МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Інститут інженерії та інформаційних технологій

Кафедра комп’ютерної інженерії та електромеханіки

**КВАЛІФІКАЦІЙНА РОБОТА**

на тему:

**Пристрій для перевірки блоків живлення з інтегрованою   
системою вимірювання і регулювання навантаження**

Рівень вищої освіти другий (магістерський)

Спеціальності 123 «Комп’ютерна інженерія»

Освітня програма «Комп’ютерна інженерія»

Виконав: студент групи МгКІ-23.

Кошелюк Максим Васильович

(прізвище та ініціали)

Керівник: к.т.н., доц. Стаценко Д.В.

Рецензент.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(науковий ступінь, вчене звання, прізвище та ініціали)

Київ 2024

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Інститут інженерії та інформаційних технологій .

Кафедра комп’ютерної інженерії та електромеханіки .

Спеціальність 123 «Комп’ютерна Інженерія».

Освітня програма «Комп’ютерна Інженерія»

**ЗАТВЕРДЖУЮ**

Завідувач кафедри КІЕМ

\_\_\_\_\_\_\_\_\_\_\_\_ Дмитро СТАЦЕНКО

“\_\_\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 року

**ЗАВДАННЯ**

**НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ**

Кошелюку Максиму Васильовичу

1. Тема кваліфікаційної роботи пристрій для перевірки блоків живлення з інтегрованою системою вимірювання і регулювання навантаження

.

Науковий керівник роботи \_Стаценко Дмитро Володимирович, к.т.н., доцент . (прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

.

затверджені наказом вищого навчального закладу від 03.09.2024 № 188-уч.

2. Вихідні дані до кваліфікаційної роботи: технічне завдання, технічна та патентна література

3. Зміст кваліфікаційної роботи (перелік питань, які потрібно опрацювати):   
Аналіз існуючих пристроїв тестування блоків живлення ПК, Огляд та аналіз інструментів для створення пристрою тестування блоків живлення ПК.

4. Дата видачі завдання \_\_\_\_\_\_\_\_\_\_\_\_2024

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів кваліфікаційної роботи | Терміни виконання етапів | Примітка про виконання |
| 1 | Вступ | 09.09.2024 |  |
| 2 | РОЗДІЛ 1. АНАЛІЗ ПРИСТРОЇВ  ТЕСТУВАННЯ БЛОКІВ ЖИВЛЕННЯ ПК | 20.09.2024 |  |
| 3 | РОЗДІЛ 2. ОГЛЯД ТА АНАЛІЗ  ІНСТРУМЕНТІВ ДЛЯ СТВОРЕННЯ  ПРИСТРОЮ ДЛЯ ТЕСТУВАННЯ БЛОКІВ ЖИВЛЕННЯ ПК | 05.10. 2024 |  |
| 4 | РОЗДІЛ 3. РОЗРОБКА ПРИСТРОЮ ДЛЯ ПЕРЕВІРКИ БЖ ПК | 25.10.2024 |  |
| 5 | Висновки | 28.10.2024 |  |
| 6 | Оформлення (чистовий варіант) | 31.10.2024 |  |
| 7 | Подача кваліфікаційної роботи науковому керівнику для відгуку | 01.11.2024 |  |
| 8 | Подача кваліфікаційної роботи для рецензування (за 14 днів до захисту) | 09.11.2024 |  |
| 9 | Перевірка кваліфікаційної роботи на наявність ознак плагіату та текстових співпадінь (за 10 днів до захисту) | 11.11.2024 |  |
| 10 | Подання кваліфікаційної роботи на завідувачу кафедри (за 7днів до захисту) | 18.11.2024 |  |

З завданням ознайомлений:

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_ \_** Максим КОШЕЛЮК.

( підпис ) (Власне ім’я та ПРІЗВИЩЕ)

Науковий керівник \_\_\_\_\_\_\_\_\_\_\_\_ \_\_Дмитро СТАЦЕНКО.

( підпис ) (Власне ім’я та ПРІЗВИЩЕ)

# АНОТАЦІЯ

**Кошелюк М.В. Пристрій тестування блоків живлення ПК. – Рукопис.**

Кваліфікаційна робота за спеціальністю 123 – Комп’ютерна інженерія освітньої програми «Комп’ютерні системи та мережі». – Київський національний університет технологій та дизайну, Київ, 2024 рік.

Робота присвячена дослідженню і розробці пристрою для тестування блоків живлення персональних комп'ютерів. Пристрій забезпечує вимірювання напруги та струму на кожному каналі, а також загальну потужність блоку живлення.

У першому розділі детально розглянуто існуючі пристрої для тестування блоків живлення, проведено аналіз їх характеристик та обмежень.  
Другий розділ присвячений вибору інструментів та методів для розробки пристрою, включаючи огляд програмного забезпечення MikroC та мікроконтролера PIC16F877A.

У третьому розділі описано розробку проекту, створення електричної схеми та друкованої плати, а також надано інструкції щодо використання і калібрування пристрою.

Пояснювальна записка виконана в текстовому редакторі Microsoft Word, у роботі використані програми для моделювання та тестування, такі як Proteus, sPlan та Sprint-Layout.

*Ключові слова: тестер блоків живлення, мікроконтролер PIC, вимірювання напруги, вимірювання струму, калібрування.*

**ABSTRACT**

**Koshelyuk M.V. Device for Testing PC Power Supplies. – Manuscript.**

Qualification work for the degree in specialty 123 – Computer Engineering, educational program "Computer Systems and Networks." – Kyiv National University of Technologies and Design, Kyiv, 2024.

This thesis is dedicated to the research and development of a device for testing PC power supplies. The device measures voltage and current on each channel, as well as the total power of the power supply.

The first chapter provides a detailed overview and analysis of existing power supply testing devices, examining their features and limitations.  
The second chapter focuses on the selection of tools and methods for the device development, including an overview of the MikroC software and the PIC16F877A microcontroller.

The third chapter describes the development of the project, the creation of the electrical schematic and printed circuit board, and provides instructions for the use and calibration of the device.

The explanatory note was prepared in Microsoft Word, and the work utilized software for modeling and testing, such as Proteus, sPlan, and Sprint-Layout.

*Keywords: power supply tester, PIC microcontroller, voltage measurement, current measurement, calibration.*

**ЗМІСТ**

|  |  |
| --- | --- |
| ВСТУП | 8 |
| РОЗДІЛ 1. АНАЛІЗ ПРИСТРОЇВ ТЕСТУВАННЯ БЛОКІВ ЖИВЛЕННЯ ПК | 10 |
| 1.1 Різновиди пристроїв тестування блоків живлення ПК | 10 |
| 1.2 Огляд блоків живлення ПК | 13 |
| 1.3 Огляд існуючих пристроїв тестування блоків живленняПК | 16 |
| 1.4 Огляд предметного середовища | 18 |
| ВИСНОВОК ДО РОЗДІЛУ 1 | 21 |
| РОЗДІЛ 2. ОГЛЯД ТА АНАЛІЗ ІНСТРУМЕНТІВ ДЛЯ СТВОРЕННЯ ПРИСТРОЮ ДЛЯ ТЕСТУВАННЯ БЛОКІВ ЖИВЛЕННЯ ПК | 22 |
| 2.1 Етапи створення приладу тестування БЖ ПК | 22 |
| 2.2 Вибір та аналіз доступних технологій та програмного забезпечення | 24 |
| 2.3 Мікроконтролери | 25 |
| 2.4 Загальні характеристики архітектури PIC | 29 |
| 2.4.1 Мікроконтролери сімейства AVR | 32 |
| 2.5 Програмування мікроконтролерів. Вибір середовища розробки | 34 |
| 2.5.1 MikroC Pro for PIC | 35 |
| 2.5.2  MPLAB X IDE | 38 |
| 2.5.3 Atmel Studio | 40 |
| 2.5.4 Arduino IDE | 42 |
| 2.6 sPlan | 44 |
| 2.7 Proteus | 47 |
| 2.8 Мова програмування мікроконтролера | 50 |
| ВИСНОВОК ДО РОЗДІЛУ 2 | 53 |
| РОЗДІЛ 3. РОЗРОБКА ПРИСТРОЮ ДЛЯ ПЕРЕВІРКИ БЖ ПК | 54 |
| 3.1 Розробка принципової електричної схеми | 54 |
| 3.2 Мікроконтролер PIC16F877A | 55 |
| 3.2.1 Архітектура PIC16F877A | 56 |
| 3.2.2Переваги використання PIC16F877A та застосування в проекті | 58 |
| 3.3 Електрична принципова схема пристрою | 60 |
| 3.4 Основні вузли електронної схеми | 62 |
| 3.5 Розробка програмного забезпечення для PIC16F877A | 64 |
| 3.6 Опис програмного коду | 66 |
| 3.7 Створення друкованої плати | 69 |
| 3.8 Використання тестера для перевірки блоків живлення | 71 |
| 3.8.1 Калібрування тестера для перевірки блоків живлення | 74 |
| ВИСНОВОК ДО РОЗДІЛУ 3 | 77 |
| ВИСНОВКИ | 78 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ | 80 |
| ДОДАТКИ | 82 |
| ДОДАТОК А | 83 |
|  |  |

**ВСТУП**

**Актуальність роботи.** Блок живлення є критично важливим компонентом комп'ютерної системи, оскільки забезпечує енергією всі елементи, включаючи материнську плату, процесор, відеокарту та інші периферійні пристрої. Неправильна робота блоку живлення може призвести до нестабільної роботи системи, а в деяких випадках і до пошкодження апаратного забезпечення. Регулярне тестування блоку живлення є важливою складовою підтримки його ефективності та надійності. Для цього необхідно використовувати спеціалізовані пристрої, які можуть точно вимірювати параметри електроживлення, такі як напруга, струм та загальна потужність.

**Об’єктом дослідження** є процеси вимірювання параметрів електроживлення блоків живлення ПК.

**Предметом дослідження** є підходи та методи тестування блоків живлення, а також принципи роботи мікроконтролерів у системах вимірювання.

**Метою дослідження** є розробка пристрою для тестування блоків живлення ПК на базі мікроконтролера PIC, який зможе забезпечити точні вимірювання напруги, струму та загальної потужності. Пристрій буде оснащений дисплеєм для відображення даних, а також функціями регулювання струму навантаження та зміни режимів вимірювання.

**Методологічною і теоретичною основою дослідження** є принципи електротехніки, електроніки, а також алгоритми програмування для мікроконтролерів, які дозволяють реалізувати точні вимірювання та контроль параметрів живлення.

**Наукова новизна** полягає в розробці унікального пристрою, який не тільки вимірює основні електричні параметри, але і забезпечує високу точність та надійність результатів, завдяки використанню мікроконтролера PIC.

**Практична значимість** даної розробки полягає в можливості забезпечення ефективного тестування блоків живлення для комп'ютерних систем. Тестер може бути корисним як для індивідуальних користувачів, так і для сервісних центрів, оскільки дозволяє точно оцінити роботу блоку живлення, уникнути проблем з апаратним забезпеченням і підвищити загальну стабільність системи. Додатково, пристрій може бути використаний у різних областях, де потрібні точні вимірювання електричних параметрів.

**Структура та обсяг роботи.** Дипломна робота магістра складається зі вступу, 3 розділів та висновків по них, загальних висновків, списку використаних джерел та додатків. Основний текст роботи викладений на 79 сторінках, містить 21 рисунок, список джерел з 30 найменувань. Загальний обсяг роботи, враховуючи додаток, складає 110 аркуша.

**РОЗДІЛ 1.**

**АНАЛІЗ ПРИСТРОЇВ ТЕСТУВАННЯ БЛОКІВ ЖИВЛЕННЯ ПК**

* 1. **Різновиди пристроїв тестування блоків живлення ПК**

Аналогові тестери — це прості пристрої, які використовують стрілочні індикатори для вимірювання напруги та струму. Вони зазвичай працюють за принципом електромеханічного вимірювання і не потребують батарей або зовнішнього живлення. Використовують магнітний або електромагнітний механізм для відображення значень. Мають обмежений діапазон вимірювань, зазвичай 0-15 В для напруги та 0-10 А для струму [1].

Переваги:

* Прості в експлуатації та інтерпретації результатів, що робить їх зручними для початківців.
* Мають високу надійність, оскільки не залежать від електроніки.

Недоліки:

* Обмежена точність — стрілочні індикатори можуть бути важкими для точного читання.
* Відсутність додаткових функцій, таких як автоматичне калібрування або зберігання даних.

Цифрові мультиметри

Цифрові мультиметри (ДМ) є універсальними вимірювальними пристроями, які можуть вимірювати різні електричні параметри, включаючи постійну та змінну напругу, струм, опір, частоту та інші. Сучасні моделі мають функції автоматичного вибору діапазону вимірювання, що дозволяє з легкістю вимірювати різні параметри. Деякі мультиметри оснащені додатковими функціями, такими як вимірювання температури або частоти [2].

Переваги:

* Висока точність і надійність вимірювань, що робить їх ідеальними для професійного використання.
* Зручні в експлуатації завдяки цифровому дисплею, який чітко відображає результати.

Недоліки:

* Можуть бути дорожчими, ніж аналогові моделі.
* Потребують живлення від батарей, що може бути незручним в умовах обмеженого доступу до електрики.

Спеціалізовані тестери блоків живлення. Ці тестери спеціально розроблені для перевірки роботи блоків живлення ПК. Вони зазвичай мають кілька входів для підключення до різних виходів БЖ, таких як 3.3V, 5V, 12V, а також можуть контролювати загальну потужність. Можуть мати звукові або світлові індикатори, які сповіщають про несправності. Деякі моделі мають функцію тестування під навантаженням, що дозволяє перевіряти стабільність роботи БЖ під різними умовами [3].

Переваги:

* Швидка та проста перевірка основних параметрів блоку живлення без складних налаштувань.
* Чітке і зрозуміле відображення результатів.

Недоліки:

* Обмежені можливості вимірювання — можуть не вимірювати струм та опір.
* Зазвичай не підходять для детального аналізу, а тільки для базових перевірок.

Професійні стенди для тестування

Ці пристрої призначені для проведення комплексного тестування блоків живлення. Вони можуть імітувати різні навантаження, що дозволяє тестувати БЖ під реальними умовами. Висока точність вимірювань, можливість підключення до комп'ютера для моніторингу результатів. Можуть мати різні порти для підключення до інших пристроїв для тестування [4].

Переваги:

* Можливість проведення детальних тестів, які допомагають виявити слабкі місця блоку живлення.
* Підходять для професійного використання у сервісних центрах та лабораторіях.

Недоліки:

* Висока вартість, що робить їх менш доступними для домашнього використання.
* Складність у налаштуванні та використанні.

Модулі тестування

Компактні пристрої, які використовуються для швидкої перевірки основних параметрів блоку живлення. Модулі можуть підключатися безпосередньо до роз'ємів блоку живлення та швидко показувати результати на дисплеї. Зазвичай невеликого розміру, що робить їх зручними для перенесення. Деякі моделі мають вбудовані світлодіоди для індикації роботи.

Переваги:

* Легкість використання, що дозволяє швидко перевірити основні параметри БЖ.
* Доступність за ціною — зазвичай дешевші, ніж повнофункціональні тестери.

Недоліки:

* Обмежена функціональність; зазвичай не можуть вимірювати струм або опір.
* Не підходять для глибокого аналізу або тестування під навантаженням.

Пристрої з функцією навантаження

Ці тестери здатні імітувати різні навантаження на блоки живлення, що дозволяє перевірити їхню роботу під реальними умовами. Можуть мати різні режими навантаження для тестування БЖ під різними умовами. Деякі моделі оснащені вбудованими датчиками для вимірювання температури та напруги [5].

Переваги:

* Можливість детального аналізу роботи блоку живлення під навантаженням, що дозволяє виявити проблеми, які можуть виникнути під реальними умовами експлуатації.
* Підходять для професійного використання та сервісних центрів.

Недоліки:

* Вартість може бути значною, особливо для моделей з високими характеристиками.
* Складність у налаштуванні та використанні для непідготовлених користувачів.

**1.2 Огляд блоків живлення ПК**

Блоки живлення (БЖ) є ключовими компонентами комп'ютера, що забезпечують постачання необхідної електроенергії для всіх внутрішніх компонентів, таких як материнська плата, процесор, відеокарта та інші периферійні пристрої. Від їх якості та стабільності залежить надійність роботи всієї системи. У цьому розділі розглянуто основні типи блоків живлення для ПК, їх характеристики, класифікацію за різними критеріями та особливості роботи [6].

**Рис1.1** Зовнішній вигляд БЖ

Класифікація блоків живлення ПК

Блоки живлення ПК класифікують за різними параметрами: потужністю, типом конструкції, сертифікацією ефективності та іншими технічними характеристиками.

1. За типом форм-фактора:
   * ATX (Advanced Technology eXtended): Це найпоширеніший тип блоків живлення, який використовується в більшості настільних ПК. Стандарт ATX визначає розміри та розташування компонентів на БЖ, а також забезпечує стандартизовані роз'єми живлення.
   * SFX (Small Form Factor): Блоки живлення цього типу використовуються в компактних корпусах або в ПК форм-фактора Mini-ITX. Вони мають менші розміри, але часто пропонують нижчу потужність у порівнянні з ATX.
   * TFX (Thin Form Factor): Ці блоки живлення призначені для тонких ПК-корпусів. Вони мають вузький і довгий дизайн, щоб вміщатися в компактні системи.
   * Flex ATX: Менша версія стандартного ATX, яка використовується в дуже компактних системах або серверах. Забезпечує економію місця, але може мати обмежену потужність [7].
2. За потужністю:
   * Маломощні блоки живлення (до 300 Вт): Використовуються у офісних або компактних системах, де немає потреби в великій потужності для підтримки високопродуктивних компонентів.
   * Середньомощні блоки (300-500 Вт): Найпоширеніший варіант для більшості домашніх ПК. Ці блоки живлення підходять для систем з однією відеокартою середнього рівня.
   * Високомощні блоки живлення (500-1200 Вт і більше): Використовуються в ігрових, робочих станціях або серверних системах, де є потреба в живленні потужних компонентів, таких як кілька відеокарт, великий обсяг ОЗП та процесор з високими вимогами до енергоживлення.

Основні технічні характеристики блоків живлення

1. Потужність(Вт):   
   Це основний показник, що вказує на загальну кількість електричної потужності, яку блок живлення може надати системі. Важливо підбирати БЖ із достатнім запасом потужності, щоб забезпечити стабільну роботу всіх компонентів [8].
2. Ефективність (сертифікація 80 Plus):  
   Ефективність блоків живлення вимірюється в процентах та визначає, скільки електроенергії, що надходить у блок живлення, перетворюється на корисну енергію для компонентів, а скільки втрачається у вигляді тепла. Для підвищення ефективності роботи БЖ існує сертифікація 80 Plus, яка має кілька рівнів:
   * 80 Plus Standard (ефективність ≥ 80%)
   * 80 Plus Bronze (ефективність ≥ 82%)
   * 80 Plus Silver (ефективність ≥ 85%)
   * 80 Plus Gold (ефективність ≥ 87%)
   * 80 Plus Platinum (ефективність ≥ 89%)
   * 80 Plus Titanium (ефективність ≥ 90%)

Висока ефективність не лише знижує втрати енергії, але й зменшує тепловиділення, що сприяє зменшенню навантаження на систему охолодження.

1. Лінії живлення (12V, 5V, 3.3V):

Блоки живлення ПК зазвичай постачають кілька різних ліній живлення для компонентів. Основною є лінія 12V, яка живить процесор, відеокарту та інші високопродуктивні компоненти. Лінії 5V і 3.3V використовуються для живлення менших пристроїв, таких як материнська плата, жорсткі диски та USB-порти.

1. Модульність кабелів:

Блоки живлення можуть мати немодульну, напівмодульну або повністю модульну конструкцію. У немодульних БЖ всі кабелі підключені до блоку і не можуть бути від'єднані. У напівмодульних кабелі живлення материнської плати та процесора незмінно підключені, а інші кабелі можуть бути від'єднані. У повністю модульних БЖ всі кабелі можна від'єднати, що дозволяє краще організувати простір усередині корпусу[9].

Різновиди блоків живлення

1. Стандартні (офісні) блоки живлення:

Призначені для використання в офісних ПК та виконання базових завдань. Вони мають невелику потужність, зазвичай до 400 Вт, і не завжди мають сертифікацію 80 Plus. Їхня головна перевага — доступна ціна, однак їх може не вистачати для високопродуктивних комп'ютерів.

1. Геймерські блоки живлення:

Призначені для ігрових систем, що вимагають високої потужності та стабільної роботи під навантаженням. Вони зазвичай мають сертифікацію від 80 Plus Bronze до 80 Plus Titanium і потужність від 500 Вт до 1000 Вт. Такі БЖ мають кращі компоненти, такі як японські конденсатори та більш надійні вентилятори, що забезпечує довговічність та стабільну роботу.

1. Ентузіастські блоки живлення:

Призначені для високопродуктивних систем із кількома відеокартами (SLI або CrossFire), процесорами високого рівня та великою кількістю додаткових пристроїв. Їхня потужність зазвичай перевищує 1000 Вт, вони оснащені потужною системою охолодження і часто мають повністю модульну конструкцію[10].

1. Блоки живлення для серверів:

Ці блоки живлення призначені для використання у серверних середовищах, де потрібна безперебійна робота під постійним навантаженням. Вони часто мають резервування (подвійні блоки живлення), щоб у разі виходу з ладу одного з них інший автоматично активувався і забезпечував роботу системи.

**1.3 Огляд існуючих пристроїв тестування блоків живлення ПК**

У процесі дослідження ринку пристроїв для тестування блоків живлення ПК було виявлено кілька моделей, які можуть служити основою для мого проекту. Аналіз цих пристроїв допомагає визначити їх переваги та недоліки, а також порівняти з власними цілями розробки.

1. Tester для ATX блоків живлення

Цей пристрій дозволяє перевіряти основні параметри блоків живлення стандарту ATX, включаючи напругу на лініях 12V, 5V і 3.3V. Тестер використовує стандартний підхід для вимірювання напруги, що є важливим для оцінки стабільності блоку живлення при підключенні до комп'ютера. Основні функції пристрою:

* Вимірювання напруги на кожному каналі.
* Проста індикація справності або несправності блоку живлення через світлодіоди.
* Відсутність можливості контролю сили струму та зворотного зв’язку для регулювання навантаження[11].

Основний недолік цього тестера полягає в тому, що він не забезпечує комплексної діагностики всіх параметрів блоку живлення, таких як сила струму та загальна потужність. Це робить пристрій менш універсальним для професійного використання.

2. Електронна навантаження для блоків живлення

Цей пристрій дозволяє перевіряти блоки живлення під навантаженням, що дає змогу оцінити їхню стабільність під робочими умовами. Основні характеристики цього пристрою включають:

* Можливість точного налаштування навантаження.
* Вимірювання напруги і сили струму на різних лініях блоку живлення.
* Підтримка контролю за стабільністю напруги під час збільшення навантаження.

Електронна навантаження є корисним інструментом для перевірки блоків живлення в умовах, близьких до реальної експлуатації. Проте, даний пристрій не відображає всі параметри у реальному часі та не має зворотного зв’язку для динамічного регулювання[12].

Порівняння з данним проєктом

Представленний проект тестера для блоків живлення ПК відрізняється від описаних вище пристроїв кількома важливими особливостями:

1. Більш детальне вимірювання параметрів. Тестер використовує мікроконтролер для точного вимірювання напруги, сили струму та потужності на кожному каналі блоку живлення. Відсутність зворотного зв’язку для стабілізації параметрів надає простоту конструкції, але забезпечує точне відображення значень без зовнішнього втручання.
2. Індивідуальні налаштування. Завдяки перемінному резистору можна регулювати навантаження блоку живлення вручну. Користувач також може перемикати режими вимірювання за допомогою кнопок інтерфейсу, що надає додаткову гнучкість у налаштуваннях.
3. Довготривале навантаження до 500 Вт. тестер спроектований для роботи під навантаженням до 500 Вт, що робить його придатним для тестування високопотужних блоків живлення, які використовуються у сучасних комп'ютерних системах[13].
4. Покращена фільтрація сигналу. Використання RC-фільтрів допомагає усунути пульсації та високочастотні шуми, які можуть виникати в блоках живлення низької якості. Це дозволяє покращити точність вимірювань та забезпечити надійність результатів.
5. Операційні елементи. Додаткові компоненти, такі як низькоомні резистори та потужні алюмінієві радіатори з вентилятором, забезпечують ефективне охолодження вихідних транзисторів та стабілізацію струму.

Наявні на ринку пристрої для тестування блоків живлення мають свої переваги та обмеження. Вони надають базову інформацію про параметри блоку живлення, такі як напруга і стабільність, але не завжди дозволяють отримати повну картину роботи пристрою під навантаженням. Тестер, заснований на мікроконтролері PIC, пропонує більш точне вимірювання, покращену фільтрацію сигналу, регулювання навантаження та можливість довготривалого тестування, що дозволяє отримати більш детальні і надійні результати.

**1.4 Огляд предметного середовища**

Тестування блоків живлення ПК є важливим процесом, який дозволяє оцінити їхню працездатність і відповідність заявленим технічним характеристикам. Блок живлення відповідає за подачу стабільної напруги та струму на всі компоненти комп'ютера, тому тестування його параметрів є критичним завданням як для виробників, так і для користувачів[14].

Основні завдання тестування блоків живлення

1. Перевірка стабільності напруги на кожному каналі. Блок живлення повинен забезпечувати стабільну напругу для різних компонентів комп'ютера, таких як процесор, материнська плата, відеокарта тощо. Під час тестування вимірюються напруги на лініях 12V, 5V і 3.3V, оскільки саме ці лінії забезпечують живлення основних компонентів. Важливо, щоб напруга залишалася в межах допусків, зазначених виробником, навіть під максимальним навантаженням.
2. Вимірювання сили струму. Важливо перевірити не тільки напругу, але й силу струму на кожній з ліній живлення. Сила струму вказує на те, яку максимальну потужність може забезпечити блок живлення для конкретного каналу. Тестування дозволяє визначити, чи відповідають фактичні параметри заявленим характеристикам і чи здатний блок живлення справлятися з піковими навантаженнями [15].
3. Обчислення загальної потужності. Крім тестування окремих ліній, необхідно виміряти загальну потужність блоку живлення. Це дозволяє оцінити, чи відповідає блок живлення потребам конкретної системи. Тестер повинен забезпечувати точні вимірювання, щоб користувач міг визначити, чи достатня потужність для живлення всіх компонентів комп'ютера, включаючи високопродуктивні відеокарти, процесори та інші периферійні пристрої.

Структура пристрою для тестування блоків живлення

Для реалізації пристрою тестування використовується PIC мікроконтролер, що дозволяє інтегрувати функції вимірювання напруги, струму та обчислення загальної потужності. Програмна частина пристрою розроблена мовою С, що дає можливість реалізувати точні алгоритми для збирання та обробки даних з сенсорів [16].

Основні компоненти пристрою:

* Мікроконтролер PIC: використовується для збору та обробки даних з різних каналів живлення блоку. Мікроконтролер відповідає за управління сенсорами, інтерпретацію сигналів та передачу даних на дисплей.
* Сенсори напруги та струму: ці сенсори підключаються до різних ліній блоку живлення для точного вимірювання електричних параметрів.
* Дисплей: на ньому відображаються результати тестування, включаючи напругу, струм на кожному каналі та загальну потужність блоку живлення.
* Елементи керування: кнопки або регулятори, що дозволяють змінювати режим тестування, вмикати/вимикати тестер, змінювати навантаження або інші параметри.

Програмне забезпечення

Програма, написана мовою С, забезпечує взаємодію з мікроконтролером та управління всіма етапами тестування. Основні функції програмного забезпечення включають:

* Збір даних з сенсорів напруги і струму.
* Обчислення загальної потужності блоку живлення.
* Фільтрація сигналів для усунення можливих перешкод або шумів.
* Виведення результатів на дисплей у реальному часі.
* Захист від перевантаження або короткого замикання.

Особливості пристрою

1. Точність вимірювань: завдяки використанню високоточних сенсорів і мікроконтролера з відповідними алгоритмами, пристрій забезпечує точні вимірювання напруги, струму та потужності. Це дозволяє ефективно оцінити якість роботи блоку живлення та виявити можливі проблеми [17].
2. Гнучкість і налаштування: користувач може змінювати режими тестування за допомогою кнопок або інших елементів керування. Наприклад, можна протестувати блок живлення під різними рівнями навантаження або виміряти параметри для різних ліній живлення окремо.
3. Компактність і зручність у використанні: пристрій спроектований таким чином, щоб бути компактним і легким у використанні. Це дозволяє використовувати його як у сервісних центрах, так і в домашніх умовах.

Таким чином, даний пристрій забезпечує комплексне тестування блоків живлення ПК, надаючи точні вимірювання основних параметрів, таких як напруга, струм і потужність, що допомагає виявляти несправності або підтвердити якість роботи блоку живлення.

**ВИСНОВОК ДО РОЗДІЛУ 1**

У цьому розділі дипломної роботи було проведено всебічний аналіз різновидів пристроїв для тестування блоків живлення ПК, а також огляд їх ключових характеристик та функціональних можливостей. Дослідили, як різні типи тестерів допомагають оцінювати роботу блоків живлення, включаючи вимірювання напруги, сили струму, потужності та інших важливих параметрів. Це дозволило чітко окреслити, якими технічними особливостями повинен володіти новий тестер для забезпечення точних і надійних результатів.

Огляд існуючих блоків живлення ПК показав, що сучасні блоки живлення мають розширений функціонал, зокрема можливості стабілізації напруги, захист від перенапруги, а також різні режими роботи залежно від навантаження. Це підкреслило важливість створення тестера, здатного працювати з різними типами блоків живлення, включаючи як бюджетні, так і високоефективні моделі. Визначення вимог до тестування дозволило зрозуміти, що тестер повинен забезпечувати точні вимірювання при різних рівнях навантаження та можливість довготривалого тестування.

В підрозділі огляду існуючих пристроїв тестування блоків живлення ПК було розглянуто декілька типових тестерів та електронних навантажень, які використовуються на ринку. Аналіз цих пристроїв дав змогу зрозуміти їхні обмеження, зокрема відсутність зворотного зв'язку для контролю навантаження та недостатню гнучкість у налаштуванні параметрів. Данний проект, на відміну від них, передбачає використання мікроконтролера для точного вимірювання параметрів з можливістю відображення даних на дисплеї, що забезпечує більший контроль та точність у тестуванні.

Огляд предметного середовища показав, що тестери блоків живлення здебільшого орієнтовані на стандартні вимірювання, але не завжди можуть надати повний спектр функцій для тестування потужніших блоків живлення або забезпечити довготривалі тести під високим навантаженням. Завдяки використанню PIC мікроконтролера та ряду покращень, таких як RC-фільтри для зменшення шумів і пульсацій, тестер буде забезпечувати точні вимірювання навіть у випадках неякісних блоків живлення.

**РОЗДІЛ 2.**

**ОГЛЯД ТА АНАЛІЗ ІНСТРУМЕНТІВ ДЛЯ СТВОРЕННЯ ПРИСТРОЮ ДЛЯ ТЕСТУВАННЯ БЛОКІВ ЖИВЛЕННЯ ПК**

* 1. **Етапи створення приладу тестування БЖ ПК**

Враховуючи вимоги до основного та додаткового функціоналу, викладені в Розділі 1, прилад для тестування блоків живлення ПК повинен відповідати високим стандартам точності, надійності та зручності у використанні. Основна ідея полягала в тому, щоб створити пристрій, здатний тестувати блоки живлення, вимірюючи напругу, струм і потужність з можливістю роботи як у статичному, так і динамічному режимах. Процес створення приладу пройшов через кілька важливих етапів:

Затвердження вимог проекту. Спочатку було визначено вимоги до функціоналу приладу, який мав відповідати потребам професіоналів для точного тестування блоків живлення. Важливою вимогою стала підтримка вимірювання на кількох каналах, здатність працювати під навантаженням до 500 Вт, а також можливість відображення результатів на дисплеї з налаштуванням режимів через кнопки. Прилад також мав включати функції калібрування для забезпечення точності вимірювань та доступності інформації для користувача [18].

Дослідження ринку. Було проведено огляд існуючих комерційних пристроїв для тестування блоків живлення. Це дало змогу проаналізувати їхні функціональні обмеження, зокрема брак динамічних режимів тестування та обмежений набір налаштувань. В результаті цього етапу стало зрозуміло, що ринок потребує більш гнучкого рішення, яке забезпечить користувачам не лише базове вимірювання, а й розширений функціонал для аналізу продуктивності блоків живлення під різними умовами.

Вибір мікроконтролера Основу пристрою склав мікроконтролер PIC, який забезпечує достатню обчислювальну потужність для виконання вимірювань та керування процесами. Мікроконтролер було обрано завдяки його можливостям роботи з АЦП, підтримці кількох каналів, а також наявності необхідних периферійних інтерфейсів для управління дисплеєм та кнопками. Важливими критеріями вибору були максимальна тактова частота, обсяг вбудованої пам'яті та наявність документації, що спростила процес навчання[19].

Створення електричної схеми. Розробка електричної схеми включала використання операційних підсилювачів для вимірювання струму, польових транзисторів для стабілізації навантаження, а також додавання RC-фільтрів для усунення перешкод. При проектуванні схеми було враховано, що пристрій повинен працювати під високими навантаженнями, до 500 Вт, з надійною системою охолодження. Також було враховано необхідність захисту елементів схеми від перевантаження і короткого замикання, що забезпечить безпеку в роботі.

Програмування та прошивка мікроконтролера. Для контролю та обробки даних використовувалося програмне забезпечення, яке дозволяло відстежувати зміни напруги й струму, а також переходити між різними режимами тестування. У прошивці були реалізовані алгоритми для динамічного вимірювання напруги, статичного режиму та можливості калібрування. Цей етап включав розробку користувацького інтерфейсу, який дозволяє легко перемикатися між режимами, налаштовувати параметри тестування та переглядати результати вимірювань [20].

Процес калібрування. Калібрування є важливим етапом для досягнення точних вимірювань. Включено кілька процедур калібрування:

Калібрування опорної напруги для АЦП: цю процедуру виконано для забезпечення точності аналого-цифрових перетворень. Зовнішнє джерело напруги подавалося на вхід АЦП, а приладу давалася можливість порівняти виміряне значення з еталонним.

Калібрування напруги 5VSB та PG: для цього етапу використовувалося коротке замикання входів, після чого подавалося стандартне значення напруги. Це дозволило усунути відхилення в результатах вимірювання.

Калібрування для кожного каналу вимірювання: для забезпечення точності у широкому діапазоні напруг і струмів вимірювання здійснювалися на кількох етапах з різними рівнями навантаження.

Розробка друкованої плати. Після завершення проектування схеми плата була розроблена за допомогою SprintLayout. Особливу увагу було приділено компонуванню елементів для мінімізації перешкод та забезпечення стабільності роботи під великим навантаженням. Також була передбачена система охолодження для уникнення перегріву під час роботи. Після розробки плати було проведено тестування для виявлення потенційних помилок у з'єднаннях та розташуванні компонентів.

Тестування та усунення помилок. Фінальний етап включав тестування пристрою під різними умовами навантаження. Було перевірено точність вимірювань напруги й струму, стабільність роботи пристрою, а також правильність переходу між статичним і динамічним режимами. Тестування також включало випробування охолодження та роботу пристрою на потужностях до 500 Вт. У випадку виявлення помилок проводилося їх усунення та повторне тестування до досягнення задовільних результатів [21].

Таким чином, кожен з етапів створення пристрою був критично важливим для досягнення необхідної функціональності та точності тестера. Це дозволило створити потужний та гнучкий інструмент для професійного тестування блоків живлення комп'ютерів, який відповідає сучасним вимогам ринку та потребам користувачів.

**2.2 Вибір та аналіз доступних технологій та програмного забезпечення**

Для реалізації проекту створення пристрою для тестування блоків живлення ПК було обрано ряд технологій та програмного забезпечення, що відповідають вимогам та специфікаціям розробки. Пристрій базується на мікроконтролері PIC16F877A, що забезпечує необхідну продуктивність та функціональність.

Програмування мікроконтролера здійснювалося за допомогою MikroC, яка є потужним інструментом для написання та налагодження коду на мові C, що є найпоширенішою мовою для розробки програм для мікроконтролерів. Ця платформа забезпечила зручний інтерфейс, широкий набір бібліотек та функцій, які полегшують роботу з периферійними пристроями.

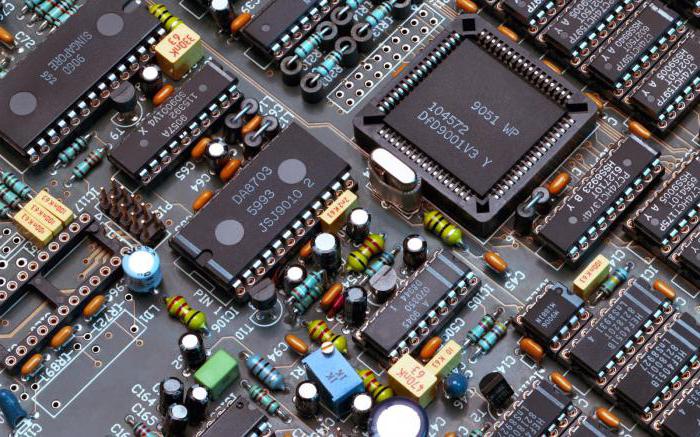
Для створення електричної схеми було обрано програму sPlan 7.0. Це програмне забезпечення дозволяє швидко та зручно проектувати електричні схеми, що є критично важливим етапом у розробці пристрою. sPlan 7.0 надає можливість створювати чисті та професійні схеми, що відповідають стандартам галузі [22].

Для розробки друкованих плат використано Sprint Layout, що забезпечує зручний та інтуїтивно зрозумілий інтерфейс для проектування друкованої плати. Ця програма дозволяє легко розміщувати компоненти, прокладати доріжки та оптимізувати макет плати, що знижує ризик помилок при виготовленні.

На етапі тестування та налагодження використовувалася програма Proteus, яка надає можливість віртуального моделювання електронних схем. Це програмне забезпечення дозволяє проводити симуляцію роботи пристрою до фізичного виготовлення, що значно скорочує час на виявлення та виправлення помилок. Завдяки можливостям Proteus, можна аналізувати роботу мікроконтролера, перевіряти логіку програмного забезпечення та виконувати тести в реальному часі.

**2.3 Мікроконтролери**

Мікроконтролер — це програмований мікроелектронний пристрій, спеціально розроблений для управління різноманітними системами та процесами. Він широко використовується в автоматизованих системах, що потребують точного контролю та швидкої реакції, зокрема в побутовій техніці, медичних апаратах, системах безпеки, засобах зв'язку, електронних музичних інструментах, автомобільній електроніці, комп'ютерних пристроях (таких як клавіатури, принтери, ігрові контролери), автоматичних системах контролю руху, а також у промислових роботах та верстатах. Завдяки своїй універсальності мікроконтролери стали невід'ємною частиною сучасних технологій, забезпечуючи ефективне управління

в різних сферах життя [23].

**Рис. 2.1** Області застосування мікроконтролерів

Більшість сучасних цифрових систем керування будуються на основі мікроконтролерів (МК), які є спеціалізованими мікропроцесорними пристроями, орієнтованими на виконання керуючих функцій. Мікроконтролери об'єднують на одному кристалі високопродуктивний процесор, пам'ять і широкий набір периферійних пристроїв, що дозволяє створювати ефективні системи управління з мінімальними витратами. На сьогодні мікроконтролери є наймасовішими представниками мікропроцесорної техніки, з річним обсягом виробництва близько 2,5 мільярдів одиниць. Виробництвом МК займаються компанії, такі як Intel, Motorola, Philips, Siemens, Atmel, Dallas, Temic, Oki, AMD та інші. Вони пропонують широкий асортимент продукції — від простих 8-розрядних до складніших 32-розрядних моделей на основі CISC- та RISC-процесорів [24].

**Рис. 2.2** Області застосування мікроконтролерів

Найбільш поширеними в масовому виробництві є 8-розрядні однокристальні мікроконтролери. Вони мають подібні обчислювальні ядра, але відрізняються за обсягом і типом вбудованої пам'яті, переліком периферійних пристроїв, які розміщені на кристалі, та іншими технічними характеристиками.

Мікроконтролери поділяються на три основні типи:

1. 8-розрядні вбудовані мікроконтролери: Мають всі необхідні компоненти на одному кристалі, включаючи пам'ять, порти введення/виведення та таймери. Їхні обсяги пам'яті програм зазвичай варіюються від 1 до 128 КБ, а пам'ять даних – від 64 байт до 4 КБ. Ці мікроконтролери використовують малу кількість енергії і працюють в широкому температурному діапазоні. Програмуються вони, як правило, на мовах асемблера або С.
2. 16- і 32-розрядні мікроконтролери: Використовують зовнішню пам'ять для програм та даних, що робить їх придатними для застосувань, де необхідний великий обсяг пам'яті. Такі МК застосовуються в пристроях, наприклад, в контролерах жорстких дисків, що забезпечують буферизацію та зберігання великих обсягів даних.
3. Цифрові сигнальні процесори (DSP): Призначені для обробки аналогових сигналів у реальному часі, виконуючи завдання, що раніше вирішувалися аналоговими схемами. Ці процесори використовуються для обробки звуку, відео, а також у модемах. Вони мають високі вимоги до швидкодії, компактності та ефективної взаємодії з АЦП та ЦАП.

Для підключення зовнішніх пристроїв мікроконтролери використовують як паралельні порти введення/виведення, так і послідовні шини типу I2C або SPI, що дозволяє зменшити кількість ліній підключення, розміри пристроїв та спростити їх розробку. Мікроконтролери використовують три основні види пам'яті: пам'ять для програм, пам'ять для даних і регістри. Кожен із цих типів пам'яті виконує специфічну роль у процесі функціонування мікроконтролера, зберігаючи й надаючи доступ до інформації на різних етапах виконання коду. Пам'ять для програм — це постійна пам'ять, яка містить програмний код і константи. Вміст цієї пам'яті не змінюється під час виконання програми. Пам'ять для даних використовується для збереження змінних під час роботи програми, а регістри включають внутрішні регістри процесора і спеціальні регістри для керування периферійними пристроями [25].

У мікроконтролерах серій MCS-48/51 регістри реалізовані у вигляді окремої області внутрішньої пам'яті. Крім внутрішньої пам'яті, більшість мікроконтролерів підтримують підключення зовнішньої пам'яті для програм і даних. Для збереження програм використовуються різні види постійної пам'яті, зокрема ROM, PROM, EPROM або EEPROM. Такі види пам'яті є незалежними, тому їх вміст зберігається навіть після вимкнення живлення. Програмування мікроконтролерів зазвичай здійснюється за допомогою спеціальних програматорів, підключених до комп'ютера, за винятком мікроконтролерів із вбудованою ROM-пам'яттю, яку програмують на етапі виробництва. Сучасні мікроконтролери дозволяють програмувати EEPROM-пам'ять безпосередньо на платі, що значно спрощує процес розробки. Такий підхід називається ISP (In-System Programming) і дозволяє оновлювати прошивку без демонтажу мікроконтролера.

Сучасні моделі мікроконтролерів мають вбудовану Flash-пам'ять, яка є аналогом EEPROM. Основна різниця між ними полягає в тому, що Flash-пам'ять стирається цілими блоками, тоді як EEPROM дозволяє стирати окремі осередки. Для збереження даних під час виконання програм використовується RAM — оперативна пам'ять. Вона має необмежену кількість циклів читання та запису, але її вміст втрачається при вимкненні живлення.

Перші моделі мікроконтролерів були розроблені для роботи при напрузі живлення 5В. Проте, з розвитком технологій і збільшенням кількості пристроїв, що працюють на автономних джерелах живлення, все більше сучасних мікроконтролерів функціонують при нижчій напрузі, такій як 3,3В або навіть менше. Це дозволяє зменшити енергоспоживання та подовжити час роботи від батарей [26].

Багато мікроконтролерів також оснащені спеціальними функціями для управління енергоспоживанням. Вони можуть переходити в режими зниженого енергоспоживання, такі як режим холостого ходу або режим повної зупинки. У режимі холостого ходу процесорне ядро припиняє свою роботу, але залишаються активними синхронізаційний генератор і сторожовий таймер. Регістри зберігають свої значення, а вихід з цього режиму можливий через переривання або сигнал скидання. В холостому режимі споживання потужності може складати близько 30% від номінальної.Режим повної зупинки передбачає повне припинення роботи всередині кристала мікроконтролера, при цьому він споживає мінімальний струм, який може становити всього 1 мкА або менше. Вихід з цього режиму можливий лише після отримання сигналу скидання або зовнішнього переривання. Такі функції управління живленням дозволяють використовувати мікроконтролери у пристроях з дуже низьким споживанням енергії, що є критично важливим для автономних та мобільних пристроїв.

**2.4 Загальні характеристики архітектури PIC**

Мікроконтролери сімейства PIC від компанії Microchip Technology є одними з найпопулярніших у світі завдяки своїй простоті, енергоефективності та універсальності. Основою їхньої архітектури є RISC (Reduced Instruction Set Computer), що передбачає використання обмеженого набору інструкцій для прискорення виконання програм. Це дозволяє виконувати більшість команд за один такт процесора, що зменшує затримки і підвищує швидкість роботи.

PIC мікроконтролери реалізують гарвардську архітектуру, яка відрізняється тим, що має окремі шини для програм і даних. Це дає можливість одночасно завантажувати команди і обробляти інформацію, що підвищує продуктивність мікроконтролера в порівнянні з традиційними архітектурами фон Неймана, де програми і дані зберігаються в одній і тій самій пам'яті [27].

Інша важлива характеристика PIC мікроконтролерів – це наявність великої кількості інтегрованих периферійних пристроїв, таких як таймери, аналогово-цифрові перетворювачі (ADC), інтерфейси зв'язку (UART, SPI, I2C), широтно-імпульсні модулятори (PWM), а також можливість підключення зовнішніх пристроїв через GPIO порти. Це дозволяє використовувати мікроконтролери PIC в різноманітних проєктах, від простих домашніх пристроїв до складних промислових систем.

Основні сімейства мікроконтролерів PIC

PIC мікроконтролери поділяються на кілька сімейств, кожне з яких адаптоване під різні завдання:

* PIC10/PIC12: Це найбільш компактні і прості 8-розрядні мікроконтролери, які використовуються для елементарних завдань управління в пристроях з обмеженими ресурсами. Завдяки малому енергоспоживанню та мінімальному набору функцій, вони є ідеальними для систем з автономним живленням та компактними розмірами.
* PIC16: Це найбільш популярне сімейство 8-розрядних мікроконтролерів, яке пропонує більше можливостей для управління периферійними пристроями та введення/виведення даних. Ці мікроконтролери часто використовуються в більш складних системах, де важливіша багатозадачність і можливість обробки даних з різних джерел.
* PIC18: Розширені 8-розрядні мікроконтролери з вищою продуктивністю та покращеними периферійними можливостями. PIC18 зазвичай використовуються в системах, де потрібно більше обчислювальної потужності і більше пам'яті для програм і даних. Також вони підходять для використання у реальних системах, де час виконання критично важливий.
* PIC24: Це вже 16-розрядні мікроконтролери, які дозволяють здійснювати більш складні розрахунки і керувати складнішими периферійними пристроями. Вони підходять для додатків, де потрібна висока точність обробки сигналів та управління великими системами.
* dsPIC: Це поєднання традиційного мікроконтролера та цифрового сигнального процесора (DSP). dsPIC мікроконтролери використовуються для цифрової обробки сигналів, що необхідно в складних математичних обчисленнях, обробці аудіо- та відеоінформації, а також у системах керування електродвигунами [28].

Кожне з цих сімейств має свої особливості та специфікації, що дозволяє розробникам вибирати потрібний мікроконтролер відповідно до вимог проєкту.

Переваги і сфери застосування мікроконтролерів PIC

Мікроконтролери PIC мають ряд важливих переваг, які роблять їх популярними серед розробників електроніки:

1. Простота використання: Завдяки зрозумілому набору інструкцій, доступним інструментам розробки та великій кількості документації, PIC мікроконтролери є чудовим вибором як для початківців, так і для досвідчених інженерів.
2. Гнучкість: Різноманітність сімейств дозволяє вибрати оптимальне рішення для конкретного завдання, від простих сенсорних пристроїв до складних промислових контролерів.
3. Енергозбереження: Мікроконтролери PIC можуть працювати в різних режимах енергоспоживання, що дозволяє оптимізувати роботу системи з автономним живленням. Особливості режимів "холостого ходу" та "зупинки" дають змогу значно зменшити споживання енергії під час простою системи.
4. Надійність: Мікроконтролери PIC відомі своєю стабільністю та довговічністю в роботі, що є важливим фактором для критично важливих систем, які працюють безперервно.

Завдяки цим перевагам, мікроконтролери PIC знайшли широке застосування в різних галузях:

* Автомобільна електроніка: Використовуються для управління різними системами в автомобілях, такими як системи безпеки, клімат-контроль, освітлення та управління двигунами.
* Промислова автоматизація: Використовуються в системах управління виробничими процесами, де важлива точність, надійність та адаптивність.
* Побутова електроніка: PIC мікроконтролери часто зустрічаються в таких пристроях, як пральні машини, мікрохвильові печі, холодильники, телевізори та інша техніка.
* Медичне обладнання: Завдяки своїй точності та надійності, ці мікроконтролери використовуються в пристроях для медичної діагностики, моніторингу життєвих показників пацієнтів та інших медичних системах.
* Телекомунікаційне обладнання: Використовуються в модемах, маршрутизаторах та інших пристроях зв'язку, де важливі швидкість і ефективність передачі даних.

Таким чином, мікроконтролери PIC є універсальним рішенням для широкого кола задач, що потребують ефективного, надійного та енергоефективного управління різними процесами в електронних системах.

**2.4.1 Мікроконтролери сімейства AVR**

Мікроконтролери AVR були створені у дослідницькому центрі компанії Atmel у Норвегії. Їх розробила група інженерів, серед яких виділяються Alf Bogen та Vergard Wollan. Саме перші літери їхніх імен, а також посилання на архітектуру RISC, сформували назву "AVR". Перший мікроконтролер цього типу, модель AT90S1200, з'явився на ринку в період між 1996 та 1997 роками.

Завдяки збалансованому співвідношенню «ціна – швидкодія – енергоспоживання», AVR мікроконтролери швидко стали популярними та здобули репутацію одного з найкращих рішень серед 8-бітних контролерів з архітектурою RISC. За свою коротку історію ці мікроконтролери отримали значне визнання на ринку. Їхній продаж зростає настільки стрімко, що кожного року обсяг подвоюється. За даними огляду Atmel Product Selection Guide 2015 року, загалом було виготовлено понад 7 мільярдів AVR мікроконтролерів, що свідчить про їхній великий успіх та популярність у різних галузях [29].

Галузі застосування

AVR мікроконтролери знаходять застосування у найрізноманітніших сферах – від простих пристроїв до складних систем збору та обробки інформації. Вони використовуються як у побутових пристроях, так і в промислових системах управління. Завдяки своїй надійності та енергоефективності, ці мікроконтролери стали основою для багатьох інноваційних рішень у світі електроніки.

Перший офіційний каталог мікроконтролерів AVR від компанії Atmel вийшов у 1997 році. У нього увійшли пристрої першого сімейства під назвою "Classic". Однак уже через два роки, у 1999 році, другий випуск каталогу представив нові сімейства мікроконтролерів: Classic, Mega і Tiny, що додатково розширило спектр можливостей цих пристроїв.

Варто зазначити, що компанія продовжувала розвивати та вдосконалювати свою лінійку мікроконтролерів. У 2008 році вона запустила нову серію 8-бітних мікроконтролерів XMega, яка швидко завоювала популярність на ринку.

Загальні характеристики AVR мікроконтролерів

На сьогоднішній день компанія Atmel виробляє різні види AVR мікроконтролерів, включаючи 8, 16 та 32-бітні моделі. Загалом на ринку представлено 294 різних типи цих мікроконтролерів, кожен з яких має низку спільних рис:

* Архітектура RISC з Гарвардською структурою: Мікроконтролери AVR використовують Гарвардську архітектуру, що передбачає розділення шин даних і адресних просторів пам'яті програм та даних. Це забезпечує ефективне виконання операцій та збільшення швидкодії.
* Вбудована пам'ять: Мікроконтролери мають вбудовану Flash-пам'ять для зберігання програм, статичну пам'ять SRAM для обробки даних і енергонезалежну EEPROM пам'ять для зберігання конфігурацій або змінних. Така архітектура дозволяє використовувати мову програмування високого рівня для розробки програм, що значно спрощує процес розробки.
* Продуктивність: Продуктивність AVR мікроконтролерів досягає 1 MIPS (мільйон інструкцій за секунду) на частоті 1 МГц, що є досить високим показником для 8-бітних мікроконтролерів.
* Регістри загального призначення: У кожному мікроконтролері AVR є 32 регістри загального призначення, які використовуються для зберігання даних та виконання арифметичних і логічних операцій. Це сприяє підвищенню продуктивності та ефективності програм, дозволяючи виконувати операції швидше та оптимальніше.
* Конвеєр обробки команд: AVR мікроконтролери використовують 1-рівневий конвеєр, який дозволяє одночасно виконувати поточну інструкцію та вибирати наступну. Це значно прискорює виконання програмних алгоритмів.
* Можливість програмування: AVR мікроконтролери підтримують як зовнішнє, так і внутрішньосхемне програмування, що робить їх зручними для розробників та інженерів.
* Сектори пам'яті: Пам'ять програм AVR мікроконтролерів розподілена на два сектори – для прикладної програми та для програми завантажувача. Це дозволяє виконувати функцію самопрограмування, яка є важливою для багатьох сучасних систем.
* Фюзи (Fuses): AVR мікроконтролери мають спеціальні засоби для налаштування своєї конфігурації, які називаються фюзи. Це дає можливість гнучко змінювати поведінку мікроконтролера відповідно до вимог системи.

Класифікація сімейств AVR

Сімейства мікроконтролерів AVR мають різні призначення та характеристиками, що дозволяє обрати найбільш відповідний варіант для кожного конкретного застосування. Сьогодні існують такі основні сімейства:

1. tinyAVR – Це найменші за розмірами та функціональністю мікроконтролери, що пропонують 38 різних моделей з 8-бітною архітектурою.
2. megaAVR – Найбільш розповсюджене сімейство, що містить 105 моделей 8-бітних мікроконтролерів, які відрізняються великою кількістю периферійних модулів.
3. AVR XMEGA – Це сімейство включає 46 моделей 8/16-бітних мікроконтролерів, призначених для більш складних завдань, що потребують вищої швидкодії та багатофункціональності.
4. Battery Management MCUs – 8-бітні мікроконтролери для керування живленням, що налічують 5 моделей.
5. Automotive AVR MCUs – Мікроконтролери для автомобільної промисловості, які пропонують 38 різних моделей.
6. 32-bit AVR UC3 – Сімейство 32-бітних мікроконтролерів, яке включає 62 різні моделі.

Окрім стандартних рішень, AVR ядро може використовуватись в комбінації з FPGA матрицями для створення ASIC платформ або систем на кристалі (SoC). Це дозволяє інтегрувати AVR ядра у більш складні та спеціалізовані рішення, які відповідають потребам сучасної електронної промисловості.

**2.5 Програмування мікроконтролерів. Вибір інструментів розробки**

Програмування мікроконтролерів є одним із ключових етапів розробки будь-якої вбудованої системи. На цьому етапі інженер створює програмний код, що визначає, як мікроконтролер буде взаємодіяти з периферійними пристроями, датчиками та іншими компонентами системи. Саме через програмування мікроконтролери отримують інструкції для виконання складних алгоритмів, таких як обробка сигналів, керування електромоторами, обробка даних із сенсорів або виконання певних завдань у реальному часі. Мікроконтролер виконує всі ці операції на рівні апаратних команд, що дозволяє досягти високої швидкодії та ефективності при обмеженому споживанні енергії [30].

Програмування дає можливість не лише автоматизувати обчислювальні процеси, але й оптимізувати взаємодію між апаратними і програмними компонентами системи. Завдяки гнучкості у створенні програмного забезпечення, мікроконтролери можуть бути налаштовані для роботи в різних умовах і з різними вимогами. Інженери можуть використовувати мікроконтролери для побудови як простих, так і складних систем – від звичайних контролерів для домашньої техніки до високоточних промислових автоматизованих систем.

Важливим аспектом програмування мікроконтролерів є вибір правильного середовища розробки. Це середовище включає в себе інструменти для написання, налагодження та завантаження програмного коду в мікроконтролер. Вибір такого середовища залежить від кількох факторів, таких як тип архітектури мікроконтролера, складність завдань, функціональні можливості середовища та специфічні потреби проекту.

Кожен виробник мікроконтролерів пропонує свої інструменти для розробки програмного забезпечення. Наприклад, для програмування мікроконтролерів AVR використовується середовище розробки Atmel Studio, яке надає всі необхідні інструменти для компіляції, налагодження та завантаження програмного коду. Аналогічно, мікроконтролери PIC програмуються за допомогою середовища MPLAB X, розробленого компанією Microchip, що дозволяє працювати з різними моделями контролерів цього бренду.

**2.5.1MikroC Pro for PIC**

Одним із популярних середовищ для програмування мікроконтролерів є MikroC PRO for PIC, яке спеціально створене для роботи з мікроконтролерами PIC від компанії Microchip. Це середовище пропонує зручний інтерфейс для написання коду, налагодження і компіляції програм для мікроконтролерів. MikroC підтримує велику кількість бібліотек, які дозволяють працювати з периферійними пристроями та протоколами зв’язку, такими як I2C, SPI, UART, та іншими.

MikroC Pro for PIC – це професійне середовище розробки (IDE) для програмування мікроконтролерів серії PIC, розроблене компанією MikroElektronika. Воно спеціально створене для мікроконтролерів, виготовлених компанією Microchip Technology. Це програмне забезпечення широко використовується як новачками, так і професіоналами в галузі вбудованих систем, оскільки воно має інтуїтивно зрозумілий інтерфейс та потужні функціональні можливості.

Основні особливості MikroC Pro for PIC:

1. Зручність у використанні:
   * MikroC Pro має простий інтерфейс, який дозволяє легко створювати, редагувати та компілювати код. Новачки можуть швидко освоїти базові операції без необхідності поглибленого вивчення архітектури мікроконтролерів.
   * IDE підтримує автодоповнення, що прискорює написання коду.
2. Велика бібліотека готових функцій:
   * MikroC Pro має широкий набір готових бібліотек, які підтримують різні периферійні пристрої, такі як LCD дисплеї, датчики температури, клавіатури, порти UART, SPI, I2C тощо. Це суттєво скорочує час розробки, оскільки більшість компонентів можна використовувати з мінімальною кількістю додаткового коду.
3. Підтримка великої кількості мікроконтролерів PIC:
   * MikroC Pro підтримує понад 1000 моделей мікроконтролерів серії PIC, включаючи 8-бітні, 16-бітні та деякі 32-бітні моделі. Це робить його універсальним інструментом для розробників, які працюють з різними проектами.
4. Вбудовані інструменти налагодження:
   * IDE включає емулятор, який дозволяє перевіряти правильність роботи програмного забезпечення, що значно полегшує налагодження. Він також підтримує відлагоджування реального обладнання через програматори та дебагери.
   * Ви можете аналізувати стан регістрів, змінних та інші критичні аспекти виконання програми.
5. Підтримка графічного інтерфейсу для проєктування:
   * У MikroC Pro є інструмент Visual TFT, який дозволяє розробляти графічний інтерфейс для дисплеїв із сенсорним екраном, використовуючи простий інструмент перетягування (drag-and-drop). Це полегшує створення графічних елементів інтерфейсу для ваших проєктів.
6. Оптимізатор коду:
   * MikroC Pro має вбудований компілятор, який оптимізує код під час компіляції, що допомагає ефективно використовувати пам'ять і ресурси мікроконтролера. Це особливо важливо для мікроконтролерів із обмеженими ресурсами, наприклад, малою кількістю пам'яті.
7. Підтримка роботи з програматорами:
   * MikroC Pro сумісний із різними програматорами і дебагерами, включаючи оригінальні програматори від компанії MikroElektronika – mikroProg, а також програматори від Microchip, наприклад, PICkit.
8. Велика кількість прикладів:
   * IDE містить велику бібліотеку прикладів коду для різних завдань. Це значно полегшує процес навчання та розробки нових проектів.
9. Доступ до підтримки та спільноти:
   * Компанія MikroElektronika активно підтримує своє середовище, регулярно випускаючи оновлення, а також надає документацію та форуми для обговорень. Також є багато навчальних матеріалів, таких як посібники та відеоуроки.

Переваги використання MikroC Pro for PIC:

* Простота: IDE дуже зручне та легке в освоєнні, що робить його підходящим для швидкого старту навіть для новачків.
* Широкі можливості: Завдяки великій кількості готових бібліотек і підтримці різноманітних мікроконтролерів, розробники можуть реалізовувати складні проекти з мінімальними витратами часу на написання базових функцій.
* Швидке налагодження: Вбудовані інструменти для відлагодження та тестування дозволяють швидко знаходити та виправляти помилки в коді.

Недоліки:

* Залежність від середовища: MikroC Pro специфічний для мікроконтролерів PIC, тому для інших архітектур потрібні інші середовища розробки.

MikroC Pro for PIC – це потужне середовище розробки, яке надає розробникам широкий спектр можливостей для програмування мікроконтролерів PIC. Завдяки простоті у використанні, великим наборам бібліотек та можливостям для налагодження, це IDE стає надійним інструментом для розробників як простих, так і складних вбудованих систем.

**2.5.2 MPLAB X IDE**

MPLAB X IDE — офіційна інтегрована середа розробки від компанії Microchip для роботи з мікроконтролерами PIC, dsPIC та AVR. Це потужний інструмент для написання, компіляції та налагодження програм. У порівнянні з MikroC, MPLAB X пропонує більш глибоку інтеграцію з апаратними засобами програмування та розширені можливості налагодження.

MPLAB X дозволяє працювати з такими мовами програмування, як C, C++, і навіть Assembler для високоефективного коду. Це середовище широко використовується в професійних проектах завдяки можливостям детального контролю над апаратними ресурсами мікроконтролера.

MPLAB X IDE – це інтегроване середовище розробки (IDE), створене компанією Microchip Technology, яке призначене для програмування мікроконтролерів серій PIC, dsPIC, AVR, та інших продуктів Microchip. Це потужний інструмент, що забезпечує розробникам всі необхідні засоби для проектування, написання, компіляції та налагодження програмного забезпечення для вбудованих систем.

Основні особливості MPLAB X IDE:

1. Крос-платформність:
   * MPLAB X IDE є крос-платформним, що означає, що його можна запускати на різних операційних системах, таких як Windows, macOS і Linux. Це забезпечує гнучкість для розробників, незалежно від платформи, яку вони використовують.
2. Підтримка різних мікроконтролерів:
   * IDE підтримує широкий спектр мікроконтролерів Microchip, включаючи 8-бітні, 16-бітні та 32-бітні моделі. Це робить його універсальним інструментом для розробників, які працюють з різними архітектурами.
3. Вбудовані інструменти для налагодження:
   * MPLAB X надає потужні інструменти для налагодження, такі як MPLAB REAL ICE та MPLAB PICkit, які дозволяють розробникам тестувати та відлагоджувати код на реальному обладнанні. Це забезпечує можливість детального аналізу роботи програмного забезпечення та швидкого виправлення помилок.
4. Можливість використання різних компіляторів:
   * MPLAB X підтримує різні компілятори, зокрема MPLAB XC8, MPLAB XC16 та MPLAB XC32. Це дає змогу розробникам вибирати компілятор, найбільш відповідний для їх проекту, а також забезпечує максимальну гнучкість у написанні коду.
5. Гнучка архітектура проектів:
   * IDE використовує проектну архітектуру, що дозволяє організовувати код у зрозумілій структурі. Це робить управління великими проектами простішим і зручнішим.
6. Розширювальність:
   * MPLAB X має можливість інтеграції з різними плагінами та розширеннями, що дозволяє налаштовувати середовище відповідно до потреб розробника. Це може включати додаткові бібліотеки, інструменти для графічного інтерфейсу та інше.
7. Документація та підтримка:
   * Microchip надає всебічну документацію, що включає довідники, посібники, відеоуроки та приклади коду. Це полегшує навчання новачків і допомагає досвідченим розробникам швидше знаходити необхідну інформацію.
8. Інтеграція з іншими інструментами Microchip:
   * MPLAB X IDE безшовно інтегрується з іншими інструментами Microchip, такими як MPLAB Harmony для розробки на 32-бітних мікроконтролерах, та MPLAB Code Configurator для графічного налаштування периферійних пристроїв.

Переваги використання MPLAB X IDE:

* Багатофункціональність: Підтримка різних мікроконтролерів і компіляторів робить його універсальним інструментом для різноманітних проектів.
* Потужні інструменти для налагодження: Інтегровані можливості для тестування та налагодження на реальному обладнанні значно полегшують процес розробки.
* Сильна документація та підтримка: Доступ до навчальних ресурсів та активна спільнота користувачів дозволяють швидко вирішувати проблеми.

Недоліки:

* Складність для новачків: Хоча MPLAB X надає потужні можливості, новачкам може знадобитися час для освоєння всіх функцій та можливостей IDE.
* Вимоги до ресурсів: MPLAB X може бути досить вимогливим до системних ресурсів, що може бути проблемою для старіших комп'ютерів.

MPLAB X IDE є одним із найпопулярніших середовищ розробки для програмування мікроконтролерів Microchip. Завдяки своїй потужності, гнучкості та великій кількості інтегрованих інструментів, це ідеальне рішення для розробників, які прагнуть створювати високоякісні вбудовані системи. Незважаючи на деякі недоліки, його функціональність і підтримка роблять MPLAB X важливим інструментом у розробці вбудованих проектів.

**2.5.3 Atmel Studio**

Atmel Studio – це інтегроване середовище розробки (IDE), яке було розроблено компанією Microchip Technology (після придбання Atmel) для програмування мікроконтролерів серій AVR і SAM. Це потужний інструмент, що забезпечує розробникам усі необхідні засоби для створення, компіляції, налагодження та тестування програмного забезпечення для вбудованих систем, що базуються на мікроконтролерах Atmel.

Основні особливості Atmel Studio:

1. Підтримка різних архітектур:
   * Atmel Studio підтримує широкий спектр мікроконтролерів, включаючи 8-бітні AVR, 32-бітні ARM Cortex-M та інші. Це робить IDE універсальним інструментом для розробників, які працюють з різними типами мікроконтролерів.
2. Інтеграція з компіляторами:
   * IDE постачається з вбудованими компіляторами AVR GCC і ARM GCC, що дозволяє розробникам легко компілювати свій код без необхідності встановлення додаткових інструментів. Це спрощує процес налаштування середовища.
3. Потужні інструменти для налагодження:
   * Atmel Studio надає потужні засоби для налагодження, такі як Atmel-ICE та JTAGICE, які дозволяють проводити детальне налагодження програм на реальному обладнанні. Це дозволяє виявляти і виправляти помилки на етапі тестування.
4. Графічний інтерфейс:
   * IDE має зручний і інтуїтивно зрозумілий графічний інтерфейс, що полегшує навігацію між проектами, налаштуваннями та функціями. Це робить Atmel Studio доступним навіть для новачків.
5. Бібліотеки і приклади коду:
   * Atmel Studio містить великий набір готових бібліотек і прикладів коду, які можуть допомогти розробникам у швидкому старті проекту. Це спрощує інтеграцію з периферійними пристроями і модулями.
6. Гнучкість у налаштуваннях:
   * IDE дозволяє розробникам налаштовувати середовище відповідно до своїх потреб, включаючи можливість створення власних шаблонів проектів, конфігурацій та макросів.
7. Аналіз продуктивності:
   * Atmel Studio має інструменти для аналізу продуктивності, що дозволяють розробникам вивчати споживання ресурсів та оптимізувати код для кращої ефективності.

Переваги використання Atmel Studio:

* Широкі можливості: Підтримка різних мікроконтролерів та компіляторів робить його універсальним інструментом для розробників.
* Зручний інтерфейс: Інтуїтивно зрозумілий графічний інтерфейс полегшує роботу з IDE, особливо для новачків.
* Доступність бібліотек: Наявність готових бібліотек і прикладів коду значно скорочує час на розробку проектів.

Недоліки:

* Специфічність для продуктів Atmel: Atmel Studio в основному призначено для роботи з мікроконтролерами Atmel, що може бути обмеженням для розробників, які працюють з іншими брендами.
* Вимоги до системних ресурсів: Як і багато інших IDE, Atmel Studio може бути вимогливим до ресурсів комп'ютера, особливо при роботі з великими проектами.

Atmel Studio є потужним і універсальним середовищем розробки, яке підходить для програмування мікроконтролерів AVR та ARM. Завдяки широкому набору інструментів, інтуїтивно зрозумілому інтерфейсу та підтримці різних архітектур, це ідеальне рішення для розробників, які прагнуть створити високоякісні вбудовані системи. Хоча Atmel Studio має свої недоліки, його переваги роблять його важливим інструментом у розробці вбудованих проектів.

* + 1. **Arduino IDE**

Arduino IDE – це інтегроване середовище розробки, спеціально розроблене для програмування мікроконтролерів на основі платформи Arduino. Це популярний інструмент серед розробників, які працюють з вбудованими системами, оскільки він простий у використанні і підходить як для новачків, так і для досвідчених програмістів.

Основні характеристики sPlan:

1. Простота використання:
   * Arduino IDE має інтуїтивно зрозумілий інтерфейс, що робить його доступним для користувачів без попереднього досвіду програмування. Інструмент підтримує основні функції редагування коду, такі як підсвітка синтаксису, автоматичне завершення коду та валідація.
2. Підтримка різних плат Arduino:
   * Arduino IDE підтримує широкий спектр плат Arduino, таких як Arduino Uno, Mega, Nano, Leonardo та інші. Це дозволяє розробникам легко переходити між різними платформами та адаптувати свій код.
3. Вбудовані бібліотеки:
   * IDE містить безліч готових бібліотек, що спрощує інтеграцію з різними датчиками, модулями і периферійними пристроями. Це дозволяє розробникам швидше реалізовувати свої проекти, не витрачаючи час на написання базових функцій.
4. Спільнота і ресурси:
   * Arduino має велику спільноту користувачів, що забезпечує безліч ресурсів, таких як форуми, документація, навчальні матеріали та приклади проектів. Це допомагає новачкам швидше вчитися і отримувати допомогу в разі потреби.
5. Крос-платформеність:
   * Arduino IDE доступно для різних операційних систем, включаючи Windows, macOS і Linux, що робить його доступним для широкого кола користувачів.
6. Програмування мовою C/C++:
   * Arduino IDE використовує мови програмування C і C++, що дозволяє розробникам створювати складні алгоритми та управляти функціями мікроконтролера. Для зручності використання IDE надає спрощений синтаксис, що дозволяє новачкам легко зрозуміти основи програмування.

Переваги використання Arduino IDE:

* Доступність: Легко завантажити і встановити, що робить його ідеальним вибором для початківців.
* Готові рішення: Багато бібліотек і прикладів дозволяють швидко запускати проекти.
* Активна спільнота: Широке коло користувачів забезпечує постійну підтримку і розвиток платформи.

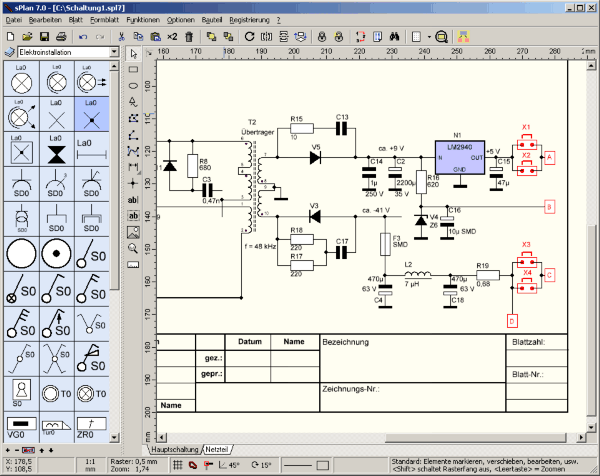
Недоліки:

* Обмеження продуктивності: Arduino IDE не завжди підходить для великих і складних проектів через обмеження в ресурсах.
* Відсутність розширених функцій: На відміну від більш професійних IDE, таких як MPLAB або Atmel Studio, Arduino IDE може бути обмеженим у функціональності та можливостях налагодження.

Arduino IDE – це потужний інструмент для розробки програмного забезпечення для мікроконтролерів Arduino. Завдяки своїй простоті, доступності та широкій спільноті, це ідеальне середовище для початківців, які прагнуть вивчити основи програмування вбудованих систем. Хоча Arduino IDE має свої обмеження, його переваги роблять його важливим інструментом для багатьох проектів у сфері електроніки та робототехніки.

Arduino IDE призначене для роботи з мікроконтролерами AVR серії ATmega, воно також підтримує інші архітектури через додаткові платформи та пакети. Arduino IDE популярне завдяки простоті використання, зручному інтерфейсу і вбудованим бібліотекам. Воно підходить для швидкого прототипування і навчальних проектів, оскільки пропонує можливість роботи з простими конструкціями на C/C++. Однак, для складніших завдань варто використовувати більш професійні середовища розробки, такі як MPLAB X або Atmel Studio.

**2.6 sPlan**

sPlan — це потужна програма для проектування електричних схем, яка користується популярністю серед електроніків, інженерів і студентів технічних спеціальностей. Вона дозволяє створювати, редагувати та документувати електричні схеми з високою точністю та ефективністю. sPlan відрізняється інтуїтивно зрозумілим інтерфейсом і багатим набором функцій, що робить його ідеальним інструментом для різних потреб у проектуванні електроніки.

**Рис. 2.3** Зразок схеми в програмі sPlan

Основні характеристики sPlan

1. Інтуїтивно зрозумілий інтерфейс:
   * sPlan має простий і зручний інтерфейс, що дозволяє навіть новачкам швидко орієнтуватися в програмі. Основні інструменти для малювання схем доступні на панелі інструментів, що спрощує процес проектування. Візуально привабливий інтерфейс сприяє зосередженню на створенні схем, а не на навігації по складних меню.
2. Багатий набір бібліотек компонентів:
   * Програма містить велику кількість бібліотек стандартних електронних компонентів, таких як резистори, конденсатори, діоди, транзистори, мікроконтролери тощо. Користувачі можуть також додавати свої власні компоненти або редагувати існуючі для більш точного відповідності їхнім проектам. Це забезпечує можливість швидкого доступу до необхідних елементів і спрощує процес створення схем.
3. Функції редагування схем:
   * sPlan надає широкий спектр функцій для редагування схем, включаючи копіювання, вставку, видалення, переміщення та групування компонентів. Ці інструменти дозволяють гнучко управляти схемами, спрощуючи редагування вже існуючих проектів. Також є можливість використання багаторівневих груп для організації елементів, що покращує структуру проекту.
4. Можливість експорту та імпорту:
   * Користувачі можуть експортувати свої схеми в різні формати, включаючи графічні формати (таких як BMP, JPG, PNG) та PDF. Це робить схеми зручними для друку, публікації або подальшого використання в документації. Імпорт схем з інших програм також підтримується, що дозволяє інтегрувати раніше створені проекти в нові роботи.
5. Підтримка багатошарових схем:
   * Програма дозволяє створювати багатошарові схеми, що особливо корисно для проектів з великою кількістю компонентів. Користувачі можуть організувати свої схеми на різних шарах, що покращує сприйняття та структурованість проекту. Це також допомагає уникнути плутанини при роботі з складними електричними з’єднаннями.
6. Діагностика та верифікація:
   * sPlan надає функції для перевірки коректності з’єднань, що дозволяє виявити помилки в схемах до їх реалізації. Ця можливість допомагає запобігти потенційним проблемам, заощаджуючи час і зусилля на етапі реалізації проекту.
7. Документація:
   * Програма автоматично створює документацію, що містить інформацію про використовувані компоненти, їх з’єднання та інші важливі дані, необхідні під час реалізації проекту. Це зручний функціонал, який спрощує підготовку матеріалів для презентацій або подачі проекту.

Переваги sPlan

* Простота у використанні: Завдяки інтуїтивно зрозумілому інтерфейсу та простим інструментам, навіть новачки можуть швидко навчитися працювати з програмою. Це робить sPlan доступним для широкого кола користувачів, незалежно від їх рівня підготовки.
* Великий вибір компонентів: Наявність великої бібліотеки електронних компонентів дозволяє швидко знаходити і додавати необхідні елементи до схеми, що прискорює процес проектування.
* Гнучкість та налаштування: Користувачі можуть адаптувати програму відповідно до своїх потреб, додаючи нові компоненти та налаштовуючи функції редагування, що робить sPlan гнучким інструментом для різних проектів.

Недоліки sPlan

* Обмежені можливості 3D-рендерингу: На відміну від деяких CAD-програм, sPlan не підтримує 3D-моделювання, що може бути обмеженням для проектів, що вимагають візуалізації в тривимірному просторі.
* Вартість: Хоча існує безкоштовна версія програми, повнофункціональна версія може вимагати фінансових витрат, що може бути перешкодою для деяких користувачів.

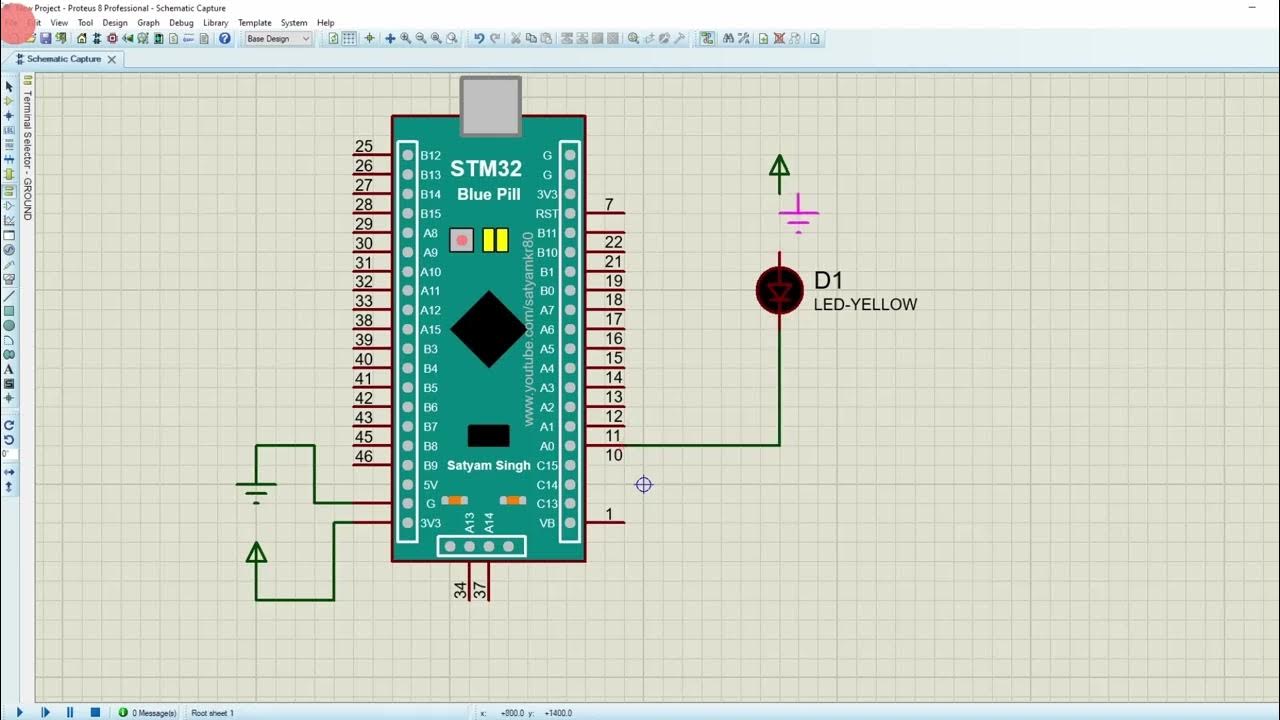
sPlan — це універсальний та ефективний інструмент для створення електричних схем, що підходить для широкого кола користувачів, від студентів до професіоналів. Його простота у використанні, багатий набір функцій та можливість експорту роблять його важливим інструментом у розробці електронних проектів. Незважаючи на деякі обмеження, sPlan залишається популярним вибором серед розробників, які шукають ефективний і надійний інструмент для створення схем. Сумарно, програма допомагає заощаджувати час, підвищує продуктивність та полегшує процес проектування, що робить її цінним ресурсом у світі електроніки.

**2.7 Proteus**

Proteus — це один з найбільш популярних програмних пакетів для проектування електронних схем і моделювання вбудованих систем, який активно використовується фахівцями у галузі електроніки, а також студентами та викладачами вищих навчальних закладів. Ця програма дозволяє не лише створювати схеми, а й симулювати їх роботу, що робить її важливим інструментом для тестування та верифікації проектів. Розглянемо детальніше функції, переваги та особливості використання Proteus.

Основні функції Proteus

1. Схемотехнічний редактор:
   * Proteus пропонує потужний редактор для створення електричних схем. Користувачі можуть легко додавати компоненти, такі як резистори, конденсатори, транзистори, мікроконтролери та багато інших елементів, завдяки зручному інтерфейсу drag-and-drop. Всі компоненти представлені у вигляді символів, що відповідають їхнім функціям.
2. Моделювання мікроконтролерів:
   * Одна з унікальних можливостей Proteus полягає в моделюванні програмного забезпечення для мікроконтролерів. Користувачі можуть написати код для мікроконтролерів на таких мовах, як C або Assembly, і протестувати його в середовищі симуляції. Це дозволяє виявляти помилки і оптимізувати програми без необхідності виготовлення фізичних прототипів.
3. Симуляція схем:
   * Proteus дозволяє користувачам проводити реальні симуляції електронних схем. Це включає в себе тестування сигналів, напруг і струмів у різних частинах схеми. Можливість спостерігати за динамікою роботи схеми в реальному часі надає користувачам уявлення про те, як система поводитиметься при різних умовах.
4. Підтримка численних компонентів:
   * У бібліотеці Proteus міститься тисячі компонентів, включаючи стандартні електронні елементи, мікроконтролери, аналогові та цифрові пристрої, що дозволяє розробникам працювати з різноманітними проектами. Користувачі також можуть створювати власні бібліотеки для специфічних потреб.
5. Графічна візуалізація:
   * Proteus надає можливості для візуалізації електронних схем та їх роботи. Користувачі можуть переглядати потоки даних, напругу і струм у реальному часі, що допомагає їм краще розуміти процеси, які відбуваються у схемі.
6. Вимірювання та аналіз:
   * У Proteus вбудовані інструменти для вимірювання різних параметрів схем. Користувачі можуть аналізувати графіки змінення напруги та струму, що дозволяє глибше розуміти електричні процеси.



**Рис. 2.4** Зразок схеми в програмі Proteus

Переваги Proteus

1. Ефективність у розробці:
   * Proteus дозволяє значно скоротити час на розробку та тестування електронних схем завдяки можливості моделювання та симуляції.
2. Інтуїтивно зрозумілий інтерфейс:
   * Зручний і зрозумілий графічний інтерфейс робить Proteus доступним для користувачів з різним рівнем підготовки. Новачки можуть швидко освоїти основи, а досвідчені фахівці знайдуть необхідні інструменти для складних проектів.
3. Гнучкість:
   * Proteus підтримує широкий спектр мікроконтролерів, що дозволяє використовувати його для різних проектів, від простих до складних систем.
4. Інтеграція з програмуванням:
   * Користувачі можуть безпосередньо впроваджувати свій код у симуляцію, що спрощує процес налагодження та тестування.

Недоліки Proteus

1. Вартість:
   * Proteus є комерційним продуктом, і для повнофункціонального використання необхідно придбати ліцензію. Це може бути перешкодою для студентів або малих підприємств.
2. Системні вимоги:
   * Для ефективної роботи з програмою потрібні досить потужні комп’ютерні ресурси, що може обмежити використання на старіших системах.

Proteus активно використовується в навчальних закладах для навчання студентів основам електроніки. Студенти можуть реалізувати свої проекти, перевіряти теоретичні знання на практиці та отримувати зворотний зв'язок про результати роботи. Це робить програму цінним інструментом для викладачів, які прагнуть навчити студентів проектувати та тестувати електронні схеми.

Proteus — це універсальне програмне забезпечення для проектування електроніки, яке поєднує в собі можливості схематичного редагування, симуляції мікроконтролерів та візуалізації. Його функціональність, інтуїтивно зрозумілий інтерфейс і широкий вибір компонентів роблять його потужним інструментом для професіоналів і студентів. Proteus сприяє розвитку навичок у проектуванні та тестуванні електронних систем, а також допомагає втілити в життя інноваційні ідеї в галузі електроніки.

**2.8 Мова програмування мікроконтролера**

Програмування мікроконтролерів є важливим етапом у розробці вбудованих систем, оскільки саме від вибору мови програмування залежить зручність, ефективність та продуктивність реалізації проекту. Існує кілька мов програмування, які застосовуються для написання програм для мікроконтролерів, і кожна з них має свої особливості, переваги та недоліки.

Основні мови програмування мікроконтролерів

Assembly (асемблер):

* + Асемблер є низькорівневою мовою програмування, яка безпосередньо відповідає машинному коду мікроконтролера. Програми, написані на асемблері, забезпечують високу продуктивність і оптимізацію, оскільки дають змогу програмісту точно контролювати ресурси процесора.
  + Переваги:
    - Максимальна продуктивність та контроль над апаратними ресурсами.
    - Низький обсяг пам’яті, що використовується.
  + Недоліки:
    - Складність у написанні та налагодженні коду.
    - Потреба у глибокому розумінні архітектури мікроконтролера.
    - Довгий час розробки.

C:

* + Мова C є однією з найпопулярніших мов програмування для мікроконтролерів. Вона забезпечує хорошу продуктивність, високу переносимість та зручність у використанні. Мова C дозволяє програмісту писати код, який можна легко адаптувати для різних платформ.
  + Переваги:
    - Збалансованість між продуктивністю та простотою написання коду.
    - Широка підтримка бібліотек та фреймворків.
    - Легкість у налагодженні та тестуванні.
  + Недоліки:
    - Можливі проблеми з управлінням пам’яттю (особливо в embedded системах).
    - Досить обмежений контроль над апаратним забезпеченням у порівнянні з асемблером.

C++:

* + C++ є розширенням мови C, яке підтримує об’єктно-орієнтоване програмування (ООП). Це дозволяє використовувати класи, об’єкти та інші концепції ООП для організації коду, що може бути корисним у великих проектах.
  + Переваги:
    - Структурованість коду та можливість повторного використання.
    - Багатий набір функцій і можливостей для створення складних проектів.
  + Недоліки:
    - Більш складний синтаксис та вища вартість ресурсів у порівнянні з C.
    - Вимоги до компілятора, який повинен підтримувати OOP.

Python:

* + Хоча Python не є типовою мовою для програмування мікроконтролерів, його застосування стає все більш популярним, особливо з розвитком платформ, таких як MicroPython. Python забезпечує простоту написання коду, що робить його ідеальним для навчання та прототипування.
  + Переваги:
    - Легкість в освоєнні та використанні.
    - Швидкість розробки завдяки простому синтаксису.
  + Недоліки:
    - Менша продуктивність у порівнянні з C або асемблером.
    - Обмежена підтримка вбудованих функцій та бібліотек.

Вибір мови програмування для мікроконтролера залежить від кількох факторів:

* Тип проекту: Якщо потрібно реалізувати критично важливі для продуктивності завдання, кращим вибором буде асемблер або C. Для проектів середньої складності, де важлива швидкість розробки, доцільно використовувати C або C++.
* Кількість доступної пам’яті та ресурсів: В обмежених умовах ресурсів (пам’ять, обчислювальна потужність) мови низького рівня можуть бути кращими.
* Досвід команди: Якщо команда має досвід роботи з певною мовою, її доцільно використовувати, щоб прискорити процес розробки.
* Наявність бібліотек та інструментів: Якщо для конкретної мови є доступ до широкого спектру бібліотек та інструментів, це може спростити процес розробки.

Мова програмування є критично важливим аспектом при розробці проектів на основі мікроконтролерів. Правильний вибір мови може суттєво вплинути на продуктивність, зручність розробки та можливості реалізації проекту. Важливо враховувати специфіку проекту, наявні ресурси та досвід команди, щоб вибрати оптимальну мову для програмування мікроконтролера.

**ВИСНОВОК ДО РОЗДІЛУ 2**

У другому розділі магістерської роботи було здійснено детальний аналіз мікроконтролерів, їх архітектур, а також процесу програмування, що є невід'ємною частиною створення сучасних вбудованих систем. Мікроконтролери виконують ключову роль у реалізації різноманітних функцій, від простих побутових пристроїв до складних промислових систем. У ході дослідження було виявлено, що мікроконтролери, такі як AVR, PIC та ARM, пропонують різні переваги, що визначаються їхньою архітектурою та функціональними можливостями.

Програмування мікроконтролерів є складним процесом, що вимагає знань про мови програмування, специфіку архітектур, а також про середовища розробки, в яких реалізуються проекти. Було прийнято рішення використовувати мікроконтролер PIC, оскільки він поєднує в собі високу продуктивність, енергоефективність та достатню простоту для розробки. PIC мікроконтролери забезпечують широкий спектр функцій, що робить їх ідеальним вибором для реалізації різноманітних завдань у проекті.

У якості мови програмування обрано C, що є однією з найпоширеніших мов для розробки програмного забезпечення для мікроконтролерів. Використання C дозволяє реалізувати складні алгоритми з високою ефективністю, а також забезпечує зручність у написанні та налагодженні коду. Вона надає доступ до низькорівневих функцій, що є важливими для управління апаратними засобами.

Крім того, в процесі розробки були використані потужні програмні інструменти для підтримки кожного етапу. Серед них Proteus, який надав можливість моделювання та симуляції електронних схем, що дозволило знизити ризики при розробці та тестуванні. sPlan використовувався для проектування схем. Ці інструменти дозволили ефективно реалізувати проект, забезпечуючи точність і якість на всіх етапах розробки.

Отже, цей розділ окреслює основні аспекти, які були враховані при розробці проекту, і підтверджує правильність рішень щодо використання мікроконтролера PIC, мови C та відповідних програмних засобів. Вони не лише відповідають вимогам проекту, але й відкривають можливості для подальшого розвитку та впровадження інновацій у сфері вбудованих систем.

**РОЗДІЛ 3.**

**РОЗРОБКА ПРИСТРОЮ ДЛЯ ПЕРЕВІРКИ БЖ ПК**

**3.1 Розробка принципової електричної схеми**

Принципова електрична схема є основою для розробки пристрою, що перевіряє блоки живлення (БЖ) ПК. Вона відображає взаємозв'язки між компонентами, їхні функції та взаємодію, що дозволяє зрозуміти, як пристрій буде працювати в цілому. У цьому підрозділі буде детально описано, як розроблялась принципова електрична схема, які елементи були обрані, та які особливості потрібно врахувати.

Основними компонентами, які будуть використані в схемі, є:

1. Мікроконтролер: Вибрано мікроконтролер серії PIC, оскільки він має оптимальні характеристики для проекту, включаючи достатню кількість входів/виходів та підтримку програмування на мові C. Він буде відповідати за обробку даних і управління іншими компонентами системи.
2. Датчики: Для вимірювання напруги та струму використовуватимуться аналогові датчики. Вони будуть підключені до мікроконтролера для збору інформації про роботу БЖ.
3. Дисплей: Для виведення результатів вимірювань буде використовуватися ЖК-дисплей, що дозволяє зручно відображати дані в режимі реального часу.
4. Кнопки управління: Додаткові кнопки дозволять користувачу управляти пристроєм, запускати вимірювання або скидання налаштувань.
5. Живлення: Передбачено використання стабільного джерела живлення для забезпечення роботи всіх компонентів.

На етапі розробки принципової електричної схеми було важливо правильно спроектувати зв'язки між компонентами. Схема включає:

* Підключення датчиків напруги і струму до аналогових входів мікроконтролера.
* Підключення ЖК-дисплея до цифрових виходів, що забезпечить виведення даних на екран.
* Наявність кнопок управління, які підключені до вільних входів, дозволяючи користувачу взаємодіяти з пристроєм.
* Організацію схеми живлення, що забезпечить стабільну роботу всіх елементів.

Для візуалізації та перевірки електричної схеми використовувались програми, такі як Proteus та sPlan. Ці інструменти дозволяють створювати та моделювати електричні схеми, що допомагає виявити можливі помилки на етапі проектування.

У Proteus можна виконати симуляцію роботи схем, що дає змогу перевірити коректність підключень і функціонування мікроконтролера в режимі реального часу. Це особливо важливо для налаштування алгоритмів обробки даних та перевірки точності вимірювань.

**3.2 Мікроконтролер PIC16F877A**

Мікроконтролер PIC16F877A, розроблений компанією Microchip Technology, є одним із найвідоміших представників серії PIC (Peripheral Interface Controller). Вперше представлений на ринку у 1999 році, цей мікроконтролер швидко набув популярності серед розробників завдяки своїй простоті, потужності та універсальності. Він відзначається архітектурою RISC (Reduced Instruction Set Computing), що забезпечує високу продуктивність при виконанні різноманітних завдань.

PIC16F877A характеризується 8-розрядною архітектурою, що дозволяє виконувати операції над 8-бітними даними. Цей мікроконтролер містить 40 пінів, з яких 33 можна використовувати як порти вводу-виводу (GPIO). Це надає змогу підключати різноманітні периферійні пристрої, датчики та виконавчі механізми. Однією з ключових особливостей PIC16F877A є можливість роботи з аналоговими сигналами, завдяки вбудованому 10-розрядному аналогово-цифровому перетворювачу (ADC), що дозволяє зчитувати значення напруги з аналогових датчиків.

Мікроконтролер оснащений 14 Кб Flash-пам'яті, 368 байт SRAM та 256 байт EEPROM, що забезпечує достатню пам'ять для зберігання програмного коду та даних. Flash-пам'ять дозволяє здійснювати повторне програмування мікроконтролера, що є важливим для розробки та налагодження програм. Це також дозволяє використовувати мікроконтролер у різних проектах, змінюючи програмне забезпечення відповідно до нових вимог.

На додаток до цього, PIC16F877A має три 8-розрядних таймери та один 16-розрядний таймер, що дозволяє точно вимірювати час і управляти подіями у програмі. Комунікаційні можливості мікроконтролера включають підтримку SPI (Serial Peripheral Interface), I2C (Inter-Integrated Circuit) та USART (Universal Asynchronous Receiver/Transmitter), що робить його сумісним із різними периферійними пристроями.

Основні переваги PIC16F877A включають простоту програмування, завдяки підтримці мови C, широкий спектр доступних бібліотек та активну спільноту користувачів, що надає можливість швидко знаходити рішення для виникаючих проблем. Мікроконтролер також демонструє високу надійність, що є критично важливим для вбудованих систем, які працюють протягом тривалого часу.

У проекті з перевірки блоків живлення (БЖ) ПК PIC16F877A буде використовуватись для збору та обробки даних, управління пристроєм, а також для виведення результатів вимірювань на екран. Його характеристики роблять його ідеальним вибором для реалізації вимірювальних і контролюючих функцій, що забезпечує точність і ефективність роботи пристрою. У наступних підрозділах буде докладно описано, як використовувати мікроконтролер у проекті, включаючи його програмування, налаштування та інтеграцію з іншими елементами системи.

**3.2.1 Архітектура PIC16F877A**

Мікроконтролер PIC16F877A побудований на основі архітектури RISC (Reduced Instruction Set Computing), що забезпечує ефективне виконання інструкцій з мінімальною кількістю циклів тактового сигналу. Це дозволяє досягти високої продуктивності, забезпечуючи виконання до 35 мільйонів інструкцій на секунду (MIPS) при тактовій частоті до 20 МГц. Основні складові архітектури PIC16F877A включають:

* 8-розрядний процесор: Мікроконтролер працює з 8-розрядними даними, що робить його підходящим для виконання основних арифметичних і логічних операцій.
* Програмна пам'ять: Містить 14 Кб Flash-пам'яті, яка використовується для зберігання програми. Flash-пам'ять підтримує повторне програмування, що дозволяє вносити зміни у програму без заміни мікроконтролера.
* Дані пам'яті: Має 368 байт статичної SRAM для зберігання змінних під час виконання програми та 256 байт енергонезалежної EEPROM, яка використовується для зберігання даних, що зберігаються після відключення живлення.
* Регістри: Мікроконтролер обладнаний 36 регістрами загального призначення, які використовуються для виконання арифметичних та логічних операцій. Висока кількість регістрів дозволяє підвищити ефективність програмування.

Периферійні елементи

* Порти вводу-виводу: PIC16F877A має 33 порти вводу-виводу (GPIO), які можна налаштувати як входи або виходи. Це дозволяє підключати різноманітні датчики, виконавчі механізми та інші пристрої.
* Аналогово-цифровий перетворювач (ADC): Вбудований 10-розрядний ADC дозволяє зчитувати аналогові сигнали з датчиків, що забезпечує можливість роботи з різними сенсорами та пристроями.
* Таймери: Мікроконтролер має три 8-розрядних таймери та один 16-розрядний таймер, які можуть використовуватися для точного вимірювання часу та генерування затримок.
* Комунікаційні інтерфейси: PIC16F877A підтримує декілька протоколів для зв'язку з іншими пристроями, включаючи SPI, I2C та USART. Це надає можливість інтеграції з різними периферійними пристроями та модулями.

Споживана потужність

* Енергоспоживання: Мікроконтролер споживає невелику кількість енергії, що робить його підходящим для застосувань, де енергозбереження є критично важливим. Під час роботи в режимі сну споживана потужність може знижуватись до 20 нА, що забезпечує тривалу роботу від батарей.

Інші характеристики

* Кількість інструкцій: PIC16F877A має 35 інструкцій, що робить програмування простим і зручним.
* Захист від перешкод: Мікроконтролер обладнаний засобами захисту від електромагнітних перешкод, що підвищує його надійність у складних умовах експлуатації.

Узагальнюючи, PIC16F877A є потужним та універсальним мікроконтролером, який підходить для широкого спектру застосувань, включаючи автоматизацію, обробку даних, управління пристроями та багато інших завдань у вбудованих системах. Його характеристики та функціональні можливості роблять його ідеальним вибором для розробки пристрою для перевірки блоків живлення ПК.

**3.2.2 Переваги використання PIC16F877A та застосування в проекті**

Мікроконтролер PIC16F877A має ряд переваг, які роблять його одним із найкращих виборів для розробки вбудованих систем. Ось основні з них:

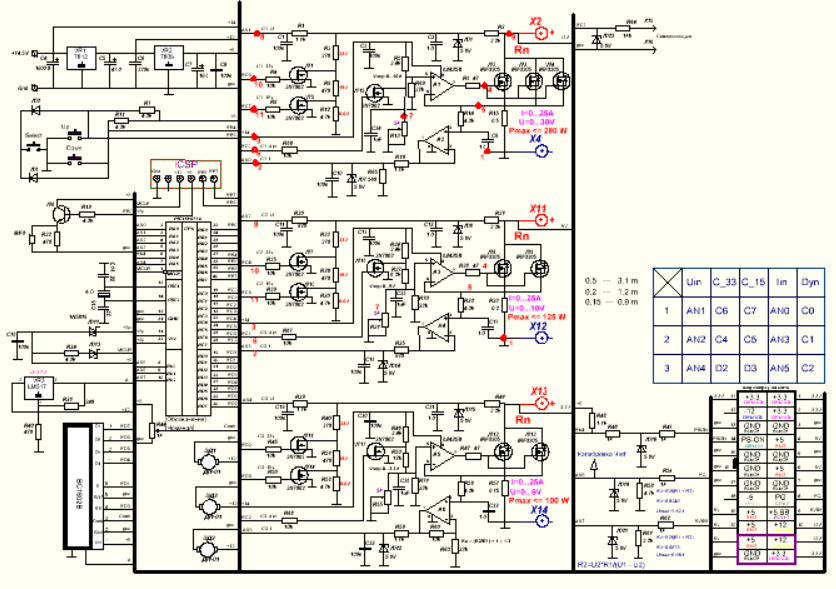
1. Висока продуктивність:
   * Завдяки архітектурі RISC, цей мікроконтролер здатний виконувати до 35 MIPS при тактовій частоті до 20 МГц. Це дозволяє реалізовувати складні алгоритми обробки даних швидко та ефективно.
2. Достатня пам'ять:
   * 14 Кб Flash-пам'яті для програм, 368 байт SRAM для тимчасових даних та 256 байт EEPROM для зберігання конфігураційних параметрів забезпечують необхідний об'єм пам'яті для реалізації різноманітних проектів.
3. Розвинена периферія:
   * Мікроконтролер включає широкий набір периферійних елементів, таких як АЦП, таймери, порти вводу-виводу та комунікаційні інтерфейси (SPI, I2C, USART), що робить його універсальним у використанні в різних застосуваннях.
4. Енергоефективність:
   * Завдяки низькому споживанню енергії, PIC16F877A є ідеальним вибором для проектів, де важливо зберігати енергію, таких як портативні пристрої.
5. Простота програмування:
   * Завдяки широкій підтримці мов програмування, зокрема C, а також численним бібліотекам і прикладам, розробка на базі PIC16F877A стає доступною навіть для початківців.
6. Гнучкість у використанні:
   * Велика кількість GPIO портів та можливість налаштування режимів роботи дозволяють використовувати мікроконтролер в багатьох типах проектів, від простих до складних.

У проекті, що присвячений розробці пристрою для перевірки блоків живлення ПК, PIC16F877A обрано завдяки його чудовій продуктивності та розвиненій периферії. Ось кілька конкретних застосувань цього мікроконтролера:

1. Обробка вимірювальних даних:
   * Використовуючи вбудований АЦП, мікроконтролер може зчитувати аналогові сигнали від датчиків напруги і струму, перетворюючи їх на цифрові значення для подальшої обробки.
2. Контроль параметрів живлення:
   * Мікроконтролер здійснює моніторинг параметрів живлення, таких як напруга і струм, що дозволяє виявляти проблеми в блоках живлення ПК, зокрема перевантаження або коротке замикання.
3. Керування інтерфейсом:
   * За допомогою GPIO портів PIC16F877A може управляти різними індикаторами, кнопками та іншими елементами інтерфейсу, забезпечуючи зручне та інтуїтивно зрозуміле управління пристроєм.
4. Комунікаційні можливості:
   * Завдяки підтримці протоколів SPI, I2C та USART, PIC16F877A може взаємодіяти з іншими пристроями, такими як комп'ютери або інші контролери, для збору та передачі даних.
5. Зберігання та обробка даних:
   * З використанням EEPROM мікроконтролер може зберігати налаштування та результати вимірювань, що дозволяє зберігати важливу інформацію навіть після вимкнення живлення.

Таким чином, PIC16F877A є відмінним вибором для реалізації проекту, оскільки він поєднує в собі високу продуктивність, гнучкість та багатий набір функцій, що робить його ідеальним для розробки пристрою для перевірки блоків живлення ПК.

**3.3 Електрична принципова схема пристрою**

Пристрій для перевірки блоків живлення (БЖ) є складною системою, яка інтегрує кілька важливих модулів, кожен з яких відповідає за виконання специфічних функцій. Основне призначення пристрою полягає в тому, щоб забезпечити можливість тестування основних характеристик БЖ — вимірювання напруги, струму та потужності на різних виходах під час навантаження.

**Рис. 3.1** Електронна схема

На схемі чітко видно використання PIC16F877A як основного мікроконтролера. Він виконує роль керуючого елемента, забезпечуючи обробку вхідних даних, контроль за роботою всієї системи та передавання інформації на дисплей. Кожен канал пристрою пов'язаний з мікроконтролером, що дозволяє йому точно відстежувати стан БЖ у реальному часі.

Основною задачею пристрою є вимірювання та обробка електричних параметрів. Це досягається за допомогою комбінації шунтів, операційних підсилювачів та АЦП (аналогово-цифрових перетворювачів) на базі мікроконтролера. Шунти, які розташовані на кожному каналі (X3, X4, X11, X12), служать для вимірювання струму через відповідний канал. Сигнал з шунта потім подається на операційний підсилювач, який підсилює його для подальшого оцифровування та обробки мікроконтролером.

Операційні підсилювачі, зображені на схемі, виконують важливу роль, забезпечуючи підсилення сигналу до рівня, необхідного для точного вимірювання. Це дозволяє знизити рівень шуму та підвищити точність вимірювань. Для кожного каналу використовуються окремі підсилювачі, що забезпечує незалежність вимірювань та їх стабільність.

Окрім вимірювання параметрів БЖ, пристрій також може створювати навантаження на його виходи для перевірки під різними умовами. Це забезпечується за допомогою MOSFET-транзисторів, які контролюють потоки струму через навантажувальні резистори. MOSFET-транзистори працюють у поєднанні з керуючими сигналами від мікроконтролера, що дозволяє точно регулювати потужність на виходах та стабільність роботи під навантаженням.

Ця система дозволяє не тільки вимірювати напругу та струм, але й проводити довготривалі тестування блоків живлення, оцінюючи їх стабільність та здатність витримувати різні рівні навантаження. Силові транзистори також допомагають уникнути перегріву та перевантаження пристрою.

Точність вимірювань є ключовим аспектом роботи пристрою. Для цього у схемі використовуються додаткові компоненти для згладжування сигналу та зниження перешкод. Конденсатори, що зображені на схемі, виконують роль фільтрів, зменшуючи пульсації та інші небажані коливання в ланцюзі живлення. Це дозволяє отримати стабільні дані та уникнути помилок при вимірюваннях.

Мікроконтролер обробляє ці сигнали через вбудований АЦП, перетворюючи їх з аналогових у цифрові. Далі ці дані аналізуються та відображаються на дисплеї, надаючи користувачеві інформацію про стан БЖ. Важливо зазначити, що пристрій дозволяє вимірювати різні параметри одночасно, що значно підвищує його функціональність.

Управління пристроєм здійснюється за допомогою кількох кнопок, розташованих поруч з мікроконтролером, що дозволяє користувачеві перемикати режими роботи. Наприклад, можна вибрати режими вимірювання різних параметрів БЖ або змінювати рівень навантаження.

Дисплей, що також підключений до мікроконтролера, використовується для відображення всіх основних параметрів роботи пристрою. Користувач може бачити значення напруги, струму, потужності, а також інформацію про режими роботи. Це робить пристрій зручним у використанні для тестування різних типів блоків живлення.

Таким чином, електрична схема пристрою для перевірки блоків живлення є комплексною системою, яка забезпечує вимірювання всіх основних параметрів БЖ. Вона поєднує у собі точне вимірювання напруги та струму, створення навантаження, а також зручний інтерфейс для користувача. Використання сучасних компонентів, таких як PIC16F877A, MOSFET-транзистори та операційні підсилювачі, дозволяє досягти високої точності вимірювань та стабільності роботи пристрою.

**3.4** **Основні вузли електронної схеми**

Електронна схема пристрою для перевірки блоків живлення складається з кількох основних вузлів, кожен з яких виконує критично важливі функції для забезпечення роботи системи. Розглянемо їх детальніше:

1. Мікроконтролерний блок

Основним компонентом схеми є мікроконтролер PIC16F877A, який виконує роль керуючого елемента. Він обробляє сигнали з різних датчиків та елементів системи, забезпечуючи їх правильну роботу. Мікроконтролер відповідає за:

* Збір та обробку даних про напругу, струм та інші параметри від датчиків.
* Керування транзисторами, які регулюють навантаження.
* Передачу оброблених даних на дисплей для візуалізації результатів.

Цей мікроконтролер має розвинену систему АЦП (аналогово-цифрового перетворення), що дозволяє перетворювати аналогові сигнали з датчиків у цифрову форму для подальшої обробки.

2. Модуль вимірювання напруги та струму

Вузол вимірювання напруги та струму є ключовим компонентом системи, оскільки саме він забезпечує можливість контролю основних параметрів роботи блоку живлення. Для цього використовується поєднання таких компонентів:

* Шунти для вимірювання струму. Шунти генерують падіння напруги пропорційно до струму, що протікає через них, і цей сигнал подається на операційні підсилювачі.
* Операційні підсилювачі, які збільшують слабкий сигнал з шунтів, роблячи його доступним для точного вимірювання мікроконтролером.

Кожен канал має окремий операційний підсилювач для покращення точності та зниження перешкод під час вимірювань.

3. Силовий модуль керування навантаженням

Цей вузол відповідає за створення навантаження на тестований блок живлення для перевірки його роботи в різних режимах. Основними компонентами цього модуля є MOSFET-транзистори, які керують потоком струму через навантажувальні резистори. Перевага MOSFET-транзисторів полягає у їх високій швидкості перемикання та здатності керувати великими струмами при мінімальних втратам потужності.

Мікроконтролер через сигнали керування визначає, які транзистори повинні бути активовані, що дозволяє створювати змінне навантаження на виході блоку живлення.

4. Фільтрація та стабілізація сигналів

Для забезпечення точності вимірювань і стабільної роботи пристрою, в схемі передбачені фільтри, які складаються з конденсаторів та резисторів. Ці елементи виконують функцію фільтрації перешкод та згладжування напруги, що дозволяє уникати небажаних коливань сигналу під час вимірювань.

Фільтри використовуються як у ланцюгах вимірювання, так і в ланцюгах живлення, що допомагає стабілізувати роботу системи і забезпечує надійність отриманих результатів.

5. Дисплей та керування пристроєм

Пристрій має інтерфейс для взаємодії з користувачем, що забезпечується за допомогою дисплея і кількох кнопок керування. Дисплей виводить на екран результати вимірювань, такі як напруга, струм та потужність, дозволяючи користувачу легко контролювати стан блоку живлення.

Керування пристроєм здійснюється через кнопки, які підключені до мікроконтролера. Вони дозволяють перемикати режими роботи, змінювати навантаження та переглядати інші параметри.

6. Живлення та стабілізація системи

Пристрій потребує стабільного джерела живлення для забезпечення безперебійної роботи всіх компонентів. У схемі передбачені блоки стабілізації напруги, які забезпечують необхідний рівень живлення для мікроконтролера та інших компонентів схеми.

Стабілізація напруги реалізована за допомогою стабілітронів та конденсаторів, що забезпечують надійну роботу схеми, захищаючи її від перепадів напруги.

**3.5 Розробка програмного забезпечення для PIC16F877A**

Для розробки програмного забезпечення мікроконтролера PIC16F877A була використана мова програмування **C**, яка забезпечує гнучкість, простоту та ефективність у керуванні периферійними пристроями. Завдяки цьому підходу стало можливим забезпечити необхідну функціональність мікроконтролера для точного вимірювання параметрів блоків живлення, а також для інтеграції з інтерфейсом користувача.

Програмне забезпечення було розроблене у середовищі MikroC PRO for PIC v7.6.0, що надає потужні інструменти для програмування та налагодження мікроконтролерів. У програмі використовуються вбудовані модулі мікроконтролера, такі як ADC (Аналогово-цифровий перетворювач), таймери, порти вводу-виводу та обробка переривань, що дозволяє ефективно вирішувати завдання управління та контролю.

Програмне забезпечення для PIC16F877A виконує ряд ключових завдань, що забезпечують функціональність пристрою для тестування блоків живлення:

1. Аналогово-цифрове перетворення (ADC):
   * Основною задачею програми є вимірювання напруги та струму на різних каналах живлення блоку живлення. Для цього використовується вбудований ADC мікроконтролера. Оцифровані значення зчитуються та обробляються для подальшого аналізу.
   * Наприклад, для кожного каналу здійснюється зчитування аналогових сигналів, їхнє перетворення у цифрову форму, після чого ці значення використовуються для виведення на дисплей і прийняття рішень про стан блоку живлення.
2. Керування навантаженням:
   * В програмному забезпеченні реалізовані функції керування навантаженням на кожному каналі блоку живлення. Навантаження регулюється шляхом перемикання транзисторів, керованих мікроконтролером, залежно від рівня напруги та струму.
   * Також реалізована функція зміни навантаження, що дозволяє пристрою імітувати роботу блоку живлення під різними навантаженнями. Це досягається за допомогою змінних резисторів або керуванням навантажувальними резисторами через транзистори.
3. Відображення даних:
   * Програмне забезпечення передає результати вимірювань на LCD-дисплей. Дані про напругу, струм та загальну потужність відображаються у реальному часі. Це дозволяє користувачу легко контролювати стан кожного каналу блоку живлення.
   * Користувач також може взаємодіяти з пристроєм через кнопки, що дозволяє змінювати режими вимірювання або налаштувати певні параметри.
4. Захист від перевантаження:
   * Програма реалізує захист від перевантаження шляхом постійного моніторингу значень струму і напруги. Якщо ці параметри перевищують допустимі межі, мікроконтролер автоматично відключає навантаження та інформує користувача про перевантаження через дисплей.
   * Алгоритм захисту базується на перевірці значень з ADC у кожному циклі програми. Якщо виявлено аномальні значення, активується відповідний захист.

Структура програми

Програмне забезпечення побудоване таким чином, що його можна розділити на кілька функціональних блоків, які відповідають за різні аспекти роботи пристрою:

Ініціалізація:

* На початку роботи програми виконується налаштування мікроконтролера та його периферійних пристроїв. Це включає ініціалізацію портів вводу-виводу, модулів ADC, таймерів та налаштування дисплея.
* У функції ініціалізації також завантажуються калібрувальні коефіцієнти з EEPROM, що необхідні для точного вимірювання напруги та струму.

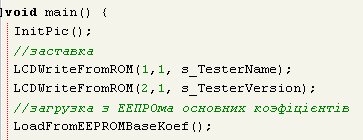
Основний цикл:

* Після ініціалізації програма переходить до виконання основного циклу. У ньому програма періодично зчитує значення з ADC, перевіряє, чи не перевищують вони допустимі межі, і виводить результати на дисплей.
* У цьому циклі також виконується контроль навантаження і вмикання/вимикання каналів блоку живлення залежно від отриманих даних. Обробка переривань:
* Для обробки критичних подій, таких як зміна навантаження або перевищення порогових значень, використовуються апаратні переривання мікроконтролера. Це дозволяє швидко реагувати на небажані зміни у параметрах блоку живлення.
* Обробка переривань дозволяє покращити продуктивність програми, оскільки основні вимірювання та контроль навантаження можуть виконуватись паралельно.

**3.6 Опис програмного коду**

Основними завданнями програми є вимірювання напруги та струму блоків живлення, контроль перевантажень, а також керування інтерфейсом користувача через дисплей та кнопки.

Після ввімкнення пристрою виконується ініціалізація мікроконтролера та периферійних пристроїв. Програма завантажує калібрувальні коефіцієнти з EEPROM та відображає заставку на дисплеї.



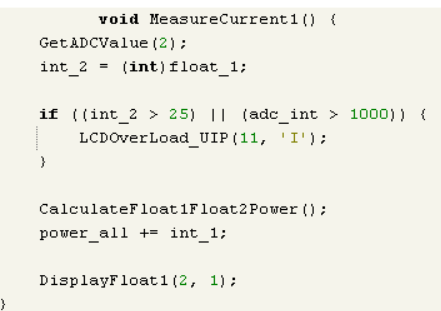
**Рис.3.2** Вивід на дисплей меню вибору

Ця частина коду виконує початкове налаштування мікроконтролера та периферійних пристроїв, зокрема дисплея. Після цього на екран виводиться інформація про пристрій, і користувачу надається можливість вибору режиму тестування блоку живлення.

Важливим компонентом програмного забезпечення є процедура вимірювання напруги на різних каналах. Для запобігання перевантаженню АЦП використовується перемикання дільників напруги.

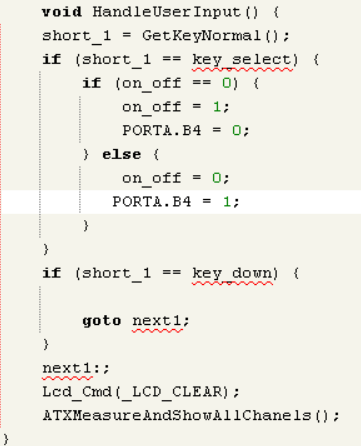
**Рис.3.3** Вимірювання напруги

У цьому коді здійснюється вибір діапазону вимірювання напруги для точного визначення значень. АЦП перемикається між різними діапазонами в залежності від рівня напруги, щоб запобігти перевантаженню.

Для кожного каналу програма також вимірює струм та перевіряє, чи не перевищує він допустимі межі.

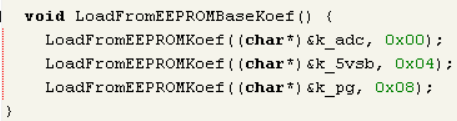
**Рис.3.4** Вимірювання струму та контроль перевантаження

Ця функція перевіряє рівень струму на кожному каналі і виводить попередження про перевантаження, якщо перевищені порогові значення. Також обчислюється потужність для кожного каналу.

Користувач може взаємодіяти з пристроєм через кнопки. Програма реагує на натискання кнопок для перемикання режимів та активації тестування.

**Рис.3.5** Обробка введення та керування

Ця частина коду реалізує функціонал управління пристроєм за допомогою клавіш. Натискання кнопок дозволяє користувачу включати або вимикати блок живлення, а також перемикати режими вимірювань.

Прилад має функцію калібрування, яка забезпечує точність вимірювань на всіх каналах. Калібрувальні коефіцієнти зберігаються в EEPROM і використовуються для обчислення точних значень напруги та струму.

**Рис.3.6** Завантаження коефіцієнтів

Процедура LoadFromEEPROMBaseKoef завантажує коефіцієнти для різних параметрів, що використовуються при вимірюванні. Це дозволяє забезпечити точність вимірювань на кожному каналі.

Програма дозволяє вимірювати напругу та струм на різних каналах, контролювати перевантаження, а також взаємодіяти з користувачем через дисплей та кнопки. Калібрування пристрою гарантує точність вимірювань, а можливість зберігати коефіцієнти в EEPROM робить систему зручною та надійною для використання.

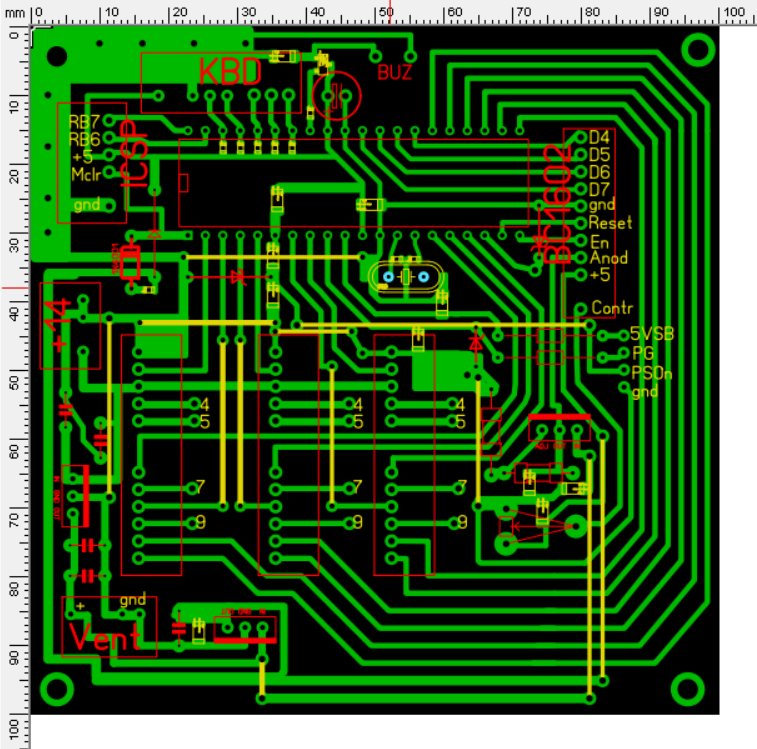
**3.7 Створення друкованої плати**

Для забезпечення ефективної та надійної роботи тестера блоків живлення, було розроблено комплекс друкованих плат. Основна плата виконує функцію центрального модуля, до якого підключаються три додаткові плати, кожна з яких відповідає за вимірювання на окремих каналах блоку живлення.

Основна друкована плата є головним модулем тестера. Вона містить мікроконтролер PIC16F877A, керуючі елементи (кнопки, інтерфейси для введення даних), а також підключення до дисплея для виведення результатів вимірювання. Крім того, основна плата забезпечує з'єднання з іншими модулями та здійснює обробку сигналів, що надходять від каналів вимірювання.

Основні компоненти на платі:

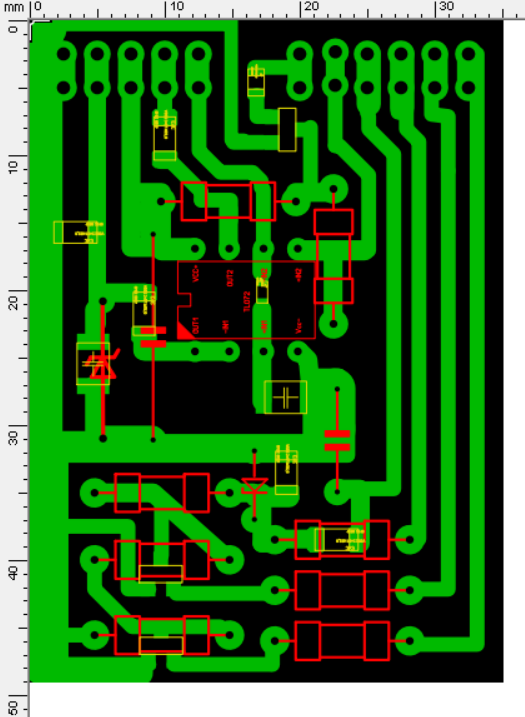
* Мікроконтролер PIC16F877A – головний елемент управління всією системою.
* Інтерфейси для підключення периферійних пристроїв – дисплея, кнопок, живлення.
* Порти для підключення додаткових плат – три порти для підключення вимірювальних модулів.
* Схема живлення – стабілізатор живлення для забезпечення коректної роботи мікроконтролера та периферійних елементів.



**Рис.3.7** Основна плата

Основна плата також відповідає за збір та обробку даних з вимірювальних каналів, що надходять від додаткових модулів, і виводить їх на дисплей для подальшого аналізу.

Кожна з трьох додаткових друкованих плат відповідає за вимірювання напруги та струму на окремому каналі блоку живлення. Вони виконують функцію зняття аналогових даних з каналів та передачі цих даних на основну плату для подальшої обробки.



**Рис. 3.8** Плата n-каналу

Основні компоненти на платах:

* Аналогово-цифрові перетворювачі (АЦП) – для перетворення аналогових сигналів напруги та струму в цифрові дані, які потім передаються на основну плату.
* Резистивні дільники – для регулювання вхідних сигналів та захисту від перевищення допустимих рівнів напруги.
* Транзистори та реле – для керування навантаженням і стабілізації сигналів на вимірювальних каналах.

Кожна з цих плат може працювати незалежно одна від одної, що дозволяє виконувати тестування кількох каналів одночасно.

Основна плата взаємодіє з додатковими через спеціальні порти для передачі даних і керування сигналами. Всі вимірювальні плати підключаються до головної плати, яка отримує дані від кожного каналу та обробляє їх.

Завдяки такій модульній конструкції система є гнучкою та зручною для обслуговування. Якщо один з каналів потребує корекції або оновлення, можна просто замінити відповідну додаткову плату, не втручаючись у роботу всієї системи.

Для проектування друкованих плат використовувалося програмне забезпечення Sprint-Layout. Ця програма дозволяє створювати високоякісні макети плат, оптимізувати їхнє розташування, а також створювати необхідні шари для виготовлення друкованих плат.

Основні кроки при розробці плат:

* Створення схеми з'єднань компонентів на основній платі.
* Оптимізація розташування компонентів на додаткових вимірювальних платах для мінімізації втрат сигналу та зниження перешкод.
* Виготовлення друкованих плат із застосуванням стандартної технології травлення.
* Перевірка працездатності плат та їхнє тестування перед інтеграцією в систему.

Після проектування плати були виготовлені та зібрані. Важливою частиною процесу була перевірка якості монтажу, оскільки точність вимірювань залежить від правильного з'єднання компонентів та їхньої функціональності.

**3.8 Використання тестера для перевірки блоків живлення**

Тестер для блоків живлення ПК є зручним та функціональним інструментом, що дозволяє користувачеві легко здійснювати вимірювання напруги, струму та потужності, а також виконувати різні тести на блоках живлення. Керування тестером відбувається за допомогою кількох кнопок, кожна з яких виконує певну функцію в залежності від тривалості натискання. Система звукових сигналів супроводжує всі дії користувача, підтверджуючи виконання команд.

Загальний принцип управління

Керування тестером реалізовано через п'ять основних кнопок, що дозволяють виконувати всі необхідні операції:

* Кнопка Select: коротке натискання цієї кнопки підтверджує вибір пункту меню або функції, аналогічно до кнопки "Enter" на комп'ютері. Вона використовується для запуску тестування або підтвердження вибору режиму роботи.
* Кнопка Select\_long: довге натискання цієї кнопки (більше 2 секунд) викликає додаткові функції, наприклад, перехід на рівень глибше в меню або запис налаштувань у пам'ять EEPROM. Ця функція важлива для калібрування пристрою або налаштування параметрів тестування.
* Кнопка Up: використовується для навігації по меню. Коротке натискання кнопки дозволяє змінювати режим роботи або пересуватися по пунктах меню. Це зручно, коли необхідно обрати певний режим або перевірити певний канал блоку живлення.
* Кнопка Up\_long: при довгому натисканні кнопка дозволяє повернутися на рівень вище в меню або вийти з поточного режиму. Це допомагає швидко перемикатися між режимами та пунктами меню, особливо коли користувач налаштовує пристрій або здійснює детальне тестування.
* Кнопка Down: аналогічно кнопці Up, але зворотна функція — пересування вниз по меню або зміна режиму. Ця кнопка дозволяє швидко перемикати параметри тестування або переглядати різні канали блоку живлення.
* Кнопка Down\_long: довге натискання цієї кнопки використовується для виходу з глибших рівнів меню або швидкого повернення до основного режиму.



**Рис. 3.9** Головне меню тестера

У головному меню тестера доступні кілька основних функцій:

1. Перевірка ATX блоку живлення: цей режим дозволяє протестувати стандартні ATX блоки живлення. Пристрій автоматично вимірює напругу на ключових каналах блоку живлення (+12В, +5В, +3.3В) та відображає ці значення на дисплеї. Також у цьому режимі можна вимірювати силу струму на кожному з каналів.
2. Перевірка довільного блоку живлення: цей режим дозволяє перевірити будь-який інший тип блоку живлення, не обмежуючись стандартами ATX. Він забезпечує гнучкість при тестуванні нестандартних пристроїв.

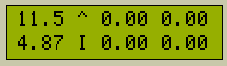
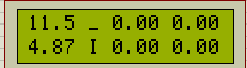
Кожен з цих режимів дозволяє здійснювати вимірювання як в статичному, так і в динамічному режимах. Динамічне тестування допомагає визначити, наскільки стабільно блок живлення витримує змінні навантаження.

Детальний опис режимів

1. Режим тестування ATX блоку живлення: При переході до цього режиму за допомогою кнопки Select, тестер автоматично починає вимірювати параметри блоку живлення. На дисплеї відображаються напруга на каналах +12В, +5В та +3.3В, а також поточні значення сили струму або потужності для кожного каналу.

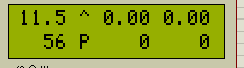
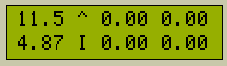


**Рис. 3.10** Режим тестування ATX блоку живлення

Користувач може керувати станом блоку живлення за допомогою кнопки Select. Коротке натискання цієї кнопки дозволяє увімкнути або вимкнути блок живлення для перевірки його працездатності під нав антаженням.

**Рис. 3.11** Селектор стану вимкнення / увімкненняблоку живлення

Всі зміни режиму підтверджуються звуковими сигналами, що інформують користувача про успішне виконання дій.

Додатково, коротке натискання кнопки Down дозволяє змінювати відображувану інформацію на нижньому рядку дисплея — перемикатися між показаннями сили струму та потужності.

**Рис. 3.12** Селектор показанням сили струму та потужності

Це особливо зручно під час динамічних тестів, коли необхідно відстежувати роботу блоку живлення під різним навантаженням.

У разі необхідності виходу з цього режиму, користувач може довго натиснути кнопку Up, щоб повернутися до головного меню або перейти до інших налаштувань.

2.Режим перевірки довільного блоку живлення: Цей режим надає більше гнучкості для тестування нестандартних блоків живлення, які не відповідають ATX стандарту. Використання цього режиму подібне до перевірки ATX блоків, однак тут користувач може налаштовувати додаткові параметри та проводити специфічні тести на різних каналах. Як і в попередньому режимі, результати вимірювань відображаються на дисплеї в реальному часі.

**3.8.1 Калібрування тестера для перевірки блоків живлення**

Окрім стандартних режимів тестування, тестер має вбудовану функцію калібрування. Це дозволяє точно налаштувати вимірювальні ланцюги пристрою, забезпечуючи коректні показники напруги та струму на кожному з каналів. Калібрування виконується через меню, яке доступне після довгого натискання кнопки Select\_long у головному меню.

Під час калібрування користувач може вносити корективи в роботу тестера, зберігаючи налаштування у пам'яті EEPROM, що гарантує їх збереження навіть після вимкнення пристрою. Це дозволяє точно налаштувати пристрій під конкретні потреби та забезпечити стабільну роботу під час тривалих тестувань.

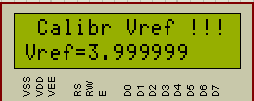
1.Калібрування опорної напруги для АЦП

Цей етап калібрування є першочерговим і обов’язковим. Він проводиться для налаштування точного вимірювання напруги через аналогово-цифровий перетворювач (АЦП), оскільки будь-які неточності в його налаштуванні можуть вплинути на всі подальші вимірювання.

Процедура калібрування:

1. Спочатку, для входу в режим калібрування, необхідно натиснути та утримувати кнопку Select в головному меню тестера.
2. Після цього на вхід AN6 (9-я нога PIC мікроконтролера) подається зовнішня

напруга 4.00 В від еталонного джерела напруги.



**Рис. 3.13** Поточне виміряне значення напруги

1. На дисплеї відображається поточне виміряне значення напруги. Якщо воно не відповідає 4.00 В, необхідно продовжити калібрування (крок 5).
2. Якщо відображене значення максимально відповідає 4.00 В, калібрування не потрібне, можна вийти (перейти до кроку 6).
3. Для виконання калібрування слід натиснути і утримувати Select, щоб розрахувати та зберегти коефіцієнт Kadc у пам'ять EEPROM.
4. Для виходу без збереження результатів слід натиснути Up\_long.

Коефіцієнт Kadc розраховується за формулою:

Kadc = Uin / ADC = 4.00 В / значення АЦП

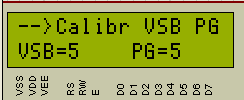
Цей коефіцієнт визначає, скільки вольт відповідає одній одиниці АЦП. Калібрування проводиться одноразово, але може бути повторене, якщо є підозри, що вимірювання неточні.

2. Калібрування дільників для 5VSB та PG

Калібрування цих каналів дозволяє коректно вимірювати напругу на лініях 5VSB та PG. Цей етап є необхідним для забезпечення точності показників на цих каналах.

Процедура калібрування:

1. Потрібно закоротити входи 5VSB і PG на роз'ємі.
2. На ці закорочені входи подається еталонна напруга 5.00 В.



**Рис. 3.14** Поточне значення напруги

1. Якщо дисплей показує правильне значення 5 В, калібрування не потрібне (перейдіть до кроку 5). Якщо відображене значення відрізняється, потрібно виконати калібрування (перейдіть до кроку 4).
2. Натисніть і утримуйте Select, щоб розрахувати і зберегти коефіцієнти дільників у пам'ять EEPROM.
3. Для виходу без збереження результатів натисніть Up\_long.

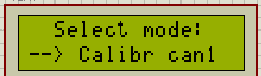
Варто зазначити, що перед калібруванням дільників для 5VSB і PG необхідно виконати калібрування опорної напруги для АЦП.

3. Калібрування каналів вимірювання напруги

Для кожного з трьох каналів вимірювання виконується окрема калібрування, яка забезпечує точність роботи тестера на всіх діапазонах напруги.

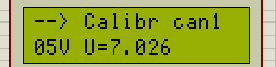
Процедура калібрування каналів:

1. В головному меню виберіть клавішами Up або Down режим калібрування для відповідного каналу.



**Рис. 3.15** Вибір каналу

1. Для входу в режим калібрування натисніть Select.
2. На вхід каналу подайте напругу 5.00 В. Якщо відображене значення відповідає 5 В, натисніть Down, щоб перейти до наступного діапазону. Якщо значення відрізняється, натисніть і утримуйте Select для калібрування.



**Рис. 3.16** Поточне значення напруги

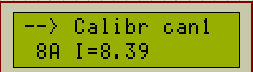
1. Повторіть процедуру для напруги 12.00 В.
2. Повторіть для напруги 21.00 В.
3. Якщо на всіх діапазонах вимірювання напруга відповідає еталонному значенню, натисніть Up\_long для виходу без збереження результатів.

4. Калібрування датчика струму

Цей етап виконується для точного вимірювання сили струму. Для цього подаються еталонні значення напруги та струму на відповідний канал, після чого проводиться налаштування коефіцієнтів для коректної роботи датчика.

Процедура калібрування датчика струму:

1. На вхід подається напруга у межах 5–10 В та струм 8.00 А за допомогою еталонного амперметра.



**Рис. 3.17** Поточне значення струму

1. Далі виконується калібрування датчика струму за допомогою довгого натискання кнопки Select.
2. Для виходу без збереження результатів натисніть Up\_long.

**ВИСНОВОК ДО РОЗДІЛУ 3**

У цьому розділі було проведено детальний аналіз і опис процесу розробки тестера для перевірки блоків живлення, що включає в себе кілька ключових етапів: створення електричної схеми, проектування друкованих плат, вибір мікроконтролера PIC16F877A, а також розробку програмного забезпечення з використанням MikroC.

Модульна архітектура пристрою, що складається з основної плати та трьох додаткових плат, які вимірюють параметри на окремих каналах, є одним з найважливіших аспектів розробки. Основна плата виконує роль центрального контролера, на якій розміщується мікроконтролер. Вона відповідає за обробку сигналів з усіх підключених модулів та керує роботою тестера. Додаткові плати виконують функцію зняття аналогових даних з каналів живлення і передачі цих даних на основну плату для подальшої обробки.

Програмне забезпечення, реалізоване мовою C у середовищі MikroC, демонструє гнучкість у управлінні мікроконтролером і інтеграції з різними компонентами.

Проектування друкованих плат за допомогою Sprint-Layout дозволило створити якісні макети, оптимізувати розташування компонентів та забезпечити їх надійне з’єднання. Виготовлення друкованих плат відповідно до проекту зменшує ймовірність помилок під час монтажу та підвищує ефективність роботи тестера.

Отже, створений тестер для перевірки блоків живлення відповідає сучасним вимогам до точності, надійності та функціональності. Він є універсальним інструментом для виконання різноманітних тестувань, корисним як для професіоналів, так і для любителів електроніки. Результати реалізації даного проекту підтверджують досягнення поставлених цілей і відкривають нові можливості для подальшого розвитку та вдосконалення пристрою.

**ВИСНОВКИ**

У даній дипломній роботі була проведена комплексна розробка пристрою для тестування блоків живлення, що включає в себе три основних розділи: аналіз існуючих технологій, розробка тестера та створення програмного забезпечення.

У першому розділі було вивчено різні типи тестерів для блоків живлення, включаючи їх функціональні можливості, переваги та недоліки. Результати аналізу показали, що сучасні тестери повинні забезпечувати точність вимірювань, зручність у використанні та гнучкість у налаштуваннях. Це стало основою для розробки тестера, що відповідає актуальним вимогам ринку. Приділено увагу різним підходам до тестування, зокрема, використовуванню аналогово-цифрових перетворювачів, які є критично важливими для точної оцінки параметрів.

У другому розділі охоплюються важливі етапи проектування електричної схеми, вибору компонентів та створення друкованих плат. Особливу увагу було приділено вибору мікроконтролера PIC16F877A, який забезпечує достатню обчислювальну потужність та можливість інтеграції з різними периферійними пристроями. Модульна архітектура пристрою, що складається з основної плати та трьох вимірювальних плат, дозволяє виконувати вимірювання одночасно на кількох каналах, що підвищує ефективність тестування. Основна плата виконує роль центрального контролера, на якій розміщується мікроконтролер, а додаткові плати відповідають за вимірювання параметрів на різних каналах, що дозволяє проводити комплексні тести.

У третьому розділі була розглянута розробка програмного забезпечення, що реалізує всі необхідні функції для управління тестером. Програмне забезпечення, написане мовою C у середовищі MikroC, демонструє гнучкість у управлінні мікроконтролером і периферійними пристроями. Основні функції програми включають:

Аналогово-цифрове перетворення (ADC), яке є критично важливим для точного вимірювання напруги та струму на різних каналах. Це дозволяє отримувати точні дані, які надалі обробляються та відображаються на дисплеї.

Керування навантаженням, що реалізується через транзистори та реле, дозволяє зменшити навантаження на тестовані блоки живлення та забезпечує безпеку під час проведення тестів.

Відображення даних на дисплеї, що забезпечує користувачеві можливість контролювати параметри блоків живлення в реальному часі. Інтерфейс користувача спрощує взаємодію з пристроєм, дозволяючи швидко отримувати необхідну інформацію.

Захист від перевантаження, що реалізовано через алгоритми моніторингу струму та напруги. При перевищенні заданих порогових значень, мікроконтролер автоматично відключає навантаження, запобігаючи можливим пошкодженням.

Важливу роль у забезпеченні точності вимірювань грає калібрування тестера. Процедури калібрування охоплюють як налаштування опорної напруги для АЦП, так і налаштування дільників для вимірювання напруги та струму на різних каналах. Кожна процедура була чітко описана, що гарантує правильність роботи пристрою в різних умовах.

Створений тестер для перевірки блоків живлення є універсальним інструментом, що може бути використаний як у професійних сервісах, так і для домашнього використання. Він відповідає вимогам сучасних стандартів якості, що робить його цінним внеском у галузь електроніки. Це пристрій, який спрощує процес тестування, забезпечуючи точність і швидкість.

Дана розробка має великий потенціал для подальшого розвитку. Можливість інтеграції з новими технологіями, такими як бездротові системи моніторингу або програмні рішення для аналізу даних, відкриває нові горизонти для вдосконалення пристрою. Наприклад, можна розширити функціонал тестера, додаючи нові режими вимірювання або вдосконалюючи алгоритми калібрування. Це може включати підтримку нових стандартів блоків живлення або інтеграцію з сучасними засобами обробки даних, такими як мобільні додатки для збирання та аналізу інформації.

Таким чином, тестер не лише виконує основні функції з перевірки блоків живлення, але й має потенціал для адаптації до швидко змінюваних вимог ринку електроніки. Це робить його важливим інструментом для всіх, хто займається розробкою, виробництвом або обслуговуванням електронних пристроїв. Результати реалізації проекту підтверджують досягнення поставлених цілей, а також відкривають можливості для подальшого розвитку та оптимізації технологій тестування електронних пристроїв.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Billings K., Pressman A. I. Switching Power Supply Design, 3rd Ed. 3-тє вид. McGraw-Hill Professional, 2008. 744 с.
2. Designing Control Loops for Linear and Switching Power Supplies. Artech House Publishers, 2012.
3. Кобзарьов, Ю. В. Теорія та практика конструкції пристроїв індуктивної зарядки високовольтних акумуляторів: Ю. В. Кобзарьов. – Київ : НТУУ "КПІ", 2009. – 294 с.
4. Maniktala S. Switching power supply design & optimization. McGraw-Hill Professional, 2004. 386 с.
5. Маляр В. С. Теоретичні основи електротехніки. Львів: Львівська політехніка, 2018. – 416 с.
6. Матвієнко M.П. Основи електроніки. Підручник. Київ: Ліра-К, 2021. – 360 с.
7. Winfield H. The Art of Electronics, 3-тє вид. Cambridge : Cambridge University Press, 2015. 1224 с.
8. Коман Б. Основи комп’ютерної електроніки / Б. Коман, М. Мисько. – Львів: ЛНУ, 2019. – 430 с.
9. Barr M. Programming Embedded Systems in C and C ++. O'Reilly Media, Inc., 1999. 194 с.
10. Mazidi M. A., Naimi S., Naimi S. Avr Microcontroller and Embedded Systems: Pearson New International Edition. Pearson Education, Limited, 2015. 749 с.
11. Predko M. Programming and customizing the PIC microcontroller, 3-тє вид. New York: McGraw Hill, 2008. 1263 с.
12. Rizvi S. R. Microcontroller Programming: An Introduction. Taylor & Francis Group, 2016. 546 с.
13. Mazidi M. A., Naimi S., Naimi S. Avr microcontroller and embedded systems the: using assembly and C. Pearson Education, Limited, 2011. 791 с.
14. Sanchez J. Microcontroller programming: The microchip PIC. Boca Raton, FL: CRC Press, 2007. 804 с.
15. McKinlay R., Causey D., Mazidi M. A. PIC Microcontroller and Embedded Systems: Using Assembly and C for PIC18. MicroDigitalEd, 2016. 630 с.
16. Barr M. Programming embedded systems: With C and GNU development tools, 2-ге вид. Sebastopol, CA: O'Reilly, 2007. 301 с.
17. Zurawski R. Embedded Systems Handbook: Embedded Systems Design and Verification. Taylor & Francis Group, 2018. 666 с.
18. Massa A., Barr M. Programming Embedded Systems: With C and GNU Development Tools, 2nd Edition. O'Reilly Media, Inc., 2006. 301 с.
19. Scherz P. Practical Electronics for Inventors, 2-ге вид. McGraw-Hill/TAB Electronics, 2006. 952 с.
20. Рябенький В.М. та ін. Схемотехніка: Пристрої цифрової електроніки. Київ: 399 с.
21. Etc M. N. Power Electronics: Converters, Applications and Design, 3-тє вид. NJ: Wiley, 2003. 824 с.
22. Bates D. J., Malvino A. P. Electronic Principles. McGraw Hill Higher Education, 2007.
23. Buchla D. Electric Circuits Fundamentals, 7-ме вид. Not Avail, 2006. 400 с.
24. Maksimovic D., Erickson R. W. Fundamentals of Power Electronics. Springer, 2012. 908 с.
25. Матюшов М.В. Початок роботи з мікроконтролерами STM8. Київ, 2016. – 208 с.
26. Mazidi M. A. The 8051 microcontroller and embedded systems: Using Assembly and C, 2-ге вид. Upper Saddle River, N.J: Pearson/Prentice Hall, 2006. 626 с.
27. Grace T. Programming and Interfacing ATMEL AVR Microcontrollers. 2016. 272 с.
28. Wright C., Sloss A., Symes D. ARM System Developer's Guide: Designing and Optimizing System Software. Elsevier Science & Technology Books, 2004. 689 с.
29. Berger A. Embedded Systems Design: An Introduction to Processes, Tools, and Techniques. Lawrence, Kan: CMP Books, 2002. 237 с.
30. Brown M. Power Supply Cookbook. Boston: Butterworth-Heinemann, 1994. 238с.

**ДОДАТКИ**

Аркушів 26

**Київ 2024**

**ДОДАТОК А**

**activeload.c**

extern short mode1;

extern short short\_1;

extern short dyn\_mode\_flag;

extern const code char s\_SelectMode[];

const code char s\_TesterName[];

const code char s\_TesterVersion[];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "built\_in.h"

#include "global.h"

#include "keyboard.h"

#include "calibr.h"

#include "atxtester.h"

#include "anypowertester.h"

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void interrupt\_vector() iv 0x0004 ics ICS\_AUTO {

//ставим флаг по переполнению T1

if (PIR1.TMR1IF == 1){

T1CON.TMR1ON = 0;

PIR1.TMR1IF = 0;

dyn\_mode\_flag.B7 = 1;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void main() {

InitPic();

//заставка

LCDWriteFromROM(1,1, s\_TesterName);

LCDWriteFromROM(2,1, s\_TesterVersion);

//загрузка с ЕЕПРОма основных коэффициентов

LoadFromEEPROMBaseKoef();

Delay\_1sec();

Delay\_1sec();

//меню что делать

Lcd\_Cmd(\_LCD\_CLEAR); // Clear display

mode1 = 0;

start\_soft:;

LCDWriteFromROM(1,1, s\_SelectMode);

LCDShowStringFromEEPROM(2, 8 + mode1); //

do{

short\_1 = GetKeyNormal();

// short\_1 = key\_select;

//key up short

if (short\_1 == key\_up){

if (mode1 > 0){

mode1 --;

LCDShowStringFromEEPROM(2, 8 + mode1); //

}

}

//key down short

else if (short\_1 == key\_down){

if (mode1 < 1){

mode1 ++;

LCDShowStringFromEEPROM(2, 8 + mode1); //

}

}

//key select short

else if (short\_1 == key\_select){

switch(mode1){

//atx power

case 0:

DoATXTester();

goto start\_soft;

break;

//any power

case 1:

DoAnyPowerTester();

goto start\_soft;

break;

}

}

else if (short\_1 == key\_select\_long){

DoCalibrationMenu();

goto start\_soft;

}

}while(1);

}

**atxtester.c**

extern const code char s\_SelectMode[];

extern const code char s\_VSB[];

extern const code char s\_PG[];

extern const code char s\_OFF[];

extern const code char s\_ON[];

extern short short\_1;

extern short mode1;

extern short on\_off;

extern short i\_p;

extern short dyn\_mode\_flag;

extern unsigned int adc\_in;

extern unsigned int adc\_int;

extern unsigned int adc\_middle;

extern unsigned int power\_all;

extern unsigned int int\_1;

extern unsigned int int\_2;

extern unsigned long int key\_press;

extern float k\_adc;

extern float k\_5vsb;

extern float k\_pg;

extern float k\_can;

extern float adc\_float;

extern float float\_1;

extern float float\_2;

extern char txt\_6[6];

extern char txt\_9[9];

extern char txt\_19[19];

//-----------------------------------------------------------------------------

#include "built\_in.h"

#include "global.h"

#include "keyboard.h"

//#include "dynamics.h"

//-----------------------------------------------------------------------------

void Show5VSBandPG(){

LCDWriteFromROM(1,1,s\_VSB);

GetADCValue(7);

float\_1 = adc\_float / k\_5vsb;

DisplayFloat1(1, 5);

LCDWriteFromROM(2,1,s\_PG);

GetADCValue(6);

float\_1 = adc\_float / k\_pg;

DisplayFloat1(2, 4);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void ATXMeasureAndShowAllChanels(){

//

if (dyn\_mode\_flag.B0 == 1){

LCDClearString2();

}

//

if (i\_p == 1){

Lcd\_Chr(2, 6, 'I');

}

power\_all = 0;

//------------------ can1 ----------------------------------

MeasureVoltage1();

int\_1 = (int)float\_1;

float\_2 = float\_1;

//overload U

if ( (int\_1 > 30) || (adc\_int > 1000) ){

LCDOverLoad\_UIP(1,1, 'U');

}

else{

DisplayFloat1(1, 1);

}

//I

MeasureCurrent1();

int\_2 = (int)float\_1;

if ( (int\_2 > 25) || (adc\_int > 1000) ){

LCDOverLoad\_UIP(1,1,'I');

}

CalculateFloat1Float2Power();

power\_all += int\_1;

//show current

if (i\_p == 1){

DisplayFloat1(2, 1);

}

//show power

else{

DisplayInt1ThreeDigit(2, 1);

}

//åñëè ïåðåáîð ïî ìîùíîñòè

if ( int\_1 > 290 ){

LCDOverLoad\_UIP(1,1,'P');

}

//---------------------- can2 ----------------------------------------

MeasureVoltage2();

int\_1 = (int)float\_1;

float\_2 = float\_1;

//overload U

if ( (int\_1 > 21) || (adc\_int > 1000) ){

LCDOverLoad\_UIP(1,8, 'U');

}

else{

DisplayFloat1(1, 8);

}

//I

MeasureCurrent2();

int\_2 = (int)float\_1;

if ( (int\_2 > 25) || (adc\_int > 1000) ){

LCDOverLoad\_UIP(1,8,'I');

}

CalculateFloat1Float2Power();

power\_all += int\_1;

//show current

if (i\_p == 1){

DisplayFloat1(2, 8);

}

//show power

else{

DisplayInt1ThreeDigit(2, 5);

}

//åñëè ïåðåáîð ïî ìîùíîñòè

if ( int\_1 > 125 ){

LCDOverLoad\_UIP(1,8,'P');

}

//----------------------- can3 ------------------------------------

MeasureVoltage3();

int\_1 = (int)float\_1;

float\_2 = float\_1;

//overload U

if ( (int\_1 > 21) || (adc\_int > 1000) ){

LCDOverLoad\_UIP(1,13,'U');

}

else{

DisplayFloat1(1, 13);

}

//I

MeasureCurrent3();

int\_2 = (int)float\_1;

if ( (int\_2 > 25) || (adc\_int > 1000) ){

LCDOverLoad\_UIP(1,13,'I');

}

CalculateFloat1Float2Power();

power\_all += int\_1;

//show current

if (i\_p == 1){

DisplayFloat1(2, 13);

}

//show power

else{

DisplayInt1ThreeDigit(2, 9);

}

//åñëè ïåðåáîð ïî ìîùíîñòè

if ( int\_1 > 100 ){

LCDOverLoad\_UIP(1,13,'P');

}

//power all

if (i\_p == 0){

int\_1 = power\_all;

DisplayInt1ThreeDigit(2, 14);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void DoATXTester(){

Lcd\_Cmd(\_LCD\_CLEAR);

on\_off = 0;

i\_p = 1;

PORTA.B4 = 1;

PORTC.RC0 = 0;

PORTC.RC1 = 0;

PORTC.RC2 = 0;

//

start\_:;

do{

Show5VSBandPG();

if (on\_off == 1){

LCDWriteFromROM(2,14,s\_ON);

}

else{

LCDWriteFromROM(2,14,s\_OFF);

}

Delay\_100ms();

//

short\_1 = GetKeyNormal();

// short\_1 = key\_down;

start\_1:;

//key up short

if (short\_1 == key\_up){

}

//key up long

if (short\_1 == key\_up\_long){

goto end\_;

}

//key down short

else if (short\_1 == key\_down){

goto next1;

}

//key select

else if (short\_1 == key\_select){

if (on\_off == 0){

on\_off = 1;

PORTA.B4 = 0;

}

else{

on\_off = 0;

PORTA.B4 = 1;

}

}

//key select long

else if (short\_1 == key\_select\_long){

}

}while(1);

//-- ìåðÿåì íàïðÿæåíèå -------------------------------------------------

next1:;

Lcd\_Cmd(\_LCD\_CLEAR);

do{

ATXMeasureAndShowAllChanels();

//show on/off status

//on

if (on\_off == 1){

Lcd\_Chr(1, 6, '^');

}

//off

else{

Lcd\_Chr(1, 6, '\_');

}

//

PlaySound\_500\_50();

Delay\_10ms();

//

short\_1 = GetKeyNormal();

// short\_1 = key\_select\_long;

//key up short

if (short\_1 == key\_up){

Lcd\_Cmd(\_LCD\_CLEAR);

goto start\_;

}

//key up long

else if (short\_1 == key\_up\_long){

goto end\_;

}

//key down short

else if (short\_1 == key\_down){

if (i\_p == 0){

i\_p = 1;

}

else{

i\_p = 0;

}

dyn\_mode\_flag.B0 = 1;

}

//key down long

else if (short\_1 == key\_down\_long){

}

//key select long

else if (short\_1 == key\_select\_long){

}

//key select

else if (short\_1 == key\_select){

if (on\_off == 0){

on\_off = 1;

PORTA.B4 = 0;

}

else{

on\_off = 0;

PORTA.B4 = 1;

}

}

}while(1);

end\_:;

LCDWriteFromROM(1,1, s\_SelectMode);

LCDShowStringFromEEPROM(2, 8 + mode1); //

}

**global.c**

#include "stdlib.h"

#include "built\_in.h"

#include "global.h"

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// LCD module connections

sbit LCD\_RS at RD0\_bit;

sbit LCD\_EN at RD1\_bit;

sbit LCD\_D4 at RD4\_bit;

sbit LCD\_D5 at RD5\_bit;

sbit LCD\_D6 at RD6\_bit;

sbit LCD\_D7 at RD7\_bit;

sbit LCD\_RS\_Direction at TRISD0\_bit;

sbit LCD\_EN\_Direction at TRISD1\_bit;

sbit LCD\_D4\_Direction at TRISD4\_bit;

sbit LCD\_D5\_Direction at TRISD5\_bit;

sbit LCD\_D6\_Direction at TRISD6\_bit;

sbit LCD\_D7\_Direction at TRISD7\_bit;

// End LCD module connections

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

extern const code char s\_SelectMode[];

extern const code char s\_VSB[];

extern const code char s\_PG[];

extern const code char s\_OFF[];

extern const code char s\_ON[];

extern short short\_1;

extern short mode1;

extern short on\_off;

extern short i\_p;

extern short counter;

extern short dyn\_mode\_flag;

extern unsigned int adc\_in;

extern unsigned int adc\_int;

extern unsigned int adc\_middle;

extern unsigned int power\_all;

extern unsigned int int\_1;

extern unsigned int int\_2;

extern unsigned long int key\_press;

extern float k\_adc;

extern float k\_5vsb;

extern float k\_pg;

extern float k\_can;

extern float adc\_float;

extern float float\_1;

extern float float\_2;

extern char txt\_6[6];

extern char txt\_9[9];

extern char txt\_19[19];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void CalculateFloat1Float2Power(){

float\_2 \*= float\_1;

int\_1 = (int)float\_2;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void CalculateFloat1ADC(){

float\_1 = adc\_float / k\_can;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MeasureVoltage1(){

PORTC.RC6 = 0;

PORTC.RC7 = 0;

LoadFromEEPROMKoef((char\*)&k\_can, 0x0c);

GetADCValue(1);

if (adc\_int > 950) {

PORTC.RC6 = 0;

PORTC.RC7 = 1;

LoadFromEEPROMKoef((char\*)&k\_can, 0x10);

GetADCValue(1);

if (adc\_int > 950) {

PORTC.RC6 = 1;

PORTC.RC7 = 0;

LoadFromEEPROMKoef((char\*)&k\_can, 0x14);

GetADCValue(1);

}

}

CalculateFloat1ADC();

}

//-----------------------------------------------------------------------------

void MeasureCurrent1(){

LoadFromEEPROMKoef((char\*)&k\_can, 0x30);

GetADCValue(0);

CalculateFloat1ADC();

}

//-----------------------------------------------------------------------------

void MeasureVoltage2(){

PORTC.RC4 = 0;

PORTC.RC5 = 0;

LoadFromEEPROMKoef((char\*)&k\_can, 0x18);

GetADCValue(2);

if (adc\_int > 950) {

PORTC.RC4 = 0;

PORTC.RC5 = 1;

LoadFromEEPROMKoef((char\*)&k\_can, 0x1c);

GetADCValue(2);

if (adc\_int > 950) {

PORTC.RC4 = 1;

PORTC.RC5 = 0;

LoadFromEEPROMKoef((char\*)&k\_can, 0x20);

GetADCValue(2);

}

}

CalculateFloat1ADC();

}

//-----------------------------------------------------------------------------

void MeasureCurrent2(){

LoadFromEEPROMKoef((char\*)&k\_can, 0x34);

GetADCValue(3);

CalculateFloat1ADC();

}

//-----------------------------------------------------------------------------

void MeasureVoltage3(){

PORTD.RD2 = 0;

PORTD.RD3 = 0;

LoadFromEEPROMKoef((char\*)&k\_can, 0x24);

GetADCValue(4);

if (adc\_int > 950) {

PORTD.RD2 = 0;

PORTD.RD3 = 1;

LoadFromEEPROMKoef((char\*)&k\_can, 0x28);

GetADCValue(4);

if (adc\_int > 950) {

PORTD.RD2 = 1;

PORTD.RD3 = 0;

LoadFromEEPROMKoef((char\*)&k\_can, 0x2c);

GetADCValue(4);

}

}

CalculateFloat1ADC();

}

//-----------------------------------------------------------------------------

void MeasureCurrent3(){

LoadFromEEPROMKoef((char\*)&k\_can, 0x38);

GetADCValue(5);

CalculateFloat1ADC();

}

//-----------------------------------------------------------------------------

void DisplayInt1ThreeDigit(short row, short col){

WordToStr(int\_1, txt\_6);

txt\_6[0] = txt\_6[2];

txt\_6[1] = txt\_6[3];

txt\_6[2] = txt\_6[4];

txt\_6[3] = 0;

Lcd\_Out(row, col, txt\_6);

}

//-----------------------------------------------------------------------------

void DisplayFloat1(short row, short col){

// xx.xx

Lcd\_Chr(row, col, ' ');

for (counter = 0; counter < 4; counter++){

Lcd\_Chr\_CP(' ');

}

FloatToStr(float\_1, txt\_6);

//3.254e-3

if (float\_1 < 0.01){

txt\_6[0] = '0';

txt\_6[1] = '.';

txt\_6[2] = '0';

txt\_6[3] = '0';

}

//3.254e-2

else if (float\_1 < 0.1){

Lo(adc\_in) = txt\_6[0];

txt\_6[0] = '0';

txt\_6[1] = '.';

txt\_6[2] = '0';

txt\_6[3] = Lo(adc\_in);

}

//3.254e-1

else if (float\_1 < 1.0){

Hi(adc\_in) = txt\_6[0];

Lo(adc\_in) = txt\_6[2];

txt\_6[0] = '0';

txt\_6[1] = '.';

txt\_6[2] = Hi(adc\_in);

txt\_6[3] = Lo(adc\_in);

}

//2.35698

else{

}

txt\_6[4] = 0;

Lcd\_Out(row, col, txt\_6);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void SaveToEEPROMKoef(char \*p1, short b1){

//êîýô ÷åðåç óêàçàòåëü

for (short\_1 = 0; short\_1 < 4; short\_1++){

EEPROM\_Write(b1 ++, \*p1);

Delay\_10ms();

Delay\_10ms();

p1++;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void LoadFromEEPROMKoef(char \*p1, short b1){

//÷èòàåì â ïàìÿòü èç ååïðîìà êîýô ÷åðåç óêàçàòåëü

for (short\_1 = 0; short\_1 < 4; short\_1++){

\*p1 = EEPROM\_Read(b1 ++);

Delay\_10ms();

p1++;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void LoadFromEEPROMBaseKoef(){

LoadFromEEPROMKoef((char\*)&k\_adc, 0x00);

LoadFromEEPROMKoef((char\*)&k\_5vsb, 0x04);

LoadFromEEPROMKoef((char\*)&k\_pg, 0x08);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void GetADCValue(short canal){

//âûáèðàåì íóæíûé êàíàë

ADCON0.CHS2 = 0; ADCON0.CHS1 = 0; ADCON0.CHS0 = 0;

switch(canal){

case 0:

break;

case 1:

ADCON0.CHS0 = 1;

break;

case 2:

ADCON0.CHS1 = 1;

break;

case 3:

ADCON0.CHS1 = 1; ADCON0.CHS0 = 1;

break;

case 4:

ADCON0.CHS2 = 1;

break;

case 5:

ADCON0.CHS2 = 1; ADCON0.CHS0 = 1;

break;

case 6:

ADCON0.CHS2 = 1; ADCON0.CHS1 = 1;

break;

case 7:

ADCON0.CHS2 = 1; ADCON0.CHS1 = 1; ADCON0.CHS0 = 1;

break;

}

adc\_int = 0;

short\_1 = 32;

do{

Delay\_22us();

ADCON0.GO = 1;

do{

}while(ADCON0.GO == 1);

//

Lo(adc\_in) = ADRESL;

Hi(adc\_in) = ADRESH;

adc\_int += adc\_in;

}while( --short\_1 > 0);

//

adc\_int /= 32;

//

adc\_float = adc\_int \* k\_adc;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void LCDShowStringFromEEPROM(char lcd\_row, char num){

//read 16 bytes fromm eeprom string

num \*= 16;

for (short\_1 = 0; short\_1 < 16; short\_1++){

txt\_19[short\_1] = EEPROM\_Read(num + short\_1);

}

txt\_19[16] = 0;

Lcd\_Out(lcd\_row, 1, txt\_19);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void InitPic(){

INTCON = 0;

/\*

bit 7 RBPU: PORTB Pull-up Enable bit

bit 6 INTEDG: Interrupt Edge Select bit

bit 5 T0CS: TMR0 Clock Source Select bit

bit 4 T0SE: TMR0 Source Edge Select bit

bit 3 PSA: Prescaler Assignment bit

bit 2-0 PS2:PS0: Prescaler Rate Select bits

\*/

OPTION\_REG = 0b00001000;

/\*

bit 7 PSPIE: Parallel Slave Port Read/Write Interrupt Enable bit(1)

bit 6 ADIE: A/D Converter Interrupt Enable bit

bit 5 RCIE: USART Receive Interrupt Enable bit

bit 4 TXIE: USART Transmit Interrupt Enable bit

bit 3 SSPIE: Synchronous Serial Port Interrupt Enable bit

bit 2 CCP1IE: CCP1 Interrupt Enable bit

bit 1 TMR2IE: TMR2 to PR2 Match Interrupt Enable bit

bit 0 TMR1IE: TMR1 Overflow Interrupt Enable bit

\*/

PIE1 = 0;

/\*

bit 7 Unimplemented: Read as ‘0’

bit 6 CMIE: Comparator Interrupt Enable bit

bit 5 Unimplemented: Read as ‘0’

bit 4 EEIE: EEPROM Write Operation Interrupt Enable bit

bit 3 BCLIE: Bus Collision Interrupt Enable bit

bit 2-1 Unimplemented: Read as ‘0’

bit 0 CCP2IE: CCP2 Interrupt Enable bit

\*/

PIE2 = 0;

TRISA = 0b11101111;

//îòêðûòûé ñòîê - èíâåðñèÿ

PORTA.F4 = 1;

TRISB = 0b11111110;

TRISC = 0b00000000;

TRISD = 0b00000000;

TRISE = 0b11101111;

//

T1CON.TMR1ON = 0;

T2CON.TMR2ON = 0;

ADCON0.ADON = 0;

CVRCON = 0;

//ADC

//ADCS1 ADCS0 CHS2 CHS1 CHS0 GO/DONE — ADON

//0 1 = FOSC/8

// 0 0 0 = Channel 0 (AN0)

ADCON0 = 0b01000000;

//ADFM ADCS2 — — PCFG3 PCFG2 PCFG1 PCFG0

// 1 = Right justified. Six (6) Most Significant bits of ADRESH are read as ‘0’.

// 0 = FOSC/8

// 7 6 5 4 3 2 1 0 VREF+ VREF-

// 0 0 0 0 A A A A A A A A VDD VSS

ADCON1 = 0b10000000;

ADCON0.ADON = 1;

//

PORTC = 0b00000000;

PORTD = 0b00000000;

//

Lcd\_Init();

Lcd\_Cmd(\_LCD\_CLEAR); // Clear display

Lcd\_Cmd(\_LCD\_CURSOR\_OFF); // Cursor off

// Sound\_Init(&PORTB, 0);

Delay\_10ms();

//

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void LCDClearString2(){

Lcd\_Chr(2, 1, ' ');

for (counter = 0; counter < 15; counter++){

Lcd\_Chr\_CP(' ');

}

dyn\_mode\_flag.B0 = 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void LCDOverLoad(short row, short col){

Lcd\_Chr(row, col, 'O'); Lcd\_Chr\_CP('V'); Lcd\_Chr\_CP('E'); Lcd\_Chr\_CP('R');

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void LCDOverLoad\_UIP(short row, short col, char ch){

Lcd\_Chr(row, col, '-'); Lcd\_Chr\_CP('-'); Lcd\_Chr\_CP(ch); Lcd\_Chr\_CP('-');

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void MeasureAndShowAllChanels(){

//

if (i\_p == 1){

Lcd\_Chr(2, 6, 'I');

}

else{

Lcd\_Chr(2, 6, 'P');

}

//------------------ can1 ----------------------------------

MeasureVoltage1();

int\_1 = (int)float\_1;

float\_2 = float\_1;

//overload

if ( (int\_1 > 30) || (adc\_int > 1000) ){

LCDOverLoad(1,1);

}

else{

DisplayFloat1(1, 1);

}

MeasureCurrent1();

int\_2 = (int)float\_1;

if ( (int\_2 > 25) || (adc\_int > 1000) ){

LCDOverLoad(2, 1);

}

else{

CalculateFloat1Float2Power();

//show current

if (i\_p == 1){

DisplayFloat1(2, 1);

}

//show power

else{

DisplayInt1ThreeDigit(2, 1);

}

}

//åñëè ïåðåáîð ïî ìîùíîñòè

if ( int\_1 > 290 ){

LCDOverLoad(1,1);

LCDOverLoad(2,1);

}

//---------------------- can2 ----------------------------------------

MeasureVoltage2();

int\_1 = (int)float\_1;

float\_2 = float\_1;

//overload

if ( (int\_1 > 21) || (adc\_int > 1000) ){

LCDOverLoad(1, 8);

}

else{

DisplayFloat1(1, 8);

}

MeasureCurrent2();

int\_2 = (int)float\_1;

if ( (int\_2 > 25) || (adc\_int > 1000) ){

LCDOverLoad(2, 8);

}

else{

CalculateFloat1Float2Power();

//show current

if (i\_p == 1){

DisplayFloat1(2, 8);

}

//show power

else{

DisplayInt1ThreeDigit(2, 8);

}

}

if ( int\_1 > 125 ){

LCDOverLoad(1, 8);

LCDOverLoad(2, 8);

}

//----------------------- can3 ------------------------------------

MeasureVoltage3();

int\_1 = (int)float\_1;

float\_2 = float\_1;

//overload

if ( (int\_1 > 21) || (adc\_int > 1000) ){

LCDOverLoad(1, 13);

}

else{

DisplayFloat1(1, 13);

}

MeasureCurrent3();

int\_2 = (int)float\_1;

if ( (int\_2 > 25) || (adc\_int > 1000) ){

LCDOverLoad(2, 13);

}

else{

CalculateFloat1Float2Power();

//show current

if (i\_p == 1){

DisplayFloat1(2, 13);

}

//show power

else{

DisplayInt1ThreeDigit(2, 13);

}

}

//åñëè ïåðåáîð ïî ìîùíîñòè

if ( int\_1 > 100 ){

LCDOverLoad(1, 13);

LCDOverLoad(2, 13);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void PlaySound\_1000\_50(){

PORTB.B0 = 0;

for (counter = 0; counter < 15; counter++){

PORTB.B0 = 1;

Delay\_500us();

PORTB.B0 = 0;

Delay\_500us();

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void PlaySound\_500\_50(){

PORTB.B0 = 0;

for (counter = 0; counter < 10; counter++){

PORTB.B0 = 1;

Delay\_1ms();

PORTB.B0 = 0;

Delay\_1ms();

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void LCDWriteFromROM( char row, char col, const char \*text ){

Lcd\_Out( row, col, "" );

while( \*text )

Lcd\_Chr\_CP( \*text++ );

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**variables.c**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//in ROM

const code char s\_5V[] = " 5V";

const code char s\_12V[] = "12V";

const code char s\_19V[] = "19V";

const code char s\_8A[] = " 8A";

const code char s\_U[] = "U=";

const code char s\_I[] = "I=";

const code char s\_VSB[] = "VSB=";

const code char s\_PG[] = "PG=";

const code char s\_Vref[] = "Vref=";

const code char s\_SelectMode[] = " Select mode: ";

const code char s\_SelectCalibr[] = "Sel calibration:";

const code char s\_NothingChanged[] = "Nothing changed.";

const code char s\_EEPROMSaved[] = "EEPROM Saved !!!";

const code char s\_OFF[] = "OFF";

const code char s\_ON[] = "On ";

const code char s\_TesterName[] = " Power Tester ";

const code char s\_TesterVersion[] = "(c)KWWSoft v2.3";

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

char txt\_4[4] absolute 0x00A0; //IRP not work !!!!!!!!

char txt\_19[19] absolute 0x00A0;

char txt\_6[6] absolute 0x00A4;

char txt\_15[15] absolute 0x00A4;

char txt\_9[9] absolute 0x00AA;

//

short short\_1 absolute 0x00CD;

//1 - on, 0 - off

short on\_off absolute 0x00e7;

//1 - I, 0 - P

short i\_p absolute 0x00c6;

short counter absolute 0x00BE;

unsigned int power\_all absolute 0x00BF;

unsigned int int\_1 absolute 0x00CE;

unsigned int int\_2 absolute 0x00D9;

float float\_1 absolute 0x00e8;

float float\_2 absolute 0x00c2;

float k\_adc absolute 0x00ec; //EEPROM 0x00 0x01 0x02 0x03 Vref

float k\_5vsb absolute 0x00DF; //EEPROM 0x04 0x05 0x06 0x07 5VSB

float k\_pg absolute 0x00DB; //EEPROM 0x08 0x09 0x0a 0x0b PG

float k\_can absolute 0x00E3;

//EEPROM 0x0C 0x0d 0x0e 0x0f 1 - 6V

//EEPROM 0x10 0x11 0x12 0x13 1 - 15V

//EEPROM 0x14 0x15 0x16 0x17 1 - 33V

//EEPROM 0x30 0x31 0x32 0x33 1 - current

//EEPROM 0x18 0x19 0x1a 0x1b 2 - 6V

//EEPROM 0x1C 0x1d 0x1e 0x1f 2 - 15V

//EEPROM 0x20 0x21 0x22 0x23 2 - 33V

//EEPROM 0x34 0x35 0x36 0x37 2 - current

//EEPROM 0x24 0x25 0x26 0x27 3 - 6V

//EEPROM 0x28 0x29 0x2a 0x2b 3 - 15V

//EEPROM 0x2C 0x2d 0x2e 0x2f 3 - 33V

//EEPROM 0x38 0x39 0x3a 0x3b 3 - current

short mode1 absolute 0x00D0;

short mode2 absolute 0x00C1;

//------------------------------------------------------------------------------

//------------------------------------------------------------------------------

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* --> keyboard.c begin <-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned long int key\_press absolute 0x00D5;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* --> keyboard.c end <-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* --> atxtester.c begin <-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

unsigned int adc\_in absolute 0x00D1;

unsigned int adc\_int absolute 0x00D3;

float adc\_float absolute 0x00C7;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* --> atxtester.c end <-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* --> dynamics.c begin <-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

short dyn\_mode absolute 0x00CB;

short dyn\_mode\_flag absolute 0x00CC;

/\* 7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0

1 - åñòü ïðåðûâàíèå îò òàéìåðà

1 - èçìåíèëñÿ ðåæèì ðàáîòû íàãðóçêè

0 - ïåðâûé ïîëóïåðèîä, 1 - âòîðîé ïîëóïåðèîä ïåðâûé êàíàë

0 - ïåðâûé ïîëóïåðèîä, 1 - âòîðîé ïîëóïåðèîä âòîðîé êàíàë

0 - ïåðâûé ïîëóïåðèîä, 1 - âòîðîé ïîëóïåðèîä òðåòèé êàíàë

1 - ïîêàçûâàòü íàïðÿæåíèå/òîê èëè 0 - íè÷åãî

1 - ïîêàçûâàòü íàïðÿæåíèå, 0 - ïîêàçûâàòü òîê

1 - íóæåí âûçîâ LCDClearString2()

void LoadFromEEPROMBaseKoef() {

LoadFromEEPROMKoef((char\*)&k\_adc, 0x00); // Ç÷èòóâàííÿ êîåô³ö³ºíòà äëÿ ÀÖÏ

LoadFromEEPROMKoef((char\*)&k\_5vsb, 0x04); // Ç÷èòóâàííÿ êîåô³ö³ºíòà äëÿ 5VSB

LoadFromEEPROMKoef((char\*)&k\_pg, 0x08); // Ç÷èòóâàííÿ êîåô³ö³ºíòà äëÿ PG

}

**global.h**

#ifndef \_\_global\_h

#define \_\_global\_h

//-----------------------------------------------------------------------------

#define key\_nothing 0

#define key\_up 1

#define key\_down 2

#define key\_select 3

#define key\_up\_long 4

#define key\_down\_long 5

#define key\_select\_long 6

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//-----------------------------------------------------------------------------

void InitPic();

void LCDShowStringFromEEPROM(char lcd\_row, char num);

void GetADCValue(short canal);

void LoadFromEEPROMKoef(char \*p1, short b1);

void LoadFromEEPROMBaseKoef();

void SaveToEEPROMKoef(char \*p1, short b1);

void DisplayFloat1(short row, short col);

void MeasureVoltage1();

void MeasureCurrent1();

void MeasureVoltage2();

void MeasureCurrent2();

void MeasureVoltage3();

void MeasureCurrent3();

void CalculateFloat1ADC();

void LCDClearString2();

void LCDOverLoad(short row, short col);

void LCDOverLoad\_UIP(short row, short col, char ch);

void MeasureAndShowAllChanels();

void PlaySound\_500\_50();

void PlaySound\_1000\_50();

void LCDWriteFromROM( char row, char col, const char \*text );

void DisplayInt1ThreeDigit(short row, short col);

void CalculateFloat1Float2Power();

//-----------------------------------------------------------------------------

#endif