Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра електроніки та телекомунікацій
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

ОРГАНІЗАЦІЯ СИСТЕМИ КОМУТАЦІЇ КІНЦЕВИХ ПРИСТРОЇВ НА БАЗІ STM32

ORGANIZATION THE SWITCHING SYSTEM OF TERMINAL DEVICES BASED ON STM32

спеціальність 172 Електронні комунікації та радіотехніка
(шифр і назва спеціальності)

освітня програма «Телекомунікації та радіотехніка»
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи ЕКРм-21
Карпінський Назар Костянтинович

_____
(підпис)

Керівник: к.т.н., доцент
Якимчук Наталія Миколаївна

_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«___» грудня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Селепина Йосип Романович

_____
(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*

Кафедра *електроніки та телекомунікацій*

Ступінь вищої освіти: *магістр*

Галузь знань: *17 Електроніка, автоматизація та електронні комунікації*

Спеціальність: *172 Електронні комунікації та радіотехніка*

Освітня програма: *«Телекомунікації та радіотехніка»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. В. ЗАБЛОЦЬКИЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Карпінському Назару Костянтиновичу

1. Тема кваліфікаційної роботи: *Організація системи комутації кінцевих пристроїв на базі STM32*

Керівник роботи: *к.т.н., доцент Якимчук Наталія Миколаївна*

затверджені наказом закладу вищої освіти від «14» січня 2025 р. № 20/01-02

2. Строк подання здобувачем роботи: 05.12.2025 р.

3. Вихідні дані до роботи: *Інформаційні матеріали щодо мікроконтролерів STM32 та архітектури ARM Cortex-M, технічні характеристики периферійних інтерфейсів (UART, SPI, I²C, USB, CAN) та можливостей апаратної комутації на базі STM32. Дані про особливості підключення кінцевих пристроїв, режими енергоспоживання, принципи роботи DMA та контролера переривань NVIC. Матеріали щодо схем включення мікроконтролерів, початкових режимів завантаження, методів тестування апаратної частини та рекомендацій із проєктування друкованих плат. Структурований план розробки апаратної та програмної частини комутаційної системи.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Вступ

Розділ 1 Основи платформи STM32

Розділ 2 Технологія кінцевих пристроїв STM32

Розділ 3 Інтеграція кінцевих пристроїв STM32 у телекомунікаційні системи

Розділ 4 Надійність, тестування апаратної частини та рекомендації з проєктування плат

Висновки

5. Перелік графічного (ілюстративного) матеріалу:

Презентація PowerPoint 11 слайдів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Розділ 1	*Якимчук Н. М., доцент*		
Розділ 2	*Якимчук Н. М., доцент*		
Розділ 3	*Якимчук Н. М., доцент*		
Розділ 4	*Якимчук Н. М., доцент*		
Висновки	*Якимчук Н. М., доцент*		
Нормоконтроль	*Селепина Й. Р., доцент*		

7. Дата видачі завдання 03.02.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Розділ 1 Основи платформи STM32*	до 18.09.2025 р.	
2.	*Розділ 2 Технологія кінцевих пристроїв STM32*	до 26.10.2025 р.	
3.	*Розділ 3 Інтеграція кінцевих пристроїв STM32 у телекомунікаційні системи*	до 02.11.2025 р.	
4.	*Розділ 4 Надійність, тестування апаратної частини та рекомендації з проєктування плат*	до 09.11.2025 р.	
5.	*Висновки*	до 16.11.2025 р.	
6.	*Формування переліку використаних джерел*	до 23.11.2025 р.	
7.	*Оформлення ілюстративного матеріалу*	до 28.11.2025 р.	
8.	*Нормоконтроль*	до 01.12.2025 р.	
9.	*Інструментальна перевірка на академічний плагіат*	до 05.12.2025 р.	
10.	*Представлення кваліфікаційної роботи магістра до захисту*	до 30.12.2025 р.	

Здобувач вищої освіти _____ Карпінський Н. К.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Якимчук Н. М.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Карпінський Н. К. Організація системи комутації кінцевих пристроїв на базі STM32. Рукопис. Кваліфікаційна робота магістра ОП «Телекомунікації та радіотехніка» спеціальності 172 Електронні комунікації та радіотехніка. Луцький національний технічний університет. Луцьк, 2025. 69 с.

Кваліфікаційна робота магістра складається з вступу, чотирьох розділів, висновків, переліку використаних джерел. У роботі розглянуто питання побудови системи комутації кінцевих пристроїв на основі мікроконтролерів STM32. Проведено аналіз сучасних підходів до організації комутаційних процесів у телекомунікаційних системах, досліджено архітектуру ARM Cortex-M та технічні характеристики STM32, визначено їх переваги для реалізації апаратного керування комутацією. Розроблено структурну та функціональну схему системи, обґрунтовано вибір апаратного забезпечення та програмних засобів (STM32CubeMX, STM32CubeIDE, HAL / LL бібліотеки). Проведено моделювання роботи системи, оцінено її продуктивність, стабільність та енергоефективність. Наведено рекомендації щодо інтеграції розробленої системи в телекомунікаційні мережі.

Ключові слова: STM32, ARM Cortex-M, комутація, мікроконтролер, телекомунікаційна система, UART, SPI, енергозбереження, STM32CubeMX.

ANNOTATION

Karpinskyi N. Organization of the Switching System of Terminal Devices Based on STM32. Manuscript. Master's qualification thesis of the educational program «Telecommunications and radio engineering» in specialty 172 Electronic communications and radio engineering. Lutsk National Technical University. Lutsk, 2025. 69 p.

The master's qualification thesis consists of an introduction, four chapters, conclusions, a list of references, and appendices?. The thesis explores the design and implementation of a switching system for terminal devices based on STM32 microcontrollers. It analyzes current approaches to switching in telecommunication systems, examines the ARM Cortex-M architecture and technical features of STM32, and highlights their advantages for hardware-level switching control. A structural and functional diagram of the system is developed, with justification for the selection of hardware components and software tools (STM32CubeMX, STM32CubeIDE, HAL/LL libraries). The system's performance, stability, and energy efficiency are evaluated through modeling. Recommendations are provided for integrating the developed system into telecommunication networks.

Keywords: STM32, ARM Cortex-M, Switching, Microcontroller, Telecommunication System, UART, SPI, Energy Efficiency, STM32CubeMX.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ ТА СКОРОЧЕНЬ

TIM – апаратний таймер у мікроконтролерах STM32

КВВП – контролер векторизованих вкладених переривань

STM32 (STMicroelectronics Microcontroller Family) – мікроконтролери сімейства STM32 компанії STMicroelectronics

ARM (Advanced RISC Machine) – архітектура мікропроцесорів

Cortex-M (ARM Cortex-M Series) – серія енергоефективних процесорних ядер ARM для вбудованих систем

UART (Universal Asynchronous Receiver Transmitter) – асинхронний приймач-передавач

USART (Universal Synchronous / Asynchronous Receiver Transmitter) – синхронно-асинхронний приймач-передавач

SPI (Serial Peripheral Interface) – послідовний периферійний інтерфейс

I²C (Inter-Integrated Circuit) – інтерфейс для зв’язку між мікросхемами

DMA (Direct Memory Access) – прямий доступ до пам’яті

ADC (Analog-to-Digital Converter) – аналого-цифровий перетворювач

DAC (Digital-to-Analog Converter) – цифро-аналоговий перетворювач

PWM (Pulse Width Modulation) – широтно-імпульсна модуляція

HAL (Hardware Abstraction Layer) – рівень абстракції апаратного забезпечення

LL (Low-Layer Library) – низькорівнева бібліотека для STM32

CMSIS (Cortex Microcontroller Software Interface Standard) – стандарт програмного інтерфейсу для ARM Cortex

IDE (Integrated Development Environment) – інтегроване середовище розробки

ART (Adaptive Real-Time Accelerator) – адаптивний прискорювач реального часуSCR (System Control Register) – системний регістр керування

RTOS (Real-Time Operating System) – операційна система реального часу

GPON (Gigabit Passive Optical Network) – гігабітна пасивна оптична мережа

IoT (Internet of Things) – Інтернет речей

ВСТУП

У сучасному світі стрімкий розвиток телекомунікаційних систем є визначальним чинником науково-технічного прогресу. Ефективна передача, обробка та комутація інформації формують основу цифрової інфраструктури, що охоплює промисловість, побут, оборонну сферу та інтернет речей (IoT). Зростання кількості кінцевих пристроїв, обсягів переданих даних і вимог до швидкодії мереж зумовлює потребу у створенні надійних, масштабованих і енергоефективних систем комутації.

Одним із перспективних рішень для реалізації таких систем є використання мікроконтролерів STM32 виробництва STMicroelectronics. Завдяки високій продуктивності, широким можливостям периферії, підтримці сучасних інтерфейсів зв'язку та гнучкому енергоспоживанню, STM32 забезпечують апаратне керування комутаційними процесами, що дозволяє зменшити затримки, підвищити стабільність з'єднання та оптимізувати роботу мережі.

Актуальність теми полягає у необхідності розробки комутаційних систем, здатних забезпечити ефективну взаємодію між великою кількістю пристроїв, зокрема, в умовах сенсорних мереж, систем реального часу та IoT-середовищ. У таких системах важливими є точність синхронізації, гнучкість масштабування, енергоефективність та надійність.

Метою кваліфікаційної роботи є розробка та аналіз системи комутації кінцевих пристроїв на базі STM32, яка забезпечує ефективний обмін даними в локальних і телекомунікаційних мережах.

Для досягнення мети передбачено виконання таких завдань:

– аналіз архітектури ARM Cortex-M та можливості мікроконтролерів STM32 для побудови комутаційних систем;

– розробка структурної та функціональної схеми комутаційної системи;

– дослідження продуктивності і режимів енергоспоживання системи;

– дослідження налаштування периферії STM32 та реалізації надійності керування комутаційними процесами;

– розробка рекомендацій щодо проєктування друкованих плат для систем комутації.

Об’єктом дослідження є процес комутації кінцевих пристроїв у телекомунікаційних та автоматизованих системах. Предметом – організація комутаційної системи на основі STM32.

Практичне значення роботи полягає у можливості застосування розробленої системи в локальних мережах, системах управління, моніторингу та критичних інфраструктурах, де важлива висока надійність і мінімальні затримки передачі сигналів. Окремі результати роботи були представлені у статті Якимчук Н. М., Карпінський Н. К. Організація UART-комунікації в STM32 для системи комутації кінцевих пристроїв. Actual Problems of Automation and Control. Lutsk. Lutsk. № 13. 2025. С.152-156.

Новизна роботи полягає в обґрунтуванні та реалізації комплексного підходу до побудови енергоефективної та масштабованої системи комутації на базі STM32 з використанням апаратних механізмів контролю надійності, оптимізованих режимів енергоспоживання та моделювання реального часу.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ СИСТЕМ КОМУТАЦІЇ

1.1 Загальні характеристики мікроконтролерів STM32

У сучасній електроніці мікроконтролери займають ключове місце, оскільки вони забезпечують інтелектуальне керування різноманітними пристроями – від простих побутових систем до складних телекомунікаційних комплексів та промислової автоматики. Серед багатьох виробників мікроконтролерних платформ особливе місце посідає компанія STMicroelectronics, яка у 2007 році представила нове сімейство 32-розрядних мікроконтролерів під торговою маркою STM32 (табл. 1.1). Саме ця лінійка швидко стала однією з найпопулярніших у світі завдяки поєднанню високої продуктивності, широкого набору периферійних можливостей та енергоефективності.

Таблиця 1.1 – Периферійні інтерфейси в серіях STM32

Серія STM32	UART	SPI	I²C	USB	CAN	Ethernet
STM32F1	+	+	+	–	+	–
STM32F4	+	+	+	+	+	+
STM32L4	+	+	+	+	–	–
STM32H7	+	+	+	+	+	+
STM32WB	+	+	+	+	–	–

Особливістю STM32 є їхня масштабованість: розробники можуть обрати як простий і недорогий мікроконтролер для масового пристрою, так і високопродуктивний варіант для систем із підвищеними вимогами до обчислювальних ресурсів. При цьому, сумісність між різними серіями та наявність єдиної екосистеми програмного забезпечення дозволяють мінімізувати витрати на перехід між моделями.

« Мікроконтролери STM32 є одними з кращих для використання в вбудованих системах завдяки використанню 32-х бітного мікропроцесорного ядра ARM із розвинутою архітектурою, яка забезпечує високу швидкість обчислень, гнучку систему переривань та різні режими енергоспоживання. » [1]

Процесори Cortex-M забезпечують кілька рівнів керування живленням, які можна розділити на дві основні групи: вбудовані функції та режими живлення, визначені користувачем.

STM32 підтримують кілька режимів енергоспоживання:

– Run mode;

– Sleep mode;

– Stop mode;

– Standby mode.

Процесори Cortex-M забезпечують визначені користувачем режими живлення через системний реєстр управління (SCR). Перший – це режим Run mode (рис. 1.1), у якому процесор використовує всі свої можливості. У даному режимі роботи споживання енергії залежить від тактової частоти та задіяних периферійних пристроїв. [1]

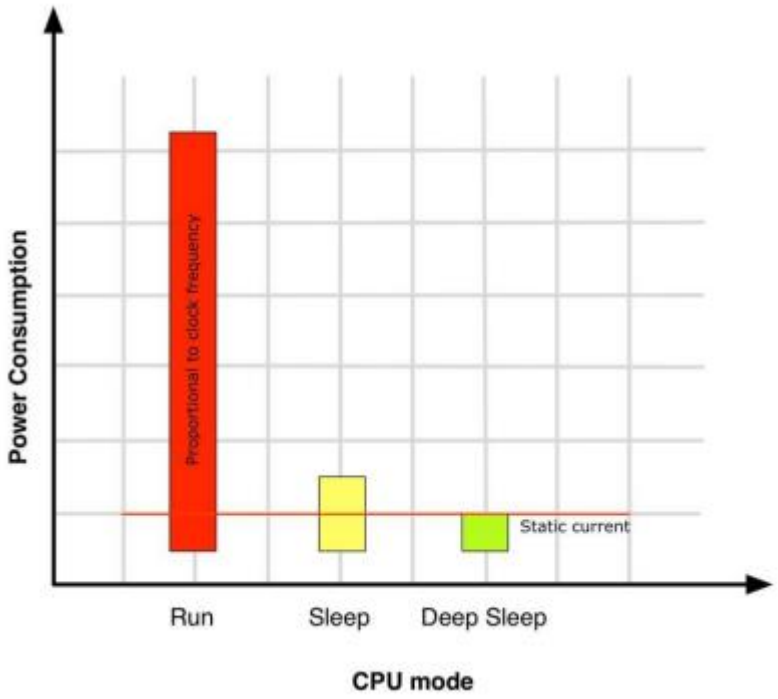


Рисунок 1.1 – Споживання енергії Cortex-M при різних режимах живлення [1]

« Сплячий режим (Sleep mode) є першим доступним режимом низького енергоспоживання для зниження споживання енергії. Після активації більшість функціональних можливостей призупиняється, частота процесора знижується, а

його діяльність зменшується до необхідної для його пробудження. У режимі глибокого сну (Deep sleep mode) всі тактові сигнали припиняються і ЦП потребує зовнішньої події, щоб вийти з цього стану.

На рисунку 1.2 відображено споживану потужність мікроконтролера STM32F2, що працює на частоті 80 МГц при 30° С. Як видно з рисунка, максимальне споживання енергії досягається в режимі Run (тобто в активному режимі) з вимкненим Adaptive Real-Time (ART) прискорювачем. Увімкнувши ART прискорювач, можна заощадити до 10 мА, а також досягти кращої обчислювальної продуктивності. » [1]

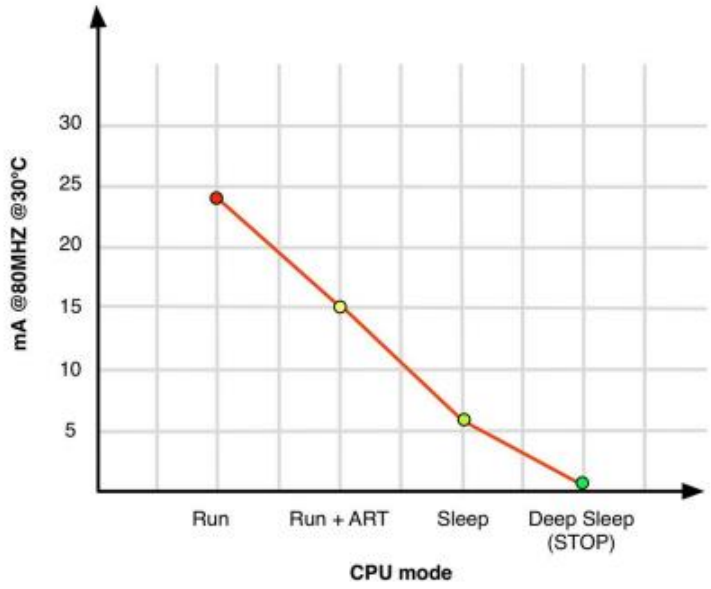


Рисунок 1.2 – Споживання електроенергії STM32F2 при різних режимах живлення [1]

Таким чином, STM32 є не просто як окремі інтегральні мікросхеми, а як ціла платформа, яка охоплює різні класи задач і забезпечує єдиний підхід до проєктування вбудованих систем.

Сімейство STM32 охоплює понад десять підсімейств, що відрізняються за рівнем продуктивності, енергоспоживанням та орієнтацією на конкретні сфери застосування. Найпоширеніші серії включають:

- STM32F0/F1/F3 – базові та універсальні рішення;
- STM32F4/F7 – продуктивніші мікроконтролери з FPU;

– STM32L0/L4/L5 – енергоефективні моделі для IoT;

– STM32H7 – високопродуктивні контролери для мультимедіа;

– STM32WB/WL – моделі з бездротовими інтерфейсами.

Для зручності порівняння основні характеристики популярних серій STM32 наведено в таблиці 1.2.

Таблиця 1.2 – Порівняльна характеристика серій мікроконтролерів STM32

Серія STM32	Частота (МГц)	Пам’ять (Flash / RAM)	Енергоспоживання	Призначення
STM32F1	до 72	до 512 КБ / 64 КБ	середнє	універсальні рішення
STM32F4	до 180	до 2 МБ / 256 КБ	високе	мультимедіа, DSP
STM32L4	до 80	до 1 МБ / 320 КБ	низьке	IoT, автономні пристрої
STM32H7	до 480	до 2 МБ / 1 МБ	високе	обробка відео, графіка
STM32WB	до 64	до 1 МБ / 256 КБ	низьке	бездротові технології (BLE, Zigbee)

« Кожен пристрій збору та передачі інформації має у складі, як мінімум, один мікроконтролер, який виконує основні задачі по збору, обробці та передачі інформації. » [1]

Окремою рисою STM32 є широкий діапазон продуктивності. Наприклад, серія STM32F0 має частоту ядра до 48 МГц і підходить для простих задач керування, тоді як STM32H7 працює на частотах понад 550 МГц і може виконувати складні алгоритми цифрової обробки сигналів чи обслуговувати графічні інтерфейси.

Багато моделей підтримують апаратний DSP-набір інструкцій, що робить їх придатними для обробки аудіо- та відеопотоків у реальному часі. Крім того, можливість масштабування продуктивності всередині однієї сім’ї забезпечує зручність для розробників: проєкт може починатися з базової моделі, а згодом масштабуватися на потужніший контролер без серйозних змін у коді та апаратурі.

Мікроконтролери STM32 мають багатий набір вбудованих периферійних модулів:

– UART/USART, SPI, I²C;

– CAN, USB, Ethernet;

– АЦП, ЦАП, компаратори;

– таймери, ШІМ;

– DMA, криптографія, дисплеї.

USART (Universal Synchronous / Asynchronous Receiver Transmitter) підтримує як синхронну, так і асинхронну передачу даних, тоді як UART (Universal Asynchronous Receiver Transmitter) – лише асинхронну. На рисунку 1.3 показано, що USART використовує додатковий сигнал синхронізації (тактовий сигнал), який дозволяє точніше узгоджувати передачу між пристроями, чого немає в UART.

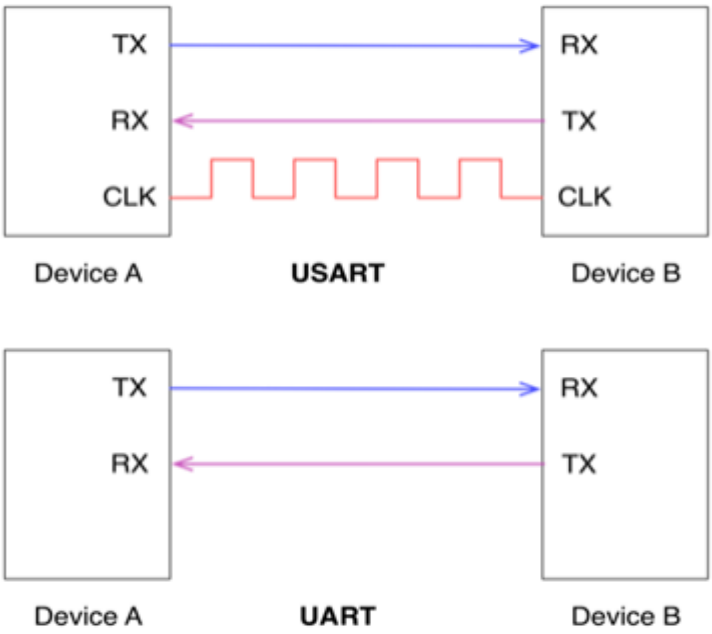


Рисунок 1.3 – Різниця в сигналах між USART і UART [6]

« Наведені інтерфейси використовуються для комунікації через канали прийому передачі даних, а також для отримання інформації від різноманітних датчиків. » [1]

Для зручності розробників компанія STMicroelectronics створила цілу екосистему програмних засобів:

– STM32CubeMX – графічний інструмент для початкової конфігурації мікроконтролера та генерації коду;

– STM32CubeIDE – інтегроване середовище розробки, що об’єднує компілятор, відлагоджувач і редактор;

– HAL (Hardware Abstraction Layer) та LL (Low-Layer) бібліотеки – для спрощення програмування периферії;

– підтримка стандарту CMSIS, який забезпечує уніфікований доступ до ресурсів Cortex-M.

Крім того, STM32 підтримується багатьма сторонніми інструментами (Keil, IAR, GCC), а також операційними системами реального часу (FreeRTOS, Zephyr). Це створює зручне середовище для швидкого переходу від ідеї до готового продукту. На рисунку 1.4 показане головне вікно програми STM32CubeMX.

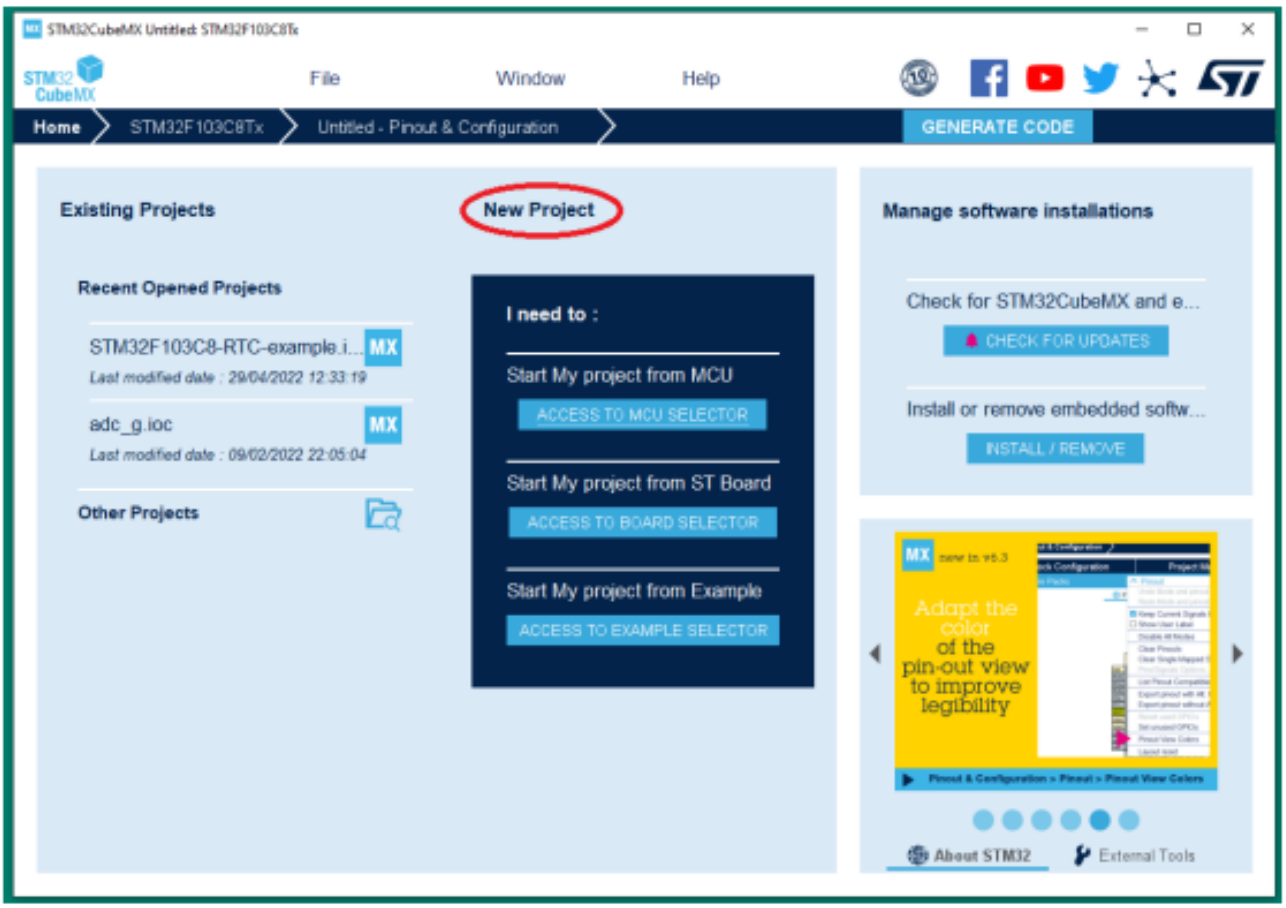


Рисунок 1.4 – Головне вікно програми STM32CubeMX [8]

Переваги STM32CubeMX:

– широкий асортимент підсімейств для різних класів задач;

– баланс між ціною та можливостями;

– масштабованість і сумісність між моделями;

– низьке енергоспоживання при високій продуктивності;

– активна спільнота та велика кількість прикладів коду.

Недоліки STM32CubeMX:

– складність для початківців через велику кількість налаштувань;

– іноді недостатньо структурована офіційна документація;

– наявність помилок у бібліотеках HAL, що може ускладнювати розробку.

1.2 Місце STM32 у телекомунікаційних системах та актуальність їх використання

У сучасних телекомунікаційних системах мікроконтролери відіграють ключову роль у забезпеченні апаратного керування процесами обміну даними, синхронізації, маршрутизації та комутації. Серед доступних рішень особливу увагу привертає платформа STM32, яка поєднує високу продуктивність, гнучкість конфігурації та підтримку широкого спектра інтерфейсів, що критично важливо для реалізації комунікаційних вузлів.

Мікроконтролери STM32 здатні ефективно виконувати функції керування в таких компонентах телекомунікаційної інфраструктури, як:

– термінальні пристрої доступу (наприклад, ONU в GPON);

– локальні вузли комутації в промислових мережах;

– сенсорні шлюзи в IoT-середовищах;

– модулі синхронізації та буферизації в системах реального часу.

Завдяки підтримці інтерфейсів UART, SPI, I²C, USB, Ethernet, CAN та інших, STM32 забезпечують гнучку інтеграцію з різними типами периферійних пристроїв – від датчиків і виконавчих механізмів до мережевих адаптерів. Наприклад, у системах GPON STM32 можуть використовуватись для локального керування оптичними приймачами, моніторингу стану лінії та реалізації протоколів взаємодії з центральним вузлом.

Крім того, STM32 активно застосовуються у побудові розумних вузлів комутації для IoT, де важливими є енергоефективність, компактність та здатність до автономної роботи. У таких системах мікроконтролер виконує роль локального процесора, який приймає сигнали від сенсорів, обробляє їх, приймає рішення та передає дані до хмарних платформ або центральних серверів.

STM32 застосовуються у багатьох типах телекомунікаційних пристроїв – від оптичних терміналів до промислових шлюзів. У таблиці 1.3 наведено приклади таких застосувань із зазначенням ролі мікроконтролера в кожному випадку.

Таблиця 1.3 – Приклади застосування STM32 у телекомунікаційних системах

Сфера застосування	Пристрій / вузол	Роль STM32
GPON	ONU (оптичний термінал)	Керування інтерфейсами, моніторинг лінії
IoT	Сенсорний шлюз	Збір даних, передача по LoRa/Wi-Fi
Промислова автоматизація	Локальний вузол комутації	Перемикання каналів, буферизація
Системи безпеки	Контролер доступу	Обробка сигналів, шифрування, логування
Резервні системи	Гарячий резерв комутаційного вузла	Перехоплення керування, watchdog-контроль

Особливу перевагу STM32 становить наявність апаратної підтримки криптографії (у серіях STM32L5, STM32H7), що дозволяє реалізувати захищені канали зв’язку, що є критично важливим для телекомунікаційних систем, які працюють з конфіденційними даними або в умовах підвищених вимог до кібербезпеки.

У контексті побудови комутаційних систем STM32 дозволяють реалізувати:

– апаратне перемикання каналів з мінімальними затримками;

– буферизацію та обробку пакетів на рівні периферії;

– адаптивне керування енергоспоживанням залежно від навантаження;

– взаємодію з RTOS для забезпечення детермінованої поведінки в системах реального часу.

Таким чином, STM32 є не лише апаратною платформою, а й інструментом для побудови інтелектуальних телекомунікаційних рішень, здатних адаптуватися

до вимог сучасних мереж – від локальних до глобальних, від дротових до бездротових, від простих до критично важливих.

1.3 Огляд архітектури процесорів ARM Cortex

Архітектура ARM (Advanced RISC Machine) є загальновизнаним стандартом для проектування енергоефективних та високопродуктивних мікропроцесорів, що використовуються в широкому спектрі пристроїв – від мобільних телефонів і планшетів до вбудованих систем та обладнання для Інтернету речей (IoT). Базуючись на принципах скороченого набору команд (RISC), архітектура ARM відрізняється спрощеною структурою інструкцій, великим регістровим файлом та завантажувально-зберігальною моделлю, що забезпечує високу швидкість обробки даних при мінімальному енергоспоживанні.

ARM Holdings, розробник архітектури, працює за моделлю ліцензування, що дозволяє численним виробникам (включаючи STMicroelectronics, NXP, Texas Instruments) створювати власні мікросхеми, інтегруючи ядра ARM. Така екосистема сприяє постійному вдосконаленню, стандартизації та доступності інструментів розробки.

Ключовим моментом в еволюції архітектури ARM, який має вирішальне значення для сучасних вбудованих систем, стало впровадження набору інструкцій Thumb-2. Ця технологія дозволила ефективно вирішити дилему між щільністю коду та максимальною продуктивністю, характерну для попередніх архітектур.

« ЦПУ Cortex підтримує набір інструкцій Thumb-2, який являє собою суміш 16- та 32-бітних інструкцій. Інструкції Thumb-2 дають покращення щільності коду на 26 % порівняно з 32-бітними інструкціями ARM та продуктивності на 25 % порівняно з 16-бітними інструкціями Thumb. Завдяки цьому, процесори Cortex мають змогу використовувати Flash-пам'ять з меншим обсягом. Менший обсяг Flash-пам'яті також зменшує потребу в споживанні енергії. » [1]

Використання змінної довжини інструкцій (16 або 32 біти) дозволяє компілятору оптимізувати програмний код, вибираючи компактні 16-бітні команди для звичайних операцій та потужні 32-бітні – для критичних ділянок.

Наприклад, ядра Cortex-M4, які часто використовуються в мікроконтролерах STM32F4, додатково включають модуль Digital Signal Processing (DSP) та підтримку інструкцій з плаваючою комою (FPU), що значно підвищує їхню ефективність у задачах обробки сигналів та керування. З метою максимальної адаптації до вимог різних ринкових сегментів, архітектура ARMv7 була розділена на три основні профілі, кожен з яких має чітко визначену оптимізацію:

1. Профіль A (Application): сконструйований для високої продуктивності та роботи з багатозадачними ОС, де потрібен блок керування пам'яттю (MMU) для віртуальної адресації та захисту пам'яті. Приклади: Cortex-A72, A53 використовуються у смартфонах, планшетах, комп'ютерах.

2. Профіль R (Real-time): орієнтований на жорсткі системи реального часу, де необхідна мінімальна, детермінована та передбачувана затримка. Цей профіль забезпечує швидке перемикання контексту та високу стійкість до збоїв. Використовується в автомобільній промисловості, промисловій автоматизації.

3. Профіль M (Microcontroller): спеціально розроблений для мікроконтролерів та недорогих вбудованих рішень. Цей профіль є основою для мікроконтролерів STM32. Його ключові пріоритети – низьке енергоспоживання, інтеграція периферії та ефективна обробка переривань.

«Це сімейство ядер, розроблене для подальшої інтеграції з периферійними пристроями виробника, щоб утворити готовий мікроконтролер. Процесори на базі Cortex-M є ідеальними для систем Інтернету речей (IoT), де пріоритетом є тривалий термін служби батареї та компактність коду.» [1]

Ключовою архітектурною перевагою ядер Cortex-M, яка відрізняє їх від багатьох інших мікроконтролерних архітектур, є інтегрований блок NVIC. Саме він забезпечує швидку, детерміновану та гнучку обробку всіх асинхронних подій.

«Центральний процесор призупиняє виконання поточного завдання, зберігає його контекст (тобто вказівник стека) і починає виконання підпрограми, призначеної для обробки події переривання. Ця підпрограма називається Процедурою обслуговування переривань (Interrupt Service Routine, ISR). NVIC забезпечує швидкий та ефективний перехід до цієї процедури, керуючи вкладеністю та пріоритетами. NVIC дозволяє призначати рівні пріоритету

виняткам і перериванням, що дозволяє перериванню з вищим пріоритетом перервати виконання ISR з нижчим пріоритетом.» [1]

NVIC виконує векторизований перехід, автоматично визначаючи адресу обробника без програмного сканування. Це забезпечує детерміновану затримку (латентність) обробки переривань. Ця затримка є фіксованою: «Затримка переривання становить: 12 циклів для всіх ядер Cortex-M3/4, 15 циклів для Cortex-M0, 16 циклів для Cortex-M0+, незалежно від поточного стану процесора. Це робить час реакції системи передбачуваним, що є життєво необхідним для додатків, які критичні до часу.» [1]

Ядра Cortex-M інтегрують також системний таймер SysTick, що має вирішальне значення для підтримки операційних систем реального часу. «SysTick – це 24-розрядний таймер зворотного відліку, який використовується для забезпечення системних інтервалів часу (тіків) для операційних систем реального часу (RTOS), таких як FreeRTOS. Він також може використовуватися для створення точних затримок, навіть якщо RTOS не використовується.» [1]

Для кращого розуміння функціональних можливостей архітектури ARM Cortex-M у контексті телекомунікаційних систем доцільно розглянути основні системні блоки ядра та їхнє призначення (табл. 1.4).

Таблиця 1.4 – Основні системні блоки архітектури ARM Cortex-M та їх функції

Компонент ядра	Призначення	Значення для телекомунікацій
NVIC (Nested Vectored Interrupt Controller)	Керування перериваннями	Забезпечує швидку реакцію на події в мережі
SysTick Timer	Системний таймер	Синхронізація обміну даними, тайм-аутів
MPU (Memory Protection Unit)	Захист пам’яті	Підвищення безпеки при передачі даних
FPU (Floating Point Unit)	Обробка чисел з плаваючою крапкою	DSP-обробка сигналів, QoS-аналіз
DWT (Data Watchpoint and Trace)	Трасування та налагодження	Аналіз продуктивності комутаційних процесів
ITM (Instrumentation Trace Macrocell)	Вивід діагностичних даних	Моніторинг стану мережевих інтерфейсів
ART Accelerator	Адаптивне кешування	Зменшення затримок при обробці пакетів
SCR (System Control Register)	Керування режимами живлення	Енергозбереження в режимах очікування

Вбудовані функції керування живленням дозволяють оптимізувати енергоспоживання мікроконтролера. Процесор може бути переведений у кілька режимів:

1. Активний режим (Run mode). Повна продуктивність. «Споживання енергії залежить від тактової частоти та задіяних периферійних пристроїв».

2. Сплячий режим (Sleep mode). Ядро процесора зупиняється, але периферійні пристрої та пам'ять продовжують працювати.

3. Режим глибокого сну (Deep Sleep mode / STOP/Standby). Найглибший режим енергозбереження.

« У режимі глибокого сну всі тактові сигнали припиняються, і ЦП потребує зовнішньої події, щоб вийти з цього стану. Споживання струму в цьому режимі наближається до статичного (струм витоку), що дозволяє пристроям працювати роками від невеликої батареї. » [1]

Ці механізми, керовані через системний регістр управління (SCR), дозволяють розробнику балансувати між швидкістю виконання та автономністю пристрою.

Для забезпечення сумісності програмного забезпечення в екосистемі Cortex-M було створено Cortex Microcontroller Software Interface Standard (CMSIS). Це стандартизований рівень апаратної абстракції, який дозволяє розробникам переносити код між мікроконтролерами різних виробників, які використовують однакове ядро Cortex-M.

« CMSIS-CORE забезпечує стандартизований інтерфейс для Cortex-M0/3/4/7. Він уніфікує доступ до регістрів NVIC, SysTick та інших ключових системних компонентів. » [1]

Незважаючи на наявність CMSIS, виробники, як STMicroelectronics (STM32), часто розробляють власні бібліотеки апаратної абстракції, такі як Hardware Abstraction Layer (HAL).

« Офіційний ST HAL – це основний спосіб розробки додатків для платформи STM32, хоча він часто використовує CMSIS-CORE як основу. Проте, підтримка всіх компонентів [CMSIS] від ST все ще є недостатньою, що вимагає

від розробника використання фірмових бібліотек STMicroelectronics для керування периферією. » [1]

Таким чином, архітектура ARM Cortex-M є ідеальною платформою для розробки вбудованих систем та IoT-пристроїв, поєднуючи високу обчислювальну ефективність, гнучкість обробки переривань та найвищу енергоефективність.

1.4 Інтеграція архітектури ARM Cortex у мікроконтролери

Архітектура ARM реалізується через низку ядер, зокрема Cortex-M, Cortex-R та Cortex-A, кожне з яких орієнтоване на певний клас застосувань. У мікроконтролерах STM32, що випускаються компанією STMicroelectronics, інтегровано ядра серії Cortex-M, спеціально розроблені для вбудованих систем.

« Cortex-M – це сімейство ядер, розроблене для подальшої інтеграції з периферійними пристроями виробника, щоб утворити готовий мікроконтролер » [1]

Інтеграція ARM Cortex у мікроконтролери передбачає створення складної системи взаємозв'язків між ядром, пам'яттю та периферійними пристроями (рис. 1.6). У STM32 для цього застосовується багаторівнева шина BusMatrix, яка з'єднує ядро Cortex-M із периферією та DMA-контролером. Це забезпечує одночасний доступ до кількох пристроїв без конфлікту запитів.

Такий підхід дозволяє зберігати високу продуктивність навіть при роботі з численними периферійними пристроями. Крім того, інтеграція ARM Cortex забезпечує низьке енергоспоживання – критично важливу характеристику для IoT-пристроїв і систем із живленням від батарей. Завдяки апаратним режимам сну (Sleep, Stop, Standby), мікроконтролери можуть знижувати енергоспоживання на порядок, залишаючи активними лише необхідні модулі.

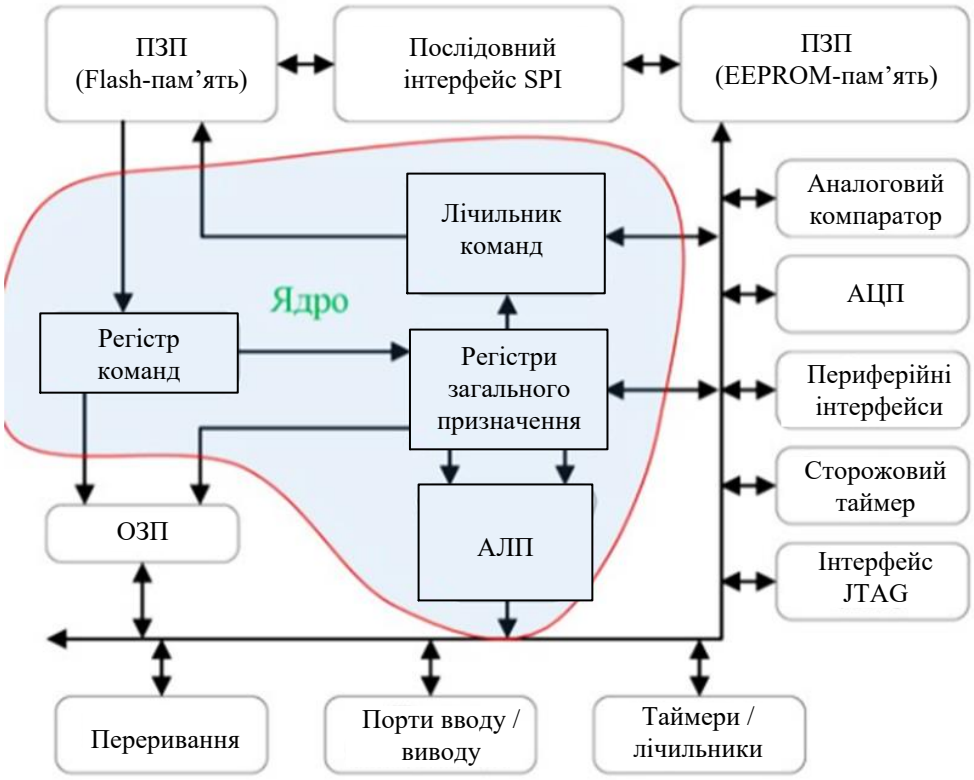


Рисунок 1.6 – Розширена RISC (reduced instruction set computer) архітектура мікроконтролерів

Одним із важливих аспектів інтеграції є стандартизація програмного інтерфейсу. ARM розробила Cortex Microcontroller Software Interface Standard (CMSIS), який створює єдиний рівень абстракції для всіх ядер Cortex-M. «CMSIS – це незалежний від виробника рівень апаратної абстракції для серії процесорів Cortex-M і визначає інтерфейси налагоджувачів». [1]

Завдяки CMSIS розробники можуть переносити код між різними моделями мікроконтролерів без значних змін, що пришвидшує розробку і підвищує надійність програмних рішень. Важливими компонентами CMSIS є:

- CMSIS-CORE – забезпечує доступ до регістрів ядра;
- CMSIS-Driver – визначає стандартизовані драйвери периферійних пристроїв;
- CMSIS-RTOS API – уніфікує взаємодію з операційними системами реального часу.

Таким чином, інтеграція ARM Cortex у мікроконтролери STM32 супроводжується не лише апаратною, а й програмною сумісністю, що відкриває можливість розробки кросплатформених систем.

Мікроконтролери STM32 стали еталоном інтеграції ARM Cortex у промисловій практиці. Як зазначено, «STM32 – це широкий спектр мікроконтролерів, розділених на дев’ять підсімейств, кожне з яких має свої особливості. Усі мікроконтролери STM32 мають ядро Cortex-M, а також деякі притаманні тільки STM характеристики (наприклад, прискорювач ART)». [1]

Інтеграція дозволяє реалізувати широкий набір функцій – від базових керуючих операцій до складних цифрових обчислень у реальному часі. Наприклад, ядра Cortex-M4 і Cortex-M7 підтримують апаратні блоки для обробки сигналів (DSP), що розширює можливості у сфері робототехніки, систем керування рухом та обробки аудіосигналів.

РОЗДІЛ 2
ТЕХНОЛОГІЯ КІНЦЕВИХ ПРИСТРОЇВ STM32

2.1 Кінцеві пристрої та апаратні інтерфейси STM32

У телекомунікаційних системах кінцеві пристрої є основними елементами, що забезпечують взаємодію між фізичним середовищем та цифровою інфраструктурою. Вони виконують функції збору, передачі, обробки, візуалізації та реагування на інформацію. До таких пристроїв належать сенсори, виконавчі механізми, дисплеї, модулі зв'язку, контролери доступу, вузли моніторингу та керування. Їх ефективна інтеграція в систему залежить від правильно підібраних апаратних інтерфейсів, які забезпечують стабільний обмін даними, синхронізацію, захист та енергозбереження.

Мікроконтролери STM32 здатні працювати з великим набором периферійних пристроїв. Завдяки цьому вони є універсальним інструментом для побудови систем комутації та управління. У межах даної роботи розглянуто підключення таких типів пристроїв, характеристики яких наведено в таблиці 2.1:

– сенсори температури та вологості (DS18B20, DHT22) – GPIO, 1-Wire;

– модулі бездротового зв'язку (SIM800L, ESP8266, NRF24L01) – UART, SPI;

– дисплеї (OLED SSD1306, TFT ILI9341) – SPI;

– акселерометри та гіроскопи (ADXL345, MPU6050) – I²C;

– зовнішні пам'яті (SD-карти, EEPROM) – SPI, SDIO;

– USB-пристрої – USB FS/HS;

– модулі живлення – окремі лінії з контролем напруги.

Таблиця 2.1 – Характеристики кінцевих пристроїв

Назва пристрою	Тип інтерфейсу	Напруга живлення	Швидкість передачі	Примітка
DS18B20	1-Wire (GPIO)	3.0–5.5 В	~16 біт/сек	Сенсор температури
SIM800L	UART	3.7–4.2 В	9600 бод	GSM-модуль
SSD1306	SPI	3.3 В	до 10 МГц	OLED-дисплей
ADXL345	I²C	3.3 В	до 400 кГц	Акселерометр
SD-карта	SPI / SDIO	3.3 В	до 25 МБ/с	Зовнішня пам'ять

На схемі в рисунку 2.1 зображено підключення кінцевих пристроїв до мікроконтролера STM32 через відповідні апаратні інтерфейси: SIM800L через UART, SSD1306 через SPI, ADXL345 через I²C, DS18B20 через GPIO, а також USB-з'єднання з персональним комп'ютером. Така конфігурація забезпечує ефективну комутацію даних, дозволяючи STM32 керувати сенсорами, дисплеєм, модулем зв'язку та зовнішнім інтерфейсом у реальному часі.

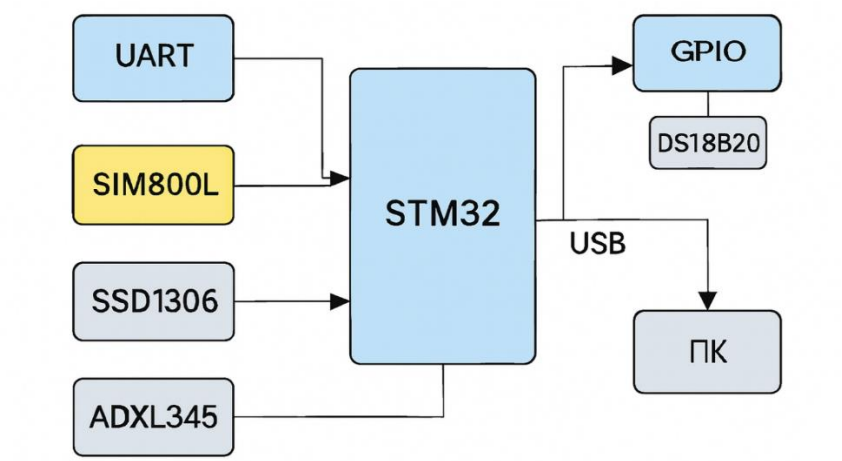


Рисунок 2.1 – Схема підключення кінцевих пристроїв до STM32

Кожен інтерфейс має свої переваги та обмеження. Наприклад, UART забезпечує просту реалізацію та стабільну передачу даних, що ідеально підходить для GSM-модулів. SPI дозволяє досягти високої швидкості обміну, критичної для дисплеїв та пам'яті. I²C забезпечує можливість підключення кількох пристроїв на одну шину, що зручно для сенсорних вузлів. USB використовується для інтеграції з ПК або іншими хостами, забезпечуючи високу пропускну здатність.

« Кожен пристрій збору та передачі інформації має у складі, як мінімум, один мікроконтролер, який виконує основні задачі по збору, обробці та передачі інформації ». [1]

У проєкті реалізовано підключення GSM-модуля SIM800L через UART1 (9600 бод), OLED-дисплея SSD1306 через SPI, сенсора DS18B20 через 1-Wire, акселерометра ADXL345 через I²C. Для захисту сигнальних ліній використано TVS-діоди, RC-фільтри, оптрони. Живлення – через AMS1117 та DC-DC перетворювачі. Усі лінії протестовано на перешкоди та втрати сигналу.

Інтерфейси STM32 підтримують DMA, що дозволяє передавати дані між периферією та пам’яттю без участі ядра. Це знижує навантаження на процесор та підвищує продуктивність. Для оптимізації роботи використано HAL та LL-бібліотеки. LL – для точного керування таймінгами SPI, HAL – для швидкої реалізації UART-комунікації.

У системі передбачено RS-485 трансивер – багатоточкова шина з диференціальним сигналом (до 32 пристроїв). Для синхронізації – апаратні таймери TIM, керування PWM-сигналами для виконавчих механізмів.

« STM32 підтримують кілька режимів енергоспоживання: Run mode; Sleep mode; Stop mode; Standby mode ». [4] Це дозволяє адаптувати систему до різних сценаріїв роботи – від активного обміну даними до енергозбереження в режимі очікування. Наприклад, сенсори можуть працювати в режимі Stop, а дисплей – в Run mode лише при активації.

У контексті телекомунікаційної системи кінцеві пристрої можуть бути розміщені на значній відстані. Враховано довжину ліній, тип кабелю, рівень сигналу, використання повторювачів. Застосовано екрановані кабелі, гальванічну розв’язку, диференціальні передавачі – для зменшення електромагнітних завад.

Реалізовано механізми автоматичного виявлення пристроїв – періодичне опитування шин UART та I²C, протоколи з автоконфігурацією. Це дозволяє масштабувати систему без ручного налаштування.

Таким чином, апаратна організація інтерфейсів STM32 у системі комутації кінцевих пристроїв базується на принципах надійності, масштабованості, енергоефективності та гнучкості. Вибір інтерфейсів обґрунтований технічними вимогами, а реалізація – адаптована до умов експлуатації. Це забезпечує стабільну роботу системи в реальному часі, з можливістю інтеграції в телекомунікаційні мережі, промислові середовища та IoT-інфраструктуру.

2.2 Апаратна організація процесора Cortex у STM32

Мікроконтролери STM32 побудовані на базі високопродуктивних ядер ARM Cortex, що забезпечують оптимальне співвідношення між швидкодією,

енергоефективністю та вартістю. Архітектура процесора Cortex у STM32 є типовим прикладом вбудованої системи з 32-бітним RISC ядром, оптимізованим для роботи в реальному часі. Вона включає процесорний блок, пам’ять програм і даних, систему шин, контролер переривань, системний таймер, периферійні інтерфейси та модулі керування живленням.

Ядра серії ARM Cortex-M розроблені спеціально для мікроконтролерів, що функціонують в умовах жорстких часових обмежень. « Архітектура ARM – це набір специфікацій, який описує роботу мікропроцесорного ядра ARM та містить моделі виконання, набір інструкцій, організацію та розташування пам’яті ». [1]. Основу архітектури становить концепція RISC (Reduced Instruction Set Computing), яка передбачає використання спрощеного набору інструкцій для досягнення максимальної швидкодії. У процесорах Cortex реалізовано тристадійний конвеєр, що дозволяє одночасно виконувати вибірку, декодування та виконання інструкцій. « ЦПУ Cortex-M3, також як і ARM7 / ARM9, використовує трьохступеневий конвеєр. Проте Cortex-M3 підтримує логіку передбачення переходів, що мінімізує кількість перезавантажень конвеєра ». [4]

Крім того, ядра Cortex-M підтримують набір інструкцій Thumb-2, який поєднує 16- та 32-бітні команди. Це забезпечує високу щільність коду при збереженні повної швидкодії 32-бітного процесора. « Thumb-2 – обширний набір інструкцій, орієнтований на компілятори мов C / C++, що дозволяє програмістам створювати повністю Сі-сумісні програми для мікроконтролерів Cortex ». [4]

У мікроконтролерах STM32 найчастіше використовуються ядра Cortex-M3, M4 або M7, які відрізняються рівнем продуктивності та підтримкою додаткових функцій, таких як блок плаваючої коми (FPU), цифрова обробка сигналів (DSP), розширені режими енергозбереження та апаратне прискорення.

Важливою особливістю архітектури є поділ 4-гігабайтного адресного простору на окремі області для коду, оперативної пам’яті, периферії та системних регістрів. « 4-гігабайтний адресний простір Cortex-M3 розділений на чітко визначені області коду програми, статичного ОЗУ, пристроїв вводу-виводу та системних ресурсів ». [4]

Система шин STM32 реалізована за принципами Гарвардської архітектури, що дозволяє одночасний доступ до пам’яті команд і пам’яті даних. Це досягається через окремі канали:

– I-Bus (Instruction bus) – для вибірки інструкцій із Flash-пам’яті;

– D-Bus (Data bus) – для читання та запису даних;

– S-Bus (System bus) – для доступу до периферійних пристроїв і системних регістрів.

Для ефективного обміну між ядром і периферією використовується матриця шин AHB / APB, яка дозволяє зменшити затримки та уникнути арбітражу. « Система шин STM32 побудована у вигляді багатоканальної матриці, де кожен майстер (CPU, DMA, переривання) має власний доступ до пам’яті та периферії, що мінімізує арбітраж і збільшує швидкодію ». [4]

Одним із ключових елементів апаратної організації є блок прямого доступу до пам’яті (DMA). Він дозволяє передавати дані між периферійними пристроями та пам’яттю без участі центрального процесора. Це особливо важливо для обробки потокових даних, таких як аудіо, відео або телеметрія, де затримка неприпустима.

На рисунку 2.2 показано два варіанти організації обміну даними між периферійним пристроєм і пам’яттю:

1. Передача без DMA: дані проходять через процесор, що створює додаткове навантаження, затримки та знижує загальну продуктивність системи.

2. Передача з DMA: дані передаються безпосередньо між периферією та пам’яттю, а процесор виконує інші задачі паралельно.

Такий підхід дозволяє розвантажити ядро Cortex-M, зменшити накладні витрати та забезпечити ефективну обробку потокових даних у реальному часі. Як зазначено в джерелі: « Цей контролер дозволяє периферійним пристроям мікроконтролера отримувати доступ до внутрішньої пам’яті без втручання ядра процесора ». [11]

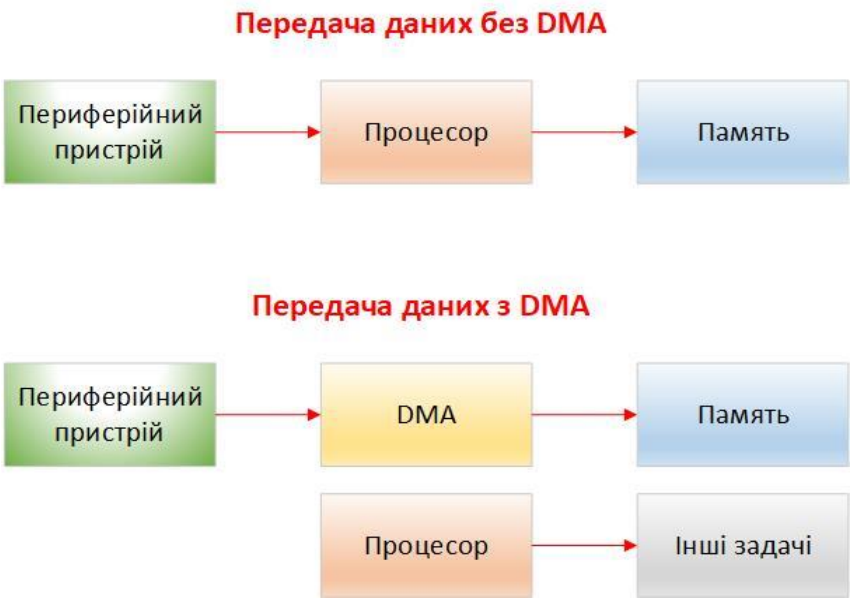


Рисунок 2.2 – Порівняння передачі даних з DMA і без DMA [2]

Обробка асинхронних подій у STM32 здійснюється через контролер переривань NVIC (Nested Vectored Interrupt Controller). « NVIC має можливість обробляти як сигнали, що надходять від периферійних пристроїв, так і винятки, що генеруються ядром процесора, забезпечуючи пріоритетне обслуговування подій ». [1] Контролер підтримує до 240 джерел переривань з програмно налаштовуваними пріоритетами, апаратне маскування, вкладеність та швидке реагування – перша інструкція обробника виконується вже через 12 тактів після запиту.

Для організації переривань використовується таблиця векторів, яка зберігає адреси всіх обробників. Це дозволяє системі швидко і детерміновано реагувати на зовнішні сигнали, що критично для систем реального часу.

У кожному процесорі Cortex-M інтегровано 24-бітний системний таймер SysTick, який використовується для генерації системних тік-сигналів, формування затримок, реалізації RTOS. « SysTick – це 24-розрядний таймер зворотного відліку, який використовується для забезпечення системного інтервалу часу (тіків) для RTOS ». [1] Таймер може працювати автономно або синхронно з системним тактовим генератором. У бібліотеках HAL він часто використовується для створення точних затримок без ОС.

Окрім SysTick, STM32 має таймери загального призначення (TIMx), розширені таймери для керування електродвигунами, RTC (Real-Time Clock) для відліку часу, а також Watchdog для контролю стабільності виконання програми.

Підсистема пам’яті STM32 включає Flash-пам’ять, SRAM, а також можливість підключення зовнішньої пам’яті через FSMC або QSPI. Архітектура Cortex підтримує буфер попередньої вибірки (prefetch buffer), кеш інструкцій, механізм вирівнювання даних, що забезпечує ефективне використання ресурсів. « Cortex-M має можливість оперувати з фрагментованими даними, що гарантує максимальну ефективність використання внутрішнього статичного ОЗУ ». [4] Ключові компоненти системи представлено в таблиці 2.2.

Таблиця 2.2 – Основні апаратні компоненти процесора Cortex у STM32

Компонент / Підсистема	Призначення / Опис	Особливості реалізації в STM32
Ядро ARM Cortex-M	Центральний процесор з архітектурою RISC	Тристадійний конвеєр; логіка передбачення переходів; висока швидкодія
Набір інструкцій Thumb-2	Компресія коду без втрати продуктивності	Поєднання 16- та 32-бітних інструкцій, оптимізованих для C / C++
Адресний простір (4 ГБ)	Розподіл пам’яті	Структурована карта пам’яті: код, дані, периферія, системні регістри
Система шин (I-Bus, D-Bus, S-Bus)	Обмін між ядром, пам’яттю та периферією	Гарвардська архітектура, паралельний доступ
Матриця шин AHB / APB	Взаємодія між ядром, DMA, периферією	Мінімальні затримки, паралельна робота
DMA	Передача даних без участі ЦП	Потокова передача аудіо, відео, телеметрії
NVIC	Керування перериваннями	До 240 джерел, пріоритети, вкладеність, реакція за 12 тактів
SysTick таймер	Системний таймер	24-бітний, підтримка HAL, RTOS
Таймери (TIMx, RTC, Watchdog)	Керування подіями та часом	PWM, реальний час, сторожові таймери
Підсистема пам’яті	Зберігання коду та даних	Flash, SRAM, FSMC / QSPI, кеш, prefetch
Інтерфейси вводу-виводу	Комунікація з периферією	UART, SPI, I²C, CAN, USB; підтримка DMA і переривань
Модулі енергокерування	Зменшення споживання енергії	Sleep, Stop, Standby; масштабування частоти

Архітектура Cortex у STM32 тісно інтегрована з периферією. Інтерфейси UART, SPI, I²C, CAN, USB безпосередньо підключені до матриці шин і можуть генерувати переривання або запити DMA. Це дозволяє системі обробляти

комунікаційні події паралельно з виконанням основного коду, забезпечуючи багатозадачність і високу продуктивність.

2.3 Схема включення та початкові режими STM32

Схема включення мікроконтролера STM32 є критично важливою складовою проєктування вбудованих систем, оскільки визначає стабільність роботи, надійність запуску та передбачуваність поведінки пристрою в початкових режимах. Вона охоплює електричне підключення, конфігурацію джерел живлення, запуск тактових генераторів, процедури скидання, вибір джерела завантаження, а також налаштування енергетичних режимів. Від правильності реалізації цих етапів залежить функціонування як апаратної, так і програмної частини системи.

Мікроконтролери STM32 підтримують роботу в широкому діапазоні напруги живлення – від 1,8 до 3,6 В, що забезпечує їх застосування в побутових, промислових та автономних пристроях. Основні виводи живлення включають VDD (живлення цифрової логіки), VDDA (живлення аналогових модулів, таких як АЦП і ЦАП), VREF+ (опорна напруга для АЦП) та VBAT (резервне живлення для годинника реального часу). Для забезпечення стабільної роботи рекомендовано встановлювати керамічні конденсатори ємністю 100 нФ безпосередньо біля кожного виводу VDD, а для VDDA – додатково конденсатори ємністю 1-4,7 мкФ. Також доцільним є використання LC-фільтра для аналогового живлення, що дозволяє мінімізувати високочастотні завади від цифрової частини.

У схемах із автономним живленням часто застосовуються стабілізатори типу LDO або DC/DC-перетворювачі, які підтримують стабільну напругу навіть за значних коливань навантаження. Для зменшення шумів у колах живлення рекомендується розділення цифрових і аналогових доменів, що підвищує точність вимірювань та зменшує ризик перешкод у критичних вузлах.

Після подачі живлення мікроконтролер автоматично переходить у початковий режим, активуючи внутрішній RC-генератор HSI, який забезпечує базову частоту 8 або 16 МГц. Він використовується для початкової ініціалізації

системи, після чого користувач може програмно перемкнути систему на зовнішній генератор HSE (High-Speed External) або на PLL (Phase-Locked Loop), який дозволяє масштабувати частоту до необхідного значення. « Після скидання ядро STM32 запускається від HSI, а перемикання на інше джерело здійснюється програмно після стабілізації зовнішнього генератора » [4].

Тактова система STM32 включає декілька джерел: HSI, HSE, LSI (внутрішній низькочастотний генератор), LSE (зовнішній кварц для RTC) та PLL. Кожне з джерел може бути вибрано як основне або допоміжне для окремих доменів – ядра, шин AHB / APB, периферійних блоків. Універсальна структура RCC (Reset and Clock Control) дозволяє конфігурувати дільники частот, маршрутизацію сигналів та вибір джерел для кожного функціонального блоку, що забезпечує оптимальний баланс між швидкодією та енергоспоживанням.

Механізм скидання (Reset) гарантує переведення мікроконтролера у відомий початковий стан та відновлення стабільної роботи після збоїв. Існує кілька типів скидання: Power-On Reset (POR), який активується при первинній подачі живлення; Brown-Out Reset (BOR), що спрацьовує при падінні напруги нижче встановленого порогу; External Reset (NRST), ініційований апаратним сигналом; Watchdog Reset (IWDG, WWDG), що активується при зависанні програми або перевищенні часу тайм-ауту; та Software Reset, який виконується через запис у спеціальний регістр RCC. « При спрацюванні сигналу RESET усі регістри, крім деяких системних, отримують значення за замовчуванням, а виконання програми починається з адреси вектору скидання » [1].

Для підвищення надійності на лінію NRST додають RC-коло із кнопкою ручного скидання, а також TVS-діод для захисту від електростатичних розрядів. Сигнал скидання повинен бути активним не менше 20 мкс, щоб гарантувати повне перезавантаження ядра.

Одразу після скидання STM32 перевіряє стан виводів BOOT0 та BOOT1. Логічні рівні на цих контактах визначають джерело, з якого мікроконтролер завантажує початковий код – з основної Flash-пам’яті, системної пам’яті або SRAM. У системній пам’яті STM32 знаходиться System Loader – вбудований завантажувач, який дозволяє оновлювати прошивку через стандартні інтерфейси

(UART, USB, SPI) без використання зовнішнього програматора (табл. 2.3). Така функція є особливо корисною при серійному виробництві або дистанційному оновленні програмного забезпечення.

Таблиця 2.3 – Режими завантаження мікроконтролера STM32

BOOT1	BOOT0	Режим завантаження	Опис
X	0	Flash memory	Звичайна робота користувацької програми
0	1	System memory	Завантаження через UART, USB або SPI
1	1	SRAM	Тестовий або налагоджувальний режим

На рисунку 2.3 зображено типову апаратну реалізацію конфігурації BOOT0 та BOOT1 мікроконтролера STM32. Кожен із BOOT-пінів підключено через резистор 10 кОм до землі, а також до перемикача, який дозволяє подати логічну «1» через VDD. Така схема забезпечує гнучкий вибір режиму завантаження – з Flash-пам’яті, системної пам’яті або SRAM – без потреби перепрошивки або змін у програмному коді. Вона особливо корисна на етапі налагодження, серійного виробництва або при впровадженні функції аварійного оновлення прошивки.

Після визначення режиму завантаження мікроконтролер виконує стандартну послідовність ініціалізації: зчитування таблиці векторів, завантаження основного вказівника стека (MSP), виклик обробника Reset_Handler, ініціалізацію сегментів пам’яті (.data, .bss), виконання функції SystemInit() та передачу керування у функцію main(). Цей механізм реалізовано на рівні стандартних бібліотек CMSIS та HAL, що забезпечує уніфікований процес старту для всіх серій STM32.

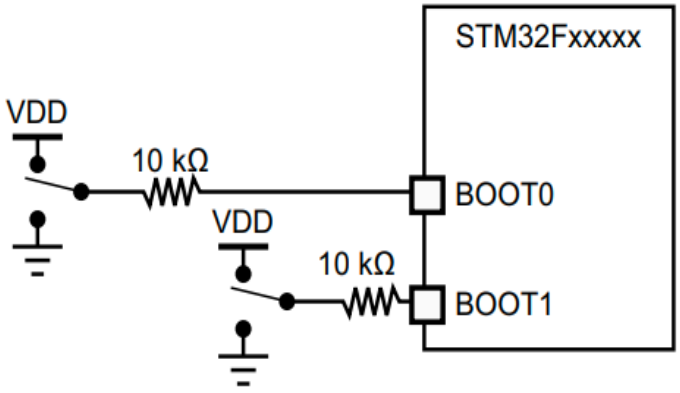


Рисунок 2.3 – Конфігурація BOOT-пінів для вибору режиму завантаження STM32

Після запуску користувач може обрати один із кількох режимів енергоспоживання, залежно від вимог до продуктивності та автономності: Run mode – повноцінна робота процесора на заданій частоті; Sleep mode – процесор зупиняється, периферія залишається активною; Stop mode – зупиняються більшість підсистем, але зберігається вміст пам’яті; Standby mode – мінімальне споживання, активним залишається лише RTC.

Для переходу між режимами енергоспоживання використовуються регістри системного контролю (SCR) та бібліотечні функції HAL_PWR_EnterSLEEPMode(), HAL_PWR_EnterSTOPMode(). Додатково може бути активований RTC wake-up або EXTI interrupt, який пробуджує систему з низькоенергетичного стану.

Для підвищення надійності стартової послідовності у деяких серіях STM32 (наприклад, STM32H7 або STM32L4) реалізовано функцію Boot Checksum, яка виконує контрольну перевірку вмісту Flash-пам’яті перед передачею керування користувацькому коду [4]. Також можливе програмне відстеження стабільності генераторів і напруги живлення за допомогою функцій HAL_RCC_GetFlag() та HAL_PWR_GetFlag(), що дозволяє що дозволяє своєчасно виявити потенційні збої на ранніх етапах запуску системи та вжити відповідних заходів для їх усунення. Такі діагностичні механізми є особливо важливими в критичних застосуваннях, де стабільність та передбачуваність поведінки мікроконтролера мають вирішальне значення.

Схема включення STM32 поєднує простоту реалізації з високою надійністю. Завдяки гнучкому вибору джерел живлення, модульній системі тактування, захисту від перенапруги та апаратним механізмам скидання, мікроконтролери STM32 здатні стабільно працювати у найрізноманітніших умовах. Вбудовані режими завантаження спрощують оновлення прошивки, а продумана система енергозбереження забезпечує мінімальне споживання енергії у спокої. Саме ці властивості роблять STM32 універсальною платформою для побудови сучасних комутаційних систем, сенсорних вузлів і телекомунікаційних пристроїв.

РОЗДІЛ 3
ІНТЕГРАЦІЯ КІНЦЕВИХ ПРИСТРОЇВ STM32 У ТЕЛЕКОМУНІКАЦІЙНІ СИСТЕМИ

3.1 Послідовні комунікаційні інтерфейси

У сучасних телекомунікаційних системах, що базуються на мікроконтролерах, одним із ключових аспектів є забезпечення стабільного, детермінованого та енергоефективного обміну даними між функціональними модулями. Послідовні комунікаційні інтерфейси відіграють фундаментальну роль у реалізації цієї взаємодії, оскільки вони забезпечують фізичний та логічний рівень зв’язку між мікроконтролером і периферійними пристроями, а також між мікроконтролерами в межах одного вузла або між вузлами мережі. У контексті побудови системи комутації кінцевих пристроїв на базі STM32, вибір, конфігурація та оптимізація послідовних інтерфейсів визначають загальну продуктивність, масштабованість і надійність системи.

Мікроконтролери STM32, розроблені компанією STMicroelectronics, підтримують широкий спектр апаратних інтерфейсів, що дозволяють реалізувати як прості, так і складні схеми комунікації. До найпоширеніших послідовних інтерфейсів належать UART, USART, SPI та I²C. Кожен із них має специфічні характеристики, які слід враховувати при проєктуванні телекомунікаційного вузла. Вибір конкретного інтерфейсу залежить від вимог до швидкості передачі, типу периферійного пристрою, енергоспоживання, топології мережі та необхідного рівня надійності.

Рисунок 3.1 ілюструє архітектурну модель взаємодії STM32 з периферією через послідовні інтерфейси. Він демонструє, що:

– STM32 може одночасно підтримувати кілька типів інтерфейсів, що забезпечує гнучкість у проєктуванні телекомунікаційних систем;

– кожен інтерфейс має свою логіку, електричні лінії та протокол передачі, що дозволяє підключати різноманітні пристрої – від сенсорів до модемів;

– така структура є основою для масштабованих, енергоефективних та надійних систем, де мікроконтролер виконує роль комунікаційного вузла.

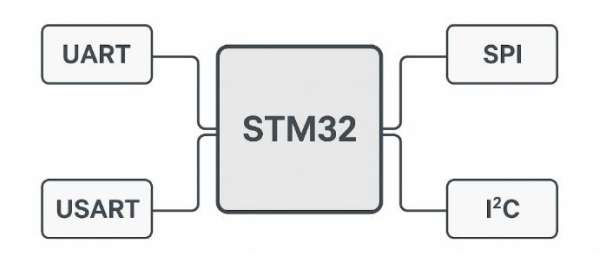


Рисунок 3.1 – Архітектура взаємодії STM32 з послідовними комунікаційними інтерфейсами UART, USART, SPI та I²C

Для наочного порівняння продуктивності послідовних інтерфейсів у STM32 на рисунку 3.2 наведено графік, що ілюструє максимальну швидкість передачі даних для кожного з них.

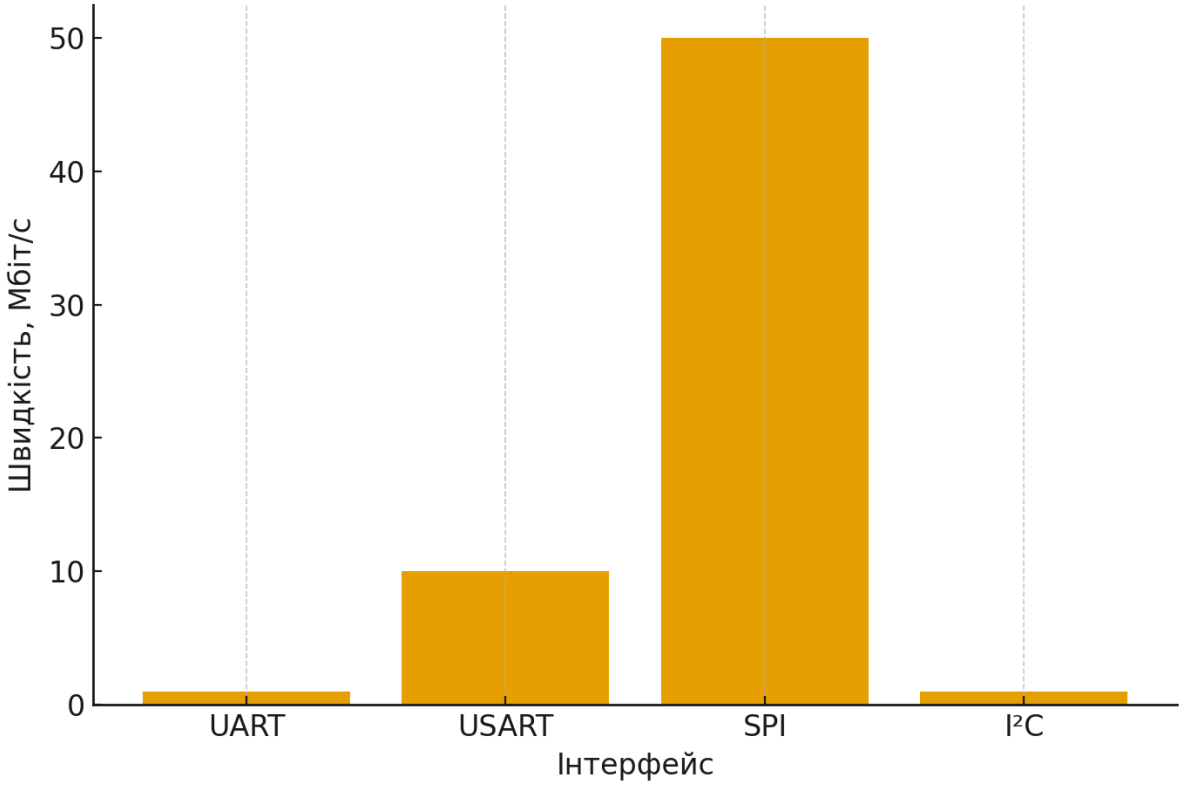


Рисунок 3.2 – Порівняння швидкості передачі даних між послідовними інтерфейсами STM32

Графік на рисунку 3.2 демонструє максимальні теоретичні швидкості передачі даних для основних послідовних інтерфейсів, реалізованих у мікроконтролерах STM32. Як видно, інтерфейс SPI забезпечує найвищу пропускну здатність – до 50 Мбіт/с, що робить його оптимальним для

високошвидкісного обміну з дисплеями, пам’яттю та АЦП. USART, завдяки підтримці синхронного режиму, досягає до 10 Мбіт/с і використовується для стабільного зв’язку між мікроконтролерами. UART та I²C мають нижчу швидкість – до 1 Мбіт/с, однак залишаються ефективними для простих периферійних пристроїв, таких як датчики, модулі GPS або RTC. Така візуалізація дозволяє швидко оцінити продуктивність кожного інтерфейсу та обґрунтовано обрати відповідний протокол для конкретного телекомунікаційного завдання.

Інтерфейс UART (Universal Asynchronous Receiver Transmitter) є одним із базових засобів асинхронної передачі даних. Його перевагою є простота реалізації, мінімальна кількість необхідних ліній зв’язку та широке поширення серед периферійних пристроїв. У STM32 UART реалізовано апаратно, з підтримкою буферизації, генерації переривань та прямого доступу до пам’яті (DMA), що дозволяє зменшити навантаження на центральне ядро та забезпечити ефективну обробку даних у реальному часі. « UART – це простий асинхронний послідовний протокол зв’язку, який використовує дві лінії: TX для передачі та RX для прийому даних » [5].

USART (Universal Synchronous/Asynchronous Receiver Transmitter) є розширеним варіантом UART, який підтримує як асинхронний, так і синхронний режим передачі. У синхронному режимі до стандартних ліній TX та RX додається лінія тактування, що дозволяє забезпечити точну синхронізацію між передавачем і приймачем. Це особливо важливо у системах, де передача даних здійснюється з високою частотою або в умовах жорстких вимог до часових параметрів. « USART може працювати в синхронному режимі, забезпечуючи тактовий сигнал для синхронізації передачі даних між пристроями » [6]. У STM32 реалізовано апаратну підтримку USART з можливістю конфігурації режимів, пріоритетів переривань, використання DMA та інтеграції з RTOS.

Інтерфейс SPI (Serial Peripheral Interface) є синхронним протоколом, який забезпечує швидкий обмін даними між головним пристроєм (master) та підлеглими (slave). У STM32 SPI реалізовано з можливістю налаштування режимів передачі, полярності та фази сигналу, а також з підтримкою DMA та багатобайтових буферів. Висока швидкість передачі, простота реалізації та

підтримка повнодуплексного режиму роблять SPI оптимальним вибором для зв’язку з дисплеями, зовнішньою пам’яттю, сенсорами, АЦП, ЦАП та іншими модулями, що потребують швидкої реакції. « SPI – це протокол повнодуплексного зв’язку, який дозволяє одночасну передачу та прийом даних » [7].

Інтерфейс I²C (Inter-Integrated Circuit) є двопровідним синхронним протоколом, який дозволяє підключати до одного мікроконтролера велику кількість периферійних пристроїв. Його головною перевагою є наявність адресації, що дозволяє уникнути конфліктів при передачі даних. У STM32 реалізовано апаратну підтримку I²C з можливістю роботи в режимах master / slave, генерацією переривань, підтримкою швидкісного режиму (Fast Mode, Fast Mode Plus) та використанням DMA. «I²C використовує лише дві лінії (SCL і SDA) та підтримує кілька пристроїв через унікальну адресацію» [5]. Завдяки цьому інтерфейс активно використовується для підключення датчиків, EEPROM, RTC та інших низькошвидкісних пристроїв.

У спеціалізованих телекомунікаційних рішеннях можуть застосовуватись менш поширені інтерфейси, такі як LIN (Local Interconnect Network), IrDA (Infrared Data Association) та SmartCard. LIN використовується переважно в автомобільних системах для зв’язку між модулями з низькою швидкістю передачі, де важлива простота та низька вартість реалізації. IrDA дозволяє реалізувати бездротову передачу даних через інфрачервоний канал, що може бути корисним у системах, де необхідна гальванічна розв’язка або обмежений доступ до фізичних з’єднань. SmartCard інтерфейс забезпечує взаємодію з картками доступу, що актуально для систем безпеки, контролю доступу та ідентифікації, зокрема в телекомунікаційних вузлах, що інтегрують функції автентифікації.

Конфігурація послідовних інтерфейсів у STM32 здійснюється за допомогою програмного середовища STM32CubeMX, яке дозволяє обрати потрібний режим роботи, призначити функції пінів, активувати переривання, налаштувати швидкість передачі та згенерувати початковий код. У середовищі STM32CubeIDE реалізовано підтримку HAL-бібліотек, які спрощують програмування інтерфейсів, забезпечуючи доступ до функцій передачі, прийому, обробки помилок, керування

буферами та взаємодії з DMA. « STM32CubeMX надає графічний інтерфейс для конфігурації периферії та генерації коду ініціалізації » [8].

У складних системах доцільно використовувати операційні системи реального часу, такі як FreeRTOS, що дозволяє організувати багатозадачну обробку комунікаційних подій, розподіл пріоритетів між задачами та забезпечити детерміновану поведінку системи. У STM32 реалізовано апаратну підтримку контекстного перемикання, таймерів та механізмів синхронізації, що дозволяє ефективно інтегрувати послідовні інтерфейси у багатозадачне середовище.

У контексті побудови системи комутації кінцевих пристроїв на базі STM32, послідовні комунікаційні інтерфейси відіграють роль основного каналу обміну даними між мікроконтролером та периферією. Їх правильна конфігурація, оптимальний вибір, апаратна підтримка та інтеграція у загальну архітектуру системи дозволяють забезпечити стабільну, швидку та енергоефективну передачу інформації, що є критично важливим для сучасних телекомунікаційних систем.

3.2 Мережеві та спеціалізовані інтерфейси

У телекомунікаційних системах, що базуються на мікроконтролерах, мережеві та спеціалізовані інтерфейси відіграють ключову роль у забезпеченні масштабованості, гнучкості та інтеграції з зовнішніми інформаційними середовищами. На відміну від базових послідовних інтерфейсів, які здебільшого використовуються для локальної взаємодії з периферією, мережеві інтерфейси дозволяють мікроконтролерам функціонувати як повноцінні вузли телекомунікаційної інфраструктури. У мікроконтролерах STM32 реалізовано низку таких інтерфейсів, зокрема Ethernet, USB, CAN, а також підтримку протоколів TCP/IP, HID, CDC, RNDIS та інших, що забезпечує широкі можливості для побудови комунікаційних систем різного рівня складності.

Рисунок 3.2 ілюструє архітектурну модель взаємодії STM32 з мережевими та спеціалізованими інтерфейсами.

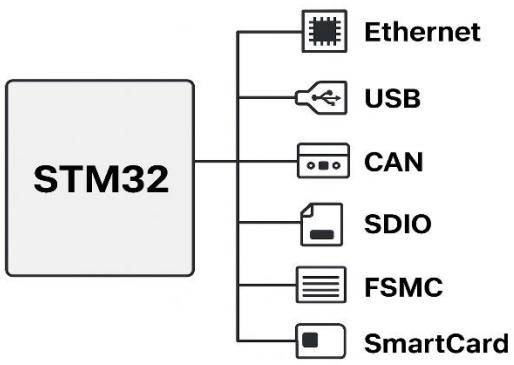


Рисунок 3.2 – Архітектура взаємодії STM32 з мережевими та спеціалізованими інтерфейсами Ethernet, USB, CAN, SDIO, FSMC та SmartCard [8]

У центрі зображено мікроконтролер STM32, від якого відходять лінії до шести типових інтерфейсів:

- Ethernet – для підключення до локальної мережі або хмарних платформ;
- USB – для зв’язку з ПК, оновлення прошивки, віртуальних COM-портів;
- CAN – для промислових шин, телеметрії та систем автоматизації;
- SDIO – для взаємодії з картками пам’яті;
- FSMC – для підключення дисплеїв та зовнішньої пам’яті;
- SmartCard – для автентифікації, безпеки та ідентифікації.

Ця схема демонструє, що STM32 може виступати як багатофункціональний вузол телекомунікаційної системи, здатний до інтеграції з різними типами периферії та інформаційних середовищ.

Одним із найважливіших мережевих інтерфейсів є Ethernet, який дозволяє мікроконтролеру безпосередньо підключатися до локальної мережі без використання зовнішніх шлюзів або контролерів. У STM32 реалізовано апаратну підтримку Ethernet MAC-контролера, що забезпечує обробку кадрів, контроль CRC, управління буферами, пріоритетами та взаємодію з PHY-чипом. « STM32 підтримує Ethernet через вбудований MAC-контролер, який дозволяє реалізувати стек TCP/IP без зовнішнього процесора » [5]. Це відкриває можливість створення пристроїв, які можуть обмінюватися даними з серверами, хмарними платформами або іншими вузлами мережі, включаючи IoT-системи, SCADA-вузли та інтелектуальні сенсорні мережі.

Інтерфейс USB також є критично важливим для телекомунікаційних застосувань, особливо в контексті оновлення прошивки, передачі даних, реалізації віртуальних COM-портів або підключення до ПК. У STM32 реалізовано підтримку USB Full-Speed (до 12 Мбіт/с) та High-Speed (до 480 Мбіт/с), а також апаратну реалізацію стеку USB Device. « USB-інтерфейс у STM32 дозволяє реалізувати класи пристроїв CDC, HID, MSC, що забезпечує гнучкість у побудові систем зв'язку з комп'ютером » [8]. Крім того, можливе використання USB для реалізації Ethernet over USB (CDC-RNDIS), що дозволяє створити віртуальний мережевий інтерфейс без фізичного Ethernet-порту, що особливо актуально для мобільних або енергообмежених пристроїв.

CAN (Controller Area Network) є промисловим інтерфейсом, який широко використовується в автомобільній електроніці, системах автоматизації, телеметрії та енергетичних мережах. У STM32 реалізовано один або два CAN-модуля, які підтримують стандарт CAN 2.0A/B, апаратну фільтрацію кадрів, пріоритети повідомлень, контроль помилок та автоматичне повторне передавання. «CAN-порт у STM32 дозволяє реалізувати повноцінну шину обміну даними між мікроконтролерами з високою надійністю та низькою затримкою» [4]. Завдяки можливості переназначення пінів, CAN-інтерфейс легко інтегрується у друковану плату з урахуванням топології системи, а його стійкість до електромагнітних завад робить його придатним для використання в критичних умовах.

Окрім фізичних інтерфейсів, важливу роль відіграє підтримка мережевих протоколів. У STM32 можлива реалізація стеку TCP/IP за допомогою бібліотек lwIP або uIP, що дозволяє створювати сервери, клієнти, SNMP-агенти, DHCP-клієнти, DNS-резолвери та інші мережеві компоненти. « Завдяки підтримці lwIP, STM32 може працювати як веб-сервер, FTP-клієнт або SNMP-агент без використання зовнішнього процесора » [1]. Це особливо актуально для систем моніторингу, керування, збору даних у реальному часі, а також для реалізації RESTful API, MQTT-брокерів та інших сучасних протоколів взаємодії.

У спеціалізованих застосуваннях можуть використовуватись інтерфейси SmartCard, SDIO, FSMC, а також інтерфейси для взаємодії з дисплеями, сенсорними панелями, криптографічними модулями та картами пам'яті.

Наприклад, FSMC (Flexible Static Memory Controller) дозволяє підключати зовнішню статичну пам’ять або дисплеї з паралельним інтерфейсом, що розширює можливості STM32 у мультимедійних, графічних та візуалізаційних системах. « STM32 підтримує спеціалізовані інтерфейси для підключення дисплеїв, карт пам’яті, криптомодулів, що дозволяє створювати комплексні телекомунікаційні рішення » [2]. SDIO (Secure Digital Input Output) забезпечує високошвидкісну взаємодію з картками пам’яті SD, що є актуальним для систем запису логів, зберігання телеметрії та локального кешування даних.

Конфігурація мережевих інтерфейсів у STM32 здійснюється через програмне середовище STM32CubeMX, де можна обрати тип інтерфейсу, налаштувати пін-контакти, активувати стек протоколів, призначити пріоритети переривань та згенерувати код ініціалізації. У середовищі STM32CubeIDE реалізовано підтримку бібліотек HAL та middleware-компонентів, що дозволяє швидко інтегрувати Ethernet, USB, CAN, SDIO, FSMC та інші інтерфейси у проєкт. « STM32CubeMX дозволяє налаштувати мережеві інтерфейси, обрати стек lwIP або USB Device, а також згенерувати код для взаємодії з периферією » [8].

У складних телекомунікаційних системах доцільним є використання операційних систем реального часу (RTOS), таких як FreeRTOS, які дозволяють організувати багатозадачну обробку мережевих подій, розподіл ресурсів, синхронізацію потоків та детерміновану реакцію на зовнішні сигнали. У STM32 реалізовано апаратну підтримку контекстного перемикання, таймерів, механізмів взаємодії між задачами, що дозволяє ефективно інтегрувати мережеві інтерфейси у багатозадачне середовище. Ці дані дозволяють обґрунтовано обирати інтерфейс залежно від енергетичних обмежень системи, особливо у мобільних, автономних або сенсорних вузлах, де критично важлива оптимізація живлення.

Таким чином, мережеві та спеціалізовані інтерфейси мікроконтролерів STM32 є основою для побудови сучасних телекомунікаційних систем, які потребують високої швидкості обміну, надійності, гнучкості та інтеграції з різними протоколами. Їх правильна конфігурація, апаратна підтримка, програмна реалізація та адаптація до специфіки проєкту дозволяють створювати як локальні вузли, так і повноцінні мережеві пристрої, здатні до взаємодії з глобальними

інформаційними системами, хмарними платформами, промисловими контролерами та інтелектуальними сенсорними мережами.

3.3 Режими завантаження та взаємодія з іншими пристроями

У телекомунікаційних системах, побудованих на базі мікроконтролерів STM32, важливу роль відіграє не лише функціональність обробки та передавання даних, а й здатність пристрою адаптуватися до змін середовища, працювати в умовах обмеженого енергоспоживання та забезпечувати надійний запуск і взаємодію з іншими компонентами системи. У цьому контексті критичними є режими завантаження, механізми взаємодії з периферією, реалізація енергозберігаючих стратегій та засоби забезпечення стабільності роботи, які дозволяють досягти високої ефективності, безпеки та довговічності пристроїв.

Графік на рисунку 3.4 демонструє порівняння енергоспоживання різних мережевих та спеціалізованих інтерфейсів STM32, виміряне у міліамперах (мА). Найвищу споживану потужність мають Ethernet та FSMC (~70 мА), що пов’язано з високою пропускною здатністю та складністю обробки даних.

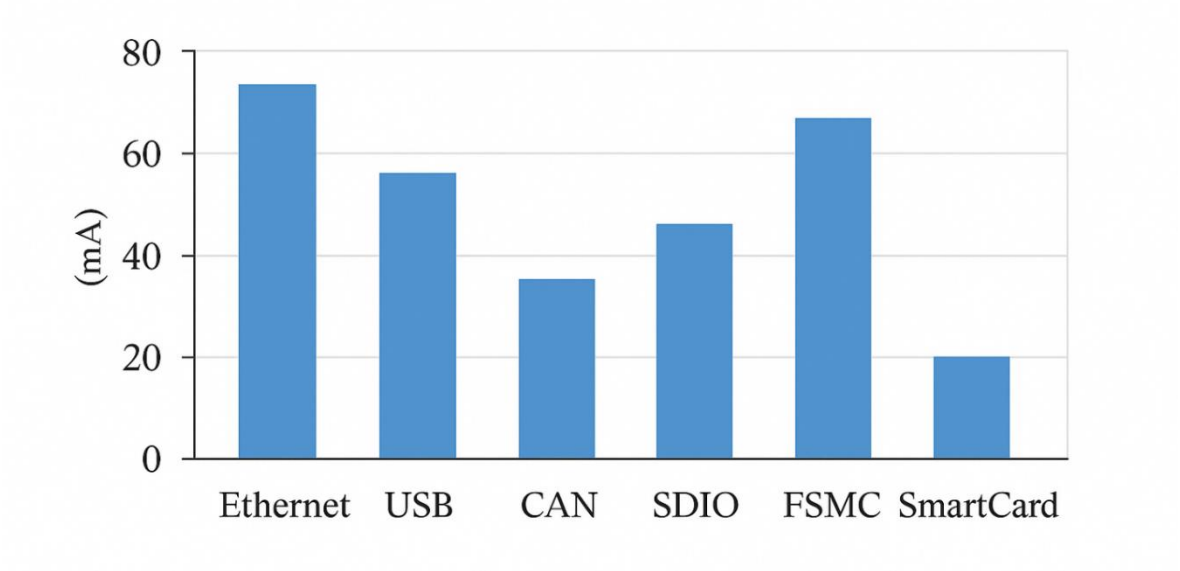


Рисунок 3.4 – Енергоспоживання мережевих та спеціалізованих інтерфейсів STM32

USB споживає близько 60 мА, SDIO – 50 мА, CAN – 40 мА, а SmartCard – лише 20 мА, що робить його найбільш енергоефективним серед представлених.

Мікроконтролери STM32 підтримують кілька режимів завантаження, які визначають джерело виконання коду після скидання. Вибір режиму здійснюється апаратно – через логічні рівні на пін-контактах BOOT0 та BOOT1. Залежно від конфігурації, мікроконтролер може запускати програму з внутрішньої Flash-пам’яті, системної пам’яті (System Memory) або оперативної пам’яті (SRAM). Як зазначено в офіційній документації, « стан логічних рівнів на входах BOOT0/BOOT1 визначає адресу стартового вектора після скидання: 0x08000000 для Flash, 0x1FFF0000 для системної пам’яті » [5].

Це дозволяє реалізувати як стандартний запуск програми, так і оновлення прошивки через UART, USB або SPI без використання зовнішнього програматора. У системах, де важлива можливість перепрошивки в польових умовах, така гнучкість є критичною. Наприклад, у пристроях, що працюють у важкодоступних місцях або в умовах обмеженого фізичного доступу, оновлення через системну пам’ять дозволяє уникнути втручання в апаратну частину. У багатьох промислових і телекомунікаційних системах така функціональність є основою для реалізації безперервного обслуговування та дистанційного адміністрування.

Після вибору джерела завантаження виконується послідовність ініціалізації, яка включає читання таблиці векторів, встановлення покажчика стеку, виклик функції Reset_Handler, ініціалізацію пам’яті та запуск основної програми. У бібліотеках CMSIS та HAL цей процес стандартизовано, що забезпечує уніфікований запуск для всіх серій STM32. Крім того, у деяких моделях реалізовано перевірку контрольної суми прошивки (CRC), що дозволяє виявити пошкодження коду ще до початку його виконання. Це підвищує надійність системи, особливо в критичних застосуваннях, де помилки в коді можуть призвести до збоїв у телекомунікаційній інфраструктурі.

Мікроконтролери STM32 мають широкий набір апаратних інтерфейсів, які забезпечують взаємодію з іншими пристроями – як периферійними, так і мережевими. До них належать UART, SPI, I²C, USB, CAN, Ethernet, GPIO, SDIO, FSMC та інші. Кожен інтерфейс може бути налаштований відповідно до вимог

конкретної системи, що дозволяє реалізувати як прості сенсорні вузли, так і складні комунікаційні платформи з багаторівневою логікою. Наприклад, UART використовується для зв’язку з модемами, GPS-модулями, Bluetooth-пристроями, а SPI – для підключення дисплеїв, зовнішньої пам’яті, сенсорів та ЦАП. USB-інтерфейс дозволяє реалізувати віртуальні COM-порти, а Ethernet – повноцінну мережеву взаємодію з підтримкою TCP/IP.

Взаємодія з периферією може здійснюватися як у режимі опитування, так і через переривання або DMA. Це дозволяє оптимізувати навантаження на процесор, зменшити затримки та забезпечити ефективну обробку даних у реальному часі. Як зазначено в інструкції з енергозбереження, « взаємодія з периферією повинна бути організована так, щоб активні модулі працювали лише за необхідності, а решта – перебували в режимі сну » [9]. У системах, де важлива синхронізація, застосовуються апаратні таймери, зовнішні генератори та протоколи з підтримкою часових міток. Наприклад, Ethernet-інтерфейс у STM32 дозволяє реалізувати протоколи з точним часом (PTP), що актуально для промислових мереж, а USB – для реалізації CDC, HID або MSC-класів.

Енергозбереження є одним із ключових аспектів при побудові телекомунікаційних пристроїв, особливо тих, що працюють автономно або мають обмежене джерело живлення. Мікроконтролери STM32 підтримують кілька режимів енергоспоживання: Run, Sleep, Stop, Standby та Shutdown. Кожен із них має свої характеристики щодо споживання струму, часу пробудження та доступності периферії. У режимі Sleep ядро зупиняється, але периферійні модулі можуть продовжувати працювати. У режимі Stop вимикається більшість внутрішніх блоків, зберігаючи лише необхідні для пробудження. Режим Standby забезпечує мінімальне споживання енергії, зберігаючи лише RTC та регістри резервного домену. « Режими Stop і Standby дозволяють суттєво знизити споживання енергії, зберігаючи при цьому можливість швидкого повернення до активного стану » [10].

Таблиця 3.1 узагальнює ключові режими роботи, енергозбереження, взаємодії з периферією та механізми забезпечення надійності, реалізовані у мікроконтролерах STM32.

Таблиця 3.1 – Режими завантаження, енергозбереження, взаємодії з периферією та механізми надійності STM32

Категорія	Режим / Механізм	Основні характеристики	Призначення / Переваги
Режими завантаження	Flash Boot	Завантаження основної програми з внутрішньої Flash-пам’яті (адреса 0x08000000)	Стандартний запуск користувацького ПЗ
	System Memory Boot	Запуск з системної пам’яті, підтримка UART/USB/SPI Bootloader	Оновлення прошивки без зовнішнього програматора
	SRAM Boot	Запуск з оперативної пам’яті (тимчасовий код)	Тестування та налагодження в середовищі розробки
Енерго-збереження	Run	Повна активність ядра та периферії	Нормальна робота системи
	Sleep	Зупинка ядра, активна периферія	Зниження енергоспоживання
	Stop	Зупинка більшості блоків, RTC та SRAM зберігаються	Баланс між споживанням і швидким пробудженням
	Standby	Активні лише RTC і wake-up пін	Мінімальне споживання енергії, автономні системи
	Shutdown	Повне вимкнення живлення ядра	Максимальне енергозбереження
Взаємодія з периферією	DMA	Автоматична передача даних без участі CPU	Зниження навантаження на ядро
	Interrupt Mode	Реакція на події у реальному часі	Зменшення затримки обробки
	Polling Mode	Циклічне опитування периферії	Простота реалізації
Надійність роботи	IWDG / WWDG	Незалежні сторожові таймери	Автоматичне відновлення після зависань
	Brown-Out Reset (BOR)	Скидання при падінні напруги	Захист пам’яті та логіки
	CRC перевірка	Контроль цілісності коду при запуску	Виявлення пошкодження прошивки
	Температурний моніторинг	Контроль перегріву ядра	Захист від критичних умов експлуатації
	Логування подій / дублювання	Програмна діагностика та резервування каналів зв’язку	Підвищення надійності телекомунікаційних систем

Основні завантажувальні режими (Flash, System Memory, SRAM) визначають джерело стартового коду та сценарії його використання. Режими енергозбереження (Run, Sleep, Stop, Standby, Shutdown) дозволяють гнучко балансувати між продуктивністю та споживанням енергії, що є критично важливим для автономних телекомунікаційних систем.У таблиці також наведено

механізми взаємодії з периферією – DMA, Interrupt Mode та Polling Mode – які забезпечують ефективну передачу даних з мінімальним навантаженням на центральне ядро. Окрему увагу приділено засобам забезпечення надійності: сторожовим таймерам (IWDG / WWDG), контролю напруги (BOR), перевірці цілісності коду (CRC), температурному моніторингу та логуванню подій.

Ці функції дозволяють підвищити стабільність роботи системи, забезпечити її самовідновлення після збоїв та адаптацію до критичних умов експлуатації.

Для керування енергоспоживанням у STM32 передбачено спеціальні регістри PWR та SCB, які дозволяють програмно перемикати режими, контролювати джерела пробудження та налаштовувати часові параметри переходу між станами. У середовищі STM32CubeMX реалізовано графічний інтерфейс для налаштування енергетичних режимів, що значно спрощує проєктування енергоефективних систем. « STM32CubeMX дозволяє налаштувати енергетичні параметри, активувати wake-up джерела та оптимізувати конфігурацію живлення » [8]. У системах з живленням від батарей або енергозалежних джерел доцільно реалізовувати динамічне масштабування частоти, активацію периферії за запитом та використання таймерів для контролю активності.

Надійність роботи системи забезпечується як апаратними, так і програмними засобами. До апаратних належать watchdog-таймери (IWDG, WWDG), механізми виявлення падіння напруги (Brown-Out Reset), захист від перенапруги, контроль температури, перевірка цілісності пам’яті та контроль тактових генераторів. Програмні засоби включають обробку винятків, контроль стану периферії, логування подій та автоматичне перезавантаження у разі виявлення помилок. У системах з високими вимогами до безперервності роботи доцільно реалізовувати дублювання функцій, резервування каналів зв’язку, періодичну перевірку цілісності даних та автоматичне перезавантаження у разі виявлення критичних помилок у виконанні програми, порушення логіки обробки даних або втрати зв’язку з периферійними модулями. Такий підхід дозволяє забезпечити автоматичне відновлення функціональності системи без втручання

оператора, що є особливо важливим для телекомунікаційних пристроїв, які працюють у безперервному режимі або в умовах обмеженого доступу.

3.4 Діагностика, обробка помилок та забезпечення відмовостійкості

У телекомунікаційних системах, що базуються на мікроконтролерах STM32, діагностика, обробка помилок та забезпечення відмовостійкості є критично важливими аспектами, які визначають стабільність, безпеку та довговічність роботи пристрою. У контексті комутації кінцевих пристроїв, де кожен вузол може бути точкою збору, передачі або обробки даних, здатність системи виявляти та реагувати на помилки є основою її надійності. STM32 надає широкий набір апаратних і програмних засобів для реалізації цих функцій, що дозволяє створювати системи з високим рівнем самодіагностики, захисту та автоматичного відновлення (табл. 3.2).

Таблиця 3.2 – Механізми діагностики, обробки помилок та забезпечення відмовостійкості у STM32.

Механізм	Призначення	Тип (апаратний / програмний)	Реакція системи при спрацюванні	Застосування у телекомунікаціях
IWDG / WWDG	Виявлення зависань програми	Апаратний	Автоматичний перезапуск MCU	Безнаглядні вузли, сенсори
Brown-Out Reset (BOR)	Захист при падінні напруги	Апаратний	Рестарт / блокування запису	Пристрої з нестабільним живленням
Exception Handlers (HardFault, BusFault)	Виявлення критичних помилок	Програмний	Логування, аварійне завершення	Програмна стабільність систем
CRC-модуль	Контроль цілісності даних	Апаратний	Повторна передача / перезапуск	Передача даних по мережі
Самодіагностика периферії	Перевірка стану АЦП, GPIO, таймерів	Програмний	Попередження / адаптація режимів	Промислові вузли, сенсорні системи
Зовнішня діагностика (UART, USB, Ethernet)	Передача логів і моніторинг	Програмно-апаратний	Надсилання повідомлень / SNMP / JSON	Системи моніторингу і керування
Readout Protection (RDP)	Захист пам’яті від несанкціонованого доступу	Апаратний	Блокування читання / запису	Безпечні мережеві контролери

Мікроконтролери STM32 оснащені низкою апаратних засобів, які дозволяють виявляти критичні помилки на ранніх етапах. До них належать watchdog-таймери (IWDG, WWDG), механізми виявлення падіння напруги (Brown-Out Reset), контроль температури, захист пам’яті та система обробки винятків. Watchdog-таймери дозволяють виявити зависання програми або порушення логіки виконання, автоматично перезапускаючи систему. Використання незалежного watchdog-таймера дозволяє гарантувати відновлення роботи системи навіть у разі критичних збоїв [10]. У телекомунікаційних пристроях, які працюють без постійного нагляду, це є основним механізмом самовідновлення.

Brown-Out Reset (BOR) активується при падінні напруги живлення нижче допустимого порогу, що дозволяє уникнути некоректного запису в пам’ять або порушення роботи периферії. У STM32 реалізовано кілька рівнів порогів BOR, які можна налаштувати відповідно до вимог системи. Це особливо актуально для пристроїв, що живляться від нестабільних джерел, таких як батареї або сонячні панелі. Крім того, у деяких серіях доступний температурний моніторинг, що дозволяє виявити перегрів і вжити заходів – наприклад, зниження частоти, активацію енергозберігаючого режиму або переведення в режим Stop. У системах, що працюють у промислових умовах, температурна стабільність є критичною для збереження точності та безпеки.

На рівні програмного забезпечення STM32 підтримує обробку винятків через систему переривань та спеціальні обробники помилок. До них належать HardFault_Handler, MemManage_Handler, BusFault_Handler, UsageFault_Handler та інші. Ці обробники активуються при виникненні критичних ситуацій – доступ до недозволеної області пам’яті, порушення вирівнювання, неправильне використання інструкцій тощо. « Обробники винятків у STM32 дозволяють локалізувати джерело помилки та виконати аварійне завершення або перезапуск системи » [11]. У складних системах доцільно реалізовувати логування помилок – збереження інформації про винятки у зовнішню пам’ять або передачу через

UART / USB. Це дозволяє аналізувати причини збоїв та вдосконалювати прошивку. Наприклад, при виникненні HardFault можна зчитати регістри стеку, визначити адресу інструкції, що спричинила помилку, та передати ці дані на сервер діагностики або зберегти у EEPROM.

Відмовостійкість – це здатність системи зберігати працездатність у разі часткових збоїв або несправностей. У STM32 реалізовано кілька підходів до забезпечення цієї властивості. По-перше, це дублювання критичних функцій – наприклад, використання двох незалежних каналів зв'язку (UART + CAN), резервних джерел живлення (VBAT + VDD), або паралельних таймерів для контролю подій. По-друге, це використання CRC-контролю для перевірки цілісності даних, як у Flash-пам'яті, так і при передачі по інтерфейсах. « CRC-модуль у STM32 дозволяє обчислювати контрольні суми для блоків пам'яті або потоку даних, що забезпечує виявлення пошкоджень » [5]. У телекомунікаційних системах CRC використовується для перевірки конфігураційних файлів, прошивки, пакетів даних, що надходять з мережі. У разі виявлення невідповідності система може автоматично запустити процедуру повторного завантаження, повідомити про помилку або запитати повторну передачу.

Ще одним важливим елементом є механізми самодіагностики. У STM32 можлива реалізація періодичних тестів периферії – перевірка АЦП, таймерів, пам'яті, GPIO, внутрішніх генераторів. Наприклад, система може зчитувати еталонне значення з внутрішнього джерела напруги та порівнювати його з очікуваним, виявляючи зсув або нестабільність. « Автоматичне тестування периферії дозволяє виявити деградацію компонентів або порушення конфігурації без участі оператора » [12]. У системах з високими вимогами до точності – наприклад, у сенсорних вузлах або медичних пристроях – така перевірка дозволяє підтримувати стабільність вимірювань.

У телекомунікаційних системах важливо забезпечити не лише локальну діагностику, а й можливість взаємодії з зовнішніми системами моніторингу. У STM32 це реалізується через інтерфейси UART, USB, Ethernet, CAN, які дозволяють передавати діагностичні дані на сервер, ПК або інший вузол. Наприклад, через USB CDC можна реалізувати віртуальний COM-порт, який

передає логи помилок у реальному часі. Через Ethernet – надсилати SNMP-повідомлення, HTTP-запити або JSON-пакети з діагностичною інформацією. У системах з критичними вимогами до безпеки доцільно реалізовувати механізми підтвердження працездатності – heartbeat-сигнали, watchdog-перевірки, контроль циклів обробки. Це дозволяє виявити не лише апаратні збої, а й логічні помилки – зависання, порушення таймінгів, некоректну поведінку алгоритмів, втрату зв'язку з центральним вузлом або порушення протоколу обміну.

У STM32 також реалізовано механізми захисту пам'яті, зокрема Readout Protection (RDP), що дозволяє обмежити доступ до внутрішньої Flash-пам'яті. Це актуально для систем, де важлива безпека – наприклад, у пристроях доступу, промислових контролерах або медичних системах. «Захист пам'яті STM32 дозволяє запобігти несанкціонованому читанню або модифікації прошивки» [9]. У поєднанні з апаратним захистом від запису, контрольними сумами та шифруванням даних, це дозволяє створювати телекомунікаційні пристрої з високим рівнем інформаційної безпеки.

Таким чином, діагностика, обробка помилок та забезпечення відмовостійкості є фундаментальними складовими побудови телекомунікаційних систем на базі STM32. Завдяки поєднанню апаратних засобів (watchdog-таймери, механізми контролю напруги, CRC-модулі, обробники винятків) та програмних стратегій (логування, самотестування, резервування, взаємодія з зовнішніми діагностичними системами), можливо створити пристрої, здатні до автономного виявлення, локалізації та усунення збоїв. Це забезпечує стабільну роботу системи навіть у складних умовах експлуатації, підвищує її життєвий цикл та зменшує потребу в технічному обслуговуванні. Такий підхід є критично важливим для мережевих вузлів, сенсорних платформ, систем моніторингу, керування та безпеки, що функціонують у рамках сучасної телекомунікаційної інфраструктури.

РОЗДІЛ 4
НАДІЙНІСТЬ, ТЕСТУВАННЯ АПАРАТНОЇ ЧАСТИНИ ТА РЕКОМЕНДАЦІЇ З ПРОЄКТУВАННЯ ПЛАТ

4.1 Механізми забезпечення безпечної роботи

Забезпечення надійності апаратної частини телекомунікаційних систем на базі STM32 є критично важливим етапом проєктування, що охоплює електричну стійкість, логічну відмовостійкість, захист від зовнішніх впливів та здатність до автономного відновлення. У системах комутації кінцевих пристроїв, де мікроконтролер виконує функції маршрутизації, обробки та передачі даних, навіть короткочасна відмова може призвести до втрати інформації, порушення синхронізації або зупинки мережевого сегмента. Тому при розробці апаратної платформи необхідно враховувати комплекс заходів, що забезпечують стабільну роботу в умовах промислових завад, нестабільного живлення та програмних збоїв.

Таблиця 4.1 узагальнює ключові апаратні засоби захисту та забезпечення надійності мікроконтролерів STM32 у телекомунікаційних пристроях.

Таблиця 4.1 – Ключові апаратні засоби захисту та надійності STM32

Категорія	Механізм	Призначення
Електрозахист	TVS-діоди	Захист від перенапруг та ESD
	LC-фільтри + конденсатори	Стабілізація живлення
	Захист від зворотної полярності	Запобігання пошкодженню
	Розділення землі	Зменшення паразитних зв’язків
Відмовостійкість	IWDG / WWDG	Перезапуск при зависанні
	BOR	Скидання при падінні напруги
	Обробники винятків	Локалізація критичних помилок
Цілісність даних	CRC-модуль	Перевірка даних
	RDP	Захист пам’яті
Діагностика	POST	Перевірка при старті
	Логування подій	Аналіз збоїв
Теплове управління	Теплові vias / радіатори	Відведення тепла
Проєктування плат	Багатошарова структура	Екранування, стабільність
	Мінімізація трас	Уникнення втрат сигналу

Вона структурована за категоріями, що охоплюють основні аспекти апаратного проєктування: електрозахист, відмовостійкість, контроль цілісності даних, діагностику, теплове управління та загальні принципи трасування плат.

Кожен рядок таблиці містить три елементи:

– категорію – тип захисту або функціонального напрямку (наприклад, «Електрозахист», «Відмовостійкість»);

– механізм – конкретний технічний засіб, що реалізується у STM32 або на рівні плати (наприклад, TVS-діоди, CRC-модуль, POST);

– призначення – коротке пояснення функції механізму (наприклад, «Захист від перенапруг», «Перезапуск при зависанні», «Відведення тепла»).

Така структура дозволяє швидко зорієнтуватися в доступних засобах захисту, оцінити їхню роль у системі та визначити, які з них є критично важливими для конкретного типу телекомунікаційного пристрою.

Однією з найпоширеніших причин виходу з ладу мікроконтролерів у польових умовах є електричні перевантаження, імпульсні завади, електростатичні розряди (ESD) та нестабільне живлення. Як зазначено в технічній документації, « електричні перевантаження є однією з основних причин виходу з ладу мікроконтролерів у польових умовах » [1]. Для мінімізації ризиків рекомендується використовувати TVS-діоди на інтерфейсах USB, UART, CAN, які здатні поглинати імпульси перенапруги та захистити входи мікроконтролера. Фільтрація живлення за допомогою LC-фільтрів з низьким ESR дозволяє стабілізувати напругу та зменшити рівень шуму, особливо у чутливих аналогових модулях. Розділення аналогової та цифрової землі є критично важливим для зменшення паразитних зв’язків, що можуть впливати на точність АЦП, генераторів та інших модулів. Захист від зворотної полярності реалізується через діоди Шоткі або P-канальні MOSFET, що дозволяє уникнути пошкодження при неправильному підключенні живлення.

Мікроконтролери STM32 чутливі до якості живлення, особливо в режимах зниженого енергоспоживання, таких як Sleep, Stop або Standby. Для забезпечення стабільної роботи необхідно використовувати стабілізатори з низьким рівнем шуму, які мають захист від короткого замикання та термічних перевантажень. Резервне живлення (VBAT) дозволяє зберігати дані реального часу (RTC), а також критичну інформацію в умовах втрати основного живлення. У STM32 реалізовано механізм Brown-Out Reset (BOR), який активується при падінні напруги нижче

критичного рівня. Як зазначено в документації STM32, « BOR дозволяє уникнути некоректного виконання коду при нестабільному живленні » [5]. Додатково доцільно реалізувати моніторинг напруги живлення через АЦП або зовнішні компаратори, що дозволяє виявити деградацію джерела живлення до виникнення критичної ситуації.

Для виявлення та локалізації збоїв у роботі системи STM32 підтримує низку апаратних і програмних засобів самодіагностики. Watchdog-таймери (IWDG, WWDG) забезпечують автоматичне перезавантаження у разі зависання програми або порушення логіки виконання. Як зазначено в офіційному посібнику, « використання незалежного watchdog-таймера дозволяє гарантувати відновлення роботи системи навіть у разі критичних збоїв » [10]. Обробники винятків (HardFault, BusFault, UsageFault) дозволяють локалізувати джерело помилки, зчитати регістри стеку, визначити адресу інструкції, що спричинила збій, та виконати аварійне завершення або перезапуск. « Обробники винятків у STM32 дозволяють локалізувати джерело помилки та виконати аварійне завершення або перезапуск системи » [11].

CRC-модуль у STM32 дозволяє обчислювати контрольні суми для блоків пам’яті або потоку даних, що забезпечує виявлення пошкоджень прошивки, конфігураційних файлів або пакетів, що надходять з мережі. « CRC-модуль у STM32 дозволяє обчислювати контрольні суми для блоків пам’яті або потоку даних, що забезпечує виявлення пошкоджень » [5]. У разі виявлення невідповідності система може автоматично запустити процедуру повторного завантаження, повідомити про помилку або запитати повторну передачу. Логування подій – ще один важливий механізм, який дозволяє зберігати інформацію про помилки, перезапуски, нестабільність живлення у резервній пам’яті або зовнішньому EEPROM. Це дає змогу аналізувати причини збоїв, вдосконалювати прошивку та підвищувати стабільність системи.

У складних системах доцільно реалізовувати тестові процедури при старті (POST), які перевіряють працездатність основних вузлів: Flash, SRAM, периферії, тактових генераторів. Це дозволяє виявити деградацію компонентів або порушення конфігурації ще до початку основного циклу роботи. У STM32 також

реалізовано механізми захисту пам’яті, зокрема Readout Protection (RDP), що дозволяє обмежити доступ до внутрішньої Flash-пам’яті. Як зазначено в документації, « захист пам’яті STM32 дозволяє запобігти несанкціонованому читанню або модифікації прошивки » [9]. У поєднанні з апаратним захистом від запису, контрольними сумами та шифруванням даних, це дозволяє створювати телекомунікаційні пристрої з високим рівнем інформаційної безпеки.

Проєктування друкованої плати для STM32 потребує дотримання низки принципів, які забезпечують фізичну надійність, електричну стабільність та захист від зовнішніх впливів. Багатошарова структура плати з окремими шарами для живлення та землі дозволяє зменшити паразитні зв’язки, покращити екранування та знизити рівень електромагнітних завад. Розміщення конденсаторів розв’язки біля кожного піну живлення забезпечує локальну стабілізацію напруги та зменшення імпульсних коливань. Мінімізація довжини критичних трас, особливо для високошвидкісних інтерфейсів (USB, SPI, CAN), дозволяє уникнути втрат сигналу, відбиттів та порушення таймінгів. Використання тестових точок на платі – важливий елемент для діагностики, оновлення прошивки та вимірювання сигналів у процесі експлуатації.

Теплове моделювання – ще один аспект, який слід враховувати при проєктуванні. Розрахунок теплового навантаження на мікроконтролер, стабілізатори, драйвери та силові компоненти дозволяє уникнути перегріву, деградації матеріалів та зниження ресурсу пристрою. У системах з високою щільністю компонентів доцільно використовувати теплові отвори, мідні площини, тер мідні площини, термопровідні пасти та радіатори, які сприяють ефективному відведенню тепла від критичних елементів. У випадках, коли мікроконтролер працює з високим навантаженням або у середовищі з обмеженим повітряним охолодженням, необхідно передбачити додаткові засоби пасивного або активного охолодження. Наприклад, застосування алюмінієвих радіаторів, термопрокладок або вентиляторів у корпусах із щільною компоновкою дозволяє знизити температуру кристала та уникнути термічного тротлінгу.

« Перевищення допустимого теплового навантаження на мікроконтролер може призвести до нестабільної роботи, зниження точності аналогових модулів та

передчасного виходу з ладу » [1]. Тому на етапі проєктування доцільно використовувати інструменти теплового аналізу – наприклад, моделювання розподілу температури у середовищах Altium Designer, KiCad або SolidWorks PCB. Це дозволяє виявити зони перегріву, оптимізувати розміщення компонентів та забезпечити рівномірне теплове навантаження.

Особливу увагу слід приділяти зонам навколо стабілізаторів живлення, драйверів двигунів, силових ключів та мікроконтролера, оскільки саме ці елементи генерують найбільше тепла. У багатошарових платах рекомендується використовувати внутрішні теплові шари, які з'єднані з корпусом через термопровідні отвори (vias). Також важливо забезпечити належне теплове з'єднання між корпусом мікроконтролера (особливо у варіантах QFN або BGA) та тепловими площинами плати.

У телекомунікаційних системах, що працюють у промислових умовах або в закритих корпусах, теплове моделювання є не просто рекомендованим, а обов'язковим етапом проєктування. Це дозволяє забезпечити стабільну роботу пристрою протягом тривалого часу, уникнути деградації матеріалів, зменшити ризик термічних збоїв та продовжити життєвий цикл системи. Встановлено, що « правильне розведення плати є критичним для уникнення паразитних зв'язків, перешкод та теплових перевантажень » [1].

4.2 Модуль Flash-пам'яті та контроль цілісності даних

У телекомунікаційних пристроях, побудованих на базі STM32, модуль Flash-пам'яті виконує функцію довготривалого зберігання критичних даних – прошивки, конфігураційних параметрів, таблиць маршрутизації, журналів подій, а також резервних копій. Надійність цієї пам'яті є визначальною для стабільної роботи системи, особливо в умовах промислової експлуатації, де доступ до пристрою обмежений, а оновлення можливе лише дистанційно. Тому контроль цілісності даних, захист від пошкоджень і правильне використання Flash-пам'яті – це ключові аспекти проєктування безпечної апаратної частини телекомунікаційного вузла.

Flash-пам’ять у STM32 є енергонезалежною пам’яттю типу NOR, організованою у вигляді сторінок або секторів. Згідно з технічним посібником, « Flash-пам’ять STM32 організована у вигляді сторінок фіксованого розміру, які можуть бути стерті лише повністю. Запис можливий лише в попередньо очищені області » [5]. Це означає, що кожна операція запису потребує попереднього стирання, що впливає на ресурс пам’яті та вимагає обережного планування структури зберігання. У більшості мікроконтролерів STM32 Flash-пам’ять розділена на основну область (Main Flash), призначену для зберігання коду, та системну пам’ять (System Memory), яка містить заводський завантажувач. Доступ до пам’яті здійснюється через шину AHB, що забезпечує швидке читання інструкцій та даних, а також дозволяє ефективно реалізовувати алгоритми оновлення прошивки.

Операції запису та стирання Flash-пам’яті виконуються через контролер Flash, який керується спеціальними регістрами. Як зазначається « для запису у Flash необхідно розблокувати контролер, активувати режим програмування, записати дані та дочекатися завершення операції, контролюючи прапор завершення » [4]. Стирання виконується на рівні сторінок або банків пам’яті, і в разі переривання процесу – наприклад, через падіння живлення – можливе часткове пошкодження даних. Тому важливо реалізовувати контроль завершення операцій, дублювання критичних ділянок та перевірку результату запису. У телекомунікаційних системах, де оновлення прошивки відбувається через мережу, особливо важливо забезпечити атомарність запису та захист від неповного оновлення.

STM32 підтримує кілька механізмів захисту Flash-пам’яті, які дозволяють обмежити доступ до вмісту, запобігти несанкціонованому запису та забезпечити цілісність критичних ділянок. Readout Protection (RDP) – один із базових механізмів, що обмежує доступ до вмісту пам’яті через зовнішні інтерфейси. « RDP дозволяє запобігти несанкціонованому копіюванню прошивки або витоку конфіденційних даних » [3]. Крім того, механізм Write Protection блокує запис у вибрані сторінки, що особливо важливо для захисту завантажувача, таблиць конфігурації або ділянок, які не повинні змінюватися в процесі експлуатації.

Конфігураційні байти (Option Bytes) дозволяють визначити режими захисту, джерело завантаження, поведінку при скиданні та інші параметри, що впливають на безпеку системи. У поєднанні з апаратним захистом від запису, ці механізми дозволяють створювати пристрої з підвищеним рівнем інформаційної безпеки, що відповідає вимогам телекомунікаційних систем.

Цілісність даних у Flash-пам’яті може порушуватись через електричні збої, деградацію комірок пам’яті або помилки при записі. Для виявлення таких ситуацій у STM32 реалізовано апаратний CRC-модуль, який дозволяє обчислювати контрольні суми. « CRC-модуль у STM32 дозволяє перевіряти цілісність прошивки, конфігураційних таблиць або даних, що зберігаються у Flash » [5]. Рекомендовано зберігати контрольні суми для кожного критичного блоку даних та перевіряти їх при кожному запуску системи. У разі виявлення невідповідності система може перейти в безпечний режим (safe mode), виконати перезапис з резервної копії або повідомити про помилку через інтерфейс зв’язку. У телекомунікаційних системах доцільно реалізовувати двоступеневе оновлення прошивки: нова версія записується в окрему область пам’яті, перевіряється за CRC, і лише після цього активується. Це дозволяє уникнути запуску пошкодженого коду та забезпечити стабільність роботи пристрою після оновлення.

Flash-пам’ять має обмежену кількість циклів запису / стирання (зазвичай 10 000…100 000), що накладає обмеження на частоту оновлення даних. Як зазначено в джерелі, « часте перезаписування однієї й тієї ж сторінки Flash може призвести до її деградації, тому критично важливо реалізовувати механізми контролю ресурсу » [2]. Для зменшення навантаження на пам’ять рекомендується мінімізувати частоту записів, використовувати wear leveling – рівномірний розподіл навантаження по сторінках, а також зберігати змінні дані у зовнішній EEPROM або FRAM, якщо це дозволяє архітектура пристрою. У системах, де зберігаються журнали подій, статистика або конфігураційні параметри, доцільно реалізовувати циклічні буфери, які дозволяють уникнути надмірного навантаження на окремі ділянки пам’яті.

У промислових телекомунікаційних системах, що працюють без постійного нагляду, важливо забезпечити не лише збереження даних, а й їхню достовірність. Для цього рекомендується реалізовувати періодичну перевірку контрольних сум, дублювання критичних блоків, а також механізми автоматичного відновлення. Наприклад, при виявленні пошкодження таблиці маршрутизації система може автоматично завантажити резервну копію з окремої області пам’яті або з зовнішнього джерела. У поєднанні з механізмами захисту доступу, це дозволяє створити систему, здатну до самовідновлення та захисту від логічних атак.

Таким чином, модуль Flash-пам’яті у STM32 є не лише засобом зберігання коду, а й критичним елементом забезпечення надійності, безпеки та стабільності телекомунікаційного пристрою. Його правильне використання, захист, контроль цілісності та управління ресурсом – це фундаментальні аспекти проєктування апаратної частини, які мають бути враховані на всіх етапах розробки. У поєднанні з програмними засобами перевірки, резервування та логування, Flash-пам’ять стає основою для створення відмовостійких систем, здатних до автономної роботи в умовах реального часу.

4.3 Інструментальні засоби для проєктування та налагодження

Проєктування та налагодження апаратної частини телекомунікаційних пристроїв на базі STM32 потребує використання спеціалізованих інструментальних засобів, які забезпечують точне конфігурування периферії, трасування друкованих плат, генерацію прошивки, а також діагностику та тестування на етапі запуску. Висока інтеграція мікроконтролерів STM32, наявність великої кількості периферійних модулів та підтримка промислових інтерфейсів зумовлюють потребу в комплексному підході до розробки, який включає апаратні, програмні та симуляційні засоби.

Одним із базових інструментів для конфігурування мікроконтролера є STM32CubeMX – графічне середовище, що дозволяє налаштувати периферійні модулі, задати параметри тактування, енергоспоживання, пін-контакти та згенерувати початковий код. Як зазначено в офіційному посібнику, «

STM32CubeMX дозволяє налаштувати мікроконтролер без глибоких знань апаратної архітектури, що значно пришвидшує розробку » [8]. Інструмент підтримує автоматичну перевірку конфліктів пінів, візуалізацію схеми підключення, вибір стеків протоколів (USB, TCP/IP, FreeRTOS) та інтеграцію з середовищем розробки. Це дозволяє швидко створити базову прошивку, яка вже містить ініціалізацію периферії, обробники переривань та шаблони функцій, що значно скорочує час розробки та знижує ризик помилок.

Для повноцінної розробки, компіляції та налагодження прошивки використовується STM32CubeIDE – інтегроване середовище, що поєднує редактор коду, компілятор, відладчик та засоби моніторингу. Воно базується на платформі Eclipse і включає компілятор GCC, засоби трасування, перегляд регістрів, змінних, стеку та пам’яті. « STM32CubeIDE дозволяє повноцінно розробляти, компілювати, налагоджувати та тестувати прошивку для STM32 у єдиному середовищі » [8]. Середовище підтримує покрокове виконання коду, встановлення точок зупину, перегляд значень регістрів у реальному часі, інтеграцію з STM32CubeMX, підтримку FreeRTOS Debug та профілювання продуктивності. Це дозволяє ефективно виявляти логічні помилки, некоректну ініціалізацію периферії, порушення таймінгів та інші критичні проблеми, що особливо важливо для телекомунікаційних систем, де стабільність і точність мають вирішальне значення.

Для прямого доступу до пам’яті мікроконтролера, оновлення прошивки та тестування використовуються ST-LINK Utility або STM32CubeProgrammer. Ці інструменти дозволяють зчитувати та записувати Flash, стирати пам’ять, змінювати Option Bytes, активувати захист RDP, перевіряти CRC прошивки та оновлювати прошивку через UART, USB, SWD або JTAG. « STM32CubeProgrammer дозволяє здійснювати повний контроль над пам’яттю мікроконтролера, включно з конфігурацією захисту та завантаженням прошивки » [5]. Ці засоби особливо корисні на етапі виробництва, коли необхідно масово прошивати пристрої, перевіряти контрольні суми або активувати захист від копіювання. Крім того, вони дозволяють проводити діагностику пам’яті,

перевірку конфігураційних байтів та оновлення прошивки без використання середовища розробки.

Налагодження апаратної частини потребує використання фізичних інструментів, які дозволяють аналізувати сигнали, перевіряти електричні параметри та виявляти апаратні збої. До таких засобів належать ST-LINK/V2 або ST-LINK/V3 – апаратні програматори/відладчики для SWD / JTAG, Discovery/Nucleo плати – референсні платформи з вбудованим ST-LINK, логічні аналізатори – для аналізу сигналів UART, SPI, I²C, осцилографи – для перевірки таймінгів, форм сигналів, стабільності живлення, мультиметри та термокамери – для контролю електричних параметрів та теплових режимів. Як зазначено в джерелі, « апаратні засоби налагодження дозволяють виявити проблеми, які неможливо діагностувати програмно – зокрема, паразитні сигнали, нестабільність живлення, перегрів » [1]. У телекомунікаційних системах, де пристрої працюють у складних умовах, ці засоби є незамінними для забезпечення стабільної роботи.

На етапі проєктування друкованої плати використовуються програмні засоби трасування, перевірки правил (DRC), моделювання сигналів та оцінки електромагнітної сумісності. Найпоширенішими є KiCad, Altium Designer, EasyEDA – для трасування, Gerber Viewer – для перевірки виробничих файлів, SPICE-симулятори – для моделювання аналогових ланцюгів, EMC-аналізатори – для оцінки електромагнітної сумісності. У телекомунікаційних системах особливо важливо враховувати розміщення фільтрів на входах, розділення аналогової та цифрової землі, мінімізацію довжини трас високошвидкісних сигналів, екранування чутливих вузлів та теплове моделювання. « Правильне трасування плати є критичним для забезпечення стабільної роботи мікроконтролера в умовах електромагнітних завад » [2]. Це дозволяє уникнути паразитних зв’язків, втрат сигналу та порушення таймінгів, що особливо важливо для інтерфейсів UART, SPI, USB.

Для моніторингу роботи пристрою в реальному часі використовуються спеціалізовані засоби, які дозволяють візуалізувати змінні, графіки, логи, а також аналізувати поведінку задач у FreeRTOS. До таких засобів належать STM32CubeMonitor – для візуалізації змінних, FreeRTOS Tracealyzer – для аналізу

задач, таймерів, черг, USB CDC / Virtual COM – для передачі логів на ПК, Ethernet SNMP / HTTP – для віддаленого моніторингу. Ці засоби дозволяють виявляти нестабільність, перевантаження, втрату даних, порушення таймінгів. « Моніторинг у реальному часі дозволяє виявити проблеми, які не проявляються в лабораторних умовах, але критичні в польовій експлуатації » [1]. У телекомунікаційних системах, де пристрої працюють без постійного нагляду, можливість дистанційного моніторингу є ключовою для забезпечення надійності.

Таким чином, інструментальні засоби для проєктування та налагодження систем на базі STM32 охоплюють повний цикл розробки – від конфігурування периферії до моніторингу в реальному часі. Їхнє правильне використання дозволяє створити стабільні, масштабовані та безпечні телекомунікаційні пристрої, здатні до автономної роботи в складних умовах експлуатації.

4.4 Рекомендації щодо проєктування плат для телекомунікаційних

Проєктування друкованих плат для телекомунікаційних пристроїв на базі STM32 є складним інженерним завданням, яке виходить далеко за межі базового електричного з’єднання компонентів. У таких системах мікроконтролер виконує роль вузла комутації, обробки сигналів, маршрутизації або моніторингу, і тому будь-які похибки в апаратному дизайні можуть призвести до критичних збоїв у роботі мережі. Особливу увагу слід приділяти електромагнітній сумісності, стабільності живлення, захисту від імпульсних завад, а також забезпеченню надійної роботи інтерфейсів зв’язку, які є основою телекомунікаційної взаємодії.

У таблиці 4.3 узагальнено основні інструментальні засоби, що використовуються для проєктування, налагодження та моніторингу телекомунікаційних пристроїв на базі STM32. Вони охоплюють як програмні, так і апаратні компоненти, необхідні для повного циклу розробки – від конфігурації до польового тестування.

Першим кроком у проєктуванні плати є розуміння умов експлуатації пристрою. Телекомунікаційні системи часто функціонують у промислових або зовнішніх середовищах, де присутні сильні електромагнітні поля, температурні

коливання, нестабільне живлення та ризик електростатичних розрядів. Як зазначено в джерелі, « у телекомунікаційних системах стабільність роботи апаратної частини залежить не лише від мікроконтролера, а й від якості плати, її розведення та захисту » [1]. Це означає, що плата повинна бути не просто функціональною, а стійкою до зовнішніх впливів, здатною забезпечити довготривалу роботу без втручання оператора.

Таблиця 4.3 – Інструментальні засоби для проєктування та налагодження STM32 у телекомунікаційних системах.

Тип засобу	Назва / Приклад	Основне призначення
Конфігуратор периферії	STM32CubeMX	Налаштування пінів, тактування, генерація стартового коду
Середовище розробки	STM32CubeIDE	Редагування, компіляція, налагодження, FreeRTOS Debug
Програматор / утиліта	STM32CubeProgrammer, ST-LINK	Запис прошивки, стирання пам’яті, захист, CRC, Option Bytes
Апаратне налагодження	ST-LINK/V2, Nucleo, осцилограф	Аналіз сигналів, таймінгів, живлення, температури
Трасування плат	KiCad, Altium Designer, EasyEDA	Розведення, DRC, моделювання, перевірка Gerber-файлів
Симуляція схем	SPICE, EMC-аналізатори	Аналіз аналогових ланцюгів, електромагнітна сумісність
Моніторинг у реальному часі	STM32CubeMonitor, Tracealyzer	Візуалізація змінних, логів, аналіз задач FreeRTOS
Комунікаційні інтерфейси	USB CDC, Ethernet SNMP/HTTP	Передача логів, віддалений моніторинг, діагностика

Організація живлення є одним із ключових аспектів. Мікроконтролери STM32 чутливі до коливань напруги, особливо в режимах зниженого енергоспоживання. Тому трасування живлення повинно забезпечувати мінімальний опір, ефективну фільтрацію та захист від імпульсних перенапруг. Суцільна площина землі, яка проходить під мікроконтролером, є обов’язковою умовою для зменшення шуму та покращення тепловідведення. Розділення аналогової та цифрової землі дозволяє уникнути паразитних зв’язків, які можуть впливати на точність АЦП або стабільність генераторів. Конденсатори розв’язки слід розміщувати безпосередньо біля пінів живлення, причому використовувати комбінацію ємностей – наприклад, 100 нФ для високочастотного шуму та 1 мкФ для низькочастотного згладжування.

Розведення інтерфейсів зв’язку потребує особливої уваги. У телекомунікаційних пристроях використовуються UART, SPI, I²C, USB, CAN, Ethernet – кожен з яких має специфічні вимоги до трасування. Наприклад, для USB необхідно забезпечити контрольований імпеданс диференціальної пари D+/D– на рівні 90 Ом, симетрію трас, мінімізацію відхилень та уникнення гострих кутів. Як зазначено в документації STM32, « USB-периферія потребує точного дотримання електричних стандартів, зокрема розведення диференціальної пари » [5]. Для CAN-інтерфейсу важливо забезпечити термінування лінії резистором 120 Ом, а також використовувати трансивери з вбудованим захистом від ESD. Ethernet вимагає застосування трансформаторів, симетричних трас та розділення аналогових і цифрових контурів, щоб уникнути перешкод. Усі ці вимоги мають бути враховані на етапі трасування, оскільки порушення електричних стандартів може призвести до втрати даних, нестабільності зв’язку або повної відмови інтерфейсу.

Захист від електричних завад є ще одним критичним елементом. У телекомунікаційних системах пристрої часто підключаються до довгих кабелів, які можуть діяти як антени для імпульсних перенапруг. Тому на входах інтерфейсів слід встановлювати TVS-діоди, які здатні поглинати імпульсні струми без пошкодження мікроконтролера. Фільтрація живлення за допомогою LC-фільтрів дозволяє згладити пульсації, особливо при живленні від нестабільних джерел. Захист від зворотної полярності реалізується через діоди Шотткі або P-канальні MOSFET, які не допускають подачу напруги у зворотному напрямку. Розділення контурів живлення – аналогового, цифрового та силового – дозволяє уникнути взаємного впливу між модулями, що особливо важливо для систем з високою щільністю інтеграції.

Теплове моделювання є важливим етапом, особливо якщо пристрій працює в умовах підвищеного навантаження або обмеженої вентиляції. Мікроконтролери STM32, хоча й енергоефективні, можуть нагріватися при активному використанні периферії, особливо USB, Ethernet або CAN. Тому компонування плати повинно враховувати теплові потоки: силові елементи слід розміщувати ближче до краю, мікроконтролер – у центрі, а навколо нього – теплові полігони, які забезпечують

відведення тепла. Теплове моделювання дозволяє уникнути перегріву мікроконтролера та забезпечити стабільну роботу в умовах високого навантаження. У корпусі пристрою слід передбачити вентиляційні отвори або теплові канали, які сприяють природній конвекції. У складних системах доцільно використовувати термопрокладки, теплові отвори (vias) та мідні площини, які покращують тепловідведення.

Тестування та обслуговування – ще один аспект, який часто недооцінюється на етапі проєктування. Для зручності діагностики слід передбачити тестові точки на ключових сигнальних лініях, роз'єм для оновлення прошивки (SWD, UART, USB), а також індикатори стану – світлодіоди, які сигналізують про живлення, активність, помилки. Можливість вимірювання струму через шунт або розрив живлення дозволяє контролювати енергоспоживання пристрою в реальному часі. У телекомунікаційних системах, де пристрої можуть бути встановлені у важкодоступних місцях, ці функції є критично важливими для обслуговування. Крім того, слід передбачити можливість дистанційного оновлення прошивки, що дозволяє зменшити витрати на сервісне обслуговування.

Нарешті, слід враховувати типові помилки, які можуть виникнути при проєктуванні плат. До них належать паразитні зв'язки через неправильне розведення землі, відбиття сигналу через довгі траси без термінування, перешкоди між інтерфейсами через близьке розміщення, перегрів через відсутність теплових полігонів, нестабільність живлення через недостатню фільтрацію. Як зазначено в інженерному посібнику, « правильне розведення плати є критичним для уникнення паразитних зв'язків, перешкод та теплових перевантажень » [1].

Таким чином, проєктування плат для телекомунікаційних пристроїв на базі STM32 – це не лише процес електричного з'єднання компонентів, а комплексна інженерна дисципліна, що поєднує знання з електроніки, теплотехніки, захисту від завад, тестування, обслуговування та забезпечення інформаційної безпеки. У телекомунікаційних системах, де пристрої працюють у складних умовах якість проєктування плати визначає не лише функціональність, а й довговічність, ремонтопридатність та здатність до автономної роботи.

ВИСНОВКИ

У магістерській роботі виконано комплексне дослідження системи комутації кінцевих пристроїв на базі мікроконтролерів STM32.

1. Проведено аналіз архітектури ARM Cortex-M та технічних характеристик мікроконтролерів STM32, що дозволило визначити їхню придатність для реалізації апаратного керування комутаційними процесами. Встановлено, що STM32 забезпечують широкий набір периферійних інтерфейсів (UART, SPI, I²C, USB, CAN, Ethernet), підтримують гнучке енергоспоживання та мають високий рівень інтеграції, що є критично важливим для телекомунікаційних застосувань.

2. Розроблено структурну та функціональну схему комутаційної системи, обґрунтовано вибір апаратного забезпечення та програмних засобів, зокрема STM32CubeMX, STM32CubeIDE, HAL / LL бібліотек.

3. Виконано аналіз функціонування системи, що дозволило оцінити її продуктивність, стабільність та здатність до роботи в реальному часі. Визначено оптимальні режими енергоспоживання, що забезпечують баланс між швидкодією та автономністю.

4. Особливу увагу приділено питанням надійності апаратної частини, контролю цілісності даних, захисту пам’яті та стабільності живлення. Розглянуто механізми самодіагностики, обробки помилок, CRC-контролю, а також засоби захисту Flash-пам’яті від несанкціонованого доступу.

5. Сформовано рекомендації щодо проєктування друкованих плат, з урахуванням електромагнітної сумісності, теплового моделювання та захисту від імпульсних завад. Узагальнено інструментальні засоби для проєктування, налагодження та моніторингу системи, що охоплюють повний цикл розробки. Визначено перспективи інтеграції системи в телекомунікаційні системи різного призначення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Телекомунікаційне обладнання. URL: https://ela.kpi.ua/server/api/core /bitstreams/7474eba0-f4cc-4882-a787-562592b59329/content (дата звернення: 15.09.2025).

2. Мікроконтролери. URL: https://ppt-online.org/226762 (дата звернення: 15.09.2025).

3. Arm. Silicon IP: CPU. URL: https://www.arm.com /products/silicon-ip-cpu (дата звернення: 15.09.2025).

4. Пояснення STM32: функції та застосування. URL: https://www.wonderfulpcb.com/uk/blog/stm32-explained-features-applications/ (дата звернення: 10.10.2025).

5. STMicroelectronics. RM0008 Reference Manual for STM32F103x. URL: https://www.st.com/resource/en/reference_manual/rm0008-stm32f103xx-reference-manual-stmicroelectronics.pdf (дата звернення: 15.09.2025).

6. STMicroelectronics. AN2582 Application Note: USART communication in STM32 microcontrollers. URL: https://www.st.com/resource/en/application_note /an2582-usart-communication-in-stm32-microcontrollers-stmicroelectronics.pdf (дата звернення: 15.09.2025).

7. STMicroelectronics. AN1796 Application Note: SPI communication with STM32 microcontrollers. URL: https://www.st.com/resource/en /application_note/an1796-spi-communication-with-stm32-microcontrollers-stmicroelectronics.pdf (дата звернення: 15.09.2025).

8. STMicroelectronics. UM1718 User Manual: STM32CubeMX for STM32 configuration and code generation. URL: https://www.st.com /resource/en/user_manual/um1718-stm32cubemx-for-stm32-configuration-and-code-generation-stmicroelectronics.pdf (дата звернення: 15.09.2025).

9. Урок 9 по STM32: режими енергоспоживання. URL: https://itmaster.biz.ua/electronics/stm32/stm32-power-managment.html (дата звернення: 15.09.2025).

10. Power Management Mastering STM32). URL: https://blog.radiotech.kz/stm32/power-management-chast-1-perevod-iz-knigi-mastering-stm32/ (дата звернення: 15.09.2025).

11. STMicroelectronics. AN4750: Handling of soft errors in STM32 applications. URL: https://www.st.com/resource/en/application_note/an4750-handling-of-soft-errors-in-stm32-applications-stmicroelectronics.pdf (дата звернення: 15.09.2025).

12. STM32F4 Error Handler and allowed operations. Stack Overflow. URL: https://stackoverflow.com/questions/48984041/stm32f4-error-handler-and-allowed-operations (дата звернення: 15.09.2025).

13. STMicroelectronics. RM0008 Reference Manual for STM32F103x. URL: https://www.st.com/resource/en/reference_manual/rm0008-stm32f103xx-reference-manual-stmicroelectronics.pdf (дата звернення: 15.09.2025).

14. Power Management частина 1 (переклад з книги Mastering STM32). URL: https://blog.radiotech.kz/stm32/power-management-chast-1-perevod-iz-knigi-mastering-stm32/ (дата звернення: 15.09.2025).

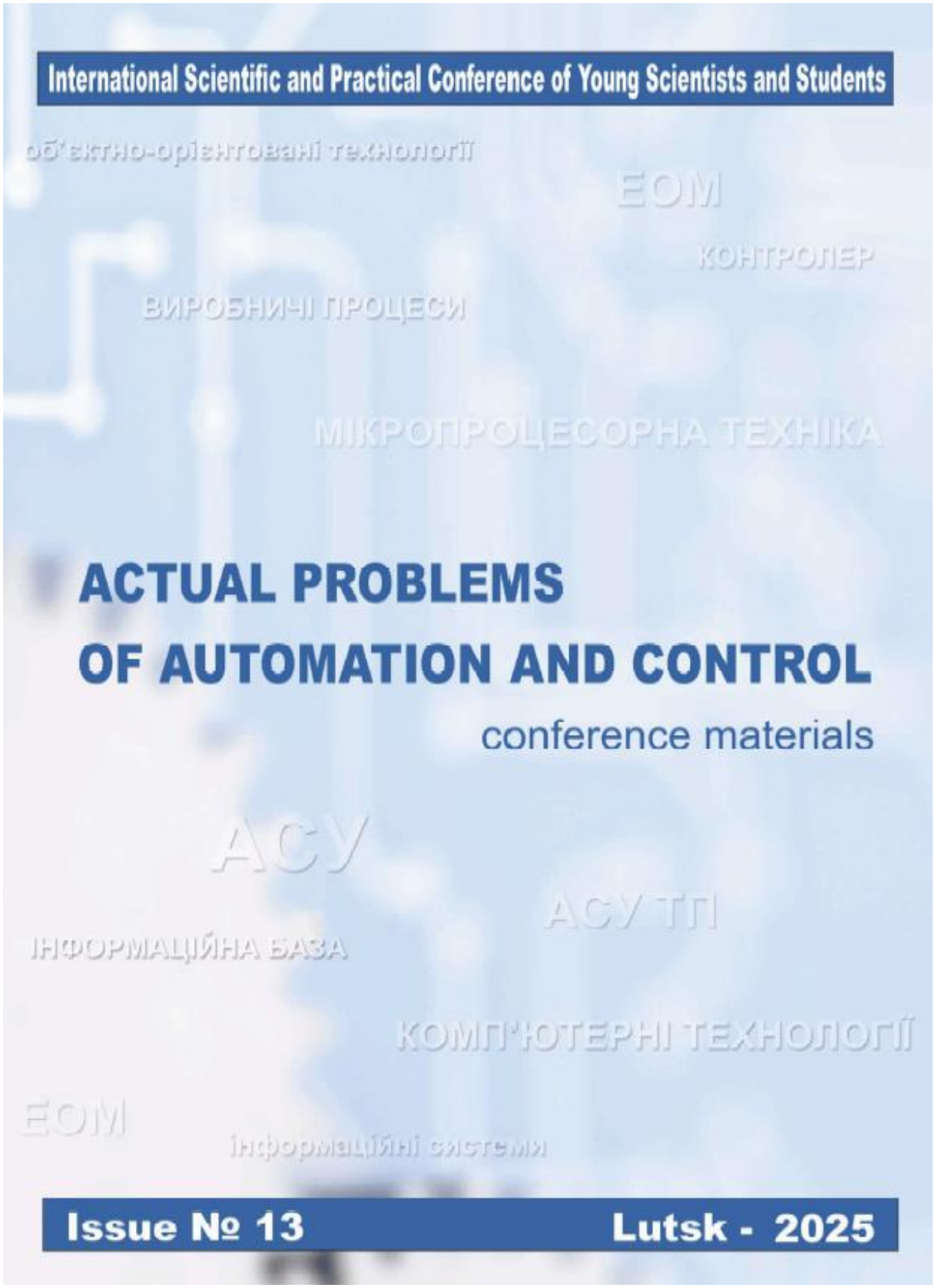
15. STM32: DMA (в архітектурі CortexM0). URL: https://itmaster.biz.ua/electronics/stm32/stm32-dma.html (дата звернення: 15.09.2025).

16. Якимчук Н. М., Карпінський Н. К. Організація UART-комунікації в STM32 для системи комутації кінцевих пристроїв. Actual Problems of Automation and Control. Lutsk. Lutsk. № 13. 2025. С.152-156.

ДОДАТКИ

ДОДАТОК А

Публікації здобувача



СЕКЦІЯ «МОДЕЛЮВАННЯ ТА ОПТИМІЗАЦІЯ СИСТЕМ УПРАВЛІННЯ»

СЕКЦІЯ «МЕХАТРОННІКА ТА РОБОТИЗОВАНІ СИСТЕМИ»

СЕКЦІЯ «АВТОМАТИКА В ЕЛЕКТРОНІЦІ ТА ТЕЛЕКОМУНІКАЦІЯХ»

Додаткову увагу в межах проекту приділено питанню безпеки роботи голосової системи, оскільки керування побутовими пристроями потребує запобігання випадковим або небажаним активаціям. Зокрема, передбачено використання ключового активаційного слова, що мінімізує ризик помилкового розпізнавання у шумному середовищі. Окремі команди, пов'язані з критичними функціями, такими як увімкнення живлення високопотужних приладів або доступ до мультимедійних чи мережевих ресурсів, можуть бути обмежені для виконання без додаткової локальної перевірки. Такий підхід забезпечує більш надійну та контрольовану взаємодію з системою в умовах домашньої експлуатації.

Таким чином, реалізована система демонструє можливість ефективного використання одноплатного комп'ютера Raspberry Pi для організації голосового керування побутовими пристроями. Поєднання автономності, точності розпізнавання та широких можливостей апаратної взаємодії робить таке рішення конкурентоспроможним і перспективним у контексті розвитку Smart Home-технологій [5]. Додатково, система є відкритою для подальшого вдосконалення та модернізації, що дозволяє інтегрувати нові функції, алгоритми штучного інтелекту та сучасні стандарти безпеки, роблячи її універсальним інструментом для досліджень і впровадження інтелектуальних рішень у домашньому середовищі.

Висновки. Запропонована система розпізнавання голосових команд на базі Raspberry Pi довела свою ефективність як сучасне та гнучке рішення для побудови інтелектуальних інтерфейсів керування. Реалізована модель забезпечує повний цикл роботи з аудіосигналом: від його захоплення та попередньої обробки до подальшого розпізнавання команд і виконання відповідних дій підключеними пристроями. Практична реалізація показала, що обчислювальні можливості Raspberry Pi є достатніми для стабільної роботи алгоритмів голосового управління без необхідності використання дорогих апаратних платформ. Робота системи може здійснюватися як у локальному режимі, так і з розширеними можливостями через додаткові модулі зв'язку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Raspberry Pi Documentation. URL: https://www.raspberrypi.com/documentation/ (дата звернення 13.11.2025р.)

2. SpeechRecognition Library for Python. URL: https://pypi.org/project/SpeechRecognition/ (дата звернення 13.11.2025р.)

3. Monk, S. Raspberry Pi Cookbook (4th Edition). – Sebastopol, CA: O'Reilly Media, URL: https://archive.org/details/raspberrypicookb0000monk_x0t8 (дата звернення 15.11.2025р.)

4. Jurafsky D., Martin J. H. Speech and Language Processing (3rd Edition, Draft). – Stanford University, URL: https://web.stanford.edu/~jurafsky/slp3 (дата звернення 15.11.2025р.)

5. Богдан Ю., Зенів І.О. Технології Інтернету речей. – КПІ ім. І. Сікорського URL: https://ela.kpi.ua/server/api/core/bitstreams/dcd9e1aa-8bcc-4e76-b1e0-ed133bf616b2/content (дата звернення 16.11.2025р.)

УДК 004.383.2:621.391

Якимчук Н. М., Карпінський Н. К.
Луцький національний технічний університет
E-mail: n.yakymchuk@lntu.edu.ua

ОРГАНІЗАЦІЯ UART-КОМУНІКАЦІЇ В STM32 ДЛЯ СИСТЕМИ КОМУТАЦІЇ КІНЦЕВИХ ПРИСТРОЇВ

У статті досліджено апаратні та програмні аспекти організації асинхронної передачі даних у мікроконтролерах STM32 для створення системи комутації кінцевих пристроїв. Розглянуто підключення через USB-UART перетворювач FT232, конфігурацію інтерфейсу UART у середовищі STM32CubeMX та механізми забезпечення надійності обміну. Проведений аналіз демонструє можливість використання STM32 як комунікаційного вузла у телекомунікаційних застосуваннях.

Ключові слова: STM32, UART, комутація, FT232, телекомунікації.

N. M. Yakymchuk, N. K. Karpinsky. Organization of UART communication in STM32 for the terminal device switching system. The article investigates the hardware and software aspects of

organizing asynchronous data transmission in STM32 microcontrollers for building an end-device switching system. The study examines the connection via an FT232 USB-UART converter, the configuration of the UART interface in STM32CubeMX, and the mechanisms ensuring communication reliability. The analysis demonstrates that STM32 can effectively operate as a communication node in modern telecommunication applications.

Keywords: STM32, UART, switching, FT232, telecommunications.

Постановка проблеми. У сучасній науковій та технічній літературі значна увага приділяється проблемам організації надійного телекомунікаційного обміну між кінцевими пристроями на базі систем-на-кристалі. У працях, присвячених архітектурі STM32, детально описано апаратну реалізацію послідовних інтерфейсів, зокрема UART та USART, а також наведено їхні функціональні характеристики, що визначають можливості побудови комунікаційних модулів різного призначення [1; 2]. Дослідження програмної конфігурації периферії за допомогою засобів STM32CubeMX дозволяє розглядати мікроконтролери STM32 як гнучку платформу для побудови систем збору та маршрутизації даних у телекомунікаційних вузлах, де важливими є точність синхронізації, енергоефективність та надійність [4].

У наукових виданнях, присвячених вбудованим системам, автори наголошують на важливості оптимізації послідовної передачі даних у режимах з інтенсивним інформаційним навантаженням, а також аналізують механізми апаратної підтримки, які забезпечують мінімізацію затримок і помилок під час передавання повідомлень у режимі реального часу [5; 6]. Окремі праці звертають увагу на роль USB-UART перетворювачів як інтерфейсної ланки між мікроконтролером і комп'ютером, що є ключовим для дослідження процесів комутації кінцевих пристроїв у телекомунікаційних системах [3].

У галузевих публікаціях наведено аналіз ефективності асинхронних протоколів у структурі вбудованих мереж та підкреслено значення контролю помилок, коректності часових параметрів і адаптивності каналів передавання у складних телекомунікаційних умовах [7]. Це підтверджує потребу у комплексному дослідженні комутаційних механізмів, що базуються на UART-інтерфейсі STM32, з урахуванням апаратної специфіки, програмної конфігурації та реальних умов експлуатації.

Проведений огляд свідчить, що, незважаючи на широкий спектр доступних інструментів та технічної документації, питання інтегрованої організації каналу комутації для кінцевих пристроїв з використанням STM32 у ролі комунікаційного вузла лишається актуальним. Особливої уваги потребує дослідження стабільності, детермінованості та відмовостійкості UART-каналу, оскільки саме ці параметри визначають надійність роботи сучасних телекомунікаційних систем.

Метою дослідження є аналіз апаратних та програмних засобів організації UART-комунікації в STM32 та оцінювання її придатності для системи комутації кінцевих пристроїв.

Основний матеріал дослідження. Апаратна організація асинхронного зв'язку між мікроконтролером STM32 і персональним комп'ютером є фундаментально важливою для побудови телекомунікаційних систем низького та середнього рівня складності. Інтерфейс UART, реалізований апаратно у всіх сучасних контролерах STM32, забезпечує простий і надійний канал обміну між вузлами, де дані передаються у вигляді послідовного потоку бітів. Оскільки персональні комп'ютери не мають фізичного UART-порту, використовується USB-UART перетворювач, найпоширенішим варіантом якого є мікросхема FT232R. Її застосування дозволяє сформувати універсальний міст між TTL-рівнями мікроконтролера та USB-інтерфейсом ПК.

На рисунку 1 подано схему, у якій сигнальні лінії TX і RX STM32 з'єднуються безпосередньо з відповідними входами FT232R. Лінія TX мікроконтролера спрямована на RX перетворювача, що забезпечує коректний напрям потоку даних, тоді як сигнальна лінія RX контролера отримує інформацію з TX FT232R. Додатково використовуються загальна шина живлення та спільна точка землі, що гарантує правильність рівнів сигналів і завадостійкість каналу.

Таке підключення реалізує базову модель асинхронної взаємодії, в якій обмін даними відбувається без зовнішнього тактового сигналу, а синхронізація забезпечується тривалістю бітів та структурою кадру UART.

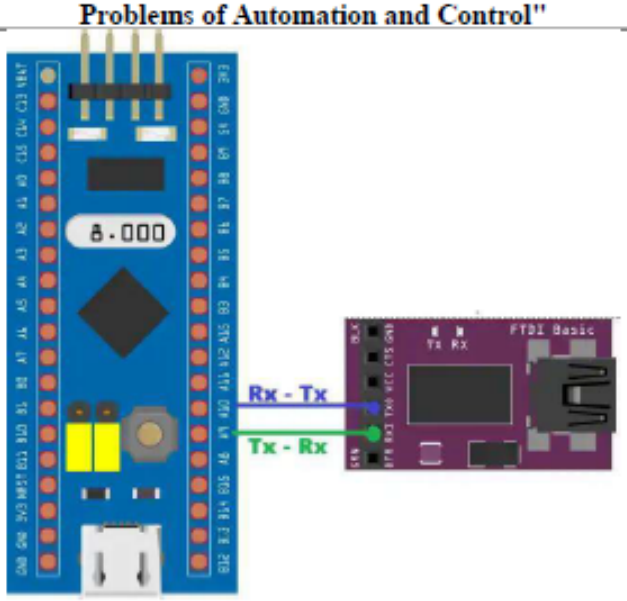


Рисунок 1 – Зв'язок між MCU та FT232 USB за допомогою UART конвертера

Інтерфейс UART є одним з ключових засобів обміну між STM32 та периферійними модулями. Його апаратна реалізація включає буферизацію, генерацію переривань та можливість використання DMA, що зменшує навантаження на ядро мікроконтролера та забезпечує стабільність передавання даних у системах реального часу. У контексті підключення через FT232R ці переваги дозволяють формувати надійний послідовний канал для конфігурації модуля, збору телеметрії, передавання параметрів або діагностичної інформації у телекомунікаційних вузлах.

Згідно з практичними рекомендаціями, конфігурація UART повинна відповідати параметрам, які встановлені драйвером FT232R, зокрема швидкості передачі, кількості бітів даних та стоп-бітів. На комп'ютері FT232R створює віртуальний COM-порт, а STM32 виступає в ролі джерела або приймача даних, що надходять у ASCII-форматі. Такий спосіб з'єднання широко застосовується в системах тестування, відлагодження та передачі керуючих команд.

Особливу увагу слід приділити електричній сумісності. UART у STM32 працює на рівнях 3.3 В, тому FT232R повинен бути налаштований на аналогічну напругу живлення або містити логічний перетворювач. Наявність неправильно узгоджених рівнів може призвести до помилок передачі або апаратних збоїв, що є критичним для телекомунікаційних систем. У реальних застосуваннях додатково враховують вплив завад, довжину ліній передачі та можливість екранізації кабелю.

Коректне функціонування телекомунікаційного вузла на базі STM32 забезпечують програмні механізми, зокрема середовище STM32CubeMX. В умовах побудови системи комутації кінцевих пристроїв базовим елементом є модуль, що здатний приймати, обробляти та передавати інформаційні пакети у вигляді послідовних телеметричних або керуючих повідомлень. Саме інтерфейс UART виступає фундаментальним каналом комунікації, а його конфігурація визначає якість взаємодії у системі. Передавання даних у STM32 реалізується через апаратний блок USART, який у асинхронному режимі виконує функції UART, забезпечуючи формування кадру з урахуванням стартового біта, інформаційних бітів, бітів парності та стоп-біта. Вибір цих параметрів є критичним для уникнення помилок і узгодження мікроконтролера з USB-UART перетворювачем FT232R. Саме тому першим кроком налаштування є синхронізація швидкості передачі та формату кадру з параметрами віртуального COM-порту, створеного комп'ютером.

У середовищі STM32CubeMX процес конфігурації починається з активації периферії USARTx у режимі Asynchronous. Після вибору частоти тактування шини та налаштування мультиплексування виводів TX і RX середовище автоматично генерує код ініціалізації, у якому фіксуються встановлені параметри. Це дозволяє зменшити ймовірність помилок, що виникають при ручному програмуванні регістрів, і забезпечує уніфікованість конфігурації для різних серій STM32. Важливим кроком є визначення функцій, які керуватимуть передаванням та прийманням даних. У бібліотеці HAL передбачено використання функцій блокуючого, неблокуючого та DMA-режимів. Кожен із них має свої переваги залежно від вимог телекомунікаційного застосування. У простих системах комутації доцільно використовувати блокуючий режим, оскільки він забезпечує повну

завершеність передачі перед переходом до наступної операції. У більш складних системах, які одночасно обробляють декілька потоків комунікаційних подій, ефективнішими є неблокуючі або DMA-підходи.

Передавання інформації у UART-каналі базується на формуванні буфера даних та ініціації функції надсилання, яка перетворює масив байтів у послідовність бітів відповідно до встановленої структури кадру. Прийом даних реалізується за аналогічним принципом, однак потребує ретельнішого контролю помилок, оскільки у телекомунікаційних мережах можливі завади, нестабільність сигналу або порушення таймінгів. Застосування пріоритетних переривань дозволяє миттєво реагувати на появу нового байта у приймальному регістрі, що є важливим у системах, де одночасно обробляються команди комутації та інформаційні повідомлення. Цей підхід дає змогу уникнути втрати даних і забезпечити детерміновану реакцію мікроконтролера навіть у випадку інтенсивного інформаційного навантаження.

Особливого значення набуває механізм DMA, який дозволяє автоматично передавати блоки даних між пам'яттю та UART-периферією без участі центрального процесора. Така можливість суттєво підвищує продуктивність системи комутації, оскільки ядро STM32 звільняється для виконання задач маршрутизації, аналізу команд або керування периферійними модулями. DMA мінімізує затримки та забезпечує сталість швидкості передачі, що важливо в умовах багатоканальної телекомунікаційної архітектури. Використання DMA-підходу особливо доцільне, коли передаються структуровані пакети, у яких STM32 виконує роль вузлового комутатора та повинен перенаправляти отримані повідомлення іншим пристроям.

Дослідження системи комутації кінцевих пристроїв на базі STM32 вимагає проведення аналізу надійності обміну та особливостей функціонування асинхронного каналу в умовах реальних телекомунікаційних навантажень. Після апаратного підключення та програмної конфігурації UART важливо оцінити стабільність передачі, точність синхронізації та здатність системи своєчасно реагувати на зміни стану лінії. У практичних реалізаціях робота каналу визначається структурою кадру UART, де помилки можуть виникати внаслідок некоректної тривалості біта, завад або перевищеної тривалості затримки між байтами. Мікроконтролери STM32 містять апаратні механізми виявлення таких порушень, включаючи контроль парності, детекцію framing error та overrun error, що дозволяє формувати детермінований канал комунікації навіть у складних умовах роботи.

У системах комутації суттєве значення має своєчасність обробки даних, оскільки затримки можуть спотворювати логіку взаємодії між вузлами мережі. Для мінімізації таких ризиків процес прийому зазвичай організовується через переривання або DMA, що дає змогу виключити пропуски даних і забезпечити постійну готовність приймача. Важливою є також узгодженість швидкості передачі між STM32 та FT232R, оскільки навіть невеликі відхилення тактових частот можуть спричинити накопичення похибок у довгих сеансах обміну. Окрім цього, при побудові телекомунікаційного вузла необхідно враховувати електричні параметри лінії, включаючи рівень напруги, довжину провідника та чутливість до електромагнітних завад, що особливо актуально для комутаційних модулів, де одночасно працюють кілька інтерфейсів.

Для підтвердження теоретичних положень було проведене дослідження, у вигляді у створенні локального комунікаційного вузла на базі STM32, підключеного до ПК через FT232 та до периферійних модулів через UART. Мікроконтролер приймав дані, аналізував їх зміст і переспрямовував відповідним пристроям, моделюючи роботу маршрутизатора. Експеримент проводився на STM32F103 з використанням двох UART-каналів: UART1 для з'єднання з ПК через FT232 та UART2 для підключення периферійного модуля. Швидкість передачі встановлювали 115200 бод, розмір пакета — 16 байт. Під час експерименту оцінювали стабільність прийому, швидкодію обробки та коректність пересилання пакетів у режимі реального часу. Особливу увагу приділяли відсутності втрат даних та відповідності реакції STM32 часовим вимогам комутаційного процесу. Особливу увагу під час експериментального дослідження було зосереджено на здатності STM32 виконувати роль маршрутизуючого елемента в локальній системі комутації. Результати показали повну відсутність пропусків кадрів, коректну маршрутизацію та стійкість роботи каналу протягом тривалого тестування.

Завдяки поєднанню апаратної підтримки UART, гнучкості програмної конфігурації та можливості послідовної обробки даних, мікроконтролер забезпечив стабільне переспрямування повідомлень між кінцевими вузлами. Це підтверджує перспективність використання STM32 як основи для малогабаритних комунікаційних систем, у яких важливими є енергоефективність, надійність та адаптивність. Проведений аналіз демонструє, що за належної конфігурації UART-

підсистеми STM32 здатний функціонувати як універсальний комунікаційний модуль, здатний інтегруватися у ширші телекомунікаційні структури.

Висновки. Проведений аналіз засвідчив, що інтерфейс UART у поєднанні з мікроконтролером STM32 та FT232 забезпечує стійкий асинхронний канал зв'язку, придатний для використання у системах комутації кінцевих пристроїв. Завдяки апаратній підтримці переривань і DMA, а також можливості гнучкої конфігурації UART через STM32CubeMX, мікроконтролер забезпечує низькі затримки й високу надійність обміну. Отримані результати підтверджують ефективність STM32 як комунікаційного вузла в локальних телекомунікаційних системах та його перспективність для побудови маршрутизуючих модулів різного рівня складності.

ПЕРЕЛІК ПОСИЛАНЬ

1. STMicroelectronics. RM0008. Reference manual. STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced Arm®-based 32-bit MCUs. Rev., 2018. URL: https://www.st.com.

2. STMicroelectronics. AN4908. Getting started with USART automatic baud rate detection for STM32 MCUs. Rev. 8, 2024. URL: https://www.st.com/resource/en/application_note/an4908-getting-started-with-usart-automatic-baud-rater-detection-for-stm32-mcus-stmicroelectronics.pdf.

3. STMicroelectronics. UM1850. Description of STM32F1 HAL and low-layer drivers (STM32CubeF1 user manual). 2020. URL: https://www.st.com.

4. STMicroelectronics. AN2548. Introduction to DMA controller for STM32 MCUs. Rev. 9, February 2024. URL: https://www.st.com/resource/en/application_note/an2548-introduction-to-dma-controller-for-stm32-mcus-stmicroelectronics.pdf.

5. Future Technology Devices International Ltd. FT232R. USB UART IC Datasheet. DS_FT232R, 2024. URL: https://ftdichip.com

6. Zang F., Niu H. Design and Implementation of High Speed UART Based on DMA. Proc. 3rd Int. Conf. on Electrical, Automation and Mechanical Engineering (EAME). 2018. DOI:10.2991/eame-18.2018.26.

7. Bardis N. Synchronization error correction for asynchronous channels data transmission. ITM Web of Conferences (AMCSE 2017), 2017. DOI: 10.1051/ itmconf/201 7090 3007.