МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ
ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ВНУТРІШНІХ СПРАВ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ УПРАВЛІННЯ, ПСИХОЛОГІЇ ТА БЕЗПЕКИ

Кафедра інформаційних технологій

РОЗРОБЛЕННЯ ПРИСТРОЮ НА МІКРОКОНТРОЛЕРІ ДЛЯ ТЕСТУВАННЯ ПОВНОКОЛІРНИХ СВІТЛОДІОДНИХ СТРІЧОК

Кваліфікаційна робота
здобувача вищої освіти
4 курсу денної форми навчання
Олега Михайловича ФЕЦЯКА

Науковий керівник:
Наталія Володимирівна ГАЛАЙКО

Рецензент:

_____
вчене звання, науковий ступінь

_____
(Ім’я ПРІЗВИЩЕ рецензента)

Кваліфікаційна робота допущена до захисту
«___» _____ 2025 р., протокол № ____

Завідувач кафедри інформаційних технологій
_____Ігор ФЕДЧАК_
(підпис)

Львів
2025

СПИСОК УМОВНИХ СКОРОЧЕНЬ

MCU (МК) Microcontroller Unit (мікроконтролер);

ADC (АЦП) Analog-to-Digital Converter (аналого-цифровий перетворювач);

АЗ апаратне забезпечення;

ПЗ програмне забезпечення;

ВС вбудована система;

ЦП цифровий пристрій;

САПР система автоматизованого проектування;

AVR сімейство 8-розрядних мікроконтролерів фірми Microchip (Atmel);

FLASH різновид напівпровідникової технології пам’яті, яка електричним методом перепрограмовується;

EEPROM Electrically Erasable Programmable Read-Only Memory (постійна пам’ять, яку можна стирати та перепрограмовувати електричним методом);

RAM (ОЗП) – Random Access Memory (оперативна пам’ять або оперативний запам’ятовуючий пристрій);

ROM (ПЗП) – Read-Only Memory (постійний запам’ятовуючий пристрій. Незалежна пам’ять, використовується для зберігання незмінних даних);

LCD (РКД) – Liquid-Crystal Display (рідкокристалічний дисплей);

LED RGB strip – Light-emitting diode Red-Green-Blue strip (повноколіріна світлодіодна стрічка);

SPI – Serial Peripheral Interface, SPI bus (послідовний периферійний інтерфейс, шина SPI) - послідовний синхронний стандарт передачі даних в режимі повного дуплексу, призначений для високошвидкісного обміну даними мікроконтролерів і периферії);

I^2C – Inter-Integrated Circuit (послідовна шина даних, що використовує дві лінії зв’язку SDA і SCL);

PWM (ШІМ) – Pulse Width Modulation (широтно-імпульсна модуляція).

АНОТАЦІЯ

Фецяк О.М., Галайко Н.В. (керівник). Розроблення пристрою на мікроконтролері для тестування повноколірних світлодіодних стрічок. Бакалаврська кваліфікаційна робота. – Львівський державний університет внутрішніх справ, Львів, 2025.

У дипломній роботі розроблено апаратне та програмне забезпечення універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок. Універсальний пристрій дає можливість досліджувати динамічні та світлові характеристики світлодіодних повноколірних (RGB) стрічок , має гнучке меню налаштувань з можливістю зберігання параметрів тестування в довготривалій памяті, має в памяті прості тести для перевірки справності RGB стрічок, та потенціометри ручного налаштування параметрів тестування. Параметри тестування виводяться на рідкокристалічний дисплей.

В роботі розроблено принципову електричну схему і модель універсального пристрою для тестування світлодіодних повноколірних стрічок в САПР Proteus VSM. Розроблено його алгоритм роботи і програмне забезпечення на мові С в середовищі розробки вбудованого ПЗ для МК сімейства AVR CodeVisionAVR. Проведено моделювання пристороюв в емуляторі Proteus ISIS.

Ключові слова: тестування RGB стрічок, рідкокристалічний дисплей, мікроконтролер AVR, САПР Proteus VSM, мова програмування С, вбудоване програмне забезпечення, інструментарій розробки програмного забезпечення для МК AVR CodeVisionAVR.

ABSTRACT

Fetsyak O.M., Halayko N.V. (supervisor). Development of a microcontroller-based device for for testing RGB strips. Bachelor's thesis. – Lviv State University of Internal Affairs, Lviv, 2025.

In the diploma thesis, hardware and software of the universal device for testing RGB strips have been developed.

Universal device for testing RGB led strips give ability for testing dynamic and light characteristics RGB led strips. Device have testing menu with ability to save testing parameters in flash memory, also have simple tests for testing RGB led strips and variable resistors for manual setting testing parameters. Testing parameters outputs onto the LCD.

Universal device for testing RGB led strips has been built on the AVR MCU, to which LED strips and liquid-crystal display are connected.

In the work, the electric circuit and model of the universal device for testing RGB led strips have been developed using the CAD system – Proteus VSM. Its operation algorithm and software for the AVR family microcontroller have been developed in C programming language using the CodeVisionAVR environment. The simulation of the measuring information system has been performed in Proteus ISIS.

Keywords: testing RGB led strips, LCD, AVR microcontroller, CAD Proteus VSM, embedded C, embedded software, software development tool for AVR microcontrollers CodeVisionAVR.

ЗМІСТ

ВСТУП

З появою мікроконтролерів, є можливість створювати недорогі тестові вимірювальні та управляючі системи. Мікроконтролери є основою багатьох сучасних пристроїв та приладів. З допомогою мікроконтролерів легше і набагато дешевше реалізувати алгоритм роботи системи. МК може управляти різними пристроями отримувати від них дані, виконувати математичні обчислення, так як МК містить в собі регістри, порти вводу-виводу, флеш-пам’ять, таймера-лічильники, аналого-цифрові перетворювачі та т.п.) все це реалізовано безпосередньо на кристалі МК. Таке виконання дозволяє зменшити розміри конструкції і зменшити енергоспоживання.

Аналіз останніх досліджень і публікацій. Створенню цифрових пристроїв на мікроконтролерах присвячено чимало публікацій і проєктів [1-5]. Ці праці заклали основу для розроблення універсального пристрою на мікроконтролері для тестування повноколірних (RGB) світлодіодних стрічок.

Слід також зазначити, що методи та засоби дослідження динамічних та світлових характеристик світлодіодних стрічок, розвивають і вдосконалюють, тому розроблення мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок є актуальним.

У зв’язку зі сказаним можемо сформулювати мету та завдання кваліфікаційної роботи.

Метою роботи є створення апаратного і вбудованого програмного забезпечення універсального пристрою для тестування повноколірних світлодіодних стрічок, який дає можливість досліджувати динамічні та світлові характеристики світлодіодних повноколірних стрічок, що має гнучке меню налаштувань з можливістю зберігання параметрів тестування в довготривалій пам’яті, мати в пам’яті прості тести для перевірки справності RGB стрічок, та потенціометри ручного налаштування параметрів тестування. Параметри тестування мають виводитися на рідкокристалічний дисплей. Універсальний пристрій може використовуватися для тестування будь-яких світлодіодних стрічок в тому числі і повноколірних.

Для розробки апаратного забезпечення універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок вибрано МК AVR.

Для досягнення мети необхідно вирішити наступні **завдання**:

- розробити пристрій на мікроконтролері для тестування повноколірних світлодіодних стрічок на основі наступних даних:
  - 8-розрядний мікроконтролер серії AVR фірми Atmel ATmega128;
  - світлодіодна повноколірна RGB стрічка.
  - символьний LCD-дисплей 16x2. Модель: WH1602B-YGK-CTK;
- спроєктувати універсальний мікроконтролерний пристрій для тестування повноколірних світлодіодних стрічок засобами САПР ProteusVSM;
- розробити алгоритм роботи універсального мікроконтролерного пристрою;
- створити програмний модуль тестування повноколірних світлодіодних стрічок;
- створити програмний модуль виводу інформації на символьний LCD-дисплей 16x2;
- розробити основне вбудоване програмне забезпечення універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок згідно алгоритму його роботи;
- провести моделювання роботи спроєктованого універсального пристрою для тестування повноколірних світлодіодних стрічок та проаналізувати отримані результати в емуляторі Proteus ISIS.

Об’єкт дослідження – проєктування універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок.

Предмет дослідження – моделі та засоби проєктування універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок.

Методи досліджень. Під час виконання роботи застосовано низку загальновідомих методів пізнання, які дозволили у повному обсязі досягти мети та вирішити завдання дослідження. Так, за допомогою бібліометричного методу здійснено опрацювання опублікованих наукових праць, що дозволило осягнути динаміку розвитку даної проблеми. За допомогою різних філософських (діалектичного, феноменологічного, синергетичного, аксіологічного) методів пізнання здійснено всебічний аналіз поглядів на предмет дослідження. Використання значної кількості загальнонаукових методів (структурного, індуктивного, дедуктивного, аналізу, синтезу, моделювання) дозволило досягнути виконання визначених вище теоретичних та практичних завдань дослідження.

Структура роботи. Кваліфікаційна робота складається із вступу, чотирьох розділів, висновків, списку використаних джерел, додатків. Обсяг основного тексту роботи складає 63 сторінки, 44 рисунки, 10 таблиць, 3 додатки і 27 бібліографічних джерел. Загальний обсяг роботи – 80 сторінок.

РОЗДІЛ 1
СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Види світлодіодних стрічок та основні пристрої для їх управління

Світлодіодні стрічки – це нова перспективна форма освітлення. Існує багато варіантів світлодіодних стрічок, але для більшості примінень вони мають наступні характеристики [6]:

- складаються з багатьох світлодіодів змонтованих на гнучку основу;
- працюють від низької постійної напруги;
- мають широкий спектр випромінювання різні кольори і яскравість;
- змотуються в рулон довжиною по 5 метрів і більше з можливістю порізки на коротші стрічки.

Ширина світлодіодних стрічок як правило 10–12 мм. Крок порізки стрічки 1–2 дюйми. Густина розміщення випромінюючих світло діодів 60–120 штук на метр довжини. Якість світлодіодної стрічки залежить в основному від характеристик випромінюючих світлодіодів, а ці характеристики в різних виробників відрізняються. Оцінити якість стрічки можна візуально використовуючи відповідне тестове обладнання. Гнучка основа світлодіодної стрічки дозволяє монтувати її на будь-яку поверхню. На рис. 1.1 зображено типову світлодіодну стрічку.

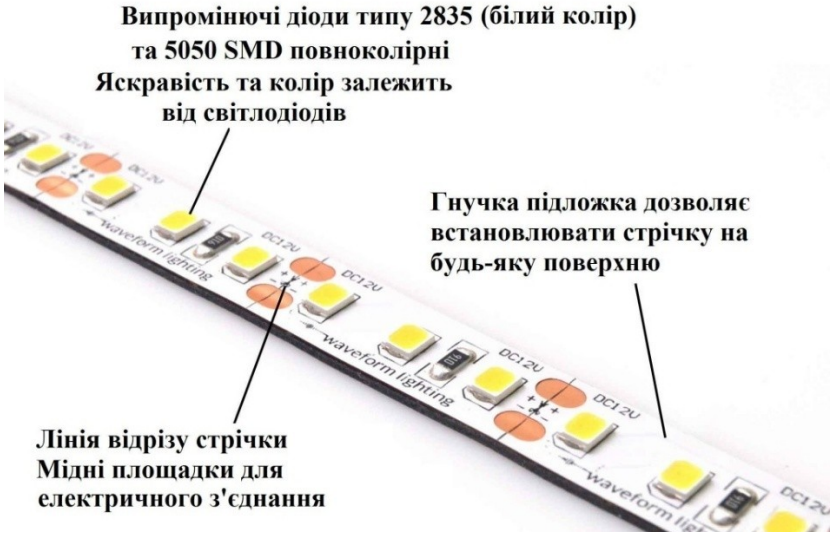


Рис. 1.1. Світлодіодна стрічка з 2835 SMD світлодіодами

Яскравість світлодіодних стрічок вимірюється в люменах. Різні стрічки мають різну яскравість, що не завжди залежить від потужності. Світлодіодна стрічка хорошої якості повинна мати яскравість 1500 люмен на метр (люмінесцентна лампа має 1800 люмен).

Яскравість світлодіодних стрічок залежить від трьох факторів:

- характеристик світлодіодів;
- кількості світлодіодів на метр довжини стрічки;
- підведеної потужності на метр довжини.

Наприклад стрічка довжиною 30 см з яскравістю 450 люмен відповідає лампі розжарення потужністю 40 Вт (рис. 1.2).

Рис. 1.2. Порівняльні характеристики світлодіодної стрічки і лампи розжарення

Існує широкий асортимент стрічок з різними типами світлодіодів (2835, 3528, 5050 або 5730) суттєвим є кількість світлодіодів на метр довжини (див. рисунок 1.3.) і потужність яку вони споживають. Від цих характеристик залежить ціна. Світлодіодна стрічка хорошої якості повинна споживати 15 Ват/метр, або більше.

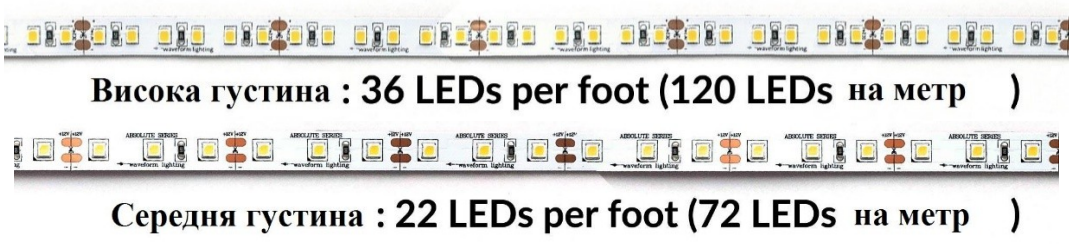


Рис. 1.3. Порівняльні характеристики світлодіодних стрічок

Світло від світлодіодних стрічок може бути білим або кольоровим. Найбільш вживані стрічки з білим світлом. Основними характеристиками світлодіодної стрічки є кольорова температура (color temperature (CCT)) і індекс кольоропередачі (color rendering index (CRI)). Від кольорової температури залежить те наскільки людина буде відчувати себе комфортно в приміщенні освітленим світлодіоднними стрічками. Кольорова температура вимірюється в градусах по шкалі Кельвіна (К). За європейськими нормами всі джерела світла по колірності розділені на три групи:

- теплий білий (CCT = нижче 3500 K);
- нейтральний білий або денний (CCT = 3500-5300 K);
- холодний білий (CCT = выще 5300 K).

Кольорова температура звичайної лампи розжарення – приблизно 2700 К, Тепло-білий колір свічення светлодіодних ламп найбільше підходить оку (від 2700 до 3500К). На рис. 1.4. зображено діаграму сприйняття оком кольорової температури.

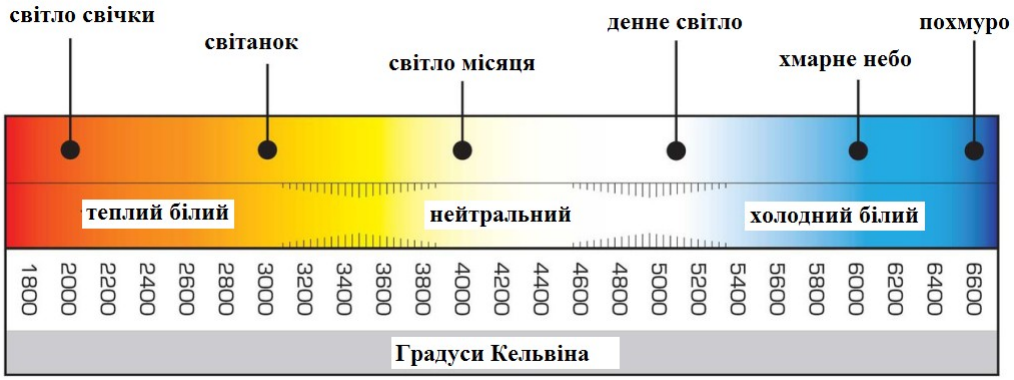


Рис. 1.4. Колірність джерел світла (CCT)

Для більшості видів робіт і приміщень рекомендуються “нейтральні” джерела світла (CCT = 4000 – 4500 К). Тепле світло розслаблює і створює атмосферу затишку, а більш холодні тона допомагають організму концентруватися і налаштовуватися на робочий лад.

На робочому місці кольорова температура повинна бути максимально близька до кольору природнього освітлення. При білому світлі (денне освітлення) і тривалій роботі людини приймемо її продуктивність за 100%, то при жовтому світлі вона буде тільки 93%, при зеленому 92%, при голубому 78%, при червоному і оранжевому 76%. Отже на робочому місці денне світло буде більш корисне (приблизно 4000 – 4500 К). Для читання корисне більш холодне біле світло (але тільки до 6500 К).

Мяке біле – тепле біле світло (2700 – 3500К): найкраще підходить для спалень та віталень.

Яскраво-біла холодна білизна (5300 – 6500 К): найкраще підходить в кухнях, ваннах, гаражі, створює більш енергійний настрій.

Денне світло (4000 – 5000 К): добре підходить для ванн, кухонь і підвалів; ідеальне для читання, для работи зі складними проектами, забезпечує найбільший контраст між кольорами.

Колір свічення світлодіодної стрічки грає основну роль при її виборі. Випускають одноколірні і багатоколірні (повноколірні) (RGB) стрічки.

Одноколірні стрічки бувають тільки: синього, червоного, зеленого, жовтого і білого кольору. Перші 4 кольори отримують при використанні спеціальних матеріалів в процесі виготовлення кристалів, та з білим кольором ситуація складніша. Білий колір отримують двома способами:

– перший спосіб полягає на застосуванні синіх світлодіодів. Зверху синій світлодіод покривають люмінофором, під дією випромінювання синього светлодіода люмінофор починає випромінювати світло білого кольору. Змінюючи товщину люмінофора отримують різну температуру кольору, тобто білий колір може бути теплим (з жовтим віддтінком) або холодним (з синім відтінком);
– другий спосіб полягає в застосуванні повноколірної стрічки з RGB світлодіодами. RGB світлодіод має три кристали: червоний, зелений і синій. Як відомо, якщо змішати зелений, червоний і синій колір, отримаємо білий. У цього способа є перевага: в будь-який момент можна

змінити колір свічення стрічки. На рис. 1.5. зображено два світлодіоди: білий і RGB.

Рис. 1.5. Типи SMD світлодіодів (білий SMD 3528 і RGB SMD5050)

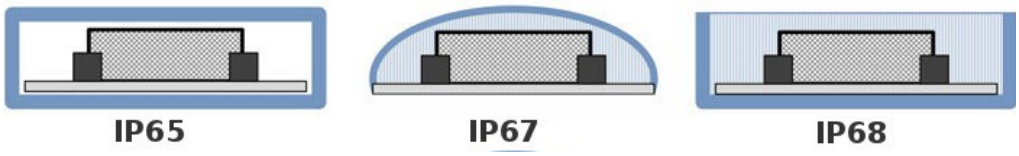
Найбільш поширені світлодіодні стрічки з діодами SMD 3528 і SMD5050. Аббревіатура SMD – означає повехневий тип монтажа світлодіода, тобто контакти діода припаяні прямо до поверхні монтування (рис. 1.6). Цифри після SMD означають геометричні розміри світлодіода, 3528 – діод 3,5х2,8 мм, 5050 – діод 5х5 мм.

Рис. 1.5. Стрічки зі змонтованими SMD світлодіодами

Якщо глянути на світлодіоди, то можно побачити, що SMD5050 має 6 виводів, а SMD3528 тільки 2. Світлодіод 5050 має в собі 3 окремих кристала, а 3528 один кристал. Це впливає на яскравість свічення: стрічки SMD5050 більш яскраві і більш дорогі. Більшість світлодіодних стрічок живлять напругою 12V або 24V DC, але випускаються і на напругу 5V.

З появою нових типів світлодіодів асортимент стрічок поповнюється. Так почався випуск світлодіодів 2835 на заміну 3528. SMD2835 мають меншу товщину і більшу яскравість, а також одноколірні з підвищеною яскравістю SMD 5630/ SMD 5730. Світлодіодні стрічки використовуються для основного освітлення, декоративної підсвітки елементів інтер'єру всередині приміщень з низьким рівнем вологості (ніші, стелі, карнізи, сходи і т.п.), а також – підсвітка різних рекламних конструкцій і торгового обладнання всередині приміщень (вивіски, вітрини, шафи і т.п.); підсвітка меблів (шафи-купе, кухні, і т.п.); підсвітка басейнів, кімнатних рослин, домашніх декоративних клумб і ландшафтних елементів, колон, танцювальних площадок елементів днища і номерних знаків автомобіля і т.п. Зрозуміло, що стрічки використовують не тільки для підсвітки інтер'єрів спалень, часто їх використовують для зовнішнього оформлення будинків, магазинів і т.п.. Стрічка для зовнішнього застосування та для вологих приміщень повинна бути захищена. В зв'язку з цим стрічки діляться на вологозахищені і вологонезахищені. У волого незахищених стрічок видно струмопровідні доріжки, місця пайки елементів ніяк не захищені.

Вологозахищені стрічки як правило заливаються силіконом. Про степінь захисту від зовнішніх впливів можно судити по класу захищеності (IP). Так стрічку з IP68 можна застосувати для підсвітки басейнів (рис. 1.6).



IP65 – стрічка вдягнута в чохол; IP67 – стрічка залита силіконом; IP68 – стрічка в чохлі, який залитий силіконом

Рис. 1.6. Клас захищенності світлодіодних стрічок

Для управління одноколірними світлодіодними стрічками достатньо мати блок живлення на відповідну напругу і потужність. Напруга живлення і її полярність вказані на стрічці. Для під'єднання такої стрічки достатньо два

проводи. Для управління повноколірною світлодіодною стрічкою потрібно чотири проводи (рис. 1.7).

Рис. 1.7. Повноколірна світлодіодна стрічка (фрагмент)

Характеристики такої стрічки можуть бути наступні:

- тип стрічки: RGB (багатоколірна);
- розмір (довжина): 5 м (постачається в катушках по 5 метрів);
- ширина підложки: 15 мм;
- розмір світлодіода: 5,0х5,0 мм;
- кількість світлодіодів: 60 шт/метр;
- напруга живлення: 12 В;
- споживана потужність: 28 Вт/метр;
- яскравість: 11-12 Лм/діод;
- кут випромінювання світла: 120°;
- колір світла: RGB (повноколірна);
- рівень захисту: IP 67;
- робоча температура: від -40° до 60°С;
- кратність порізки: 5 см (6 світлодіодів).

Для зміни кольору RGB-стрічки використовують спеціальні RGB-контролери. На рис. 1.8, 1.9, 1.10 зображено контролери для управління повноколірними світлодіодними стрічками [7-9].

Рис. 1.8. RGB LED Controller з Wireless IR Remote – 3 Amps/на канал – ціна $14,95

Рис. 1.9. RGB LED Controller – Wireless RF Remote – 5 Amps/на канал – ціна $29,95

Рис. 1.10. RGB LED Controller – Wireless RF Touch Color Remote – 8 Amps/на канал – ціна $24,95

1.2. Огляд існуючих підходів до розробки систем управління світлодіодними стрічками

Провівши пошук подібних розробок отримано інформацію про існуючі підходи по розробках таких систем управління світлодіодними стрічками [10]. На сьогоднішній день пропонуються однотипні рішення щодо систем

управління світлодіодними стрічками. В основному це драйвер з пультом управління. Яскравістю світлодіодних стрічок керувати простою зміною напруги не вдається, як це реалізовано в димерах ламп розжарення, тут інший підхід.

Димер світлодіодної стрічки – це ШІМ димер. Світлодіодна повноколірна стрічка – це RGB світлодіоди (рис. 1.11).

- Колір випромінювання: синій, червоний, зелений
- R: Довжина хвилі 630-640 nm
- G: Довжина хвилі 515-525 nm
- B: Довжина хвилі 465-475 nm

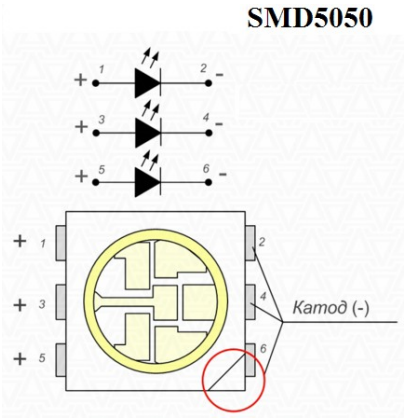


Рис. 1.11. RGB світлодіоди і їх характеристики

Розглянемо управління стрічки на прикладі управління RGB світлодіодом. Для адаптації до різних варіантів схем управління, RGB діоди виробляють в таких модифікаціях:

- виконання з спільним катодом;
- виконання з спільним анодом;
- без спільного анода або катода, з шістьма виводами (рис. 1.11).

В першому випадку світлодіод керується сигналами позитивної полярності, які поступають на аноди, у другому – негативними імпульсами, які поступають на катоди. Третя модифікація виконання допускає будь-які варіанти комутації і випускається у виді SMD компонента. На рис. 1.12 наведено можливі схеми підключення RGB діодів.

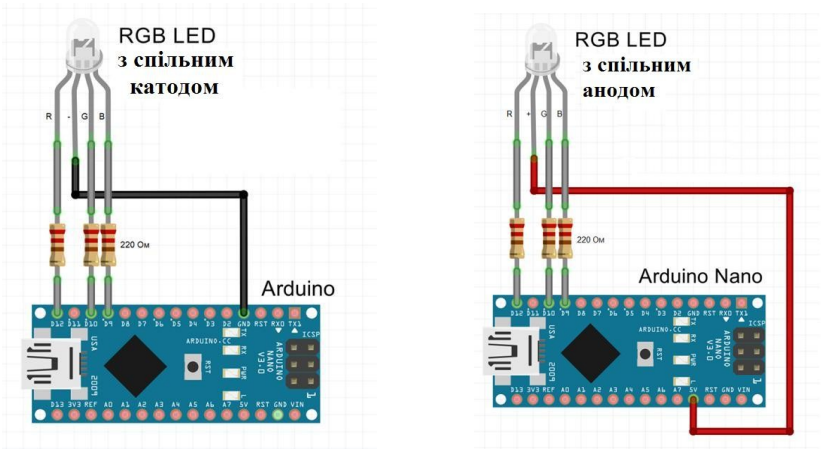


Рис. 1.12. Схеми підключення RGB діодів до плат ARDUINO

Включення світлодіода відбувається при проходженні через нього прямого струму, коли анод під'єднано до плюса, катод до мінуса. Багатоколірний спектр випромінювання можна отримати, змінюючи інтенсивність світла кожного світлодіода окремо (RGB). Результуючий колір визначається співвідношенням яскравостей окремих (RGB) кольорів. Якщо всі 3 кольори однакові за інтенсивністю світла, то результуючий колір буде білим.

На цифрових виходах плат Arduino формуються періодичні прямокутні імпульси напруги, (рис. 1.12.), з різною шпаруватістю.

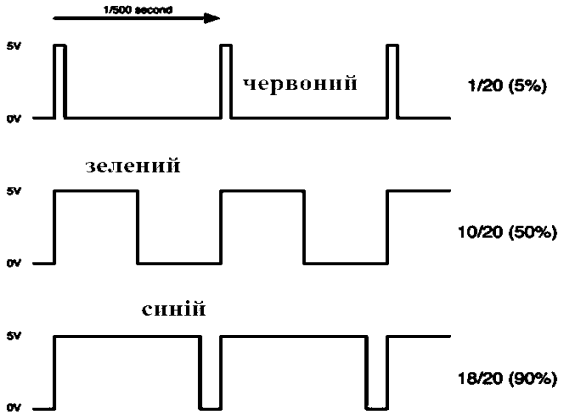


Рис. 1.13. Широтно-імпульсне управління інтенсивністю світлодіода

Шпаруватість імпульсів наприклад в ARDUINO можна змінювати від 0..255 (0…100%) таким чином керувати інтенсивністю світлодіодів, а отже і результуючим кольором (рис. 1.14). Частота імпульсів фіксована і як правило становить 500 Гц для контролерів які випускаються масово.

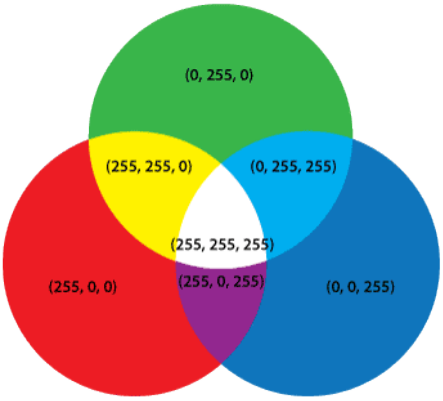


Рис. 1.14. Діаграма залежності кольору від інтенсивності RGB

В дипломній роботі запропоновано універсальний пристрій для тестування повноколірних світлодіодних стрічок, який дає можливість досліджувати динамічні та світлові характеристики світлодіодних повноколірних і монохромних (одноколірних) стрічок, має гнучке меню налаштувань з можливістю зберігання параметрів тестування в довготривалій памяті, має в пам’яті прості тести для перевірки справності RGB стрічок, та потенціометри ручного налаштування параметрів тестування. Параметри тестування виводяться на рідкокристалічний дисплей. Пристрій має ширші можливості дешевший в порівнянні з існуючими контролерами, а саме рідкокристалічний екран, меню управління, можливість встановлення частоти ШІМ, відображення інтенсивності кольорів на РКД і має тестове призначення.

1.3. Управління світлодіодними повноколірними стрічками

В дипломній роботі для управління світлодіодною повно колірною стрічкою використано мікроконтролер ATmega128, а саме його 16-ти розрядний таймер3 в режимі Fast PWM. Таймер3 має три канали ШІМ з виходом на відповідні контакти мікроконтролера, що дозволяє оперативно змінювати шпаруватість імпульсів незалежно по кожному каналу, таким чином регулюючи колір повноколірної світлодіодної стрічки. Для випробування динамічних характеристик світлодіодної стрічки можна переналаштовувати частоту ШІМ імпульсів в діапазоні від 1 Гц до 30 000 Гц з

точністю 1 Гц у всьому діапазоні. В контролерах які випускаються промислово такої можливості не передбачено.

В МК ATmega128 на цифрових виходах формуються періодичні прямокутні імпульси напруги, з різною шпаруватістю. Шпаруватість можна задавати за допомогою трьох потенціометрів в діапазоні 1..99 % по кожному каналу окремо. Включення світлодіода відбувається при проходженні через нього прямого струму, коли анод підєднано до плюса, катод до мінуса. Багатоколірний спектр випромінювання можна отримати, змінюючи інтенсивність світла кожного світлодіода окремо (RGB). Результуючий колір визначається співвідношенням яскравостей окремих (RGB) кольорів. Якщо всі 3 кольори однакові за інтенсивністю світла, то результуючий колір буде білим. Для управління стрічками різної потужності і з різною напругою живлення розроблено силовий модуль, який після відповідних налаштувань забезпечує їх роботу і тестування. Для регулювання частоти і шпаруватості ШІМ каналів використано Таймер3 мікроконтролера ATmega128 в режим Fast PWM. В цьому режимі регістр ICR3 задає TOP, а регістри OCR3A, OCR3B, OCR3C використано для управління шпаруватістю відповідного каналу ШІМ. Прямокутні імпульси генеруються на трьох виходах МК OC3A, OC3B, OC3C. Вихідна частота імпульсів на виході обчислюється за формулою:

$$Focrnx = \frac{fclk}{N * (1 + TOP)} \quad (1.1)$$

де fclk – тактова частота кварцевого генератора МК, N – масштабний коефіцієнт (prescaler divider: 1, 8, 64, 256, або 1024), TOP – значення в регістрі ICR3.

Регістр OCR3A(OCR3B, OCR3C) – змінна, яка відповідає за величину заповнення імпульса. Змінюється дана величина від 0 до 65536 (2^{16}). 65536 відповідає 100%-му заповненню. Отже, якщо потрібно 30% заповнення – (65536/100)*30=19661 (4CCD) – ICR3=4CCD. Змінюючи дану величину можна регулювати частоту ШІМ. Величина частоти работи ШІМ кратна частоті, на якій працює мікроконтролер.

РОЗДІЛ 2
ЗАСОБИ РОЗРОБКИ АПАРАТНОГО І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ УНІВЕРСАЛЬНОГО ПРИСТРОЮ ДЛЯ ТЕСТУВАННЯ ПОВНОКОЛІРНИХ СВІТЛОДІОДНИХ СТРІЧОК

2.1. Система автоматизованого проектування і моделювання програмованих пристроїв Proteus VSM

Proteus VSM – це ПЗ розробки і моделювання і симуляції електричних схем в реальному масштабі часу, які можуть містити мікроконтролери різних типів, а також платформи ARDUINO [11]. Proteus VSM – це розробка компанії Labcenter Electronics (Великобританія). Програмне забезпечення Proteus має два основних модуля:

1) ISIS – це редактор, для створення електричних схем та їх моделювання.
2) ARES – редактор друкованих плат з автотрасувальником, можливістю розстановки в автоматичному режимі компонентів на друкованій платі.

Proteus дозволяє не тільки симуляцію розробленої електричної схеми, але й розведення та виготовлення друкованої плати, формування документації та програм для верстатів з ЧПУ та створення фотошаблонів. Особливістю (Proteus-а) є можливість моделювання роботи програмованих пристроїв: мікропроцесорів, мікроконтролерів, модулів ARDUINO , DSP і т.п.

Програма Proteus ISIS використовується для створення і симуляції електричних схем має широкий набір електронних компонентів. Серед моделей ISIS джерела живлення, сигнальні генератори, амперметри, вольтметри, для аналізу сигналів – осцилограф, проби для моніторингу параметрів схеми, мотори, дискретні компоненти, резистори, конденсатори, вимикачі, індуктивності, трансформатори, цифрові та аналогові мікросхеми, транзистори, реле, мікроконтролери різних типів, процесори, сенсори та багато іншого.

В Proteus включено програму ARES для проектування і моделювання друкованих плат з їх тривимірною візуалізацією. Програма ARES дозволяє

проектування до 14 шарів доріжок, має функції само-розміщення і автоматичного трасування доріжок з'єднання. Електричну схему змодельовану у програмі ISIS можна автоматично передати в ARES для подальшої розробки друкованої плати.

В Proteus можна перевірити роботу схеми з такими мікроконтролерами як: ARM7, AVR, PIC, Motorola, Basic Stamp. Внутрішня бібліотека компонентів підтримує МК: PIC, AVR, 8051, ARM7/LPC2000. В бібліотеці їх більше 6000 цифрових та аналогових моделей різних мікросхем і електричних компонентів. Моделювання електричних схем не може, абсолютно точно відтворити роботу реального пристрою. Для відлагодження, алгоритмів роботи МК, програми Proteus цілком достатньо. Proteus налічує у бібліотеці багато електронних компонентів, а компоненти для яких моделі відсутні можна дозавантажити, або створити.

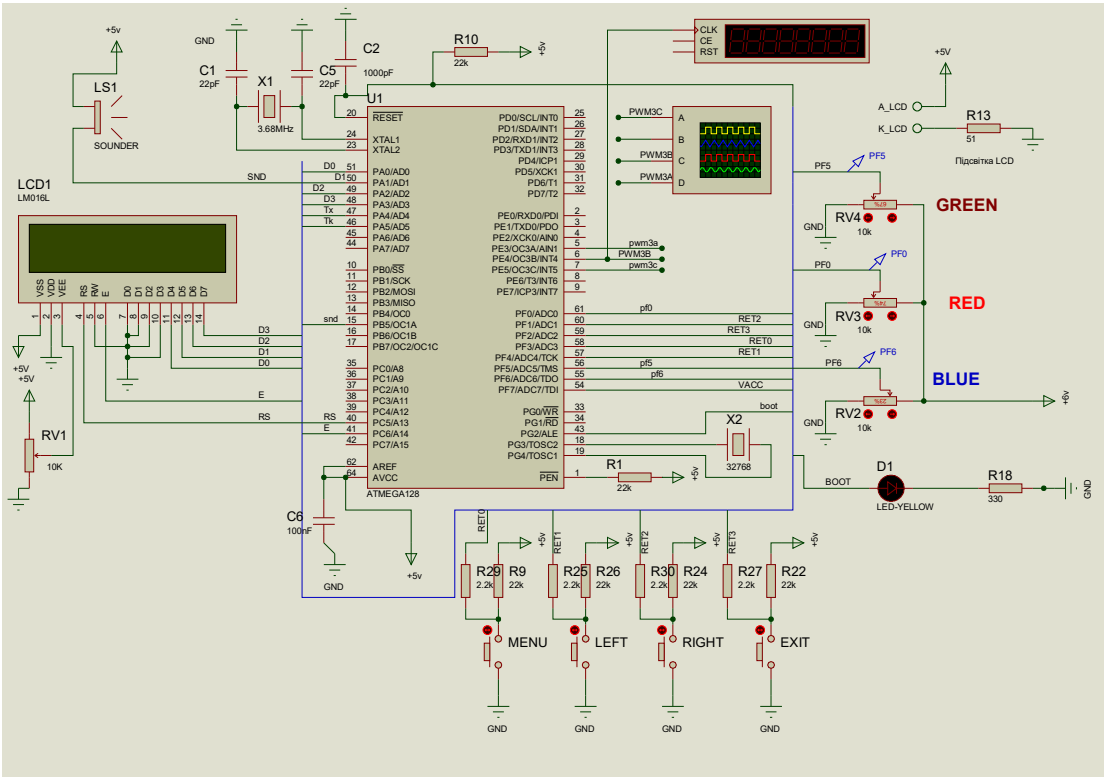


Рис. 2.1. Електрична схема створена в Proteus ISIS

Розглянемо приклад створення простої електричної схеми в Proteus ISIS. Відкриємо програму Proteus ISIS і створемо новий проєкт (рис. 2.2). Proteus

File Menu. В меню виберемо пункт New Design – створимо новий проєкт, відкриється листок для проєктування електричної схеми (рис. 2.3).

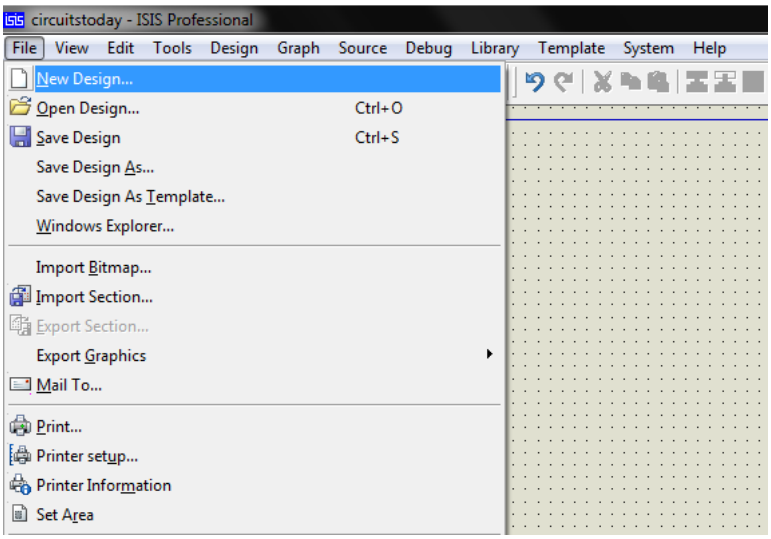


Рис. 2.2. Proteus File Menu

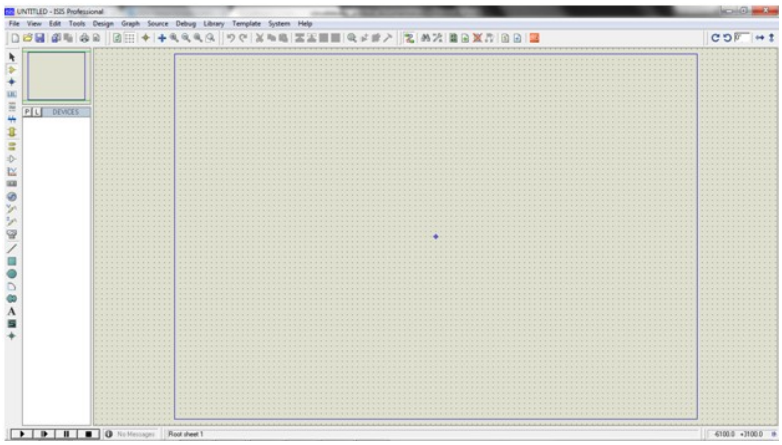


Рис. 2.3. Proteus Design Sheet (робоче поле для проєктування)

Виберемо комплектуючі і розмістимо їх на робочому полі (рис. 2.4 – 2.6).

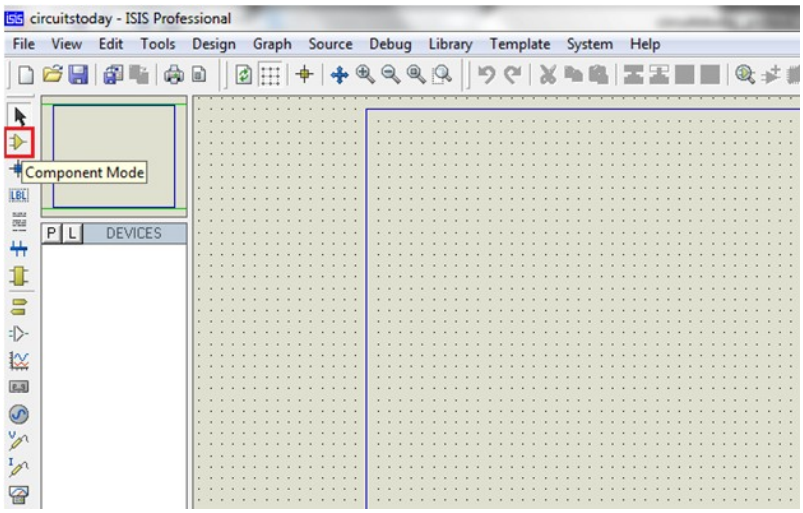


Рис. 2.4. Proteus Component Mode – режим вибору комплектуючих

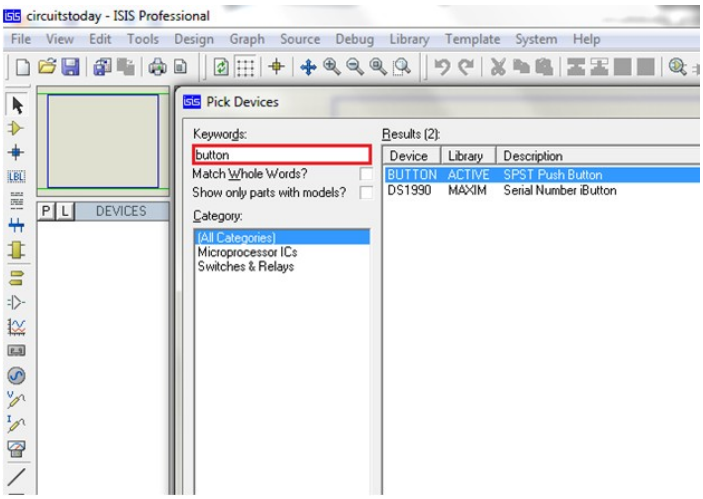


Рис. 2.5. Вибір з бібліотеки компоненти BUTTON – клавіша

Вибрані компоненти відображаються у списку для проєктування.

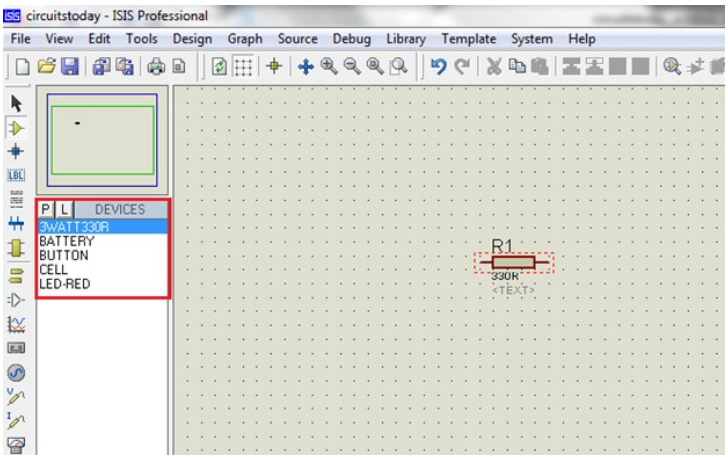


Рис. 2.6. Список вибраних компонент для проєктування

Розмістимо вибрані компоненти на робочому полі і зробимо потрібні з’єднання. Натиснемо клавішу Run для симуляції проєкту (рис. 2.7).

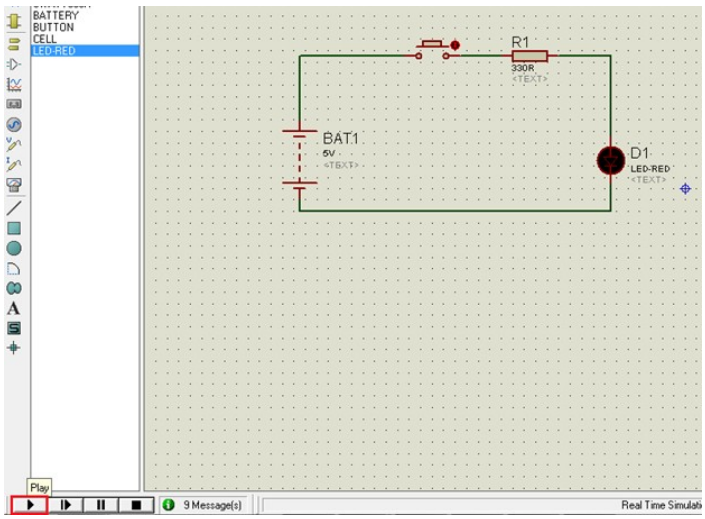


Рис. 2.7. Симуляція проєкту в Proteus ISIS

Симуляцію можна призупинити, завершити, або виконувати по-крокове. В такій послідовності створюють складні проекти – просто і ефективно. Програма PROTEUS має такі можливості:

– COMPIM – модуль для роботи з реальним COM-портом ПК;

– USBCONN –модуль для роботи з USB портом комп'ютера.

Proteus працює з такими середовищами розробки ПЗ як: CodeVision AVR (МК AVR); WinAVR (МК AVR); ICC (МК AVR, msp430, ARM7); HiTECH (МК 8051 і PIC).

2.2. Інструментарій розробки ПЗ в середовищі CodeVisionAVR

Програма CodeVisionAVR – це інтегроване середовище розробки програмного забезпечення для мікроконтролерів AVR від компанії Atmel. CodeVisionAVR складається з GNU GCC компілятора, підтримує мови C, C+ і Objective-C, забезпечує повний цикл розробки для AVR/AVR32 та має інструменти, для написання програм для МК [12-14]. Серед них найбільш вживані: Programmers Notepad (редактор програм текстовий); AVR-LibC (AVR бібліотека МК); GNU Debugger (GNU відладчик).

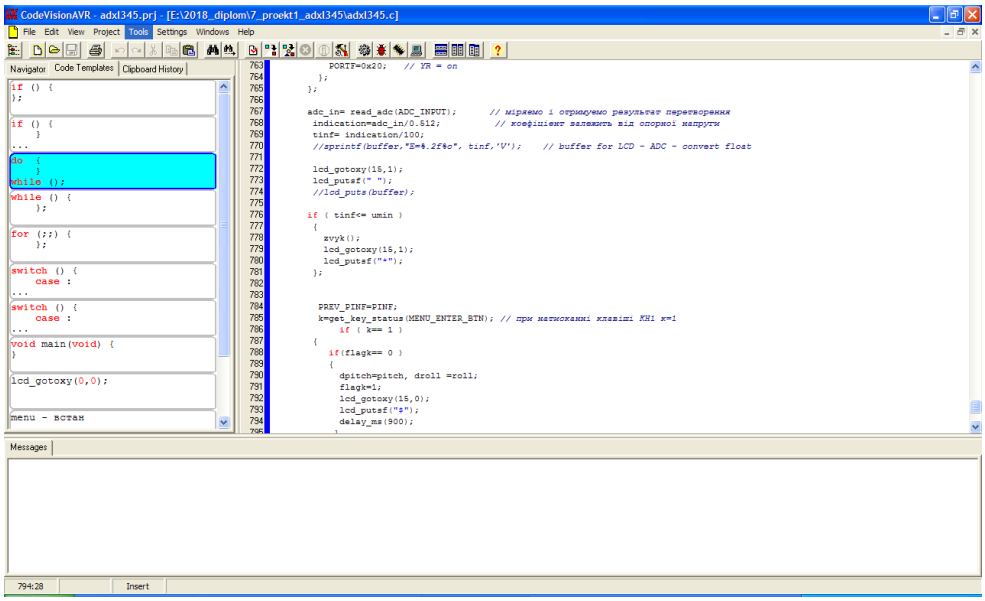


Рис. 2.8. Вікно розробки ПЗ CodeVisionAVR

Програмне забезпечення CodeVisionAVR добре документоване. Результат роботи в програмі CodeVisionAVR можна подати у вигляді HEX,

ROM або BIN-файлу для прямої прошивки МК використавши сторонній програматор. Крім цього можна сформувати формат COFF або OBJ. Число бібліотек CodeVisionAVR росте з кожною новою версією. CodeVisionAVR працює з ПЗ МК Atmel Studio. Всі ці пакети дозволяють розробляти та відлагоджувати програми для МК AVR, PIC та інших [15-22]. Приклади розміщені в папці CodeVisionAVR \cvavr\bin. Характеристики CodeVisionAVR:

- операційні системи: всі ОС MS Windows;
- програмування: мови Асемблер, С;
- С-компілятор безкоштовний (GNU GCC);
- підтримка вбудована в AVR Studio;
- avr-libc – С-бібліотека для МК AVR;
- GNU GCC – безкоштовний С-компілятор;
- MFile – редактор Make-файлу.

Для програмування МК CodeVisionAVR підтримує роботу з програматорами AVRProg (AVR910), Kanda Systems STK200+, STK300, AVRISP MkII, Atmel STK500, AVR Dragon, JTAGICE MkII, Dontronics DT006, Vogel Elektronik VTEC-ISP, Futurlec JRAVR and MicroTronics ATCPU, Mega2000. CodeVisionAVR має бібліотеки для роботи з:

- LCD модулями та шиною Philips I2C;
- сенсорами температури National Semiconductor LM75;
- годинниками реального часу PCF8563, PCF8583, DS1302 і DS1307;
- 1-Wire протоколом Maxim/Dallas Semiconductor. Сенсором температури Maxim/Dallas Semiconductor DS1820, DS18S20, DS18B20.

CodeVisionAVR має автоматичний генератор програм, для написання коду для портів вводу-виводу, ініціалізації лічильників, зовнішніх переривань, роботи з UART (USART), з компаратором, АЦП, інтерфейсами SPI, I2C, 1-Wire шиною, зовнішньою пам’яттю. В бібліотеку CodeVisionAVR також входять функції для роботи зі стрічками, та математичні функції.

РОЗДІЛ 3
АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ УНІВЕРСАЛЬНОГО ПРИСТРОЮ ДЛЯ ТЕСТУВАННЯ ПОВНОКОЛІРНИХ СВІТЛОДІОДНИХ СТРІЧОК НА МК AVR

3.1. Вибір елементної бази для проектування універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок

Елементною базою для проектування універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок взято для розробки:

- МК AVR (ATmega128);
- символьний РКД –16х2. Модель: WH1602B-YGK-CTK;
- світлодіодна повоноколірна стрічка з світлодіодами SMD5050.

Опис мікроконтролера AVR ATmega128. AVR – це сімейство мікроконтролерів розроблених компанією Atmel в 1996 році [23]. Ці мікроконтролери одні з перших, що мають флеш-пам’ять програм та 8-бітну RISC-архітектуру. AVR мікроконтролери мають, хорошу швидкодію, розвинену систему команд, невисоку ціну, використання їх є перспективним.

Є три серії МК AVR: Tiny AVR – невеликі за розмірами МК у 8-вивідному корпусі; Classic AVR-серія МК з продуктивністю до 16 машинних інструкцій за секунду, FLASH-пам’ять програм 2...8 КБ, пам’ять даних EEPROM 64...512 байт, оперативна пам’ять даних SRAM 128 ... 512 байт; та Mega AVR з продуктивністю 4...16 машинних інструкцій за секунду для складних задач, FLASH-пам’ять програм до 128 КБ, пам’ять даних EEPROM 64...512 байт, оперативна пам’ять даних SRAM 2...4 байт. В AVR МК система команд сумісна при перенесенні програм з дешевших на більш потужні і дорожчі мікроконтролери.

Оптимальним для проектування універсального мікроконтролерного пристрою, провівши аналіз співвідношення ціна, продуктивність,

енергоспоживання, а також технічні характеристики було вибрано мікроконтролер ATmega128 фірми Atmel. ATmega128 – це 8-розрядний AVR мікроконтролер з 128 КБ FLASH пам’яттю програм. ATmega128 має RISC архітектуру [24]:

- 133 команди, більшість з яких виконуються за один машинний такт;
- 32 регістра загального призначення;
- продуктивність ATmega128 до 16 MIPS при частоті тактового генератора 16 МГц;
- незалежну пам’ять даних і програм;
- 128 Кбайт внутрішньої FLASH пам’ятті з можливістю циклів перепрограмування до 10000;
- 4 Кбайти EEPROM з можливістю циклів стирання - запису до 100000.
- 4 Кбайт SRAM;
- підтримка відладчика.
- програмування FLASH, EEPROM, фюзів через SPI інтерфейс;
- периферійні модулі: два 8-розрядні таймер/лічильника з подільниками, що програмуються і режимами порівняння;
- два 16-бітних таймера/лічильника з програмованим подільником, і режимами порівняння;
- лічильник реального часу з програмованим генератором;
- шість каналів ШІМ з програмованою роздільною здатністю до 16 біт;
- 8-канальний, 10-ти бітний АЦП;
- програмований USART;
- Master/Slave SPI послідовний інтерфейс;
- вбудований аналоговий компаратор;
- внутрішній калібрований RC-генератор;
- програмований Watchdog-таймер;
- зовнішні та внутрішні джерела переривань;

– шість режимів роботи: не діяльний, зменшення шумів АЦП, режим очікування, економний, режим виключення, режим розширеного очікування;
– 53 програмованих ліній вхід/вихід;
– робоча напруга живлення: 2,7В до 5,5В для ATmega128L; або 4,5В до 5,5В для ATmega128;
– робоча частота: 0-8 МГц для ATmega128L; 0-16 МГц для ATmega128.

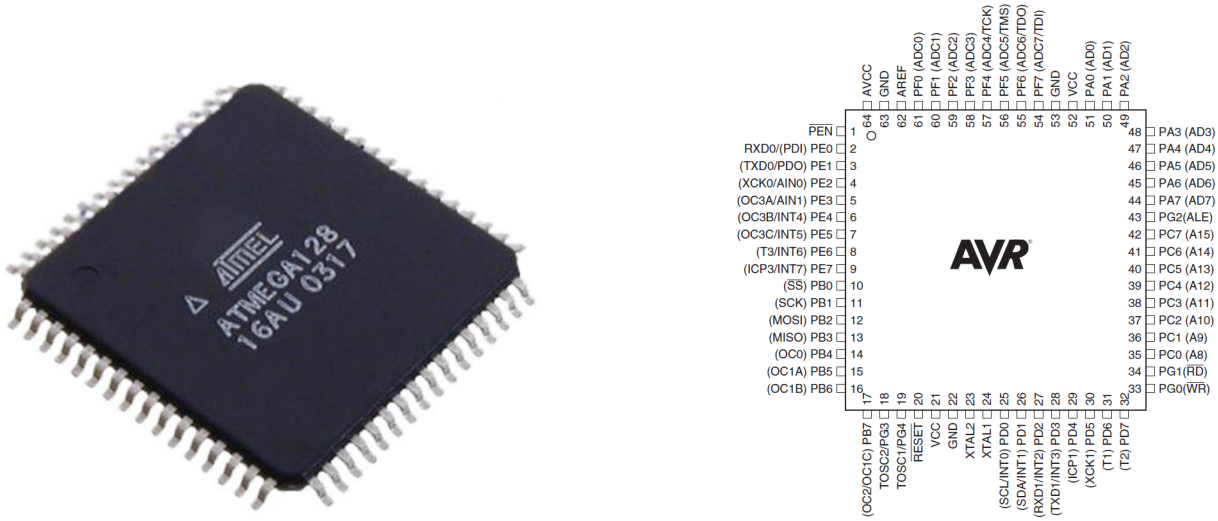


Рис. 3.1. Призначення виводів МК AVR ATmega128 в корпусах MLF і TQFP

AVR має 32 робочі регістри загального призначення та розширений набір команд. Усі 32 регістри підєднані до арифметично-логічним пристроєм, для доступу до двох незалежних регістрів під час виконання команди за один машинний такт. Завдяки такій архітектурі досягнуто найвищу швидкодію виконання коду і відповідно продуктивність в 10 разів більша за CISC мікроконтролер. У холостому режимі (Idle) ЦПУ не функціонує.

ATmega128 має режим зменшення шумів АЦП, для цього в сплячому режимі функціонує тільки асинхронний таймер і АЦП. У режимі відключення, процесор зберігає вміст всіх регістрів, вимикає генератор тактових сигналів, зупиняє всі модулі до приходу переривання зовнішнього або надходження команди Reset. У режимі очікування – функціонує тільки один тактовий генератор. Завдяки швидкому переходу в нормальний режим роботи по перериванню ATmega128 ефективно працює за різних умов і має мале

енргоспоживання. У розширеному режимі очікування в робочому стані знаходяться асинхронний і основний генератор. FLASH пам'ять можна перепрограмувати через послідовний SPI-інтерфейс стандартним програматором. Блок-схема МК AVR ATmega128 зображена на рис. 3.2. Поєднання 8-розрядної RISC-архітектури ЦП і FLASH пам'яті забезпечують для ATmega128 хорошу продуктивність та економічність у мікроконтролерних системах управління.

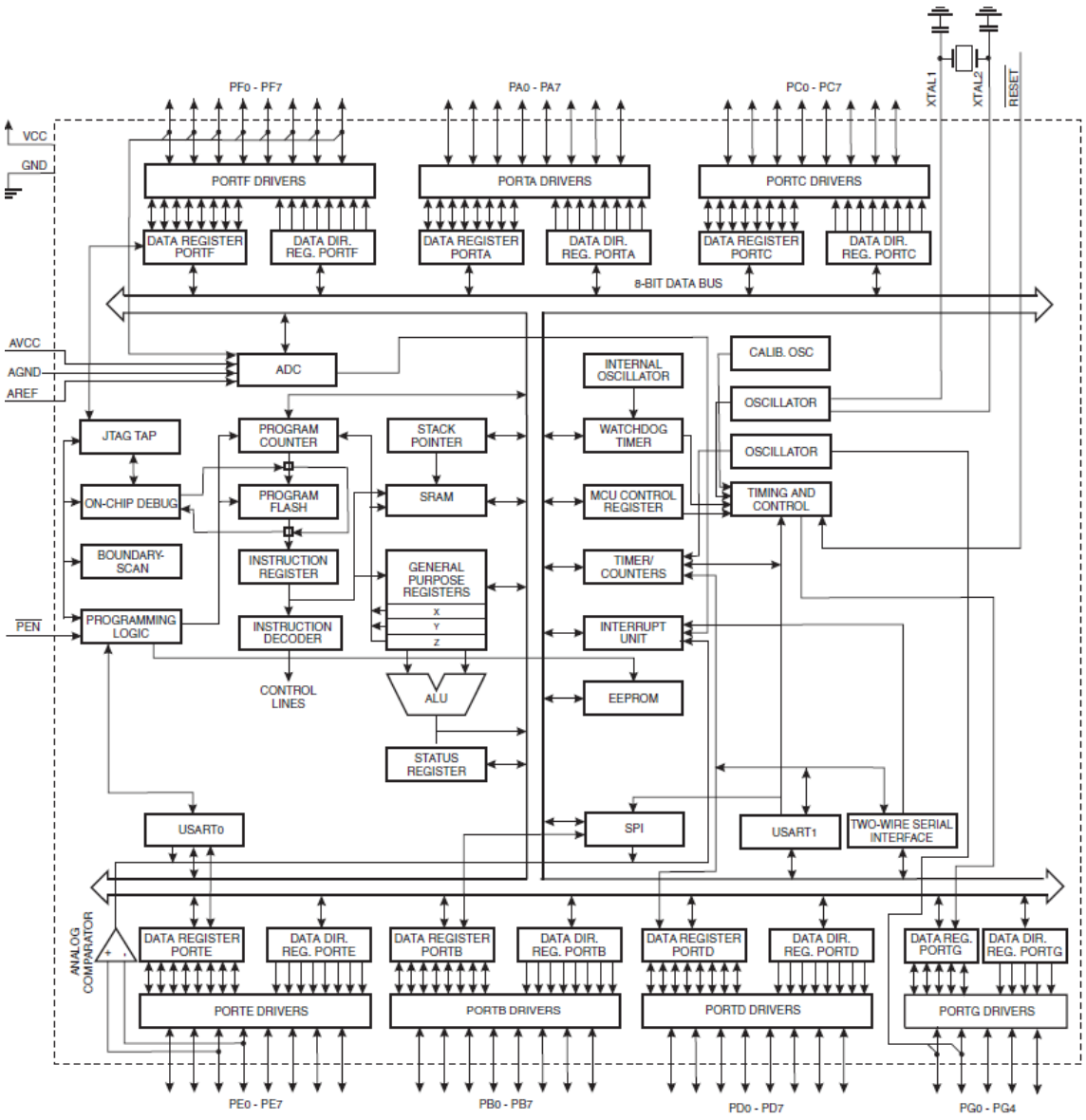


Рис. 3.2. Блок-схема МК ATmega128

Аналого-цифровий перетворювач МК ATmega128. МК обмінюється цифровою інформацією за допомогою портів вводу/виводу. Цифрові сигнали - логічний нуль або логічна одиниця. Для МК ATmega128 при напрузі живлення

5 В лог. “0” - це 0 до ≈1,5В, а логічна одиниця - від ≈1,9 до 5В. При потребі вимірювати напругу аналогових сигналів, МК AVR має аналого-цифровий перетворювач (АЦП) (рис. 3.3).

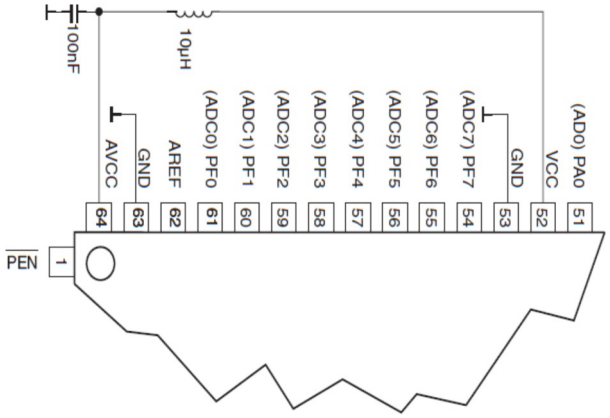


Рис. 3.3. Входи АЦП МК ATmega128

В ATmega128 можна підєднати АЦП до будь-якої ноги порту F (PORTF) Для роботи з АЦП програмують наступні регістри:

- ADCH – старший байт результату перетворення;
- ADCL – молодший байт результату перетворення;
- ADCSRA – регістр А управління та стану;
- ADMUX – регістр управління мультиплексором;
- SFIOR – регістр спеціальних функцій.

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Таблиця 3.1. Вибір опорного джерела живлення для АЦП

REFS1	REFS0	Джерело опорної напруги ADC
0	0	Внутрішнє VREF відключено. Зовнішнє джерело, підключено до AREF
0	1	AVCC із зовнішнім конденсатором на виводі AREF.
1	0	резерв
1	1	джерело опорної напруги 2,56 В - внутрішнє із зовнішнім конденсатором на виводі AREF.

Джерело опорної напруги (AREF) задає діапазон АЦП перетворення. Якщо рівень сигналу більше AREF, то результат перетворення буде 0x3FF. Опорну напругу задають через AREF або AVCC. Внутрішня опорна напруга 2,56 В виробляється внутрішнім джерелом. Зовнішній вивід AREF з’єднаний

безпосередньо з АЦП, для зменшення впливу шумів на опорне джерело підключають конденсатор між виводами AREF і спільним виводом (землею). Напругу AREF можна виміряти на виводі AREF високоомним вольтметром.

АЦП перетворення це 10-бітний код, який розміщений в парі ADCH і ADCL регістрів даних АЦП. Результат перетворення розміщується в молодших 10-ти розрядах 16-бітного слова (вирівнювання справа), або може бути розміщений в старших 10-ти розрядах (вирівнювання ліворуч) для цього треба встановити біт ADLAR в регістрі ADMUX.

Таблиця 3.2. Регістри даних АЦП

Регістри даних ADC – ADCL і ADCH

ADLAR=0:

Розряд	15	14	13	12	11	10	9	8	
	-	-	-	-	-	-	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
читання/запис	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Вих. значення	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR=1:

Розряд	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	-	-	-	-	-	-	ADCL
	7	6	5	4	3	2	1	0	
читання/запис	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Вих. значення	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Таблиця 3.3. Вибір каналу АЦП та коефіцієнта підсилення

MUX4..0	Недиференційний вхід	Неінвертуючий диференційний вхід	Інвертуючий диференційний вхід	Коефіцієнт підсилення Ку
00000	ADC0	Диференційних входів і підсилення немає		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			

00101	ADC5			
00110	ADC6			
00111	ADC7			
01000	Недиференційних входів немає	ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010(1)		ADC0	ADC0	200x
01011(1)		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110(1)		ADC2	ADC2	200x
01111(1)		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010		ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x
11101		ADC5	ADC2	1x
11110	1,22 В (VBG)	Диференційних входів і підсилення немає		
11111	0 В (GND)			

Аналоговий ввід і диференційне підсилення вибираються записом 1-біту MUX в регістр ADMUX. Однополярний аналоговий вхід АЦП можуть бути входи ADC0 ... ADC7, або GND та вихід джерела опорної напруги 1,22 В. В режимах диференційного вводу є можливість вибору інвертуючих і неінвертуючих входів диференційного підсилювача. При виборі диференційного режиму аналогового вводу, диференційний підсилювач підсилює різницю напруг між вибраною парою входів помножену на заданий коефіцієнт підсилення. Підсилене значення поступає на вхід АЦП.

Біти ADCSRA регістра та їх призначення:

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Таблиця 3.4. Управління подільником АЦП

ADPS2	ADPS1	ADPS0	Коефіцієнт ділення

0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

АЦП має дільник, який працює від тактової частоти МК. Коефіцієнт поділу можна змінювати від 2 до 128. Коефіцієнт ділення встановлюється бітом ADPS в регістрі ADCSRA. Рахунок подільник починається з моменту включення АЦП встановленням біта ADEN в регістрі ADCSRA. Подільник працює якщо біт ADEN=1 і вимкнутий при ADEN=0. Для досягнення максимальної роздільної здатності (10 розрядів), частота дискретизації встановлюється в діапазоні 50...200 кГц. Нормальне перетворення потребує 13 тактів АЦП. Перше перетворення при включенні АЦП (ADEN=1 в ADCSRA) потребує 25 тактів синхронізації. По завершенні перетворення результат розміщується в регістрах даних АЦП і прапорець ADIF=1. У режимі разового перетворення одночасно скидається біт ADSC. В автоматичному режимі нове перетворення починається відразу після завершення попереднього перетворення, при цьому ADSC залишається рівне лог. “1”.

Біт ADSC: Разове перетворення стартує при записі лог. “1” в біт запуску перетворення АЦП ADSC. Цей біт залишається рівний лог.одиниці в процесі перетворення і встановлюється в лог нуль по завершенні цього перетворення. У режимі автоматичного перезапуску АЦП обробляє аналоговий сигнал і поновлює регістр даних АЦП. Цей режим можна задати записом лог. “1” в біт ADFR регістра ADCSRA. Перше перетворення відбувається при записі лог. “1” в біт ADSC регістра ADCSRA. У цьому режимі АЦП виконує послідовні перетворення, незалежно від того чи скидається прапорець переривання АЦП ADIF.

Біт ADATE: При записі лог. 1, ADC перейде в режим автоматичного перезапуску. ADC автоматично запускає перетворення по передньому фронту сигналу. Для вибору джерела запуску встановити біти ADTS регістра SFIOR.

Біт ADIF: Цей прапорець встановлюється по завершені перетворення ADC і поновлення регістрів даних (ADCH:ADCL). При встановленні бітів ADIE і I (регістр SREG), по завершенні перетворення буде виклик обробника переривання. ADIF- прапорець скидається апаратно при переході по відповідному вектору переривання. Програмно прапорець ADIF скидається шляхом запису в нього лог. 1.

Біт ADIE: Якщо біт рівний лог. 1, та встановлено біт I в регістрі SREG, то дозволене переривання по завершенні перетворення ADC.

Таблиця 3.5. Регістр SFIOR і призначення його бітів ADIS2, ADIS1, ADIS0

Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ADIS2	ADIS1	ADIS0	Джерело запуску перетворення
0	0	0	Неперервне перетворення (Free Running mode)
0	0	1	Аналоговий компаратор
0	1	0	Зовнішній запит на переривання
0	1	1	Timer/Counter0 Compare Match
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

Якщо в регістрі ADCSRA ADATE рівне лог. 1, значення біт ADTS визначає, яке буде джерело для запуску перетворення. Для підключення потенціометрів управління ШІМ можна використати порти АЦП PF0, PF5, PF6. МК ATmega128 має два 16- розрядні таймера Timer/Counter1 і Timer/Counter3. Регістри 16-розрядного таймера можуть містити числа від 0 до 65536 (2^{16}). Основні характеристики цих таймерів наступні:

- використання для генерування ШІМ (PWM);
- використання як генераторів імпульсів;
- наявність трьох вихідних портів ШІМ (OC3A, OC3B, OC3C для таймера3 і OC1A, OC1B, OC1C для таймера1).

16-розрядний Таймер/лічильник3 може використовуватися для формування часових інтервалів, підрахунку кількості зовнішніх синалів і для

генерації сигналів з ШІМ різної шпаруватості і тривалості на виводах OC3A, OC3B, OC3C. Аналогічне використання має Таймер/лічильник1. Блок-схема 16-розрядного Таймера/лічильника наведена на рис. 3.4.

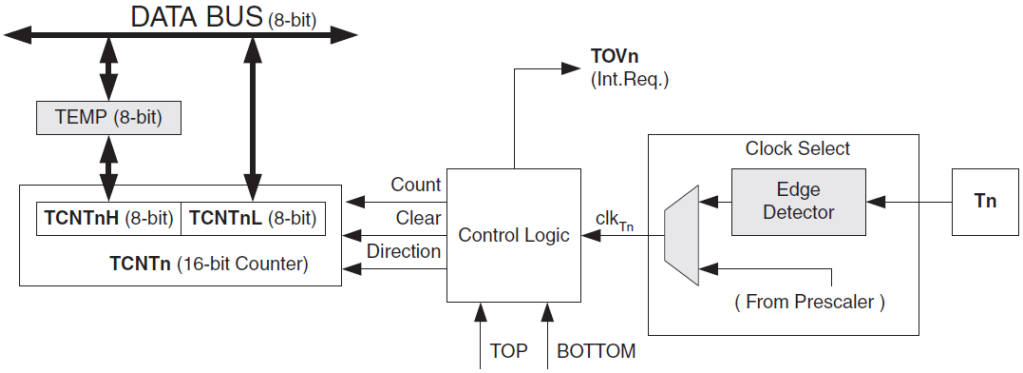


Рис. 3.4. Блок-схема 16-розрядного таймера/лічильника

Сигнали управління наступні:

- Count – зменшення або збыльшення значення регістра TCNTn на 1;
- Direction- вибір зменшення/збільшення;
- Clear – очистити регістр TCNTn (всі біти рівні 0);
- Clk – імпульси синхронізації;
- TOP – сигнал, що TCNTn досягнув свого максимального значення;
- BOTTOM – сигнал, що TCNTn досягнув мінімального значення (нуль).

16-розрядний регістр лічильника займає в пам’яті дві 8-розрядні комірки. TCNTH – містить старший байт і TCNTL – молодший. В ці регістри можна записувати дані і читати їх. При записі першим завантажується старший байт, а далі молодший, при читанні навпаки спочатку читаємо молодший потім старший. Залежно від режиму роботи вміст регіста TCNTn може обнулятися, збільшуватися на 1(інкрементуватися), зменшуватися на 1(декрементуватися) по кожному тактовому імпульсу синхронізації. 16-розрядному Таймер/лічильнику3 можна задати наступні основні режими роботи:

Режим Normal. Найпростіший режим роботи Таймера/лічильника. По кажному імпульсу тактового сигнала відбувається інкремент регістра TCNT3 (збільшення значення на 1). При переході через значення $FFFF модуля лічби

(TOP) виникає переповнення і з наступним тактом починається лічба зі значення $0000, в цей же момент встановлюється прапорець TOV3=1 в регістрі TIFR, і може бути згенероване переривання якщо встановлено прапорець TOIE3=1 в регістрі TIMSK.

Режим CTC (обнулення при співпаданні). В цьому режимі Таймер/лічильник 3 працює по такому ж принципу як в режимі Normal. Відмінність в тому, що максимально можливе значення лічильника TCNT3 обмежене значенням регістра порівняння OCR3A (OCR3B, OCR3C) або ICR3. При досягненні TCNT3 значення OCR3A (OCR3B, OCR3C) або ICR3, значення TCNT3 обнуляється в TCNT3=$0000 В цей же момент встановлюється прапорець TOV3=1.

Режим Fast PWM (швидкодіючий ШІМ). З допомогою цього режиму можна генерувати високочастотний сигал ШІМ. Принцип і порядок роботи не відрізняється від режиму Normal. Основні регістри для роботи з 16- розрядним таймером/лічильником 3 наведено нижче:

Таблиця 3.6. Структура регістра управління A TCCR3A

Bit	7	6	5	4	3	2	1	0	
	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30	TCCR3A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7:6 – COMnA1:0: Налаштування режиму порівняння для каналу A (Compare Output Mode for Channel A);
- Bit 5:4 – COMnB1:0: Налаштування режиму порівняння для каналу B;
- Bit 3:2 – COMnC1:0: Налаштування режиму порівняння для каналу C.

Таблиця 3.7. Структура регістра управління B TCCR3B

Bit	7	6	5	4	3	2	1	0	
	ICNC3	ICES3	–	WGM33	WGM32	CS32	CS31	CS30	TCCR3B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7 – ICNCn: Відфільтрування вхідних шумів. Встановлення цього біту в лог. 1, активує фільтр шуму з входу ICP3;
- Bit 5 – Зарезервовано;

- Bit 4:3 – WGMn3:2: Використовуються разом з бітами регістра TCCR3A для вибору режиму роботи лічильника (табл. 3.8).

Таблиця 3.8. Режими роботи лічильника

Режим	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter режими роботи	TOP	Update of OCRnx at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

- Bit 2:0 – CSn2:0: Вибір джерела імпульсів синхронізації лічильника (табл. 3.9).

Таблиця 3.9. Вибір джерела імпульсів синхронізації лічильника

CSn2	CSn1	CSn0	Пояснення
0	0	0	Джерело синхронізації лічильника не вибрано
0	0	1	$clk_{I/O}$/1 подільник рівний 1
0	1	0	$clk_{I/O}$/8 подільник рівний 8
0	1	1	$clk_{I/O}$/64 подільник рівний 64
1	0	0	$clk_{I/O}$/256 подільник рівний 256
1	0	1	$clk_{I/O}$/1024 подільник рівний 1024
1	1	0	Зовнішнє джерело на Tn pin. Перемикання по спаду
1	1	1	Зовнішнє джерело Tn pin. Перемикання по фронту

Таблиця 3.10. Структура регістрів порівняння OCR3AH і OCR3AL, OCR3BH і OCR3BL, OCR3CH і OCR3CL

Bit	7	6	5	4	3	2	1	0	
	OCR3A[15:8]								OCR3AH
	OCR3A[7:0]								OCR3AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
	OCR3B[15:8]								OCR3BH
	OCR3B[7:0]								OCR3BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
	OCR3C[15:8]								OCR3CH
	OCR3C[7:0]								OCR3CL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Таблиця 3.11. Структура регістра захоплення ICR3H і ICR3L

Bit	7	6	5	4	3	2	1	0	
	ICR3[15:8]								ICR3H
	ICR3[7:0]								ICR3L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Регістр захоплення ICR3H і ICR3L може використовуватися для задання TOP-значення (режим 14, табл. 3.8). В табл. 3.12 наведено сигнали на виході порта PE (OC3A, OC3B, OC3C) для режиму Fast PWM в залежності від встановлених бітів в регістрі управління A TCCR3A (табл. 3.6). Для ШІМ управління повноколірною світлодіодною стрічкою використано порти PE3, PE4, PE5, а для ШІМ управління звуковою сигналізацією порт PB5.

Таблиця 3.12. Режим Fast PWM, сигнали на виході PE (OC3A, OC3B, OC3C)

COMnA1/COMnB1/ COMnC1	COMnA0/COMnB0 /COMnC0	Пояснення
0	0	Нормальна робота порта. OCnA/OCnB/OcnC від'єднано
0	1	WGMn3:0=15: Переключити OCnA при співпадінні порівнянь, OCnB/OCnC від'єднано (нормальна робота порта). Для всіх інших налаштувань WGMn, нормальна робота порта, OCnA/OCnB/OCnC від'єднано
1	0	Очистити OCnA/OCnB/OCnC при

		співпадінні порівнянь, встановити OCnA/OCnB/OCnC при BOTTOM (неінвертуючий режим)
1	1	Встановити OCnA/OCnB/OCnC при співпадінні порівнянь, очистити OCnA/OCnB/OcnC при BOTTOM (інвертуючий режим)

Аппаратна реализація ШІМ має безумовні переваги перед програмною, так як розвантажує процесор від додаткового громіздкого коду, і вивільняє час на обслуговування ШІМ. Для цього треба провести ініціалізацію таймер/лічильника (занести необхідні значення в регістри таймера/лічильника) як він може працювати незалежно від процесора, отже процесор може виконувати інші задачі, таймер/лічильник тільки деколи, в потрібний момент, буде звертатись до процесора для коректування, або зміни режиму.

Опис та вивід значень на рідкокристалічний дисплей (РКД). РКД – це недороге рішення для виводу візуальної інформації про стан системи, при цьому забезпечується відображення достатній об’єм інформації при невеликому електроспоживанні.

РКД – мають підсвітку, що дозволяє їх експлуатацію в умовах з низької освітленістю, дозволяє експлуатацію, в переносній, польовій і в бортовій апаратурі. Рідкокристалічний модуль має контролер управління і РК панель. Модулі випускаються із світлодіодною підсвіткою. На рис. 3.5 зображено символьний жовто-зелений РКД 16x2 [25]. Модуль дає можливість відобразити 2 рядки з 16 символів. Символи відтворюються в матриці 5x8 точок. Між символами встановлено відступ шириною в одну крапку.

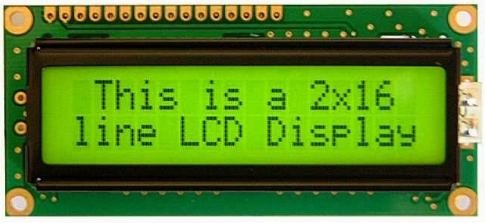


Рис. 3.5. Символьний жовто-зелений РКД – 16x2

WH1602B-YGK-CTK – РКД відтворює чорні символи на жовтозеленому фоні. Технічні характеристики WH1602B-YGK-CTK:

- 2 рядки по 16 символів;
- контролер сумісний з HD44780;
- підсвітка;
- жовто-зелений колір.

Символу в РКД відповідає код в ОЗП - комірці модуля. Модуль має два види пам’яті – це коди символів і знакогенератор користувача, а також логіку управління РК-панеллю. Технічні характеристики модуля:

- програмно-комутовані 2-ві сторінки знакогенератора (алфавіт: кирилиця та англійський);
- підєднується як по 8-ми, і по 4-х бітній шині даних (програмується при ініціалізації);
- отримує команди з шини даних;
- читає та записує дані в ОЗП з шини даних;
- дозволяє читати статус стану на шину даних;
- памятати до 8-ми зображень символів користувача;
- виводить блимаючий (або не блимаючий) курсор;
- керує контрастом і підсвіткою.

Програмування і управління. Управління контролером ведеться через интерфейс системи. Основними об’єктами для роботи є регістри DR і IR. Вибір регістра відбувається по лінії RS, при RS=0 – адресується регістр команд (IR), при RS=1 – адресується регістр даних (DR).

Відеопам’ять, має об’єм 80 байтів, вона призначена для зберігання кодів символів, які відображаються на РКД. Відеопам’ять має два рядки по 40 символів у кожному. Незалежно, скільки реальних рядків буде мати конкретний РКД, адресація відеопам’яті відбувається завжди як до двох рядків по 40 символів. Контролер виріб з динамічною індикацією, періодично проводить поновлення інформації на РКД. Контролер HD44780 має набір

внутрішніх прапорців, які визначають режим работи різних елементів контролера.

Під'єднання модуля РКД. У CodeVisionAVR є бібліотека для роботи з такими РКД – модулями. Можна працювати з форматами РКД: 1x8, 2x12, 3x12, 1x16, 2x16, 2x20, 4x20, 2x24 і 2x40 символів. Схема підєднання РКД до до МК ATmega128 на рис. 3.6.

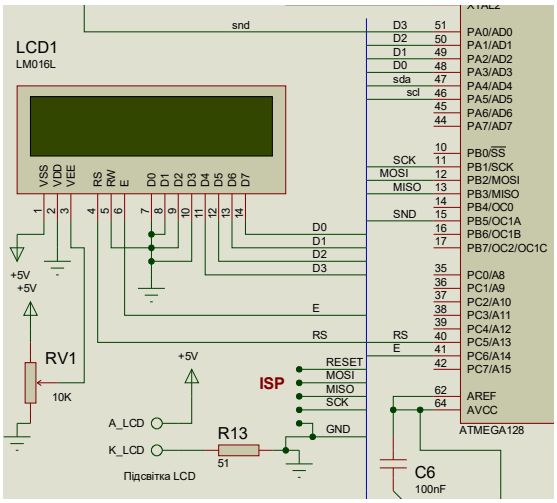


Рис. 3.6. Схема підключення РКД до МК ATmega128

Опис світлодіодної повоноколірної стрічки з світлодіодами SMD5050. На рис. 3.7. зображено світлодіодну стрічку з світлодіодами SMD5050 [26].

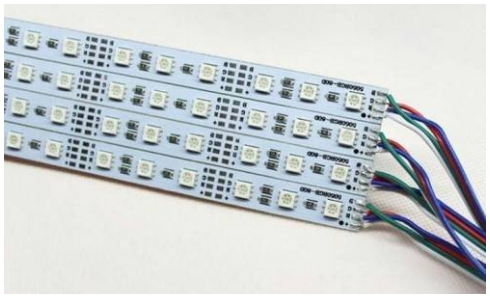


Рис. 3.7. Повноколірна світлодіодна стрічка

Стрічка має 72 світлодіоди SMD5050 на метр довжини, негерметична IP20, відноситься до серії "стандарт", має високу яскравість (850-1000 люмен метр) і строк служби до 35000 годин. Стрічка виконана на алюмінієвій полосі товщиною 1мм, що забезпечує хороший відвід тепла. Живлення 12 вольт (постійний струм). Виготовлення стрічки зі спільним анодом. Один метр стрічки споживає не більше 17,5 Вт. Робоча температура: від -30 до +60 °С.

Управління кольором і яскравістю здійснюється по чотирьох проводах один з яких + живлення, а три інші імпульси певної частоти і тривалості.

3.2. Проєктування універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок в САПР Proteus VSM

Розроблюваний універсальний мікроконтролерний пристрій повинен довільним чином регулювати колір світла світлодіодної стрічки з світло діодами трьох базових кольорів – червоного (R), зеленого (G) та синього (B). Пристрій розрахований для роботи з повноколірною світлодіодною стрічкою зі спільним анодним виводом, але може бути налаштований до стрічок зі світло діодами, що мають спільний катод шляхом заміни силового модуля.

Встановлення світла потрібного кольору RGB світлодіодної стрічки повинно відбуватися зміною яскравості її складових компонент R, G і B за рахунок зміни тривалості імпульсів живлення відповідних світлодіодів при сталій частоті їх повтору. Частоту імпульсів можна змінювати в широких межах від 1 Гц до 30 КГц. Для кожної компоненти передбачити по 256 рівнів зміни тривалості імпульсів і відповідно її яскравості у всьому діапазоні зміни частоти. Можливість запам’ятати колір стрічки і потім його відтворити. Встановлення кольору світлодіодної стрічки забезпечити трьома потенціометрами. Відсотки змішування кольорів вивести на рідкокристалічний дисплей. На дисплей також вивести частоту з якою відбувається живлення всіх RGB світлодіодів стрічки.

Універсальний мікроконтролерний пристрій повинен мати меню - чотири клавіші для для вибору пунків меню та налаштування їх параметрів. На рис. 3.8 зображено запропоновану структуру універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок. Виріб включає в себе апаратне та програмне забезпечення [27].

Апаратне забезпечення універсального пристрою включає:

- МК AVR (ATmega128), що є основним модулем обробки даних;
- клавіші налаштувань параметрів пристрою;

– РКД для виводу поточної інформації;
– силового модулю управління стрічкою;
– повноколірної світлодіодної стрічки для тестування;
– зумера для звукових повідомлень.

Структура
універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок

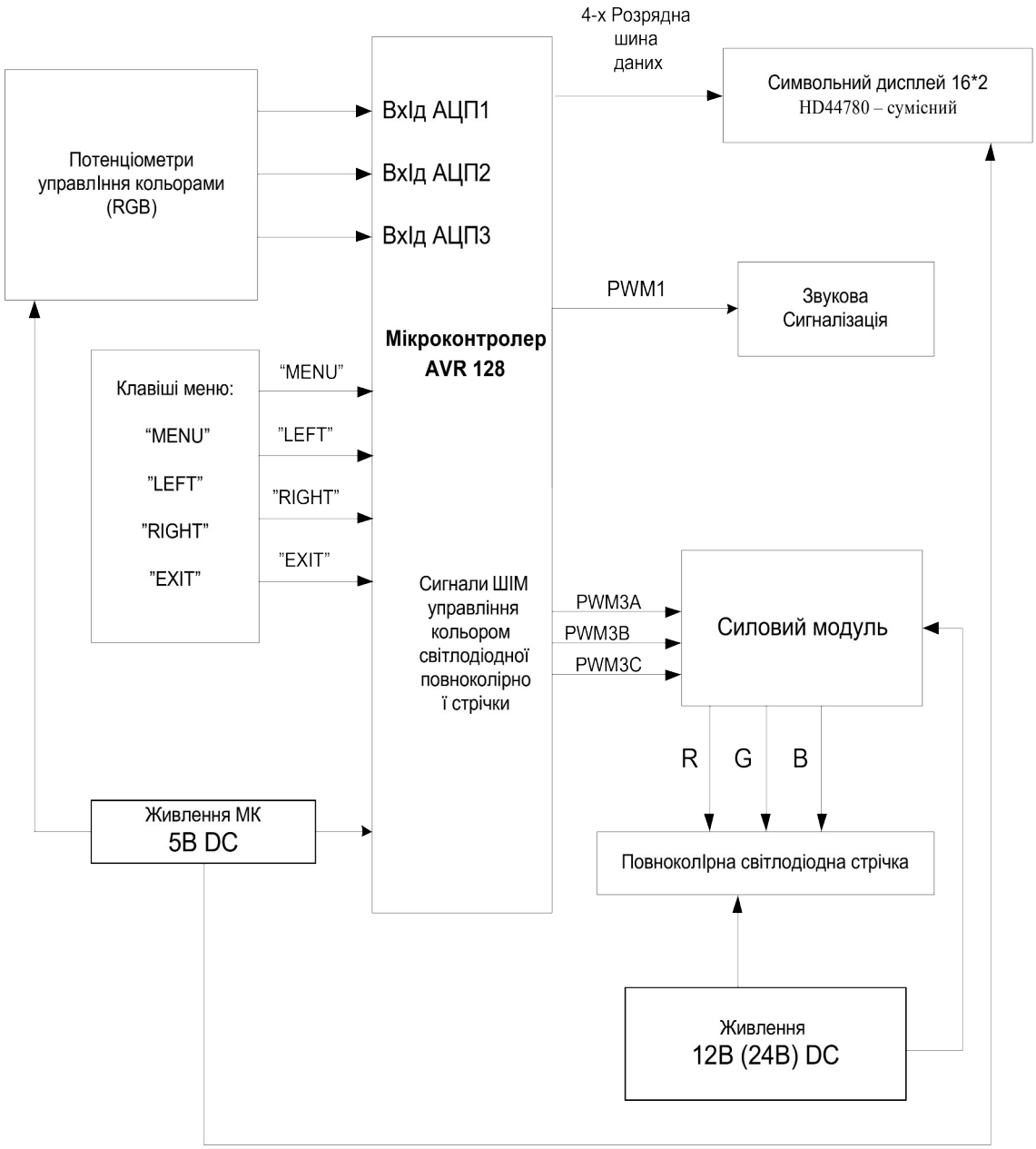


Рис. 3.8. Загальна структура універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок

Мікроконтролер ATmega128 керує повноколірною світлодіодною стрічкою, контролює напруги на потенціометрах управління, натискання клавіш меню і видає необхідні ШІМ заданої частоти сигнали на силовий модуль управління стрічкою, та виводить інформацію на РКД. На РКД можна спостерігати тривалість імпульсу у відсотках для кожного каналу RGB. Клавіші служать для вибору режимів роботи пристрою, встановлення необхідної частоти ШІМ, тестування, зберігання налаштувань в ППЗУ. Потенціометри дають можливість в реальному масштабі часу регулювати всі три кольори стрічки. Силовий модуль призначений для отримання необхідної потужності та гальванічної розв'язки модуля управління від силових напруг. Ланцюги світлодіодів стрічки комутують електронні ключі на польових транзисторах. Ключами управляють імпульси, які формує мікроконтролер.

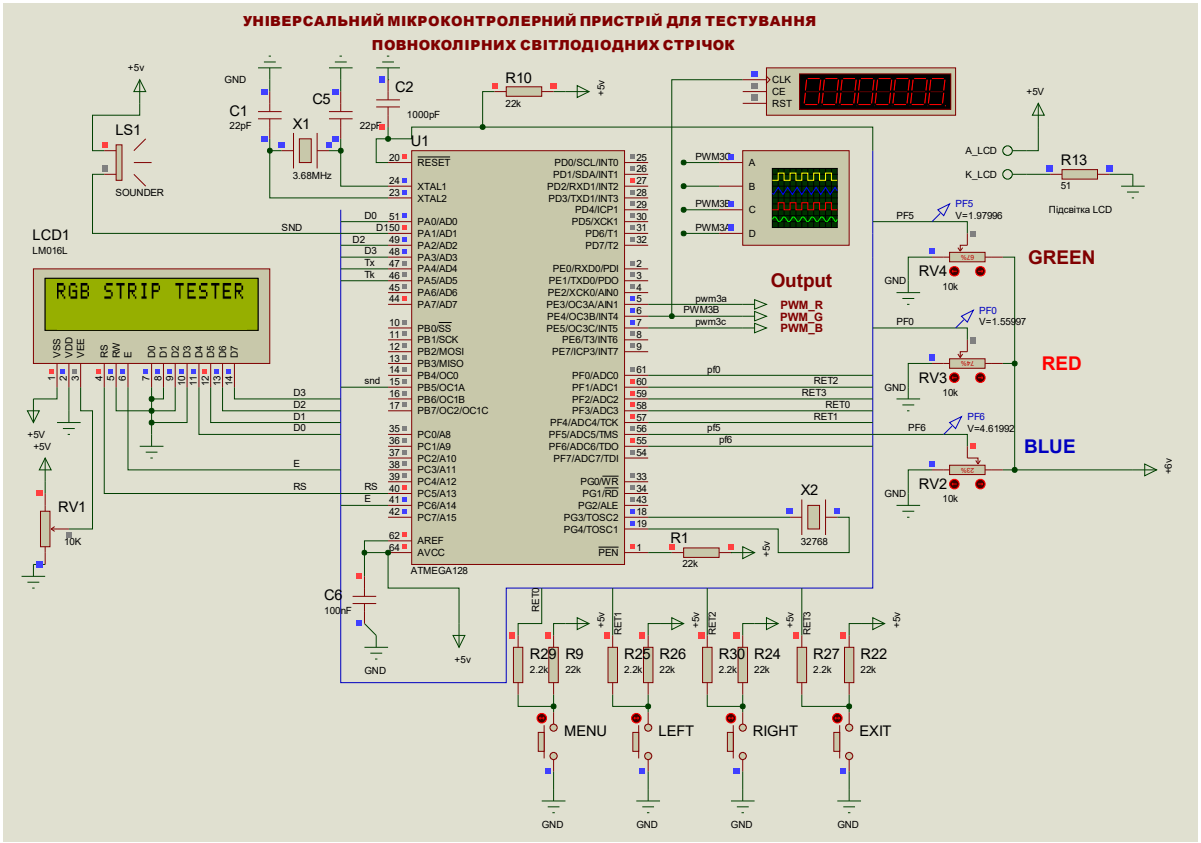


Рис. 3.9. Апаратне забезпечення універсального МК пристрою для тестування RGB світлодіодних стрічок спроєктоване в САПР Proteus VSM

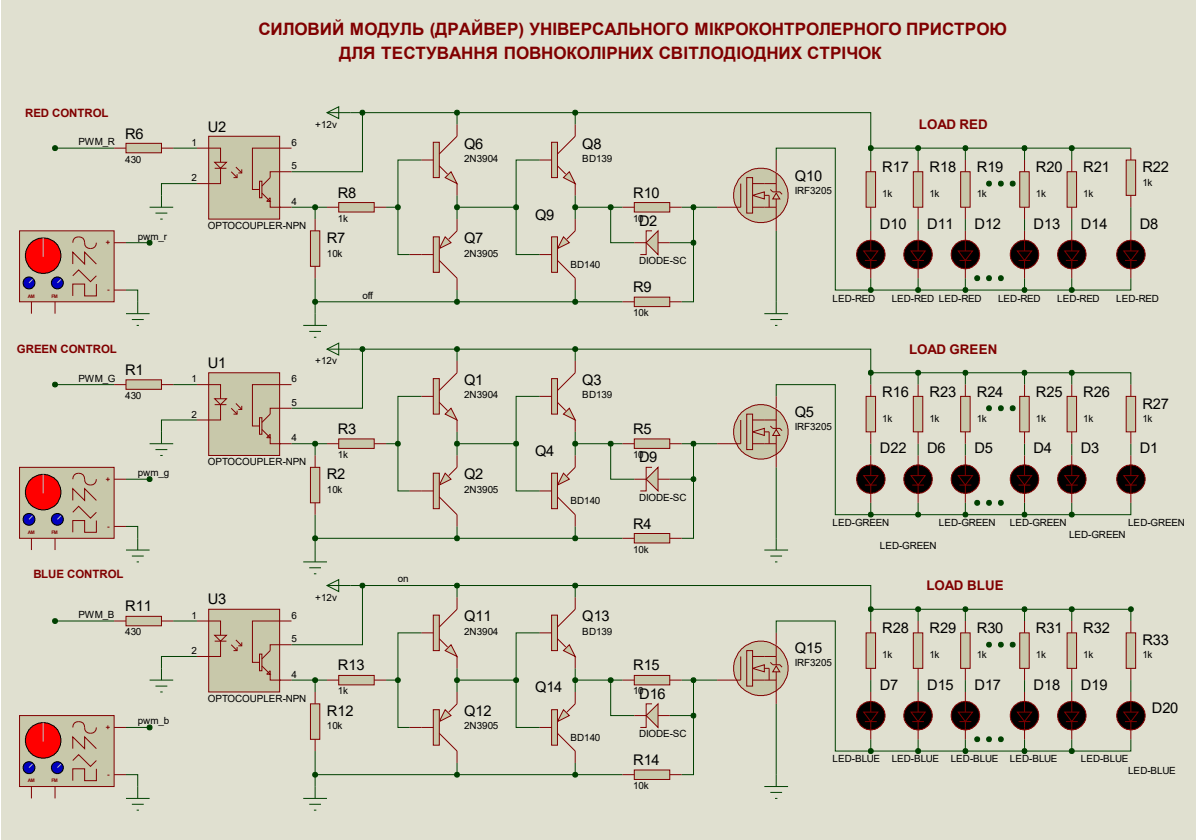


Рис. 3.10. Апаратне забезпечення універсального МК пристрою для тестування RGB світлодіодних стрічок спроєктоване в САПР Proteus VSM

МК U1 обробляє виміряні напруги на потенціометрах RV2, RV3, RV4, обчислює шпаруватість для кожного ШІМ каналу PWM3A, PWM3B, PWM3C, враховуючи налаштування зроблені в меню, виводить інформацію про частоту імпульсів, їх період і тривалість по кожному силовому каналу зокрема на дисплей. На рис. 3.9 маємо проєкт апаратної частини пристрою для тестування повноколірних світлодіодних стрічок в САПР Proteus VSM. Для роботи з меню налаштувань служать клавіші MENU, LEFT, RIGHT, EXIT, під'єднані до виводів порта F (PF1, PF2, PF3, PF4). РКД під'єднано до контактів порта A та порта C. Вхід АЦП PF7 використано для контролю напруг на потенціометрах RV2, RV3, RV4. Звуковий сигналізатор LS під'єднано до виходу ШІМ OC1A порта PB5. Для симуляції роботи пристрою використано віртуальний 4-х канальний осцилограф, та частотомір. Виконавчий силовий модуль під'єднаний до порту PE (PWM3A, PWM3B, PWM3C). До силового модуля під'єднана повно колірна світлодіодна стрічка (LOAD RED, LOAD GREEN,

LOAD BLUE). Силовий модуль універсального МК пристрою для тестування повноколірних світлодіодних стрічок призначений для отримання необхідної потужності та гальванічної розв'язки модуля управління від силових напруг. Силовий модуль має три однакових канали призначених для комутації червоної, зеленої та синьої компоненти повноколірних світлодіодних стрічок. Входи управління каналів позначені відповідно RED CONTROL, GREEN CONTROL та BLUE CONTROL. На ці входи подаються широтно-імпульсні сигнали (ШІМ) безпосередньо з мікроконтролера. Робота всіх трьох каналів аналогічна, тому розглянемо роботу одного з них, наприклад верхнього, що управляє червоною компонентою світлодіодної стрічки. ШІМ сигнал поступає на вхід оптопари U2, яка виконує роль гальванічної розв'язки модуля управління від силових напруг. Живлення МК, як правило 5 В, а світлодіодної стрічки від 12 В та більше. З виходу оптопари U2 сигнал рівнем +12 В поступає на малопотужні транзистори Q6, Q7 і далі на потужну комплементарну пару транзисторів Q8, Q9, які здатні комутувати імпульсні струми 3..5 А, що можуть виникати на затворі силового польового транзистора Q10 при швидкому його перемиканні. В сток транзистора Q10 під'єднано червоні світлодіоди повноколірної світлодіодної стрічки – навантаження LOAD RED. Навантаження LOAD GREEN – це зелені світлодіоди під'єднано до стоку транзистора Q5 каналу GREEN CONTROL і навантаження LOAD BLUE до стоку транзистора Q15 каналу BLUE CONTROL.

Силовий модуль має хороші динамічні характеристики завдяки діодам Шоткі D2, D9, D16 які швидко знімають заряд з затвору польових транзисторів таким чином отримують потужні імпульси з крутими фронтами. Робота силового модуля перевірена в емуляторі Proteus ISIS. Для перевірки використані інструментальні засоби Proteus VSM – три генератора імпульсів, що генерують сигнали PWM_R, PWM_G, PWM_B. Результати перевірки зображено на рис. 3.9.

РОЗДІЛ 4
ПРОГРАМНО-АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ УНІВЕРСАЛЬНОГО МІКРОКОНТРОЛЕРНОГО ПРИСТРОЮ ДЛЯ ТЕСТУВАННЯ ПОВНО КОЛІРНИХ СВІТЛОДІОДНИХ СТРІЧОК

4.1. Алгоритм роботи універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок

При подачі живлення починається виконання вбудованого ПЗ в МК AVR. Ініціалізуються внутрішні програмні змінні, та починається ініціалізація сенсорів. Система налаштовує АЦП , ШІМ вихід для управління пищиком LS, виходи ШІМ для управління каналами світлодіодної стрічки а також РКД для роботи по 4р. шині D0..D3. МК аналізує натискання клавіш меню.

Ініціалізація РКД відбувається функцією lcd_init(). По завершенні ініціалізації, програма виводить на РКД текст-повідомлення “RGB STRIP TESTER” (Тестер RGB стрічки) із затримкою на 1,5 с.

Далі розпочинається цикл вимірювань напруг на входах АЦП, на які підєднано потенціометри управління шпаруватістю та зчитування їх в МК, обробка виміряних значень, формування частоти імпульсів в залежності від налаштувань в меню, аналіз натиснення клавіш меню, вивід імпульсів ШІМ управління каналами RGB, вивід результатів вимірювань на РКД. Вивід сигналів управління (відповідно до алгоритму роботи), звукової індикації.

Особлива увага при розробці програмного забезпечення приділена написання зручного меню. Налаштування, зміна параметрів, вибір тестів реалізується з допомогою всього чотирьох клавіш. Алгоритм роботи меню показано на рис. 4.1.

Алгоритм роботи універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок.

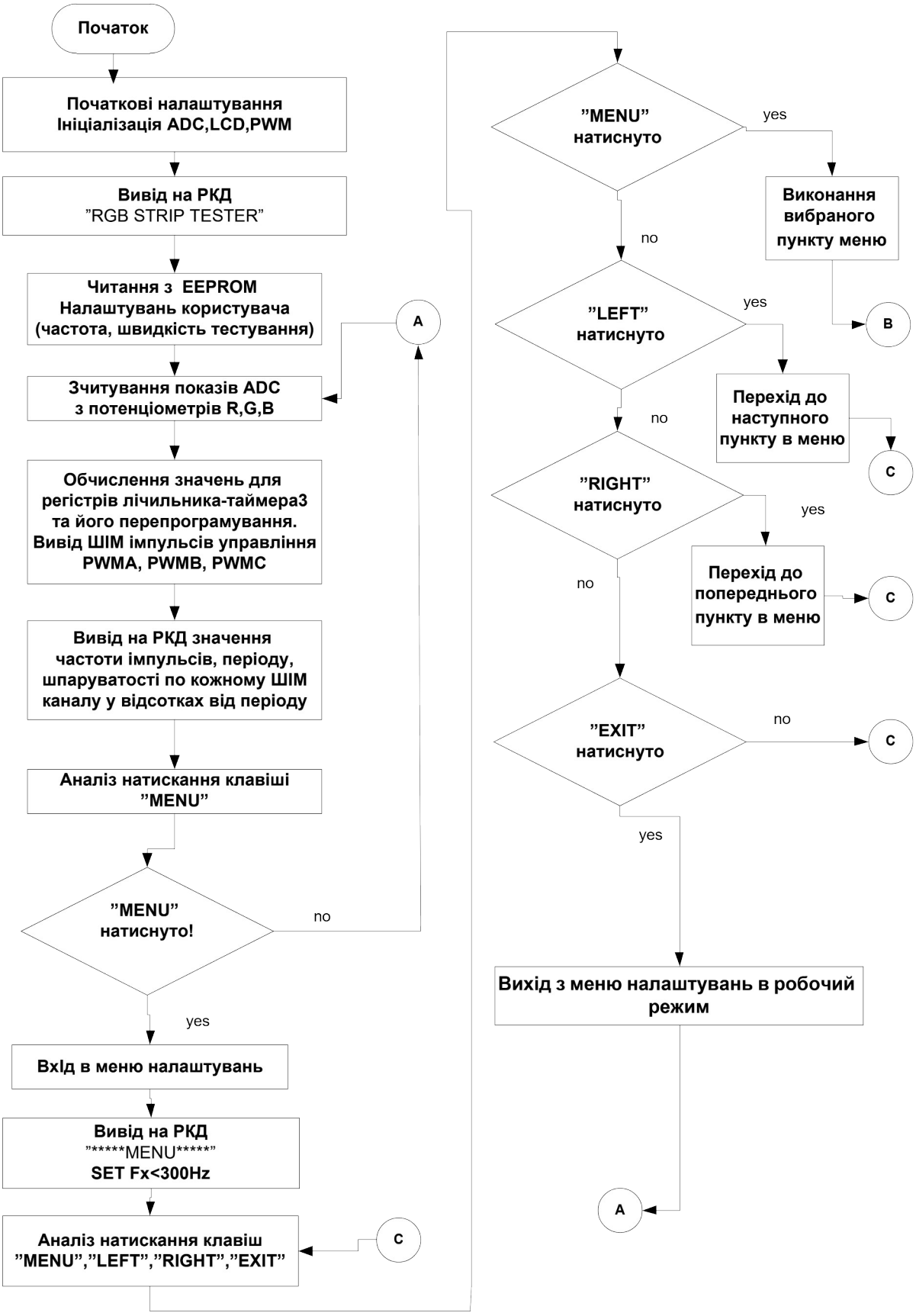


Рис. 4.1. Алгоритм роботи пристрою (початок)

Алгоритм роботи універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок (продовження)**.**

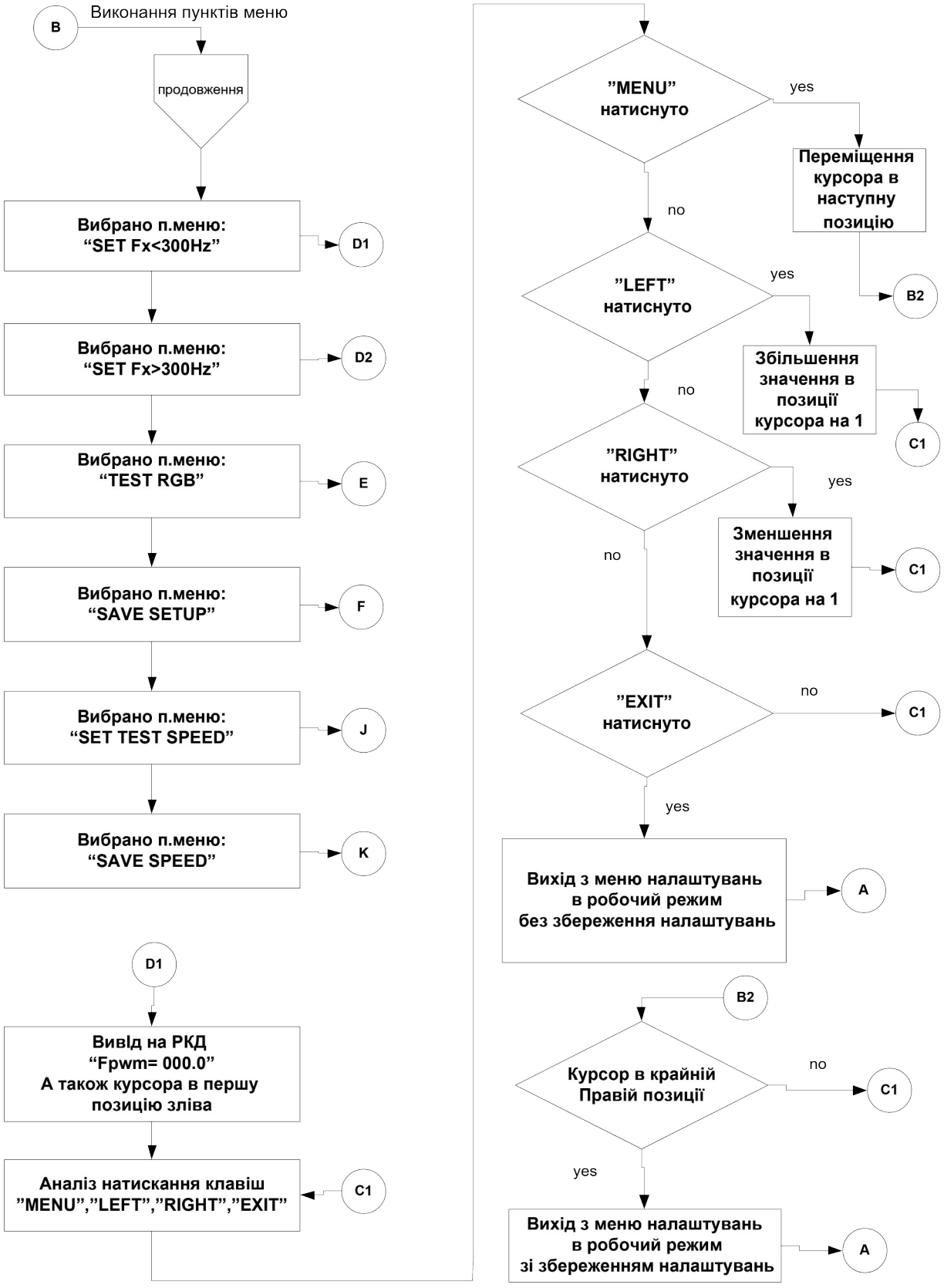


Рис. 4.2. Алгоритм роботи пристрою (продовження)

Алгоритм роботи універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок (продовження)**.**

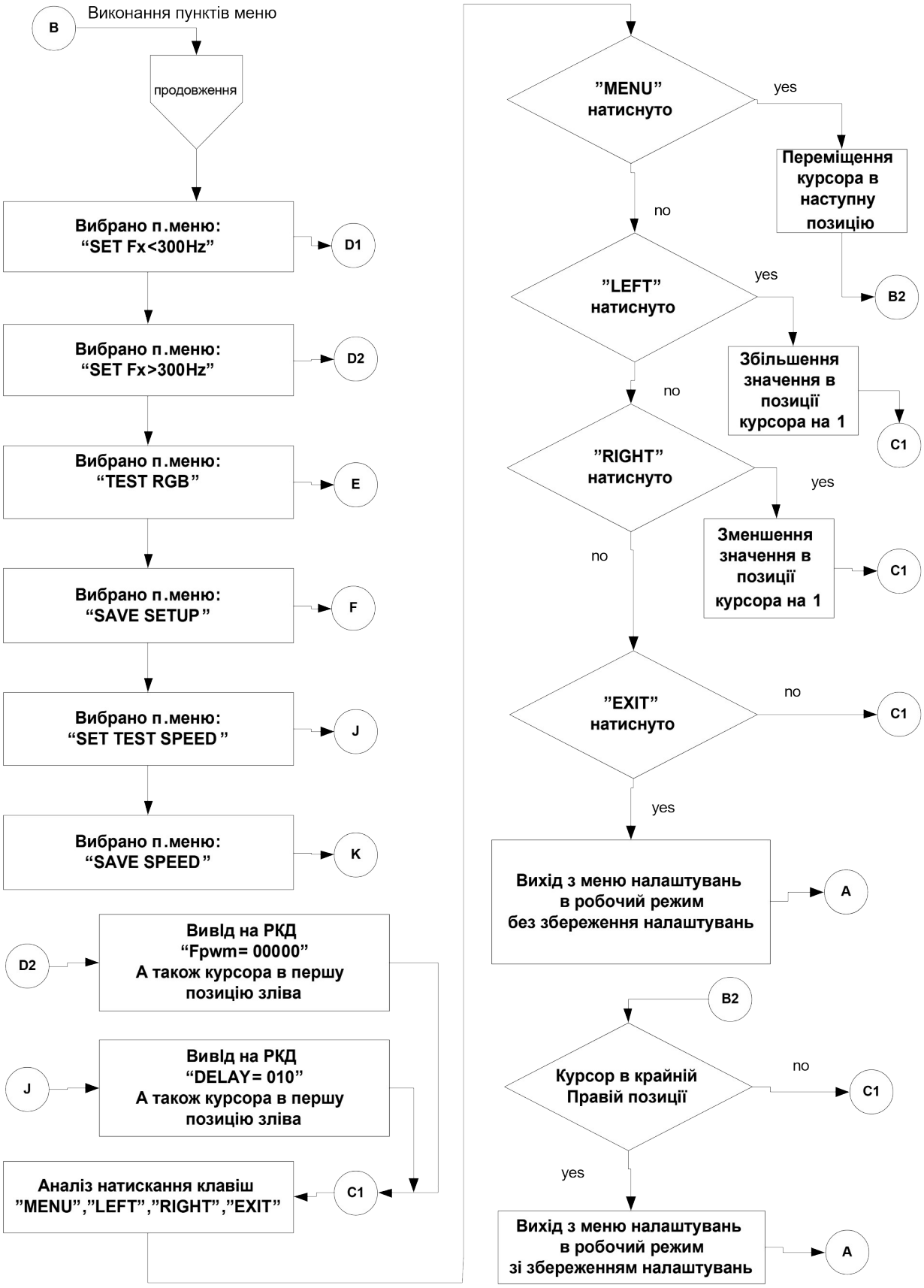


Рис. 4.3. Алгоритм роботи пристрою (продовження)

Алгоритм роботи універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок (продовження).

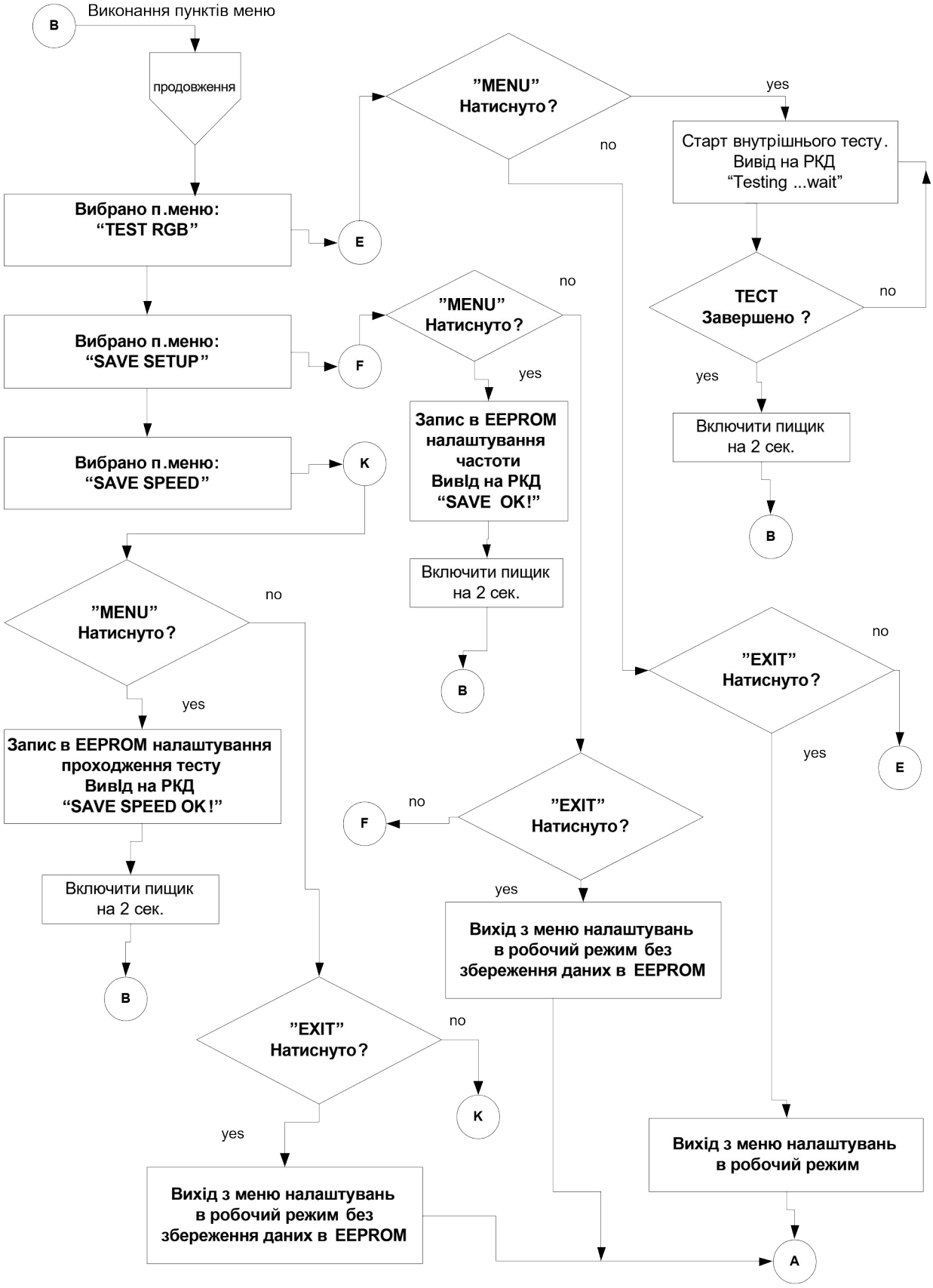


Рис. 4.4. Алгоритм роботи пристрою (кінець)

4.2. Розробка програмних модулів для роботи з повноколірною світлодіодною стрічкою

Для роботи з повноколірною світлодіодною стрічкою розроблено програмні модулі в CodeVision AVR, які містять наступні функції:

unsigned int read_adc(unsigned char adc_input) – читання інформації з порта АЦП;

void zvyk(void) – управління пищиком;

unsigned int set_page(void) – функція меню, яка встановлює частоту до 300 Гц;

unsigned int set_page2(void) – функція меню, яка встановлює частоту більше 300 Гц;

unsigned int set_speed(void) – функція меню, яка встановлює швидкість проходження тесту (встановлення затримки від 1 до 999 мс);

void Red(unsigned int dutyr) – обчислення PWM для Red каналу;

void Green(unsigned int dutyg) – обчислення PWM для Green каналу;

void Blue(unsigned int dutyb) – обчислення PWM для Blue каналу;

void test(unsigned int speedx) – тестовий модуль;

void fxcalculator(void) – функція для обчислення значень регістрів встановлення частоти;

void savespeed(void) – функція меню для збереження швидкості проходження тесту в EPROM;

void savesetup(void) – функція меню для збереження налаштувань в EPROM;

void main_menu(void) – головне меню і його підменю;

unsigned char get_key_status(unsigned char key) – функція біжучий статус клавіші;

unsigned char get_prev_key_status(unsigned char key) – функція попередній статус клавіші.

Робота внутрішнього тесту, функція test, полягає в послідовній зміні шпаруватості імпульсів кожного з трьох каналів з кроком затримки вказаним в змінній speedx по програмі наведеній у додатку 1.

4.3. Розробка програмного модуля для роботи з рідкокристалічним дисплеєм

Вивід інформації на РКД (16*2) виконується функціями з бібліотеки mylcd.c. Бібілотека має функції:

void lcd_write(unsigned char c) - запис команди управління в РКД;

void lcd_putchar(unsigned char c) - вивід символу на РКД;

void lcd_clear(void) - очистка РКД;

void lcd_putsf(unsigned char __flash *str) - вивід на LCD info з FLASH-пам’яті;

void lcd_puts(unsigned char *str) - вивід на LCD тексту;

void lcd_gotoxy(unsigned char x, unsigned char y) - позиціонування курсору;

void lcd_init(void) - функція ініціалізації РКД;

void cursor_on(void) - увімкнути курсор;

void cursor_off(void) - вимкнути курсор.

Реалізацію бібліотеки для РКД 16x2 (Модель: WH1602B-YGK-CTK) в CodeVisionAVR наведено у додатку 2.

4.4. Програмна реалізація головного алгоритму роботи універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок

Програма реалізує алгоритм роботи універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок згідно рис. 4.1 та приведена у додатку.

4.5. Моделювання та аналіз результатів роботи універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок в Proteus ISIS

Скомпільована CodeVisionAVR (IDE) програма для МК ATmega128 є файлом типу hex, який з допомогою програматора записується в мікроконтролер. Розроблену схему цифрового пристрою та роботу прошивки

перевіряємо в симуляторі Proteus ISIS. Нижче на рисунках зображено моделювання роботи спроектованого тестового пристрою.

PROTEUS VSM – це набір програм для автоматизованого проектування (САПР) електричних схем. PROTEUS VSM дозволяє моделювати роботи програмованих пристроїв: мікроконтролерів, мікропроцесорів, та інші. Пакет Proteus складається з двох частин: ISIS – програма синтезу та моделювання електронних схем і ARES – програма проєктування документації для виготовлення друкованих плат.

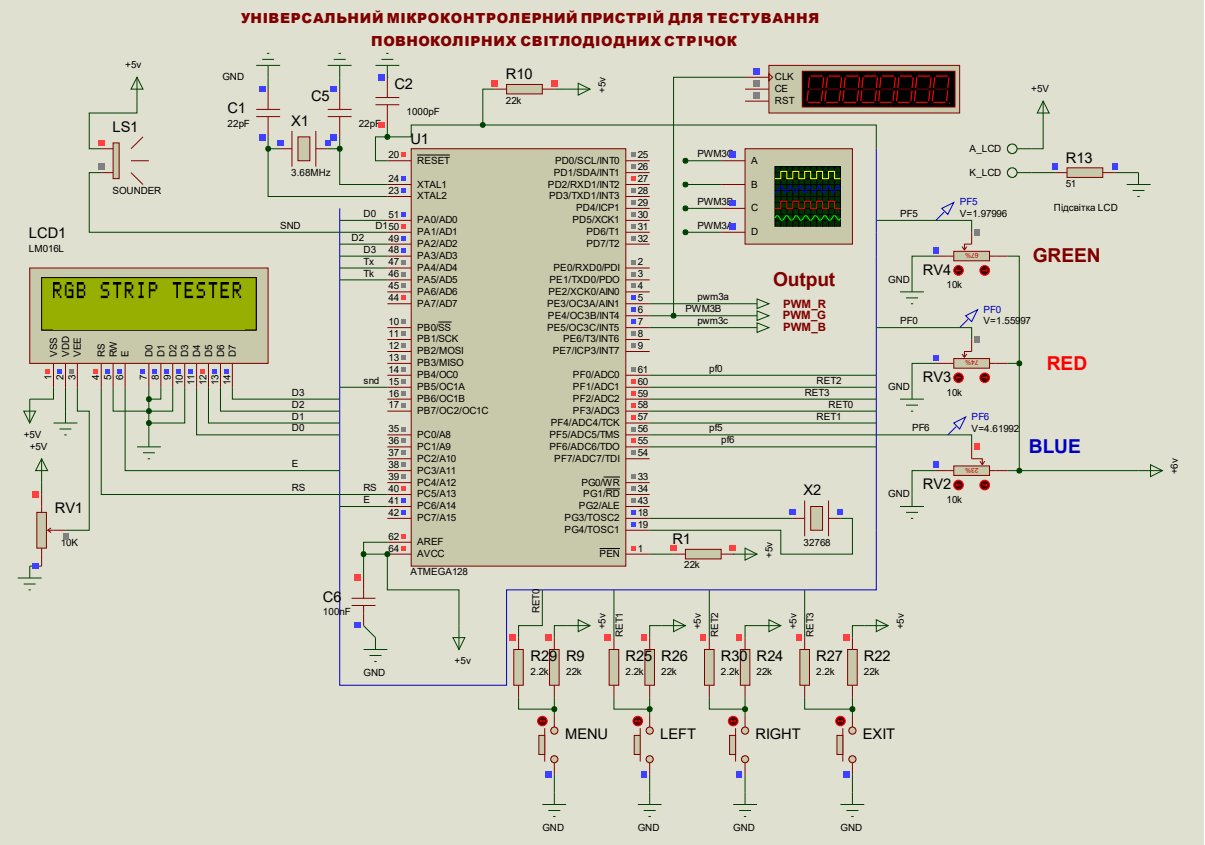


Рис. 4.5. Моделювання роботи пристрою для тестування повноколірних світлодіодних стрічок в Proteus ISIS – ініціалізація системи

При включенні пристрою МК ініціалізує РКД, 16-розрядні таймера1 і 3 для роботи в режимі ШІМ, порти АЦП, порти вводу-виводу, виводить на РКД повідомлення “RGB STRIP TESTER” і переходить в робочий режим.

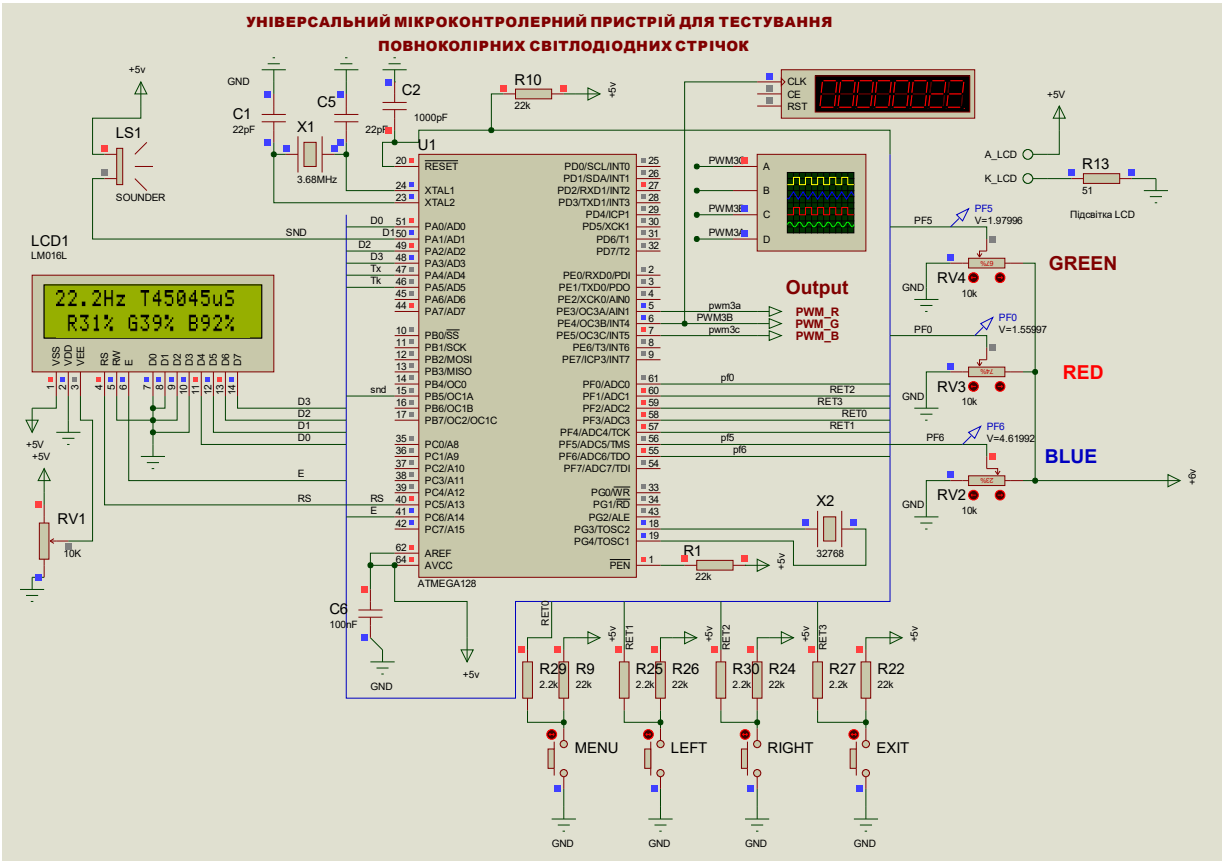


Рис. 4.6. Моделювання роботи пристрою для тестування повноколірних світлодіодних стрічок в Proteus ISIS – робочий режим.

В робочому режимі рис. 4.6 на дисплеї у верхньому рядку відображається значення робочої частоти ШІМ в Герцах, період (Т) прямокутних імпульсів в мікросекундах. У нижньому рядку виводиться інформація про шпаруватість імпульсів по кожному з трьох каналів ШІМ (Red, Green, Blue) у відсотках від періоду Т. Прямокутні імпульси з вказаними параметрами виводяться в порт PE постійно (сигнали PWM3A, PWM3B, PWM3C), а далі на силовий модуль управління повноколірною стрічкою (рис. 4.7).

В робочому режимі відбувається періодичне опитування чотирьох клавіш управління меню (MENU, LEFT, RIGHT, EXIT), а також аналіз напруг на потенціометрах управління кольорами RV2, RV3, RV4 та переналаштування каналів ШІМ у відповідності до цих напруг.

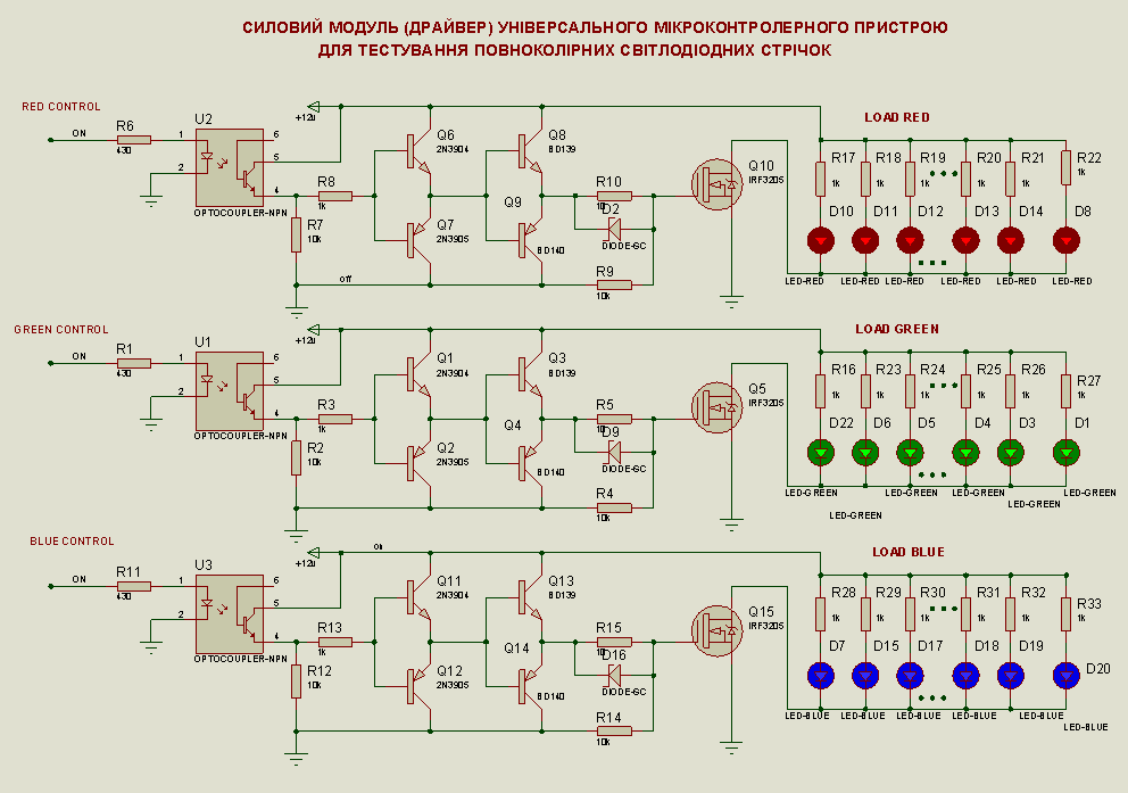


Рис. 4.7. Моделювання роботи силового модуля для тестування повноколірних світлодіодних стрічок в Proteus ISIS – робочий режим

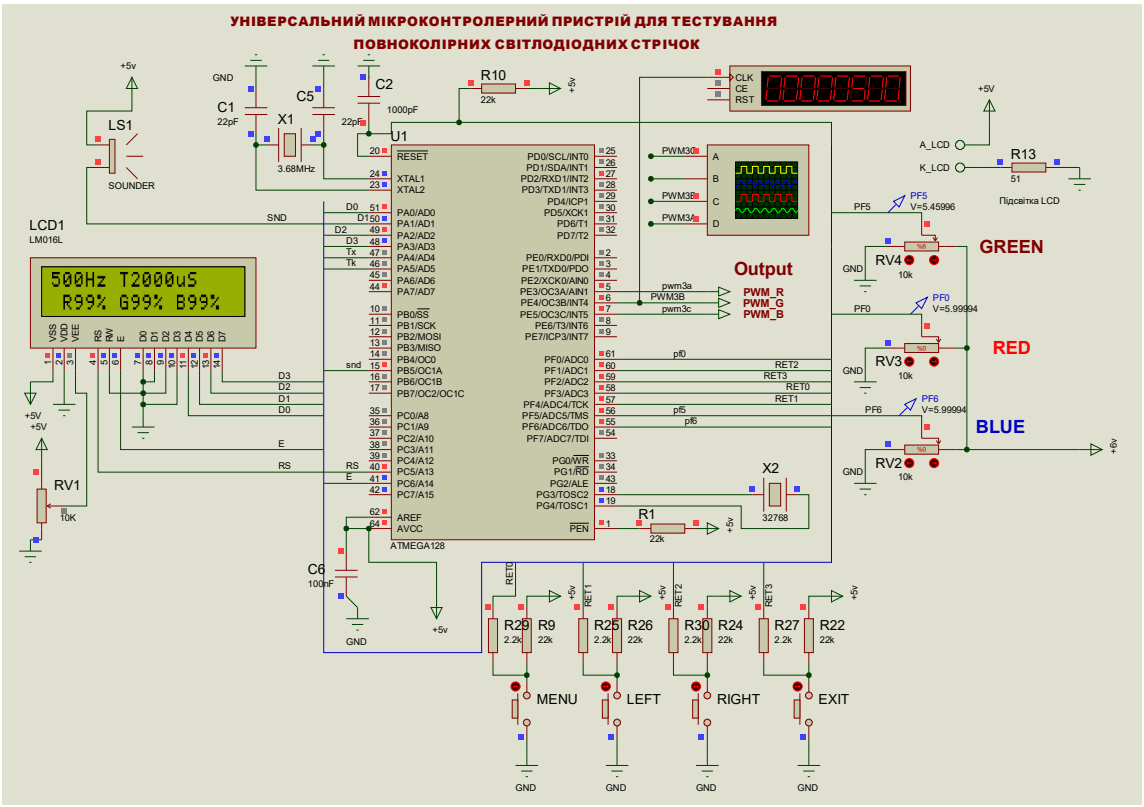


Рис. 4.8. Моделювання роботи пристрою в Proteus ISIS – робочий режим налаштовано стрічку на білий колір R99%, G99%, B99%

На рис. 4.9 змодельовано потенціометрами 99% – шпаруватість імпульсів по кожному каналу ШІМ (білий колір). Віртуальний частотомір показує частоту цих імпульсів. На рис. 4.10 підключений віртуальний осцилограф відображає тривалість імпульсів при цих налаштуваннях.

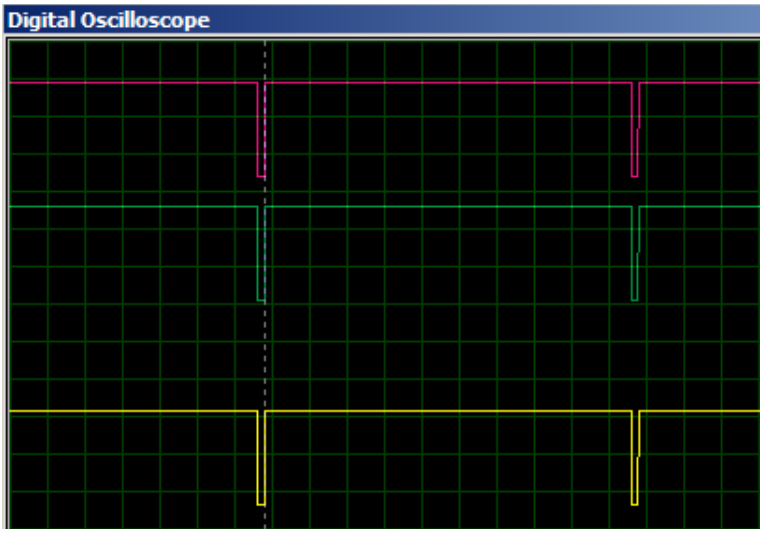


Рис. 4.9. ШІМ по трьох каналах з 99% шпаруватісю

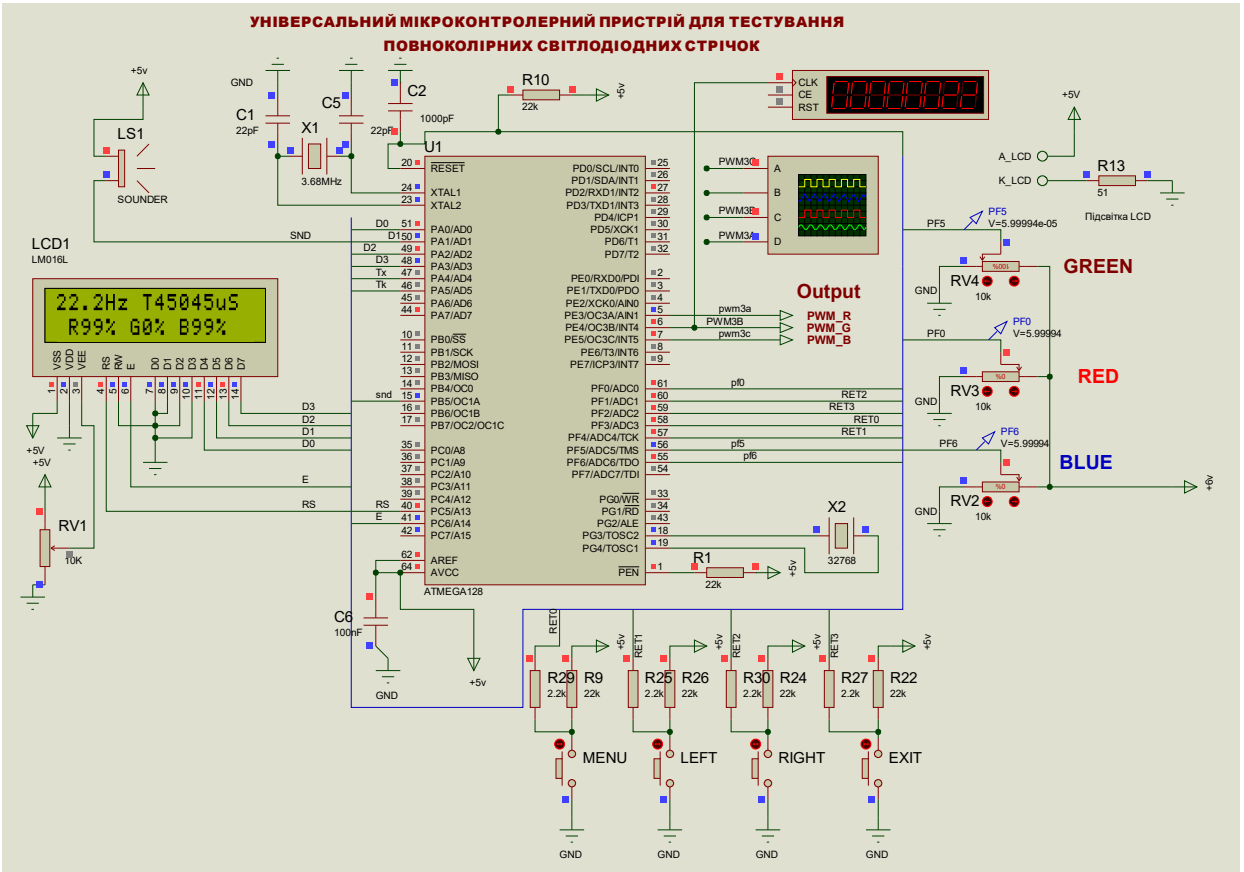


Рис. 4.10. Моделювання роботи пристрою в Proteus ISIS – робочий режим налаштовано стрічку на R99%, G0%, B99%

Результат налаштування R99%, G0%, B99% (вимкнено канал PWMG) відображено в моделюванні силового модуля на рис. 4.11.

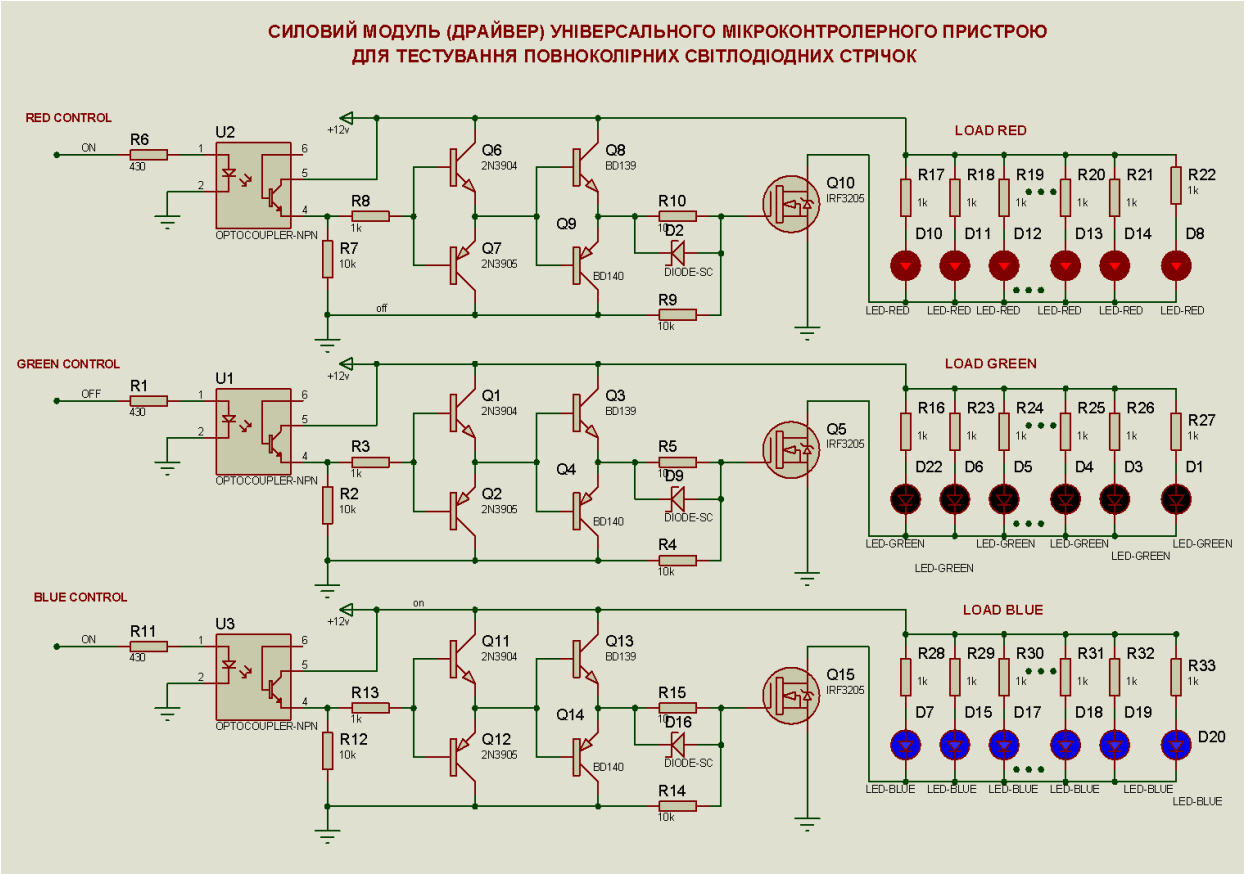


Рис. 4.11. Моделювання роботи силового модуля в Proteus ISIS – робочий режим, налаштовано стрічку на R99%, G0%, B99%

При натисканні клавіші “MENU” МК переходить на виконання підпрограми роботи з меню налаштувань режиму роботи пристрою.

Пункти меню наступні:

– Встановити частоту до 300 Гц.

В цьому пункті меню можна встановити частоту прямокутних імпульсів від 1Гц до 300 Гц з точністю 0,1 Гц.

– Встановити частоту більшу 300 Гц.

В цьому пункті меню можна встановити частоту прямокутних імпульсів від 300 Гц до 30000 Гц з точністю 1 Гц.

– Налаштування швидкості проходження вбудованого тесту повно колірної стрічки.

– Виконати тестування повноколірної стрічки.

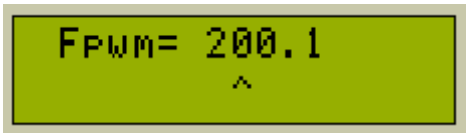
Відбувається динамічне тестування повноколірної стрічки вбудованим тестом з швидкістю, що залежить від налаштувань в пункті “SET TEST SPEED”.

– Збереження швидкості проходження тесту в довготривалій пам’яті EPROM.

– Запис в EPROM значення частоти імпульсів.

Для входження в основне меню достатньо натиснути клавішу “MENU”. Для вибору пунктів меню задіяні клавіші “LEFT”, “RIGHT”. Для виконання пункту меню натиснути клавішу “MENU”, для виходу з меню натиснути клавішу “EXIT”. Розглянемо налаштування вихідних прямокутних імпульсів на частоту 200,1 Гц. Вибрати пункт меню:

– натиснувши клавішу “MENU” отримаємо:



– підменю в якому знизу відображається курсор, а зверху біжуча частота ШІМ. Рух курсора зліва-направо здійснюється натисканням клавіші “MENU”. В позиції на яку вказує курсор вибрати потрібну цифру з допомогою клавіш “LEFT” і “RIGHT” (відповідно інкремент,

декремент числа від 0..до 9), перемістити курсор в наступну позицію і вибрати наступну цифру. Після вибору останньої цифри відбувається запис сформованого числа в робочі регістри і переналаштування всіх ШІМ каналів на нову частоту. Для виходу з меню без збереження нової частоти натиснути клавішу “EXIT”. Аналогічно встановлюється частота більша 300 Гц., але в підменю останній розряд числа – це одиниці Герц.

При виборі пункту меню: 

і його запуску на виконання отримаємо повідомлення: 

- тестування…чекайте!. Закінчення тестування супроводиться коротким звуковим сигналом пищика LS. При проходженні тесту на віртуальному осцилографі можна спостерігати динамічну зміну шпаруватості імпульсів по трьох ШІМ каналах (рис. 4.12).

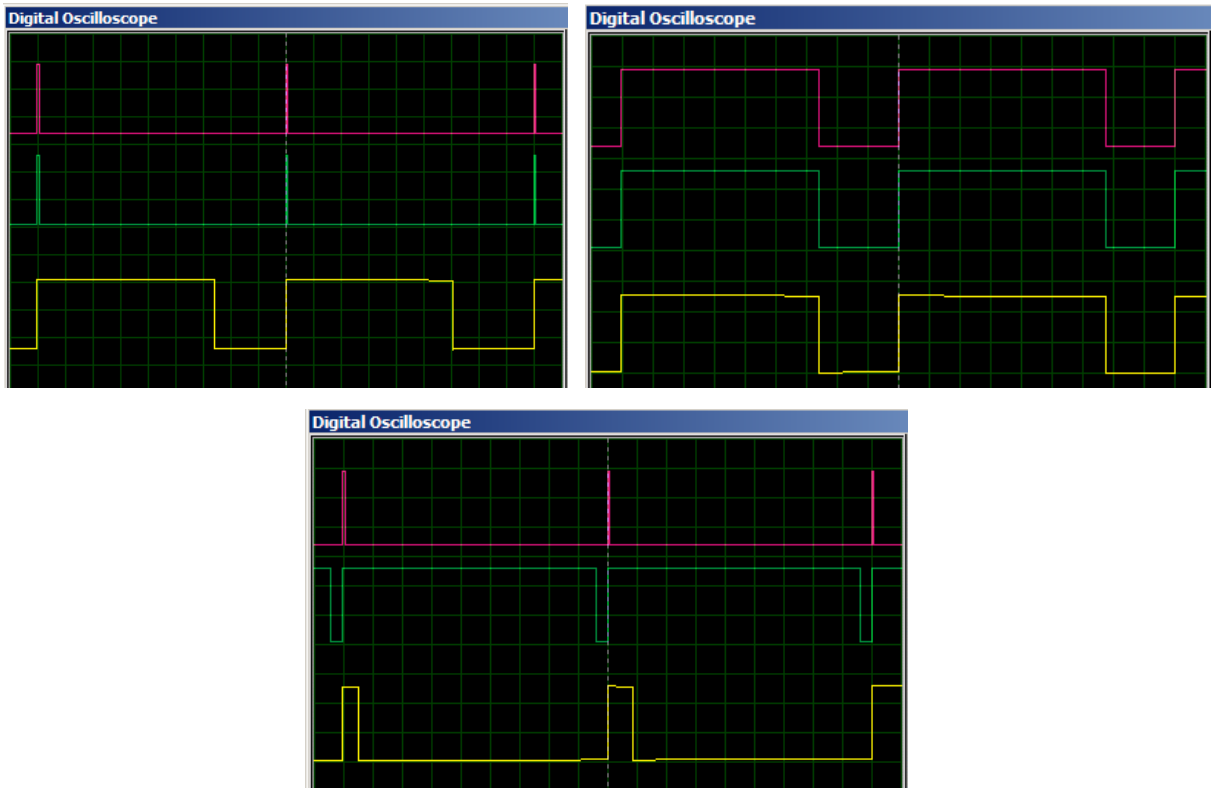
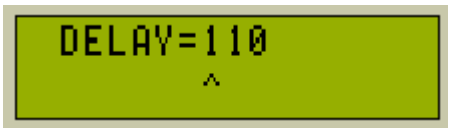


Рис. 4.12. Моделювання роботи пристрою в Proteus ISIS – тестовий режим Динамічна зміна палітри кольорів на віртуальному осцилографі.

Налаштування швидкості проходження тесту: 

Має підменю: 

в якому задаємо затримку в мілісекундах при виконанні кроків тестування. Збереження налаштувань затримки здійснюється через пункт меню: 

і супроводиться повідомлення 

– збережено, та коротким звуковим сигналом пищика LS. Збереження налаштувань частоти в довготривалій пам’яті здійснюється через пункт меню: 

і супроводиться повідомленням 

– збережено, та коротким звуковим сигналом пищика LS.

ВИСНОВКИ

В роботі розроблено апаратне та програмне забезпечення універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок.

Універсальний мікроконтролерний пристрій для тестування повноколірних світлодіодних стрічок дає можливість досліджувати динамічні та світлові характеристики світлодіодних повноколірних і монохромних (одноколірних) стрічок, має гнучке меню налаштувань з можливістю зберігання параметрів тестування в довготривалій памяті, має в памяті прості тести для перевірки справності RGB стрічок, та потенціометри ручного налаштування параметрів тестування. Параметри тестування виводяться на рідкокристалічний дисплей. Пристрій має ширші можливості дешевший в порівнянні з існуючими контролерами, а саме рідкокристалічний екран, меню управління, можливість встановлення частоти ШІМ, відображення інтенсивності кольорів на РКД і має тестове призначення.

При розробці пристрою для тестування повноколірних світлодіодних стрічок використано мікроконтролер AVR, тестову повноколірну світлодіодну стрічку з SMD5050, і РКД 16x2 (Модель: WH1602B-YGK-CTK).

Спроєктовано електричну принципову схему, а також створено модель системи в Proteus VSM. Розроблено її алгоритм роботи і програмне забезпечення для МК AVR на мові С за допомогою інструментарію розробки ПЗ для МК AVR CodeVisionAVR. Створено програмні модулі для тестування повноколірних світлодіодних стрічок, програмний модуль виводу інформації на РКД, гнучке та зручне меню налаштувань та управління, силовий модуль.

Проведено моделювання роботи пристрою в емуляторі Proteus ISIS. Результати моделювання показали коректність програмної реалізації головного алгоритму роботи пристрою та розроблених програмних модулів.

Під час виконання роботи отримав знання і навики з розробки і програмування пристроїв на AVR мікроконтролерах та під’єднання до них електричних модулів.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Bert van Dam. Microcontroller Systems Engineering: 45 projects for PIC, AVR and ARM. – Elektor Publishing, 2009. – p. 330.
2. John Boxall. AVR Workshop: A Hands-On Introduction with 60 Projects. – Published by No Starch Press, 2022. – p. 368.
3. Jonathan Bartlett. Electronics for Beginners: A Practical Introduction to Schematics, Circuits, and Microcontrollers. – Published by Apress, 1st ed., 2020. – p. 537.
4. Alan Trevennor. Practical AVR Microcontrollers: Games, Gadgets, and Home Automation with the Microcontroller Used in the Arduino (Technology in Action). – Published by Apress; 1st ed., 2012. – p. 443.
5. Alan Travennor. Experimenting with AVR Microcontrollers (Technology in Action). – Published by Apress, 1st ed., 2014. – p. 206.
6. Огляд світлодіодних стрічок [Електронні ресурси]. – Режим доступу: https://www.waveformlighting.com/led-strip-lights, https://www.super-brightleds.com/blog/ultimate-led-strip-lighting-guide.html, https://www.signliteled.com/complete-guide-to-addressable-led-strip-ic-types/.
7. RGB LED Controller з Wireless IR Remote [Електронний ресурс]. – Режим доступу: https://www.superbrightleds.com/rgb-led-controller-with-wireless-ir-remote-dynamic-color-changing-modes-3-amps-channel-rgb-controller-with-remote.
8. RGB LED Controller – Wireless RF Remote [Електронний ресурс]. – Режим доступу: https://www.superbrightleds.com/rgb-led-controller-with-lc4-connector-wireless-rf-remote-with-dynamic-color-changing-modes-5-amps-channel-mini-rgb-led-controller-lc4.
9. RGB LED Controller – Wireless RF Touch Color Remote [Електронний ресурс]. – Режим доступу: https://www.superbrightleds.com/rgb-led-controller-wireless-rf-touch-color-remote-with-dynamic-color-changing-modes-8-amps-channel.

10. Підходи до розробки систем управління світлодіодними стрічками [Електронний ресурс]. – Режим доступу: https://randomnerd-tutorials.com/guide-for-ws2812b-addressable-rgb-led-strip-with-arduino/, https://www.instructables.com/PWM-an-RGB-LED-Strip-with-Arduino/, https://racheldebarros.com/arduino-projects/how-to-connect-and-power-an-led-strip-with-arduino/, https://arduinogetstarted.com/tutorials/arduino-led-strip#google_vignette, https://howtomechatronics.com/tutorials/arduino/-how-to-control-ws2812b-individually-addressable-leds-using-arduino/#-google_vignette.
11. САПР Proteus VSM [Електронний ресурс]. – Режим доступу: https://www.labcenter.com/simulation/.
12. Інструментарій розробки ПЗ для МК AVR CodeVisionAVR [Електронний ресурс]. – Режим доступу: https://www.hpinfotech.ro/cvavr-download.html.
13. AVR033: Getting Started with the CodeVisionAVR C Compiler [Електронний ресурс]. – Режим доступу: https://www.professordan.com/-avr/techlib/techlib8/appnotes/pdf_avr/AVR033.pdf.
14. HP InfoTech. CodeVisionAVR User Manual. https://www.thierry-lequeu.fr/data/CodeVisionAVR-3-20-User-Manual.pdf.
15. Steven F. Barrett, Daniel Pack, Mitchell Thornton. Atmel AVR Microcontroller Primer: Programming and Interfacing (Synthesis Lectures on Digital Circuits and Systems). – Morgan & Claypool Publishers; 1st edition, 2007 – p. 180.
16. Steven F. Barrett and Daniel J. Pack. Atmel AVR Microcontroller Primer: Programming and Interfacing. – Morgan & Claypool Publishers; 2nd edition, 2007 – p. 194.
17. Elliot Williams. Make. AVR Programming: Learning to Write Software for Hardware. – Publisher: Make Community LLC, 2014. – p. 472.
18. Dhananjay V. Gadre. Programming and Customizing the AVR Microcontroller. – Published by McGraw-Hill Education TAB, 2000. - p. 336.

19. Muhammad Ali Mazidi, Sarmad Naimi, Sepher Naimi. The AVR Microcontroller and Embedded Systems: Using Assembly and C – Published by Pearson, 1st ed., 2015. – p. 752.
20. Richard H. Barnett, Sarah A. Cox, and Larry O'Cull. Embedded C Programming and the Atmel AVR. – Published by Cengage Learning, 2nd ed., 2006. – p. 560.
21. Panayotis Papazoglou. An Educational Guide to the AVR Microcontroller Programming. – copyright by Panayotis Papazoglou, 2018. – p. 16.
22. Elliot Williams. Make: AVR Programming. – copyright by Elliot Williams, 2014. – p. 41.
23. Навчальні ресурси по МК AVR [Електронний ресурс] – Режим доступу: https://embedds.com/avr-tutorials/, https://www.electronicwings.com/avr-atmega/getting-started-with-atmel-studio.
24. Мікроконтролер ATmega128 [Електронний ресурс] – Режим доступу: https://ww1.microchip.com/downloads/en/DeviceDoc/doc2467.pdf, https://www.theengineeringprojects.com/2018/09/introduction-to-atmega128.html.
25. Алфавітно-цифровий РК-дисплей 16x2 HD44780 [Електронний ресурс]. – Режим доступу: https://hackprojects.wordpress.com/forum/avr/display-custom-character-on-16x2-lcd-to-avr-microcontroller-atmega-8/, https://davidegironi.blogspot.com/2013/06/an-avr-atmega-library-for-hd44780-based.html, https://www.allaboutcircuits.com/technical-articles/-how-to-a-162-lcd-module-with-an-mcu/.
26. Опис LED-стрічки з світлодіодами SMD5050 https://www.iled.com/class/INNOVAEditor/assets/YeniDatasheets/4050-4055-4057.pdf, https://sirs-e.com/general/5050-led-datasheet/.
27. AN2519: AVR Microcontroller Hardware Design Considerations – Microchip Technology, 2017 – p. 27.
28.

ДОДАТКИ

Додаток 1

Лістинг тестового модуля для роботи з повноколірною світлодіодною стрічкою

```
void test (unsigned  int  speedx)   //  модуль тестування
   {
    unsigned char R=0, G=0, B=0;
    for (R=0; R<100; R++) { // red від 0 до 100 %
       Red(R); delay_ms(speedx);
    }
    for (R=99; R>0; R--) { // red від 100% до 1%
       Red(R); delay_ms(speedx);
    }
    for (G=0; G<100; G++) { // Green від 0 до 100 %
       Green(G); delay_ms(speedx);
    }
    for (G=99; G>0; G--) { // Green від 100% до 1%
       Green(G); delay_ms(speedx);
    }
    for (B=0; B<100; B++) { // Blue від 0 до 100 %
       Blue (B); delay_ms(speedx);
    }
    for (B=99; B>0; B--) { // Blue від 100% до 1%
       Blue(B); delay_ms(speedx);
    }
    for (B=0; B<100; B++) { // Red Green Blue від 0 до 100 %
       Red(B); Green(B); Blue (B); delay_ms(speedx);
    }
    for (B=99; B>0; B--) { // Red =100% Green -Blue від 100% до 1%
       Green(B); Blue(B); delay_ms(speedx);
    }
    for (B=0; B<100; B++) { // Blue від 0 до 100 %
       Blue(B); delay_ms(speedx);
    }
    for (G=0; G<100; G++) { // Green від 0 до 100 %
       Green(G); delay_ms(speedx);
    }
    for (R=99; R>0; R--) { // red від 100% до 1%
       Red(R); delay_ms(speedx);
    };
```

Додаток 2

Лістинг програмного коду модуля для роботи з рідкокристалічним дисплеєм 16x2

```
#pragma used+
//прототипи функцій
#define RS PORTC.5     //  LCD
#define E  PORTC.6     //  LCD
#define D4 PORTA.0     //  LCD  data
#define D5 PORTA.1     //  LCD  data
#define D6 PORTA.2     //  LCD  data
#define D7 PORTA.3     //  LCD  data
  //реалізація функцій
void lcd_strobe (void) //Функція строб на на виводі E
{
   E=1; delay_ms(2); E=0;
}

//Функція посилки коду на виводи D4-D7,
//спочатку передається старша тетрада, потім - молодша
void lcd_write (unsigned char c){
   if(c&0x10) D4=1; else D4=0; if(c&0x20) D5=1; else D5=0;
   if(c&0x40) D6=1; else D6=0; if(c&0x80) D7=1; else D7=0;
   lcd_strobe(); delay_ms(5);
   if(c&0x1) D4=1; else D4=0; if(c&0x2) D5=1; else D5=0;
   if(c&0x4) D6=1; else D6=0; if(c&0x8) D7=1; else D7=0;
   lcd_strobe(); delay_ms(5);
}

void lcd_putchar (unsigned char c){
   RS=1; lcd_write(c);
}

void lcd_cmd (unsigned char c){
   RS=0; lcd_write(c);
}

void lcd_clear (void){
   RS=0; lcd_write(0x01); delay_ms(5);
}

//Вивід на LCD info з FLASH-памяті
void lcd_putsf(unsigned char __flash *str)
{
   unsigned char Count=0;
   while (str[Count]!=0x00) {
      lcd_putchar(str[Count]); Count++;
   }
}

void lcd_puts(unsigned char *str)
{
   unsigned char Count=0;
   while (str[Count]!=0x00) {
      lcd_putchar(str[Count]); Count++;
   }
}

void lcd_gotoxy(unsigned char x, unsigned char y)
```

```
{
   unsigned char Temp;
   Temp=0x80;
   if(y==1) Temp=0xc0;
   lcd_cmd (Temp+x);
}

void lcd_init (void){
   //Ініціалізація диспелея в 4-х бітному режимі.
   delay_ms(25);//витримуємо паузу не менше 15мс
   RS=0;//Передача команди
   D7=0; D6=0; D5=1; D4=0; lcd_strobe(); delay_ms(5);
   D7=0; D6=0; D5=1; D4=0; lcd_strobe(); delay_ms(5);
   D7=0; D6=0; D5=1; D4=0; lcd_strobe(); delay_ms(5);
   lcd_cmd(0x28); delay_ms(5); lcd_cmd(0x01); delay_ms(5); lcd_cmd(0x0C);
}

void cursor_on (void) {
   RS=0; lcd_cmd(0x0E); // курсор!
 }

void cursor_off (void) {
   RS=0; lcd_cmd(0x0C);
}

 #pragma used-
```

Додаток 3

Лістинг коду головного програмного модуля універсального мікроконтролерного пристрою для тестування повноколірних світлодіодних стрічок

```
/*****************************************************
Project : УНІВЕРСАЛЬНИЙ МІКРОКОНТРОЛЕРНИЙ ПРИСТРІЙ ДЛЯ ТЕСТУВАННЯ ПОВНОКОЛІРНИХ
СВІТЛОДІОДНИХ СТРІЧОК
Comments: this program is designed on the base of AVR ATmega128 microcontroller
Clock frequency     : Clock frequency     : 3,680000 MHz
*****************************************************/
#include <mega128.h>
#include <stdio.h>
#include <delay.h>
#include "mylcd.c"     // бібліотека для LCD
#define SYS_FREQ        3680000UL    // тактова частота 3.68 MHz
//налаштування АЦП
#define ADC_VREF_TYPE  0x00   //  Bit 5 - ADLAR=0 - 10 біт  використовуємо ADCH і
ADCL
#define REFS0   6             // selector Vref  - вибір джерела опорної напруги
#define REFS1   7             // selector Vref
//клавіші  MENU/ENTER, SELECT+, SELECT-, ESC/EXIT
#define BTNS_PORT              PORTF    //  for buttons on  PORTF
#define BTNS_PORT_DDR          DDRF     //  for  buttons on PORTF
#define MENU_ENTER_BTN       3          //  kn-menu
#define SELECT_PLUS_BTN      4          //  kn- MAX
#define SELECT_MINUS_BTN     1          //  kn - min
#define ESC_BTN              2          //  kn -esc
#define CNTREPEAT            300        //  утримання клавіш
// Declare your global variables here
unsigned char k, PREV_PINF = 0xff;                          // menu 1111 1111
unsigned int gSelectPlusCounter=0, gSelectMinusCounter=0;  // menu селектор
volatile  unsigned int  hour = 0;    // частота встановлена ціле
volatile  unsigned int start_p=0;    // встановлена частота
eeprom float rom_fxx;      //  частота з ROM
char  buffer[33];               // для LCD перетворення
char N=1; // prescaler
volatile unsigned long int  fx=0, tx=0;  // частота і період
volatile float  fxx=22.2;  // опорна частота- можна встановити від 1... 29999 Гц
volatile unsigned char  redpwml=0,redpwmh=0, greenpwml=0,greenpwmh=0,
bluepwml=0,bluepwmh=0;  // pwm , pwm_val
volatile unsigned int ICR3top=0;
volatile unsigned int speed=10; // затримка в ms -більше значення-повільніше
тестування
unsigned int ICR3, RedPwm, GreenPwm, BluePwm ;
unsigned int Dred, Dgreen, Dblue;  // duty = red green blue
unsigned int adc_in;          // змінна для ADC - код
float indication=0, tinf=0;
char ADC_INPUT = 0x05;       // порт з якого читаємо Analog signal = PF5(ADC5)
char vref=5;                 // опорна напруга
char od='m'  ;
volatile unsigned char value[5] = {0, 0, 0, 0, 0};
volatile unsigned int A0,A1,A2,A3,A4;
volatile unsigned int V0,V1,V2,V3,V4;
//  key - функція  біжучий статус клавіші
  unsigned char get_key_status(unsigned char key)    //  key - номер клавіші  на
порту
{
```

```
   return (!(PINF & (1<<key)));
}

// функція попередній статус клавіші
unsigned char get_prev_key_status(unsigned char key)  // key - номер клавіші  на
порту
{
  return (!(PREV_PINF & (1<<key)));
}

// Timer 2 overflow interrupt service routine  використовується для меню!!!
interrupt [TIM2_OVF] void timer2_ovf_isr(void)
{
    if ( get_key_status(SELECT_PLUS_BTN) )  // аналіз натискання клавіші SELECT+
      {
         if (gSelectPlusCounter < CNTREPEAT)
            gSelectPlusCounter++;
       }
    else  gSelectPlusCounter = 0;

  if ( get_key_status(SELECT_MINUS_BTN) )  // аналіз натискання клавіші SELECT-
  {
    if (gSelectMinusCounter < CNTREPEAT)
        gSelectMinusCounter++;
  }
  else gSelectMinusCounter = 0;
}


// Timer 3 overflow interrupt service routine
interrupt [TIM3_OVF] void timer3_ovf_isr(void)
{
   OCR3AH=redpwmh;  OCR3AL=redpwml;      // RED - червоний
   OCR3BH=greenpwmh;  OCR3BL=greenpwml;  // GREEN - зелений
   OCR3CH=bluepwmh;  OCR3CL=bluepwml;    // BLUE - синій
 }

 // Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
   ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
   delay_us(10);      // Delay needed for the stabilization of the ADC input voltage
                      // Start the AD conversion
  ADCSRA|=0x40;
  // Wait for the AD conversion to complete
  while ((ADCSRA & 0x10)==0);
  ADCSRA|=0x10;
   return ADCW;
}

 void zvyk(void) // функція управління пищиком
 {
  DDRB.5=1; // управління звуком on/off
  delay_ms(200);
  DDRB.5=0; delay_ms(200);
 }

// функція меню -   set частоту до 300 гц
unsigned int set_page(void)
{
  unsigned char x[4] = { 7, 8, 9, 11 }, i = 0; // позиції символів число з 4 значень
```

```
  char time_chars[4][2] = {"^", "^", "^", "^"};  // нижній курсор-стрілка 4 позиції
  lcd_gotoxy(x[0],1);
  lcd_puts(time_chars[0]);
  if (hour>3000) {hour=2999;};        //  перше "^"  - курсор нижній
   A0=hour; V0=A0/1000; value[0]=(unsigned char)(A0/1000); // перший символ зліва
 A1=A0-V0*1000; V1=(A1/100); value[1]=(unsigned char)(A1/100); // другий символ зліва
   A2=A1-V1*100; V2=A2/10; value[2]=(unsigned char)(A2/10); // третій символ зліва
   A3=A2-V2*10; V3=A3; value[3]=(unsigned char)(A3); // четвертий символ зліва
  while (1) {
  PREV_PINF=PINF; sprintf(buffer, "%u%u%u.%u", value[0],value[1],value[2],value[3]);
 // чотири символи для виводу
    lcd_gotoxy(x[0],0); lcd_puts(buffer);
    if (get_key_status(SELECT_PLUS_BTN))  // select+  UP ---set_page()-
      if (!get_prev_key_status(SELECT_PLUS_BTN) || (gSelectPlusCounter == CNTREPEAT))
      {
        if (gSelectPlusCounter == CNTREPEAT)
          delay_ms(80);
         switch (i) {
         case 0: //  ліва_старша=нульова позиція числа page значення=( 0..2)
              if (value[0] < 2) value[0]++;
              else value[0] = 0;
         break;
         case 1: // друга позиція зліва числа page значення=(0..9)
               if (value[1] < 9) value[1]++;
               else value[1] = 0;
         break;
         case 2: // третя позиція зліва числа page значення=(0..9)
              if (value[i] < 9) value[i]++;
              else value[i] = 0;
         break;
         case 3: // четверта позиція зліва числа page значення=(0..9)
              if (value[i] < 9) value[i]++;
              else value[i] = 0;
         break;
         } }
 //-end select+  UP ---set_page()--
    if (get_key_status(SELECT_MINUS_BTN))  // select-  DOWN---set_page()--
      if (!get_prev_key_status(SELECT_MINUS_BTN) || (gSelectMinusCounter ==
CNTREPEAT)) {
        if (gSelectMinusCounter == CNTREPEAT)
          delay_ms(80);
        switch (i) {
        case 0:
              if (value[0] > 0) value[0]--;
              else value[0] = 2;
        break;
        case 1:
              if (value[1] > 0) value[1]--;
              else value[1] = 9;
        break;
        case 2:
              if (value[i] > 0) value[i]--;
              else value[i] = 9;
        break;
        case 3:
              if (value[i] > 0) value[i]--;
              else value[i] = 9;
        break;
        } }
//-end select-  DOWN---set_page()-
    if (get_key_status(MENU_ENTER_BTN))  // enter---set_page()-
```

```
      if (!get_prev_key_status(MENU_ENTER_BTN)) {
        if (i==3) { // кількість символів числа=4
         hour=(unsigned int)(value[0])*1000; //+(unsigned int)(value[1])*100);
       hour=hour+(unsigned int)(value[1])*100; hour=hour+(unsigned int)(value[2])*10;
         hour = hour+(unsigned int)(value[3]);
           if (hour> 2999) hour=2999; lcd_gotoxy(x[i],1); // затерти курсор
             lcd_putchar(' '); return(hour); // сформоване значення
        }
        lcd_gotoxy(x[i],1); lcd_putchar(' ');
        lcd_gotoxy(x[++i],1); lcd_puts(time_chars[i]);
      }
  //-end enter---set_page()-
    if (get_key_status(ESC_BTN)) // exit --set_page()-
    {
      PREV_PINF = PINF; // exitflag=1;    // натиснута клавіша exit
      return;
    } } }
//--end exit ---set_page()-
// функція меню -   set частоту більше 300 гц
unsigned int set_page2(void)
{  // символи числа сторінки
  unsigned char   x[5] = { 7, 8, 9, 10, 11 }, i = 0;  // позиції символів з 5 значень
  char time_chars[5][2] = {"^", "^", "^", "^","^"}; //нижній курсор - стрілка 4
позиції
  lcd_gotoxy(x[0],1); lcd_puts(time_chars[0]); //  перше "^" - курсор нижній
A0=hour; V0=A0/10000; value[0]=(unsigned char)(A0/10000);//старший(1-ий) символ зліва
A1=A0-V0*10000; V1=(A1/1000); value[1]=(unsigned char)(A1/1000);//другий символ зліва
   A2=A1-V1*1000; V2=A2/100; value[2]=(unsigned char)(A2/100); // третій символ зліва
   A3=A2-V2*100; V3=A3/10; value[3]=(unsigned char)(A3/10); // четвертий символ зліва
   A4=A3-V3*10; V4=A4; value[4]=(unsigned char)A4; // молодший -п'ятий символ зліва
  while (1) {
    PREV_PINF = PINF;
    sprintf(buffer, "%u%u%u%u%u", value[0], value[1], value[2], value[3], value[4]);
// чотири символи для виводу
    lcd_gotoxy(x[0],0); lcd_puts(buffer);
    if (get_key_status(SELECT_PLUS_BTN)) //   select+  UP ---set_page()-
      if (!get_prev_key_status(SELECT_PLUS_BTN) || (gSelectPlusCounter == CNTREPEAT))
      {
        if (gSelectPlusCounter == CNTREPEAT) delay_ms(80);
         switch (i) {
          case 0: // ліва_старша= перша позиція числа page значення=( 0..2)
              if (value[0] < 2) value[0]++;
              else value[0] = 0;
          break;
          case 1: // друга позиція зліва числа page значення=(0..9)
               if (value[1] < 9) value[1]++;
               else value[1] = 0;
          break;
          case 2:  // третя позиція зліва числа page значення=(0..9)
              if (value[2] < 9) value[2]++;
              else value[2] = 0;
          break;
          case 3:  // четверта позиція зліва числа page значення=(0..9)
              if (value[i] < 9) value[i]++;
              else value[i] = 0;
          break;
          case 4:  // п'ята позиція зліва числа page значення=(0..9)
              if (value[i] < 9) value[i]++;
              else value[i] = 0;
          break;
         }
```

```
      }
     //-end select+  UP ---set_page()--
    if (get_key_status(SELECT_MINUS_BTN))  // select-  DOWN---set_page()--
      if (!get_prev_key_status(SELECT_MINUS_BTN) || (gSelectMinusCounter ==
CNTREPEAT)) {
        if (gSelectMinusCounter == CNTREPEAT) delay_ms(80);
        switch (i) {
         case 0:
              if (value[0] > 0) value[0]--;
              else value[0] = 2;
         break;
         case 1:
              if (value[1] > 0) value[1]--;
              else value[1] = 9;
         break;
         case 2:
              if (value[2] > 0) value[2]--;
              else value[2] = 9;
         break;
         case 3:
              if (value[3] > 0) value[3]--;
              else value[3] = 9;
         break;
         case 4:
              if (value[4] > 0) value[4]--;
              else value[4] = 9;
         break;
        } }
     //-end select-  DOWN---set_page()-
    if (get_key_status(MENU_ENTER_BTN)) // enter---set_page()-
      if (!get_prev_key_status(MENU_ENTER_BTN)) {
        if (i==4) { // кількість символів числа =5
          hour = (unsigned int)(value[0]*10000+value[1]*1000);
          hour = hour+(unsigned int)(value[2])*100;
          hour = hour+(unsigned int)(value[3])*10+(unsigned int)(value[4]);
          if (hour>29999) hour=29999; lcd_gotoxy(x[i],1); // затерти курсор
          lcd_putchar(' '); return(hour); // сформоване значення
        }
 lcd_gotoxy(x[i],1); lcd_putchar(' '); lcd_gotoxy(x[++i],1); lcd_puts(time_chars[i]);
      }
      //-end enter---set_page()-
    if (get_key_status(ESC_BTN)) // exit --set_page()-
    {
      PREV_PINF = PINF; // натиснута клавіша exit
      return;
    } } }
//--end exit ---set_page()-

// функція меню -   set speed до 999 ms
unsigned int set_speed(void)
{
  unsigned char  x[3] = { 7, 8, 9 }, i = 0;    // позиції символів число з 3 значень
  char time_chars[3][2] = {"^", "^", "^"};  // нижній курсор - стрілка чотири позиції
  lcd_gotoxy(x[0],1);
  lcd_puts(time_chars[0]);
  if (speed>=1000) {speed=999;}; //  перше "^"  - курсор нижній
A0=speed; V0=A0/100; value[0]=(unsigned char)(A0/100); // старший-перший символ зліва
   A1=A0-V0*100; V1= (A1/10); value[1]=(unsigned char)(A1/10); // другий символ зліва
   A2=A1-V1*10; V2=A2; value[2]=(unsigned char)(A2); // третій символ зліва
  while (1) {
    PREV_PINF = PINF;
```

```
    sprintf(buffer, "%u%u%u", value[0], value[1], value[2]); // три символи для
виводу
    lcd_gotoxy(x[0],0);
    lcd_puts(buffer);
    if (get_key_status(SELECT_PLUS_BTN))             //   select+  UP --- set speed -
      if (!get_prev_key_status(SELECT_PLUS_BTN) || (gSelectPlusCounter == CNTREPEAT))
      {
        if (gSelectPlusCounter == CNTREPEAT)
          delay_ms(80);
        switch (i)
        {
        case 0:          //  ліва_старша=нульова позиція числа page значення=( 0..9)
             if (value[0] < 9)
                value[0]++;
             else value[0] = 0;
        break;
        case 1:                  // друга позиція зліва числа page значення =( 0..9)
              if (value[1] < 9)
                  value[1]++;
              else value[1] = 0;
        break;
        case 2:                  // третя позиція зліва числа page значення =( 0..9)
             if (value[i] < 9)
                value[i]++;
             else value[i] = 0;
        break;
        }
      }
 //-end select+  UP ---set speed ()--
    if (get_key_status(SELECT_MINUS_BTN))             // select-  DOWN--- set speed
--
      if (!get_prev_key_status(SELECT_MINUS_BTN) || (gSelectMinusCounter ==
CNTREPEAT))
      {
        if (gSelectMinusCounter == CNTREPEAT)
          delay_ms(80);
        switch (i)
        {
        case 0:
             if (value[0] > 0)
                 value[0]--;
             else value[0] = 9;
        break;
        case 1:
             if (value[1] > 0)
               value[1]--;
             else value[1] = 9;
        break;
        case 2:
             if (value[i] > 0)
                 value[i]--;
             else value[i] = 9;
        break;
        }
     }
//-end select-  DOWN--- set speed ()-
    if (get_key_status(MENU_ENTER_BTN))     // enter--- set speed ()-
      if (!get_prev_key_status(MENU_ENTER_BTN))
      {
        if (i==2)    // кількість символів числа =3
        {
```

```
         speed = (unsigned int)(value[0])*100 ;
         speed = speed +(unsigned int)(value[1])*10;
         speed = speed +(unsigned int)(value[2]);
           if (speed >= 1000)
             speed =999;
             lcd_gotoxy(x[i],1);              // затерти курсор
             lcd_putchar(' ');
          return(speed);                       // сформоване значення
        }
        lcd_gotoxy(x[i],1);
        lcd_putchar(' ');
        lcd_gotoxy(x[++i],1);
        lcd_puts(time_chars[i]);
      }
  //-end enter--- set speed
    if (get_key_status(ESC_BTN))       // exit --set_page()-
    {
      PREV_PINF = PINF;
      return;
    }
  }
}
//--end exit --- set speed

 void  Red(unsigned int  dutyr)  // функція для PWM - red
{
    RedPwm = (ICR3/100)* dutyr;
     if (RedPwm==0) RedPwm=1;
     redpwml=(char)(RedPwm);             // Dred% pwm  RED - low byte
     redpwmh=(char)((RedPwm)>>8);   // Dred% pwm  RED - high byte
     return;
  }

  void  Green(unsigned  int  dutyg)   // функція для PWM - Green
{
     GreenPwm = (ICR3/100)* dutyg;
     if (GreenPwm==0) GreenPwm=1;
     greenpwml=(char)(GreenPwm);
     greenpwmh=(char)((GreenPwm)>>8);   //  Dgreen% pwm  GREEN - high byte
     return;
 }

  void  Blue (unsigned  int  dutyb)   // функція для PWM - Blue
{
     BluePwm = (ICR3/100)* dutyb;
     if (BluePwm==0) BluePwm=1;
     bluepwml=(char)(BluePwm);
     bluepwmh=(char)((BluePwm)>>8);    // Dblue% pwm  BLUE - high byte
     return;
  }

void fxcalculator (void ) // функція для обчислення значень регістрів
{
 if (fxx<100) {
     N=64;  // для частот 1…100гц
     TCCR3B=0x1B;  // n=64
     ICR3top=(3680000/(fxx*N))-1;
     if (ICR3top==0) ICR3top=1;
     ICR3H=(char)((ICR3top)>>8); // - high b
     ICR3L=(char)(ICR3top); // - low byte
    } else {
```

```
     N=1; // для частот 100…30000гц
     TCCR3B=0x19;    // n-prescale=1
     ICR3top= (3680000/(fxx*N))-1;
     if (ICR3top==0) ICR3top=1;
    ICR3H =(char)(( ICR3top)>>8); /*high byte*/ ICR3L =(char)( ICR3top); /*low byte*/
    }  }
 //--end fxcalculator
void savesetup(void) { // функція menu save_setup
     fxx=1.2; rom_fxx=fxx; /* запис в ROM значення частоти* / zvyk();
  }

void savespeed(void) { // функція menu save_setup
    rom_speed=speed;    // запис в ROM значення швидкості проходження тестів
    zvyk();
  }

void main_menu(void)   // ГОЛОВНЕ МЕНЮ і його підменю
{
   char *menu_items[7]=  { " SET Fx < 300Hz ", " SET Fx > 300Hz ", "SAVE SETUP",
"SAVE SPEED", "SET TEST SPEED", "TEST RGB"};
   unsigned char x_pos[] = {0, 0, 3, 3, 1, 4};   // координати для items menu
   int sel=0;                                    // по замовчуванню ставити 0
    while(1) {
       PREV_PINF=PINF;   // кн. на порті F
       lcd_gotoxy(0,0); lcd_putsf("***** MENU *****"); lcd_gotoxy(x_pos[sel],1);
       lcd_puts(menu_items[sel]);
       if (get_key_status(SELECT_PLUS_BTN))  // натиснута кл. "select+" main_menu()
        {
          if (!get_prev_key_status(SELECT_PLUS_BTN) || (gSelectPlusCounter ==
CNTREPEAT)) {
             if (gSelectPlusCounter == CNTREPEAT) delay_ms(80);
             if (sel<5) // кількість пунктів (меню - 1)
               sel++;
             else sel = 0;
             lcd_clear();
         } }
       // -end "select+" - main_menu()-
        if (get_key_status(SELECT_MINUS_BTN)) // натиснута кл. "select-" main_menu()
         {
      if (!get_prev_key_status(SELECT_MINUS_BTN) || (gSelectMinusCounter ==
CNTREPEAT)) {
                if (gSelectMinusCounter == CNTREPEAT) delay_ms(80);
                if (sel>0) sel--;
                else sel = 5;   // кількість пунктів (меню - 1)
                lcd_clear();
             } }
         // -end select(-) -main_menu()
        if (get_key_status(MENU_ENTER_BTN)) // натиснута кл. "enter" main_menu()
          {  if (!get_prev_key_status(MENU_ENTER_BTN)) {
               switch(sel) {
                        case 0:
                           lcd_clear();             // модуль SET Fx< 300 Hz
                           lcd_gotoxy(0,0); lcd_putsf(" Fpwm="); start_p=set_page();
                           fxx= (float)start_p; // значення частоти
                           fxx=fxx/10; fxcalculator();
                        break;  // end модуль SET Fx
                        case 1:
                           lcd_clear(); // модуль SET Fx >300 Hz
                           lcd_gotoxy(0,0); lcd_putsf(" Fpwm=");
                          start_p=set_page2(); fxx=(float)start_p; // значення частоти
                           fxcalculator();
```

```
                    break;
                    case 2:
                      savesetup();  // записати налаштування в память
                      lcd_clear();  // модуль SET Fx >300 Hz
                      lcd_gotoxy(4,0); lcd_putsf("Save OK!"); delay_ms(900);
                    break;
                    case 3:
                      savespeed(); // записати в память значення затримки
                      lcd_clear(); // модуль SET Fx >300 Hz
                      lcd_gotoxy(2,0); lcd_putsf("Save speed OK!"); delay_ms(900);
                    break;
                    case 4:
                      lcd_clear(); // модуль SET Fx >300 Hz
                      lcd_gotoxy(0,0); lcd_putsf(" DELAY="); set_speed();
                    break;
                    case 5:
                      lcd_clear(); // модуль SET Fx >300 Hz
                      lcd_gotoxy(0,0); lcd_putsf("Testing....wait!"); test(speed);
                    break; // вставити модуль для п. меню
                    default: break; // вставити модуль для п. меню
                  } } }
          // end enter-main_menu()
         if (get_key_status(ESC_BTN)) { // натиснута кл. "exit" main_menu()
              if (!get_prev_key_status(ESC_BTN)) {
   TIMSK&=~(1<<6);  // disable Timer 2 overflow interrupt service routine
                   return;   // back to working mode
                } } //-end exit-main_menu()-
    } }

void main(void) {
// Input/Output Ports initialization
PORTA=0x00; DDRA=0x00; PORTB=0x00; DDRB=0x00; PORTC=0x00; DDRC=0x00;
PORTD=0x00;  DDRD=0x00; PORTE=0x00; DDRE=0x38; PORTF=0x00;  DDRF=0x00;
PORTG=0x00;  DDRG=0x00;
// Timer/Counter 0 initialization
ASSR=0x00;  TCCR0=0x00; TCNT0=0x00; OCR0=0x00;
// Timer/Counter 1 initialization
TCCR1A=0x82;      // fast PWM = 14
TCCR1B=0x19; TCNT1H=0x00; TCNT1L=0x00; ICR1H=0x07; ICR1L=0xCF; // частота
OCR1AH=0x03; OCR1AL=0xFF; //  шпаруватість- меандр
OCR1BH=0x00; OCR1BL=0x00;
// Timer/Counter 2 initialization
TCCR3A=0xAA; TCCR3B=0x19; // n-prescale=1   TCCR3B=0x1A; // n=8
TCNT3H=0x00;  TCNT3L=0x00;
ICR3H=0x8F;  ICR3L=0xBF;   // частота 100 Hz output    8FBF
OCR3AH=0x00; OCR3AL=0x00; OCR3BH=0x00; OCR3BL=0x00; OCR3CH=0x00; OCR3CL=0x00;
// External Interrupt(s) initialization
// INT0: Off , INT1: Off, INT2: Off, INT3: Off, INT4: Off, INT5: Off, INT6: Off
EICRA=0x00;  EICRB=0x00;  EIMSK=0x00;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00; ETIMSK=0x04;  // TIM3_OVF
// Analog Comparator initialization
ACSR=0x80; SFIOR=0x00;
DDRC = 0x60; // Port C RC5, RC6 ->output used for LCD
DDRA = 0x0F; // D0-D3 -> output used for LCD
//select reference voltage for ADC
ADMUX|=(0<<REFS1)|(0<<REFS0);    // AREF pin - опорна напруга 5в
ADMUX=ADC_VREF_TYPE & 0xff;    // Bit 5 - ADLAR=0 - 10 бітне АЦП
ADCSRA=0x85; // Bit 7 - ADEN: ADC Enable // Bits 2:0 - ADPS2:0: ADC Prescaler Select
Bits=101 K=32 // ADC Clock frequency: 4MHz/32= 125,000kHz (50kHz-200kHz=normal)
lcd_init(); // ініціалізація LCD
```

```
lcd_clear(); lcd_gotoxy(0,0); lcd_putsf("RGB STRIP TESTER");
delay_ms(700); delay_ms(700);
fxx=rom_fxx; // прочитати опорну частоту з  памяті в змінну  fxx
fxcalculator(); // функція для обчислення значень регістрів TCCR3B, ICR3H, ICR3L
/* ініціалізація кнопок menu на порті F */ BTNS_PORT=(1<<MENU_ENTER_BTN)|
(1<<SELECT_PLUS_BTN)|(1<<SELECT_MINUS_BTN)|(1<<ESC_BTN);
BTNS_PORT_DDR&=~(1<<MENU_ENTER_BTN)&~(1<<SELECT_PLUS_BTN)&~(1<<SELECT_MINUS_BTN)&~(1<
<ESC_BTN);
/* Global enable interrupts */ #asm("sei")
while (1) {
PREV_PINF=PINF; k=get_key_status(MENU_ENTER_BTN); // якщо натиснути клавішу ENTER то
в МЕНЮ
         if (k==1) {
   lcd_clear(); TIMSK |= (1<<6); // enable Timer 2 overflow interrupt service routine
          main_menu(); // головне меню
          TIMSK&=~(1<<6); // disable Timer 2 overflow interrupt service routine
          lcd_clear(); }
  lcd_gotoxy(0,0); lcd_putsf("                "); lcd_gotoxy(0,0); // lcd_putsf("F");
     fx=(int)ICR3L; // частота pwm
     ICR3 = ((ICR3H & 0xffff)<<8)|ICR3L; // fx= ICR3 = ICR3H=0x0E; ICR3L=0x5f, {3679}
     if (fxx>1) { tx=(1000000/(fxx)); od='u'; }
     else { tx=(1000/fxx); od='m'; };
     if (fxx<=300)
     {sprintf(buffer,"%.1fHz ", fxx);}
     else sprintf(buffer,"%.0fHz ", fxx);
     lcd_puts(buffer); delay_ms (500); lcd_putsf("T");
     if (fxx==100) {tx=10000;};
     sprintf(buffer,"%ld%cS", tx,od); lcd_puts(buffer); lcd_gotoxy(0,1);
     lcd_putsf("                ");
     ADC_INPUT = 0x00; // порт PF0
     adc_in= read_adc(ADC_INPUT); // міряємо і отримуємо результат перетворення
     indication=adc_in/0.1024; // коефіцієнт залежить від опорної напруги
     tinf= indication/1000;
Dgreen= (unsigned int)((tinf/2/vref)*100); // перевірити опорну
  lcd_gotoxy(0,1); sprintf(buffer," R%u%%", Dgreen); // buffer for LCD - ADC - Dblue
     lcd_puts(buffer); ADC_INPUT = 0x05; // порт PF5
     adc_in= read_adc(ADC_INPUT); // міряємо і отримуємо результат перетворення
     indication=adc_in/0.1024; // коефіцієнт залежить від опорної напруги
 tinf= indication/1000; Dred= (unsigned int)((tinf/2/vref)*100); // перевірити опорну
     lcd_puts(buffer); ADC_INPUT = 0x06; // порт PF6 blue
     adc_in= read_adc(ADC_INPUT); // міряємо і отримуємо результат перетворення
     indication=adc_in/0.1024; // коефіцієнт залежить від опорної напруги
 tinf=indication/1000; Dblue=(unsigned int)((tinf/2/vref)*100); // перевірити опорну
     sprintf(buffer," B%u%%", Dblue); // buffer for LCD - ADC - Dblue
     lcd_puts(buffer); RedPwm = (ICR3/100)*Dred;
     if (RedPwm==0) RedPwm=1;
     redpwml=(char)(RedPwm); // Dred% pwm  RED - low byte
     redpwmh=(char)((RedPwm)>>8); // Dred% pwm  RED - high byte
     GreenPwm = (ICR3/100)*Dgreen;
     if (GreenPwm==0) GreenPwm=1;
greenpwml=(char)(GreenPwm); greenpwmh=(char)((GreenPwm)>>8); // Dgreen% pwm GREEN
high byte
     BluePwm = (ICR3/100)*Dblue;
     if (BluePwm==0) BluePwm=1;
     bluepwml=(char)(BluePwm);
     bluepwmh=(char)((BluePwm)>>8); // Dblue% pwm BLUE - high byte
    }; }
```