Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Мікропроцесорна система реєстрування аудіосигналів»

08-54.БКР.010.00.000 ПЗ

Виконав: студент 4 курсу, групи 1КІ-21б
спеціальності 123 — «Комп'ютерна інженерія»
освітня програма — «Комп'ютерна інженерія»

_____ Киричок Л.В.

Керівник: к.т.н., доц. каф. ОТ

_____ Савицька Л.А.

Рецензент: доц. каф ПЗ

_____ Хошаба О.М

Допущено до захисту:
завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.

«20» 06 2025 р.

Вінниця ВНТУ 2025 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень бакалавр
Галузь знань — 12 Інформаційні технології
Спеціальність — 123 Комп'ютерна інженерія
Освітня програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. _____О. Д. Азаров
« 21 » березня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Киричоку Лукя'ну Вікторовичу

1 Тема роботи «Мікропроцесорна система реєстрування аудіосигналів», керівник роботи Савицька Людмила Анатоліївна, к.т.н., доцент, затверджені наказом вищого навчального закладу від 20 березня 2025р. № 97.

2 Строк подання студентом роботи 13.05.2025 р.

3 Вихідні дані до роботи: необхідні технічні характеристики елементів необхідних для створення мікропроцесорної системи реєстрування аудіо сигналів.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз сучасного стану та вибір напрямку дослідження, теоретичні основи розробки систем реєстрації аудіосигналів, проектування мікропроцесорної системи реєстрації аудіо сигналів, висновки.

5 Перелік графічного матеріалу: Головний цикл роботи системи реєстрування аудіо сигналів.

Перелік ілюстративного матеріалу: Дослідження роботи схеми підключення мікрофона та динаміка.

6 Консультанти розділів роботи представлені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-3	к.т.н., доц. каф. ОТ Савицька Л.А.	21.03.2025 р.	13.05.2025 р.
Нормоконтроль	асистент каф. ОТ Швець С.І.	13.05.2025 р.	30.05.2025 р.

7 Дата видачі завдання 21.03.2025 р.

8 Календарний план наведено в таблиці 2.

Таблиця 2—Календарний план

№	Назва етапів виконання бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи. Вступ	21.03-23.03.25	
2	Аналіз сучасного стану та вибір напрямку дослідження	24.03-30.03.25	
3	Теоретичні основи розробки систем реєстрації аудіосигналів	01.04-21.04.25	
4	Розробка структурної та принципової електричної схеми	22.04-05.05.25	
5	Розробка програмного забезпечення	06.05-10.05.25	
6	Оформлення пояснювальної записки та ілюстративного матеріалу	11.05-12.05.25	
7	Перевірка якості виконання бакалаврської роботи та усунення недоліків	13.05.25	

Студент Лук'ян КИРИЧОК

Керівник роботи к.т.н., доц. ЛюдмилаСАВИЦЬКА

АНОТАЦІЯ

УДК 621.396.6

Киричок Л. В. Мікропроцесорна система реєстрування аудіосигналів. Бакалаврська кваліфікаційна робота зі спеціальності 123 «Комп'ютерна інженерія», освітня програма «Системне програмування». Вінниця: ВНТУ, 2025. 71с.

На укр. мові. Бібліогр.: 20 назв, рис.35, табл. 4,.

У бакалаврській кваліфікаційній роботі розроблено мікропроцесорну систему для реєстрації аудіосигналів у цифровому форматі з подальшим збереженням і передачею даних. У процесі дослідження розглянуто особливості побудови систем збирання аудіоданих, проаналізовано варіанти апаратної реалізації та алгоритми обробки сигналів.

У роботі реалізовано схему з використанням мікроконтролера з підтримкою аудіоінтерфейсу, мікрофонного підсилювача та модуля збереження даних. Розроблено програмне забезпечення, що забезпечує захоплення, оцифрування, обробку та збереження аудіосигналів. Система може працювати автономно або передавати дані на зовнішні пристрої через інтерфейси зв'язку.

Проєктована система має перспективу використання в мобільних пристроях для польових досліджень, побутових системах моніторингу, а також у наукових експериментах, де потрібна автономна реєстрація звукових подій.

Ключові слова: аудіосигнал, мікроконтролер, оцифрування, реєстрація, інтерфейс, збереження даних, електроніка.

ABSTRACT

Kyrychok L. V. Microprocessor system for recording audio signals. Bachelor's qualification work in specialty 123 "Computer Engineering", educational program "System Programming". Vinnytsia: VNTU, 2025. 71 p.

In Ukrainian. Bibliography: 20 titles, fig. 35, table. 4,.

In the bachelor's qualification work, a microprocessor system for recording audio signals in digital format with subsequent data storage and transmission was developed. In the process of research, the features of building audio data collection systems were considered, hardware implementation options and signal processing algorithms were analyzed.

In the work, a scheme using a microcontroller with audio interface support, a microphone amplifier and a data storage module was implemented. Software was developed that provides capture, digitization, processing and storage of audio signals. The system can operate autonomously or transmit data to external devices via communication interfaces.

The designed system has the potential to be used in mobile devices for field research, household monitoring systems, as well as in scientific experiments where autonomous registration of sound events is required.Keywords: audio signal, microcontroller, digitization, recording, interface, data storage, electronics.

ЗМІСТ

ВСТУП

Актуальністьтеми пов’язана з тим, що з розвитком вбудованих систем, мікроконтролери зайняли ключове місце в сучасних електронних приладах, забезпечуючи керування, обробку сигналів і взаємодію з користувачем. Особливої популярності набувають компактні системи реєстрації аудіосигналів, що дозволяють здійснювати запис, зберігання та відтворення звуку з мінімальними апаратними ресурсами. Такий підхід є затребуваним у багатьох сферах — від побутової електроніки до спеціалізованих технічних пристроїв, включно з охоронними комплексами, диктофонами, аудіоаналізаторами та реєстраторами подій.

Потреба в ефективних та доступних рішеннях для обробки аудіосигналів спонукає до розробки систем на основі простих, проте функціональних мікроконтролерів, які здатні реалізувати оцифрування, передавання та збереження звуку. Використання вбудованих АЦП, периферійних інтерфейсів (наприклад, SPI) і ШІМ-модуляції дає змогу створювати повноцінні пристрої аудіозапису без складних зовнішніх модулів.

Мета роботи — розробити мікропроцесорну систему, здатну реєструвати, зберігати та відтворювати аудіосигнали за допомогою мікроконтролера AVR і зовнішньої Flash-пам’яті, використовуючи прості інтерфейси та методи обробки сигналу.

Завдання дослідження:

— дослідити сучасні підходи до побудови систем аудіозапису на базі мікроконтролерів;

— здійснити вибір мікроконтролера та зовнішніх компонентів для зберігання та відтворення звуку;

— розробити електричну та функціональну схеми пристрою;

— реалізувати алгоритм обробки аудіосигналу в мікроконтролері;

— створити програмне забезпечення з підтримкою АЦП, SPI і ШІМ;

— протестувати систему та оцінити якість і стабільність реєстрації звуку.

Об’єкт дослідження — мікропроцесорна система для обробки звукових сигналів.

Предмет дослідження — архітектура мікроконтролерів, методи цифрового перетворення аудіосигналів, передача та збереження даних у зовнішній пам’яті, програмна обробка сигналів.

Практичне значення полягає у створенні функціонального прототипу пристрою, здатного реєструвати та відтворювати звук із використанням мінімального набору компонентів. Запропонована система може використовуватись у якості лабораторного стенду, основи для побудови диктофонів, сигналізаторів або систем звукового оповіщення. Завдяки застосуванню мікроконтролера AVR і Flash-пам’яті з SPI-інтерфейсом досягається висока гнучкість конфігурації та можливість масштабування пристрою.

У процесі роботи застосовано такі м**етоди дослідження**, якметоди цифрової обробки сигналів, моделювання апаратної логіки, алгоритмічного проєктування, тестування електронних схем, а також структурно-функціонального аналізу роботи мікроконтролера.

Запропоновано комплексне рішення для запису, збереження та відтворення звуку на основі мікроконтролера з вбудованим АЦП, що працює в парі з енергонезалежною Flash-пам’яттю, використовуючи SPI-протокол. Система відтворює сигнал за допомогою ШІМ, забезпечуючи компактність та енергоефективність. Особливістю реалізації є програмна автономність і можливість використання змінних параметрів без перепрошивки мікроконтролера.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ТА ВИБІР НАПРЯМКУ ДОСЛІДЖЕННЯ

1.1 Актуальність задачі реєстрації аудіосигналів у сучасних системах

Реєстрація аудіосигналів є однією з фундаментальних задач у багатьох галузях науки і техніки, починаючи від медичних досліджень та закінчуючи системами безпеки, зв'язком, мультимедіа та інтернетом речей (IoT). Сучасний розвиток цифрових технологій, а особливо мікропроцесорної техніки, відкрив нові можливості для удосконалення процесів захоплення, обробки та збереження звукової інформації. Високоточні цифрові системи запису аудіо знайшли широке застосування в системах відеоспостереження, автономних реєстраторах подій, медичних апаратах, де контроль якості звуку і достовірність відтворення є критично важливими.

Одним із ключових чинників, що визначають ефективність системи реєстрації аудіосигналів, є вибір апаратної бази — мікроконтролера з відповідними характеристиками. Сучасні мікроконтролери мають вбудовані аналого-цифрові перетворювачі (АЦП), що дозволяють оцифровувати звуковий сигнал з високою роздільною здатністю та швидкістю, що забезпечує точне відтворення оригіналу. Наявність швидких і надійних послідовних інтерфейсів, таких як SPI чи I2C, дозволяє ефективно працювати з зовнішніми Flash-пам'яттю, що забезпечує тривале збереження великих обсягів аудіо даних (див. рис.1.1).

Важливим аспектом є також програмне забезпечення, яке має реалізовувати оптимальні алгоритми стиснення, збереження і відтворення сигналів, підтримувати інтерфейс користувача для гнучкого управління процесом запису і відтворення. Використання технологій широтно-імпульсної модуляції (ШІМ) дає змогу ефективно відтворювати аудіосигнал через зовнішні динаміки, забезпечуючи при цьому низьке енергоспоживання пристрою. Структуру каналів перетворення аудіосигналів показано на рисунку 1.2.

Особливу актуальність тема набуває у зв'язку з широким впровадженням портативних і мобільних пристроїв, для яких важливі компактність, автономність, низька вартість і простота використання. Крім того, розвиток смарт-систем і

індустрії 4.0 вимагає наявності високоякісних аудіорішень для інтеграції з голосовими асистентами, системами моніторингу і діагностики.

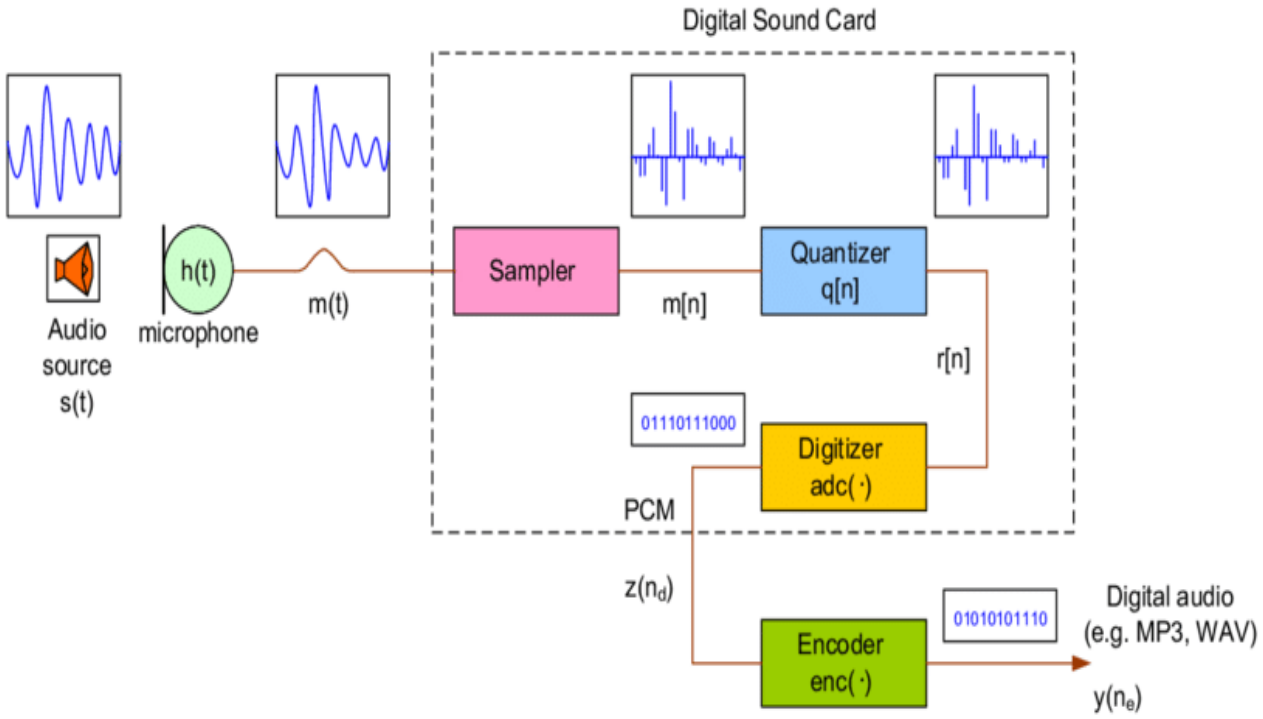


Рисунок 1.1 — Типова структура системи цифрового аудіозапису

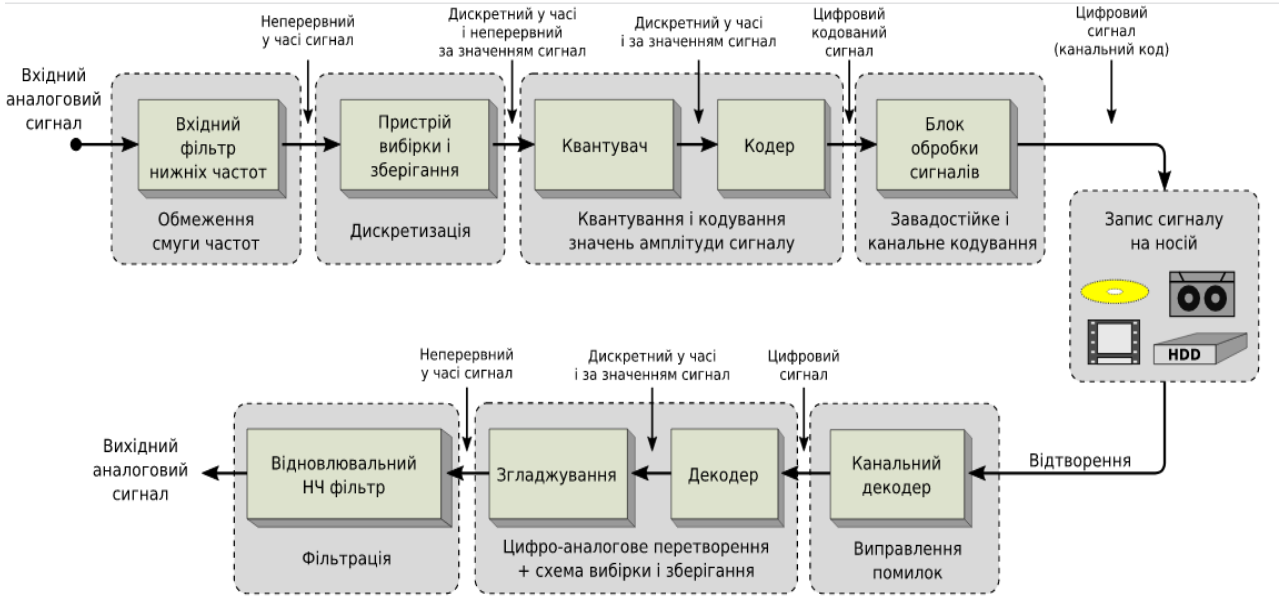


Рисунок 1.2 — Структуру каналів перетворення аудіосигналів

Отже, розробка мікропроцесорної системи реєстрації аудіосигналів є не тільки актуальним, а й перспективним напрямком інженерної діяльності.

1.2 Формування мети та основних завдань проєкту

Основна мета цієї бакалаврської роботи полягає у створенні мікропроцесорної системи, яка здатна якісно записувати, зберігати і відтворювати аудіосигнали з високою точністю, забезпечуючи зручність управління та адаптивність до різних умов експлуатації.

Для досягнення поставленої мети було сформульовано комплекс завдань, що охоплюють усі етапи розробки системи — від вибору технічної бази до тестування кінцевого продукту:

— детально проаналізувати сучасні підходи до цифрової реєстрації аудіосигналів, включаючи апаратні платформи і програмні методики, зокрема на основі мікроконтролерів сімейств AVR, STM32, ESP32 та інших;

— визначити оптимальні технічні характеристики мікроконтролера (частота роботи, розрядність АЦП, обсяг оперативної пам’яті, підтримка периферії) для забезпечення високоякісного аудіо запису;

— обрати тип і конфігурацію зовнішньої пам’яті (Flash, EEPROM, SD-карта), здатної зберігати великі обсяги даних і швидко взаємодіяти з мікроконтролером через стандартні інтерфейси;

— розробити принципову схему електронної апаратної частини, що включатиме мікрофонний підсилювач, АЦП, пам’ять, а також блоки живлення та індикації стану;

— створити алгоритми цифрового захоплення сигналу, збереження у пам’яті і подальшого відтворення через ШІМ або інші звукові виходи, оптимізуючи їх з урахуванням обмежень апаратної платформи;

— розробити користувацький інтерфейс, який дозволить задавати параметри запису (тривалість, частоту дискретизації), керувати відтворенням і контролювати стан системи;

— провести комплексні випробування прототипу, оцінити якість звуку, стабільність роботи та енергоспоживання пристрою;

— запропонувати шляхи подальшого удосконалення системи, включаючи розширення функціональності (додаткові формати стиснення, розпізнавання голосу тощо).

Виконання цих завдань дасть змогу не лише реалізувати практичний пристрій, а й набути теоретичних знань про архітектуру та програмування мікроконтролерів, що актуально для подальшої професійної діяльності.

1.3 Оцінка доцільності розробки

Для обґрунтування доцільності розробки мікропроцесорної системи реєстрації аудіосигналів проведено техніко-економічний аналіз існуючих рішень і потенційних вигод від впровадження нової системи.

З технічної точки зору, сучасні мікроконтролери пропонують широкий набір функцій, що дозволяють реалізувати повний цикл обробки аудіосигналу на одному чипі. Використання вбудованого АЦП із розрядністю 10–12 біт та частотою дискретизації до 44,1 кГц забезпечує високу якість запису, порівнянну з професійним обладнанням початкового рівня. Підключення зовнішньої Flash-пам’яті через SPI дозволяє зберігати великі обсяги звукових файлів без втрати швидкості запису, що є важливою характеристикою для реєстраторів тривалого запису.

З економічної точки зору, застосування готових мікроконтролерних платформ значно знижує вартість розробки і виробництва пристрою. Компоненти доступні на ринку, мають гарну документацію і підтримку спільноти, що скорочує час і зусилля розробників. Вартість системи, з огляду на її функціональність,

робить її привабливою як для індивідуального користувача, так і для промислових замовників, особливо в освітній сфері, де необхідні бюджетні, але якісні пристрої.

Крім того, запропоноване рішення має значний потенціал для масштабування — додавання нових функцій (цифрова фільтрація, компресія, інтеграція з бездротовими мережами) підвищить конкурентоспроможність продукту. Це дозволить використовувати систему у більш широкому спектрі застосувань: у телекомунікаціях, охоронних системах, біомедицині та промисловому моніторингу.

Враховуючи вище викладене, інвестиції в розробку мікропроцесорної системи реєстрації аудіосигналів є виправданими і мають перспективу отримання економічного і технологічного ефекту.

1.4 Перспективи розвитку мікропроцесорних систем реєстрації аудіосигналів

Сучасний рівень розвитку цифрової електроніки та програмного забезпечення створює широкі можливості для удосконалення мікропроцесорних систем реєстрації аудіосигналів. Майбутній розвиток цієї сфери тісно пов’язаний із зростанням вимог до якості звуку, обробки великих обсягів інформації, а також інтеграції з інтелектуальними системами та мережевими сервісами.

Першою важливою тенденцією є підвищення роздільної здатності та точності оцифрування аудіосигналів. Нові покоління аналого-цифрових перетворювачів (АЦП) мають все кращі характеристики — знижений рівень шумів, більшу швидкодію і ширший діапазон частот. Це дозволить системам реєструвати звуки з більшою деталізацією, що особливо актуально для медичних застосувань, акустичних досліджень і професійної аудіотехніки.

Другим напрямком є розвиток алгоритмів обробки аудіоданих на рівні мікропроцесорів. Завдяки зростанню обчислювальної потужності сучасних мікроконтролерів і впровадженню спеціалізованих цифрових сигнальних процесорів (DSP), стає можливим виконувати складні операції фільтрації, шумозаглушення, стиснення та аналізу аудіосигналів безпосередньо на пристрої в

режимі реального часу. Це відкриває можливості для автоматичного розпізнавання голосу, діагностики стану навколишнього середовища або інтелектуального моніторингу.

Третім важливим напрямком є розширення функціональності пристроїв шляхом інтеграції з бездротовими інтерфейсами (Wi-Fi, Bluetooth, ZigBee) та хмарними платформами. Це дозволяє створювати системи для віддаленого запису та контролю аудіоінформації, що актуально для систем безпеки, моніторингу здоров’я, інтернету речей (IoT) і смарт-будинків. Така інтеграція дає змогу користувачам отримувати доступ до аудіоданих із будь-якої точки світу, аналізувати їх за допомогою хмарних сервісів і реалізовувати складні сценарії автоматизації.

Четвертим перспективним напрямком є впровадження штучного інтелекту та машинного навчання для аналізу аудіосигналів. Використання нейронних мереж дозволяє підвищити якість розпізнавання звуків, покращити алгоритми шумозаглушення, автоматично класифікувати аудіоподії та адаптувати роботу системи до змін у навколишньому середовищі. Ці технології можуть суттєво підвищити функціональні можливості мікропроцесорних систем, роблячи їх більш «розумними» та адаптивними.

П’ятий напрямок — мініатюризація та енергозбереження. З огляду на зростаючу популярність мобільних і портативних пристроїв, розробка компактних, економних у споживанні енергії рішень залишається пріоритетною задачею. Сучасні мікроконтролери з низьким енергоспоживанням, використання енергоефективних алгоритмів та акумуляторів нового покоління дозволять збільшити автономність систем і розширити сфери їх застосування.

Загалом, розвиток мікропроцесорних систем реєстрації аудіосигналів відбувається на перетині інновацій у галузях електроніки, програмування, телекомунікацій і штучного інтелекту. Ці тенденції створюють умови для появи нових пристроїв із широким функціоналом, високою якістю звуку та гнучкістю у

використанні, що відкриває перспективи застосування у найрізноманітніших сферах — від науки і медицини до побуту і промисловості.

1.5 Виклики та проблеми при розробці мікропроцесорних систем реєстрації аудіосигналів

Розробка мікропроцесорних систем для реєстрації аудіосигналів супроводжується низкою технічних та організаційних викликів, які необхідно враховувати для створення ефективного і надійного пристрою. Ці проблеми стосуються як апаратної, так і програмної частини системи, а також питання інтеграції та експлуатації.

1.5.1 Високі вимоги до якості сигналу та точності оцифрування

Однією з головних проблем є забезпечення достатньої роздільної здатності та частоти дискретизації аудіосигналу. Недостатня якість АЦП або неправильний вибір параметрів перетворення може призводити до спотворень, шумів і втрати важливої інформації. Враховуючи, що багато застосувань потребують точного і чистого відтворення звуку (наприклад, медична діагностика чи наукові дослідження), розробникам доводиться шукати компроміс між апаратними можливостями та ціною компонентів.

1.5.2 Обмежені ресурси мікроконтролерів

Мікроконтролери мають обмежені обчислювальні потужності, обсяг оперативної пам’яті та енергоспоживання. Це створює складності при реалізації складних алгоритмів обробки аудіо, таких як фільтрація, стиснення або інтелектуальний аналіз. Особливо це актуально для портативних пристроїв, де енергозбереження є пріоритетом. Розробникам необхідно оптимізувати код, застосовувати апаратне прискорення та вибирати відповідні архітектури для досягнення балансу між продуктивністю та споживанням енергії.

1.5.3 Надійне зберігання великих обсягів аудіо даних

Аудіозаписи, особливо високої якості, потребують значних обсягів пам’яті. Використання зовнішньої Flash-пам’яті чи інших накопичувачів ускладнює апаратну частину, вимагає грамотного керування інтерфейсами (SPI, I2C), а також надійних алгоритмів запису та читання для запобігання втраті даних або корупції файлів. Крім того, необхідно враховувати знос флеш-пам’яті при частих циклах запису.

1.5.4 Забезпечення стабільної роботи в реальному часі

Для багатьох застосувань критичною є можливість записувати і відтворювати аудіосигнал у режимі реального часу без затримок та пропусків. Реалізація такої функціональності вимагає ефективного управління пріоритетами завдань в мікроконтролері, використання буферизації та синхронізації даних. Недотримання цих вимог може призвести до деградації якості звуку та зниження функціональності системи.

1.5.5 Інтерфейс користувача і простота управління

Навіть технічно досконалі пристрої можуть залишатися непопулярними, якщо їх інтерфейс складний і незручний для користувача. Створення інтуїтивно зрозумілого меню, можливість гнучкого керування параметрами запису і відтворення, а також відображення статусу системи є важливою частиною розробки. При цьому обмежені ресурси дисплеїв та кнопок на малих пристроях накладають додаткові обмеження на дизайн інтерфейсу.

1.5.6 Захист від зовнішніх перешкод та безпека даних

Аудіосигнали, особливо у випадках використання в медичних або безпекових системах, мають бути захищені від електромагнітних завад, а також від несанкціонованого доступу та втручання. Розробка систем з високим рівнем

екранування, застосування алгоритмів шифрування і автентифікації користувачів є складними, але необхідними заходами для забезпечення надійності та конфіденційності.

1.5.7 Сумісність та інтеграція з іншими системами

В сучасних комплексних рішеннях мікропроцесорні системи реєстрації аудіо часто мають взаємодіяти з іншими пристроями, мережами або хмарними сервісами. Забезпечення сумісності протоколів обміну даними, стандартизація форматів аудіофайлів і безперебійна інтеграція в екосистему — це окремі виклики, що вимагають комплексного підходу і додаткових ресурсів розробників.

Таким чином, для створення конкурентоздатних мікропроцесорних систем реєстрації аудіосигналів необхідно враховувати численні технічні, апаратні та програмні виклики. Розв’язання цих проблем вимагає не лише глибоких знань, а й інноваційних підходів, що відкривають широкі перспективи для подальшого розвитку цієї галузі.

2 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ СИСТЕМ РЕЄСТРАЦІЇ АУДІОСИГНАЛІВ

2.1 Аналіз функціональних вимог до системи аудіозапису

Сучасні цифрові системи запису аудіосигналів мають чітко визначені функціональні вимоги, які забезпечують їхню ефективність, надійність та зручність експлуатації. В рамках технічного завдання передбачається розробка автономного цифрового пристрою для запису звукової інформації, здатного працювати безперервно протягом 60 хвилин та гарантувати довготривале збереження записаних даних навіть при відключенні живлення.

Основою пристрою є мікроконтролер, що виконує функції керування всіма етапами обробки аудіосигналу. Завдяки високій швидкодії та багатофункціональній периферії, вбудованій в кристал, мікроконтролер здатен реалізувати операції зі зчитування сигналу, його цифрової обробки та керування зовнішніми модулями. Для довготривалого зберігання аудіоданих пристрій використовує зовнішню Flash-пам’ять, яка підключається до мікроконтролера через високошвидкісний SPI-інтерфейс. Це дозволяє надійно зберігати великі обсяги інформації з можливістю швидкого доступу до них.

Запис звуку починається із захоплення аналогового сигналу за допомогою чутливого мікрофона, сигнал з якого підсилюється спеціальним операційним підсилювачем. Перед тим як подати сигнал на аналогово-цифровий перетворювач (АЦП) мікроконтролера, він проходить через фільтрацію для усунення шумів і небажаних частотних компонентів. Оцифрування відбувається з частотою дискретизації 8 кГц, що відповідає типовим вимогам до голосових застосунків, з розрядністю 8 біт, що оптимізує обсяг даних для збереження.

Після оцифрування аудіодані масштабуються та зберігаються у Flash-пам’яті, забезпечуючи надійне збереження навіть після вимкнення пристрою. Відтворення звуку організовується зворотнім процесом: збережені цифрові значення передаються мікроконтролеру, який формує аналоговий сигнал за допомогою широтно-імпульсної модуляції (ШІМ). Цей сигнал проходить через фільтр

згладжування і підсилювач, після чого відтворюється через динамік. Така архітектура дозволяє відтворити записаний звуковий потік з достатньою якістю для більшості прикладних завдань.

Інтерфейс користувача включає органи керування, підключені до портів введення-виведення мікроконтролера, що забезпечують індикацію режимів роботи пристрою та зручне управління процесом запису і відтворення. Це дозволяє оператору контролювати стан системи і здійснювати налаштування без втручання у програмний код.

Отже, проектована система повинна відповідати низці технічних і функціональних вимог, що охоплюють усі етапи життєвого циклу аудіосигналу: від його захоплення та цифрової обробки до збереження і якісного відтворення.

2.2 Огляд архітектури мікроконтролерів, що підтримують обробку аудіосигналів

Сучасні радіоелектронні пристрої все частіше базуються на мікроконтролерах (МК), що дає змогу значно спростити їх апаратну частину та підвищити універсальність. Заміна апаратних компонентів програмним забезпеченням дозволяє гнучко оновлювати функціонал пристрою, покращувати його характеристики і розширювати можливості без необхідності внесення апаратних змін.

При розробці систем, що оперують аудіосигналами, перед інженером стоїть низка важливих завдань: вибір типу процесора, комплектації периферійних модулів (таймери, АЦП, ЦАП, послідовні інтерфейси), вибір системи живлення, а також визначення економічної доцільності розробки. Ще нещодавно вибір мікроконтролерів був обмежений — зокрема, широко застосовувалися 8-бітні однокристальні мікроконтролери сімейства Intel 8051. Проте сьогодні ринок пропонує широкий спектр рішень від різних виробників: Zilog (серія Z86), Motorola (MC68), Microchip (PIC16/17), Thomson (ST62) і National Semiconductor (COP800) — кожен із яких має свої переваги та особливості.

Для вибору оптимального МК необхідно розглянути ключові характеристики та порівняти їх відповідно до задачі. Вибір часто визначається не лише технічними параметрами, а й традиціями розробника, наявністю інструментів, вартості і простотою програмування.

Згідно з дослідженнями, найбільшу популярність здобули сімейства AVR (розробка Atmel), MCS-51 (Intel) і PIC (Microchip). Зокрема, опитування показало (таблиця 2.1).

Таблиця 2.1 — Результати опитування популярності МК

Мікроконтролер	Кількість голосів
ASI	3 (0%)
AVR	309 (29%)
MCS-51	220 (21%)
PIC	317 (30%)
Z8	35 (3%)
Z80	21 (2%)
Інший	60 (5%)
DSP	33 (3%)
Ненавиджу МК	41 (3%)

З цього видно, що серед розробників та аматорів найбільш затребуваними є саме мікроконтролери AVR і PIC, причому Microchip відома своєю великою різноманітністю моделей і доступністю, а Atmel – інноваційним підходом у розробці.

Особливості сімейства AVR та мікроконтролера ATmega8515

Сімейство AVR від Atmel базується на RISC-архітектурі (Reduced Instruction Set Computing), що дозволяє значно підвищити швидкодію за рахунок оптимізації набору інструкцій та організації пам’яті. Однією з ключових переваг є гарвардська архітектура, при якій пам’ять програм і даних фізично розділені та мають окремі шини, що дає змогу одночасно звертатись до обох видів пам’яті.

Ще одна важлива особливість AVR — конвеєризація, яка дозволяє одночасно виконувати поточну інструкцію і завантажувати наступну, що забезпечує виконання більшості команд за один тактовий цикл. Це суттєво підвищує продуктивність пристроїв на базі AVR у порівнянні з традиційними CISC-архітектурами.

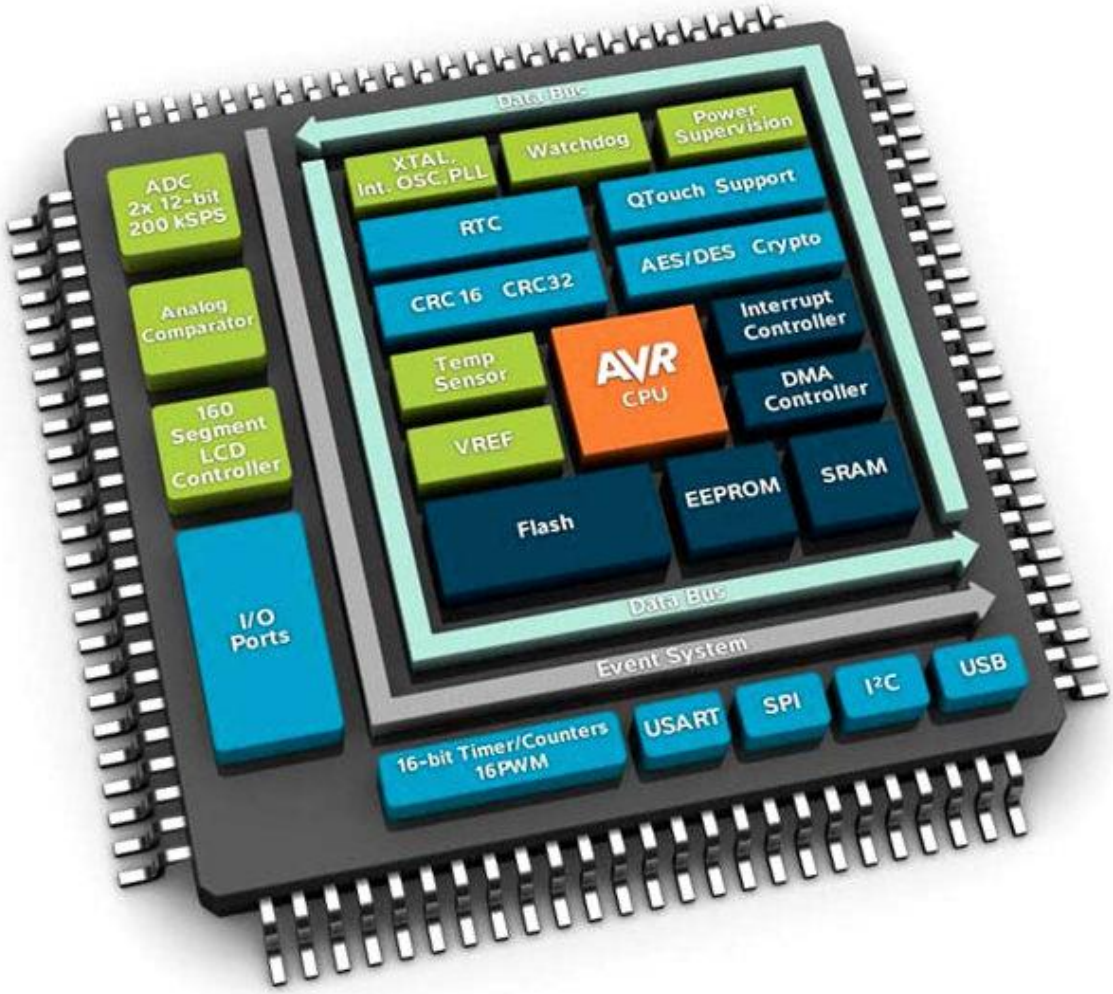


Рисунок 2.1 — Архітектура AVR

Нижче наведено технічні характеристики ATmega8515.

ATmega8515 — це 8-бітний CMOS-мікроконтролер, виконаний за технологією AVR RISC, який оптимально підходить для реалізації вбудованих систем із високими вимогами до швидкодії, низького енергоспоживання та

надійності. Завдяки виконанню однієї інструкції за такт, цей МК здатен працювати з частотою до 16 МГц, що дозволяє виконувати до 16 млн операцій в секунду.

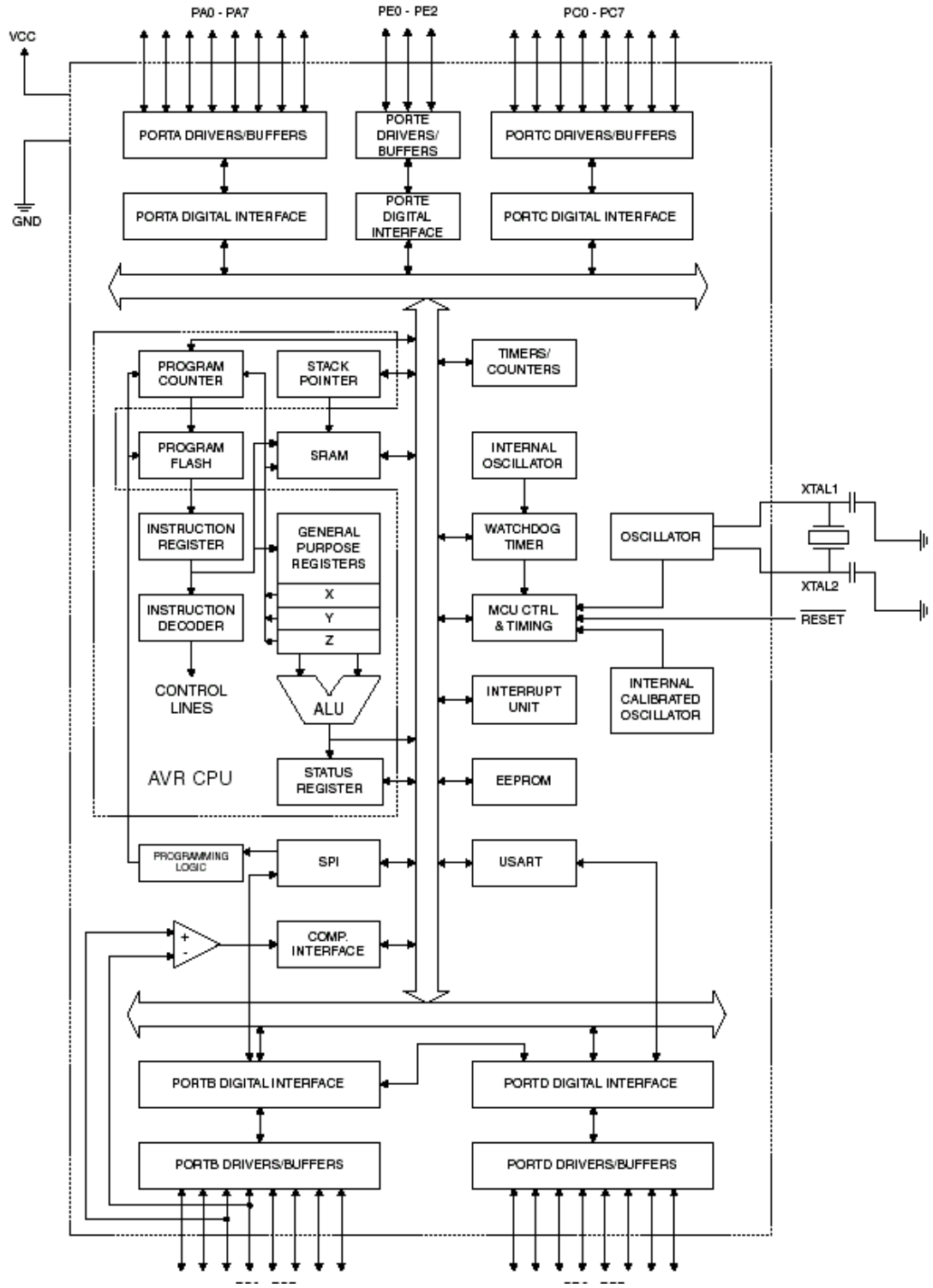


Рисунок 2.2 — Архітектура ATmega8515

Основні характеристики ATmega8515:

1) Рхітектура; розширена RISC з 130 інструкціями, зокрема інструкції множення, які виконуються за 2 такти;

2) Регістровий файл; 32 регістри загального призначення, які безпосередньо підключені до арифметико-логічного пристрою (ALU), що дозволяє виконувати операції між двома регістрами за один цикл;

3) Пам'ять:

— 8 КБ Flash-пам’яті з можливістю самопрограмування (до 10 000 циклів запису/стирання);

— 512 байт EEPROM для зберігання даних15;

— 512 байт статичної оперативної пам’яті (SRAM);

— підтримка зовнішньої пам’яті до 64 КБ;

4) Периферія:

— два таймери-лічильники (8- і 16-розрядний) з підтримкою режимів компаратора та захоплення фронтів;

— три канали широтно-імпульсної модуляції (PWM), що важливо для аудіообробки та управління потужністю;

— 10-бітний аналогово-цифровий перетворювач (АЦП) із мультиплексованими входами для підключення декількох джерел аналогового сигналу;

— вбудований аналоговий компаратор;

— програмований послідовний інтерфейс UART, SPI з режимами майстер/слейв;

— вбудований сторожовий таймер для надійності роботи;

5) Енергозбереження:

— три режими енергозбереження: холостий хід, знижене споживання, режим очікування;.

6) Інтерфейси введення/виведення (див. рис. 2.3):

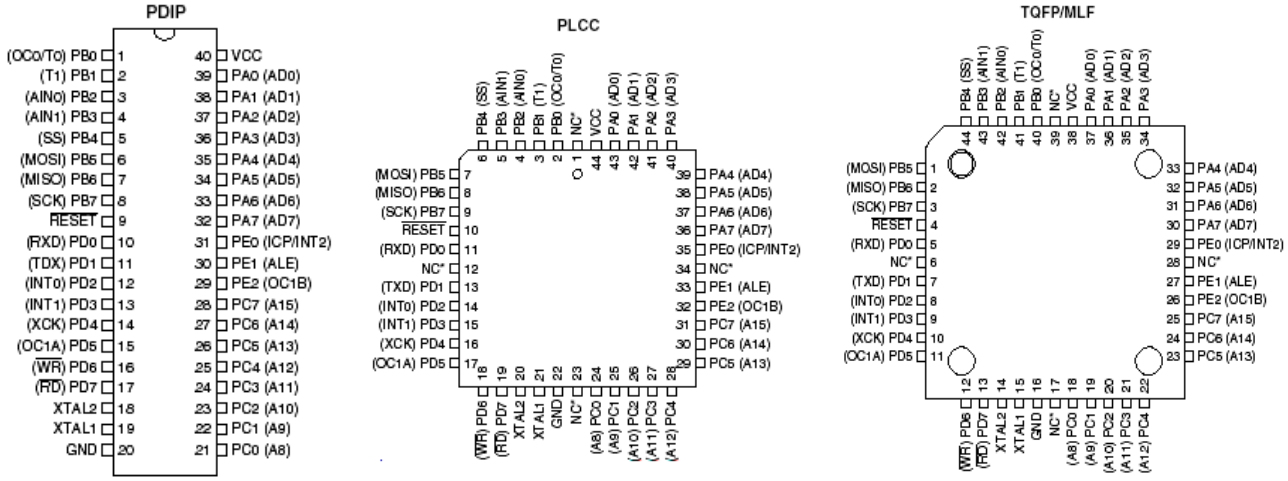
— 35 програмованих ліній вводу/виводу з можливістю підключення внутрішніх навантажувальних резисторів;

— потужні вихідні буфери, здатні керувати світлодіодами та іншими навантаженнями;

7) живлення і тактова частота:

— робоча напруга: від 4.5 до 5.5 В.

— частота роботи: до 16 МГц.



а) б) в)

Рисунок 2.3 – Розміщення виводів ATmega8515 для корпусів: а) PDIP; б) PLCC і MLF; в) TQFP

Структурна організація ATmega8515 включає такі основні компоненти:

1) Центральний процесор (CPU), що містить:

— лічильник команд (PC);

— арифметико-логічний пристрій (ALU);

— 32 регістри загального призначення;

— регістри спеціальних функцій (IOR — Input/Output Registers), які управляють периферією;

2) Пам’ять:

— внутрішня Flash-пам’ять для збереження програмного коду;

— SRAM для зберігання даних під час роботи;

— EEPROM для зберігання конфігураційних параметрів;

3) Периферійні модулі: таймери, АЦП, UART, SPI, PWM;

4) Генератор тактових імпульсів: підтримує внутрішній RC-генератор, зовнішній кварцовий резонатор або зовнішній генератор;

5) Порти вводу/виводу (див. рис. 2.3);

6) ATmega8515 має чотири 8-бітні двонаправлені порти:

— port A (PA0–PA7) — підтримує підключення внутрішніх навантажувальних резисторів, може використовуватися як мультиплексна шина адреси/даних для зовнішньої пам’яті;

— port B (PB0–PB7) — з вбудованими резисторами підтягування, підтримує апаратну підтримку SPI;.

— port C (PC0–PC7) — має аналогові входи для АЦП, використовується також для зовнішніх переривань;

— port D (PD0–PD7) — призначений для UART та інтерфейсів зв’язку.

7) Для задач обробки аудіосигналів мікроконтролер ATmega8515 має низку переваг:

— 10-бітний АЦП з мультиплексованими входами дозволяє підключати кілька мікрофонів або інших джерел аналогового сигналу;

— таймери і ШІМ-модулі можуть застосовуватись для формування тактових імпульсів та керування аналоговими пристроями;

— інтерфейси SPI/UART забезпечують зв’язок із зовнішніми аудіокодеками або цифровими процесорами;

— можливість реалізації алгоритмів цифрової обробки сигналів завдяки швидкодії (до 16 МГц) і оптимізованому набору інструкцій.

2.3 Принципи оцифрування та збереження аудіоданих

Перед збереженням аналогового сигналу мовлення у Flash-пам’яті, він повинен бути оцифрований. Для цього необхідно застосувати АЦП і оцифровувати сигнал за методом імпульсно-кодової модуляції, суть якої полягає в наступному: нехай є вихідний аналоговий сигнал, який зображено на рисунку 2.4.

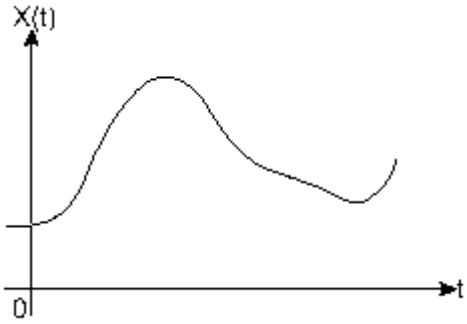


Рисунок 2.4 — Вхідний аналоговий сигнал

Вхідний аналоговий сигнал перетворюється в сигнал розділений по часу, за допомогою періодичної вибірки (рисунок 2.5).

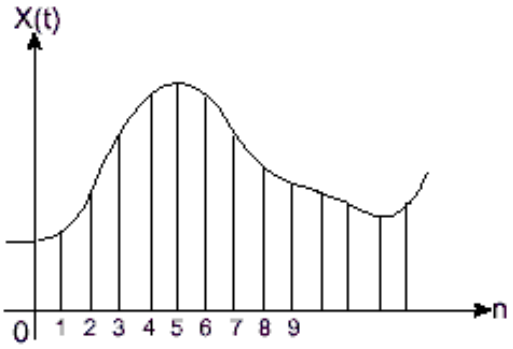


Рисунок 2.5 — Часова дискретизація сигналу

Часовий інтервал між двома вибірками називається «періодом вибірки», а його зворотній величина називається «частотою дискретизації». Згідно теоремі дискретного перетворення (теорема Котельникова), частота дискретизації повинна бути, принаймні, удвічі більшою частоти сигналу. В іншому випадку періодичне продовження сигналу в частотній області приведе до спектрального перекриття, так званому «накладення спектрів». Такий сигнал з накладенням спектру не може бути однозначно відновлений з вибірки.

Основну інформацію мовного сигналу несуть частоти до 4000 Гц. Тому для обмеження смуги частот сигналу може бути використаний фільтр нижніх частот.

Для ідеального фільтра нижніх частот з частотою 4000 Гц, частота вибірки повинна бути 8000 Гц. Залежно від фільтра змінюється його крутизна. Особливо

важливо вибрати найбільшу частоту дискретизації для фільтра першого порядку, такого як RC-фільтр, який використовується в нашому випадку. Верхня межа обмежується можливостями АЦП.

«Квантуванням» називається визначення цифрових значень, що відповідають аналоговим вибіркам, взятим на частоті дискретизації. Аналоговий сигнал квантується шляхом приписування аналоговій величині найближчого «допустимого» цифрового значення (рисунок 2.6).

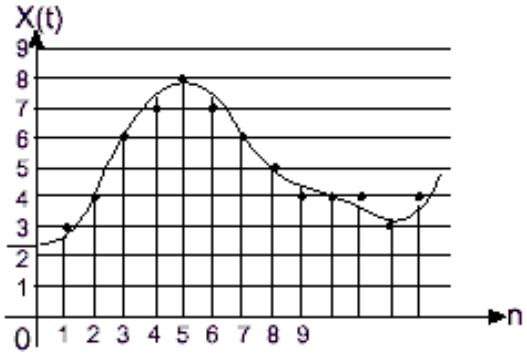


Рисунок 2.6 — Квантування сигналу

Кількість цифрових значень називається «розрядність» і завжди обмежується. Наприклад, 256 значень для 8-бітного цифрового сигналу або 10 значень у прикладі. Тому квантування аналогових сигналів завжди призводить до втрати інформації. Ця «помилка квантування» обернено пропорційна розрядності цифрового сигналу. Вона також обернено пропорційна «динамічному діапазону» сигналу, тобто інтервалу між мінімальним і максимальним значеннями (3 і 8 відповідно). АЦП мікроконтролера ATmega8515 може бути налаштований на динамічний діапазон сигналу, якщо на AGND і AREF подати відповідно мінімальне і максимальне значення сигналу.

З іншого боку, підсилювач мікрофона може бути налаштований так, що він буде перекривати динамічний діапазон АЦП.

Обидва методи знижують помилку квантування. Проте останній метод збільшує відношення сигнал / шум (SNR) і його не доцільно використовувати.

На рисунку 2.7 показані цифрові значення, які відповідають аналоговому сигналу. Ці значення отримано з виходу АЦП.

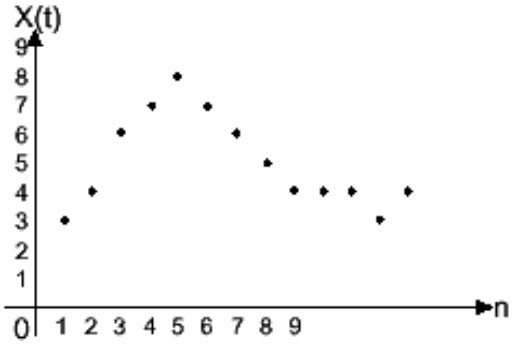


Рисунок 2.7 — Цифровий сигнал

Сигнал має мінімальне та максимальне значення, які ніколи не перевищують допустимої межі. Частини сигналу нижче мінімального і вище максимального значень не містять інформації. Вони повинні бути видалені, щоб не «заповнювати» пам'ять. Це робиться шляхом зсуву вниз всього сигналу і відкиданням частин, які перевищують максимальне значення «max» (рисунок 2.8).

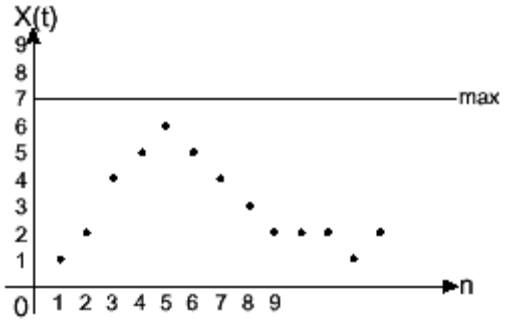


Рисунок 2.8 — Скорочений цифровий сигнал

Остаточний сигналу складається з 8 біт. Тепер він може бути збережений у Flash-пам’ять.

Flash-пам’ять не вимагає окремого циклу стирання перед програмуванням.

При використанні команд «Буфер в основну сторінку пам'яті програми з вбудованим стиранням» і «Основна сторінка пам'яті програми крізь буфер», Flash-пам’ять буде автоматично стирати певну сторінку в масиві пам'яті перед

програмуванням дійсних даних. Якщо система вимагає велику програмну пропускну здатність (більше 200Kbps), то області масиву основної пам'яті можуть бути попередньо очищені, для зменшення сумарного програмного часу. Додаткова команда «Очищення сторінки» призначена для стирання окремої сторінки пам'яті, у той час як команда «Очищення блоку» дозволяє очистити одночасно 8 сторінок пам’яті. При попередньому очищенню частини головного масиву пам’яті, для зменшення загального часу, може використовуватися команда «Буфер в основну сторінку пам'яті програми без вбудованого стирання».

Перший метод більш ефективний щодо запису програмного коду, так як не застосовуються додаткові цикли стирання. Після очищення пам'яті дані можуть записуватися до тих пір, поки не заповняться всі сторінки.

Для запису до Flash-пам’яті використовується буфер. Коли цей буфер заповниться (528 вибірками), він записується в пам'ять під час 529 перетворення. Дані записуються до тих пір, поки натиснута кнопка «Запис» або пам'ять не заповнилася. Якщо вся пам'ять заповнена, то нові дані не можуть бути записані, поки не очищена Flash-пам’ять. Якщо пам'ять заповнена лише частково, то при повторному натисканні кнопки «Запис» нові дані будуть додані відразу за вже записаними даними.

Відтворення звуку завжди починається з початку Flash-пам’яті. Воно припиняється, якщо всі записані дані відтворені або коли кнопка «Відтворення» відпущена.

Flash-пам’ять дозволяє програвати дані або безпосередньо з основної сторінки пам'яті, або шляхом копіювання сторінки в один з двох буферів і подальшим читанням з буфера. Метод прямого доступу не підходить для цього прикладу, так як це метод двохадресний (одна адреса для сторінки, інший для позиції байтів), і, отже, в Flash-пам’ять повинна бути відправлена довга послідовність для кожного окремого байта. Це займає більше одного циклу ШІМ, який триває 510 тактових імпульсів для 8-бітного ШІМ сигналу.

Тому, одна сторінка пам'яті копіюється в один з двох буферів. Поки дані читаються з цього буферу, наступна сторінка пам'яті копіюється в інший буфер. Коли всі дані зчитані з першого буфера, відбувається читання з іншого буфера, в цей час перший буфер перезавантажується. Читання даних з буфера Flash-пам’яті синхронізується частотою ШІМ.

Цифрове значення відтворюється з використанням ШІМ. На рисунку 2.9, показані вибірки сигналу.

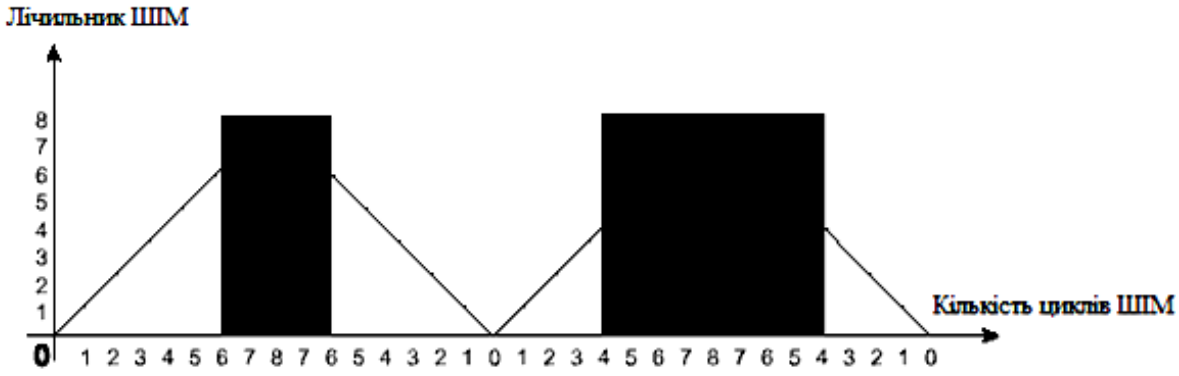


Рисунок 2.9 — Приклади ШІМ циклів

Один цикл ШІМ сигналу складається з двох етапів: перший - лічильник рахує до максимального значення, яке може бути представлене розрядністю (8 у цьому прикладі), і другий - лічильник дораховується до нуля. Вивід дозволяється, коли значення ШІМ лічильника збігається зі значенням цифрового сигналу і забороняється, коли значення ШІМ лічильника стає менше цього значення. На рисунку 2.9 темна область являє собою потужність сигналу.

Частота ШІМ повинна бути, принаймні, удвічі більша, ніж частота сигналу. Рекомендується, щоб частота ШІМ була, щонайменше, в чотири рази більша(залежно від вихідного фільтра).

Це може бути досягнуто або зниженням частоти сигналу, або збільшенням частоти тактових імпульсів, або зниженням розрядності сигналу.

Частота вихідного фільтра встановлена на рівні 4000 Гц, що складає приблизно одну чверть частоти ШІМ (15686 Гц).

Частота системного таймера і розрядність ШІМ визначають частоту ШІМ.

При частоті системного таймера 8 МГц, частота 10-бітної розрядності ШІМ дорівнює 3922 Гц (8 МГц /2*2^{10} = 3922 Гц), 7843 Гц для 9-бітної розрядності, і 15686 Гц для 8-бітної розрядності.

Тільки високе значення частоти (15686 Гц) достатньо, щоб служити в якості несучої частоти для 4000 Гц сигналу. Тому, початкова 10-бітна цифрова вибірка перетвориться в 8-бітну.

Вихідний фільтр згладжує вихідний сигнал і видаляє високочастотну несучу ШІМ сигналу (рисунок 2.10).

Рисунок 2.10 — Фільтрований вихідний ШІМ сигнал

Результуючий вихідний сигнал схожий на той, який зображено на рисунку 2.11. Якщо відкинути похибку квантування і відсутність підсилення, то сигнал повністю схожий на вхідний аналоговий сигнал (рисунок 2.4).

Об'єм пам'яті розраховується наступним чином: так як кожну секунду записується 8000 відліків по 8 біт, що становить 8 кб, то протягом години нам необхідно буде записати 3600 х 8 кб, що складе приблизно 29 Мб. Таким чином, застосувавши пам'ять ємністю 32Мб, ми забезпечимо потрібний час запису. При використанні алгоритмів архівації, обсяг записуваної інформації при необхідності можна збільшити.

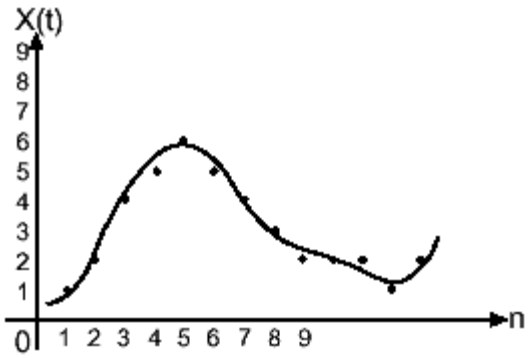


Рисунок 2.11 — Вихідний ШІМ сигнал

Flash-пам’ять безпосередньо підключається до AVR мікроконтролера через шину SPI.

2.4 Особливості програмування мікроконтролерів для аудіо запису

Програмування мікроконтролерів для задач аудіозапису має низку специфічних особливостей, які відрізняють його від традиційного кодування вбудованих систем. Обробка звукових сигналів вимагає одночасного виконання декількох завдань: захоплення аналогового сигналу, цифрової обробки, збереження даних і взаємодії з користувачем або іншими пристроями. У цьому розділі розглянемо основні виклики та особливості, які треба враховувати при розробці програмного забезпечення для мікроконтролерів, призначених для аудіозапису.

2.4.1 Реалізація безперервного захоплення сигналу

Основним завданням програмної частини є безперервне зчитування аудіосигналу з аналогово-цифрового перетворювача (АЦП). Звуковий сигнал має широку смугу частот, і для якісного запису потрібна відповідна частота дискретизації (зазвичай від 8 до 44,1 кГц і вище).

Для забезпечення безперервності запису необхідно:

— оптимізувати роботу АЦП: налаштувати мультиплексор, вибрати режим роботи АЦП, щоб мінімізувати затримки між вимірюваннями;

— використовувати апаратні переривання: це дозволяє миттєво реагувати на готовність чергового вибіркового значення АЦП, зчитувати його в буфер і звільняти МК для інших завдань;

— запроваджувати циклічні буфери (кольцеві буфери) для збереження отриманих зразків у оперативній пам’яті,це запобігає втраті даних під час запису, особливо коли доступ до зовнішньої пам’яті чи накопичувача займає додатковий час.

2.4.2 Таймінг і синхронізація

Аудіозапис потребує точного дотримання інтервалів дискретизації. Тому в програмі необхідно:

— в икористовувати апаратні таймери або внутрішні генератори переривань для формування точних часових інтервалів, які запускають запуск АЦП.

— в никати затримок у коді, які можуть призвести до пропуску вибірок.

— в разі необхідності — застосовувати механізми пріоритетів для критичних обробок, пов’язаних з аудіоданими.

2.4.3 Обробка та кодування аудіоданих

Сирі дані, отримані з АЦП, зазвичай мають великий обсяг, тому для ефективного зберігання та передачі вони можуть потребувати попередньої обробки:

— фільтрація шумів: програмне або апаратне згладжування, фільтри нижніх частот;

— компресія: застосування простих алгоритмів стиснення, наприклад, PCM з меншою розрядністю або більш складних кодеків (ADPCM, MP3 — якщо підтримується потужністю процесора);

— конвертація формату: перетворення у формат, сумісний із пристроєм відтворення або зберігання (WAV, RAW, MP3).

Важливо, щоб алгоритми обробки були адаптовані під обмежені ресурси мікроконтролера: невеликий об’єм оперативної пам’яті, обмежена тактова частота та відсутність спеціалізованих DSP-модулів.

2.4.4 Організація збереження аудіоданих

Для довготривалого запису необхідно зберігати аудіофайли у зовнішній пам’яті — флеш-карті, EEPROM, SD-карті або іншому накопичувачі. Особливості програмування у цій частині:

— використання файлових систем (FAT16/FAT32) для забезпечення сумісності з комп’ютерами;

— організація послідовного запису з перевіркою коректності;

— управління буферами, щоб уникнути втрати даних під час запису;

— обробка помилок зберігання і можливість відновлення після збоїв.

2.4.5 Взаємодія з користувачем і периферією

Для зручності користувача до системи аудіозапису часто додають:

— кнопки старт/стоп запису;

— індикацію статусу (LED, дисплей);

— інтерфейси зв’язку (UART, USB, SPI) для передачі записаних файлів.

Програмування має передбачати обробку зовнішніх подій, дебаунсінг кнопок, оновлення інтерфейсу у реальному часі та забезпечення стабільності роботи системи.

2.4.6 Оптимізація і тестування

Програмне забезпечення для аудіозапису на мікроконтролерах потрібно ретельно оптимізувати за обсягом і швидкодією:

— застосування асемблерних вставок для критичних ділянок;

— використання апаратних засобів, де це можливо (DMA, таймери);

— тестування на реальних пристроях з вимірюванням втрат вибірок, шумів та інших артефактів;

— перевірка енергоспоживання, особливо для портативних пристроїв.

Таким чином, програмування мікроконтролерів для аудіозапису — це баланс між апаратними можливостями, швидкодією і якістю звуку, який потребує комплексного підходу і ретельної оптимізації на всіх етапах розробки.

3 ПРОЕКТУВАННЯ МІКРОПРОЦЕСОРНОЇ СИСТЕМИ РЕЄСТРУВАННЯ АУДІОСИГНАЛІВ

3.1 Розробка структурної та принципової електричної схеми

Структурну схему цифрового системи наведено на рисунку 3.1 на якій можна виділити такі основні вузли: пристрій управління на базі мікропроцесора, що виконує функції управління пристроєм. В якості такого пристрою обрано мікроконтролер RISC-архітектури серії AVR фірми Atmel ATmega8515. Він володіє вбудованою пам'яттю програм обсягом 4096 слів і пам'яттю даних 512 байт. Будь-яка його команда виконується за 1 такт процесора. Тактова частота 8 МГц.

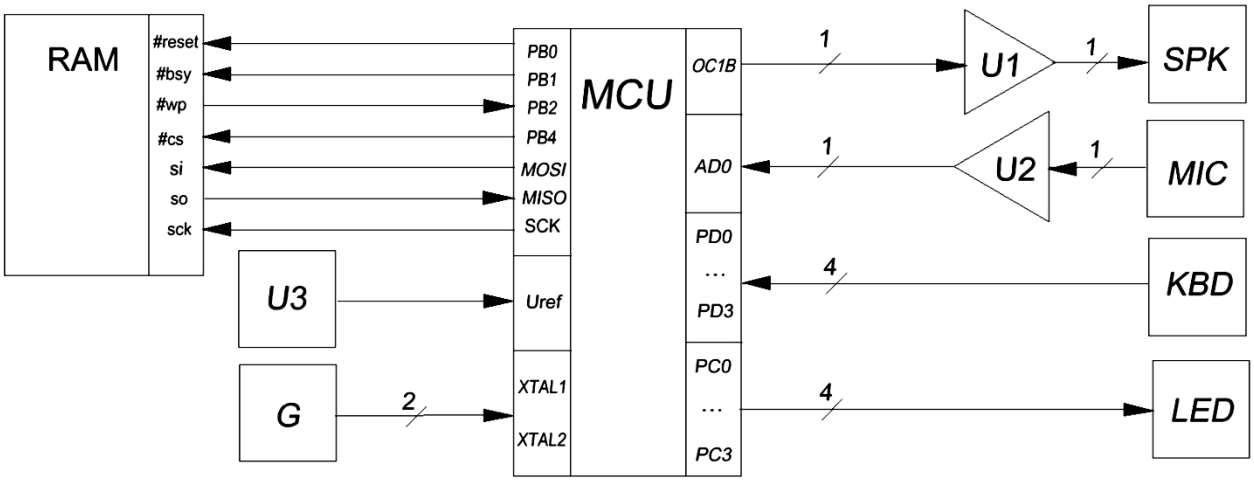


Рисунок 3.1 — Функціональна схема цифрового диктофона

На виконання процесором програмного коду для обробки і запису відліків, отриманих від АЦП, буде потрібно до 20 мс, так що обраний процесор цілком задовольняє вимогу швидкості роботи і встигає опрацювати всю необхідну інформацію.

Забезпечення протоколу роботи з пам'яттю організується тим же процесором програмно-апаратними методами, тому що в мікроконтролері є апаратна підтримка протоколу SPI.

До складу MCU також входять АЦП і ЦАП, тому мікроконтролер також виконує функції оцифрування аналогового сигналу і перетворення цифрових кодів в аналоговий сигнал. Підсилювачі U1 і U2 призначені для посилення аналогових сигналів та обмеження верхніх граничних частот цих сигналів до 4 кГц. Мікрофон і динамік призначено відповідно для введення та відтворення голосової інформації. Генератор опорної напруги U3 формує опорну напругу для вбудованого АЦП мікроконтролера MCU. Тактовий генератор G призначений для тактування всіх внутрішніх схем мікроконтролера. Енергонезалежна пам'ять RAM призначена для зберігання всієї записуваної голосової інформації. Блок клавіатури KBD призначений для управління режимами пристрою. Блок індикації LED призначений для індикації режимів роботи пристрою. Вхідний сигнал з мікрофона надходить на підсилювач-фільтр U1, де він посилюється до розмаху у кілька вольт і обмежується верхньою частотою до 4 кГц. З виходу підсилювача-фільтра U1 сигнал надходить на вхід вбудованого АЦП AD0. Оброблений сигнал програмно зберігається в RAM по лініях, передбачених протоколом SPI.

Обрано Flash-пам’ять типу AT45DB161з послідовним інтерфейсом, яка ідеально підходить для зберігання оцифрованого голосу, зображень, програмного коду та даних. Пам’ять організовано у вигляді 4096 сторінок по 528 байти кожна. На додаток до основної пам'яті AT45DB161 також має два буфера даних SRAM по 528 байти кожен. Буфери дозволяють приймати дані, у той час як сторінка в основній пам'яті перезаписується, що подібно до запису безперервного потоку даних. Емуляція EEPROM (можливість зміни біта або байта) легко обробляється трьохкроковою операцією Читання/Модифікація/Запис. На відміну від звичайної пам'яті, доступ до якої здійснюється випадковим чином за допомогою адресних ліній і паралельного інтерфейсу, Flash-пам’ять використовує послідовний інтерфейс SPI для послідовного доступу до даних. Flash-пам’ять підтримує режими SPI 0 і 3. Простий послідовний інтерфейс полегшує розміщення апаратних засобів на друкованій платі, збільшує надійність системи, мінімізує перешкоди перемикання і зменшує розмір корпусу і кількість активних виводів. Пристрій

оптимізовано для використання у багатьох комерційних та промислових пристроях, в яких істотні висока щільність монтажу, невелика кількість виводів, низьке енергоспоживання. Пристрій працює на тактових частотах до 20 МГц та типовому споживанні струму при активному читанні 4 мА. Для спрощення перезапису, AT45DB161 не потрібно високих вхідних напруг для запису. Пристрій працює від одного джерела живлення для операцій запису та читання. AT45DB161 вмикається за допомогою виводу вибору мікросхеми (CS), а доступ до неї здійснюється через трипровідний інтерфейс, що складається з ліній послідовного входу (Serial Input -SI), послідовного виходу (Serial Output - SO) та послідовної синхронізації (Serial Clock - SCK). Всі цикли запису самотактуючі і перед записом не потрібен окремий цикл стирання. Узагальнену структурну схему Flash-пам’ятіAT45DB161 наведено на рисунку 3.2, а розміщення виводів показано на рисунку 3.3. Як видно з рисунку 3.2 Flash-пам’ять AT45DB161 має такі виводи:

— вибір мікросхеми (CS);

— послідовна синхронізація (SCK);

— послідовний вхід (SI);

— послідовний вихід (SO);

— вивід апаратного захисту від запису (WP);

— скидання мікросхеми (RESET);

— готовий/зайнятий RDY / BUSY.

Послідовний вхід (Serial Input - SI) — це виключно вхідний вивід, використовується для передачі даних в пристрій. Вивід SI використовується для всіх вхідних даних, включаючи коди операцій і адресні послідовності.

Послідовний вивід (Serial Output - SO) є лише вихідним і використовується для переміщення даних з пристрою.

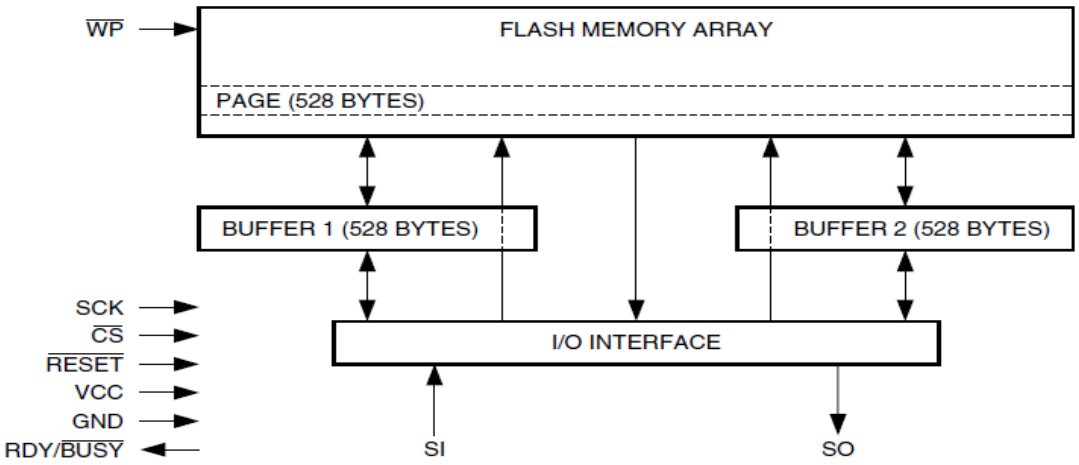


Рисунок 3.2 — Узагальнена структурна схема Flash-пам’ятіAT45DB161

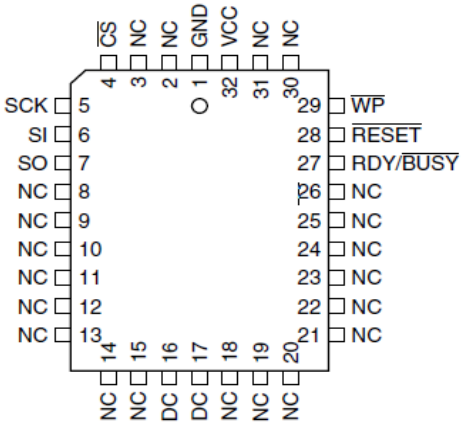


Рисунок 3.3 — Розміщення виводів Flash-пам’ятіAT45DB161

Послідовна синхронізація (Serial Clock - SCK) використовується для керування потоком даних. Дані завжди передаються в пристрій по передньому фронту SCK і видаються з пристрою по задньому фронту SCK.

Flash-пам’ять обрано, коли на виводі CS низький рівень. Коли пристрій не вибрано, дані будуть прийматися з виводу SI, а вивід SO буде залишатися у високоімпедансному стані. Перехід з високого рівня на низький на виводі CS необхідний, щоб розпочати роботу, а перехід з низького на високий, щоб закінчити роботу.

Захист від запису (Write Protect - WP) відбувається тоді, якщо на виводі WP зберігається низький рівень, перші 256 сторінок основної пам'яті перезаписати не

можна. Єдиний спосіб перезаписати перші 256 сторінок - спочатку подати на вивід захисту від запису високий рівень, а потім використовувати команди запису, наведені вище. Якщо цей вивід не використовуються, рекомендується зовні приєднати WP до шини додатного живлення.

Низький рівень на виводі скидання (RESET) перерве виконання операції і скине внутрішній кінцевий автомат в стан простою. Пристрій буде залишатися в стані скидання стільки ж, скільки на виводі RESET присутній низький рівень. Нормальна робота зможе відновитися, як тільки на вивід RESET буде повернуто високий рівень. Пристрій має внутрішню схему скидання з включенням живлення, тому немає ніяких обмежень на вивід RESET під час включення живлення. Якщо цей вивід не використовуються, рекомендується зовні приєднати RESET до шини додатного живлення.

Готовий/зайнятий (Ready / Busy - RDY / BUSY) буде встановлюватися в низький рівень, коли пристрій зайнятий виконанням внутрішньої операці. Цей вивід, зазвичай знаходиться в стані високого рівня (через зовнішній підтягаючий резистор номіналом 1 кОм), буде наближатися до низького рівня під час операцій запису, порівняння та передачі сторінки в буфер. Стан зайнятості вказує, що масив Flash-пам’яті та один з буферів недоступні, але, тим не менш, можна виконувати операції читання і запису над іншим буфером.

Коли живлення вперше подається на пристрій або коли він повертається зі стану скидання, то за умовчанням встановлюється режим SPI 3. Крім того, вивід SO буде в високоімпедансному стані, і буде потрібно перехід з високого рівня на низький на виводі CS, щоб запустити команду. Режим SPI буде автоматично вибиратися по задньому фронту сигналу CS. Після того як живлення подано, і на VCC досягнуто мінімальне значення, система повинна почекати 20 мс, перед тим як почнеться режим роботи.

У разі використання опції ISP, навантажувальний резистор на лінії Chip Select (# CS) запобігає переходу Flash-пам’яті в активний стан. Якщо опція ISP не

використовується, то цей резистор може бути відсутнім. Схему з’єднання мікроконтролера і пам’яті показано на рисунку 3.4.

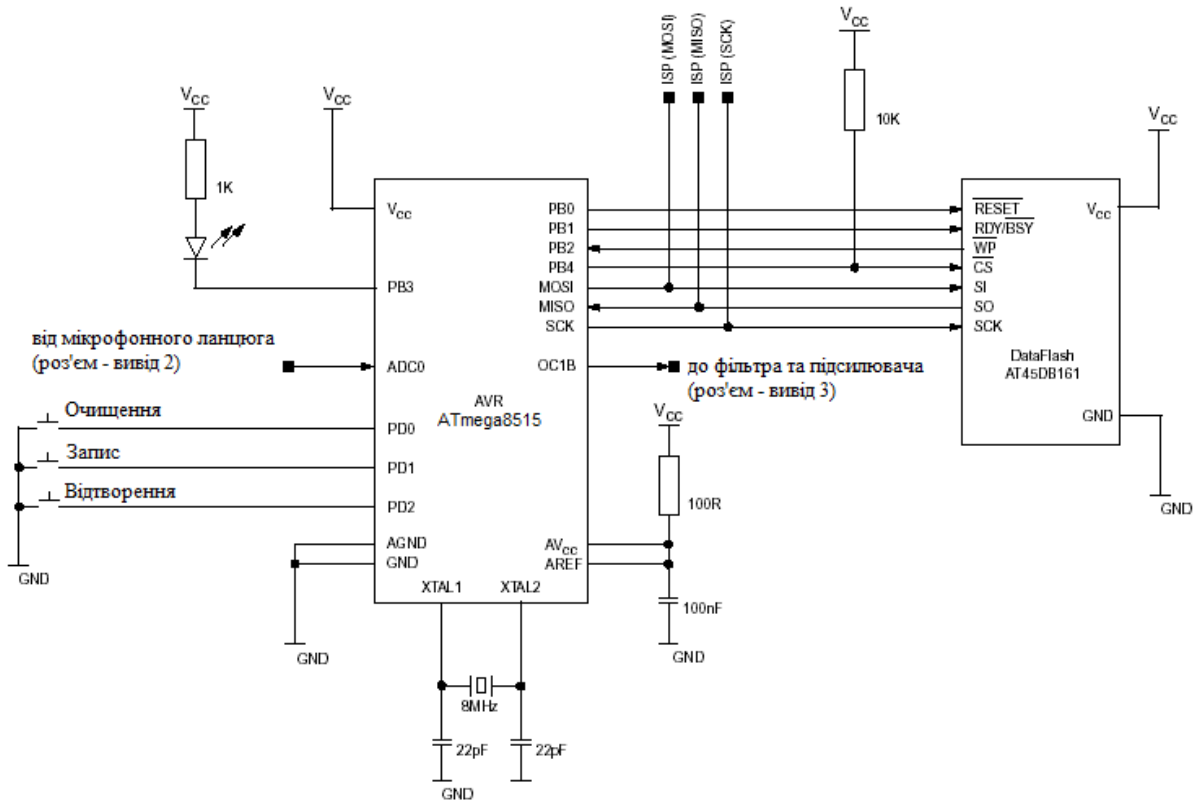


Рисунок 3.4 — Схема з’єднання мікроконтролера і пам’яті

Користувач може керувати звуковою системою через три кнопки, які називаються: «Очищення», «Запис» і «Відтворення». Якщо кнопки не натиснуті, то внутрішній навантажувальний резистор забезпечує VCC на PD0 ... PD2. Натискання кнопки переключає вхідні лінію на GND.

У якості зворотного зв'язку для користувача виступає LED, що відображає стан системи.

Аналогова напруга, AVCC, підключається до VCC через RC-фільтр нижніх частот. Опорна напруга встановлюється рівною AVCC.

Кварцовий резонатор з двома розв'язуючими конденсаторами (22 пФ) генерує системні тактові імпульси.

Аналогова напруга, AVCC, підключається до VCC через RC-фільтр нижніх частот. Опорна напруга встановлюється рівною AVCC.

Кварцовий резонатор з двома розв'язуючими конденсаторами (22 пФ) генерує системні тактові імпульси.

Схему підключення мікрофону та динаміка зображено на рисунку 3.5.

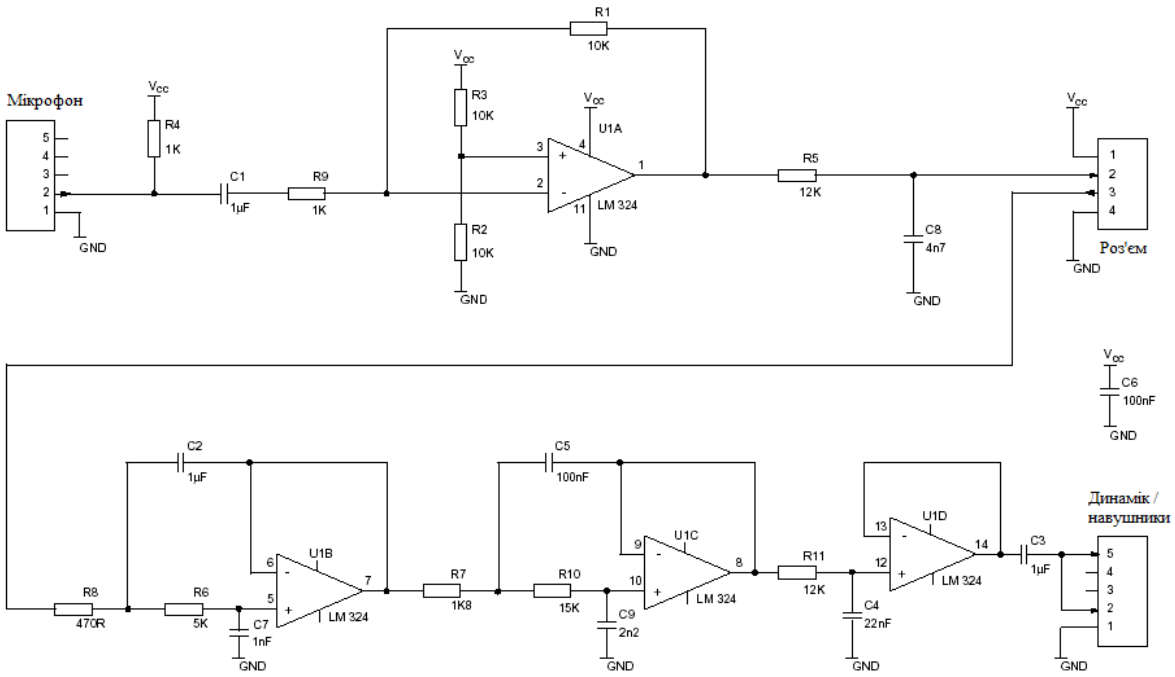


Рисунок 3.5 — Схема підключення мікрофону та динаміка

Основним елементом є мікросхема LM324, яка містить в корпусі чотири ОП та випускається в корпусах трьох типів DIP 14, SOP14, SSOP-B14. Параметри даної мікросхеми та її аналогів наведено в таблиці 3.1. Розміщення виводів показано на рисунку 3.6.

Таблиця 3.1 — Параметри мікросхеми LM324 та її аналогів

Тип ОП	N	K, дБ	Uж, В	Iж, мА	Uзм, мВ	F1, МГц	$V_{вих}$, В / мкс	$U_{ш}$, мкВ	Число виводів
LM324	4	87	3 ... 32	1,2	2 (7)	0,5	0,2	-	8
BA10324A	4	87	3 ... 32	2	2 (7)	0,25	0,2	-	14
BA14741	4	86	2 ...18	7	1 (5)	2	1	4	14

BA15218	2	87	2 ... 16	8	0,5 (5)	10	3	1	8

Мікрофонний підсилювач є простим інвертуючим підсилювачем. Коефіцієнт посилення встановлюється через R1 і R9 (коефіцієнт посилення = R1/R9). R4 призначений для живлення мікрофона, а C1 блокує будь-які постійні складові на вході підсилювача. R2 і R3 встановлюють зміщення.

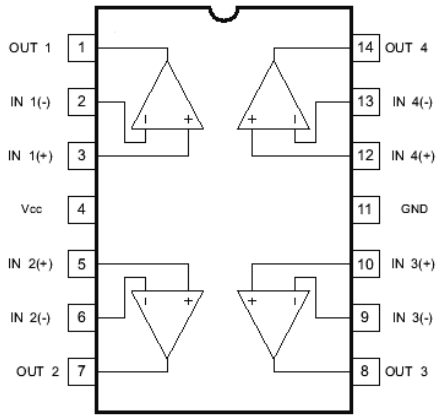


Рисунок 3.6 — Розміщення виводів мікросхеми LM324

R5 і C8 формують простий фільтр нижніх частот першого порядку. Також R5 захищає підсилювач від будь-яких ушкоджень, якщо вихід закорочено.

Ланцюг динаміка складається з фільтра нижніх частот Чебишева п'ятого порядку і підсилювача з одиничним коефіцієнтом підсилення.

Фільтр складається з двох фільтрів Чебишева другого порядку із неузгодженими контурами (R6, R7, R8, C2, C7 і R7, R10, R11, C9, C5) і пасивного фільтра першого порядку (R11, C4). Частоти цих трьох фільтрів трохи зрушені відносно одна одної («неузгоджені») для обмеження нерівномірності в смузі пропускання всього ланцюга фільтра. Сумарна частота встановлена на рівні 4000 Гц, що приблизно дорівнює одній чверті частоти ШІМ (15686 Гц).

Підсилювач з одиничним коефіцієнтом посилення захищає ланцюг від виникнення зворотного зв'язку з виходу.

С3 блокує будь-яку DC складову на вході динаміка.

3.2 Побудова алгоритму функціонування системи

Для того, щоб написати програму, яка буде керувати роботою пристрою, необхідно скласти алгоритм роботи.

Алгоритм — це скінченна послідовність команд, які потрібно виконати над вхідними даними для отримання результату. Розглянемо такі властивості алгоритмів:

— визначеність — алгоритм визначений, якщо він складається з допустимих команд виконавця, які можна виконати для деяких вхідних даних;

— скінченність — алгоритм повинен бути скінченним, послідовність команд, які потрібно виконати, має бути скінченною. Кожна команда починає виконуватись після закінчення виконання попередньої. Цю властивість ще називають дискретністю алгоритму;

— результативність — алгоритм результативний, якщо дає результати, які можуть виявитися і неправильними,приклад не результативного алгоритму — послідовність дій для виконання деяких обчислень, в якій пропущена команда виведення результатів на екран;

— правильність— алгоритм правильний, якщо його виконання забезпечує досягнення мети:помінявши місцями в алгоритмі перехід будь-які дві команди, отримаємо неправильну послідовність дій;

— формальність — алгоритм формальний, якщо його можуть виконати не один, а декілька виконавців з однаковими результатами,ця властивість означає, що коли алгоритм А застосовують до двох однакових наборів вхідних даних, то й результати мають бути однакові.

— масовість — алгоритм масовий, якщо він придатний для розв’язання не однієї задачі, а задач певного класу.

Графічні схеми алгоритмів використовують для наочного зображення алгоритмів. Є два різновиди графічних схем: а) блок-схеми; б) структурні схеми. Спочатку прийнято алгоритм формулювати словесно (письмово або в уяві), пізніше – рисувати графічну схему алгоритму, а згодом на підставі схеми, записувати відповідну програму.

Блок-схема складається з блоків декількох видів: овальних блоків «початок» і «кінець», блоків «введення і виведення даних» у вигляді паралелограмів, прямокутних блоків «процес, присвоєння» та ін. призначення блоків випливає з їхніх назв. У блоці «процес» описують одну чи декілька команд присвоєння. Формули записують довільно (символ множення тут можна не зазначати). Блоки з'єднують лініями, які описують послідовність виконання команд. Ці лінії називають лініями потоків передавання інформації. Природні напрямки потоків: зверху — вниз і зліва — направо. Якщо напрямок потоку інший, то лінія повинна мати стрілку.

Розглянемо правила побудови структурних схем. Усі команди записують у прямокутних блоках. Порядок розміщення блоків визначає порядок виконання команд. Над прямокутниками пишуть назву алгоритму. Декілька простих команд можна записувати в одному прямокутнику. Ліній потоків немає.

У свою чергу загальний алгоритм для цифрового пристрою запису голосових повідомлень, який забезпечить коректну роботу пристрою та наявність усіх функціональних можливостей, є доволі складним, тому доцільно розбити його на низку простих алгоритмів, що описують принцип роботи окремих підпрограм та функцій. Блок-схеми алгоритмів наведено в додатках.

Коли програма запущена, порти повинні бути налаштовані. Це робиться в підпрограмі «setup» (встановлення).

Протокол SPI визначає один пристрій як «головний», а інші пристрої, які підключені до «головного», як «залежні». У даному прикладі, мікроконтролер ATmega8515 виступає в ролі «головного», а Flash-пам'ять AT45DB161 в ролі

«залежного». Так як ATmega8515, в даному прикладі, є лише «головним», то в цьому прикладі вивід SS може бути використаний, як вивід вводу/виводу.

SPI інтерфейс ATmega8515 визначений як альтернативна функція PortB (PB5 ... PB7). Керуючі сигнали для Flash-пам'яті є також налаштовані на PortB (PB0 ... PB2 і PB4). Вільний вивід (PB3) використовується для управління станом світлодіода (LED). Для встановлення «головного», сигнали Serial Clock(SCK), Master Out/Slave In(MOSI), Chip Select(#CS), Write Protect(#WP) и Reset(#RST) є виходами, тоді як Master In/Slave Out(MISO) і Ready/Busy(RDY/#BSY) є входами. PB3 для LED також визначається як вихід регістра напрямку даних для PortB, встановленого як 0xBD.

Станом за замовчуванням PortB є: усі виходи у високому стані, а на всіх входах — внутрішні навантажувальні резистори.

АЦП ATmega8515 підключено до PortA. Тому PortA визначений як вхід в високоімпедансним станом.

PortD служить в якості входу для кнопок і виходу ШІМ сигналу. Тут використовується функція ШІМ Timer1 на виводі PD4.

Переривання У даному прикладі, використовуються два переривання («ADC» і «Timer1 Overflow»), які дозволяються та забороняються безпосередньо в підпрограмі, коли вони потрібні.

У головному циклі, відстежується стан усіх трьох кнопок. Якщо одна з них натиснута, то індикатор загоряється і показує, що система зайнята, і викликається відповідна підпрограма.

Додатковий цикл виконується до тих пір, поки кнопка натиснута, у якості програмного захисту для функцій «Очищення» і «Відтворення».

Під час головного циклу, індикатор погашений, це означає, що система працює в холостому режимі. Блок-схему алгоритму головного циклу наведено в додатку В.1.

Flash-пам’ять може бути попередньо очищена. Коли викликається підпрограма «erase» (очищення), встановлюється прапорець, який показує, що в наступному циклі запису нові дані можуть бути збережені на початку Flash-пам’яті.

SPI повинен бути встановлений для доступу до Flash-пам’яті. Тут не використовуються переривання. Flash-пам’ять приймає або сигнал SCK, який знаходиться в низькому стані, коли #CS перемикається з високого в низький стан (SPI режим 0), або сигнал SCK, який знаходиться у високому стані, коли #CS перемикається з низького у високий стан (SPI режим 3), під час позитивної фази тактових імпульсів. У нашому випадку SPI встановлений в режим 3. Для того щоб отримати найбільшу швидкість передачі даних, вибирається найменше поділ тактової частоти, шина SPI запускається на частоті 2 МГц, якщо використовується кварцовий генератор з частотою 8 МГц.

Для виконання очищення блоку, лінія #CS переводиться в низький стан і у Flash-пам’ять, слідом за двома зарезервованими бітами (нулями), завантажується код операції 0x50. Ця послідовність передається побайтно «залежному». Після кожного байта, регістр стану SPI (SPSR) перевіряється до тих пір, поки прапорець переривань SPI не покаже, що передача завершена. Після запису всієї послідовності, відразу після переведення лінії #CS у високий стан, починається очищення блоку. Вивід Ready/Busy переводиться Flash-пам’яттю в низький стан, до тих пір, поки блок не очиститися. Потім наступний блок буде очищений тим же самим способом, що і поточний. Очищення буде продовжуватися, поки всі блоки не очистяться. Очищені зони читаються як 0xFF. Блок-схему алгоритму очищення наведено в додатку В.2.

Підпрограма запису складається з встановлення АЦП і порожнього циклу, який триває поки натиснута кнопка «Запис». У даному випадку використовується вивід ADC0, для якого необхідно, щоб регістр вибору мультиплексора АЦП (ADMUX) був встановлений у нуль. У регістрі управління та стану АЦП (ADCSR) дозволяється робота з коефіцієнтом ділення тактової частоти 32 та встановлюється режим одиночного перетворення, дозволяються переривання, а також скидаються

прапорці переривань. Аналого-цифрове перетворення починається відразу. Перше перетворення займає більше часу, ніж наступні перетворення (832 тактових імпульсу замість 448). Після цього часу, виникає переривання АЦП, що показує, що перетворення закінчено, і результат може бути прочитаний з регістра даних АЦП.

Аналоговий сигнал з ланцюга мікрофона вибирається на частоті 15686 Гц. Це та ж сама частота, що і вихідна (ШІМ) частота.

Для досягнення частоти вибірки 15686 Гц, вибірка повинна відбуватися кожні 510 циклів (15686 Гц х 510 = 8 МГц). Для отримання одного результату АЦП, потрібно кожні 510 циклів запускати АЦП в режимі одиночного перетворення з коефіцієнтом ділення частоти 32. Одиночне перетворення займає 14 циклів АЦП. Тому перетворення буде готове після 14 х 32 = 448 циклів.

Коли перетворення закінчено, виникає переривання. Процедура переривання виконує цикл для заповнення порожніх 62 циклів (510-448), перед початком нового перетворення.

Результатом 10-розрядного перетворення є величина на вході АЦП, яка з'являється через 2 циклу після початку перетворення. Ці 10 біт перекривають діапазон від AGND до AREF (в даному прикладі від 0 до 5В). Вихідний сигнал ланцюга мікрофона обмежений діапазоном 2.3В ... 3.5В. Тому з результату 10-розрядного перетворення віднімається мінімальне вхідна напруга. Це 0x1D5 для 2.3В. Частина даних у вигляді сигналу величиною вище 3.5В, прибирається шляхом видалення двох MSB. Це робиться автоматично, коли результат перетворення передається в підпрограму «запис у флеш», так як ці змінні «flash_data» визначаються типом «char» (8-біт). Останні 8-біт даних повинні бути записані у Flash-пам’ять перед наступним перериванням перетворення. Блок-схему алгоритму запису наведено в додатку В.3.

Запис даних у Flash-пам’ять проводиться шляхом запису спочатку в буфер, а, коли цей буфер буде заповнений, його вміст запишеться в одну сторінку головної пам'яті.

У підпрограмі «write_to_flash» змінна «j» відповідає номеру байта в буфері, а змінна «k» номеру сторінки, в яку буде записуватися вміст буфера. Якщо прапорець нових даних показує, що Flash-пам’ять порожня, то обидва лічильника встановлюються в нуль.

Якщо пам'ять вже містить деякі дані, то змінні показують наступне вільне місце в пам'яті, і гарантують, що нові дані додадуться до вмісту пам'яті.

Для того щоб захистити вміст цих змінних при двох викликах функцій, вони оголошуються статичними змінними.

Для запису даних у буфер, лінія #CS переводиться в низький стан і у Flash-пам’ять завантажується операційний код 0x84. Це слідує за 14 мають сенсу бітами, що не мають сенсу, і 10-бітовою адресою розміщення всередині буфера. Потім вводяться 8-біт даних.

Ця послідовність передається «залежному» побайтно. Після кожного байта перевіряється регістр стану SPI (SPSR), поки прапорець переривання SPI не покаже, що послідовна передача завершена. Після запису всієї послідовності лінія #CS переводиться у високий стан.

Якщо буфер заповнений і залишилися порожні сторінки, то буфер копіюється на наступну сторінку Flash-пам’яті. Так як пам'ять була очищена раніше, то дані можуть бути записані без додаткового стирання.

Якщо пам'ять заповнена, то цикл виконується, поки натиснута кнопка «запис». Будь-які дані, записані в той час, коли пам'ять вже заповнена, будуть втрачені. Блок-схему алгоритму запису у Flash-пам’ять наведено в додатку В.4.

Відтворення

У процедурі «відтворення», вміст Flash-пам’яті зчитується і модулюється як 8-розрядна ШІМ на частоті 15686 Гц. Для досягнення більшої швидкості, дані не читаються безпосередньо з основної пам'яті, а передаються в один з двох буферів і потім читаються з буфера. У цей час копіюється наступна сторінка пам'яті в інший буфер. Для ШІМ, 16-розрядний Таймер/Лічильник 1 використовується з виходом ШІМ на OC1B. Це описується в регістрі управління Таймера/Лічильника A і B

(TCCRA/TCCRB). Для запуску ШІМ з можливою найбільшою частотою, дільник тактової частоти ШІМ встановлюється в 1.

Коли всановлення завершено, перша сторінка копіюється в буфер 1, за допомогою переведення лінії #CS в низький стан і передачею відповідних команд у Flash-пам’ять. Передача сторінки в буфер починається, коли лінія #CS переводиться знову у високий стан. Коли стан на виводі Ready/Busy змінюється пам'яттю Flash-пам’яттю на високий, то це означає, що буфер 1 містить дійсні дані. Потім починається передача наступної сторінки в буфер 2. Так як обидва буфера незалежні один від одного, то дані можуть завжди читатися з буфера 1, поки Flash-пам’ять залишається зайнятою копіюванням даних з другої сторінки в буфер 2.

Для читання байта з буфера, у Flash-пам’ять повинна бути записана фіктивна величина. Операція запису «головного» у SPI «залежного» призводить до того, що вміст регістра даних SPI (SPDR) буде змінено. Після запису фіктивного байта у Flash-пам’ять, регістр SPDR мікроконтролера ATmega8515 містить вихідні дані з Flash-пам’яті.

Коли значення ШІМ лічильника рівне «0», Таймер 1 викликає переривання переповнювання. Це переривання використовується для синхронізації вихідних даних Flash-пам’яті з частотою ШІМ. Коли значення з буфера заноситься в мікроконтролер ATmega8515, цикл виконується до тих пір, поки Таймер 1 не викличе переривання переповнення. Потім дані записуються у вихідний регістр порівняння Таймера/Лічильника 1 B (OCR1B), автоматично закріплюючи вихід ШІМ, коли лічильник ШІМ досягне максимального значення (255 для 8-розрядної ШІМ).

Після того, як зчитується останнє значення з буфера, активний буфер перемикається.

Якщо відтворена вся пам'ять, то всі переривання відключені і Таймер/Лічильник 1 зупинений. Блок-схему алгоритму відтворення наведено в

додатку В.5. Блок-схеми алгоритмів підпрограм «Наступна сторінка в наступний буфер» та «Активний буфер в динамік» наведено в додатках В.6 і В.7 відповідно.

Сигнал з виходу мікрофона може змінюватися в залежності від типу мікрофона, що використовується. Для досягнення кращих результатів важливо вибрати такий коефіцієнт посилення мікрофонного підсилювача, який забезпечить максимальний сигнал, найбільш близький до AREF.

Дані, записані у Flash-пам'яті, повністю відповідають даним, які були зчитані з АЦП. У випадку запису протягом великого проміжку часу або запису стерео сигналу може знадобитися стиснення даних.

Прапорець стану може застосовуватися у двох випадках. Перший спосіб – використання глобальної змінної (тобто змінна «wait» використовується в підпрограмі «відтворення»). Другий спосіб – використання незадіяного біта в регістрі. У підпрограмі «стирання», використовується біт ACIS1 (регістр управління та стану аналогового компаратора (ACSR)) для відображення того, що наступним етапом має бути збереження нових даних.

Частота вибірки рівна 15686 Гц (приблизно 510 циклів) і генерується за допомогою переривання АЦП і циклу затримки. Вона може бути замінена незалежним таймером (Таймер/Лічильник 0 або Таймер/Лічильник 2), якщо він не використовується для інших цілей.

3.3 Програмне забезпечення для збору та збереження аудіосигналів

Відповідно до алгоритму роботи пристрою розроблено спеціальну програму, яка буде керувати роботою мікроконтролера. Для цього використано програму WinAVR, яка являє собою набір програмних засобів для роботи з мікроконтролерами сімейства AVR фірми ATMEL. До нього увійшли наступні компоненти: компілятор мови C avr-gcc, бібліотека компілятора avr-libs, асемблер avr-as, інтерфейс програматора avrdude, інтерфейс JTAG ICE avarice, дебаггер avr-gdb, редактор programmers notepad (далі PN). Весь цей набір зібраний в один інсталяційний пакет і призначений для встановлення на платформу Windows. Пакет

буде встановлений в папку c:\WinAvr, а на робочому столі з'являться ярлики:

— avr-libc Manual [WinAVR]– документація по C/C++ бібліотекам, що входять в пакет;

— MFile [WinAVR] – програма для створення Мейкфайлів;

— Programmers Notepad [WinAVR] – багатофункціональний редактор коду програм.

Відкривши програму PN необхідно вибрати новий C/C++ файл. Для цього необхідно створити папку проекту, наприклад D:\project. Код файлу набирається у PN і зберігається в папку проекту під ім'ям dictophone.c.

Для компіляції програми в середовищі WinAvr використовується один з найпотужніших і найгнучкіших компіляторів - GCC. І саме через свою універсальність він має дуже багато настройок, котрі, для компіляції програми можна кожний раз вводити через командний рядок, а можна один раз записати в спеціальний makefile.

Після компіляції буде створено об'єктний файл в форматі ОС Linux (ELF). Цей об'єктний файл необхідно перетворити в зрозумілий мікроконтролеру файл в форматі Intel HEX. Ця дія забезпечується за допомогою лінковщика, який, у свою чергу, може бути визначений з командної стрічки, або керуватися настройками makefile. Тож, найкращий вихід побудови виконуваного файлу програми - це побудова з використанням makefile. Створити makefile можна або вручну, або за допомогою спеціального генератора.

Makefile для мікроконтролерів AVR є специфічними, і тому для успішного налаштування необхідна змінити лише декілька параметрів. В ролі генератора makefile буде використовуватись програма Mfile, про яку було сказано вище.

В майбутньому, після генерації шаблону makefile за допомогою програми, для адаптації поточного makefile для іншого проекту, необхідно буде змінити лише декілька значень.

Для створення makefile проекту необхідно виконати наступні кроки:

— завантажити програму Mfile. Вибрати тип мікроконтролера. В меню

Output format вибрати ihex (Intel HEX);

— в меню C standard level - gnu99;

— зберегти файл в директорію проекту,файл обов'язково повиненмати ім'я makefile;

— додати до makefile запис про вихідний текст програми, що знаходиться в файлі dictophone.c.

Зробити це можна за допомогою діалогу, що викликається при виборі пункту меню C/C++ source file(s). Але ця частина програми працює не завжди коректно, і тому, якщо змін файлу не сталося (зміни в makefile підсвічуються жовтим кольором), цей параметр потрібно встановити вручну. Для цього необхідно поставити галочку на пункті меню Makefile->Enable Editing of Makefile. Для параметра TARGET встановити значення: TARGET = dictophone. Для параметра SRC встановити значення: SRC = dictophone.c. Очистити значення параметру CPPSRC.

Далі потрібно зберегти вміст makefile.Після виконаних дій в директорії проекту з'являться 2 потрібних файли: dictophone.c і makefile.

Тепер потрібно створити виконуваний файл програми (ihex), зрозумілий мікроконтролеру:

— відкрити вихідний текст програми в редакторі PN;

— вибрати пункт меню Tools-> [WinAVR] Make Clean.

Ця дія запускає спеціальну утиліту make, що згідно з параметрами повинна очистити всі зайві файли, що могли залишитися від попередньої компіляції. Під час виконання команди, в нижній частині PN, з'явиться вікно «Output», що буде інформувати про хід виконання команди. Операція вважається успішною, якщо побачимо внизу стрічку: «>Process Exit Code: 0». Якщо значення Process Exit відмінне від нуля - це означає, що сталася помилка. Зазвичай для операції «make.exe» clean, помилку потрібно шукати внеправильній конфігурації makefile. Якщо операція пройшла успішно, виконуйється побудова виконуваного файла. Для цього вибирається пункт Tools->[WinAVR] Make all. Якщо результат компіляції

успішний, тобто "Process Exit Code: 0" - це означає, що необхідний hex файл був побудований без помилок. Інакше - помилки найвірогідніше знаходяться в тексті програми. Всю необхідну інформацію про місце виникнення помилки можна дізнатися з вікна «Output».

Якщо результат операції був успішним, то в директорії проекту з'являться такі нові файли:

— dictophone.eep — містить інформацію, що буде скопійована в EEPROM;

— dictophone.elf— об'єктний файл,саме на основі інформації цього файлу буде створений виконуваний файл;

— dictophone.hex — власне виконуваний файл, програма в кодах мікроконтролера;

— dictophone.lss —Дмістить інтерпретацію відкомпільованої програми на мовіАсемблер;

— dictophone.map;

Лістинг програми на мові С для цифрового пристрою запису звукових сигналів наведено в додатку Г.

3.4 Моделювання, перевірка та тестування роботи системи

Моделювання роботи проводяться в програмі Proteus VSM 7.2. Метою моделювання є перевірка працездатності розробленої схеми та перевірка програмного забезпечення. Схему цифрового пристрою запису звукових сигналів в програмі ISIS пакету Proteus VSM 7.2 наведено на рисунку 3.7. Після складання схеми вибирається меню Source Add/Remove Source Code Files й підключається програма dictophone.c. Вибирається режим відладки програми Debug Start/Restart Debugging. Виводиться відладочна інформація CPU Stack (стек), CPU Registers (регістри загального призначення), CPU Data Memory (пам’ять даних –ОЗП), CPU Source Code (лістинг програми), CPU Program Memory (машинні коди програми). Якщо програма не має помилок, то утворюється файл dictophone.hex, який дозволяє перевірити роботу схеми в реальному режимі роботи. Викликаючи властивості

мікроконтролера (рисунок 3.8) підключаємо отриманий файл dictophone.hex.

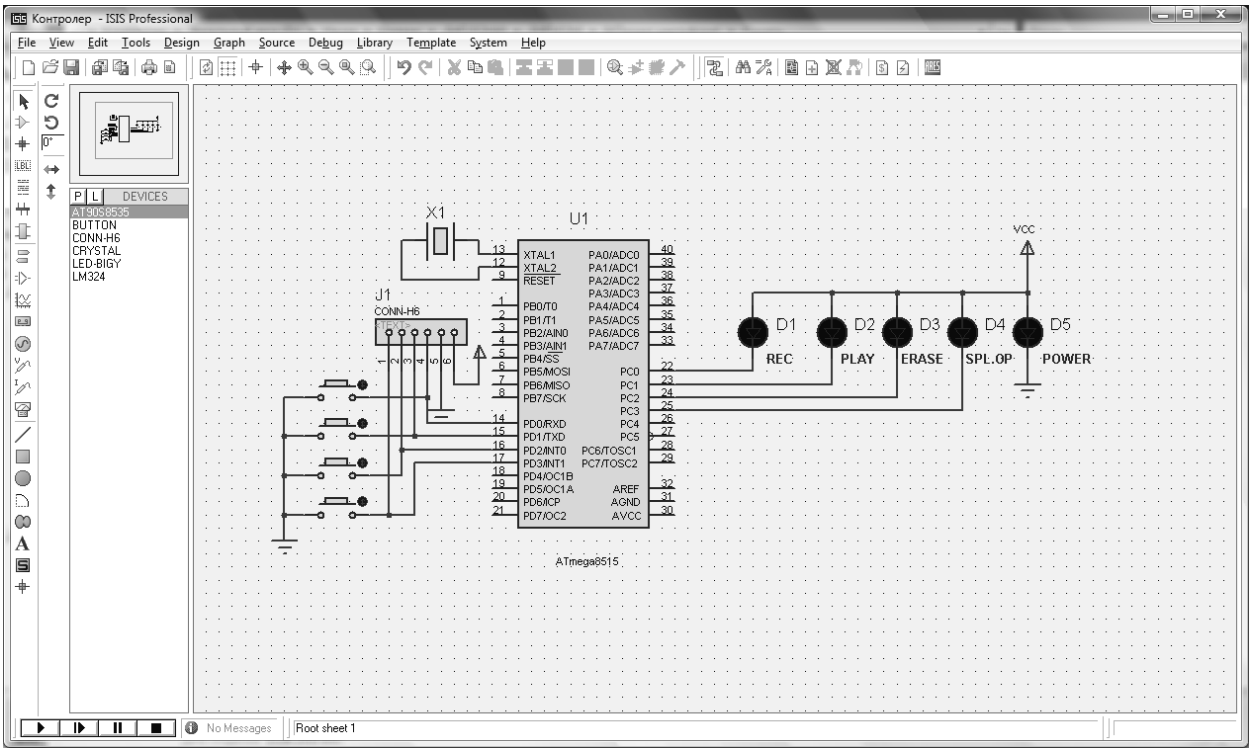


Рисунок 3.7 — Дослідження роботи цифрового пристрою запису звукових сигналів в Proteus VSM 7.2

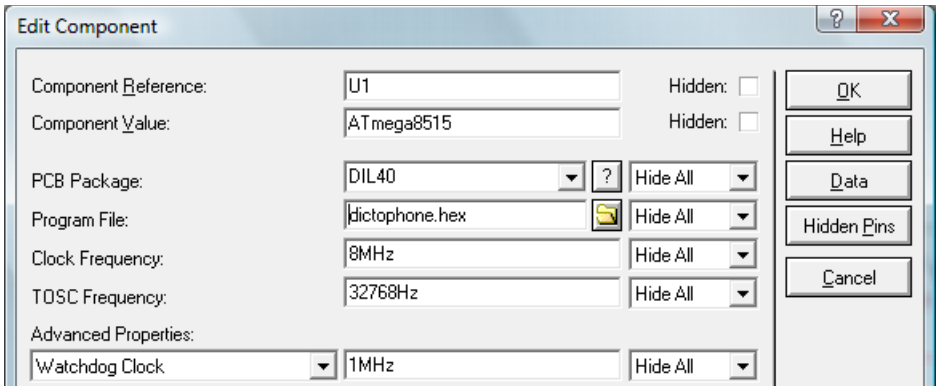


Рисунок 3.8 — Підключення програмного забезпечення dictophone.hex

Схеми підключення мікрофона та динаміка для подальшого дослідження в програмі ISIS пакету Proteus VSM 7.2 наведено на рисунку 3.9.

Основним елементом є мікросхема LM324, яка містить в корпусі чотири ОП. Для мікрофонного підсилювача взято один із ОП (U1:A), який ввімкнено за схемою інвертуючого підсилювача. Коефіцієнт посилення встановлюється через плечі

змінного резистора R3 (коефіцієнт посилення = R3.1/R3.2). R1 призначений для живлення мікрофона, а C2 блокує будь-які постійні складові на вході підсилювача.

Ланцюг динаміка складається з фільтра нижніх частот Чебишева п'ятого порядку і підсилювача з одиничним коефіцієнтом підсилення (U1:D), якийзахищає ланцюг від виникнення зворотного зв'язку з виходу.

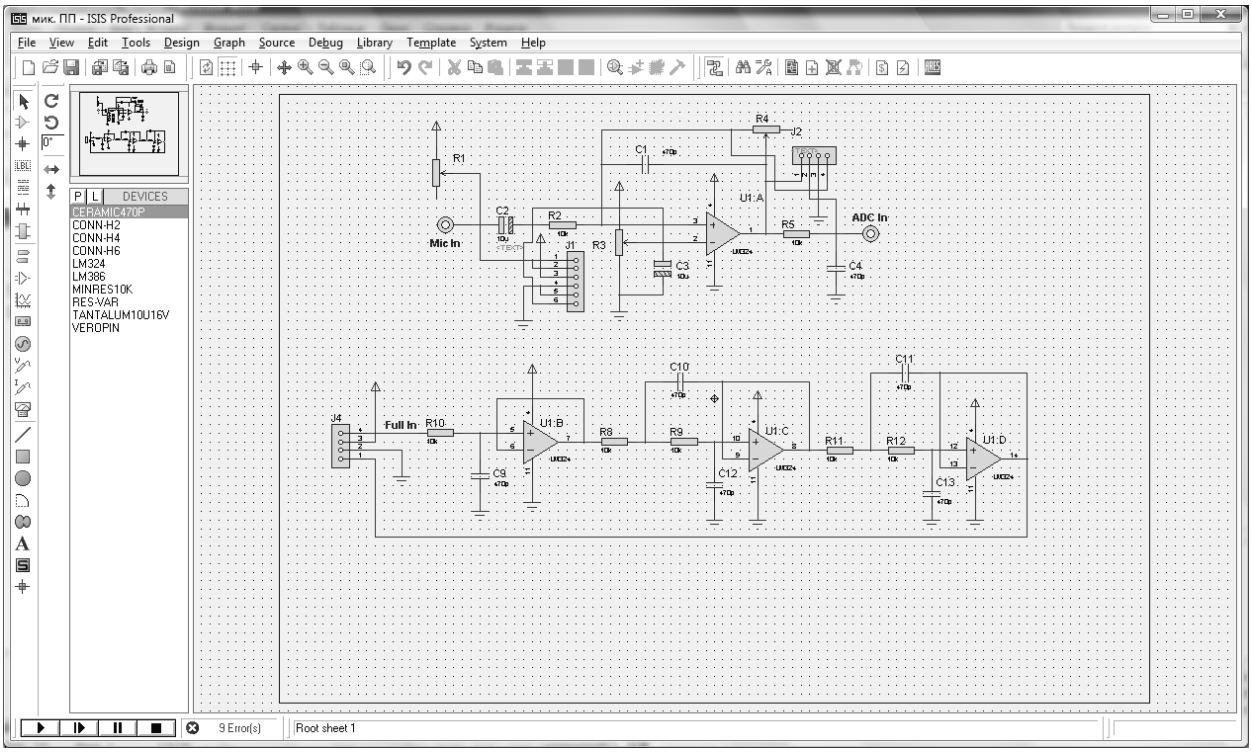


Рисунок 3.9 — Дослідження роботи схеми підключення мікрофона та динаміка

Провівши моделювання роботи цифрового пристрою запису звукових сигналів можна вважати, що схема розроблена вірно та програмне забезпечення відповідає розробленій схемі. Остаточна перевірка роботи цифрового пристрою запису звукових сигналів на мікроконтролері ATmega8515 повинна проводитись на дослідному зразку.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи була розроблена мікропроцесорна система реєстрування аудіосигналів, яка дозволяє здійснювати запис, зберігання, відтворення та очищення звукової інформації. Система підтримує три основні режими роботи — «Запис», «Очищення» та «Відтворення», а також оснащена візуальними індикаторами, які повідомляють користувача про активний режим. Збережений аудіосигнал може бути відтворений через вбудований динамік.

У процесі розробки системи було проведено аналіз існуючих рішень для цифрового запису звуку, розглянуто принципи функціонування подібних пристроїв, а також досліджено архітектуру мікроконтролера ATmega8515 і зовнішньої енергонезалежної пам’яті типу AT45DB161. Розроблено структурну схему системи, алгоритми функціонування з побудовою відповідних блок-схем, а також програмне забезпечення на мові C++ з використанням компілятора WinAVR.

Застосування середовища Proteus VSM 7.2 для моделювання дало змогу перевірити функціональність розробленої системи ще на етапі проектування, що дозволило виявити та усунути можливі недоліки. Результати моделювання підтвердили правильність обраної архітектури, стабільність роботи пристрою та відповідність технічним вимогам.

Розроблений пристрій характеризується компактністю, невисокою вартістю, простотою реалізації й достатньою швидкодією. Його конструкція базується на доступних електронних компонентах, що робить можливим самостійне виготовлення пристрою радіоаматорами чи студентами для навчальних потреб.

Загалом, створена мікропроцесорна система реєстрування аудіосигналів є ефективним засобом обробки звукової інформації й може бути використана як основа для подальших досліджень, розширення функціональних можливостей та впровадження в практичні розробки в галузі цифрової обробки сигналів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бабич В. П. Мікропроцесори та мікроконтролери: навч. посіб. / В. П. Бабич, Ю. М. Лапін. – Київ : Каравела, 2018. – 368 с.

2. Пех Н. О. Основи схемотехніки цифрових пристроїв: навч. посіб. – Львів : ЛьвДУБЖД, 2019. – 228 с.

3. AVR Microcontroller and Embedded Systems / M. A. Mazidi, S. Naimi, S. Naimi. – Pearson Education, 2014. – 832 p.

4. Іванов В. І. Основи цифрової обробки сигналів: підручник. – Харків : ХНУРЕ, 2020. – 276 с.

5. Програмування мікроконтролерів AVR мовою C / А. С. Васильєв. – Київ : ДІА, 2016. – 304 с.

6. Datasheet: ATmega8515 [Електронний ресурс]. – Режим доступу: https://www.microchip.com/en-us/product/ATmega8515

7. Datasheet: AT45DB161D [Електронний ресурс]. – Режим доступу: https://www.microchip.com/en-us/product/AT45DB161E

8. Захарченко В. Д. Вбудовані системи на базі мікроконтролерів: навч. посіб. – Одеса : ОНПУ, 2021. – 193 с.

9. Глушков А. В. Схемотехніка мікропроцесорних систем. – Київ : КНУБА, 2017. – 251 с.

10. WinAVR User Manual [Електронний ресурс]. – Режим доступу: http://winavr.sourceforge.net/

11. Proteus VSM for AVR – Labcenter Electronics [Електронний ресурс]. – Режим доступу: https://www.labcenter.com/products/vsm/overview/

12. Мікроконтролери AVR: прикладне програмування / В. О. Конончук. – Харків : ФОП Панов А. М., 2015. – 220 с.

13. Прут С. О. Проектування вбудованих систем. – Київ : Кондор, 2019. – 164с.

14. Microcontroller Projects in C for the 8051 / H. H. Barnett, S. Cox, L. O'Cull. – Thomson Delmar Learning, 2006. – 336 p.

15. Колесник С. І. Цифрова обробка сигналів у мікропроцесорних системах: навч. посіб. – Київ : НАУ, 2018. – 194 с.

16. Пат. 124783 Україна. Спосіб запису та відтворення аудіосигналу / І. М. Коваленко, О. С. Ткаченко. – Опубл. 25.06.2018, Бюл. № 12.

17. Мікропроцесорні системи: курс лекцій / В. О. Євтухов, І. А. Швець. – Херсон : ХНТУ, 2022. – 108 с.

18. Embedded Systems Design with AVR Microcontrollers / S. Sivakumar. – Springer, 2016. – 390 p.

19. Сучасні цифрові диктофони: аналіз, принципи дії, класифікація / О. А. Литвиненко // Радіоелектроніка та телекомунікації. – 2020. – №3. – С. 47–53.

20. Степанов А. О. Мікропроцесорні пристрої обробки інформації. – Дніпро : НМетАУ, 2020. – 136 с.

ДОДАТОК А

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ ВНТУ

д.т.н., проф.

_____О. Д. Азаров

“__” _____ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

«Мікропроцесорна система реєстрування аудіосигналів»

08-54.БКР.010.00.000 ТЗ

Науковий керівник к.т.н., доц. каф. ОТ
_____ Савицька Л.А.

Студент групи 1КІ-21б
_____ Киричок Л.В.

Вінниця 2025

1 Підстава виконання магістерської кваліфікаційної роботи

Наказ про затвердження теми бакалаврськоїкваліфікаційної роботи від 20.03.2025 р. №97.

2 Мета та призначення БКР

2.1 Мета роботи— розробити мікропроцесорну систему, здатну реєструвати, зберігати та відтворювати аудіосигнали за допомогою мікроконтролера AVR і зовнішньої Flash-пам’яті, використовуючи прості інтерфейси та методи обробки сигналу.

2.2 Призначення розробки — виконання бакалаврської кваліфікаційної роботи із подальшим впровадженням та розвитком продукту.

3 Вихідні дані для виконання БКР

Вихідні дані: необхідні технічні характеристики елементів необхідних для створення мікропроцесорної системи реєстрування аудіо сигналів.

4. Вимоги до виконання БКР

БКР повинна задовольняти такі вимоги:

— дослідити сучасні підходи до побудови систем аудіозапису на базі мікроконтролерів;

— здійснити вибір мікроконтролера та зовнішніх компонентів для зберігання та відтворення звуку;

— розробити електричну та функціональну схеми пристрою;

— реалізувати алгоритм обробки аудіосигналу в мікроконтролері;

— створити програмне забезпечення з підтримкою АЦП, SPI і ШІМ;

— протестувати систему та оцінити якість і стабільність реєстрації звуку.

5. Етапи БДП та очікувані результатинаведені в табл.. А1.

Таблиця А.1 — Етапи виконання роботи

№	Назва етапів виконання бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи. Вступ	21.03-23.03.25	
2	Аналіз сучасного стану та вибір напрямку дослідження	24.03-30.03.25	
3	Дослідження теоретичних основсистем реєстрації аудіосигналів	01.04-21.04.25	
4	Розробка структурної та принципової електричної схеми	22.04-05.05.25	
5	Розробка програмного забезпечення	06.05-10.05.25	
6	Оформлення пояснювальної записки та ілюстративного матеріалу	11.05-12.05.25	
7	Перевірка якості виконання бакалаврської роботи та усунення недоліків	13.05.25	

6 Матеріали, що подаються до захисту БКР: пояснювальна записка БКР, графічні і ілюстративні матеріали, протокол попереднього захисту БКР на кафедрі, відзив наукового керівника, відзив рецензента, протоколи складання державних екзаменів, анотації до БКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту БКР

Виконання етапів розрахункової та графічної документації БКРконтролюється науковим керівником згідно зі встановленими термінами. Захист БКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення БКР

8.1 При оформлювання БКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— Методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 «Комп'ютерна інженерія» (освітня програма «Комп'ютерна інженерія»). Кафедра обчислювальної техніки ВНТУ 2022;

— документами, на які посилаються у вище вказаних.

8.2 Порядок виконання БКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21»

ДОДАТОК Б
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Мікропроцесорна система реєстрування аудіосигналів
Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)
Підрозділ кафедра обчислювальної техніки
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 1,5 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату,фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішенняпро оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Азаров О.Д., зав. каф. ОТ _____ _____
(прізвище, ініціали, посада) (підпис)
Крупельницький Л.В., доц. каф ОТ _____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Здобувач _____ Киричок Л.В.
(підпис) (прізвище,ініціали)

Керівникроботи _____ Савицька Л.А., доц. каф. ОТ
(підпис) (прізвище,ініціали, посада)

ДОДАТОК В

Графічна частина

ГОЛОВНИЙ ЦИКЛ РОБОТИ СИСТЕМИРЕЄСТРУВАННЯ АУДІОСИГНАЛІВ

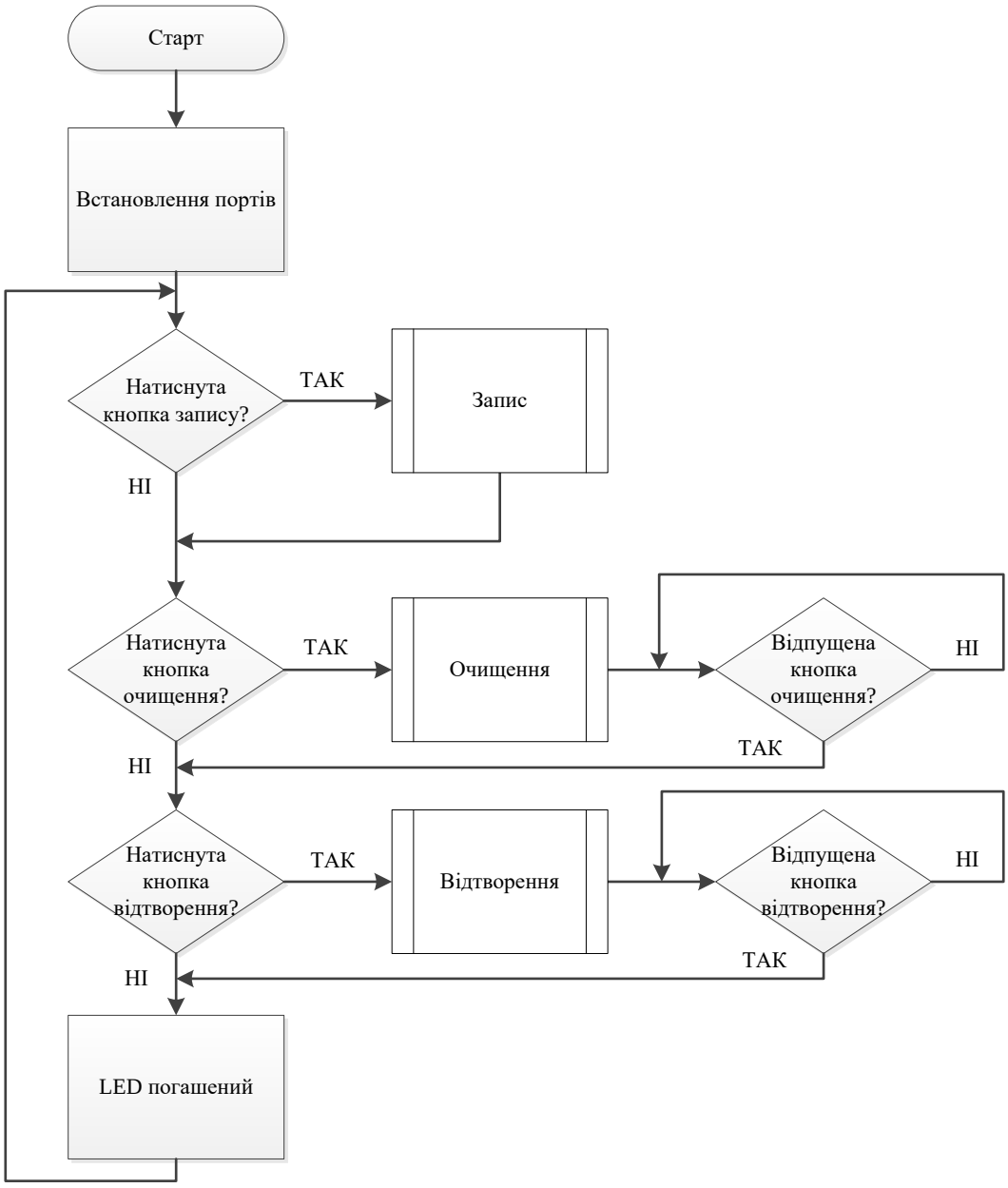


Рисунок Б.1 — Головний цикл роботи системи реєстрування аудіосигналів

ДОДАТОК Г

Ілюстративна частина

ДОСЛІДЖЕННЯ РОБОТИ СХЕМИ ПІДКЛЮЧЕННЯ МІКРОФОНА ТА ДИНАМІКА

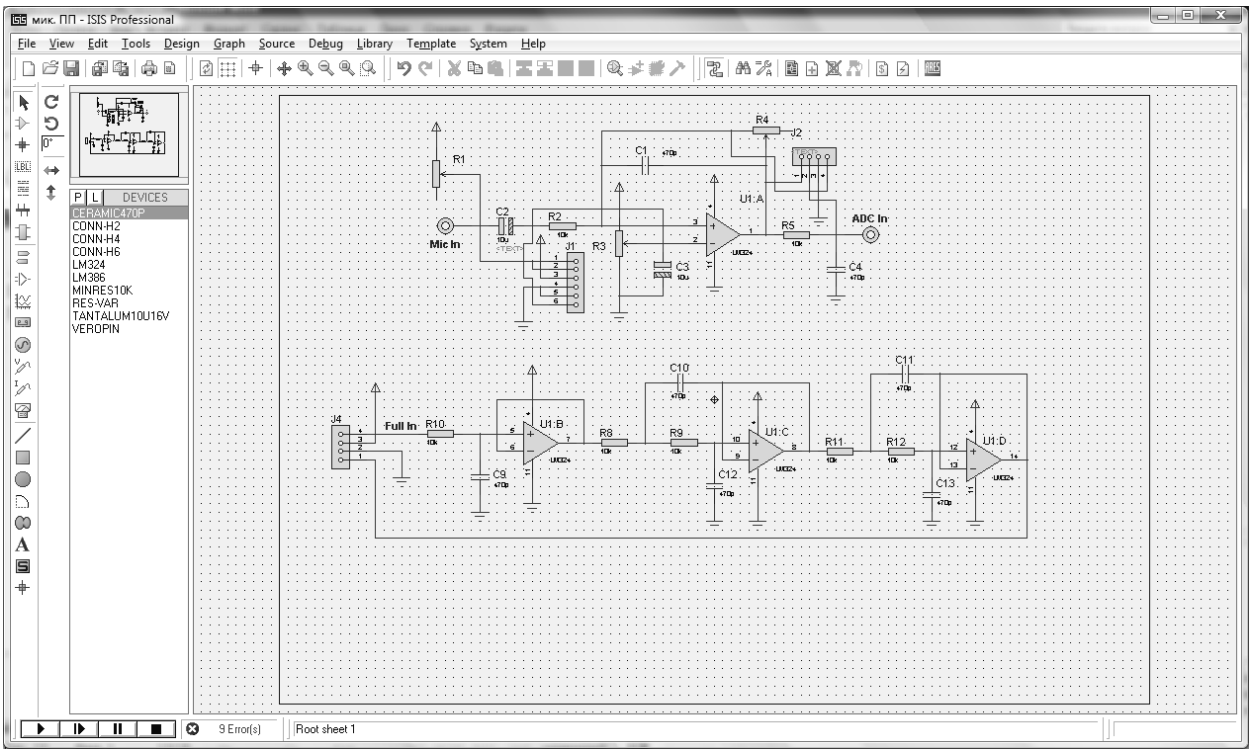


Рисунок В.1 — Дослідження роботи схеми підключення мікрофона та динаміка

ДОДАТОК Д

Лістинг основного коду

```
#include “io8535.h”
#include
#include “stdlib.h”
#include “dataflash.h”

void setup (void);
void erasing (void);
void recording (void);
void interrupt[ADC_vect] sample_ready (void);
void write_to_flash (unsigned char ad_data);
void playback (void);
void next_page_to_next_buffer (unsigned char active_buffer, unsigned int
page_counter);
void interrupt[TIMER1_OVF1_vect] out_now(void);
void active_buffer_to_speaker (unsigned char active_buffer);

// глобальні змінні
volatile unsigned char wait = 0;

voidsetup(void)
{
DDRB = 0xBD;            // Ініціалізація порта SPI
PORTB = 0xFF;   // всі виходиу високому стані, на входах
                // Навантажувальні  резистори (LED вимкнений)
DDRA = 0x00;            // PortA визначається  як  вхід
PORTA = 0x00;
DDRD = 0x10;            // PortD визначається  як вхід (D4: вихід)
_SEI();                 // преривання дозволені
}

voiderasing(void)
{
unsignedintblock_counter = 0;
unsignedchartemp = 0x80;
ACSR |= 0x02;           // встановлення прапорця, щонаступним
                // етапом повиний бути запис даних
// преривання заборонені, порт SPI ввімкнений
// «головний» режим
while (block_counter < 512)
{
```

```
PORTB &= ~DF_CHIP_SELECT;    // вмиканняFlash-пам’яті
SPDR = BLOCK_ERASE;
while (!(SPSR&temp));                // очікуваннязавершенняпередачі
SPDR = (char)(block_counter>>3);
while (!(SPSR&temp));                // очікуваннязавершенняпередачі
SPDR = (char)(block_counter<<5);
while (!(SPSR&temp));          // очікуваннязавершенняпередачі
SPDR = 0x00;
while (!(SPSR & temp));              // очікуваннязавершенняпередачі
PORTB |= DF_CHIP_SELECT;       // вимиканняFlash-пам’яті

block_counter++;
while(!(PINB & 0x02));         // очікування очисткиблока
}
SPCR = 0x00;                         //відключення SPI
}

voidrecording(void)
{
// преривання заборонені, порт SPI включений, «залежний» режим
SPCR = 0x5C;
ADMUX = 0x00;           // номер вхідного виводу АЦП = 0
ADCSR = 0xDD;          // одиничне  АЦ претворення  fCK/32, старт перетворення
while (!(PIND & 8));     // цикл продовжується поки натиснута кнопка Запис
ADCSR = 0x00;          // вимикання АЦП
SPCR = 0x00;                   // вимиканняSPI
}
voidinterrupt[ADC_vect] sample_ready(void)
{
unsignedcharcount = 0;
while (count< 6) count++;              // очікуванняпротягомкількохциклів
ADCSR |= 0x40;                 // стартновогоАЦперетворення
write_to_flash(ADC-0x1D5);          // читання даних, перетворення 8 біт і
                               //збереженняу Flash-пам’ять
}
void write_to_flash(unsigned char flash_data)
{
static unsigned int buffer_counter;
static unsigned int page_counter;
unsigned char temp = 0x80;
if((ACSR& 0x02))               //  якщопрапорецьвстановлений, тоновідані
//  повинібути встановлені
{
buffer_counter = 0;
page_counter = 0;                      // скиданнялічильникаякщоповинібути
```

```
                                    // записані   нові дані
ACSR&= 0xFD;              // очистка прапора сигнала
}
while(!(PINB& 0x02));           // перевірка занятости флеша
PORTB&= ~DF_CHIP_SELECT;      // включення DataFlash
SPDR = BUFFER_1_WRITE;
while (!(SPSR&temp));           // очікуваннязавершенняпередачі
SPDR = 0x00;
while (!(SPSR&temp));           // очікуваннязавершенняпередачі
SPDR = (char)(buffer_counter>>8);
while (!(SPSR & temp));                  // очікуваннязавершенняпередачі
SPDR = (char)buffer_counter;   // буферадреса (макс. 2^8 = 256 сторінок)
while (!(SPSR & temp));                  // очікуваннязавершенняпередачі
SPDR = flash_data;              // записданихврегістрданих SPI
while (!(SPSR & temp));                  // очікування завершення передачі

PORTB |= DF_CHIP_SELECT;        // вимиканняFlash-пам’яті
buffer_counter++;
if (buffer_counter > 528)                        // якщобуферзаповнений, тойого
                                    // вмістзаписуєтьсявсторінкупамяті
{
buffer_counter = 0;
if (page_counter< 4096)          // якщопамятьнезаповнена
{
PORTB &= ~DF_CHIP_SELECT;                        // включити DataFlash
SPDR = B1_TO_MM_PAGE_PROG_WITHOUT_ERASE;    // записати
                                    // даніз буфера 1 в сторінку
while (!(SPSR&temp));           // очікування завершення передачі
SPDR = (char)(page_counter>>6);
while (!(SPSR & temp));                  // очікуваннязавершенняпередачі
SPDR = (char)(page_counter<<2);
while (!(SPSR&temp));           // очікуваннязавершенняпередачі
SPDR = 0x00;
while (!(SPSR & temp));                  // очікуваннязавершенняпередачі
PORTB |= DF_CHIP_SELECT;        // вимикання Flash-пам’яті
page_counter++;
}
else
{
PORTB |= 0x08;   // погасити LED
while (!(PIND& 8));        //очікування нажаттякнопкиЗапис
                        //  не відпущена кнопкаЗапис
}
}
}
```

```
voidplayback(void)
{
unsigned int page_counter = 0;
unsigned int buffer_counter = 0;
unsigned char active_buffer = 1; // активнийбуфер = буфер 1
unsigned char temp = 0x80;
TCCR1A = 0x21;          // 8 бітШІМ, використовуєтьсяCOM1B
TCNT1 = 0x00;           // скиданнялічильника 1
TIFR = 0x04;                    // скидання флага превышения  счётчика 1
TIMSK = 0x04;           // разрешение прерывания переполнения счётчика 1
TCCR1B = 0x01;          // коеф. перерахункулічильника 1 = 1
OCR1B = 0x00;           // обнулення вихідного регістра порівняння
                        // переривання заборонені, порт SPIввімкнений
SPCR = 0x5C;
next_page_to_next_buffer (active_buffer, page_counter); // читаннясторінки 0
                                                        // вбуфер 1
while (!(PINB & 0x02));  // очікування завершения передачі даних із
                        //сторінки 0в буфер 1
while ((page_counter< 4095)&(!(PIND& 2)))          // поки нажата кнопка
                                                // Відтворення
{
page_counter++;                 // теперберемнаступнусторінку
next_page_to_next_buffer (active_buffer, page_counter);
active_buffer_to_speaker (active_buffer);
if (active_buffer == 1)                 // якщобуфер 1 єактивнимбуфером
{
active_buffer++;                // то встановлюємо буфер 2 в якості
                                //активного
}
else
{
active_buffer--;                // встановлюємо буфер 1 в якості активного
}
}
TIMSK = 0x00;           // забороняємо всі переривання
TCCR1B = 0x00;          // зупиняємолічильник 1
SPCR = 0x00;                    // вимикаємоSPI
}
void next_page_to_next_buffer (unsigned char active_buffer, unsigned int
page_counter)
{
unsigned char temp = 0x80;
while(!(PINB& 0x02));           // очікуємо, поканезвільниться
                                // Flash-пам’ять
```

```
PORTB&= ~DF_CHIP_SELECT;     // вимикаємоFlash-пам’ять
if (active_buffer == 1)             // якщо буфер 1 активний
{
SPDR = MM_PAGE_TO_B2_XFER;   // то передаєм наступну страницу в
                                    //буфер 2
}
else
{
SPDR = MM_PAGE_TO_B1_XFER; // передаємонаступну сторінку в буфер 1
 }
while (!(SPSR & temp));          // очікування завершення передачі
SPDR = (char)(page_counter >> 6);
while (!(SPSR & temp));                  // очікуваннязавершенняпередачі
SPDR = (char)(page_counter<< 2);
while (!(SPSR&temp));            // очікуваннязавершенняпередачі
SPDR = 0x00;                        // записуємо значения «незначущого» байт
while (!(SPSR & temp));                  // очікування завершення передачі
PORTB |= DF_CHIP_SELECT;         // вимикаємо Flash-пам’ятьі починаємо
                                    // передачу
}

void interrupt[TIMER1_OVF1_vect] out_now(void)
{
wait = 0;                       // виникнення переривання
}
void active_buffer_to_speaker (unsigned char active_buffer)
{
// поки активний буфер не очистится відтворюємойого вміст
unsigned int buffer_counter = 0;
unsigned char temp = 0x80;

PORTB &= ~DF_CHIP_SELECT;     // вмикання Flash-пам’яті
if (active_buffer == 1)             // якщобуфер 1 активнийбуфер
{
SPDR = BUFFER_1_READ;   // точитаємоізбуфера 1
}
else
{
SPDR = BUFFER_2_READ;    // читаємоізбуфера 2
}
while (!(SPSR & temp));          // очікуваннязавершенняпередачі
SPDR = 0x00;                            // запис «незначущого» значення байта
while (!(SPSR & temp));          // очікування завершення передачі
SPDR = 0x00;                            // запис «незначущого» значення байта
while (!(SPSR & temp));                  // очікування завершення передачі
```

```
SPDR = 0x00;                        // початокз адреси 0 буфера
while (!(SPSR & temp));             // очікування завершення передачі
SPDR = 0x00;                        // запис «незначущого» значення байта
while (!(SPSR & temp));         // очікування завершення передачі
while (buffer_counter < 528)
{
SPDR = 0xFF;                        // записуєм фіктивне значення в початок

while (!(SPSR & temp));         // очікування завершення передачі
while(wait);                    // очікування переривання переповнення
                                // таймера 1
OCR1B = SPDR;                   // відтворюємодані
wait = 1;
buffer_counter++;
}
PORTB |= DF_CHIP_SELECT;        // вимиканняFlash-пам'яті
}

void main(void)
{
setup();
for(;;)
{
if (!(PIND & 8))            // якщо кнопка Запис натиснута
{
PORTB&= 0xF7;               // LED
recording();
}
if (!(PIND& 4))             // якщо натиснута  кнопка Очищення ({
PORTB &= 0xF7;              // LED
erasing();
while (!(PIND& 4));
}
if (!(PIND& 2))             //якщонатиснутакнопкаВідтворення
{
PORTB &= 0xF7;                  // LED

playback();
while (!(PIND& 2));
}
PORTB |= 0x08;
}
}
```