Цей біт очищається полем RRB в регістрі CANCMR, при умові що наступне прийняте повідомлення не доступне.

Продовження таблиці 4.86

Біт	Назва	Функція
1	DOS	Data Overrun Status – переповнення даних. 0: ніякого переповнення даних не відбулося з моменту останнього виконання команди CDO (очищення переповнення даних, регістр CANCMR) або після скидання; 1: повідомлення було втрачено, тому що попереднє повідомлення не було прочитано і відповідні регістри не дуже швидко звільнилися від нього.
2	TBS	Transmit Buffer Status – статус буфера передачі. 0 – як мінімум одне повідомлення знаходиться в черзі для передачі і все ще не відіслано; 1: всі повідомлення передані для даного CAN – контролера (з усіх трьох буферів передачі) і можна програмно записувати в регістри будь – якого буфера CANTFI/CANTID/CANTDA/CANTDB;
3	TCS	Transmit Complete Status – статус виконаної передачі. 0: як мінімум одне повідомлення не було успішно відправлено; 1: всі повідомлення були успішно передані;
4	RS	Receive Status – статус прийому. 0: CAN – контролер знаходиться в стані очікування; 1: CAN – контролер приймає повідомлення.
5	TS	Transmit Status – статус передачі. 0: CAN – контролер знаходиться в стані очікування; 1: CAN – контролер відправляє повідомлення. Детальніше в регістрі CANSR.
6	ES	Error Status – статус помилки. 0: значення помилок лічильників менше граничного значення (регістр CANEWL); 1: один або обидва лічильника помилок прийому/передачі досягли граничного значення (регістр CANEWL).
7	BS	Bus Status – статус шини. 0: CAN – контролер знаходиться в режимі bus – on (активність на шині); 1: значення лічильника помилок досягло 255, CAN – контролер переходить в режими bus – off (відключення від шини–шина не працює).
23:16	RXERR	Rx Error Counter – поточне значення лічильника помилок прийому.

Продовження таблиці 4.86

Біт	Назва	Функція
31:24	TXERR	Tx Error Counter – поточне значення лічильника помилок передачі.

4.12.3.4 Регістр CANICR

CANICR це регістр стану переривання, таблиця 4.87.

Адреса:

CAN1ICR – 0xE004 400C;

CAN2ICR – 0xE004 800C.

Таблиця 4.87 – Опис розрядів регістра CANICR

Біт	Назва	Функція
0	RI	Receive Interrupt – переривання за прийомом. 1: коли біт RBS в CANxSR і біт RIE в CANxIER рівні 1, показуючи, що нове повідомлення було отримано і знаходиться в приймальному буфері.
1	TI1	Transmit Interrupt 1 – переривання за передачею з буфера 1. 1: коли біт TBS1 в CANxSR змінився з 0 в 1 (коли повідомлення з TXB1 було успішно передано або перервано), вказуючи, що буфер передачі 1 доступний, а біт TIE1 в CANxIER дорівнює 1.
2	EI	Error Warning Interrupt – переривання за попередженням про помилку. 1: при кожній зміні (встановити або очистити) біта ES або BS в регістрі CANxSR.
3	DOI	Data Overrun Interrupt – переривання за переповненням даних. Цей біт встановлюється, коли біт DOS в CANxSR змінюється з 0 в 1, а біт DOIE в CANxIER дорівнює 1. Це відбувається, коли повідомлення втрачається, оскільки попереднє повідомлення не було прочитано з буфера.
4	WUI	Wake – Up Interrupt – переривання за «пробудженням». 1: якщо CAN – контролер знаходиться у режимі сну і виявлена активність на CAN – шині, та біт WUIE в CANxIER дорівнює 1.
5	EPI	Error Passive Interrupt – переривання за пасивною помилкою. 1: якщо біт EPIE в CANxIER дорівнює 1, і контролер перемикається між пасивною помилкою і активний режимом помилки в напрямку першого режиму.

Продовження таблиці 4.87

Біт	Назва	Функція
6	ALI	Arbitration Lost Interrupt – переривання за втратою арбітражу. 1: якщо біт ALIE в CANxIER дорівнює 1, а CAN – контролер втрачає арбітраж при спробі передачі.
7	BEI	Bus Error Interrupt – переривання помилки на шині. 1: якщо біт BEIE в CANxIER дорівнює 1 і CAN – контролер виявляє помилку на шині.
8	IDI	ID Ready Interrupt – переривання за готовністю ідентифікатора. 1: якщо біт IDIE в CANxIER дорівнює 1 і повідомлення було успішно передано або перервано.
9	TI2	Аналог TI1, але для другого буфера.
10	TI3	Аналог TI2, але для третього буфера.
15 – 11	–	Зарезервовані біти.

4.12.3.5 Регістр CANIER

CANIER це регістр дозволу переривань, таблиця 4.88.

Адреса:

CAN1IER – 0xE004 4010;

CAN2IER – 0xE004 8010.

Таблиця 4.88 – Опис розрядів регістра CANIER

Біт	Назва	Функція
0	RIE	1: дозвіл на переривання, коли стан приймального буфера є "заповненим".
1	TIE1	1: дозвіл переривання при успішній передачі з буфера 1.
2	EIE	1: дозвіл на переривання, при виникненні помилка ES або зміні статусу шини BS (див. Status Register).
3	DOIE	1: дозвіл на переривання при втраті даних через «набігання» (при недостатній швидкості роботи приймача).

Продовження таблиці 4.88

Біт	Назва	Функція
4	WUIE	1: дозвіл на переривання при зміні сплячого режиму CAN – контролера на активний.
5	EPIE	1: дозвіл на переривання при зміні з активної помилки в пасивну помилку або навпаки.
6	ALIE	1: дозвіл на переривання при втраті CAN – контролером арбітражу на шині.
7	BEIE	1: дозвіл на переривання при виявленні помилка на CAN – шині.
8	IDIE	1: дозвіл на переривання при отриманні CAN – ідентифікатора.
9	TIE2	Аналогічно до TIE1, але для 2 – го буфера передачі.
10	TIE3	Аналогічно до TIE1, але для 3 – го буфера передачі.
31…11	–	Зарезервовані біти.

4.12.3.6 Регістр CANEWL

CANEWL це регістр граничного значення кількості помилок, таблиця 4.89.

Адреса:

CAN1EWL – 0xE004 4018;

CAN2EWL – 0xE004 8018.

Таблиця 4.89 – Опис розрядів регістра CANEWL

Біт	Назва	Функція
7:0	EWL	В ході операцій CAN – інтерфейсу значення EWL постійно порівнюється з лічильниками помилок передачі і прийому. Якщо вміст будь – якого з цих лічильників досягає значення EWL (за замовчуванням стоїть 96), то встановлюється біт стану помилки (ES) в регістрі CANSR.
31:8	–	Зарезервовані біти.

4.12.3.7 Регістр CANSR

CANSR це регістр стану, таблиця 4.90.

Адреса:

CAN1SR – 0xE004 401C;

CAN2SR – 0xE004 801C.

Таблиця 4.90 – Опис розрядів регістра CANSR

Біт	Назва	Функція
0, 8, 16	RBS	Ці біти ідентичні біту RBS в регістрі CANGSR
1, 9, 17	DOS	Ці біти ідентичні біту DOS в регістрі CANGSR.
2, 10, 18	TBSn	Transmit Buffer Status – статус буфера передачі. 0: не можна записувати в регістри n буфера передачі. 1: можна зробити програмний запис повідомлення в регістри обраного буфера передачі CANTFI/CANTID/CANTDA/CANTDB;
3, 11, 19	TCSn	Transmit Complete Status – статус виконаної передачі. 0: попередній запит передачі для обраного буфера передачі не виконаний; 1: попередній запит передачі для обраного буфера передачі успішно виконаний.
4, 12, 20	RS	Ці біти ідентичні біту RS в регістрі CANGSR.
5, 13, 21	TSn	Transmit Status – статус передачі. 0: CAN – контролер не відправляє повідомлення з n – го буфера передачі; 1: CAN – контролер відправляє повідомлення з n – го буфера передачі.
6, 14, 22	ES	Ці біти ідентичні біту ES в регістрі CANGSR.
7, 15, 23	BS	Ці біти ідентичні біту BS в регістрі CANGSR.

4.12.3.8 Регістр CANRDA

Регістр CANRDA містить від одного до чотирьох байтів отриманого повідомлення, таблиця 4.91.

Адреса:

CAN1RDA – 0xE004 4028;

CAN2RDA – 0xE004 8028.

Таблиця 4.91 – Опис розрядів регістра CANRDA

Біт	Назва	Функція
7:0	Data1	Якщо поле DLC регістра CANRFS >= 0001, то це поле містить перший байт даних отриманого повідомлення.
15:8	Data2	Якщо поле DLC регістра CANRFS >= 0010, то це поле містить другий байт даних отриманого повідомлення.
23:16	Data3	Якщо поле DLC регістра CANRFS >= 0011, то це поле містить третій байт даних отриманого повідомлення.
31:24	Data4	Якщо поле DLC регістра CANRFS >= 0100, то це поле містить четвертий байт даних отриманого повідомлення.

4.12.3.9 Регістр CANRDB

Даний регістр аналогічний до регістра CANRDA, тільки для 5 … 8 байтів отриманого повідомлення.

Адреса:

CAN1RDB – 0xE004 402C;

CAN2RDB – 0xE004 802C.

4.12.3.10 Регістр CANTDA

Регістр CANTDA містить від одного до чотирьох байтів повідомлення, що передається, таблиця 4.92.

Адреса:

Для першого буфера:

CAN1TDA1 – 0xE004 4038;

CAN2TDA1 – 0xE004 8038.

Для другого буфера:

CAN1TDA2 – 0xE004 4048;

CAN2TDA2 – 0xE004 8048.

Для третього буфера:

CAN1TDA3 – 0xE004 4058;

CAN2TDA3 – 0xE004 8058.

Таблиця 4.92 – Опис розрядів регістра CANTDA

Біт	Назва	Функція
7:0	Data1	Якщо поле DLC регістра TFIn >= 0001, то це поле містить перший байт даних повідомлення, що передається.
15:8	Data2	Якщо поле DLC регістра TFIn >= 0010, то це поле містить другий байт даних повідомлення, що передається.
23:16	Data3	Якщо поле DLC регістра TFIn >= 0011, то це поле містить третій байт даних повідомлення, що передається.
31:24	Data4	Якщо поле DLC регістра TFIn >= 0100, то це поле містить четвертий байт даних повідомлення, що передається.

4.12.3.11 Регістр CANTDB

Регістр CANTDB аналогічний до регістра CANTDA, тільки для 5 … 8 байтів повідомлення, що передаються.

Адреса:

Для першого буфера:

CAN1TDB1 – 0xE004 403C;

CAN2TDB1 – 0xE004 803C.

Для другого буфера:

CAN1TDB2 – 0xE004 404C;

CAN2TDB2 – 0xE004 804C.

Для третього буфера:

CAN1TDB3 – 0xE004 405C;

CAN2TDB3 – 0xE004 805C.

4.12.3.12 Регістр ідентифікатора CANTID

Регістр ідентифікатора, що передається: для CAN–ідентифікатора розміром 11–біт описано в таблиці 4.93, для CAN–ідентифікатора розміром 29–біт, описано в таблиці 4.94.

Адреса:

Для першого буфера:

CAN1TID1 – 0xE004 4034;

CAN2TID1 – 0xE004 8034.

Для другого буфера:

CAN1TID2 – 0xE004 4044;

CAN2TID2 – 0xE004 8044.

Для третього буфера:

CAN1TID3 – 0xE004 4054;

CAN2TID3 – 0xE004 8054.

При стандартному форматі повідомлення, CAN – ідентифікатор має розмір 11 – бітів, при розширеному форматі кадру повідомлення, CAN – ідентифікатор має розмір 29 – бітів. Ідентифікатор діє як ім'я повідомлення, що використовуються в приймачі для фільтрації, а також визначає пріоритет доступу до шини під час арбітражу.

Таблиця 4.93 – Опис розрядів регістра CANTID, коли CAN – ідентифікатор має розмір 11 – бітів

Біт	Назва	Функція
10:0	ID	Ідентифікатор повідомлення має розмір 11 – бітів при FF = 0 в регістрі CANTFI (CANRFS).
31:11	–	Зарезервовані біти.

Таблиця 4.94 – Опис розрядів регістра CANMOD, коли CAN – ідентифікатор має розмір 29 – бітів

Біт	Назва	Функція
28:0	ID	Ідентифікатор повідомлення має розмір 29 – бітів при FF = 1 в регістрі CANTFI (CANRFS).
29:31	–	Зарезервовані біти.

4.12.3.13 Регістр CANRID

Даний регістр є аналогом CANTID, але для прийнятих повідомлень.

Адреса:

CAN1RID – 0xE004 4024;

CAN2RID – 0xE004 8024.

В залежності від поля FF в регістрі CANRFS можемо змінювати довжину ідентифікатора: при FF = 0 розмір ID 11–бітів, при FF = 1 розмір ID 29 – бітів.

1.3.14 Регістр CANRFS

Регістр визначає характеристики прийнятого повідомлення, таблиця 4.95.

Адреса:

CAN1RFS – 0xE004 4020;

CAN2RFS – 0xE004 8020.

Доступний для запису тільки при RM (CANMOD) = 1.

Таблиця 4.95 – Опис розрядів регістра CANRFS

Біт	Назва	Функція
9:0	ID Index	Містить номер запису в таблиці фільтрації, який співпадає з прийнятим ідентифікатором повідомлення.
10	BP	1: то поточне повідомлення було отримано в AF Bypass режимі (без фільтрації, поле ID Index в даному випадку не має сенсу).
15:11	–	Зарезервовані біти.
19:16	DLC	Data Length Code – код довжини даних. Коли біт RTR = 0, то в регістрах CANRDA і CANRDB доступно для читання наступну кількість байтів: 0000… 0111 = 0…7байт, 1000 … 1111 = 8байт. Коли RTR = 1, значення поля DLC визначає, яку кількість байтів даних потрібно передати назад віддаленому користувачу з тим же ідентифікатором, що і у прийнятого повідомлення.
29:20	–	Зарезервовані біти.
30	RTR	Remote Transmission Request – запит віддаленої передачі. 0: прийнято поточний кадр даних, байти якого можна прочитати з регістра CANRDA/CANRDB при умові, що DLC зберігає відмінне від нуля значення; 1: це кадр віддаленого запиту даних і DLC в даному випадку визначає кількість даних, які запитують для передачі, віддаленому користувачу з тим же ідентифікатором, що і у прийнятого повідомлення.
31	FF	Формат ідентифікатора. 0: CAN – ідентифікатор має розмір 11 – біт; 1: CAN – ідентифікатор має розмір 29 – біт.

4.12.3.14 Регістр CANTFI

Регістр CANTFI містить інформацію про кадр в буфері n, який передається, таблиця 4.96.

Адреса:

Для першого буфера:

CAN1TFI1 – 0xE004 4030;

CAN2TFI1 – 0xE004 8030.

Для другого буфера:

CAN1TFI2 – 0xE004 4040;

CAN2TFI2 – 0xE004 8040.

Для третього буфера:

CAN1TFI3 – 0xE004 4050;

CAN2TFI3 – 0xE004 8050.

Якщо біт TBS в регістрі CANSR дорівнює 1 в, то можна записувати інформацію в цей регістр для визначення формату кадру, що передається (для вибраного буфера передачі)

Таблиця 4.96 – Опис розрядів регістра CANTFI

Біт	Назва	Функція
7:0	PRIO	Якщо біт TPM в регістрі CANMOD дорівнює 1, то можна вибрати чергу буферів передачі за допомогою цього поля. Чим менше значення цього поля, тим більшим буде пріорітет буфера передачі.
15:8	–	Зарезервовані біти.
19:16	DLC	Аналогічний полю DLC в регістрі CANRFS.
29:20	–	Зарезервовані біти.
30	RTR	Аналогічний полю RTR в регістрі CANRFS.
31	FF	Аналогічний полю FF в регістрі CANRFS.

4.12.3.15 Регістр CANBTR

Регістр керує тактуванням CAN – контролера, таблиця 4.97.

Адреса:

CAN1BTR – 0xE004 4014

CAN2BTR – 0xE004 8014

Таблиця 4.97– Опис розрядів регістра CANBTR

Біт	Назва	Функція
0:9	BRP	Baud Rate Prescaler – подільник частоти. Частота шини CAN = частота Pclk/(BRP + 1).
13:10	–	Зарезервовані біти.
15:14	SJW	Synchronization Jump Width – ширина стрибка синхронізації. Потрібна при втраті даних через «набігання» (при недостатній швидкості роботи приймача).
19:16	TSEG1	Сегмент фази 1 (PHASE_SEG1).
22:20	TSEG2	Сегмент фази 2 (PHASE_SEG2).
23	SAM	0: значення шини зчитується один раз (рекомендується для високошвидкісних шин); 1: значення на шини зчитується тричі (рекомендується для шин з низькою і середньою швидкістю передачі).
31:24	–	Зарезервовані біти.

4.12.4 Регістри фільтрації повідомлень

4.12.4.1 Регістр CAN_AFMR

CAN_AFMR це регістр керування фільтром повідомлень, таблиця 4.98.

Адреса – 0xE003 C000.

Таблиця 4.98 – Опис розрядів регістра CAN_AFMR

Біт	Назва	Функція
0	AccOff	1: прийомний фільтр відключений, усі ідентифікатори повідомлень ігноруються; 0: прийомний фільтр включений.
1	AccBP	0: необхідно також, щоб біт AccOff знаходиться у встановленому стані, тоді прийомний фільтр відключений, усі ідентифікатори повідомлень ігноруються; 1: Усі вхідні повідомлення приймаються через дозволені CAN – контролери. Призначена для користувача програма повинна встановити цей біт до зміни вмісту будь – якого з регістрів, описаних нижче
2	eFCAN	0: програмне забезпечення повинно прочитати всі повідомлення для всіх включених ідентифікаторів на всіх включених CAN – шинах; 1: фільтр буде піклуватися про отримання та зберігання повідомлення для обраних значень ID на обраній CAN – шині, що відповідає додатковому режиму «FullCAN»
31:3	–	Зарезервовані біти

4.12.4.2 Регістр CAN_SFF_SA

CAN_SFF_SA це регістр початкової адреси таблиці фільтрації за індивідуальними стандартними ідентифікаторами, таблиця 4.99.

Адреса – 0xE003 C004.

Таблиця 4.99 – Опис розрядів регістра CAN_SFF_SA

Біт	Назва	Функція
1:0	–	Зарезервовані біти

Продовження таблиці 4.99

Біт	Назва	Функція
10:2	SFF_sa	Містить адресу початку таблиці індивідуальних стандартних ідентифікаторів в оперативній пам'яті приймального фільтра. Якщо таблиця є порожньою, то слід записати в цей регістр те ж саме значення, що і в регістр SFF_GRP_sa. Якщо біт eFCAN в регістрі AFMR встановлено, то значення цих бітів визначає також розмір таблиці стандартних ідентифікаторів в оперативній пам'яті приймального фільтра, в якій приймальний фільтр буде роботи пошук і (якщо знайде збіг) автоматично збереже прийняте повідомлення
31:11	–	Зарезервовані біти

4.12.4.3 Регістр CAN_SFF_GRP_SA

CAN_SFF_GRP_SA це регістр початкової адреси таблиці фільтрації за групою стандартних ідентифікаторів, таблиця 4.100.

Адреса – 0xE003 C008.

Таблиця 4.100 – Опис розрядів регістра CAN_SFF_GRP_SA

Біт	Назва	Функція
1:0	–	Зарезервовані біти
11:2	GRP _sa	Адреса початку таблиці групи стандартних ідентифікаторів в оперативній пам'яті приймального фільтра. Якщо таблиця є порожньою, то слід записати в цей регістр те ж саме значення, що і в регістр ENDofTable. Найбільше значення, яке може бути записано в цей регістр – 0x800. У цьому випадку ця таблиця є пустою і використовується останнє слово (0x7FC) оперативної пам'яті пошукової таблиці приймального фільтра.
31:12	–	Зарезервовані біти

4.12.4.4 Регістр CAN_EFF_SA

CAN_EFF_SA це регістр початкової адреси таблиці фільтрації за індивідуальними розширеними ідентифікаторами, таблиця 4.101.

Адреса – 0xE003 C00C

Таблиця 4.101 – Опис розрядів регістра CAN_EFF_SA

Біт	Назва	Функція
1:0	–	Зарезервовані біти
10:2	EFF _sa	Адреса початку таблиці індивідуальних розширених ідентифікаторів в оперативній пам'яті приймального фільтра. Якщо таблиця є порожньою, то слід записати в цей регістр те ж саме значення, що і в регістр EFF_GRP_sa. Найбільше значення, яке може бути записано в цей регістр – 0x800. У цьому випадку обидві таблиці розширених ідентифікаторів є порожніми і використовується останнє слово (0x7FC) оперативної пам'яті пошукової таблиці приймального фільтра.
31:11	–	Зарезервовані біти

4.12.4.5 Регістр CAN_EFF_GRP_SA

CAN_EFF_GRP_SA це регістр початкової адреси таблиці фільтрації за групою розширених ідентифікаторів, таблиця 4.102.

Адреса – 0xE003 C010.

Таблиця 4.102 – Опис розрядів регістра CAN_EFF_GRP_SA

Біт	Назва	Функція
1:0	–	Зарезервовані біти
10:2	GRP _sa	Адреса початку таблиці групи розширених ідентифікаторів в оперативній пам'яті приймального фільтра. Якщо таблиця є порожньою, то слід записати в цей регістр те ж саме значення, що і в регістр ENDofTable. Найбільше значення, яке може бути записано в цей регістр – 0x800. У цьому випадку ця таблиця є пустою і використовується останнє слово (0x7FC) оперативної пам'яті пошукової таблиці приймального фільтра.

Продовження таблиці 4.102

Біт	Назва	Функція
31:11	–	Зарезервовані біти

4.12.4.6 Регістр CAN_EOT

CAN_EOT це регістр кінцевої адреси таблиці фільтрації, таблиця 4.103.

Адреса – 0xE003 C014

Таблиця 4.103– Опис розрядів регістра CAN_EOT

Біт	Назва	Функція
1:0	–	Зарезервовані біти
11:2	EndofTable	ENDofTable (кінець таблиці фільтрації) містить наступну адресу відносто останньої активної таблиці фільтрації. Якщо FullCAN – режим включено (біт eFCAN регістра AFMR встановлено), то це значення вказує на початок області фільтра в ОЗП, де приймальний фільтр буде автоматично отримувати повідомлення для обраних ідентифікаторів на обраних CAN – шинах.
31:12	–	Зарезервовані біти

4.12.4.7 Регістр LUTerrAd

LUTerrAd це регістр таблиці помилок фільтрації, таблиця 4.104.

Адреса – 0xE003 C018

Таблиця 4.104 – Опис розрядів регістра LUTerrAd

Біт	Назва	Функція
1:0	–	Зарезервовані біти
10:2	LUTerrAd	Це поле, доступне тільки для читання. Поле містить адресу в таблиці пошуку ОЗП, де приймальний фільтр виявив помилку в змісті таблиць.
31:11	–	Зарезервовані біти

4.12.4.8 Регістр LUTerr

LUTerr це регістр помилки таблиці фільтрації, таблиця 4.105.

Адреса – 0xE003 C01C

Таблиця 4.105 – Опис розрядів регістра LUTerr

Біт	Назва	Функція
0	LUTerr	Цей біт, доступний тільки для читання 1: якщо приймальний фільтр виявляє помилку в змісті таблиці, розташованої в оперативній пам'яті приймального фільтра. Очищається, коли програма читає регістр LUTerrAd.
31:1	–	Зарезервовані біти

4.12.4.9 Регістр FCANIE

FCANIE це регістр включення FullCAN – переривання, таблиця 4.106.

Адреса – 0xE003 C020

Таблиця 4.106 – Опис розрядів регістра FCANIE

Біт	Назва	Функція
0	FCANIE	0: відключення глобального FullCAN – переривання 1: включення глобального FullCAN – переривання
31:1	–	Зарезервовані біти

4.12.4.10 Регістр FCANIC0

FCANIC0 це регістр прапорців переривань для 0…31 бітів FullCan – повідомлення, таблиця 4.107.

Адреса – 0xE003 C024

Таблиця 4.107 – Опис розрядів регістра FCANIC0

Біт	Назва	Функція
0	IntPnd0	FullCan – переривання від об`єкту 0
…	IntPndx (0 < x < 30)	FullCan – переривання від об`єкту x
31	IntPnd31	FullCan – переривання від об`єкту 31

4.12.4.11 Регістр FCANIC1

FCANIC1 це регістр прапорців переривань для 32…63 бітів FullCan – повідомлення, таблиця 4.108.

Адреса – 0xE003 C028

Таблиця 4.108 – Опис розрядів регістра FCANIC0

Біт	Назва	Функція
0	IntPnd32	FullCan – переривання від об`єкту 32
…	IntPndx (33 < x < 62)	FullCan – переривання від об`єкту x
31	IntPnd63	FullCan – переривання від об`єкту 63

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1) Назвіть число CAN–контролерів, що мають мікроконтролери сімейства LPC2300.
2) Назвіть області застосування CAN–мереж.
3) Опишіть еталонну модель ISO/OSI CAN–шини.
4) Опишіть структуру вузла CAN–мережі.
5) Опишіть сигнали фізичного рівня CAN.
6) Назвіть максимальну швидкість передачі даних CAN–мережею.
7) Назвіть максимальну кількість байтів в пакеті CAN–протоколу.
8) Наведіть та поясніть формат пакета повідомлень CAN.
9) Опишіть дві версії CAN–протоколу.
10) Наведіть та поясніть спрощену структуру CAN–контролера мікроконтролера LPC2378.
11) Який завадостійкий код використовується в CAN–мережі? Дайте характеристику коду.
12) Поясніть основні характеристики CAN–модуля.
13) Перерахуйте типи помилок, що можуть бути виявлені при передачі CAN–мережею. Наведіть механізми виявлення помилок.
14) Для чого потрібна синхронізація при передачі CAN–мережею? Назвіть та опишіть види синхронізації.
15) Яка формула використовується для обчислення інтервалу передачі біта?
16) Назвіть та поясніть керуючі регістри CAN–модулем.
17) Наведіть та поясніть базові алгоритми передачі, прийому та фільтрації повідомлень.
18) Назвіть та поясніть керуючі регістри налаштування фільтра повідомлень.
19) Поясніть режим фільтрації повідомлень FullCan.
20) Скільки біт може мати ідентифікатор повідомлення?
21) Як виконується арбітраж повідомлень на CAN–шині?

ПРЕДМЕТНИЙ ПОКАЖЧИК

Внутрішні шини 37, 99;
Формування системних тактових сигналів 40;
Архітектура Фон–Неймана 48;
Логічна організація пам’яті 50;
Модуль прискорення роботи пам'яті 54;
Фазове автопідлаштовування частоти 62;
Формули отримання частоти схеми ФАПЧ 78;
Взаємодія програми з модулем I^2C 226;
Формат пакета повідомлень CAN 306;
Арбітраж на шині CAN 308;
Тактова синхронізація модуля CAN 312;
Обробка помилок в модулі CAN 314;
Фільтрації повідомлень в модулі CAN 322;

Модулі:

Переривання 83;
ФАПЧ 62;
ПДП 98 ;
порти введення/виведення 122 ;
таймери/лічильники 132;
ШІМ 151;
сторожовий таймер, 158;
УАПП, 161;
I^2C, 193 ;
SPI, 242;
АЦП 256 ;
ЦАП 281 ;
RTC 286 ;
CAN 302 ;

Керуючі регістри:

керування системою тактування 40;
ФАПЧ 67 ;
ПДП 103 ;
GPIO 126;
таймерів/лічильників 135;
ШІМ 151 ;
стотожовий таймер 158;
УАПП 175;
I^2C 229;
SPI 250;
АЦП 266;
ЦАП 283;
RTC 290;
CAN 329;

Режими роботи:

Idle 43;
Sleep 43;
Power Down 45;
Deep Power Down 45;

ADC0 264;
пам’яті 52;
зниженого енергоспоживання 43;
таймерів/лічильників 132;
ШІМ 152;
УАПП 170;
SPI 244;
I^2C 200;
АЦП 264;
ЦАП 278;
RTC 288;
CAN 318;
Стани модуля I^2C:
ведений передавач 219;
ведений приймач 210;
ведучий передавач 200;
ведучий приймач 206
Структурна схема:
МК LPC 2378 27;
інтерфейсу RS 232 161;
ядра LPC 2378 28;
мережі I2C 193;
мережі SPI 243;
з`єднання вузлів SPI 246 ;
АЦП послідовного наближення 260 ;
модуля АЦП 263 ;
модуля ЦАП 278 ;
RTC 287;
модуля CAN 310;
Розрахунок:
частоти ФАПЧ 78;
частоти передачі бітів АЦП 264;
частоти передачі бітів I^2C 199;
АЦП 256;
ЦАП 279;
ПВЗ 258;
RTC 288;
інтервалу передачі біта CAN 313;
Часові діаграми роботи:
I^2C 198;
SPI 245;
CAN 309;

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Мартин Т. Инсайдерское руководство по STM32 / Тревор Мартин. – Москва: Издательский дом «Додэка XXI», 2012.
2. Никамин В. А. Аналого–цифровые и цифро–аналоговые преобразователи. – Альтекс–А, 2003.
3. Редькин П.П. 32/16–битные микроконтроллеры ARM7 семейства AT91SAM7 фирмы Atmel. Руководство пользователя (+ CD). – М. : Издательский дом «Додэка–XXI», 2008.
4. Тревор Мартин Микроконтроллеры ARM7 семейств LPC2300/2400. Вводный курс разработчика / пер. с англ. Евстифеева А. В. – М. : Додэка–XXI, 2010.
5. Федорков Б.Г., Телец В.А. Микросхемы ЦАП и АЦП: функционирование, параметры, применение.– М.: Энергоатомиздат, 1990.
6. Электронные прмышленные устройства: Учеб. Для студ. вузов/В. И. Васильев, Ю. М. Гусев и др. –М. : Высш. шк., 1988.
7. LPC2377/78 Product data sheet / NXP Semiconductors product data sheet. – 2013. Система команд в режиме ARM [Електронний ресурс] // Рынок микроэлектроники – Режим доступу до ресурсу: http://www.gaw.ru/html.cgi/txt/doc/micros/arm/asm/asm_arm/lds_str.htm.
8. LPC2378–datasheet.
9. STM32. DAC [Електронний ресурс]. – 2012. – Режим доступу до ресурсу: http://chipspace.ru/stm32–dac–1/.