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

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Мікропроцесорна система генерування сигналів»

08-54.БКР.003.00.000 ПЗ

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

_____Борисоглебський А.П.

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
Борисоглебському Артему Павловичу

1 Тема роботи «Мікропроцесорна система генерування сигналів», 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 генерування сигналів.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ,аналіз сучасного стану та обгрунтування напрямку дослідження, теоретичні основи побудови систем на мікроконтролерах, проектування мікропроцесорної системи генерування сигналів, висновки.

5 Перелік графічного матеріалу: Узагальнена структурна схема мікроконтролера ATmega16.

Перелік ілюстративного матеріалу: Моделювання роботи генератора6 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 побудови систем на мікроконтролерах	01.04-21.04.25	
4	Розробка електричної принципової схеми	22.04-05.05.25	
5	Програмна реалізація	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 Артем БОРИСОГЛЕБСЬКИЙ

Керівник роботи к.т.н., доц. Людмила САВИЦЬК

АНОТАЦІЯ

УДК 621.396.67

Борисоглебський А. П. Мікропроцесорна система генерування сигналів. Бакалаврська кваліфікаційна робота зі спеціальності 123 «Комп'ютерна інженерія», освітня програма «Системне програмування». Вінниця: ВНТУ, 2025. 79с.

На укр. мові. Бібліогр.: 20 назв, рис.22, табл. 3,.

У бакалаврській кваліфікаційній роботі розроблено систему цифрової генерації сигналів на основі мікроконтролера з можливістю вибору типу та частоти сигналу. В процесі дослідження розглянуто принципи побудови DDS-генераторів, проаналізовано апаратні й програмні рішення для створення універсального сигналогенератора.

У роботі реалізовано електричну схему генератора та розроблено програмне забезпечення, що забезпечує формування сигналів заданої частоти та форми. Система керування дозволяє користувачеві обирати необхідні параметри через інтерфейс, що робить пристрій зручним у використанні в різних галузях — від навчальних лабораторій до вимірювальних систем.

Проєктовану систему можна розширити шляхом додавання функцій регулювання амплітуди сигналу, що дозволить суттєво розширити сферу її практичного застосування, зокрема в радіоелектроніці, біомедичних дослідженнях та технічному навчанні.

Ключові слова: генератор сигналів, DDS, мікроконтролер, частота, сигнал довільної форми, цифрове керування, електроніка.

ABSTRACT

Borisoglebsky A. P. Microprocessor signal generation system. Bachelor's qualification work in specialty 123 "Computer Engineering", educational program "System Programming". Vinnytsia: VNTU, 2025. 79 p.

In Ukrainian. Bibliography: 20 titles, fig. 22, tab. 3.

In the bachelor's qualification work, a system of digital signal generation based on a microcontroller with the ability to select the type and frequency of the signal was developed. In the process of research, the principles of constructing DDS generators were considered, hardware and software solutions for creating a universal signal generator were analyzed.The project includes the design of an electrical schematic and the development of software that enables the generation of signals with specified frequencies and waveforms. The control system provides a user-friendly interface for selecting signal parameters, making the device suitable for use in various fields — from educational laboratories to measurement systems.

The designed system can be enhanced by implementing amplitude control features, which would significantly broaden its application potential in areas such as radio electronics, biomedical research, and technical training.

Keywords: signal generator, DDS, microcontroller, frequency, arbitrary waveform, digital control, electronics.

ЗМІСТ

ВСТУП

Актуальністьтеми роботи пов’язана з тим, що сучасна електроніка стрімко розвивається, і мікроконтролери стали невід'ємною частиною більшості цифрових пристроїв. Вони забезпечують інтелектуальне управління побутовою технікою, медичним обладнанням, системами автоматизації, вимірювальними комплексами тощо. Хоча кінцевому користувачеві часто невідомо про наявність мікроконтролера в пристрої, саме він є ключовим елементом, що забезпечує зручність, надійність та багатофункціональність сучасних технічних засобів.

Однією з важливих галузей застосування мікроконтролерів є формування та управління електронними сигналами. DDS (Direct Digital Synthesis) технології відкривають широкі можливості для створення генераторів сигналів з точно керованими параметрами. Актуальність теми полягає в необхідності розробки доступного, компактного та гнучкого пристрою, здатного генерувати сигнали різної форми та частоти для використання у навчальних лабораторіях, медичних системах, технічній діагностиці тощо.

Мета роботи — створення мікропроцесорної системи генерації сигналів на основі DDS-методу з можливістю вибору форми та частоти вихідного сигналу за допомогою інтерфейсу користувача.

Завдання дослідження:

— проаналізувати сучасні підходи до реалізації цифрових генераторів сигналів;

— здійснити вибір відповідного мікроконтролера та допоміжних компонентів;

— розробити принципову схему пристрою;

— створити алгоритм роботи системи та реалізувати його програмно;

— забезпечити відображення параметрів сигналу на індикаторі;

— протестувати роботу пристрою та оцінити точність та стабільність генерації сигналів.

Об’єкт дослідження — мікропроцесорна система цифрової генерації сигналів.

Предмет дослідження — методи реалізації DDS-генераторів на основі мікроконтролерів, принципи управління параметрами сигналів та способи їх індикації.

Практичне значення отриманих результатів полягає у створенні прототипу компактного, доступного та надійного генератора сигналів, який можна адаптувати для потреб лабораторного навчання, тестування електронних схем або застосування в спеціалізованих приладах. Завдяки використанню мікроконтролера користувач отримує можливість змінювати параметри сигналу без втручання в програмний код, що підвищує зручність і гнучкість використання пристрою.

У роботі застосовано **такіметоди дослідження**, якметоди структурного аналізу, моделювання цифрових систем, об’єктно-орієнтованого проєктування, тестування електронних схем та експериментального аналізу результатів.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ТА ОБҐРУНТУВАННЯ НАПРЯМУ ДОСЛІДЖЕННЯ

1.1 Обґрунтування актуальності

У сучасному світі розвиток електронних систем і цифрових технологій створює високі вимоги до генерації електричних сигналів із точними параметрами, широким діапазоном частот та можливістю програмного керування. Мікропроцесорні системи, як інтегровані апаратно-програмні комплекси, здатні забезпечити ефективну реалізацію функцій генерації сигналів, що робить їх одними з найперспективніших рішень у різних галузях науки і техніки.

Традиційні аналогові генератори сигналів, незважаючи на їх широке поширення, мають низку істотних обмежень, таких як нестабільність частоти через температурні коливання, низька гнучкість зміни параметрів сигналу та висока вартість складних аналогових схем. Відсутність можливості швидкої перенастройки параметрів робить їх менш привабливими для сучасних задач, що вимагають високої адаптивності та багатофункціональності.

Мікропроцесорні системи генерування сигналів базуються на використанні цифрової обробки, що дозволяє досягати високої точності, стабільності і повторюваності параметрів сигналів. Цифрові алгоритми, такі як Direct Digital Synthesis (DDS), забезпечують гнучкість у формуванні сигналів різної форми — синусоїдальних, квадратних, трикутних, пилкоподібних та інших — з високою роздільною здатністю частоти і амплітуди. Це відкриває можливість застосування таких систем у складних вимірювальних, діагностичних, радіотехнічних, медичних та навчальних пристроях.

Використання мікропроцесорів як основи системи дозволяє реалізувати програмне керування не лише параметрами сигналу, а й інтегрувати додаткові функції: управління через інтерфейси користувача, збереження профілів сигналів, комунікацію із зовнішніми пристроями та мережами, автоматичну калібрування і самодіагностику. Це значно підвищує ефективність експлуатації та розширює функціональні можливості.

Крім того, сучасні мікропроцесори мають високу обчислювальну потужність і велику кількість периферійних модулів, що дозволяє реалізовувати багатозадачні алгоритми обробки сигналів у реальному часі без значних апаратних витрат. Завдяки цьому зменшуються габарити та енергоспоживання пристроїв, що особливо важливо для портативної та вбудованої електроніки.

Актуальність розробки мікропроцесорної системи генерування сигналів підтверджується широким спектром її можливих застосувань. У сфері радіотехніки такі системи використовують для тестування радіочастотних модулів, налаштування фільтрів, аналізу характеристик передавачів і приймачів. У медицині — для генерації точних електричних стимулів, необхідних у діагностичних та терапевтичних апаратах. У промисловості — для керування електроприводами, автоматизації технологічних процесів, контролю якості виробів. В освітньому процесі — для наочного демонстрування принципів роботи цифрових і аналогових систем, лабораторних робіт і досліджень.

Технічні вимоги до сучасних мікропроцесорних генераторів включають широкий діапазон частот — від одиниць герц до десятків мегагерц, високу точність підтримки заданої частоти з мінімальними відхиленнями, низький рівень фазових шумів і гармонійних спотворень, стабільність роботи при змінах температури та живлення, а також можливість програмної конфігурації і дистанційного керування.

Розробка мікропроцесорної системи генерування сигналів є важливим кроком у модернізації електронних приладів, оскільки дозволяє отримати компактний, універсальний і функціональний пристрій з можливістю легкої адаптації під конкретні завдання без зміни апаратної частини. Це сприяє зниженню вартості виготовлення, спрощенню налагодження та підвищенню надійності експлуатації.

Таким чином, науково-технічна новизна і практична значущість розробки мікропроцесорної системи генерування сигналів є безперечною, що обґрунтовує вибір теми для дослідження та її актуальність у контексті сучасних технологічних тенденцій і вимог інженерної практики.

1.2 Постановка мети та завдань дослідження

Мета дослідження полягає у створенні високоточній мікропроцесорної системи для генерації електричних сигналів різної форми і частоти з можливістю гнучкого програмного керування, яка забезпечить високу стабільність, надійність та зручність використання у широкому спектрі практичних застосувань. Система повинна підтримувати формування синусоїдальних, прямокутних, трикутних та інших складних форм сигналів, що є критично важливим для вимірювальних приладів, лабораторних стендів, медичної апаратури та навчальних комплексів.

Досягнення поставленої мети передбачає розв’язання комплексної задачі, яка включає:

— глибокий аналіз сучасних методів генерації сигналів, включаючи апаратні (аналогові, цифрові, гібридні) та програмні (DSP-алгоритми, DDS, FPGA-реалізації) підходи, що дозволить обрати найбільш ефективну архітектуру системи для поставлених технічних вимог:

— визначення детальних технічних характеристик пристрою, таких як діапазон робочих частот (від низьких герцових до мегагерцевих значень), точність формування сигналу (включно з мінімальним рівнем спотворень і шумів), швидкість перемикання між режимами, енергоспоживання, а також вимоги до інтерфейсів управління і індикації;

— розробка оптимальної апаратної платформи, що забезпечить збалансованість між продуктивністю процесорного ядра, обсягом оперативної пам’яті, периферійними можливостями (ЦАП, АЦП, таймери, порти вводу/виводу) та енергоефективністю;

— створення програмних алгоритмів для цифрової генерації сигналів з використанням методу DDS (Direct Digital Synthesis), який дозволяє гнучко управляти частотою, фазою і амплітудою сигналу в реальному часі, забезпечуючи високу точність і стабільність вихідних параметрів;

— розробка програмного інтерфейсу управління, який надає користувачу можливість інтуїтивно налаштовувати параметри сигналу через зовнішні елементи

керування або графічний інтерфейс, а також отримувати зворотний зв’язок у вигляді цифрових індикаторів або дисплеїв;

— побудова модульної конструкції системи, що дозволить легко масштабувати рішення для розширення функціоналу, інтегрувати додаткові канали генерації, розширити спектр підтримуваних типів сигналів, а також реалізувати механізми самодіагностики та захисту.

— проведення комплексного тестування і калібрування розробленої системи з метою підтвердження відповідності технічним вимогам, оцінки точності, стабільності та надійності в різних режимах експлуатації.

— оцінка потенційних сфер застосування розробленої системи, включно з вимірювальною технікою, науково-дослідними лабораторіями, медичними приладами, телекомунікаційним обладнанням, а також у навчальному процесі для підготовки інженерів та спеціалістів.

Реалізація цих завдань дозволить не тільки створити пристрій з високими технічними показниками, а й забезпечить формування фундаментальних знань у галузі цифрової обробки сигналів і мікропроцесорної техніки, що має важливе значення для подальшого розвитку електронної інженерії.

Особлива увага буде приділена оптимізації алгоритмів генерації для зменшення затримок і забезпечення безперервності сигналу, що критично у випадках високочастотних застосувань. Також важливо розробити зручний і зрозумілий інтерфейс, який забезпечить інтуїтивне управління пристроєм навіть для користувачів із базовими технічними знаннями.

В підсумку, мета роботи спрямована на створення мікропроцесорної системи, яка стане ефективним інструментом для інженерів, дослідників і освітян, сприятиме автоматизації лабораторних експериментів, підвищенню якості діагностичних і тестових процедур, а також розвитку інноваційних рішень у сфері генерації електричних сигналів.

1.3 Обґрунтування економічної доцільності

Розробка мікропроцесорної системи генерування сигналів є надзвичайно актуальною і доцільною з огляду на сучасний стан науки і техніки, а також потреби різних галузей, де необхідне точне формування сигналів з заданими параметрами.

Використання мікропроцесорних систем у генерації сигналів відкриває широкі можливості для підвищення гнучкості, точності і стабільності роботи пристроїв. Традиційні аналогові генератори мають обмежені можливості в плані програмної адаптації, складності зміни параметрів сигналу і вразливості до зносу компонентів та зовнішніх факторів.

Впровадження цифрових методів, зокрема Direct Digital Synthesis (DDS), дає змогу:

— швидко і точно змінювати частоту, амплітуду та форму сигналу без необхідності заміни апаратних частин;

— забезпечувати високу стабільність частоти, що критично для вимірювальної техніки та телекомунікацій;

— формувати довільні форми сигналів, які неможливо або важко отримати аналоговими методами;

— легко інтегрувати систему в інші цифрові рішення, зокрема у мережі, системи збору даних та автоматизації.

Застосування мікропроцесорної системи дозволяє значно знизити собівартість виготовлення багатофункціонального генератора сигналів. Використання універсальних мікроконтролерів і цифрових компонентів дозволяє замінити дорогі і складні аналогові схеми одним компактним пристроєм, що зменшує витрати на виробництво, технічне обслуговування і ремонт.

Окрім цього, розробка програмного забезпечення дає змогу:

— легко оновлювати функціонал пристрою без фізичних змін;

— використовувати один і той самий апаратний модуль для різних задач, змінюючи лише програмне забезпечення;

— скоротити час розробки нових модифікацій завдяки модульній архітектурі системи.

Використання мікропроцесорної системи генерації сигналів є важливим у багатьох практичних сферах:

— вимірювальна техніка: забезпечення надійних тестових сигналів для калібрування і налагодження приладів;

— освіта і наука: проведення лабораторних експериментів і наукових досліджень з можливістю швидкої зміни параметрів сигналу;

— медицина: формування терапевтичних і діагностичних сигналів з високою точністю;

— промисловість: контроль і тестування електронних пристроїв, систем автоматизації, а також розробка нових технологічних рішень.

Розроблена система, завдяки використанню сучасних методів цифрової обробки сигналів та гнучкій архітектурі, має конкурентні переваги порівняно з існуючими аналоговими та цифровими генераторами:

— можливість швидкого масштабування та інтеграції з іншими системами;

— підвищена точність і стабільність формування сигналів;

— зручний інтерфейс користувача та простота налаштування;

— зниження вартості за рахунок оптимізації апаратної платформи і програмних рішень.

Таким чином, розробка мікропроцесорної системи генерування сигналів є технічно обґрунтованою, економічно вигідною і має високий практичний потенціал. Вона відповідає сучасним вимогам до точності, надійності та універсальності електронних систем, що робить її доцільною для впровадження у різноманітних галузях промисловості, науки та освіти.

2 ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ СИСТЕМ НА МІКРОКОНТРОЛЕРАХ

2.1 Аналіз функціональних вимог до системи

Синтезатори частоти є ключовими компонентами в широкому спектрі сучасних електронних систем, зокрема в телекомунікаціях, радіолокації, системах радіозв’язку, радіомоніторингу, а також у високоточних вимірювальних пристроях. Серед усіх типів синтезаторів частоти особливе місце посідають синтезатори прямого цифрового синтезу (Direct Digital Synthesis, DDS) завдяки їх високій продуктивності, гнучкості та низькій вартості виробництва.

DDS синтезатори мають низку значних переваг у порівнянні із синтезаторами на основі системи фазового автопідстроювання частоти (Phase Locked Loop, PLL). По-перше, в DDS практично відсутні перехідні процеси, що забезпечує миттєву та плавну зміну вихідної частоти без затримок і спотворень. Це дозволяє реалізувати швидку перебудову частоти, що є критично важливим у системах зв’язку з динамічним частотним плануванням.

По-друге, архітектурна побудова DDS базується на цифрових компонентах, що спрощує їх інтеграцію у вигляді мікросхем великої інтеграції (VLSI), забезпечуючи компактність, надійність і можливість масового виробництва за низькою собівартістю. Окрім того, цифрове керування DDS дозволяє легко впроваджувати складні алгоритми керування та модулювання вихідного сигналу.

По-третє, DDS синтезатори є дешевшими за класичні аналоги з PLL при одночасній можливості роботи на високих частотах та підтримці широкого діапазону вихідних сигналів. Це робить DDS перспективним вибором для використання в сучасних радіоелектронних системах, де вимоги до швидкодії, стабільності та гнучкості постійно зростають.

DDS знайшли широке застосування в системах, що вимагають високої точності і стабільності сигналу, таких як цифрові радіостанції, радіолокаційні комплекси, системи демодуляції ЧМ і ФМ-сигналів (наприклад, система Костаса),

а також в системах управління діаграмою спрямованості антенних решіток, де необхідна точна і швидка зміна робочої частоти.

Класифікація методів синтезу частоти

Синтезатори частоти можна класифікувати за типом реалізованого методу синтезу, їх класифікація описана нижче.

Прямий аналоговий синтез (Direct Analog Synthesis, DAS) — цей метод базується на безпосередній обробці опорного сигналу за допомогою змішувачів, фільтрів та дільників частоти. Вихідна частота формується шляхом множення, змішування та фільтрації опорного сигналу, що забезпечує низький рівень фазового шуму та високу стабільність сигналу.

Непрямий синтез на основі системи фазового автопідстроювання частоти (Phase Locked Loop, PLL) — реалізується через зворотний зв’язок, де вихідна частота формується генератором, керованим напругою (VCO), який синхронізується з опорною частотою за допомогою фазового детектора. Цей метод характеризується хорошою стабільністю, але має обмежену швидкість перебудови і потенційно вищий рівень фазових шумів.

Прямий цифровий синтез (Direct Digital Synthesis, DDS) — генерує вихідний сигнал у цифровій формі, використовуючи числові методи формування фази та амплітуди, що забезпечує високу точність та швидку перебудову частоти без фазових розривів.

Гібридний синтез — комбінація декількох методів, що дозволяє поєднати переваги кожного з них і мінімізувати їх недоліки.

При проектуванні синтезатора частоти важливо враховувати низку характеристик, які визначають його придатність для конкретного застосування:

— чистота спектру вихідного сигналу — включає рівень побічних частотних компонентів і фазових шумів,висока чистота сигналу є критичною для зменшення перешкод у системі;

— діапазон перебудови частоти — ширина смуги частот, у межах якої синтезатор може працювати без втрати характеристик;

— швидкість перебудови — час переходу від однієї частоти до іншої, що має бути мінімальним у динамічних системах;

— частотна роздільна здатність — мінімальний крок зміни частоти, що визначає точність налаштуван;.

— кількість одночасно генерованих частот — важливо для мультиканальних систем;

— гнучкість функціонування — здатність синтезатора підтримувати різні типи модуляції (амплітудна, частотна, фазова);

— нерозривність фази — забезпечує плавність сигналу при зміні частоти, що особливо важливо для зв'язкових систем із фазовою модуляцією.

Прямий аналоговий синтез (DAS) формується за схемою змішувач/фільтр/дільник (рисунок 2.1).

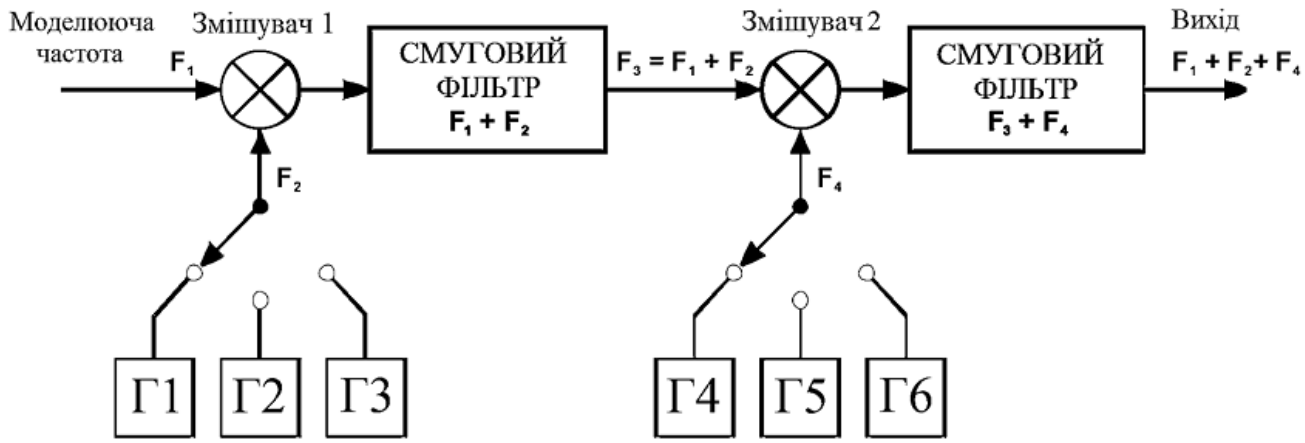


Рисунок 2.1 — Прямий аналоговий синтезатор

Його ключова особливість полягає у відсутності системи зворотного зв'язку для корекції помилки, що забезпечує мінімальний фазовий шум і високу швидкість переходу між частотами.

Однак, для забезпечення широкого діапазону вихідних частот необхідно мати велику кількість опорних генераторів, що ускладнює та подорожує виготовлення. Застосування дільників частоти дозволяє дещо оптимізувати цю проблему, але не усуває її повністю.

Додатково, DAS синтезатори мають унікальну здатність підтримувати «фазову пам’ять», тобто зберігати фазу сигналу при поверненні на попередню частоту, що особливо цінно у радіозв’язку з фіксованими каналами.

Синтезатор на основі фазового автопідстроювання частоти (PLL) базується на порівнянні частоти і фази вихідного сигналу з опорною частотою через фазовий детектор, управляючи генератором ГУН (рисунок 2.2).

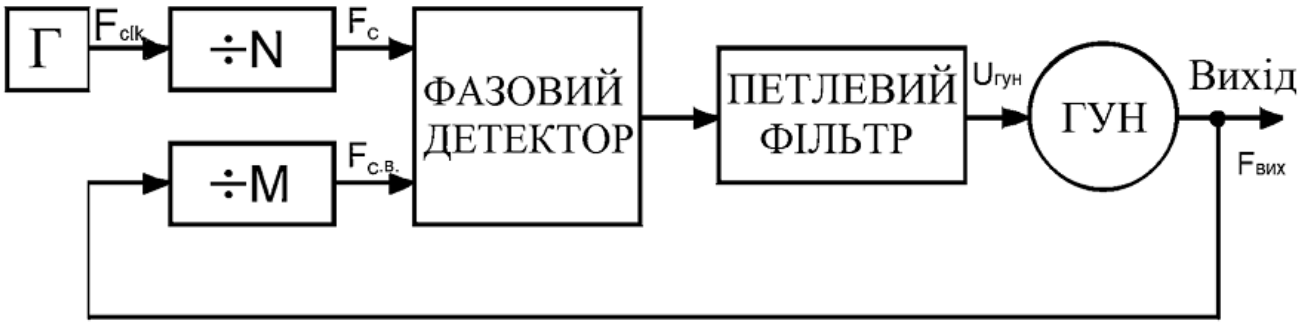


Рисунок 2.2 — Синтезатор на базі фазового автоматичного налаштування частоти

Вихідна частота формується як добуток або відношення коефіцієнтів дільників.

Головним недоліком цього методу є наявність фазових шумів, що генеруються фазовим детектором, а також складність забезпечення високої швидкості перебудови частоти. Використання багатопетльових систем PLL дозволяє дещо покращити характеристики, проте це ускладнює схему та збільшує її вартість.

Прямий цифровий синтез (DDS) — це сучасна технологія генерації сигналів з високою точністю, яка почала набувати практичного значення з 1970-х років. Попри те, що принципи різних методів синтезу були відомі ще задовго до цього, саме DDS привертає все більше уваги завдяки розвитку мікроелектроніки та зниженню вартості спеціалізованих інтегральних мікросхем.

Сьогодні DDS-системи активно застосовуються у сфері зв'язку, вимірювальних приладів, радіоелектроніки, а також у побутовій та промисловій

апаратурі завдяки зручним інструментам розробки, широкій доступності та можливості цифрового керування.

Однією з ключових особливостей DDS є її цифрова природа: всі параметри синтезованого сигналу — частота, фаза і амплітуда — можуть бути точно визначені і керовані в будь-який момент часу. Це забезпечує надзвичайно стабільну роботу, оскільки цифрові системи практично не піддаються впливу температурних змін чи старіння компонентів, що характерно для аналогових рішень. Єдиним елементом, який залишається чутливим до аналогових обмежень, є цифро-аналоговий перетворювач (ЦАП), але загалом цифрова обробка забезпечує дуже високий рівень точності.

Основні переваги DDS-технологій:

— надзвичайно висока роздільна здатність при встановленні частоти й фази, оскільки вони задаються цифровими кодами;

— швидке перемикання частоти чи фази без фази зриву або перехідних процесів, що дозволяє формувати сигнал без паразитних викидів або перешкод;

— відсутність необхідності у точному налаштуванні опорного генератора, оскільки частотна перебудова має мінімальний крок;

— можливість цифрового керування за допомогою мікроконтролерів, що робить інтеграцію DDS в сучасні системи простою;

— підтримка синтезу сигналів з квадратурною модуляцією — сучасні DDS-мікросхеми мають узгоджені виходи I та Q;

— висока частотна роздільність, яка досягає часток герца навіть при вихідних частотах у десятки мегагерц — що не під силу аналоговим генераторам;

— висока швидкодія при зміні частоти, яка визначається лише швидкістю цифрового інтерфейсу, без впливу механізмів зворотного зв’язку, як у PLL-системах;

— легкість реалізації різних видів модуляції, включаючи частотну, фазову та амплітудну, що важливо в системах зв’язку.

Сучасні DDS-системи вигідно вирізняються як за технічними характеристиками, так і за економічною ефективністю. Вони компактні, легко програмуються, споживають помірну кількість енергії (в залежності від тактової частоти) і дозволяють реалізувати нові функції лише шляхом оновлення програмного забезпечення. Мікросхеми DDS зазвичай виготовляються за сучасними CMOS-технологіями з малим енергоспоживанням та підтримкою низьковольтної логіки.

Попри численні переваги, DDS має низку обмежень, обумовлених її цифровою природою:

— максимальна вихідна частота обмежена половиною частоти тактування (за теоремою Найквіста), а на практиці — ще меншою, через особливості ЦАП та фільтрації сигналу;

— підвищений рівень побічних гармонік, які можуть виникати через особливості цифрової апроксимації сигналу та обмежену розрядність ЦАП;

— високе енергоспоживання при роботі на високих тактових частотах, що робить DDS менш придатним для мобільних або енергонезалежних пристроїв.

Щоб глибше зрозуміти принцип дії DDS, слід розглянути її типову структуру, яка на перший погляд може виглядати складною. Проте логіка її побудови цілком обґрунтована й інтуїтивно зрозуміла, якщо почати з кінцевої мети: формування синусоїдального сигналу з заданою частотою.

Оскільки генерація сигналу відбувається в цифровому вигляді, для подачі його в аналогову форму необхідний ЦАП. На виході ЦАП розміщується фільтр нижніх частот (ФНЧ), що пригнічує спектральні складові, повторювані з частотою тактування F_CLK.

Безпосереднє обчислення значення синуса для кожного відліку в режимі реального часу через математичні операції (наприклад, через поліноміальне розкладання) є занадто повільним для високошвидкісних систем. Натомість використовується метод табличного представлення функції (так званий Look-Up Table, LUT), в якому зберігаються значення одного періоду синусоїди.

Код, що подається на адресні входи ПЗП (Постійного Запам’ятовуючого Пристрою), відповідає фазі, а значення на виході — значенню синуса для цієї фази. Щоб фаза змінювалася лінійно, використовується лічильник, який з кожним тактом формує нову адресу.

У найпростішому варіанті DDS містить: лічильник, таблицю значень синуса (в ПЗП), ЦАП та ФНЧ (рис. 2.3). Для зміни вихідної частоти використовується дільник частоти, який впливає на швидкість інкременту фази. Проте така реалізація має недоліки — перебудова частоти відбувається нерівномірно через дискретний поділ частоти, а частота дискретизації змінюється разом із вихідною частотою.

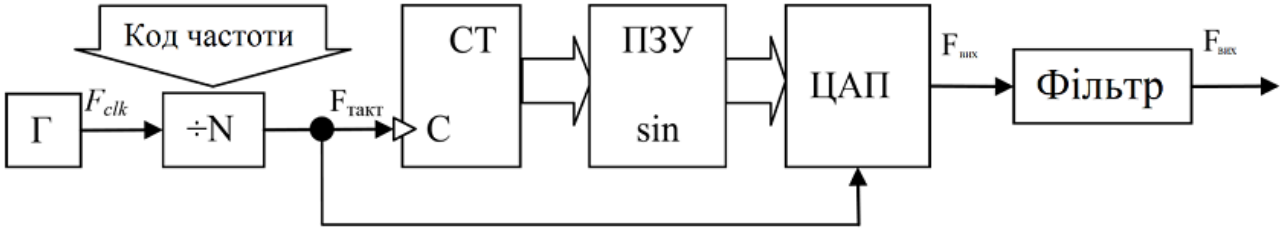


Рисунок 2.3 — Проста реалізація DDS

Щоб усунути ці недоліки, в DDS використовується накопичувальний суматор (фазовий накопичувач). Це цифровий блок, який постійно додає фіксовану величину — інкремент фази — до поточного значення фази. Це дозволяє точно керувати частотою вихідного сигналу при постійній тактовій частоті.

На кожному такті фазовий накопичувач генерує нову фазу, яка передається в ПЗП із таблицею синуса. Вихід з таблиці подається на ЦАП, а потім на ФНЧ, утворюючи гладкий синусоїдальний сигнал на виході (рис. 2.4).

Такий підхід дозволяє зберігати стабільність частоти дискретизації, забезпечуючи кращу якість фільтрації та максимальне використання можливостей ЦАП.

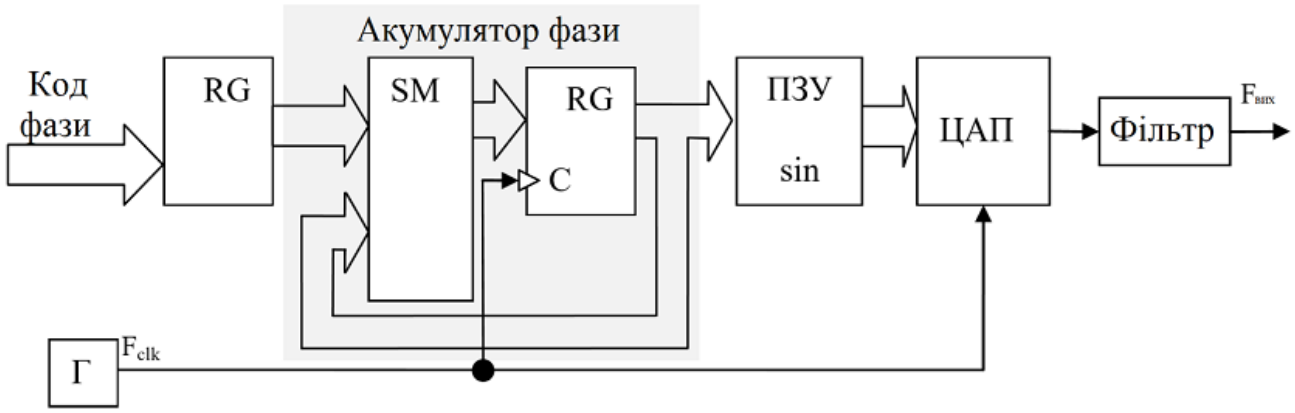


Рисунок 2.4 — Удосконалена структура DDS з фазовим накопичувачем

Для систем, що вимагають високої швидкодії, стабільності, малої вартості і гнучкості, оптимальним рішенням є застосування DDS. Водночас для завдань із широким діапазоном частот і дуже низькими фазовими шумами доцільно розглядати комбінації з PLL та DAS.

При проектуванні систем на мікроконтролерах важливо враховувати функціональні вимоги, такі як швидкість перебудови, якість спектру, можливість програмного керування та інтеграція з іншими цифровими блоками, що робить DDS особливо привабливим для інтеграції в сучасні мікроконтролерні системи.

2.2 Огляд архітектури мікроконтролерів

Сучасний розвиток електронної техніки характеризується стрімким і глибоким впровадженням мікропроцесорних рішень у всі сфери техніки, особливо у приладобудування. Це відкриває нові горизонти в автоматизації, розширює функціональні можливості пристроїв і підвищує їхню ефективність. Однією з ключових галузей, де мікропроцесори посідають провідні позиції, є інформаційно-вимірювальні системи.

Мікропроцесор являє собою інтегральну електронну схему, яка виконує обробку даних за заздалегідь визначеною програмою. Основною його функцією є автоматизоване керування процесами обробки інформації в електронних пристроях. Завдяки мікропроцесорам вимірювальні прилади набувають

інтелектуальних рис: вони можуть адаптуватися до змін зовнішніх умов, взаємодіяти з користувачем, вести діалог із системами збору й аналізу інформації, а також обробляти результати вимірювань у режимі реального часу.

Запровадження мікропроцесорного керування у вимірювальні пристрої надає низку переваг, які наведені нижче.

Розширення функціональності приладу. Завдяки програмному забезпеченню, прилади здатні виконувати широкий спектр обчислень і вимірювань, у тому числі непрямі та комплексні. Це дає змогу суттєво зменшити кількість зовнішніх пристроїв і одночасно підвищити точність, швидкодію та гнучкість вимірювального процесу. За допомогою клавіатури оператор може вибрати необхідний режим роботи, після чого система автоматично проводить обробку, зберігає проміжні результати та виводить кінцеве значення фізичної величини.

Покращення метрологічних характеристик. Мікропроцесор дозволяє реалізовувати алгоритми компенсації похибок, корекції нестабільності елементів схеми, а також обробки статистичних даних для зниження випадкових відхилень. Застосування принципу інваріантності — коли результати вимірювань не залежать від змін навколишнього середовища чи внутрішніх параметрів приладу — забезпечується шляхом багаторазових вимірювань і програмної корекції.

Підвищення рівня сервісного обслуговування. Користувач отримує можливість працювати з даними у зручному вигляді — з графічними, кольоровими, анімаційними та звуковими підказками. З’являється функція самодіагностики пристрою, автоматичного калібрування та збереження історії вимірювань.

Інтеграція у складні системи. Завдяки наявності комунікаційних інтерфейсів, прилади з мікропроцесорами можуть бути об'єднані у вимірювальні комплекси або автоматизовані системи керування. Наприклад, через послідовні порти (UART, I^2C, SPI тощо) можна організувати взаємодію з комп’ютером, ПЛК чи іншими пристроями.

Вибір мікропроцесора для створення конкретного пристрою є відповідальним етапом проєктування. Потрібно враховувати як його

продуктивність, так і можливості щодо розширення функціоналу, кількість доступних периферійних модулів, підтримку сучасних інтерфейсів тощо. Надмірне різноманіття моделей і архітектур створює додаткову складність у процесі вибору оптимального рішення, тому важливо визначити перелік необхідних функцій ще на етапі технічного завдання.

Зараз провідні виробники мікроконтролерів і мікропроцесорів постійно оновлюють свої лінійки, пропонуючи як аналогові, так і цифрові варіанти. Попит на такі рішення свідчить про актуальність теми і про те, що ринок давно був готовий до їх масового застосування.

Методи побудови пристроїв поділяють на два основних напрями — на основі мікропроцесорів (МП) і на базі мікроконтролерів (МК). Розглянемо останні докладніше.

Мікроконтролер — це спеціалізована мікросхема, яка поєднує в собі процесорне ядро, пам’ять програм і даних, а також набір периферійних інтерфейсів. Його можна уявити як повноцінний комп’ютер в одному корпусі, оптимізований для керування електронними пристроями. Така інтеграція дозволяє мінімізувати габарити, енергоспоживання та вартість кінцевого пристрою. Саме тому МК широко застосовуються в побутовій техніці, мобільних телефонах, системах автоматики, вбудованих рішеннях тощо.

Мікропроцесор, на відміну від МК, виконує переважно функції центрального процесора і потребує зовнішніх компонентів для повноцінної роботи (оперативна пам'ять, системна шина, периферія). Однак із розвитком технологій відмінності між МК і МП поступово стираються.

Одним із яскравих прикладів мікроконтролерної архітектури є AVR від компанії Atmel. Ці МК реалізовані на основі RISC-архітектури (Reduced Instruction Set Computer) — тобто архітектури зі зменшеним і оптимізованим набором команд. Основні характеристики цієї архітектури:

— мінімальна кількість універсальних, але високопродуктивних команд;

— рівноправний набір робочих регістрів замість одного суматора;

— окрема пам’ять для коду і даних (гарвардська архітектура);

— використання інструкцій лише типу "load/store" для роботи з пам’яттю;

— виконання більшості команд за один машинний такт;

— зручність для програмування мовами високого рівня, зокрема Сі.

У минулому домінували CISC-процесори (Complex Instruction Set Computer), що мали великі набори складних команд, типові для сімейств Intel x86 або Motorola 68000. Вони обробляли кожну команду мікропрограмою, вбудованою у мікросхему, що призводило до затримок при виконанні інструкцій. Але дослідження показали, що більшість програм активно використовує лише малу частину цього інструкційного набору. Це стало підґрунтям для переходу до RISC-підходу.

AVR-контролери (див. рис. 2.5) демонструють ефективність RISC-архітектури: вони здатні виконувати більшість команд за один тактовий цикл завдяки одноступеневому конвеєру — одночасно з виконанням однієї команди здійснюється завантаження наступної. Це дає змогу досягати продуктивності до 12 MIPS (мільйонів інструкцій за секунду), що надзвичайно ефективно для задач реального часу.

На відміну від класичних контролерів (наприклад, 8051), де одна команда може виконуватись декілька тактів, AVR забезпечує стабільно короткий час виконання — наприклад, при частоті 12 МГц одна команда може виконуватись за ~83 нс.

Переваги AVR також у тому, що вони проектувалися з урахуванням потреб мови Сі: інструкції для роботи з покажчиками, автоінкремент/декремент адрес, багатоформатні режими адресації пам’яті тощо. Це полегшує розробку програмного забезпечення і робить архітектуру зручною для компіляторів мов високого рівня.

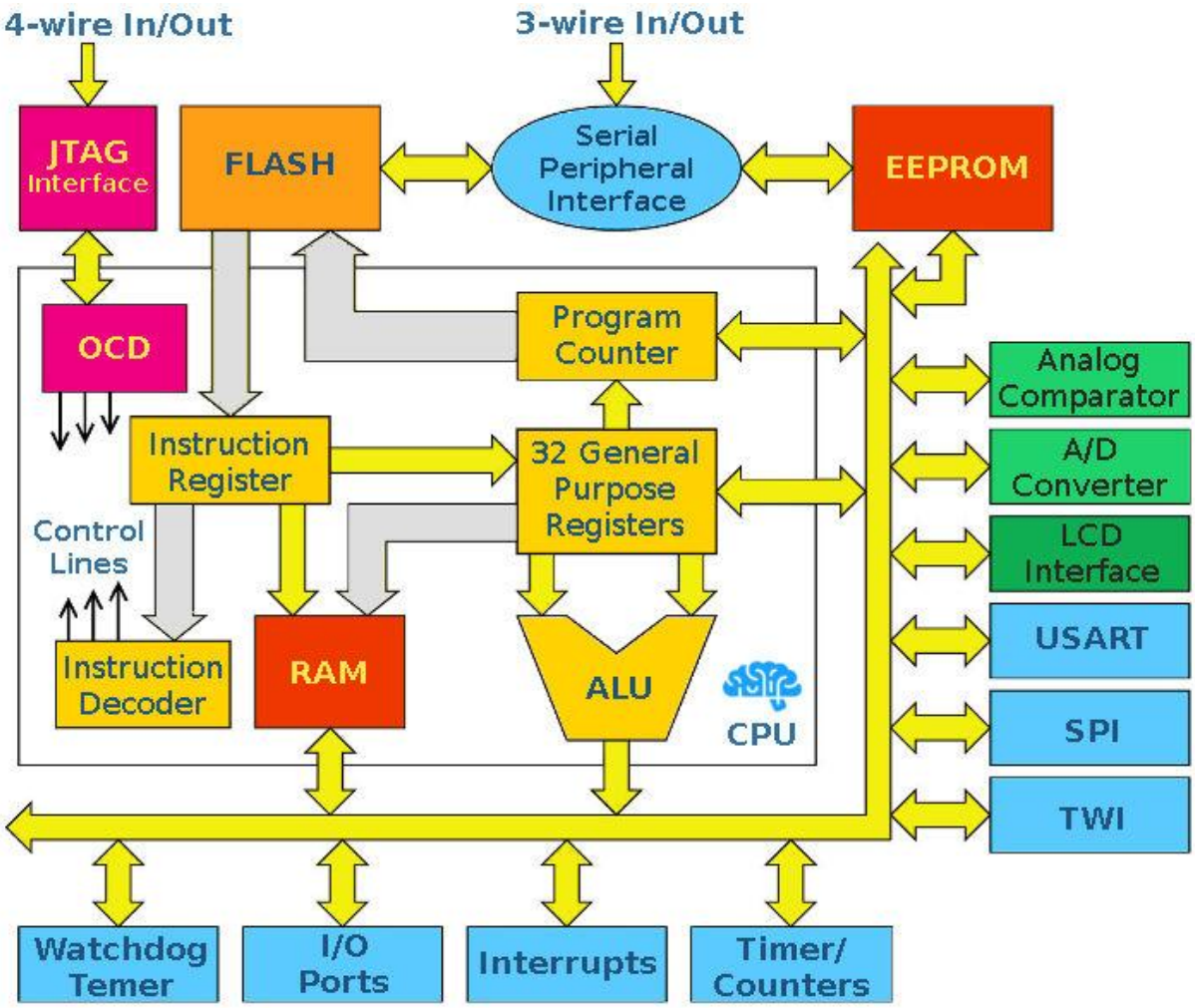


Рисунок 2.5 — Структурна схема AVR-контролерів

2.3 Методика програмування мікроконтролерів

Перед початком створення програмного забезпечення для мікроконтролерів необхідно ознайомитися з типами пам’яті, які вони використовують, а також з особливостями їх застосування у процесі програмування. Основні типи пам’яті мікроконтролера включають: пам’ять програм, оперативну пам’ять, регістрову пам’ять, енергонезалежну пам’ять даних (EEPROM).

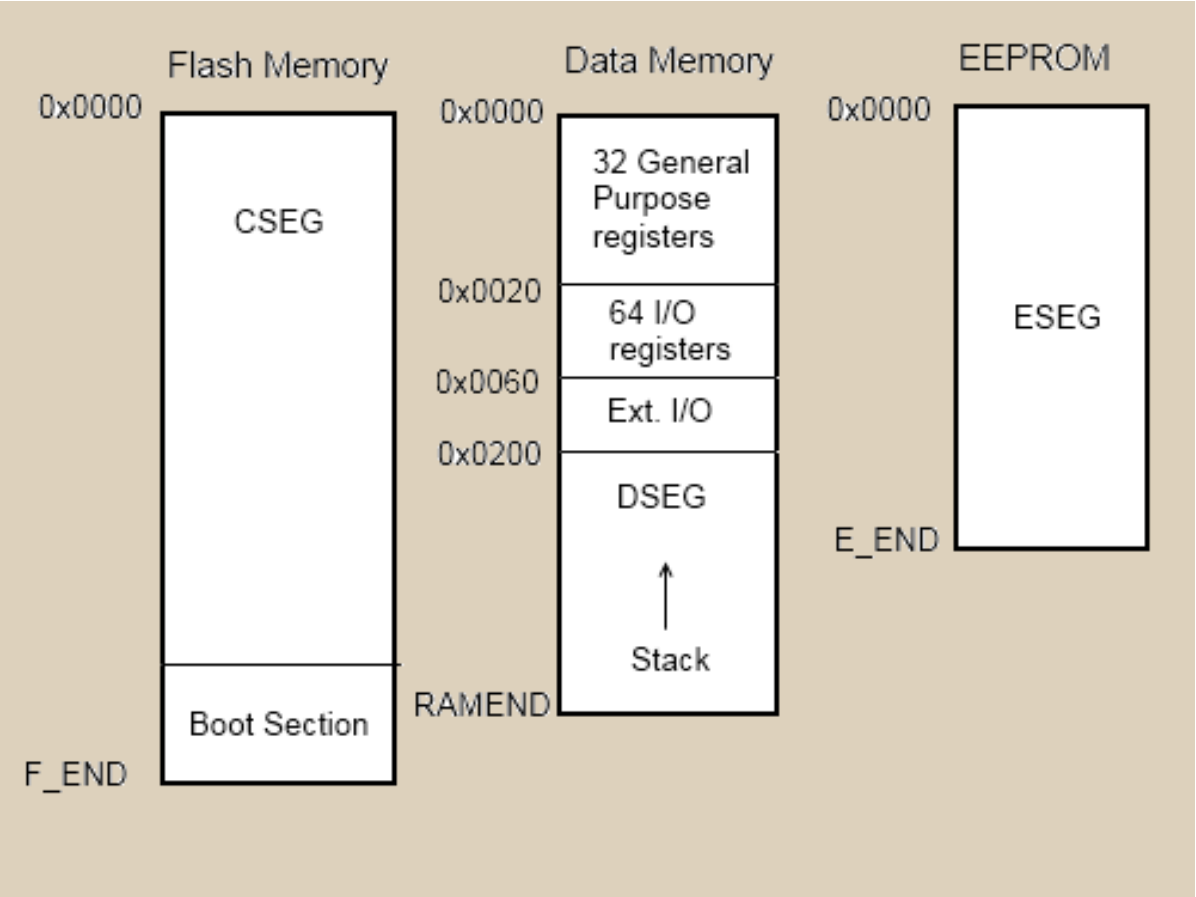


Рисунок 2.6 — Основні типи пам’яті мікроконтролера

Пам’ять програм (Flash ROM)—Flash-пам’ять є постійною енергонезалежною пам’яттю, в якій зберігається виконуваний код мікроконтролера. Вона має 16-бітну архітектуру та може мати обсяг від 1 до 256 КБ залежно від моделі. Flash-пам’ять є багаторазово перезаписуваною, тобто допускає неодноразове стирання та перезапис інформації завдяки використанню технології електричного перепрограмування.

Програмне забезпечення записується у Flash як за допомогою спеціального програматора, так і через інтерфейс SPI. Важливою перевагою мікроконтролерів AVR є можливість програмування безпосередньо на зібраній платі за допомогою технології ISP (In-System Programming). Ця функція підтримується більшістю AVR-контролерів, за винятком деяких моделей, наприклад Tiny11 і Tiny28.

У серії Mega передбачена можливість самопрограмування – зміни власного програмного коду під час виконання. Це відкриває можливість створення

адаптивних систем, які здатні змінювати свою логіку роботи у відповідь на зміну умов або сигналів ззовні. За даними компанії Atmel, кількість гарантованих циклів перезапису Flash-пам’яті становить щонайменше 10 000, хоча на практиці цей показник може досягати 100 000 циклів.

Пам’ять для збереження та обробки даних у мікроконтролері розділяється на декілька категорій:

Регістрова пам’ять (GPR та I/O-регістр) – складається з 32 регістрів загального призначення та низки службових регістрів, які керують внутрішніми і периферійними модулями мікроконтролера. Хоча вони знаходяться у тому ж адресному просторі, що й оперативна пам’ять, фактично вони не є її частиною. Програмна взаємодія з мікроконтролером здебільшого зводиться до зміни значень цих регістрів.

EEPROM (електрично стирана постійна пам’ять) – використовується для довготривалого зберігання змінної інформації, що повинна залишитися після вимкнення живлення. Обсяг EEPROM у мікроконтролерах AVR коливається в межах від 64 байт до 4 КБ. EEPROM доступна під час виконання програми, що дозволяє використовувати її для зберігання серійних номерів, конфігурацій, параметрів калібрування тощо. Кількість гарантованих циклів перезапису – не менше 100 000.

Оперативна пам’ять (SRAM) – забезпечує тимчасове зберігання даних під час виконання програм. Вона характеризується байтовою організацією і використовується для збереження змінних, стеку та проміжних результатів обчислень. Розмір SRAM варіюється від 64 байт до 4 КБ у залежності від моделі контролера. Після вимкнення живлення вміст цієї пам’яті стирається. У деяких мікроконтролерах є можливість підключення зовнішньої SRAM обсягом до 64 КБ.

Після аналізу типів пам’яті можна перейти до практичного створення програмного коду для мікроконтролера. Програмування мікроконтролерів може здійснюватися мовами низького або високого рівня – такими як Assembler, C, a

також менш поширеними: BASIC або Fortran (через використання відповідних компіляторів або інтерпретаторів).

Для розробки, відлагодження та тестування програм використовують наступні засоби:

— програмні симулятори, які моделюють поведінку мікроконтролера на ПК;

— внутрішньосхемні емулятори, що фізично підключаються до тестованого пристрою замість мікроконтролера.

— інтерфейс JTAG – універсальний стандарт IEEE 1149.1, який забезпечує відлагодження та запис програми безпосередньо в мікроконтролер.

Створення програми починається з написання коду на вибраній мові програмування. Наприклад, при використанні асемблера MASM (Microsoft Assembler) процес може проходити в діалоговому режимі, коли система запитує наступну інформацію:

— Ім’я початкового (вихідного) файлу з розширенням .ASM або без нього.

— Ім’я об’єктного файлу – результат компіляції, розширення .OBJ.

— Ім’я файлу лістингу – звіт про трансляцію, розширення .LST.

— Ім’я файлу перехресних посилань — для аналізу залежностей, розширення .CRF.

Відповіді можуть містити додаткові параметри запуску асемблера, а якщо користувач введе символ ;, програма автоматично встановить параметри за замовчуванням.

Після написання програмного коду необхідно провести компіляцію – процес перетворення коду у двійкову форму, зрозумілу мікроконтролеру. Отриманий машинний код записується до Flash-пам’яті контролера за допомогою програматора. Програматор передає бінарний файл через інтерфейс (наприклад, SPI або JTAG), що дозволяє запрограмувати контролер як до початку його роботи, так і вже після встановлення на плату.

Процес розробки програмного забезпечення для мікроконтролерів можна представити у вигляді послідовної схеми.

Ось приклад діаграми створення програми для мікроконтролерів, яка ілюструє основні етапи розробки:

— формулювання технічного завдання: визначення функціональних вимог до системи;

— розробка алгоритму: побудова логічної послідовності дій для виконання завдань;

— створення блок-схеми: графічне представлення алгоритму з використанням стандартних символів;

— написання коду програми: реалізація алгоритму на мові програмування (наприклад, C або Assembler);

— компіляція та трансляція: перетворення вихідного коду у виконуваний машинний код.

— програмування мікроконтролера: завантаження машинного коду у пам'ять мікроконтролера за допомогою програма тора;

— тестування та налагодження: перевірка роботи програми та виправлення помилок.

Ця діаграма допомагає візуалізувати процес розробки програмного забезпечення для мікроконтролерів і є корисною для планування та контролю етапів проекту.

Таким чином, програмування мікроконтролера є складовою частиною проектування мікропроцесорних систем, яка поєднує знання про типи пам’яті, можливості компіляції, інтерфейси запису, та методи відлагодження.

3 ПРОЕКТУВАННЯ МІКРОПРОЦЕСОРНОЇ СИСТЕМИ ГЕНЕРУВАННЯ СИГНАЛІВ

3.1 Вибір технічної бази та обґрунтування компонентів

Підбір мікроконтролера (МК) здійснюється з урахуванням вимог, сформульованих у технічному завданні. На основі цих вимог визначаються ключові технічні параметри, яким повинен відповідати МК, а саме:

— достатній рівень обчислювальної швидкодії;

— наявність необхідних периферійних інтерфейсів;

— відповідна кількість каналів вводу/виводу;

— низький рівень енергоспоживання.

Серед сучасних мікроконтролерів, що задовольняють зазначені умови, було обрано представника сімейства AVR. Це 8-бітові мікроконтролери з архітектурою RISC, що розроблені компанією Atmel. Вони успішно застосовуються в системах з вбудованим керуванням завдяки високій продуктивності, універсальності та простоті у програмуванні.

Мікроконтролери AVR вирізняються високою швидкодією, яка дозволяє їм у ряді завдань замінювати дорожчі 16-бітні аналоги. Це забезпечує оптимізацію як продуктивності, так і вартості розробки. До того ж, AVR-пристрої зручно програмуються мовою Assembler або мовами високого рівня.

З-поміж широкого асортименту мікроконтролерів Atmel доцільним є вибір моделі ATmega16, яка відповідає всім необхідним вимогам (див. рис. 3.1).

ATmega16 — це 8-розрядний мікроконтролер, виготовлений за технологією CMOS з використанням розширеної RISC-архітектури. Завдяки виконанню однієї інструкції за такт, забезпечується висока ефективність по відношенню до швидкодії та енергоспоживання. Основу ядра AVR складає набір з 32 універсальних регістрів, що безпосередньо з’єднані з арифметико-логічним пристроєм (АЛП), дозволяючи виконувати інструкції за один такт.

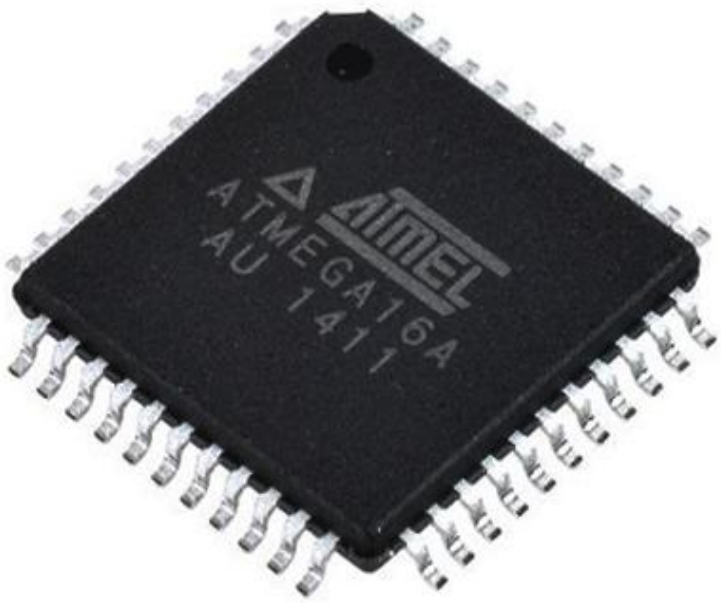


Рисунок 3.1 — Зовнішній вигляд ATmega16

Подібна архітектура забезпечує суттєве скорочення розміру коду в порівнянні з традиційними CISC-мікроконтролерами.

Ключові характеристики ATmega16:

Архітектура ядра:

— розширений набір з 130 команд, більшість з яких виконуються за один такт;

— 32 загального призначення восьмирозрядні регістри;

— повністю статичний режим роботи;

— продуктивність до 16 MIPS при 16 МГц;

— вбудований двотактний апаратний множник.

Пам’ять:

— 16 КБ Flash-пам’яті з підтримкою внутрішньої перезапису (ресурс – 1000 циклів);

— сектор завантаження з окремими бітами захисту;

— EEPROM обсягом 512 байт (до 100 тис. циклів перезапису);

— 1 КБ статичної оперативної пам’яті (SRAM);

— можливість підключення до 64 КБ зовнішньої пам’яті;

— захист коду за допомогою програмованих бітів.

Вбудовані периферійні блоки:

— два 8-бітних і один 16-бітний таймери з компараторами;

— до 4-х каналів PWM;

— 10-бітний 8-канальний АЦП;

— до 7-ми диференційних каналів (для корпусів TQFP);

— інтерфейс USART для асинхронної/синхронної передачі;

— підтримка SPI (Master/Slave);

— сторожовий таймер;

— вбудований аналоговий компаратор.

Системні функції:

— детектор включення живлення та контроль зниження напруги;

— калібрований внутрішній RC-генератор;

— переривання з зовнішніх і внутрішніх джерел;

— до 6-ти режимів енергозбереження: Idle, Power-save, Power-down, Extended, Standby тощо.

Інтерфейси вводу/виводу:

— 32 програмованих лінії вводу/виводу;

— доступні варіанти корпусів: 40-вивідний PDIP, 44-вивідні TQFP, PLCC, MLF.

Напруга живлення:

— ATmega16L: 2.7–5.5 В;

— ATmega16: 4.5–5.5 В.

Частотні характеристики:

— ATmega16L: до 8 МГц;

— ATmega16: до 16 МГц.

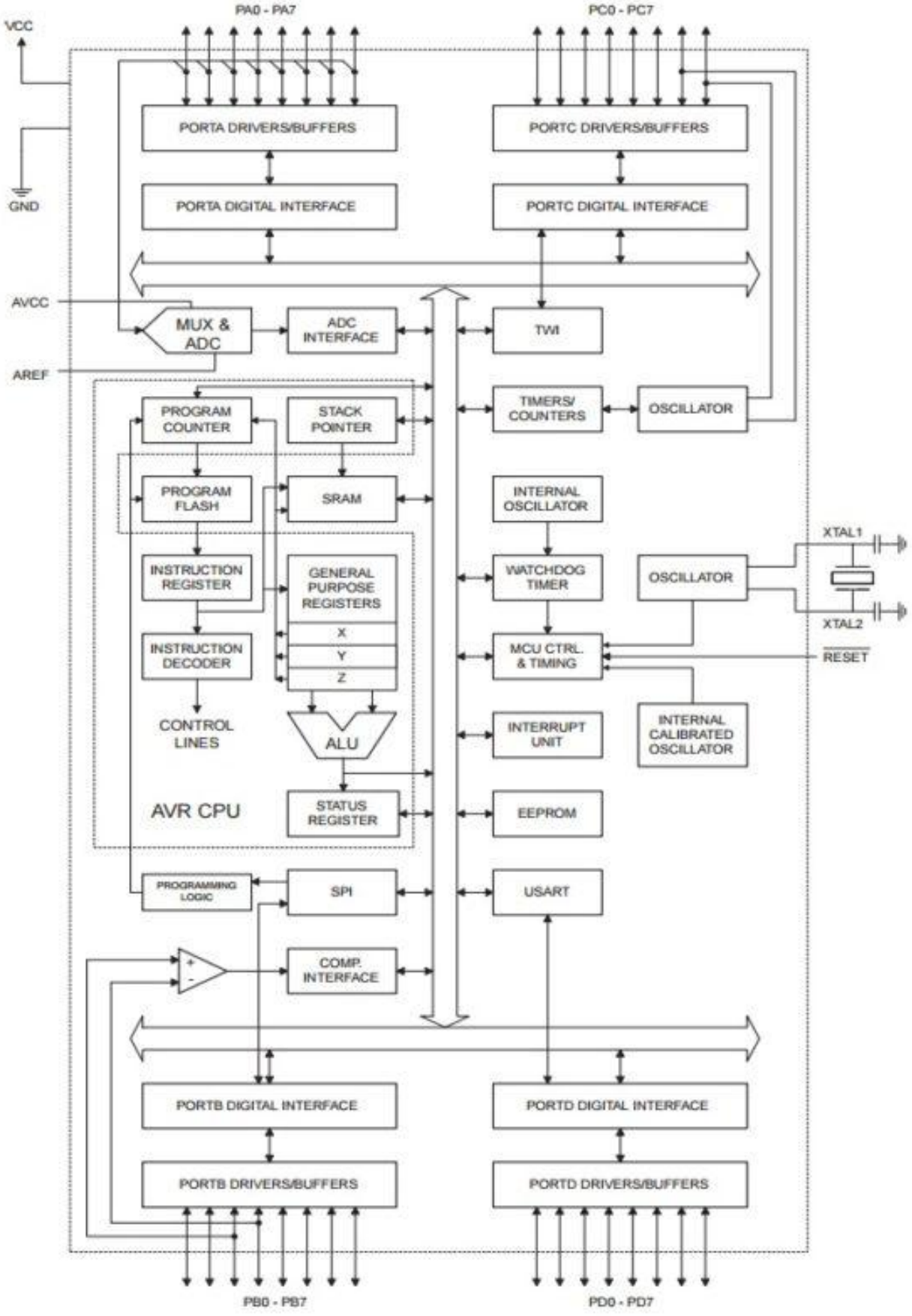


Рисунок 3.2 — Узагальнена структурна схема мікроконтролера ATmega16

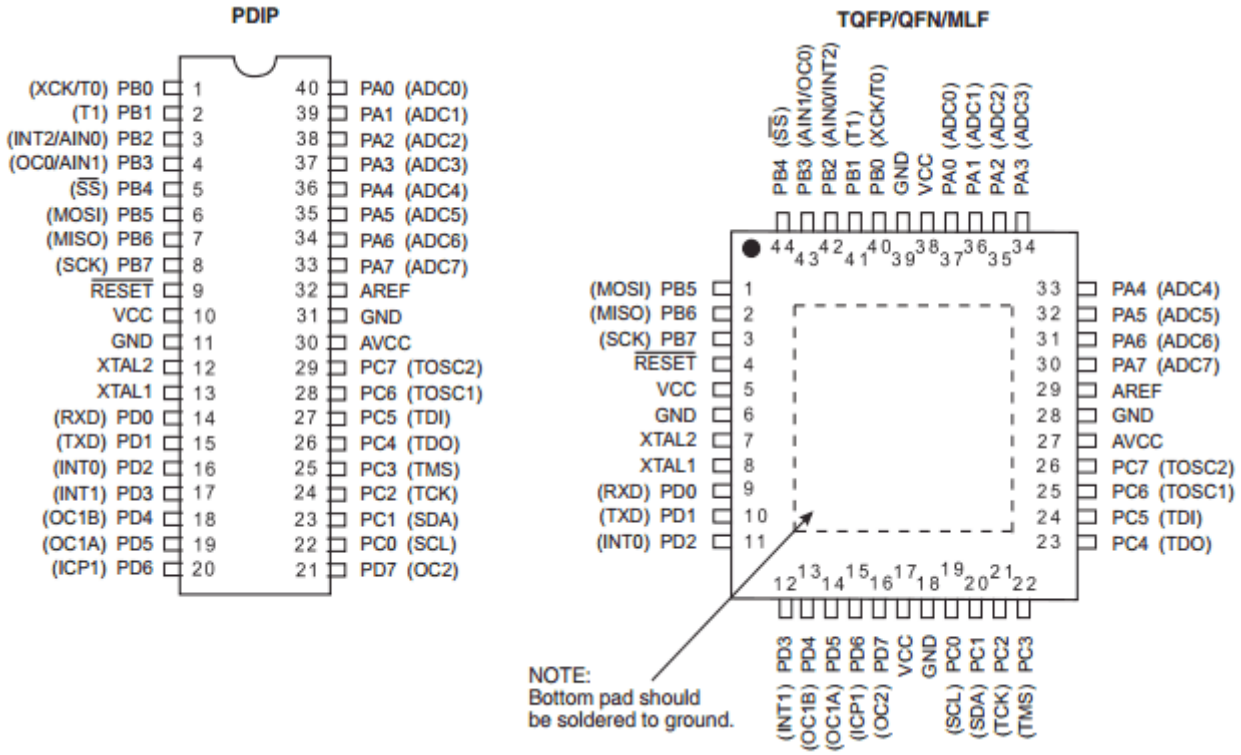


Рисунок 3.3— Розміщення виводів ATmega16 для різних корпусів

У склад ATmega16 входить набір основних функціональних блоків, серед яких:

— блок керування тактуванням (GCK);

— центральний процесор (CPU);

— Flash-пам’ять для зберігання коду;

— SRAM — статична оперативна пам’ять;

— EEPROM для збереження змінних даних;

— набір периферійних пристроїв для вводу/виводу і додаткових функцій.

Центральний процесор ATmega16 включає:

— лічильник команд (PC);

— арифметико-логічний пристрій (ALU);

— регістровий файл загального призначення (GPR);

— регістр стану SREG;

— регістри-індикатори стеку SP (SPH/SPL).

Важливою особливістю AVR є наявність регістрів спеціального призначення, які позначаються як регістри вводу/виводу (IOR). Їх використання дозволяє:

— здійснювати керування апаратними блоками;

— визначати статус внутрішніх пристроїв;

— передавати дані та команди в МК та з нього.

Порти вводу/виводу ATmega16:

— Port A (PA7–PA0) — двонаправлений порт з внутрішніми підтягувальними резисторами, також використовується як мультиплексна шина адреси/даних при підключенні зовнішньої пам’яті;

— Port B (PB7–PB0) — універсальний двонаправлений порт, також підтримує спеціальні функції;

— Port C (PC7–PC0) — порт виводу, також використовується як шина адреси;

— Port D (PD7–PD0) — двонаправлений порт з підтягувальними резисторами.

Вхід RESET забезпечує ініціалізацію системи при подачі сигналу низького рівня на протязі двох тактів.

ATmega16 є синхронним пристроєм, тобто усі процеси керуються тактовими імпульсами. Джерелом тактового сигналу може бути:

— внутрішній генератор з кварцовим або керамічним резонатором (XTAL);

— внутрішній RC-генератор (IRC);

— зовнішнє RC-коло (ERC);

— зовнішній генератор сигналів (EXT).

При використанні резонатора XTAL1 і XTAL2 виступають у ролі входу та виходу інвертуючого підсилювача, через який формується генератор тактових імпульсів. Для виводу сигналу на зовнішні пристрої можна використовувати XTAL2.

Центральний процесор виконує команди шляхом послідовної вибірки та виконання інструкцій із пам’яті. Завдяки архітектурі AVR, АЛП виконує операції безпосередньо з регістрами за один такт. Підтримуються арифметичні, логічні й бітові операції, що забезпечує гнучкість і високу продуктивність.

3.2 Розробка електричної принципової схеми

Відповідно до технічного завдання, розроблюваний генератор повинен відповідати таким основним вимогам:

— проста схема, побудована з доступних та недорогих компонентів;

— частотний діапазон вихідного сигналу – від 1 МГц до 8 МГц;

— підтримка різних типів сигналів: синусоїда, прямокутний, трикутний, пилкоподібний, ЕКГ, шум;

— наявність клавіатури з п’яти кнопок для керування режимами роботи;

— дворядковий LCD-дисплей для виводу інформації про параметри;

— налаштування частоти з кроком 1, 10, 100, 1000, 10000 Гц;

— збереження налаштувань після вимкнення живлення;

— вихід DDS-сигналу з можливістю регулювання амплітуди та постійного зсуву;

— діапазон регулювання зсуву: від -5 В до +5 В;

— регулювання амплітуди: від 0 до 10 В;

— регулювання частоти: в межах 0...65534 Гц.

З економічної точки зору, необхідно забезпечити, щоб пристрій залишався бюджетним, не втрачаючи при цьому конкурентоспроможності щодо зарубіжних аналогів.

Для виведення параметрів вихідного сигналу необхідно передбачити ефективну індикаційну систему. Найбільш відповідним варіантом є використання рідкокристалічного дисплея, який здатен відображати як символічну інформацію про форму сигналу, так і цифрові значення частоти. Для керування дисплеєм достатньо шести сигнальних ліній, які забезпечать обмін інформацією між мікроконтролером і індикатором. Цей тип дисплеїв широко застосовується в побутовій та промисловій електроніці завдяки надійності, дешевизні та простоті інтеграції.

Оскільки генератор підтримує кілька режимів роботи, виникає потреба у наявності інтерфейсу для зміни параметрів. Для цього застосовано пульт керування

на базі шести тактових кнопок без фіксації. Матриця клавіатури, яка часто використовується в складніших системах, у цьому випадку була б надмірною, оскільки потреби обмежуються шістьма елементами управління. Таким чином, проста схема з індивідуальними кнопками є більш доцільною.

Програмна частина забезпечує повне керування усіма режимами пристрою. Кнопки підключено до порту D мікроконтролера, а генерація сигналу виконується через порт A, який пов'язано з цифро-аналоговим перетворювачем (ЦАП). Саме ЦАП дозволяє сформувати аналоговий сигнал відповідної форми й амплітуди.

Регулювання зсуву і амплітуди реалізується за допомогою операційного підсилювача, який отримує сигнал із виходу ЦАП. Така схема дозволяє формувати сигнал з належними характеристиками для подальшого використання у різноманітних задачах.

Для побудови повноцінної системи необхідні наступні функціональні блоки:

— мікроконтролер — основний елемент управління, генерує сигнали та обробляє команди користувача;

— блок керування — забезпечує введення команд і зміну параметрів;

— блок індикації — виводить поточні параметри на екран;

— блок живлення — стабілізує напругу для живлення усіх модулів;

— цифро-аналоговий перетворювач — формує аналоговий сигнал з цифрових даних;

— підсилювач з регулюванням зсуву і амплітуди — дозволяє підлаштувати вихідний сигнал під вимоги користувача.

Структурну схему пристрою представлено на рисунку 3.4.

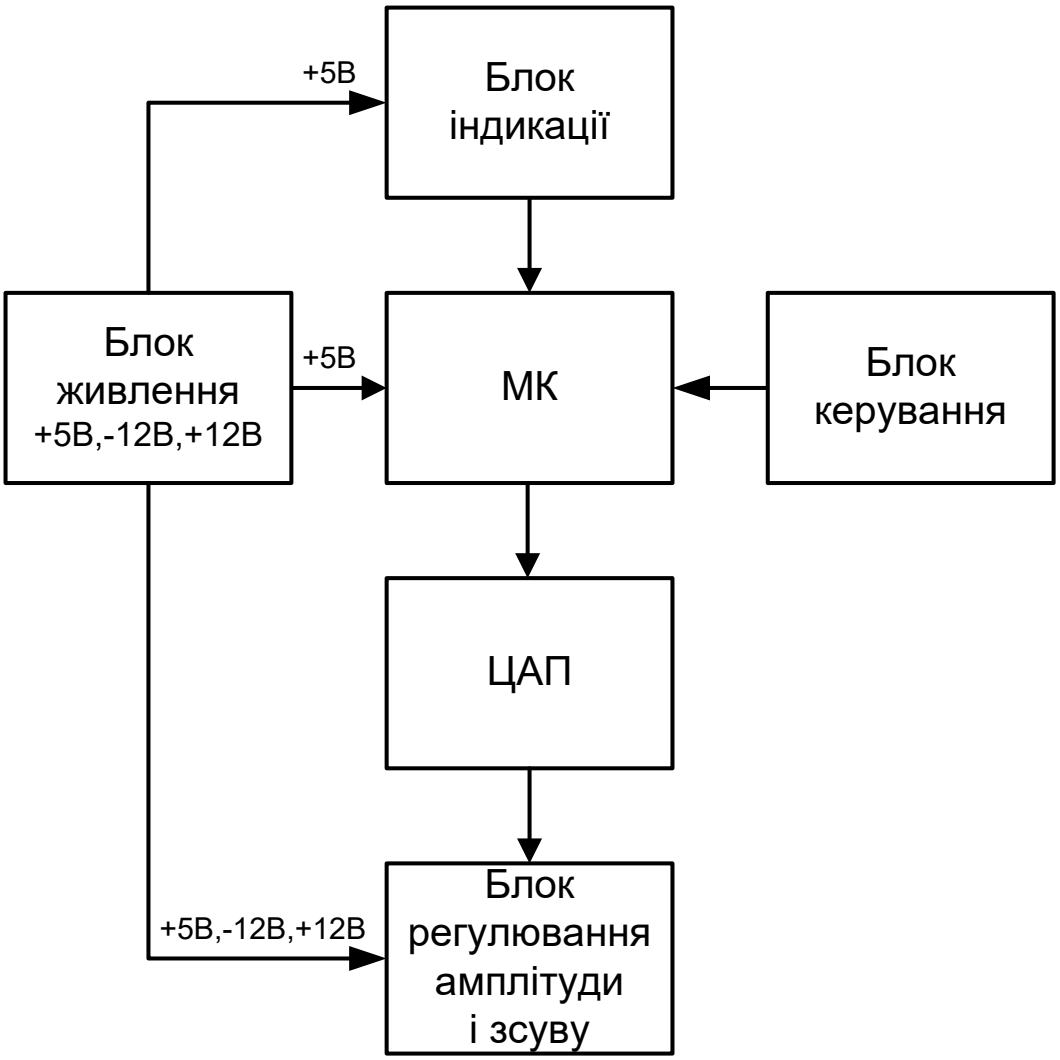


Рисунок 3.4 — Структурна схема системи

Цифро-аналоговий перетворювач (ЦАП) виконує ключову роль у формуванні аналогового сигналу на основі цифрового коду, що формується мікроконтролером. ЦАП виступає мостом між цифровим світом керування і аналоговою природою фізичних процесів. Сигнал, який формується на його виході, часто потребує фільтрації для згладжування та наближення до ідеальної форми. Приклад сигналу з ЦАП без інтерполяції показано на рисунку 3.5.

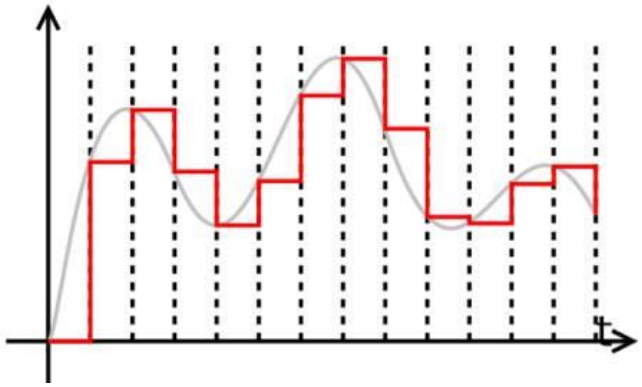


Рисунок 3.5 — Сигнал з ЦАП без інтерполяції на фоні ідеального сигналу

Нижче наведено найпоширеніші типи ЦАП.

Широтно-імпульсний (PWM) — простий та недорогий варіант, використовує ширину імпульсу для відображення амплітуди. Широко застосовується в системах керування та аудіотехніці.

ЦАП дельта-сигма (передискретизація) — базується на модуляції щільності імпульсів, забезпечує високу точність при мінімальних витратах. Ідеальний для аудіосистем високої роздільної здатності.

Ваговий (weighted) — реалізується через резисторну матрицю з індивідуальними джерелами струму. Дає високу швидкодію, але низьку точність.

R-2R ланцюгова схема — зважений тип ЦАП з високою точністю і простотою реалізації. Ідеально підходить для середньошвидкісних задач.

Сегментний ЦАП — містить окремий сегмент для кожного можливого значення. Найшвидший, але й найдорожчий.

Гібридні ЦАП — поєднують елементи кількох попередніх типів, що дозволяє балансувати між вартістю, точністю та швидкодією.

У межах цього проєкту найраціональнішим варіантом є використання ЦАП типу R-2R через простоту виготовлення, точність та стабільність характеристик. Швидкодія не є критичною, оскільки частоти, з якими працює система, не перевищують 16 кГц. Схема такого ЦАП наведена на рисунку 3.6.

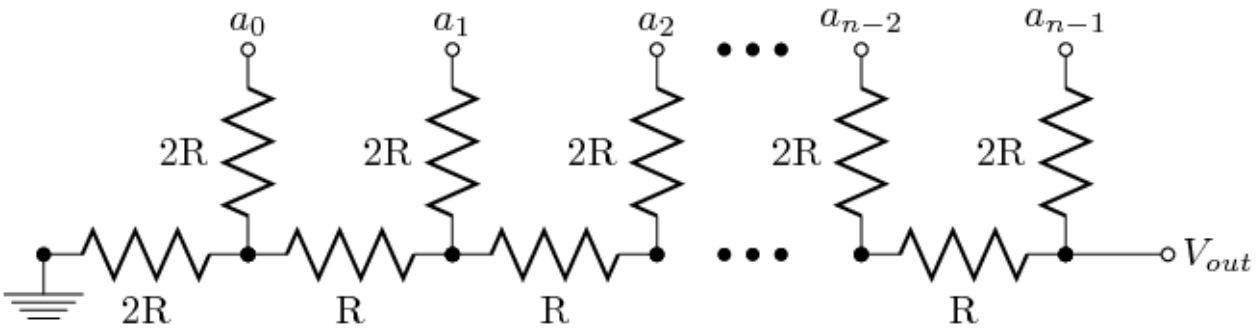


Рисунок 3.6— Структура ЦАП типу R-2R

R-2R ЦАП реалізується за допомогою резисторної драбинки, в якій використовуються лише два номінали резисторів — R і 2R. Це спрощує виготовлення та підвищує точність, оскільки всі резистори можуть бути підібрані з однієї прецизійної серії. Недоліком є потенційна немонотонність при переході між значеннями, наприклад, при зміні з 01111 на 10000.

Для реалізації такого ЦАП достатньо стандартних резисторів на 10 кОм і 20 кОм, що є оптимальними з урахуванням робочих струмів в схемі.

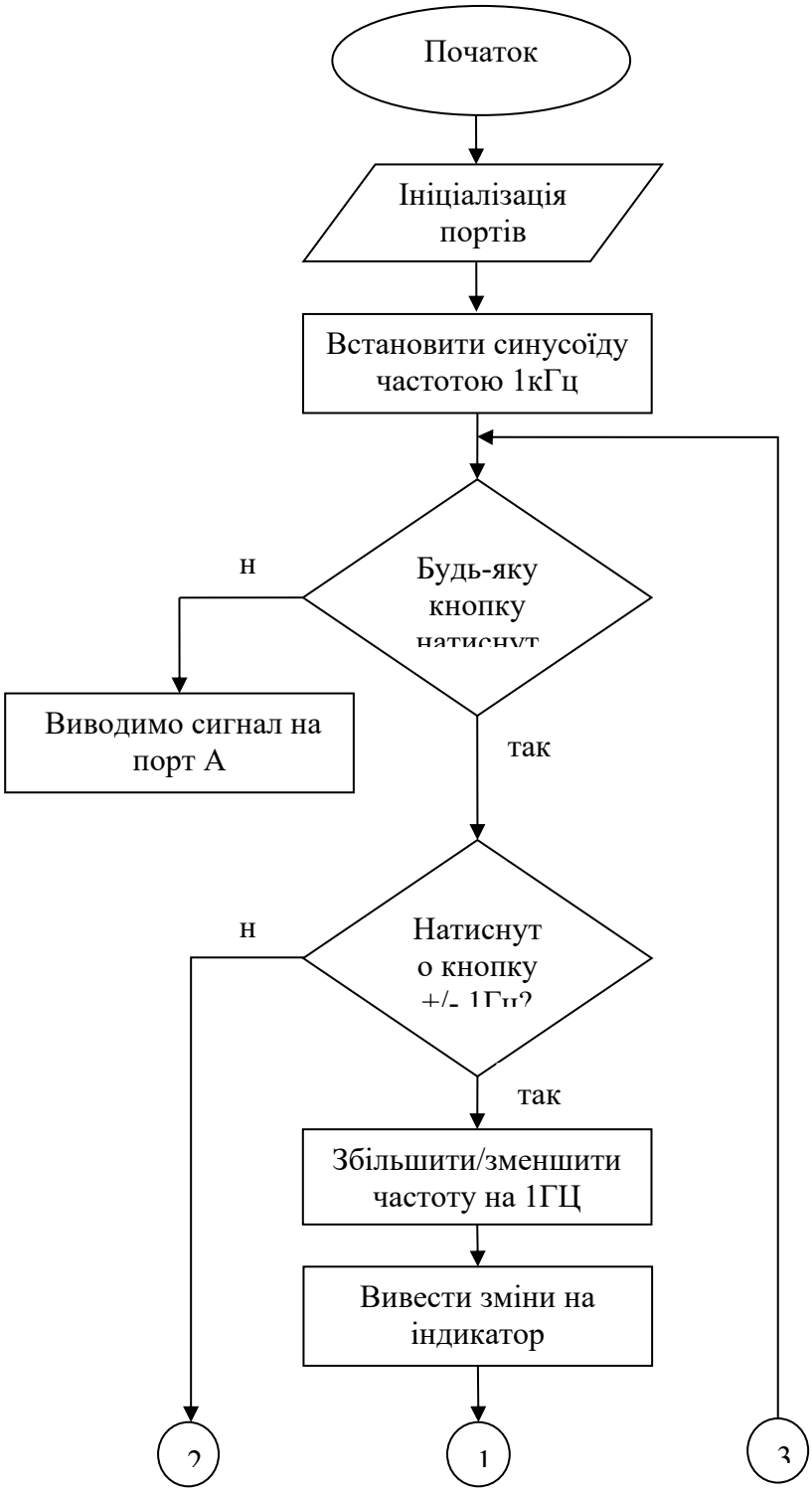
Ще одним важливим елементом є кварцовий резонатор, який забезпечує стабільну тактову частоту. У даній схемі використовується резонатор на 16 МГц. Він відповідає за формування системної частоти для мікроконтролера. Для його коректної роботи необхідно підключити конденсатори ємністю 20–30 пФ. Існують також резонатори з вбудованими конденсаторами, що полегшує монтаж.

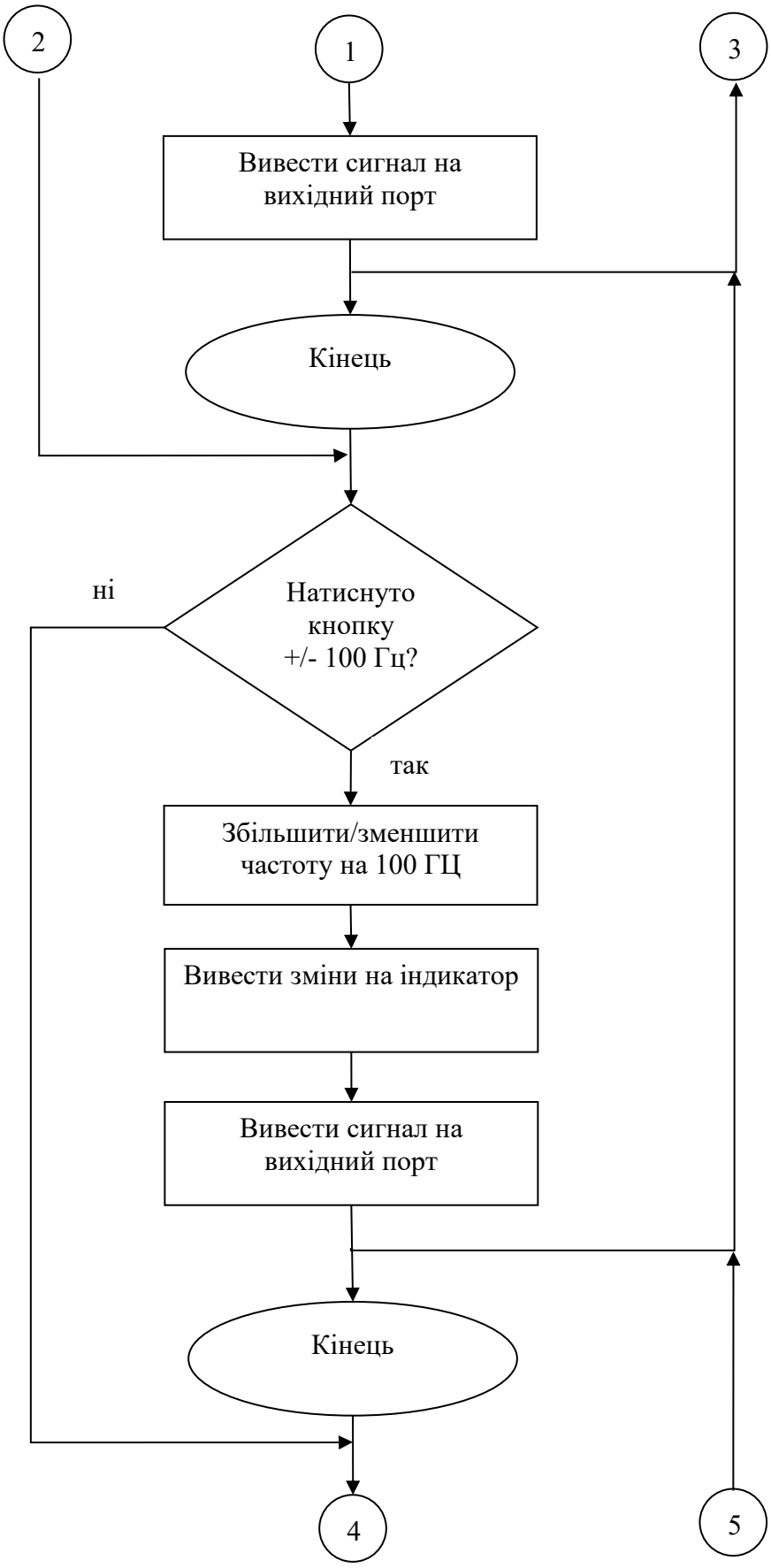
Принципову електричну схему наведено в додатку.

3.3 Алгоритмічна побудова роботи системи

При ввімкненні живлення схеми, лінії PC0-PD7 настроюються на вивід, режиму роботи на індикатор. Потім встановлюється форма сигналу і довільне значення частоти. Після чого починається генерація сигналу. Паралельно з генерацією вихідного сигналу відбувається постійна перевірка портів до яких підключена система керування. Якщо програма визначає наявність такого сигналу, вона ідентифікує натиснуту клавішу і корегує вихідний сигнал відповідно до налаштувань які прийшли на порт D. При натисненні клавіш управління вихідний

сигнал не перестає генеруватися мікроконтролером. Мікрокотролер постійно опитує порт С це дає змогу швидко відреагувати на введені зміни також відбувається корекція зображення на індикаторі яке відповідає реальному значенню вихідного сигналу. Блок-схему алогитрму показано на рисунку 3.7.




2
1
3
Вивести сигнал на вихідний порт
Кінець
ні
Натиснуто кнопку +/- 100 Гц?
так
Збільшити/зменшити частоту на 100 ГЦ
Вивести зміни на індикатор
Вивести сигнал на вихідний порт
Кінець
4
5

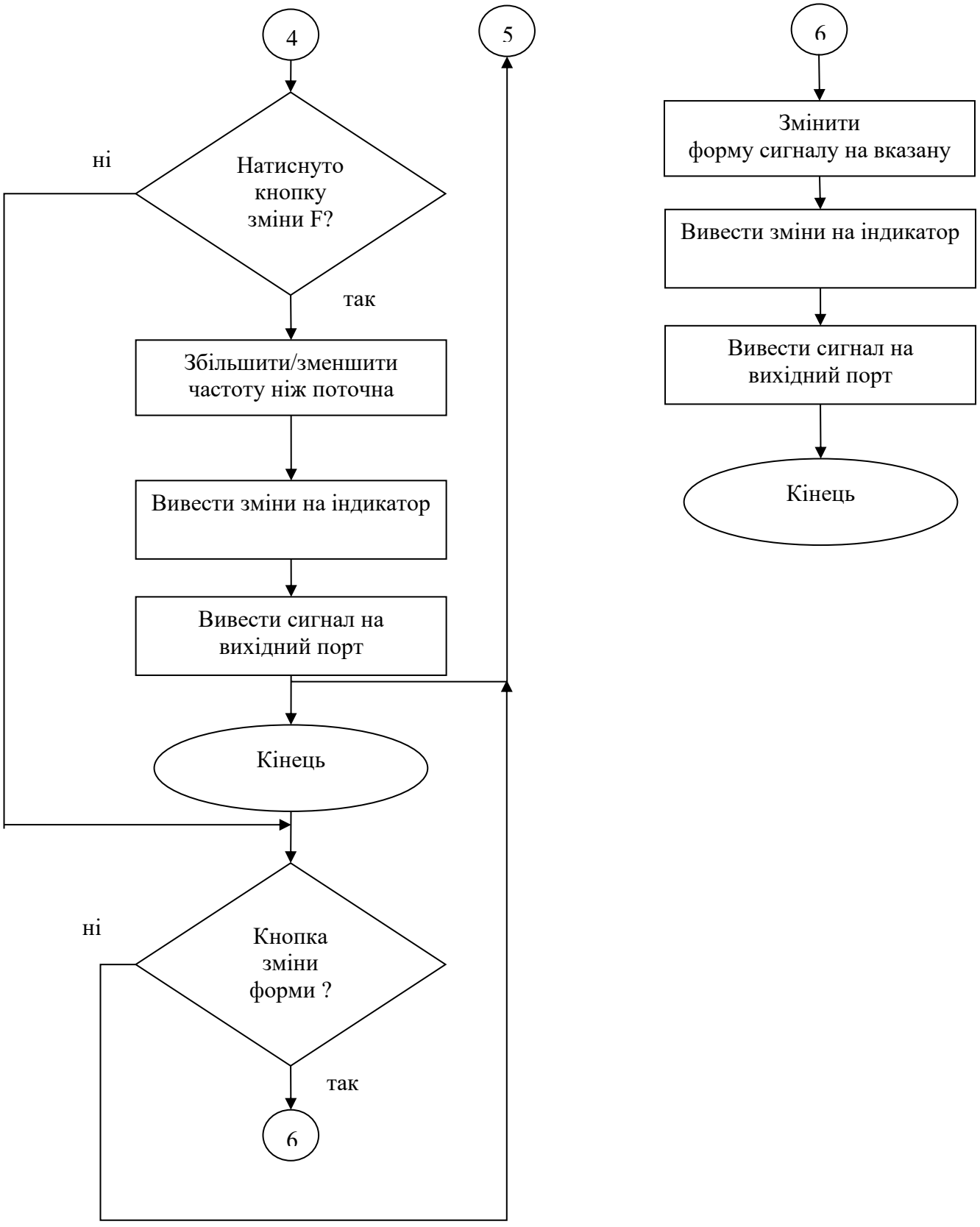


Рисунок 3.7 — Блок-схема алгоритму роботи

3.4 Програмна реалізація

В якості платформи для розробки вибрано пакет програм WinAVR. Дана платформа розповсюджується безкоштовно з відкритим вихідним кодом. Генерує досить оптимальний код. Синтаксис мови з в ньому максимально наближений до класичного Сі. Головною програмою, в якій власне і йде робота є Programmer Notepage. Відкриваємо Programmer Notepage, «File" -> "New». Відкриється вікно нового файлу. Попередньо створимо папку в якій зберігатимуться наші проекти. У цю папку збережемо новий файл з розширенням с. Набираємо власне код. Для завдання параметрів самого проекту необхідно створити так званий makefile і починаємо вводити параметри свого проекту (рис 3.8).

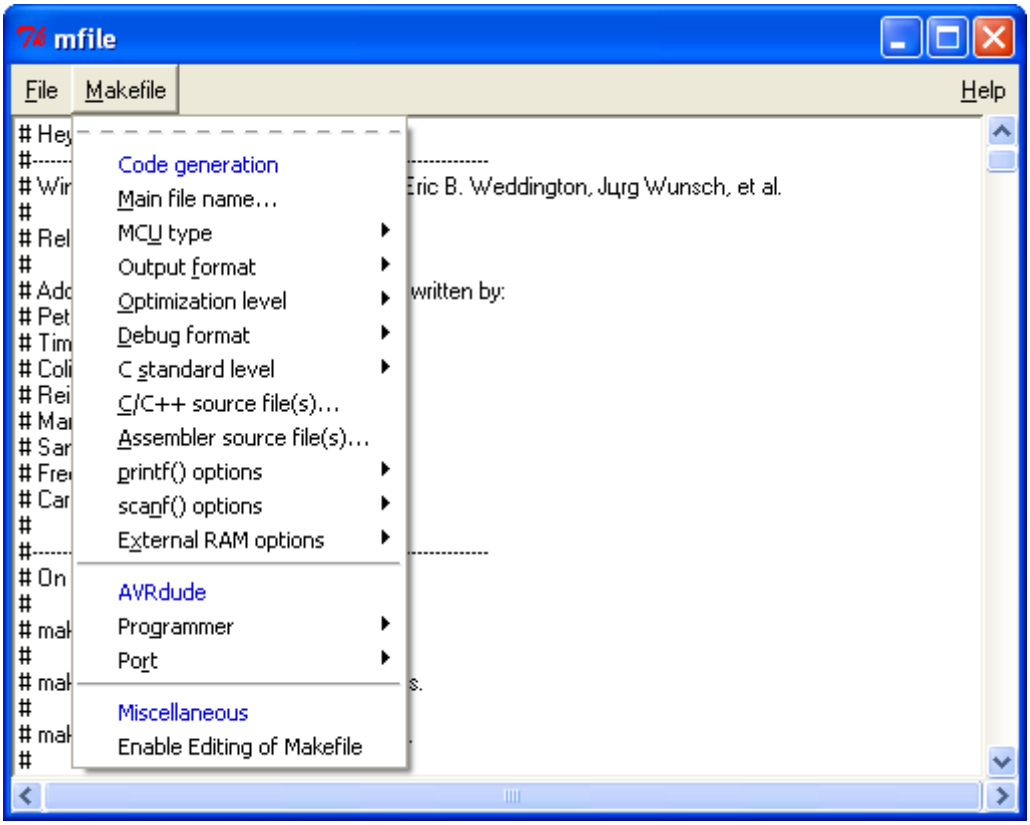


Рисунок 3.8 — Налаштування параметрів проекту

В основному там потрібно буде поміняти назву головного файлу проекту (за замовчуванням «main»), вибрати тип мікроконтролера MCU type, інше можна не чіпати, всі параметри, в тому числі і зазначені вище, можливо поміняти буде потім

в папці проекту . Потім клікаємо «File" -> "Save As» і зберігаємо його як mikefile (за замовчуванням) в папку проекту. Тепер з Programmer Notepage відкриваємо папку проекту і бачимо там чотири файлу: *.c - це основний файл проекту, **.c, **.h - файл функцій роботи з ** бібліотеками та заголовки до нього і makefile проекту, який створювали (рис. 3.9.).

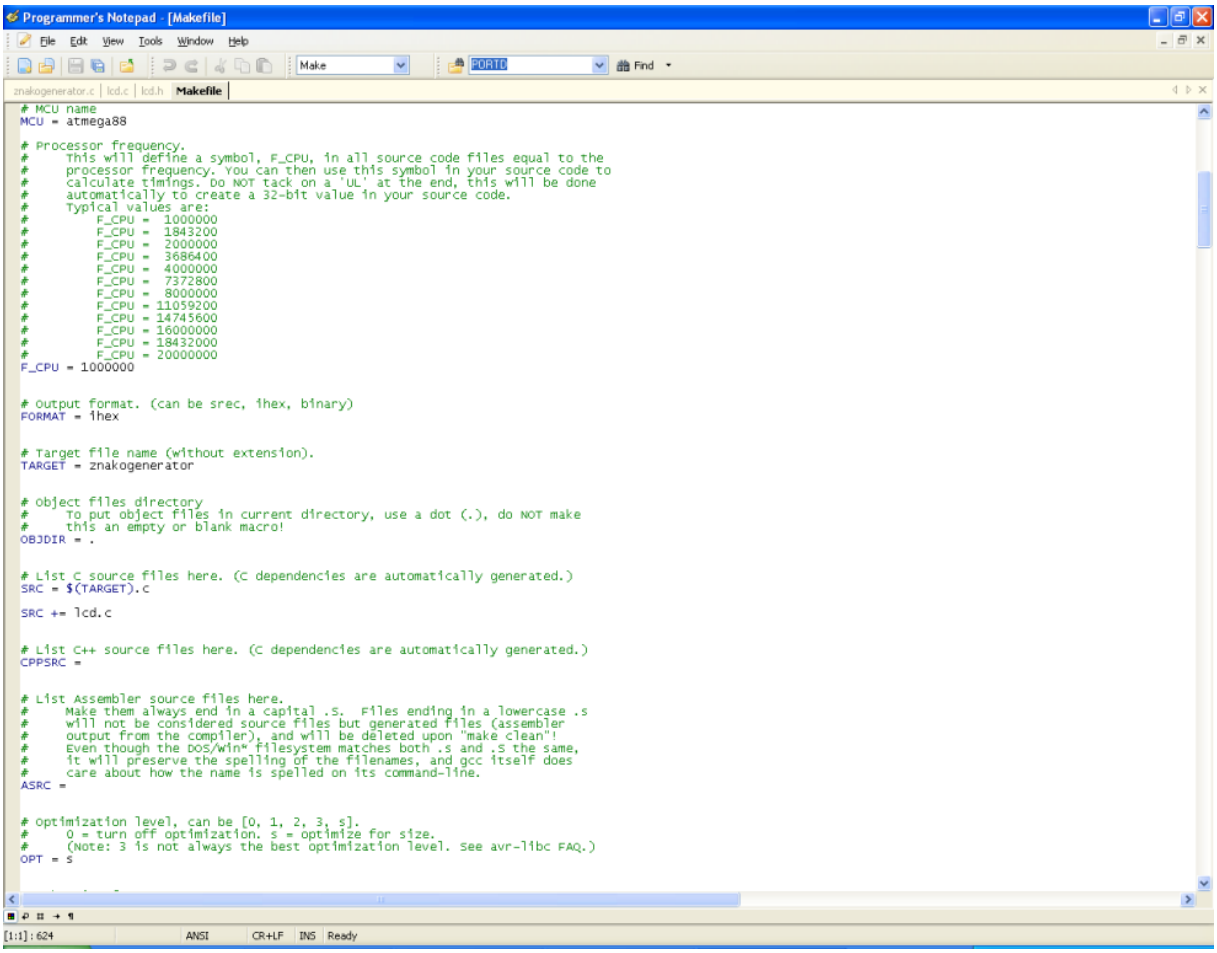


Рисунок 3.9 — Вікно Programmer Notepage

У Programmer Notepage кликаем «Tools» -> «Make Clean» - це очистить папку від попередніх результатів компіляції. Після цього «Tools» -> «Make All» - це відкомпілює проект, і, якщо не буде помилок, то вилізе вікно (рис. 3.10). Слід також зазначити, що WinAVR чудово стикується з AVR studio.

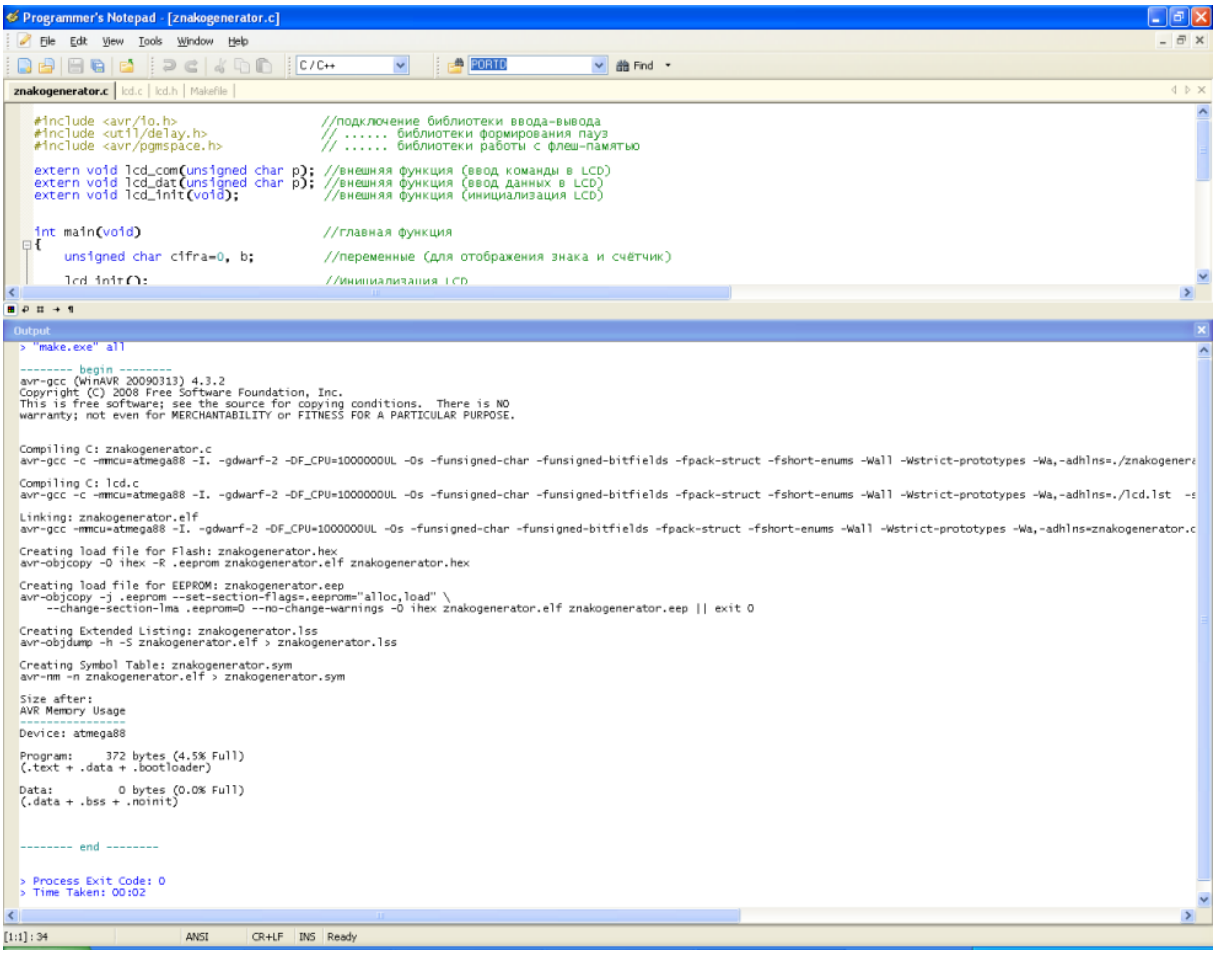


Рисунок 3.10 — Вікно компільованого проекту

Всі режими роботи і налаштування відображаються на дисплеї. Управління здійснюється за допомогою п'яти кнопок. Кнопки Up (вгору) і Down (вниз) використовуються для переміщення між пунктами меню, кнопки Left (вліво) і Right (вправо) використовуються для зміни значення частоти. Кнопка Start / Stop - запускає / зупиняє генерацію. Вид меню показано на рисунку 3.11.

Варто зауважити, що є окремий пункт меню, в якому змінюється крок перебудови частоти (Freq Step). Дана опція дозволяє змінювати значення частоти для всіх сигналів в широкому діапазоні з мінімальною кількістю натиснень кнопок і переходів по меню. При генерації шуму налаштування частоти не потрібно - використовується проста функція генерації випадкового числа, результат якої безперервно надходить на вихід DDS.

Значення частоти високочастотного сигналу можна вибрати 1 МГц, 2 МГц, 4 МГц і 8 МГц.

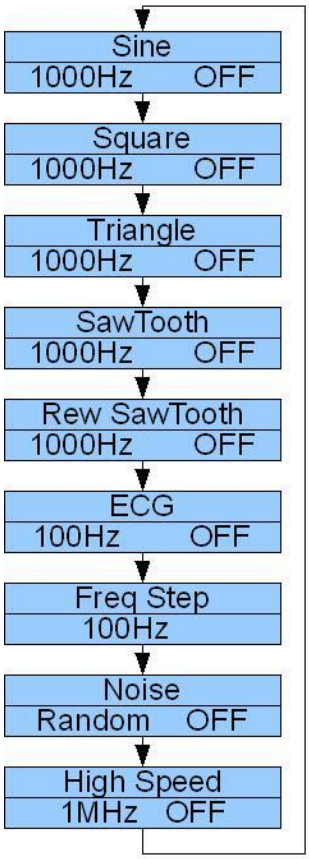


Рисунок 3.11 — Вид меню

Лістинг програмного забезпечення наведено в додатку.

3.5 Імітаційне моделювання та тестування роботи

Робота цієї схеми була перевірена у пакеті схемотехнічного аналізу – Proteus 7 Professional. Пакет Proteus складається з двох програм: ISIS – моделювання електронних схем (рис. 3.12) та ARES – програма створення друкованих плат. В програмі ISIS передбачена можливість покрокового відлагодження мікропроцесорних схем, зміна типономіналу як пасивних так і активних елементів схеми, зміна робочої тактової частоти, можливість анімації окремих елементів досліджуваної схеми. У Proteus надається можливість самостійно синтезувати

модель функціонально закінченого модуля, створивши й відлагодивши програму його схеми керування, з наступним дослідженням часових осцилограм, які показані на (рис. Г.2-Г.6)

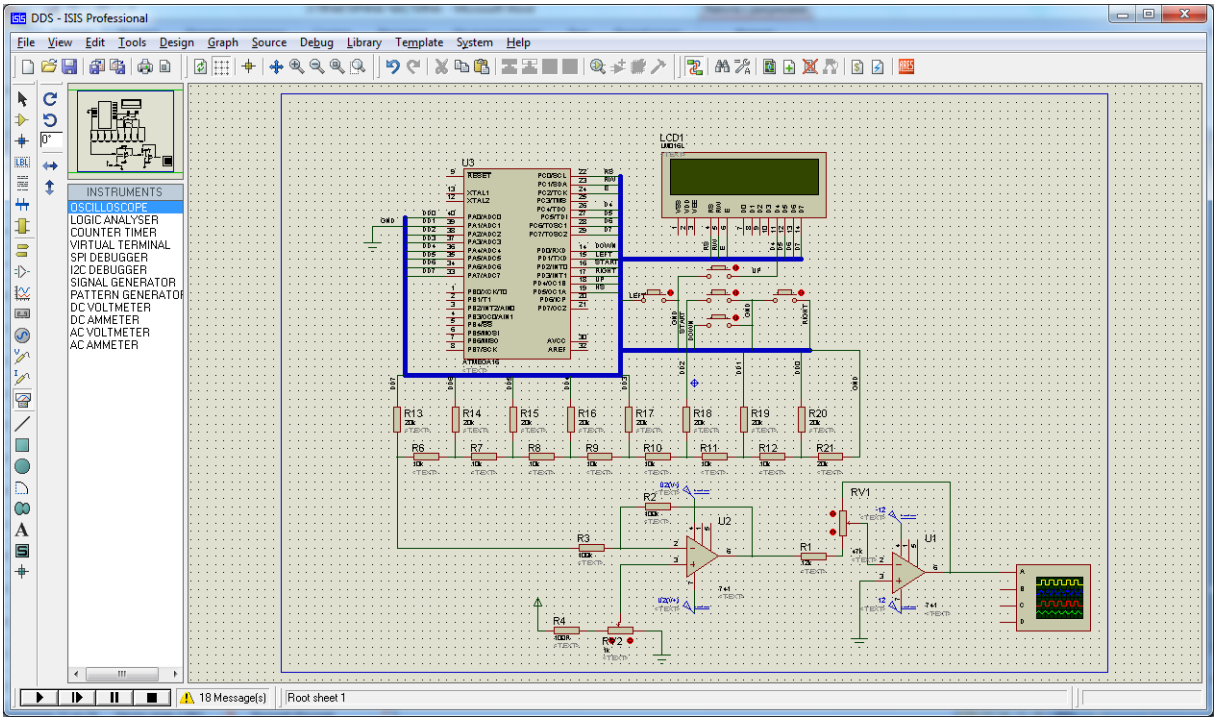


Рисунок 3.12 — Моделювання роботи генератора

3.6 Верифікація функціональності та оптимізація системи

Після проведення імітаційного моделювання наступним етапом стало підтвердження відповідності роботи пристрою попередньо визначеним технічним вимогам. Верифікація функціональності здійснювалась як шляхом спостереження за результатами моделювання, так і за допомогою аналізу вихідних сигналів на логічному аналізаторі та осцилографі.

Перш за все було перевірено коректність генерації сигналів заданих форм: синусоїди, прямокутного імпульсу, трикутника та шуму. Результати свідчили про стабільність генерації в заданому частотному діапазоні (1...8000 кГц) без суттєвих

спотворень або фазових зсувів. Вихідна частота змінювалася у відповідності до кроку, визначеного програмно — від 1 до 10000 Гц.

Окрему увагу було приділено роботі системи управління — п’ятикнопкова клавіатура коректно зчитувала натискання, а зміна параметрів оперативно відображалась на LCD-дисплеї. Після увімкнення пристрою система автоматично поверталася до останніх збережених налаштувань, що підтверджує правильну реалізацію енергонезалежної пам’яті конфігурації.

З метою покращення загальної продуктивності було проведено оптимізацію програмного коду. Зокрема, зменшено затримки в обробці подій користувача, впроваджено пріоритетне опитування клавіш, а також оптимізовано алгоритм оновлення дисплея для уникнення надлишкових викликів функцій виводу.

Крім того, було протестовано роботу системи при зміні живлення, для чого були змодельовані типові режими запуску з відключенням/включенням живлення. Пристрій стабільно відновлював роботу та не допускав збоїв у генерації сигналів.

Таким чином, результати верифікації засвідчили, що розроблена мікропроцесорна система генерації сигналів повністю відповідає технічному завданню. Після незначної оптимізації система набула необхідної стабільності, керованості та точності для подальшого переходу до етапу виготовлення фізичного прототипу.

Для подальшого покращення роботи системи передбачено проведення повноцінних стендових випробувань у реальних умовах експлуатації. Це дозволить оцінити вплив зовнішніх факторів, таких як температурні коливання, електромагнітні завади, а також перевірити довготривалу стабільність роботи.

Крім того, заплановано дослідження енергоспоживання системи з метою оптимізації живлення та підвищення енергоефективності. Це особливо важливо для портативних або автономних пристроїв, де ресурс акумулятора є обмеженим.

Для підвищення функціональності перспективним напрямом є інтеграція системи з зовнішніми пристроями через стандартні інтерфейси (UART, SPI, I2C), що розширить можливості управління та моніторингу.

Можливість використання більш потужних мікроконтролерів або модулів з вбудованими ЦАП, що може суттєво спростити апаратну частину та зменшити розміри пристрою.

Реалізація цих напрямів дозволить створити універсальний, надійний та зручний у використанні генератор сигналів, який відповідатиме сучасним вимогам та завданням, що постають перед електронними системами.

ВИСНОВКИ

В результаті виконаної роботи була розроблена мікропроцесорна система генерування сигналів, яка забезпечує гнучке формування різних типів сигналів із можливістю налаштування основних параметрів — частоти, амплітуди та форми. Обрана технічна база, включно з мікроконтролером, цифро-аналоговим перетворювачем та елементами керування, дозволила створити просту та надійну апаратну реалізацію системи з доступними комплектуючими.

Розроблена структурна та принципова схема забезпечують ефективну взаємодію між мікроконтролером, блоком керування, індикації та перетворення цифрового сигналу в аналоговий. Вибір ЦАП типу R-2R підтвердив свою доцільність у контексті співвідношення точності, простоти виготовлення та вартості.

Алгоритмічна побудова роботи системи та програмна реалізація дали змогу забезпечити необхідну функціональність із можливістю швидкого налаштування і відновлення параметрів при включенні. Проведене тестування й імітаційне моделювання підтвердили правильність обраної архітектури та стабільність роботи пристрою.

Отже, створена система є універсальним та ефективним інструментом для генерації сигналів у широкому спектрі застосувань, що відкриває перспективи її подальшого удосконалення, розширення функціоналу та інтеграції в автоматизовані електронні системи.

Отримані результати можуть бути використані як база для подальших наукових досліджень і практичних розробок у сфері цифрової генерації сигналів, а також сприяти впровадженню сучасних мікропроцесорних технологій у промисловості та наукових дослідженнях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Абрамович А.П. Цифрова електроніка: підручник / А.П. Абрамович. — К.: Видавничий дім «Освіта», 2018. — 432 с. — (Серія «Вища освіта»).

2. Білий В.І. Мікропроцесорні системи: навчальний посібник / В.І. Білий. — Львів: Львівська політехніка, 2019. — 256 с.

3. Гринь О.М. Методи цифрової обробки сигналів / О.М. Гринь. — Харків: НТУ «ХПІ», 2020. — 312 с.

4. Коваленко І.В. Сучасні методи синтезу сигналів / І.В. Коваленко // Вісник КНУТД. — 2021. — Вип. 2. — С. 45–52.

5. Петренко С.О. Програмування мікроконтролерів на С / С.О. Петренко. — К.: ТОВ «Видавництво», 2017. — 384 с.

6. Цифровий синтезатор сигналів DDS: теорія і практика / під ред. І.В. Грищука. — К.: Наукова думка, 2016. — 280 с.

7. Мікроконтролери STM32: посібник користувача / STMicroelectronics. — [Електронний ресурс]. — Режим доступу: https://www.st.com/resource/en/user_manual/dm00105823-stm32-mcu-reference-manual.pdf (дата звернення: 13.05.2025).

8. DS18B20 Digital Thermometer Datasheet / Maxim Integrated. — [Електронний ресурс]. — Режим доступу: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf (дата звернення: 13.05.2025).

9. ГОСТ 2.701-2011. Електричні схеми. Правила виконання. — Київ: Держспоживстандарт України, 2011.

10. ДСТУ IEC 61131-3:2015 Програмовані логічні контролери. Частина 3. Мови програмування. — Київ: Мінекономрозвитку, 2015.

11. Лі С.Е. Аналіз і моделювання цифрових систем / С.Е. Лі. — Х.: Фоліо, 2020. — 288 с.

12. Шевченко Ю.В. Основи цифрової обробки сигналів / Ю.В. Шевченко. — Львів: Видавництво ЛНУ, 2019. — 320 с.

13. Direct Digital Synthesis (DDS) Fundamentals / Analog Devices. — [Електронний ресурс]. — Режим доступу: https://www.analog.com/en/analog-dialogue/articles/direct-digital-synthesis-dds.html (дата звернення: 13.05.2025).

14. Мазуренко В.М. Електронні компоненти та їх застосування / В.М. Мазуренко. — К.: Техніка, 2018. — 224 с.

15. Пасічник І.Г. Програмне забезпечення вбудованих систем / І.Г. Пасічник. — К.: Видавництво КНУ, 2017. — 260 с.

16. Smith S.W. Digital Signal Processing: A Practical Guide for Engineers and Scientists / S.W. Smith. — Newnes, 2013. — 760 p.

17. Embedded Systems Design: An Introduction to Processes, Tools, and Techniques / Arnold S. Berger. — CMP Books, 2002. — 448 p.

18. IEC 61000-4-2:2008 Electromagnetic compatibility (EMC). Testing and measurement techniques — Electrostatic discharge immunity test. — [Електронний ресурс]. — Режим доступу: https://webstore.iec.ch/publication/2765 (дата звернення: 13.05.2025).

19. ARM Cortex-M4 Processor Technical Reference Manual / ARM Ltd. — [Електронний ресурс]. — Режим доступу: https://developer.arm.com/documentation/100166/0001/ (дата звернення: 13.05.2025).

20. Мікросхема ЦАП MCP4921 Datasheet / Microchip Technology Inc. — [Електронний ресурс]. — Режим доступу: https://ww1.microchip.com/downloads/en/DeviceDoc/21298e.pdf (дата звернення: 13.05.2025).

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 генерування сигналів»

08-54.БКР.003.00.000 ТЗ

Науковий керівник к.т.н., доц. каф. ОТ

_____ Савицька Л.А.

Студент групи 1КІ-21б

_____ БорисоглебськийА.П

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— створення мікропроцесорної системи генерації сигналів на основі DDS-методу з можливістю вибору форми та частоти вихідного сигналу за допомогою інтерфейсу користувача.

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 генерації сигналів.

4. Вимоги до виконання БКР

БКР повинна задовольняти такі вимоги:
— аналіз сучасних підходів до реалізації цифрових генераторів сигналів;
— здійснення вибору відповідного мікроконтролера та допоміжних компонентів;
— розробка принципової схеми пристрою;
— створення алгоритму роботи системи та реалізувати його програмно;
— забезпечення відображення параметрів сигналу на індикаторі;
— тестування роботи пристрою та оцінити точність та стабільність генерації сигналів.

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	Теоретичні основи побудови систем на мікроконтролерах	01.04-21.04.25	
4	Розробка електричної принципової схеми	22.04-05.05.25	
5	Програмна реалізація	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 генерування сигналів

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 35 %

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 _____ Борисоглебський А. П.

(підпис) (прізвище,ініціали)

Керівникроботи _____ Савицька Л.А., доц. каф. ОТ

(підпис) (прізвище,ініціали, посада)

ДОДАТОК В

Графічна частина

УЗАГАЛЬНЕНА СТРУКТУРНА СХЕМА МІКРОКОНТРОЛЕРА ATmega16

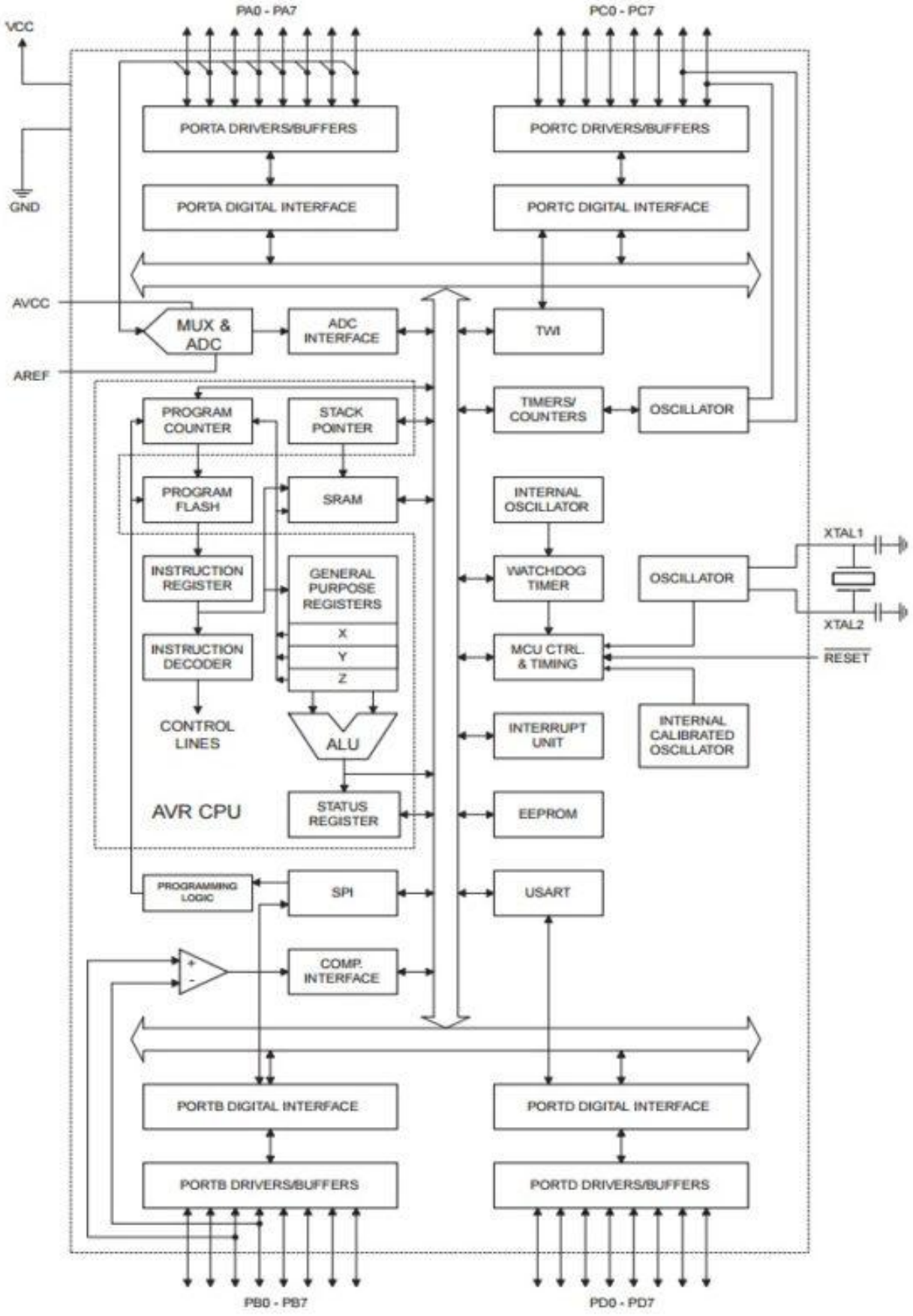


Рисунок В.1 — Узагальнена структурна схема мікроконтролера ATmega16

ДОДАТОК Г

Ілюстративна частина

МОДЕЛЮВАННЯ РОБОТИ ГЕНЕРАТОРА

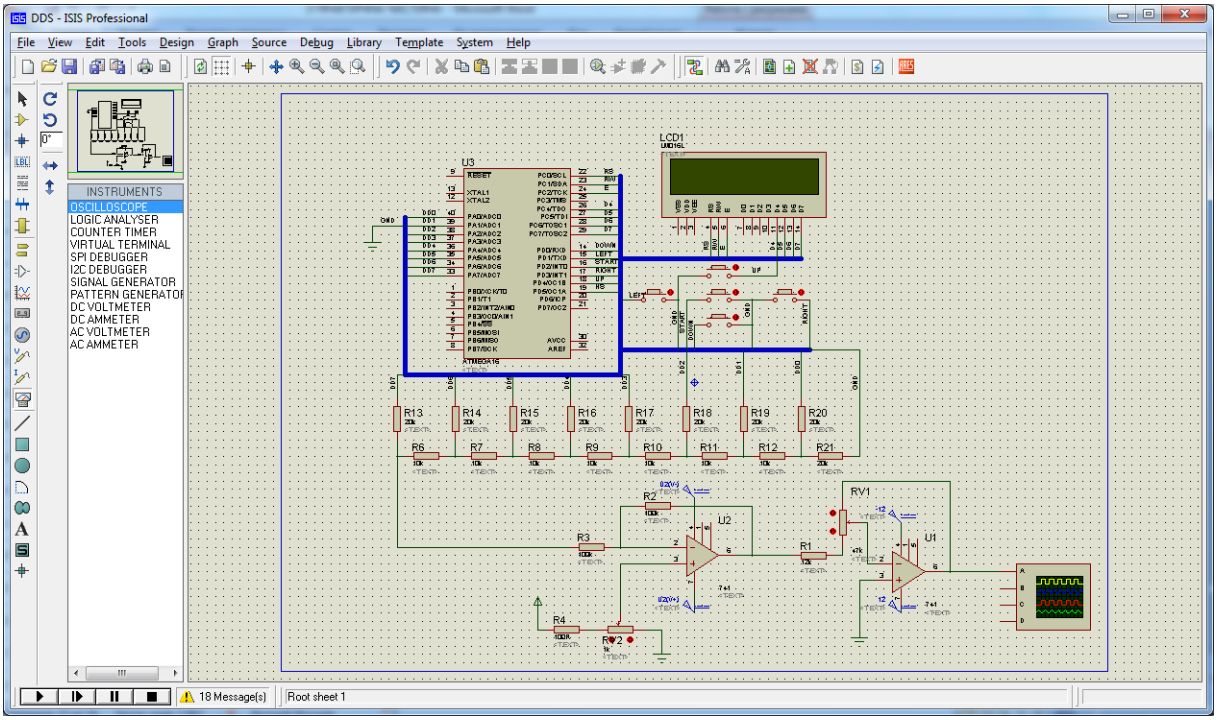


Рисунок Г.1 — Моделювання роботи генератора

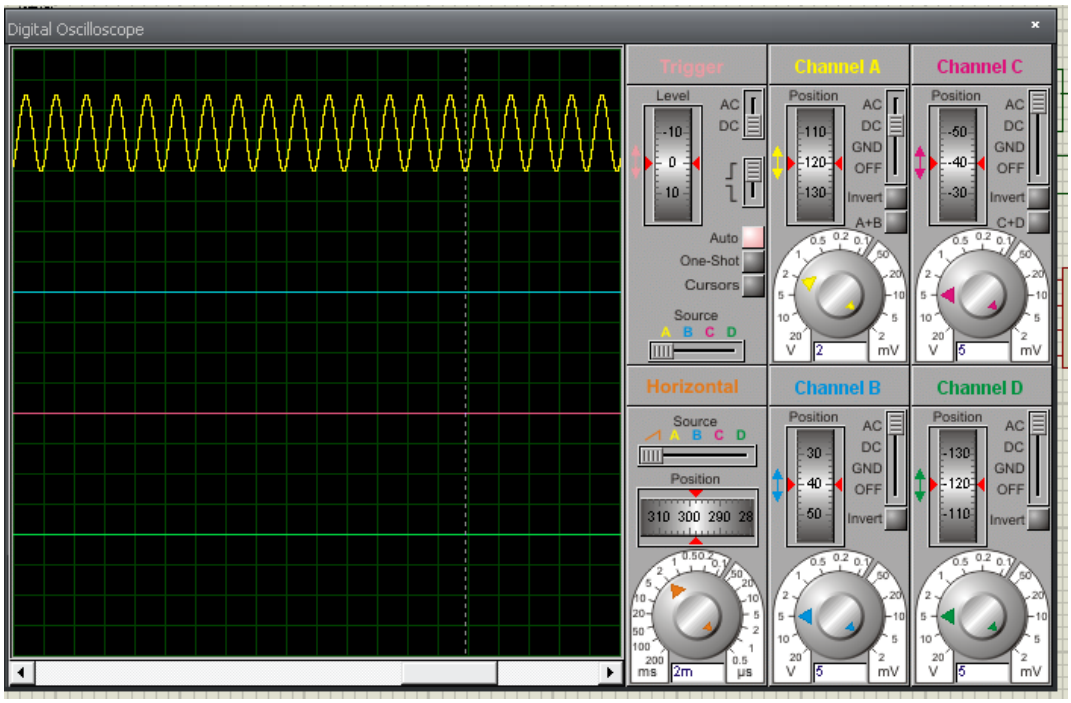


Рисунок Г.2 — Генерація синусоїдального сигналу

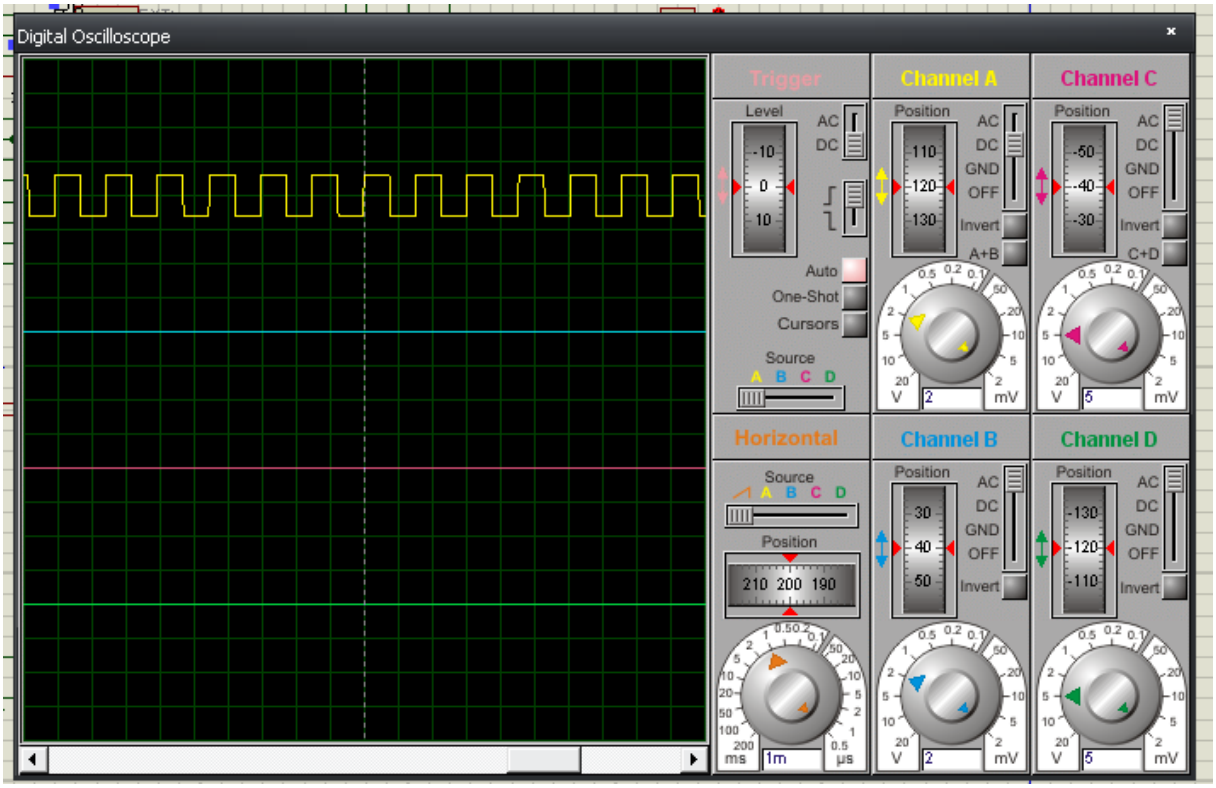


Рисунок Г.3 — Генерація прямокутних імпульсів

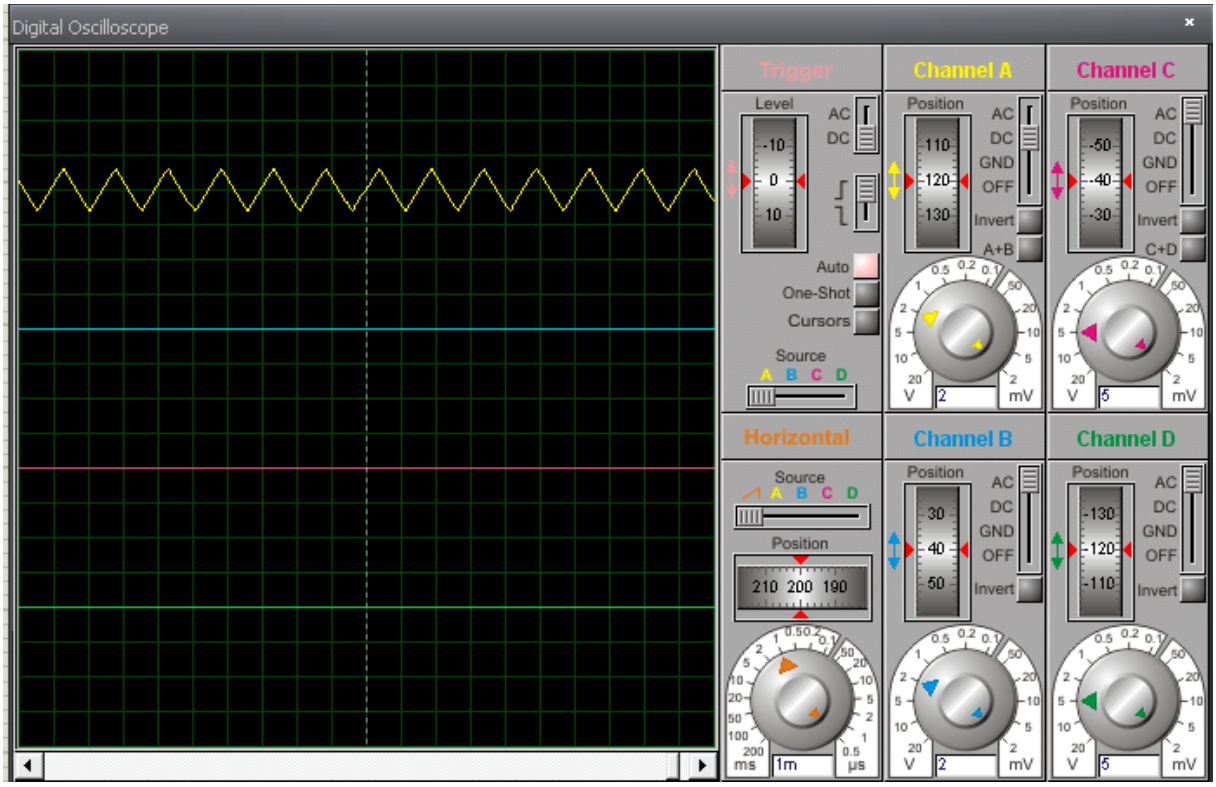


Рисунок Г.4 — Генерація пилкоподібного сигналу

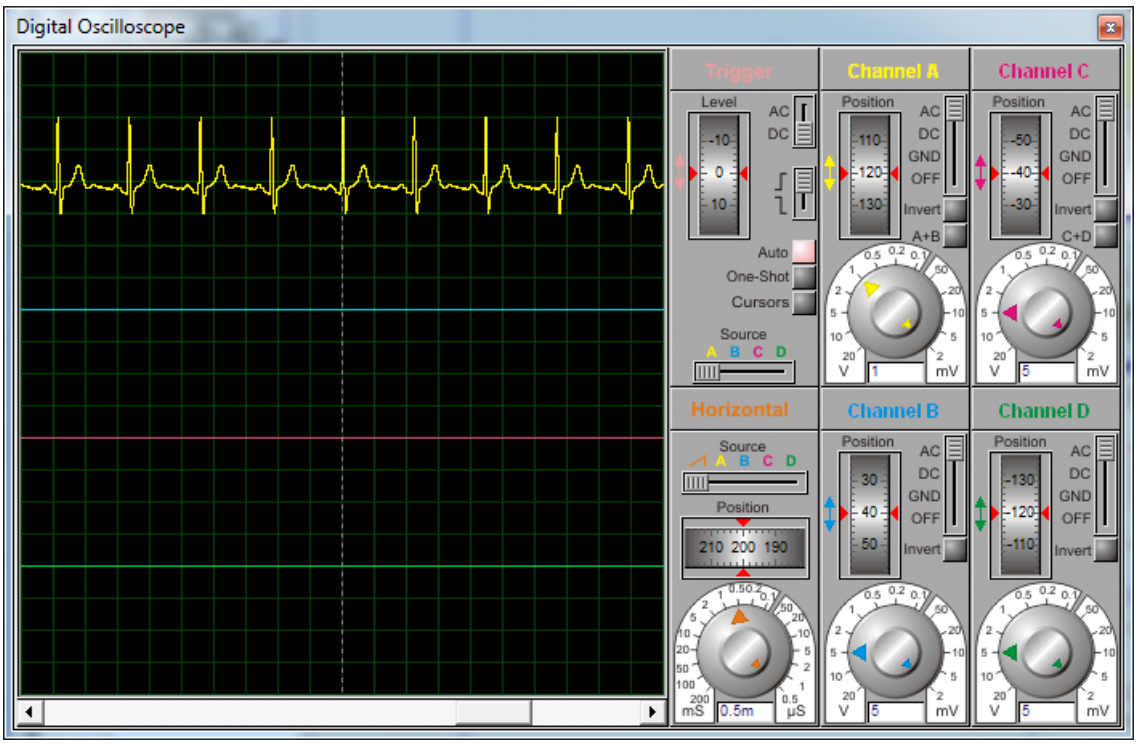


Рисунок Г.5 — Генерація електрокардіограми

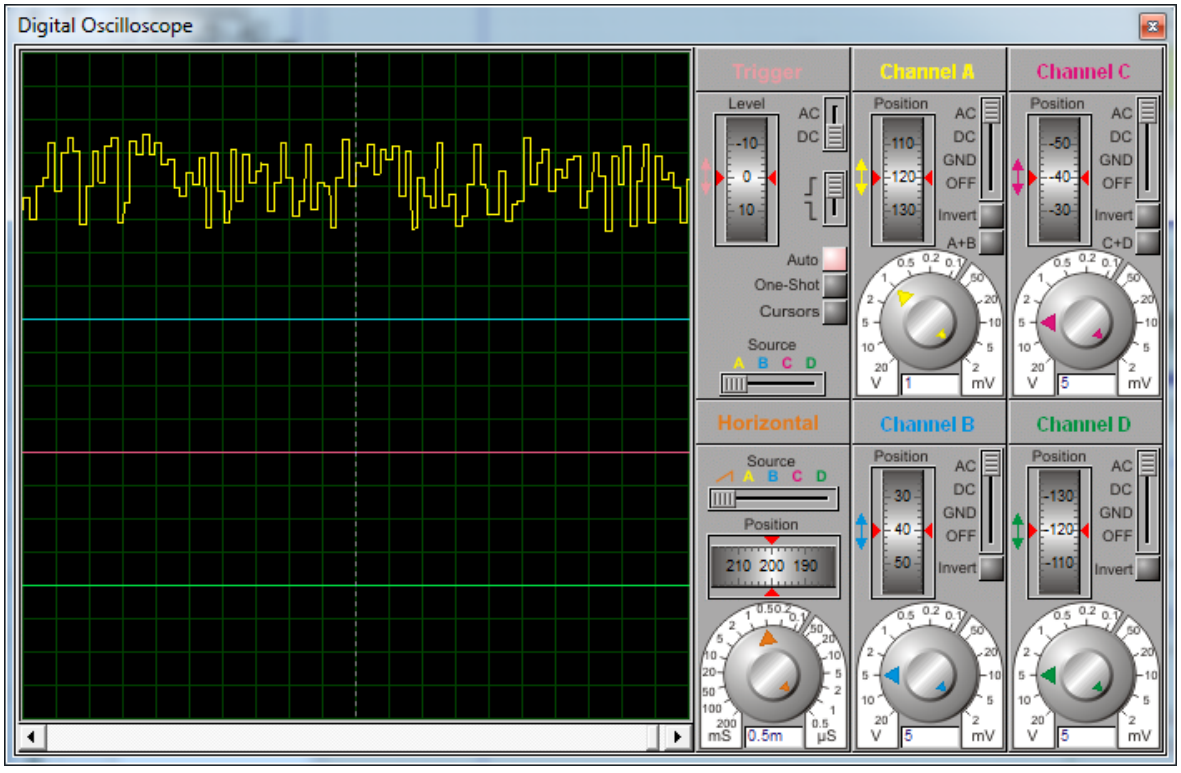


Рисунок Г.6 — Генерація шуму

ДОДАТОК Д

Лістинг основного коду

```
#include <stdio.h>
#include <stdlib.h>
#include <avr/io.h>
#include <avr/pgmspace.h>
#include <avr/eeprom.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <inttypes.h>
#include "lcd_lib.h"
//define R2R port
#define R2RPORT PORTA
#define R2RDDR DDRA
//define button port and dedicated pins
#define BPORT PORTD
#define BPIN PIND
#define BDDR DDRD
#define DOWN 0//PORTD
#define LEFT 1//PORTD
#define START  2//PORTD
#define RIGHT 3//PORTD
#define UP 4//PORTD
//Define Highs Speed (HS) signal output
#define HSDDR DDRD
#define HSPORT PORTD
#define HS 5//PD5
//define eeprom addresses
#define EEMODE 0
#define EEFREQ1 1
#define EEFREQ2 2
#define EEFREQ3 3
#define EEDUTY 4
#define EEINIT E2END
#define RESOLUTION 0.095367431640625
#define MINFREQ 0//minimum frequency
#define MAXFREQ 65534//maximum DDS frequency
#define MN_No 9// number of menu items
//function prototypes
void delay1s(void);
void Timer2_Init(void);
```

```
void Timer2_Start(void);
void Timer2_Stop(void);
void Main_Init(void);
void Menu_Update(uint8_t);
void Freq_Update(void);
void Timer1_Start(uint8_t);
void Timer1_Stop(void);
void static inline Signal_OUT(const uint8_t *, uint8_t, uint8_t, uint8_t);
//adjust LCDsendChar() function for strema
static int LCDsendstream(char c, FILE *stream);
//----set output stream to LCD-------
static FILE lcd_str = FDEV_SETUP_STREAM(LCDsendstream, NULL,
_FDEV_SETUP_WRITE);

//Menu Strings in flash
const uint8_t MN000[] PROGMEM="      Sine      \0";
//menu 1
const uint8_t MN100[] PROGMEM="     Square     \0";
//menu 2
const uint8_t MN200[] PROGMEM="    Triangle    \0";
//menu 3
const uint8_t MN300[] PROGMEM="    SawTooth    \0";
//menu 4
const uint8_t MN400[] PROGMEM="  Rev SawTooth  \0";
//menu 5
const uint8_t MN500[] PROGMEM="      ECG       \0";
//menu 6
const uint8_t MN600[] PROGMEM="    Freq Step   \0";
//menu 7
const uint8_t MN700[] PROGMEM="     Noise      \0";
//menu 8
const uint8_t MN800[] PROGMEM="   High Speed   \0";

//Array of pointers to menu strings stored in flash
const uint8_t *MENU[]={
                MN000,      //
                MN100,      //menu 1 string
                MN200,      //menu 2 string
                MN300,      //menu 3 string
                MN400,      //menu 4 string
                MN500,
                MN600,
                MN700,
                MN800
                };
```

```
//various LCD strings
const uint8_t MNON[] PROGMEM="ON \0";//ON
const uint8_t MNOFF[] PROGMEM="OFF\0";//OFF
const uint8_t NA[] PROGMEM="       NA       \0";//Clear freq value
const uint8_t CLR[] PROGMEM="                \0";//Clear freq value
const uint8_t MNClrfreq[] PROGMEM="          \0";//Clear freq value
const uint8_t TOEEPROM[] PROGMEM="Saving Settings\0";//saving to eeprom
const uint8_t ONEMHZ[] PROGMEM="      1MHz   \0";//saving to eeprom
const uint8_t welcomeln1[] PROGMEM="AVR SIGNAL\0";
const uint8_t RND[] PROGMEM="    Random\0";

//variables to control TDA7313
struct signal{
volatile uint8_t mode;              //signal
volatile uint8_t fr1;               //Frequency [0..7]
volatile uint8_t fr2;               //Frequency [8..15]
volatile int8_t fr3;          //Frequency [16..31]
volatile uint32_t freq;             //frequency value
volatile uint8_t flag;              //if "0"generator is OFF, "1" - ON
volatile uint32_t acc;              //accumulator
volatile uint8_t ON;
volatile uint8_t HSfreq;            //high speed frequency [1...4Mhz]
volatile uint32_t deltafreq;        //frequency step value
}SG;

//define signals
const uint8_t  sinewave[] __attribute__ ((section (".MySection1")))= //sine 256 values
{
0x80,0x83,0x86,0x89,0x8c,0x8f,0x92,0x95,0x98,0x9c,0x9f,0xa2,0xa5,0xa8,0xab,0xae,
0xb0,0xb3,0xb6,0xb9,0xbc,0xbf,0xc1,0xc4,0xc7,0xc9,0xcc,0xce,0xd1,0xd3,0xd5,0xd8,
0xda,0xdc,0xde,0xe0,0xe2,0xe4,0xe6,0xe8,0xea,0xec,0xed,0xef,0xf0,0xf2,0xf3,0xf5,
0xf6,0xf7,0xf8,0xf9,0xfa,0xfb,0xfc,0xfc,0xfd,0xfe,0xfe,0xff,0xff,0xff,0xff,0xff,
0xff,0xff,0xff,0xff,0xff,0xff,0xfe,0xfe,0xfd,0xfc,0xfc,0xfb,0xfa,0xf9,0xf8,0xf7,
0xf6,0xf5,0xf3,0xf2,0xf0,0xef,0xed,0xec,0xea,0xe8,0xe6,0xe4,0xe2,0xe0,0xde,0xdc,
0xda,0xd8,0xd5,0xd3,0xd1,0xce,0xcc,0xc9,0xc7,0xc4,0xc1,0xbf,0xbc,0xb9,0xb6,0xb3,
0xb0,0xae,0xab,0xa8,0xa5,0xa2,0x9f,0x9c,0x98,0x95,0x92,0x8f,0x8c,0x89,0x86,0x83,
0x80,0x7c,0x79,0x76,0x73,0x70,0x6d,0x6a,0x67,0x63,0x60,0x5d,0x5a,0x57,0x54,0x5
1,
0x4f,0x4c,0x49,0x46,0x43,0x40,0x3e,0x3b,0x38,0x36,0x33,0x31,0x2e,0x2c,0x2a,0x27
,
0x25,0x23,0x21,0x1f,0x1d,0x1b,0x19,0x17,0x15,0x13,0x12,0x10,0x0f,0x0d,0x0c,0x0a
,
0x09,0x08,0x07,0x06,0x05,0x04,0x03,0x03,0x02,0x01,0x01,0x00,0x00,0x00,0x00,0x0
0,
```

```
0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x01,0x02,0x03,0x03,0x04,0x05,0x06,0x07,0x08,
0x09,0x0a,0x0c,0x0d,0x0f,0x10,0x12,0x13,0x15,0x17,0x19,0x1b,0x1d,0x1f,0x21,0x23,
0x25,0x27,0x2a,0x2c,0x2e,0x31,0x33,0x36,0x38,0x3b,0x3e,0x40,0x43,0x46,0x49,0x4c,
0x4f,0x51,0x54,0x57,0x5a,0x5d,0x60,0x63,0x67,0x6a,0x6d,0x70,0x73,0x76,0x79,0x7c
};
const uint8_t squarewave[] __attribute__ ((section (".MySection2")))= //square wave
{
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,
0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,
0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,
0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,
0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,
0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,
0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,
0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,
};
const uint8_t sawtoothwave[] __attribute__ ((section (".MySection3")))= //sawtooth
wave
{
0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0a,0x0b,0x0c,0x0d,0x0e,0x0f,
0x10,0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1a,0x1b,0x1c,0x1d,0x1e,0x1f,
0x20,0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,
```

```
0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x3a,0x3b,0x3c,0x3d,0x3e,0x3f
,
0x40,0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x4a,0x4b,0x4c,0x4d,0x4e,0x4f
,
0x50,0x51,0x52,0x53,0x54,0x55,0x56,0x57,0x58,0x59,0x5a,0x5b,0x5c,0x5d,0x5e,0x5f
,
0x60,0x61,0x62,0x63,0x64,0x65,0x66,0x67,0x68,0x69,0x6a,0x6b,0x6c,0x6d,0x6e,0x6f
,
0x70,0x71,0x72,0x73,0x74,0x75,0x76,0x77,0x78,0x79,0x7a,0x7b,0x7c,0x7d,0x7e,0x7f
,
0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8a,0x8b,0x8c,0x8d,0x8e,0x8f
,
0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x9a,0x9b,0x9c,0x9d,0x9e,0x9f
,
0xa0,0xa1,0xa2,0xa3,0xa4,0xa5,0xa6,0xa7,0xa8,0xa9,0xaa,0xab,0xac,0xad,0xae,0xaf,
0xb0,0xb1,0xb2,0xb3,0xb4,0xb5,0xb6,0xb7,0xb8,0xb9,0xba,0xbb,0xbc,0xbd,0xbe,0xbf
,
0xc0,0xc1,0xc2,0xc3,0xc4,0xc5,0xc6,0xc7,0xc8,0xc9,0xca,0xcb,0xcc,0xcd,0xce,0xcf,
0xd0,0xd1,0xd2,0xd3,0xd4,0xd5,0xd6,0xd7,0xd8,0xd9,0xda,0xdb,0xdc,0xdd,0xde,0xdf
,
0xe0,0xe1,0xe2,0xe3,0xe4,0xe5,0xe6,0xe7,0xe8,0xe9,0xea,0xeb,0xec,0xed,0xee,0xef,
0xf0,0xf1,0xf2,0xf3,0xf4,0xf5,0xf6,0xf7,0xf8,0xf9,0xfa,0xfb,0xfc,0xfd,0xfe,0xff
};
const uint8_t rewsawtoothwave[] __attribute__ ((section (".MySection4")))= //reverse
sawtooth wave
{
0xff,0xfe,0xfd,0xfc,0xfb,0xfa,0xf9,0xf8,0xf7,0xf6,0xf5,0xf4,0xf3,0xf2,0xf1,0xf0,
0xef,0xee,0xed,0xec,0xeb,0xea,0xe9,0xe8,0xe7,0xe6,0xe5,0xe4,0xe3,0xe2,0xe1,0xe0,
0xdf,0xde,0xdd,0xdc,0xdb,0xda,0xd9,0xd8,0xd7,0xd6,0xd5,0xd4,0xd3,0xd2,0xd1,0xd0
,
0xcf,0xce,0xcd,0xcc,0xcb,0xca,0xc9,0xc8,0xc7,0xc6,0xc5,0xc4,0xc3,0xc2,0xc1,0xc0,
0xbf,0xbe,0xbd,0xbc,0xbb,0xba,0xb9,0xb8,0xb7,0xb6,0xb5,0xb4,0xb3,0xb2,0xb1,0xb0
,
0xaf,0xae,0xad,0xac,0xab,0xaa,0xa9,0xa8,0xa7,0xa6,0xa5,0xa4,0xa3,0xa2,0xa1,0xa0,
0x9f,0x9e,0x9d,0x9c,0x9b,0x9a,0x99,0x98,0x97,0x96,0x95,0x94,0x93,0x92,0x91,0x90
,
0x8f,0x8e,0x8d,0x8c,0x8b,0x8a,0x89,0x88,0x87,0x86,0x85,0x84,0x83,0x82,0x81,0x80
,
0x7f,0x7e,0x7d,0x7c,0x7b,0x7a,0x79,0x78,0x77,0x76,0x75,0x74,0x73,0x72,0x71,0x70
,
0x6f,0x6e,0x6d,0x6c,0x6b,0x6a,0x69,0x68,0x67,0x66,0x65,0x64,0x63,0x62,0x61,0x60
,
0x5f,0x5e,0x5d,0x5c,0x5b,0x5a,0x59,0x58,0x57,0x56,0x55,0x54,0x53,0x52,0x51,0x50
,
```

```
0x4f,0x4e,0x4d,0x4c,0x4b,0x4a,0x49,0x48,0x47,0x46,0x45,0x44,0x43,0x42,0x41,0x40
,
0x3f,0x3e,0x3d,0x3c,0x3b,0x3a,0x39,0x38,0x37,0x36,0x35,0x34,0x33,0x32,0x31,0x30
,
0x2f,0x2e,0x2d,0x2c,0x2b,0x2a,0x29,0x28,0x27,0x26,0x25,0x24,0x23,0x22,0x21,0x20
,
0x1f,0x1e,0x1d,0x1c,0x1b,0x1a,0x19,0x18,0x17,0x16,0x15,0x14,0x13,0x12,0x11,0x10
,
0x0f,0x0e,0x0d,0x0c,0x0b,0x0a,0x09,0x08,0x07,0x06,0x05,0x04,0x03,0x02,0x01,0x00
,
};

const uint8_t trianglewave[] __attribute__ ((section (".MySection5")))= //triangle wave
{
0x00,0x02,0x04,0x06,0x08,0x0a,0x0c,0x0e,0x10,0x12,0x14,0x16,0x18,0x1a,0x1c,0x1e
,
0x20,0x22,0x24,0x26,0x28,0x2a,0x2c,0x2e,0x30,0x32,0x34,0x36,0x38,0x3a,0x3c,0x3e
,
0x40,0x42,0x44,0x46,0x48,0x4a,0x4c,0x4e,0x50,0x52,0x54,0x56,0x58,0x5a,0x5c,0x5e
,
0x60,0x62,0x64,0x66,0x68,0x6a,0x6c,0x6e,0x70,0x72,0x74,0x76,0x78,0x7a,0x7c,0x7e
,
0x80,0x82,0x84,0x86,0x88,0x8a,0x8c,0x8e,0x90,0x92,0x94,0x96,0x98,0x9a,0x9c,0x9e
,
0xa0,0xa2,0xa4,0xa6,0xa8,0xaa,0xac,0xae,0xb0,0xb2,0xb4,0xb6,0xb8,0xba,0xbc,0xbe,
0xc0,0xc2,0xc4,0xc6,0xc8,0xca,0xcc,0xce,0xd0,0xd2,0xd4,0xd6,0xd8,0xda,0xdc,0xde,
0xe0,0xe2,0xe4,0xe6,0xe8,0xea,0xec,0xee,0xf0,0xf2,0xf4,0xf6,0xf8,0xfa,0xfc,0xfe,
0xff,0xfd,0xfb,0xf9,0xf7,0xf5,0xf3,0xf1,0xef,0xef,0xeb,0xe9,0xe7,0xe5,0xe3,0xe1,
0xdf,0xdd,0xdb,0xd9,0xd7,0xd5,0xd3,0xd1,0xcf,0xcf,0xcb,0xc9,0xc7,0xc5,0xc3,0xc1,
0xbf,0xbd,0xbb,0xb9,0xb7,0xb5,0xb3,0xb1,0xaf,0xaf,0xab,0xa9,0xa7,0xa5,0xa3,0xa1,
0x9f,0x9d,0x9b,0x99,0x97,0x95,0x93,0x91,0x8f,0x8f,0x8b,0x89,0x87,0x85,0x83,0x81
,
0x7f,0x7d,0x7b,0x79,0x77,0x75,0x73,0x71,0x6f,0x6f,0x6b,0x69,0x67,0x65,0x63,0x61
,
0x5f,0x5d,0x5b,0x59,0x57,0x55,0x53,0x51,0x4f,0x4f,0x4b,0x49,0x47,0x45,0x43,0x41
,
0x3f,0x3d,0x3b,0x39,0x37,0x35,0x33,0x31,0x2f,0x2f,0x2b,0x29,0x27,0x25,0x23,0x21
,
0x1f,0x1d,0x1b,0x19,0x17,0x15,0x13,0x11,0x0f,0x0f,0x0b,0x09,0x07,0x05,0x03,0x01
};
const uint8_t ECG[] __attribute__ ((section (".MySection6")))= //ECG wave
{
73,74,75,75,74,73,73,73,73,72,71,69,68,67,67,67,
68,68,67,65,62,61,59,57,56,55,55,54,54,54,55,55,
55,55,55,55,54,53,51,50,49,49,52,61,77,101,132,
```

```
169,207,238,255,254,234,198,154,109,68,37,17,5,
0,1,6,13,20,28,36,45,52,57,61,64,65,66,67,68,68,
69,70,71,71,71,71,71,71,71,71,72,72,72,73,73,74,
75,75,76,77,78,79,80,81,82,83,84,86,88,91,93,96,
98,100,102,104,107,109,112,115,118,121,123,125,
126,127,127,127,127,127,126,125,124,121,119,116,
113,109,105,102,98,95,92,89,87,84,81,79,77,76,75,
74,73,72,70,69,68,67,67,67,68,68,68,69,69,69,69,
69,69,69,70,71,72,73,73,74,74,75,75,75,75,75,75,
74,74,73,73,73,73,72,72,72,71,71,71,71,71,71,71,
70,70,70,69,69,69,69,69,70,70,70,69,68,68,67,67,
67,67,66,66,66,65,65,65,65,65,65,65,65,64,64,63,
63,64,64,65,65,65,65,65,65,65,64,64,64,64,64,64,
64,64,65,65,65,66,67,68,69,71,72,73
};
//array of pointers to signal tables
const uint8_t *SIGNALS[] ={
        sinewave,
        squarewave,
        trianglewave,
        sawtoothwave,
        rewsawtoothwave,
        ECG
};
//adjust LCD stream fuinction to use with printf()
static int LCDsendstream(char c , FILE *stream)
{
LCDsendChar(c);
return 0;
}
//delay 1s
void delay1s(void)
{
        uint8_t i;
        for(i=0;i<100;i++)
        {
                _delay_ms(10);
        }
}
//initialize Timer2 (used for button reading)
void Timer2_Init(void)
{
        TCNT2=0x00;
        sei();
}
```

```
//start timer2
void Timer2_Start(void)
{
	TCCR2|=(1<<CS22)|(1<<CS21); //prescaller 256 ~122 interrupts/s
	TIMSK|=(1<<TOV2);//Enable Timer0 Overflow interrupts
}
//stop timer 2
void Timer2_Stop(void)
{
	TCCR0&=~((1<<CS22)|(1<<CS21)); //Stop timer0
	TIMSK&=~(1<<TOV2);//Disable Timer0 Overflow interrupts

}

//Initial menu
//show initial signal and frequency
//generator is off
void Menu_Update(uint8_t on)
{
	LCDclr();
	CopyStringtoLCD(MENU[(SG.mode)], 0, 0 );
	LCDGotoXY(0, 1);
	if (SG.mode==6)
		{
			CopyStringtoLCD(CLR, 0, 1 );
			LCDGotoXY(0, 1);
			printf("    %5uHz", (uint16_t)SG.deltafreq);
		}
	if (SG.mode==7)
		{
			CopyStringtoLCD(CLR, 0, 1 );
			CopyStringtoLCD(RND, 0, 1 );
		}
	if (SG.mode==8)
		{
		CopyStringtoLCD(CLR, 0, 1 );
		LCDGotoXY(0, 1);
		printf(" %5uMHz", SG.HSfreq);
		}
	if((SG.mode==0)||(SG.mode==1)||(SG.mode==2)||(SG.mode==3)||(SG.mode==4)
||(SG.mode==5))
		{
			CopyStringtoLCD(CLR, 0, 1 );
			LCDGotoXY(0, 1);
			printf(" %5uHz", (uint16_t)SG.freq);
```

```
                }
        if (SG.mode!=6)
        {
                if(on==1)
                        CopyStringtoLCD(MNON, 13, 1 );
                else
                        CopyStringtoLCD(MNOFF, 13, 1 );
        }
}
//update frequency value on LCD menu - more smooth display
void Freq_Update(void)
{
if (SG.mode==6)
{
        LCDGotoXY(0, 1);
        printf("    %5uHz", (uint16_t)SG.deltafreq);
}
if (SG.mode==8)
{
//if HS signal
        LCDGotoXY(0, 1);
        printf(" %5uMHz", SG.HSfreq);
}
if((SG.mode==0)||(SG.mode==1)||(SG.mode==2)||(SG.mode==3)||(SG.mode==4)||(SG.
mode==5))
        {
                LCDGotoXY(0, 1);
                printf(" %5uHz", (uint16_t)SG.freq);
        }
}

//External interrupt0 service routine
//used to stop DDS depending on active menu
//any generator is stopped by setting flag value to 0
//DDs generator which is inline ASM is stopped by setting
//CPHA bit in SPCR register
ISR(INT0_vect)
{
SG.flag=0;//set flag to stop generator
SPCR|=(1<<CPHA);//using CPHA bit as stop mark
//CopyStringtoLCD(MNOFF, 13, 1 );
SG.ON=0;//set off in LCD menu
loop_until_bit_is_set(BPIN, START);//wait for button release
}
//timer overflow interrupt service tourine
```

```
//checks all button status and if button is pressed
//value is updated
ISR(TIMER2_OVF_vect)
{
if (bit_is_clear(BPIN, UP))
//Button UP increments value which selects previous signal mode
//if first mode is reached - jumps to last
        {
        if (SG.mode==0)
        {
                SG.mode=MN_No-1;
        }
        else
        {
                SG.mode--;
        }
        //Display menu item
        Menu_Update(SG.ON);
        loop_until_bit_is_set(BPIN, UP);
        }
if (bit_is_clear(BPIN, DOWN))
//Button Down decrements value which selects next signal mode
//if last mode is reached - jumps to first
        {
        if (SG.mode<(MN_No-1))
                        {
                                SG.mode++;
                        }
                else
                        {
                                SG.mode=0;
                        }
        //Display menu item
        Menu_Update(SG.ON);
        loop_until_bit_is_set(BPIN, DOWN);
        }
if (bit_is_clear(BPIN, RIGHT))
//frequency increment
        {
                if(SG.mode==6)
                {
                        if(SG.deltafreq==10000)
                                SG.deltafreq=1;
                        else
                                SG.deltafreq=(SG.deltafreq*10);
```

```
                Freq_Update();
                loop_until_bit_is_set(BPIN, RIGHT);
            }
            if (SG.mode==8)
                {
                //ifhigh speed signal
                if(SG.HSfreq==8)
                    SG.HSfreq=1;
                else
                    SG.HSfreq=(SG.HSfreq<<1);
                Freq_Update();
                loop_until_bit_is_set(BPIN, RIGHT);
                }

        if((SG.mode==0)||(SG.mode==1)||(SG.mode==2)||(SG.mode==3)||(SG.mode==4)
||(SG.mode==5))
                {
                    if ((0xFFFF-SG.freq)>=SG.deltafreq)
                        SG.freq+=SG.deltafreq;
                    Freq_Update();
                    uint8_t ii=0;
                    //press button and wait for long press (~0.5s)
                    do{
                        _delay_ms(2);
                        ii++;
                    }while((bit_is_clear(BPIN, RIGHT))&&(ii<=250));//wait for
button release
                    if(ii>=250)
                    {
                        do{
                            if ((0xFFFF-SG.freq)>=SG.deltafreq)
                                SG.freq+=SG.deltafreq;
                            Freq_Update();
                        }while(bit_is_clear(BPIN, RIGHT));//wait for button
release
                    }
                }
        }
if (bit_is_clear(BPIN, LEFT))
//frequency decrement
        {
            if(SG.mode==6)
            {
                if(SG.deltafreq==1)
                    SG.deltafreq=10000;
```

```
                else
                        SG.deltafreq=(SG.deltafreq/10);
                Freq_Update();
                loop_until_bit_is_set(BPIN, LEFT);
        }
        if (SG.mode==8)
                {
                //ifhigh speed signal
                if(SG.HSfreq==1)
                        SG.HSfreq=8;
                else
                        SG.HSfreq=(SG.HSfreq>>1);
                Freq_Update();
                loop_until_bit_is_set(BPIN, LEFT);
                }
        if
((SG.mode==0)||(SG.mode==1)||(SG.mode==2)||(SG.mode==3)||(SG.mode==4)||(SG.m
ode==5))
                {
                        if (SG.freq>=SG.deltafreq)
                                SG.freq-=SG.deltafreq;
                        Freq_Update();
                        uint8_t ii=0;
                        //press button and wait for long press (~0.5s)
                        do{
                                _delay_ms(2);
                                ii++;
                        }while((bit_is_clear(BPIN, LEFT))&&(ii<=250));//wait for
button release
                        if(ii>=250)
                        {
                                do{
                                        if (SG.freq>=SG.deltafreq)
                                                SG.freq-=SG.deltafreq;
                                        Freq_Update();
                                }while(bit_is_clear(BPIN, LEFT));//wait for button
release
                        }
                }
        }
if (bit_is_clear(BPIN, START))
        {
if(SG.mode!=6)
        {
                //saving last configuration
```

```
                SG.fr1=(uint8_t)(SG.freq);
                SG.fr2=(uint8_t)(SG.freq>>8);
                SG.fr3=(uint8_t)(SG.freq>>16);
                if (eeprom_read_byte((uint8_t*)EEMODE)!=SG.mode)
eeprom_write_byte((uint8_t*)EEMODE,SG.mode);
                if (eeprom_read_byte((uint8_t*)EEFREQ1)!=SG.fr1)
eeprom_write_byte((uint8_t*)EEFREQ1,SG.fr1);
                if (eeprom_read_byte((uint8_t*)EEFREQ2)!=SG.fr2)
eeprom_write_byte((uint8_t*)EEFREQ2,SG.fr2);
                if (eeprom_read_byte((uint8_t*)EEFREQ3)!=SG.fr3)
eeprom_write_byte((uint8_t*)EEFREQ3,SG.fr3);
                //Calculate frequency value from restored EEPROM values

        SG.freq=(((uint32_t)(SG.fr3)<<16)|((uint32_t)(SG.fr2)<<8)|((uint32_t)(SG.fr1)));
                //calculate accumulator value
                SG.acc=SG.freq/RESOLUTION;
                SG.flag=1;//set flag to start generator
                SG.ON=1;//set ON on LCD menu
                SPCR&=~(1<<CPHA);//clear CPHA bit in SPCR register to allow DDS
                //Stop timer2 - menu inactive
                Timer2_Stop();
                //display ON on LCD
                Menu_Update(SG.ON);
        }
        loop_until_bit_is_set(BPIN, START);//wait for button release
        }
}
/*DDS signal generation function
Original idea is taken from
http://www.myplace.nu/avr/minidds/index.htm
small modification is made - added additional command which
checks if CPHA bit is set in SPCR register if yes - exit function
*/
void static inline Signal_OUT(const uint8_t *signal, uint8_t ad2, uint8_t ad1, uint8_t
ad0)
{
asm volatile(       "eor r18, r18       ;r18<-0"    "\n\t"
                    "eor r19, r19       ;r19<-0"    "\n\t"
                    "1:"                            "\n\t"
                    "add r18, %0        ;1 cycle"               "\n\t"
                    "adc r19, %1        ;1 cycle"               "\n\t"
                    "adc %A3, %2        ;1 cycle"               "\n\t"
                    "lpm                ;3 cycles"  "\n\t"
                    "out %4, __tmp_reg__    ;1 cycle"   "\n\t"
                    "sbis %5, 2         ;1 cycle if no skip" "\n\t"
```

```
                        "rjmp 1b            ;2 cycles. Total 10 cycles"     "\n\t"
                        :
                        :"r" (ad0),"r" (ad1),"r" (ad2),"e" (signal),"I"
(_SFR_IO_ADDR(PORTA)), "I" (_SFR_IO_ADDR(SPCR))
                        :"r18", "r19"
        );
}

void Timer1_Start(uint8_t FMHz)
{
switch(FMHz){
        case 1:
                //start high speed (1MHz) signal
                OCR1A=7;
                break;
        case 2:
                OCR1A=3;//2MHz
                break;
        case 4:
                OCR1A=1;//4MHz
                break;
        case 8:
                OCR1A=0;//8MHz
                break;
        default:
                OCR1A=7;//defauls 1MHz
                break;}
        //Output compare toggles OC1A pin
        TCCR1A=0x40;
        //start timer without prescaler
        TCCR1B=0b00001001;
}
void Timer1_Stop(void)
{
        TCCR1B=0x00;//timer off
}
//main init function
void Main_Init(void)
{
//stderr = &lcd_str;
stdout = &lcd_str;
//--------init LCD----------
LCDinit();
LCDclr();
LCDcursorOFF();
```

```
//-------EEPROM initial values----------
if (eeprom_read_byte((uint8_t*)EEINIT)!='T')
{
eeprom_write_byte((uint8_t*)EEMODE,0x00);//initial mode 0 – OUT~~~~;
eeprom_write_byte((uint8_t*)EEFREQ1,0xE8);//initial frequency 1kHz
eeprom_write_byte((uint8_t*)EEFREQ2,0x03);
eeprom_write_byte((uint8_t*)EEFREQ3,0x00);
eeprom_write_byte((uint8_t*)EEINIT,'T');//marks once that eeprom init is done
//once this procedure is held, no more initialization is performed
}
//------restore last saved values from EEPROM------
SG.mode=eeprom_read_byte((uint8_t*)EEMODE);
SG.fr1=eeprom_read_byte((uint8_t*)EEFREQ1);
SG.fr2=eeprom_read_byte((uint8_t*)EEFREQ2);
SG.fr3=eeprom_read_byte((uint8_t*)EEFREQ3);
SG.freq=(((uint32_t)(SG.fr3)<<16)|((uint32_t)(SG.fr2)<<8)|((uint32_t)(SG.fr1)));
SG.acc=SG.freq/RESOLUTION;
SG.flag=0;
//default 1MHz HS signal freq
SG.HSfreq=1;
SG.deltafreq=100;
//------------init DDS output-----------
R2RPORT=0x00;//set initial zero values
R2RDDR=0xFF;//set A port as output
//-------------set ports pins for buttons----------
BDDR&=~(_BV(START)|_BV(UP)|_BV(DOWN)|_BV(RIGHT)|_BV(LEFT));
BPORT|=(_BV(START)|_BV(UP)|_BV(DOWN)|_BV(RIGHT)|_BV(LEFT));
//---------set ports pins for HS output---------
HSDDR|=_BV(HS);//configure as output
//-----------Menu init--------------
SG.ON=0;//default signal is off
Menu_Update(SG.ON);
//-----------Timer Init-------------
Timer2_Init();
//Start Timer with overflow interrupts
Timer2_Start();
}

int main(void)
{
//Initialize
Main_Init();
while(1)//infinite loop
        {
        if (SG.flag==1)
```

```
		{
		GICR|=(1<<INT0);//set external interrupt to enable stop
		if (SG.mode==7)
			{
			//Noise
			do
			{
				R2RPORT=rand();
			}while(SG.flag==1);
			//set signal level to 0
			R2RPORT=0x00;
			//display generator OFF
			Menu_Update(SG.ON);
			//stop external interrupt
			GICR&=~(1<<INT0);
			//start timer menu active
			Timer2_Start();
			}
/*		else if (SG.mode==6)
			{
			//freq step
			while((SG.flag==1))
			{
			//not implemented
			CopyStringtoLCD(NA, 0, 1 );
			}
			//set signal level to 0
			R2RPORT=0x00;
			//display generator OFF
			Menu_Update(SG.ON);
			GICR&=~(1<<INT0);//|(1<<INT1);//stop external interrupt
			//start timer menu active
			Timer2_Start();
			}*/
		else if (SG.mode==8)
			{
			//High speed signal
			Timer1_Start(SG.HSfreq);
			while((SG.flag==1))
			{
			//not implemented
			CopyStringtoLCD(MNON, 13, 1 );
			}
			Timer1_Stop();//timer off
			//set HS pin to LOW
```

```
                HSPORT&=~(1<<HS);
                //display generator OFF
                Menu_Update(SG.ON);
                GICR&=~(1<<INT0);//|(1<<INT1);//stop external interrupt
                //start timer menu active
                Timer2_Start();
                }
            else
                {
                //start DDS
                Signal_OUT(SIGNALS[SG.mode],
                                (uint8_t)((uint32_t)SG.acc>>16),
                                (uint8_t)((uint32_t)SG.acc>>8),
                                (uint8_t)SG.acc);
                //set signal level to 0
                R2RPORT=0x00;
                //display generator OFF
                Menu_Update(SG.ON);
                GICR&=~(1<<INT0);//|(1<<INT1);//stop external interrupt
                //start timer menu active
                Timer2_Start();
                }
            }
        }
        return 0;
}
```