МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЦЕНТРАЛЬНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ АВТОМАТИКИ ТА ЕНЕРГЕТИКИ

КАФЕДРА ПРОГРАМУВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ

Програмування мікроконтролерних систем

Методичні вказівки до виконання лабораторних робіт

з елементами кредитно - модульної
системи організації навчального процесу

для студентів денної форми навчання
за спеціальністю 123 «Комп'ютерна інженерія»

Укладачі:
Доцент Смірнов В.В.
Доцент Смірнова Н.В.

Кропивницький
2022 рік

Програмування мікроконтролерних систем : методичні вказівки до виконання лабораторних робіт для студентів денної форми навчання за спеціальністю 123 «Комп’ютерна інженерія» / уклад. : В.В. Смірнов, Н.В. Смірнова ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : ЦНТУ, 2022. – 139 с.

Затверджено *на засіданні кафедри програмування комп'ютерних систем та мереж:* ***10 травня 2022 р., протокол № 8***

Укладачі:

Смірнов Володимир Вікторович, к.т.н., доцент кафедри кібербезпеки та програмного забезпечення.

Смірнова Наталя Володимирівна, к.т.н., доцент кафедри кібербезпеки та програмного забезпечення.

Для студентів денної форми навчання, що вивчають навчальну дисципліну “ Програмування мікроконтролерних систем ” за спеціальністю 123 «Комп’ютерна інженерія»

У стислій формі викладені основні апаратні та програмні засоби мікроконтролерних систем для рішення завдань керування об'єктами в автоматизованих системах керування.

Представлені практичні рішення навчальних завдань для кращого освоєння досліджуваного матеріалу, представлені варіанти навчальних завдань для самостійного придбання практичних навичок.

ЗМІСТ

1.	Опис навчальної дисципліни "Програмування мікроконтролерних систем"	4
2.	Лабораторні роботи з дисципліни ”Програмування мікроконтролерних систем”	5
3.	Змістовні модулі	6
4.	Оцінка успішності в балах при повному виконанні умов і графіку навчального процесу	8
	Лабораторні роботи:	
5.	**№1** Введення – виведення дискретних сигналів	8
6.	**№2** Робота з перериваннями мікроконтролера	29
7.	**№3** Робота з АЦП	40
8.	**№4** Робота з ЦАП по інтерфейсу SPI	50
9.	**№5** Динамічна індикація. Виведення інформації на LED – дисплей	66
10.	**№6** Модуль CCP. Керування двигуном постійного струму	75
11.	**№7** Робота із зовнішньою EEPROM пам'яттю по інтерфейсу I^2C	85
12.	**№8** Управління сервоприводом	100
13.	Список літератури	108

1. Опис навчальної дисципліни
"Програмування мікроконтролерних систем"

Основна мета курсу полягає в придбанні досконалих знань і навичок роботи з апаратним і програмним забезпеченням систем управління об'єктом на базі мікро - ЕОМ.

Внаслідок проведення лекцій студенти повинні отримати теоретичні знання та методику ефективної роботи з сучасними мікроконтролерними системами управління.

Завдання вивчення дисципліни

- Вивчення теоретичних основ систем управління;
- Вивчення теоретичних основ мікроконтролерних систем;
- Вивчення інтерфейсів обміну даними.
- Вивчення теоретичних основ методів управління;
- Вивчення теоретичних основ методів перетворення сигналів;
- Вирішення завдань введення - виведення дискретних і аналогових сигналів;
- Вирішення завдань управління об'єктами;
- Набуття практичних навиків в сфері програмування мікроконтролерних систем.

Предметом навчальної дисципліни є архітектура мікроконтролерів і програмне забезпечення мікроконтролерів для систем управління об'єктом.

У результаті вивчення навчальної дисципліни студент повинен знати:

- Вирішувати завдання: дискретних сигналів, аналогових сигналів.
- Виконувати управління об’єктами на базі мікроЕОМ.
- Виконувати програмування систем управління на базі мікроЕОМ.
- Визначати параметри вхідних сигналів.

– Вирішувати завдання введення/виведення дискретних і аналогових сигналів.

вміти:

– Проводити управління об’єктами на базі мікроЕОМ.

– Розробляти прикладні та системні програми для систем управління об’єктами на базі мікроЕОМ.

2. Лабораторні роботи з дисципліни ”Програмування мікроконтролерних систем”

№ заняття	Назва практичного заняття	Кільк. годин
1.	Введення – виведення дискретних сигналів	2
2.	Робота з перериваннями мікроконтролера	2
3.	Робота з АЦП	2
4.	Робота з ЦАП по інтерфейсу SPI	2
5.	Динамічна індикація. Виведення інформації на LED – дисплей	2
6.	Модуль CCP. Керування двигуном постійного струму	2
7.	Робота із зовнішньою EEPROM пам'яттю по інтерфейсу I^2C	3
8.	Управління сервоприводом	2
	Всього годин	**17**

Шкала оцінювання

За шкалою ECTS	За національною шкалою	За шкалою навчального закладу
A	відмінно	90-100
B-C	добре	75-89
D-E	задовільно	60-74
F-X	незадовільно з можливістю повторного складання	35-59
F	незадовільно з обов’язковим повторним курсом	1-34

3. Змістовні модулі

Навчальне навантаження складається з 2 – змістовних модулів, які включають в себе лекції, лабораторні роботи, самостійну роботу і контроль знань. Система оцінки успішності в балах включає тестовий поточний контроль при виконанні лабораторних робіт, модульний контроль і оцінку самостійної роботи.

Перший модуль.

Мікроконтролери. Середовище розробки програм PSW CSS. Введення – виведення дискретних сигналів. Автоматизовані системи. Переривання контролера. АЦП. Робота з ЦАП по інтерфейсу SPI

Лабораторні роботи:

№1 Введення – виведення дискретних сигналів

№2 Робота з перериваннями мікроконтролера

№3 Робота з АЦП

№4 Робота з ЦАП по інтерфейсу SPI

Другий модуль.

Датчики. Інтерфейс SPI. Семисегментний LED – дисплей. Модуль CCP. Зовнішній таймер з інтерфейсом I^2C. Паралельне виконання програм. LCD – дисплей. EEPROM – пам'ять. Управління сервоприводом

Лабораторні роботи:

№5 Динамічна індикація. Виведення інформації на LED – дисплей

№6 Модуль CCP. Керування двигуном постійного струму

№7 Робота із зовнішньою EEPROM пам'яттю по інтерфейсу I^2C

№8 Управління сервоприводом

4. Оцінка успішності в балах при повному виконанні умов і графіку навчального процесу

№ модуля	Матеріал лекцій	Кількість лекцій (годин)	Бали за вивчення лекційного матеріалу	Лабораторні роботи				Конспект лекцій	Максимальна сума балів
				Теми лабораторних робіт	Години	Тестовий контроль	Виконання л.р.		
1.	1. Мікроконтролери. 2. Введення – виведення дискретних сигналів. 3. Автоматизовані системи. 4. Переривання контролера. 5. АЦП мікроконтролера 6. Датчики. 7. ЦАП. Інтерфейс SPI	14	10	**№1** Введення – виведення дискретних сигналів **№2** Робота з перериваннями мікроконтролера **№3** Робота з АЦП **№4** Робота з ЦАП по інтерфейсу SPI	21	15	20		50
2.	1 Модуль CCP. 2. Семисегментний LED – дисплей. 3. EEPROM – пам'ять. 4. Зовнішній таймер з інтерфейсом I^2C. 5. LCD – дисплей. 6. Управління сервоприводом. 7. Керування промисловим роботом.	14	10	**№5** Динамічна індикація. Виведення інформації на LED – дисплей **№6** Модуль CCP. Керування двигуном постійного струму **№7** Робота із зовнішньою EEPROM пам'яттю по інтерфейсу I^2C **№8** Управління сервоприводом	21	15	20	10	50
	Всього:	28	20		17	30	40	10	100

ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МІКРОКОНТРОЛЕРІВ

АПАРАТНО-ПРОГРАМНІ КОМПЛЕКСИ

Для виконання лабораторних робіт на базі PIC – мікроконтролерів, авторами були розроблені, виготовлені і впроваджені у навчальний процес спеціальні апаратно-програмний комплекси (АПК) (рис 1, 2, 5).

Рисунок 1 – Апаратно-програмний комплекс № 1

Апаратно-програмні комплекси призначені для створення та відлагодження програмного забезпечення для різних мікроконтролерних систем у рамках дисциплін **«Програмне забезпечення управляючих мікро-ЕОМ»**, **«Програмування мікроконтролерних систем»** та **«Internet Of Things»**

Рисунок 2 – Апаратно-програмний комплекс № 2

До складу Апаратно-програмного комплексу № 1 (рис. 1) входить:

- базовий інтегрований апаратно-програмний модуль;
- автономний апаратно-програмний модуль;
- модуль-емулятор об'єкта управління;
- навчальна програма дисципліни;
- завдання і методичні вказівки до виконання лабораторних робіт;
- приклади виконання завдань;
- інтегроване середовище розробки програмного забезпечення на мові програмування С;
- драйвер для завантаження бінарного коду у FLASH-пам'ять програм мікроконтролера по інтерфейсу ICSP.

Завдання, які вирішуються за допомогою Апаратно-програмного комплексу № 1:

ознайомлення з мікроконтролерами і засобами розробки:

- введення/виведення дискретних сигналів;
- виведення інформації на LCD-дисплей;
- робота з перериваннями мікроконтролера;
- введення/виведення даних по інтерфейсу RS-232;

введення і обробка аналогових сигналів за допомогою АЦП:

- організація введення з цифрової клавіатури;

виведення даних через зовнішній ЦАП по інтерфейсу SPI:

- створення драйвера інтерфейсу SPI;

виведення інформації на семисегментний LED-дисплей:

- створення драйвера динамічної індикації;

робота з модулями PWM (ШІМ):

- управління потужністю у навантаженні;
- управління двигуном постійного струму. Напрямок та швидкість обертання;

робота з зовнішньою EEPROM пам'яттю по інтерфейсу I^2C:

- запис/читання байтів і блоків даних;

розробка ПІД-регулятора:

- розрахунок параметрів ПІД-регулятора за методикою Ніколса;
- оптимізація ПІД-регулятора;
- управління об'єктом;
- управління сервоприводом.

Структура Апаратно-програмного комплексу № 2

Апаратно-програмний комплекс № 2 (рис. 2) має наступну структуру (рис. 3):

- контролери;
- пристрої введення/виведення;
- зовнішні пристрої;
- операційні системи;
- мережеві протоколи;
- інтерфейси зв'язку з об'єктами і пристроями;
- бездротові інтерфейси;
- об'єкти управління.

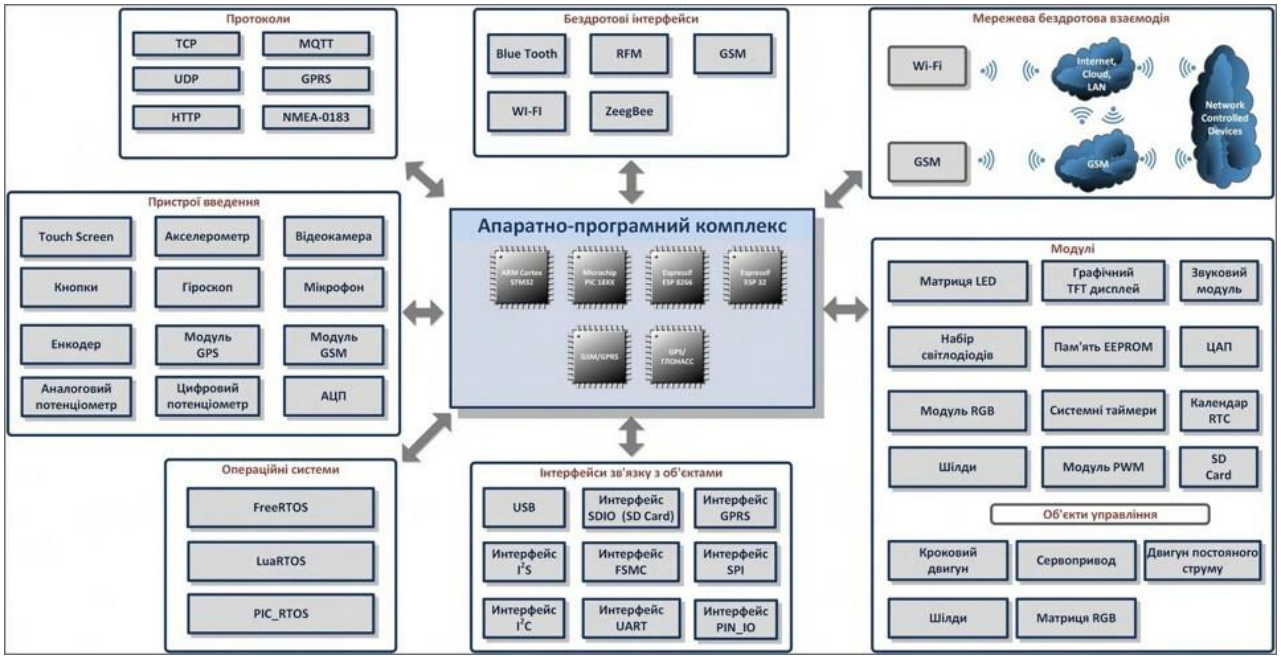


Рис. 3 - Структура Апаратно-програмного комплексу № 2

Склад апаратної частини комплексу № 2 (рис. 2)

Модулі контролерів:

- PIC 18F25K22;
- STM 32F103C8T6;
- ESP 32;
- ESP 8266 (рис. 4).

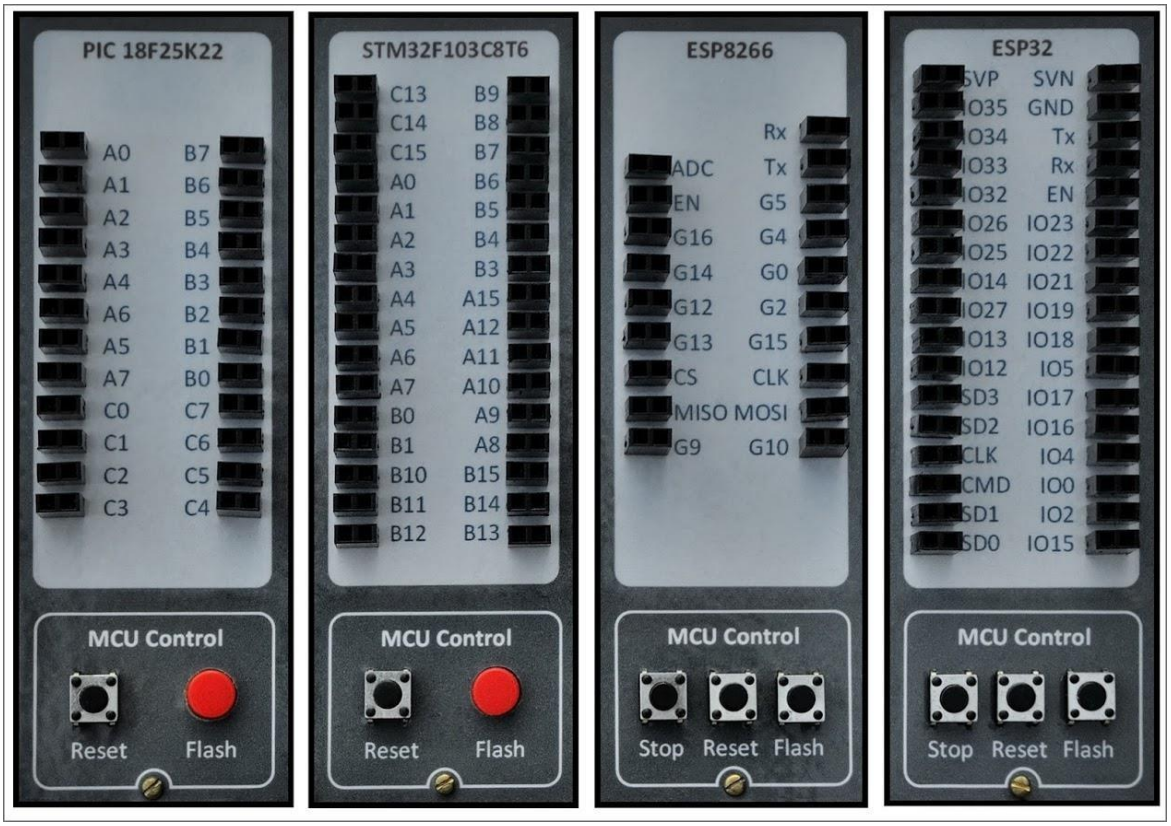


Рис. 4 - Змінні модулі контролерів

Передбачене встановлення будь-яких інших контролерів і програмованої логічної матриці. (ПЛМ).

Мережеві комунікаційні модулі:

- модуль Wi-Fi: ESP 8266;
- модуль Wi-Fi: ESP 32;
- модуль GSM: SIM800L.

Допоміжні модулі:

- гіроскоп: L3G4200D;
- акселерометр: ADXL345;
- модуль GPS: M8030 NEO-M8N;
- модуль SD Card (read/write).

Пристрої введення:

- TouchScreen SPI (TFT дисплей);
- аналоговий потенціометр;
- цифровий потенціометр: X9C103SZI;
- цифровий енкодер;
- набір кнопок;
- набір перемикачів;
- модуль UART;
- модуль Wi-Fi;
- модуль GSM;
- модуль GPS.

Пристрої виведення:

- TFT SPI дисплей;
- OLED I^2C serial дисплей;
- набір світлодіодів;
- набір RGB світлодіодів;
- матриця LED 8x8 SPI-UART;
- модуль Wi-Fi;
- модуль GSM.

Об'єкти управління:

- двигун постійного струму;
- кроковий двигун;
- сервопривід.

Розширення:

- Для розробки і/або підключення інших модулів передбачена макетна панель і клеми розширення.

Склад програмної частини комплексу № 2 (рис. 2)

Мережеві модулі:

- FTP Server, Client;
- HTTP Server, Client;
- MQTT Client;
- Websocket Client;
- Redis Client;
- Net (UDP, TCP);

- CJSON;
- CoAP;
- IMAP (e-mail.);
- WiFi (Station, Access Point);
- WiFi Monitor;
- Sqlite3 (SQL);
- Crypto;
- TLS.

Бібліотеки та драйвери для:

- управління апаратними модулями комплексу і зовнішніми шилдами;
- управління модулем GPS / ГЛОНАСС;
- управління модулем GSM;
- бібліотека графічного інтерфейсу з елементами дискретного і пропорційного управління.

Інші бібліотеки і драйвери:

- При необхідності легко встановлюються/використовуються необхідні бібліотеки і драйвери.

АПАРАТНО-ПРОГРАМНИЙ КОМПЛЕКС № 3
«INTERNET OF THINGS DEVELOPMENT BOARD»

З метою ефективного вивчення дисципліни «Internet Of Things» і підготовки фахівців в області технології IoT, авторами розроблений, виготовлений і впроваджений у навчальний процес спеціальний апаратно-програмний комплекс «Internet Of Things development board» (рис.5).

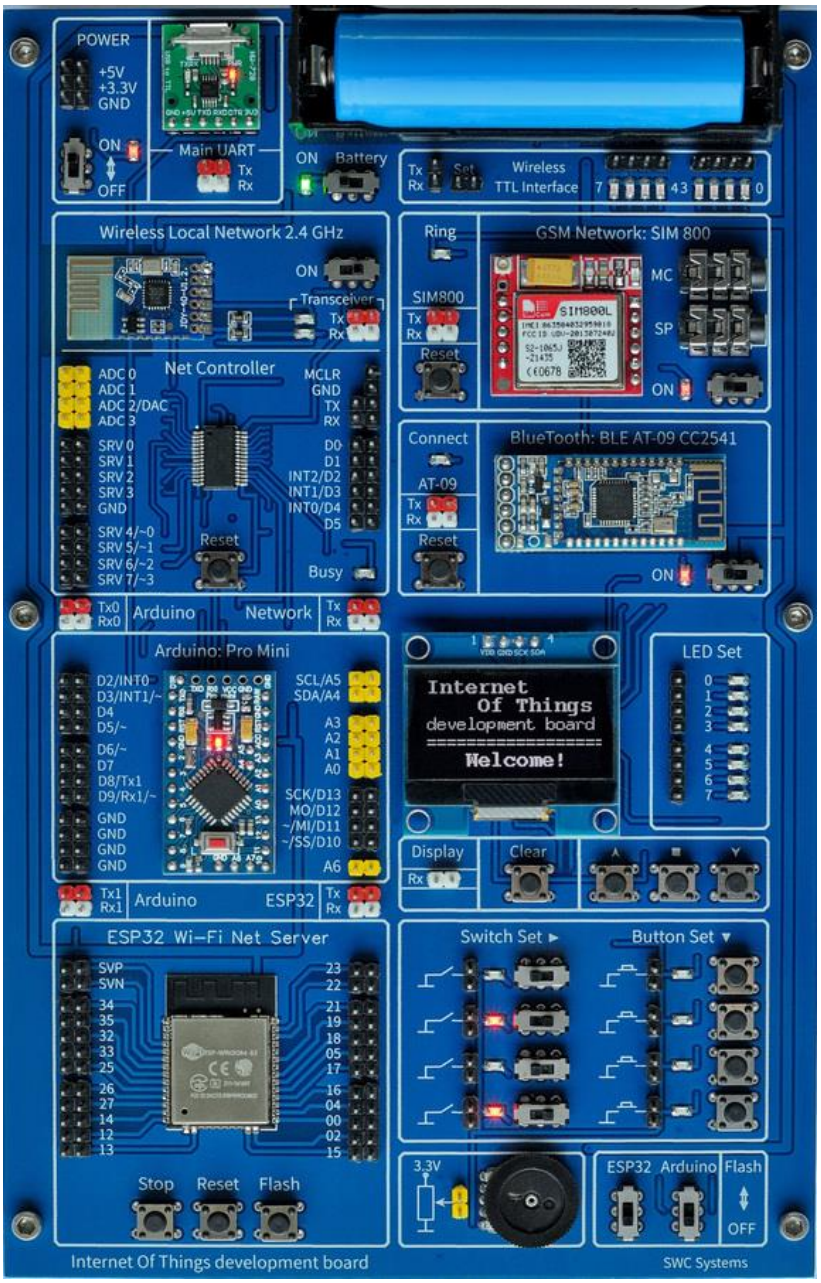


Рисунок 5 – Апаратно-програмний комплекс № 3
«Internet Of Things development board»

Склад навчального комплексу «Internet Of Things development board»:

1. Контролер / Координатор / адаптивної бездротової локальної мережі 2.4 GHz.

2. Контролер мережі Internet / Wi-Fi Access Point / Wi-Fi Station / Wi-Fi Server /: ESP-32.

3. Контролер мережі GSM / GPRS / DPMF / ...: SIM800L.

4. Контролер BlueTooth / BLE 4.0: AT-09 / HM-10.

5. Контролер загального призначення Atmel 328P (Arduino).

6. Контролер бездротового TTL інтерфейсу.

7. Контролер дисплея ESP8266 / Wi-Fi Direct / + Дисплей I^2C 128x64.

8. Набори допоміжних елементів: потенціометр, кнопки, тумблери, світлодіоди.

Розроблене програмне забезпечення (ПЗ) комплексу включає в себе:

1. ПЗ бездротової локальної мережі 2.4 GHz..

2. ПЗ сервера мережі на базі платформи NodeMCU для ESP-32.

3. Драйвер контролера мережі GSM.

4. Драйвер контролера бездротового TTL інтерфейсу.

5. Драйвер контролера дисплея (мова програмування Lua).

У контролер загального призначення Atmel 328P завантажуються як готові скетчі, так і скетчі, створені студентами самостійно в рамках лабораторних робіт з різних дисциплін.

Виводи усіх контролерів доступні студентам для розробки і макетування різних IoT пристроїв і систем.

При необхідності до комплексу підключаються зовнішні макетні плати з набором необхідних модулів і компонентів.

Особливістю комплексу є вбудований модуль управління адаптивною локальною бездротовою мережею, розроблений в рамках науково-технічної теми № 0120U104088 «Створення мобільної мережі 2.4 GHz з адаптивною аморфною топологією для управління роями БПЛА і робототехнічних об'єктів».

За допомогою навчального комплексу «Internet Of Things development board» студенти отримують реальну можливість створювати, моделювати, макетувати і програмувати різні IoT пристрої (в тому числі і робототехнічні пристрої та системи), а також пристрої для територіально-розподілених систем управління та збору інформації (рис. 6).

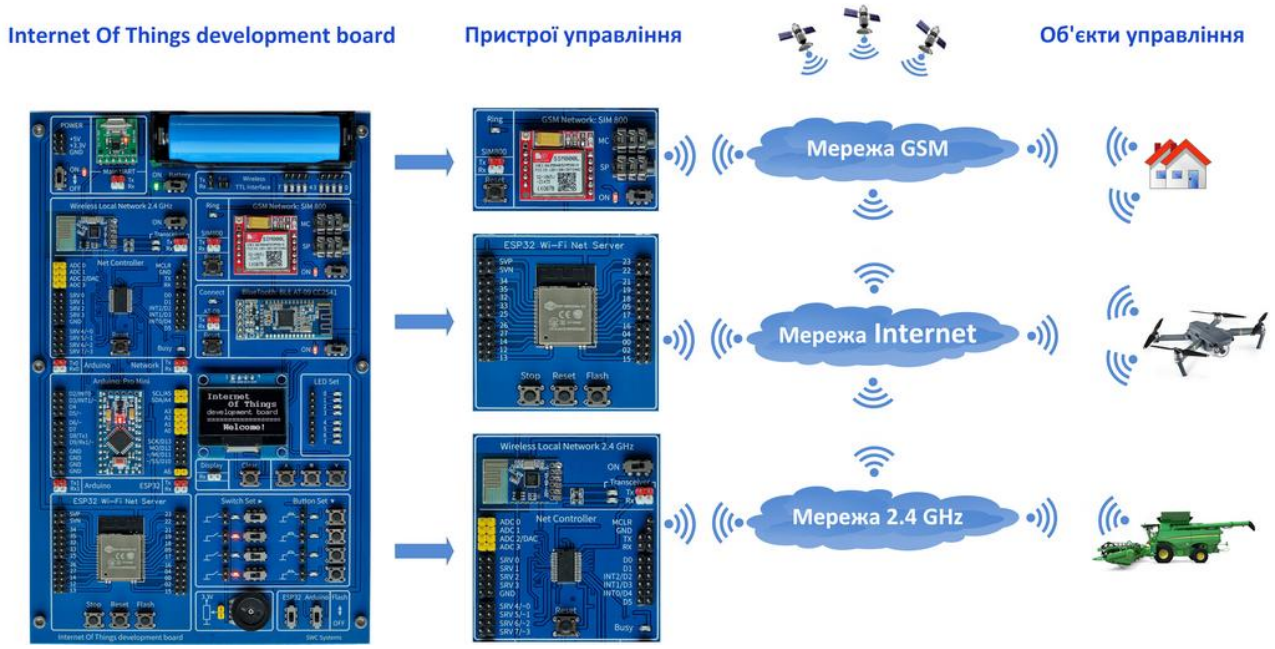


Рисунок 6 – Технологія IoT в навчальному процесі

СИСТЕМА АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ PROTEUS

Proteus Design Suite – пакет програм для автоматизованого проектування (САПР) електронних схем. Розробка компанії Labcenter Electronics (Великобританія).

Пакет являє собою систему схемотехнічного моделювання, що базується на основі моделей електронних компонентів, прийнятих у PSpice (Personal Simulation Program with Integrated Circuit Emphasis – програма симуляції аналогової і цифрової логіки).

Відмінною рисою пакета **PROTEUS VSM** є можливість моделювання роботи програмованих пристроїв: мікроконтролерів, мікропроцесорів, DSP NF та ін. Бібліотека компонентів містить довідкові дані. Додатково у пакет PROTEUS VSM входить система проектування друкованих плат.

Пакет Proteus складається з двох частин, двох підпрограм:

- **ISIS** – програма синтезу та моделювання безпосередньо електронних схем;
- **ARES** – програма розробки друкованих плат.

Разом з програмою встановлюється набір демонстраційних проектів для ознайомлення.

Також до складу пакету Proteus входить середовище розробки VSM Studio, що дозволяє швидко написати програму для мікроконтролера, використовуваного у проекті, NF скомпілювати.

Пакет є комерційним. Безкоштовна ознайомча версія характеризується повною функціональністю, але не має можливості збереження файлів.

Примітною особливістю є те, що у ARES можна побачити 3D-модель друкованої плати, що дозволяє розробнику оцінити свій пристрій ще на стадії розробки.

Система підтримує підключення нових елементів (SPICE) і підключення різних компіляторів (PICOLO, ARM-подібні, AVR і т.д.).

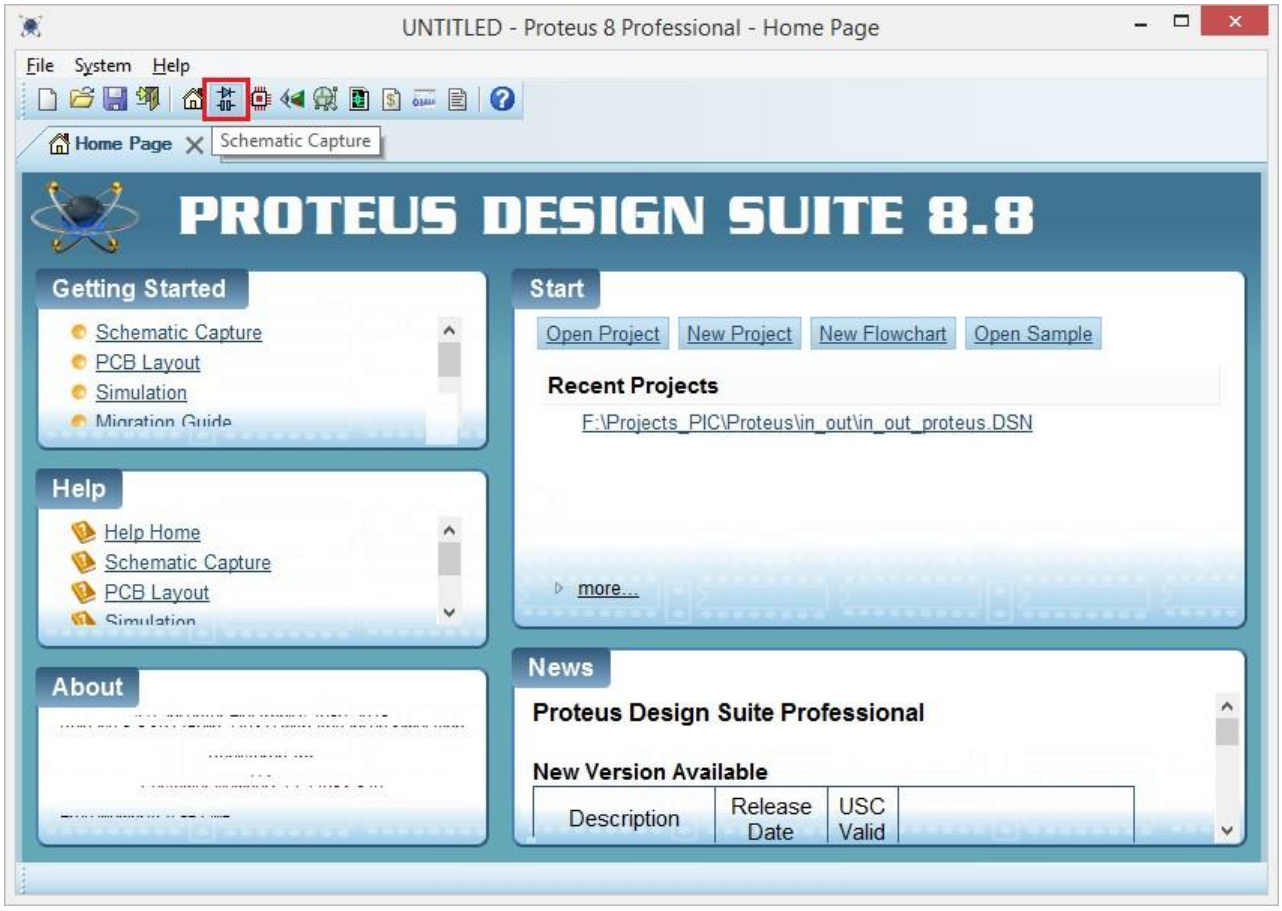


Рисунок 1 – Середовище моделювання «Proteus»

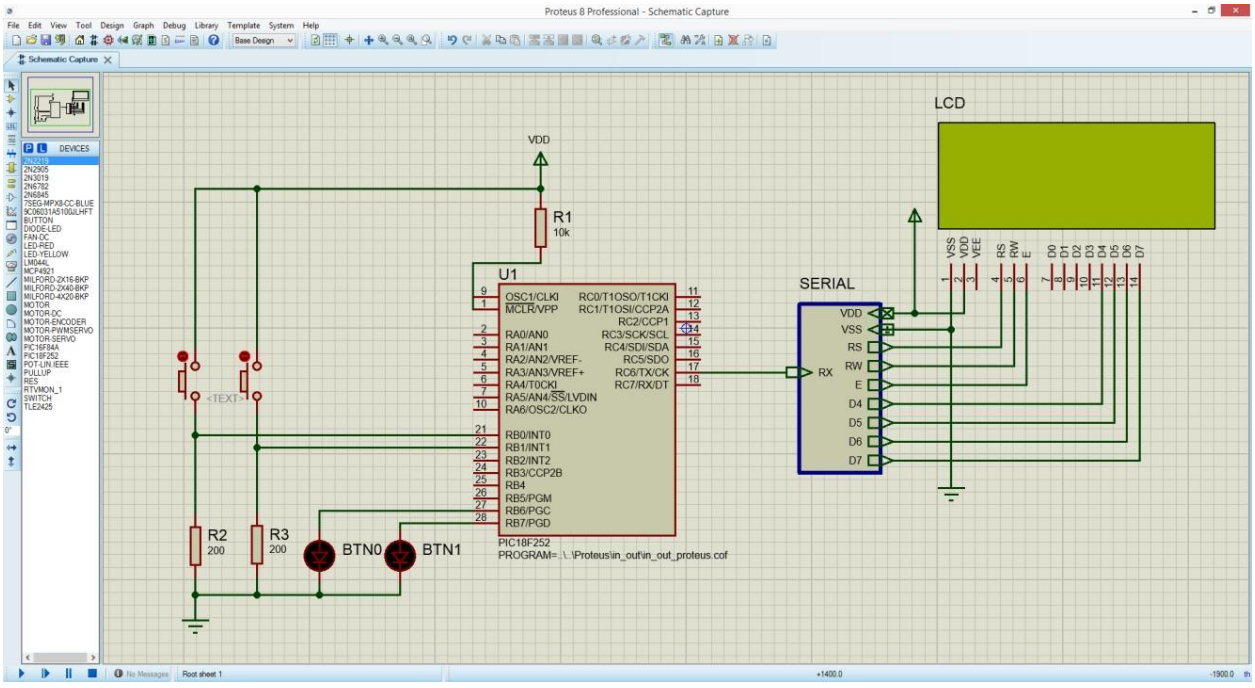


Рисунок 2 – Виконання лабораторної роботи
у середовищі моделювання «Proteus»

Лабораторна робота №1

Тема: Введення-виведення дискретних сигналів

Ціль роботи

Одержання навичок роботи з IDE PSWH, програмування мікроконтролерів, написання і налагодження програми введення-виведення дискретних сигналів.

Завдання для варіанту «Навчальний комплекс»:

- Намалювати принципову схему підключень відповідно до варіанта.
- Для варіанту «Proteus» використовувати готову схему
- Здійснити введення дискретних сигналів із кнопок і тумблерів стенда відповідно до варіанта.
- Здійснити виведення результатів на LCD і на світлодіоди. Виведення на LCD повинне містити:

Групу, прізвище та ініціали студента, вивід контролера, по якому у цей момент здійснюється читання сигналу.

Теоретичні відомості:

Розробка програмного забезпечення

Розробка програмного забезпечення для мікроконтролера здійснюється мовою програмування С у інтегрованому середовищу розробки PCWH фірми CCS.

Порядок роботи для варіанту «Навчальний комплекс»:

1. Створити свій директорій для файлів проекту.
2. Створити проект в інтегрованому середовищі розробки PCWH (c:\Temp\PICC файл PCW.exe).
3. Написати програму мовою програмування С.
4. Скомпілювати програму, одержати двійковий файл *.HEX.

5. Записати *.HEX – файл у пам'ять програм мікроконтролера програмою PICkit.exe

Загальна схема процесу розробки для варіанту «Навчальний комплекс» виглядає в такий спосіб:

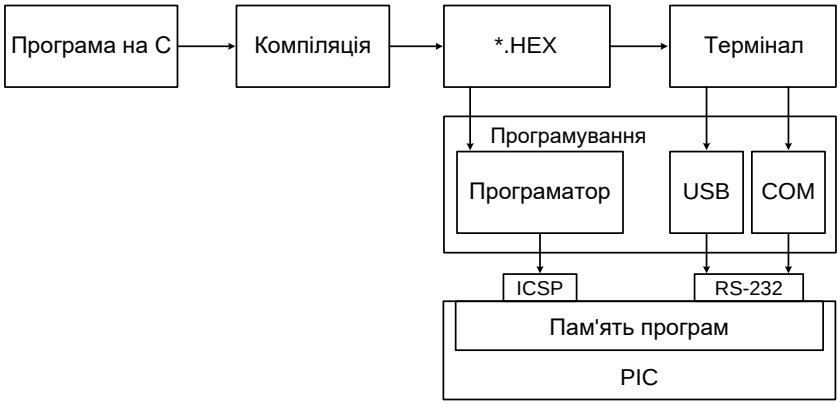


Рисунок 1 - Процес розробки програм для варіанту «Навчальний комплекс»

Порядок роботи для варіанту «Proteus»:

1. Завантажити з сервера папку «Proteus_students».
2. Відкрити середовище розробки PCWH.
3. У папці «Proteus_students» вибрати папку відповідної лабораторної роботи (LAB1... LAB8).
4. Відкрити файл проекту з розширенням *. Pjt.
5. У редакторі відкрити шаблон файлу програми з розширенням *. C.
6. Відкрити середу моделювання «Proteus».
7. У папці «Proteus_students» вибрати папку відповідної лабораторної роботи (LAB1... LAB8).
8. Відкрити файл проекту з розширенням ***. DSN**.
9. У середовищі розробки PCWH , використовуючи шаблон програми самостійно написати , відкомпілювати програму у відповідності з завданням.
10. У середовищі розробки «Proteus» виконати програму.

Примітка: Файл демонстрацій виконання лабораторної роботи розташований в папці «Demo».

Загальна схема процесу розробки для варіанту «Proteus» виглядає наступний чином:

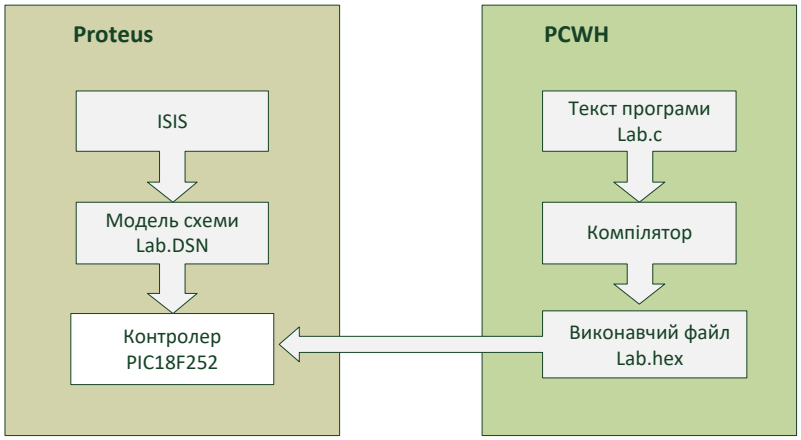
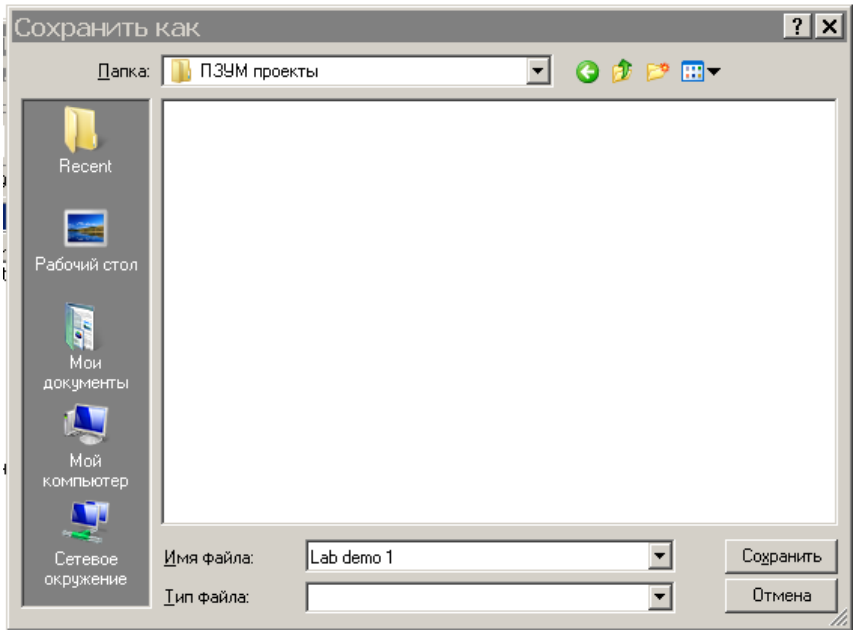


Рисунок 1.1 - Процес розробки програм для варіанту «Proteus»

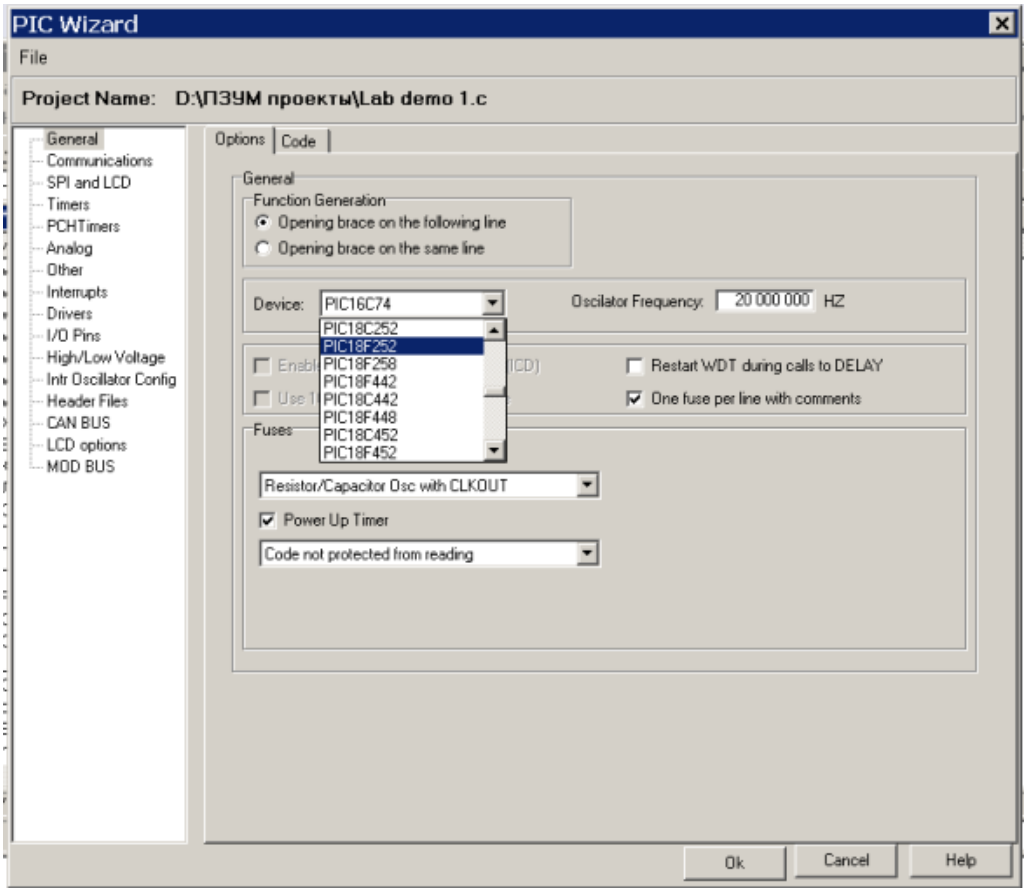
Робота з IDE PCWH

Створення проекту.

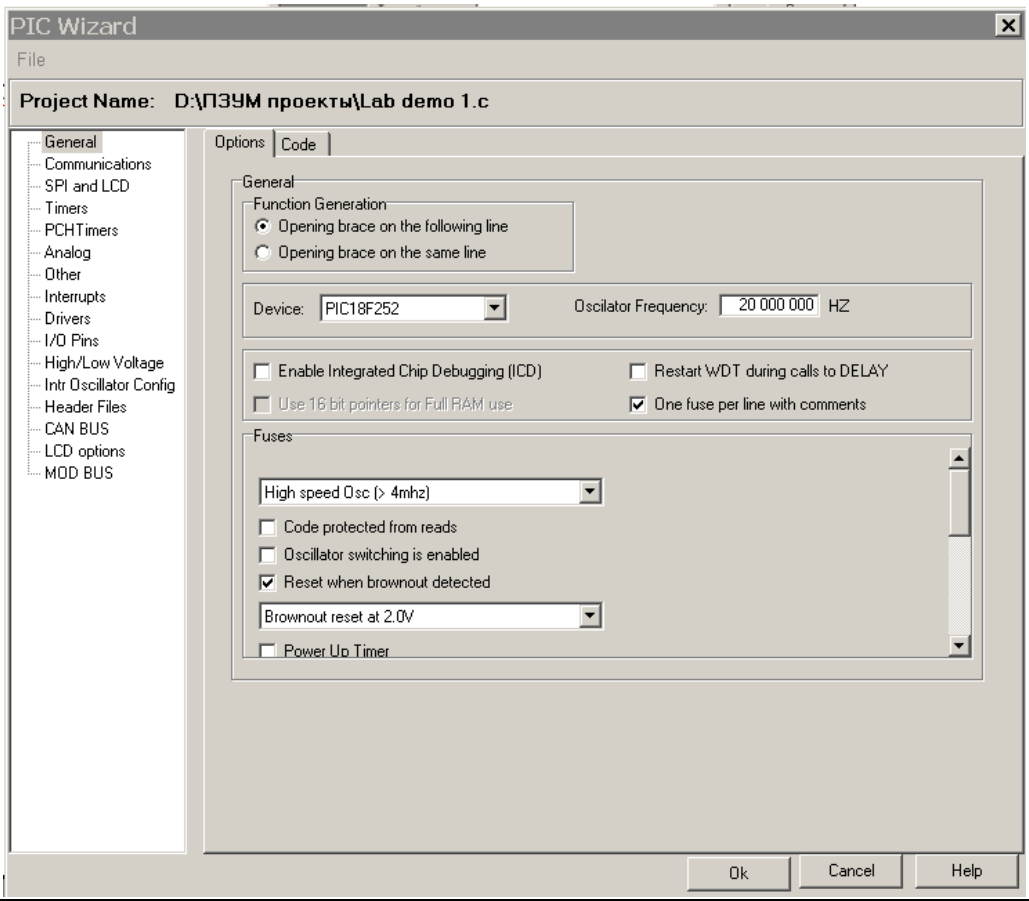
Запустити IDE, нажати кнопку WIZARD. Вибрати свій директорій і записати в нього проект під будь-яким іменем латинським шрифтом:



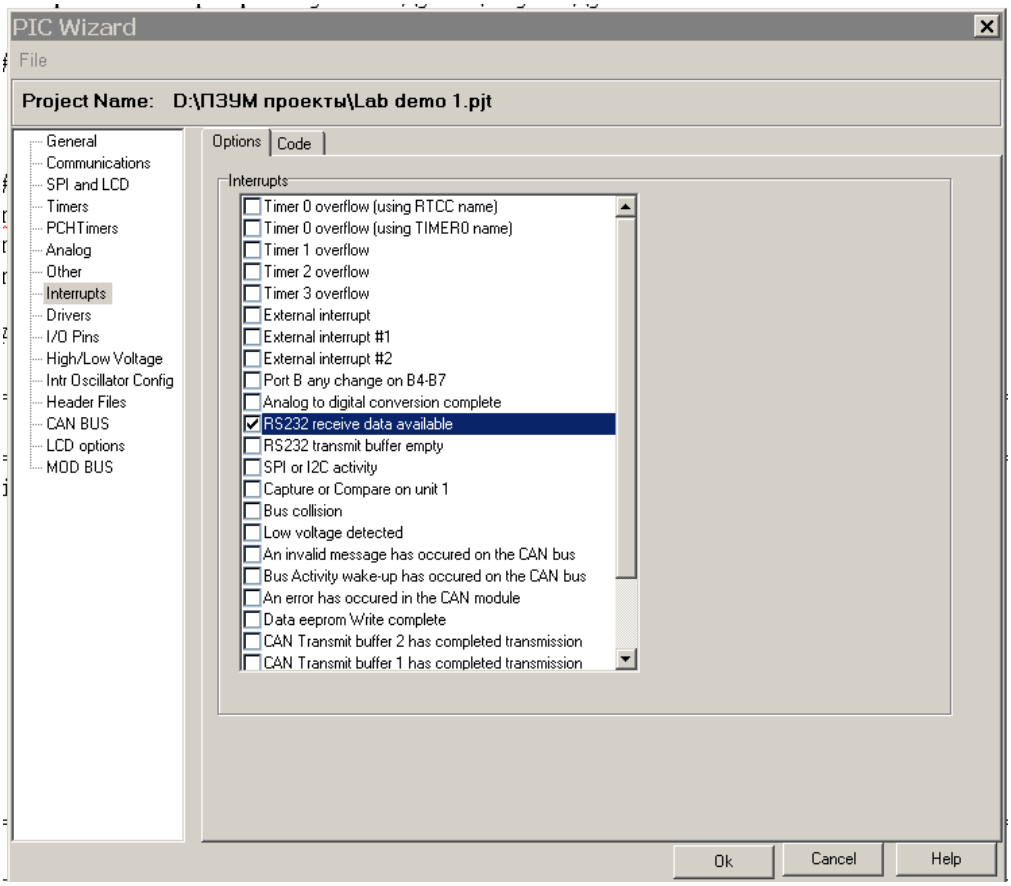
У вікні Device вибрати тип контролера **PIC18F252:**



У вікні Fuses вибрати тип генератора High speed Osc:



У лівому вікні в пункті Interrupts встановити мітку в поле переривання, яке передбачається обробляти (наприклад RS-232) або нічого не встановлювати, якщо не передбачається обробка переривань:



Нажати кнопку Ok. Проект готовий, можна приступати до написання програми. Буде згенерированний наступний код:

```
#include "D:\ПЗУМ проекти\Lab1.h"
#int_RDA
RDA_isr()
{
}
void main()
{
   setup_adc_ports(NO_ANALOGS);
   setup_adc(ADC_OFF);
   setup_spi(FALSE);
   setup_wdt(WDT_OFF);
   setup_timer_0(RTCC_INTERNAL);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
   enable_interrupts(INT_RDA);
   enable_interrupts(GLOBAL);
   setup_oscillator(False);

   // TODO: USER CODE!!
}
```

Рекомендації:

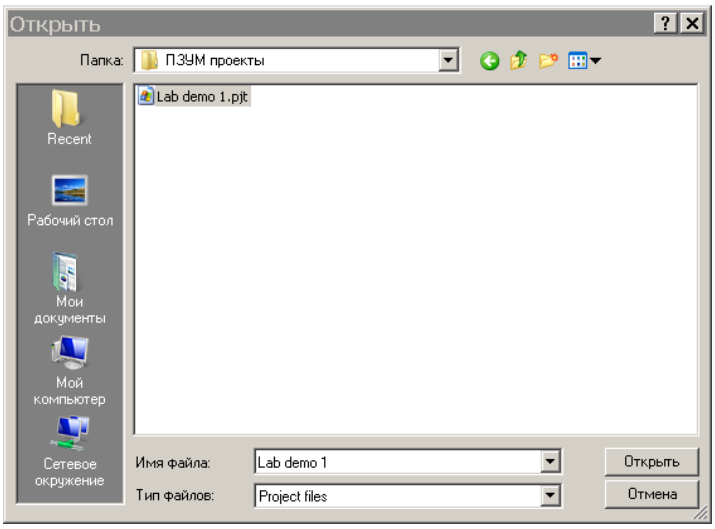
У файлі *.h рядок **#FUSES LVP** замінити на рядок **#FUSES NOLVP**

Привести програму до наступного вигляду і використовувати як шаблон:

```
//###################################################################
// Найменування програми
// file:  file_name.c
// Copyright (c) Docent Smirnova N.V. Kropyvnyts'kyy
//###################################################################
#include "Lab demo 1.h"
#include <swc_LCD.h>          //драйвер LCD дисплея
#CASE                          //враховувати регістр символів

//====================================================================
// Ініціалізація PIC
//====================================================================
void init()
{
   setup_adc_ports(NO_ANALOGS);
   setup_adc(ADC_OFF);
   setup_spi(FALSE);
   setup_wdt(WDT_OFF);
   setup_timer_0(RTCC_INTERNAL);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
   enable_interrupts(INT_RDA);
   enable_interrupts(GLOBAL);
   setup_oscillator(FALSE);
}
//====================================================================
// Оброблювач переривання RS-232
//====================================================================
#int_RDA
void RDA_isr()
{
}
//====================================================================
// Main
//====================================================================
void main()
{
   init();        //ініціалізація контролера
   lcd_init();    //ініціалізація LCD дисплея

   // Тут можна писати текст програми (виклики функцій)
}
```

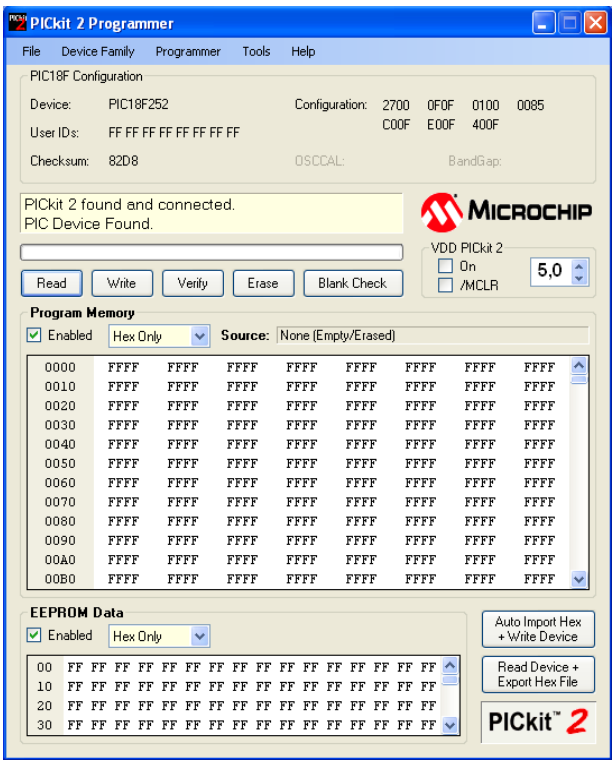
Компіляція програми здійснюється натисканням кнопки **F9.**

Відкрити проект надалі,можна нажавши кнопку Project:



Програмування контролера

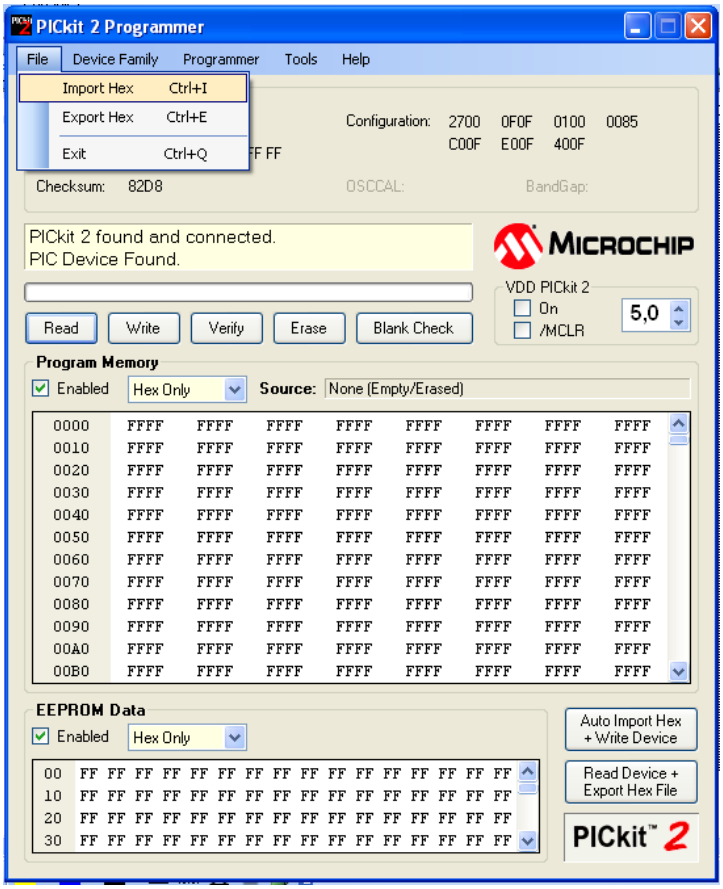
Контролер програмується через порт USB за допомогою програми PICkit.

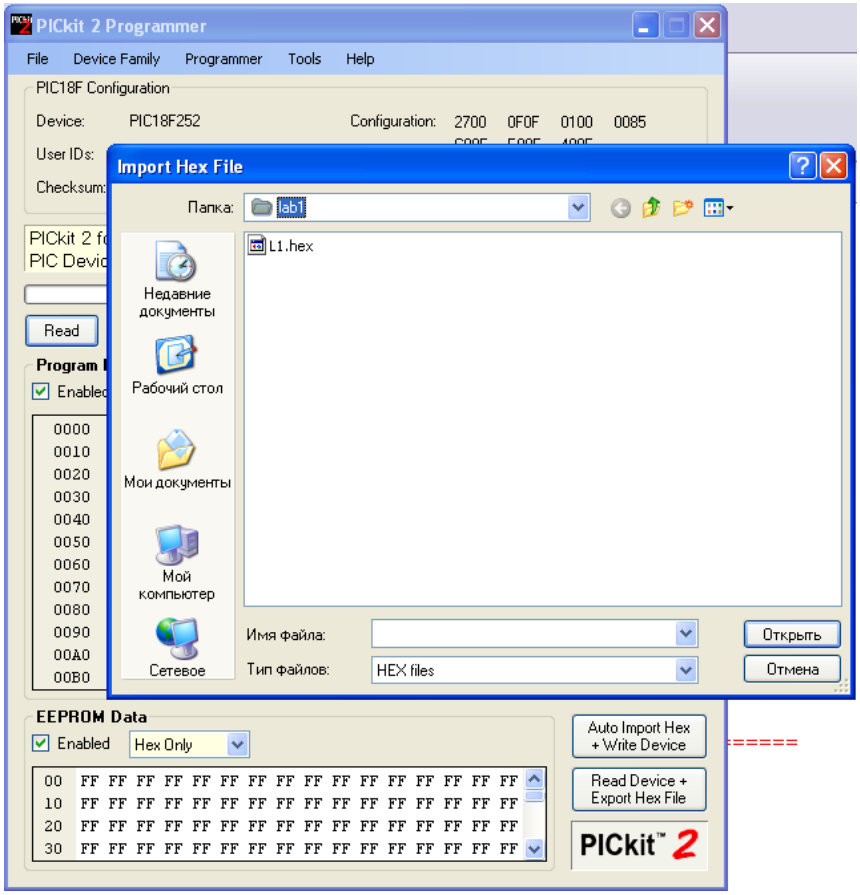


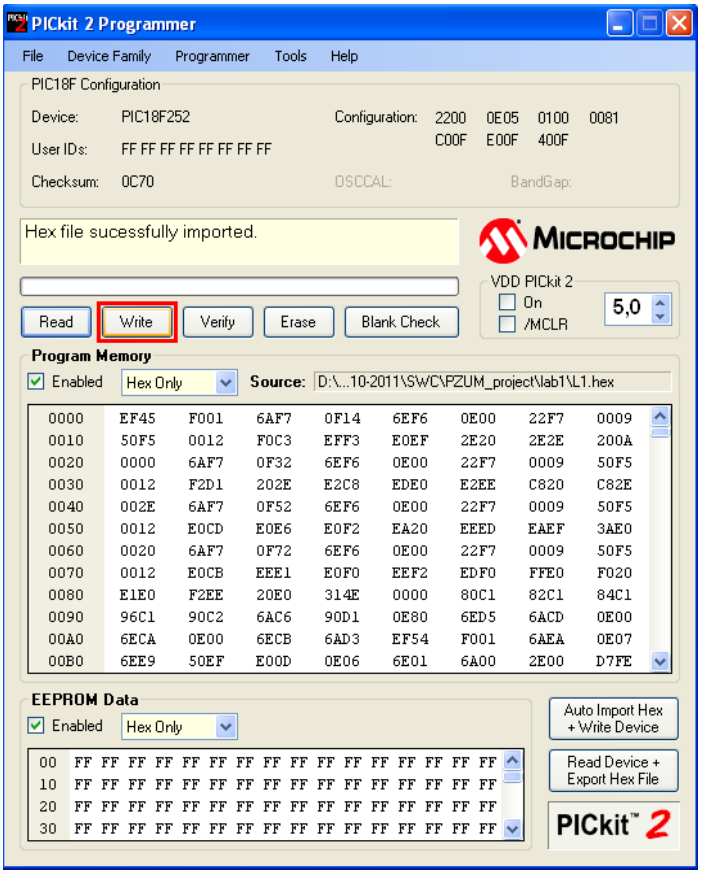
На стенді нажати кнопку **Pgm.**

Мікроконтролер готовий до приймання і завантаженню програми.

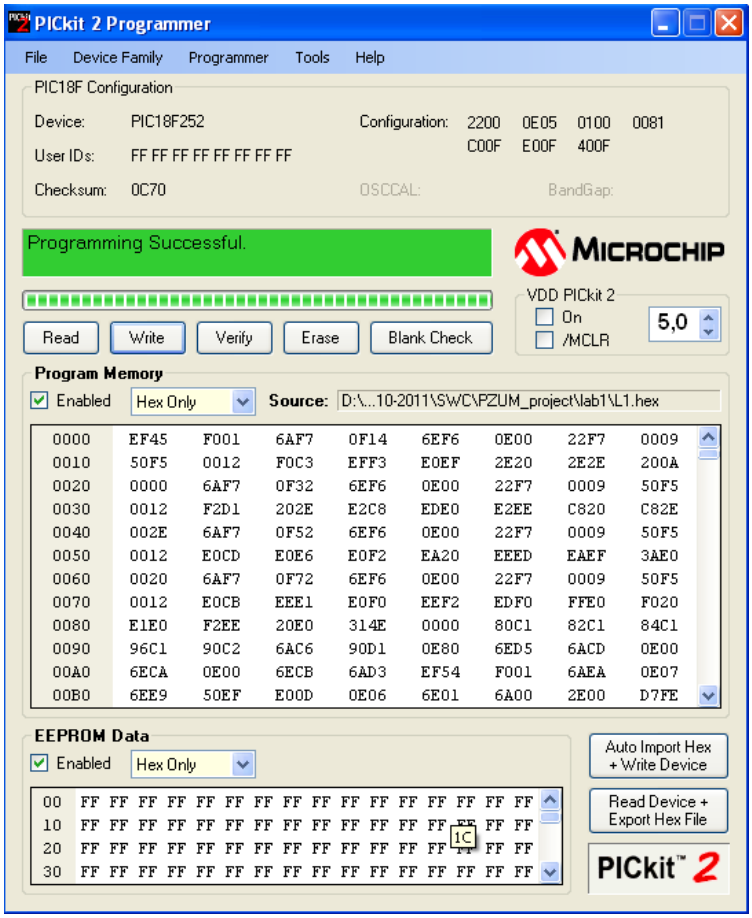
Потім у меню File | Import HEX *(ctrl+I)* вибрати *hex - файл, запустити запис програми в мікроконтролер і нажати кнопку Write:


PICkit 2 Programmer
File
Device Family
Programmer
Tools
Help
Import Hex Ctrl+I
Export Hex Ctrl+E
Exit Ctrl+Q
Configuration: 2700 0F0F 0100 0085
C00F E00F 400F
Checksum: 82D8
OSCCAL:
BandGap:
PICkit 2 found and connected.
PIC Device Found.
MICROCHIP
VDD PICkit 2
On
/MCLR
5,0
Read
Write
Verify
Erase
Blank Check
Program Memory
Enabled
Hex Only
Source: None (Empty/Erased)
EEPROM Data
Auto Import Hex + Write Device
Read Device + Export Hex File
PICkit 2


PICkit 2 Programmer
File
Device Family
Programmer
Tools
Help
PIC18F Configuration
Device: PIC18F252
Configuration: 2700 0F0F 0100 0085
User IDs:
Checksum:
Import Hex File
Папка: lab1
L1.hex
Недавние документы
Рабочий стол
Мои документы
Мой компьютер
Сетевое
Имя файла:
Тип файлов: HEX files
Открыть
Отмена
Read
EEPROM Data
Enabled
Hex Only
Auto Import Hex + Write Device
Read Device + Export Hex File
PICkit 2



При успішному записі програми з'явиться повідомлення:



Після завантаження програми на стенді нажати кнопки **Pgm,** потім **Reset** і вона відразу починає виконуватися.

Примітка

Всі лабораторні роботи можуть бути виконані і налагоджені в програмі-симуляторі "Proteus", з подальшою демонстрацією на навчальному комплексі.

Довідка:

Дискретними сигналами називаються сигнали, що змінюють свій рівень стрибком, без проміжних станів.

Активний рівень сигналу із кнопок і тумблерів стенда – логічний “0”.

Активний рівень для запалювання світлодіоду – логічна “1”.

Режим введення-виведення встановлюється функцією set_tris_x(val).

Шаблон розробки принципової схеми для лабораторних робіт:

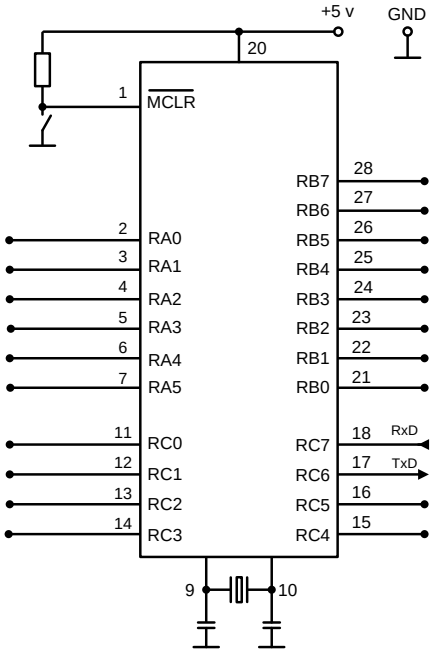


Рисунок 2 – Шаблон розробки принципової схеми для лабораторних робіт

Функції введення-виведення даних

Мікроконтролер може зчитувати дані із зовнішнього пристрою в порти і записувати дані у зовнішній пристрій через порт A, B і C.

Напрямок передачі даних визначається функціями:

```
set_tris_x(config word)
```

для передачі даних необхідно біт порту A-C (pin) встановити у ‘0’, для приймання – у ‘1’.

Наприклад: потрібен порт B встановити у наступний режим: розряди 0-3 читання даних, розряди 4-7 – виведення даних.

```
set_tris_b(0b00001111); //розташування розрядів: B7 B6 B5 B4 B3 B2 B1 B0
```

або:

```
set_tris_b(0x0F);       //розташування розрядів: B7 B6 B5 B4 B3 B2 B1 B0
```

Аналогічно для інших портів.

Читання даних з порту здійснюється функціями:

```
input_x()
```

Наприклад, прочитати значення сигналів на вході порту A:

```
int8 val;
val = input_a();
```

Аналогічно для інших портів.

Виведення даних у порт здійснюється функцією:

```
output_x()
```

Наприклад, вивести в порт B значення 0x55:

```
int8 val;
val = 0x55;
output_b(val);
```

Аналогічно для інших портів.

Операції з бітами портів здійснюється функціями:

```
output_high(PIN);
output_low(PIN);
```

Приклад: Встановити розряди 0, 3 порту B у стан “1”, а розряди 2, 4 порту A у стан “0”:

```
output_high(PIN_B0);
output_high(PIN_B3);
output_low(PIN_A2);
output_low(PIN_A4);
```

Примітка: активний рівень кнопок і тумблерів навчального комплексу – логічний ‘0’.

Активний рівень для роботи світлодіодів – логічна ‘1’.

Виведення даних на дисплей

У процесі роботи програми необхідне виведення на дисплей налагоджувальної інформації і результату виконання програми. Інформація може бути виведена у вікно термінала на PC і на LCD дисплей стенда (4 рядка по 20 символів). Команди виведення у вікно термінала стандартні:

```
putc();
puts();
printf();
```

Команди читання даних стандартні:

```
getc();
gets();
```

Команди для роботи з LCD дисплеєм:

```
void lcd_putc(int8 ch);                          //вивести символ
void printf();                                    //вивести форматований рядок
void lcd_goto(int8 line,int8 pos);               //перейти до рядка й знакоместу
void lcd_putc(int8 line, int8 pos, int8 ch); //вивести символ у координатах
void lcd_clear_line(int8 line);                  //почистити рядок
void lcd_clear();                                //почистити всі рядки
void lcd_cursor_on();                            //включити курсор
void lcd_cursor_off();                           //виключити курсор
void lcd_bs();                                   //backspace
void lcd_cursor_left();                          //курсор вліво
void lcd_cursor_right();                         //курсор вправо
void lcd_cursor_up();                            //курсор нагору
void lcd_cursor_down();                          //курсор вниз
void lcd_running_string_start();         //запустити виведення рядка, що біжить
void lcd_running_string_stop();          //зупинити виведення рядка, що біжить
```

Приклади:

Виведення рядка, що біжить:

```
lcd_clear();
lcd_running_string_begin(4);        //початок повідомлення у рядку 4
printf("Програмування мікроконтролерних систем");
lcd_running_string_end();           //кінець повідомлення
lcd_running_string_start();         //запустити рядок, що біжить
```

Виведення символу ‘Q' у другому рядку у п'ятому знакомісці:

```
lcd_putc(2,5,'Q');
```

Виведення повідомлення у другому рядку із третього знакомісця:

```
lcd_goto(2,3);
printf("Hello! How do You do?");
```

Виведення повідомлення на кілька рядків:

```
printf("1234567890\n 1234567890\n");
```

Принципова схема з'єднань для виконання лабораторної роботи №1 представлена на рис. 3.

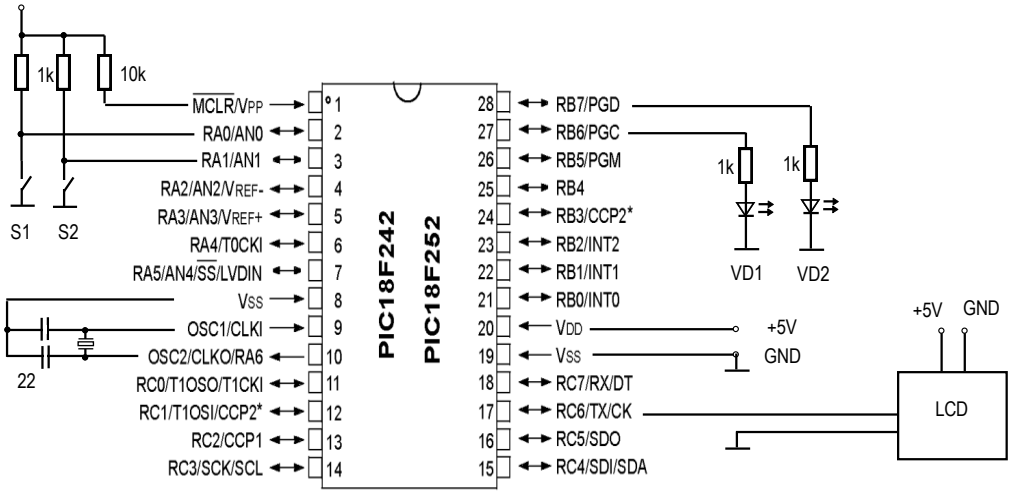


Рисунок 3 – Повна принципова схема з'єднань для виконання лабораторної роботи №1

Приклад комутації з'єднань навчального комплексу:

Рисунок 3 – Комутації з'єднань навчального комплексу для лабораторної роботи №1

Приклад побудови та оформлення програми:

```
//#################################################################
// Найменування програми
// file:  file_name.c
// Copyright (c) Docent Smirnova N.V. Kropyvnyts'kyy
//#################################################################
#include "Lab1.h"
#include <swc_LCD.h>       //драйвер LCD дисплея
#CASE                      //враховувати регістр символів
//================================================================================
// Назва функції (Функція, що викликається з функції main())
//================================================================================
void init()
{
            …
}
//================================================================================
// Назва функції (Функція, що викликається з функції function_1())
//================================================================================
void function_1_1()
{
            …
}
//================================================================================
// Назва функції (Функція, що викликається з функції main())
//================================================================================
void function_1()
{
            …
      function_1_1();
      …
}
//================================================================================
// Назва функції (Функція, що викликається з функції main())
//================================================================================
void function_2()
{
            …
}
//================================================================================
// Майн
//================================================================================
void main()
{
      init();           //ініціалізація
            …
      function_1();     //виклик функції
      function_2();     //виклик функції
}
```

Приклад реалізації програми лабораторної роботи №1

(розробка, програмування, оформлення лістингу)

1. Принципова схема підключень для варіанту «Навчальний комплекс» повинна виглядати у такий спосіб:

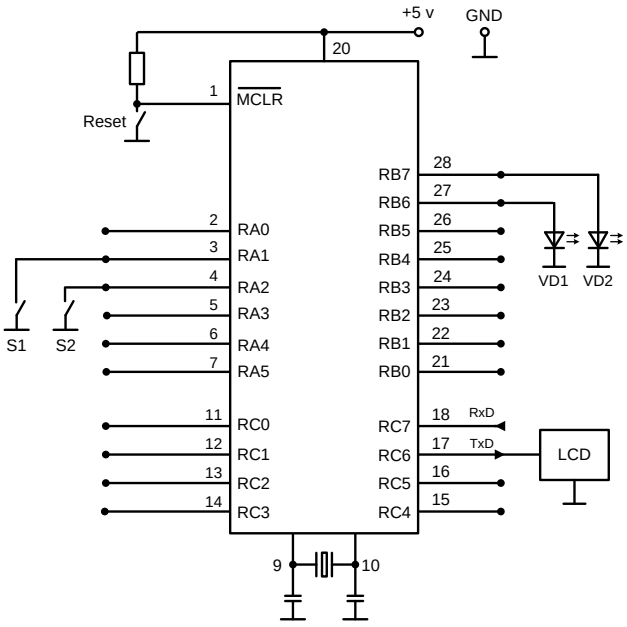


Рисунок 4 – Принципова схема підключень для лабораторної роботи №1 для варіанту «Навчальний комплекс»

1. Принципова схема підключень для варіанту «Proteus» повинна виглядати у такий спосіб:

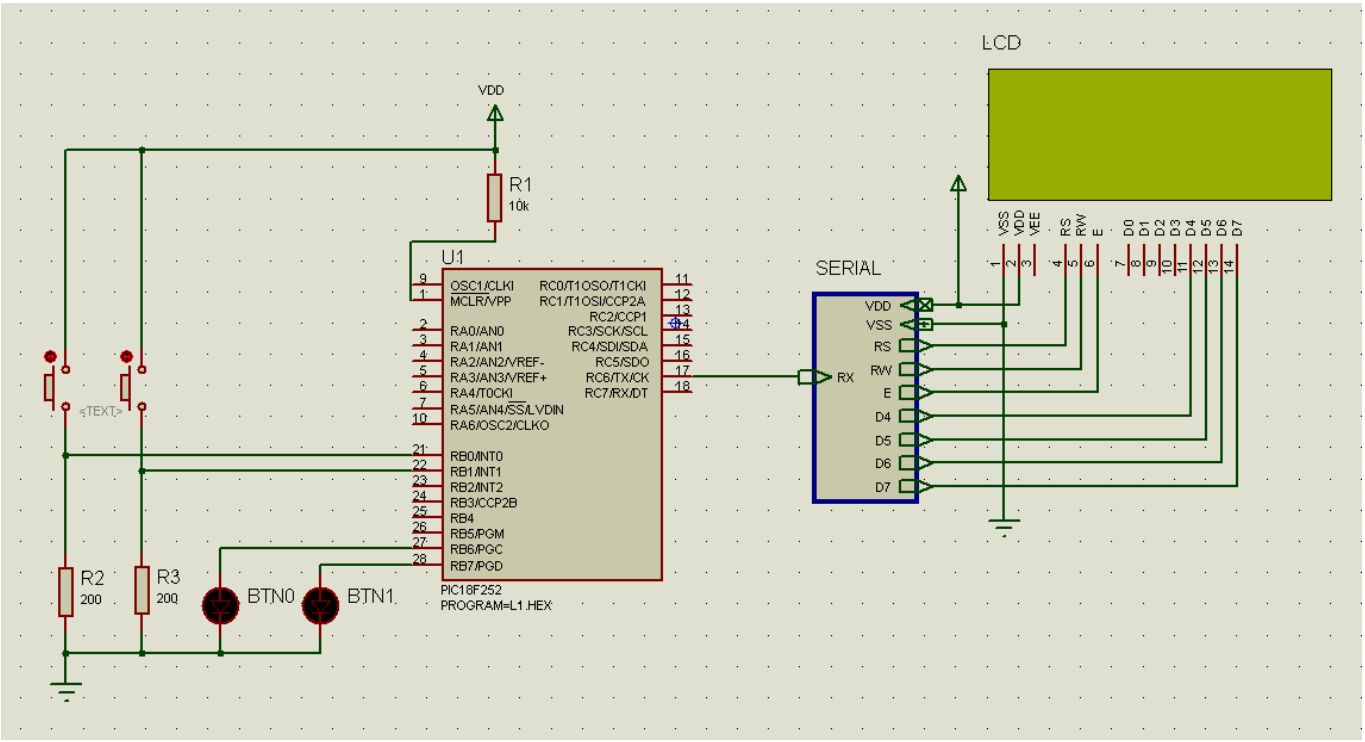


Рисунок 2.1 - Принципова схема для виконання лабораторної роботи для варіанту «Proteus»

Результат виконання програми:

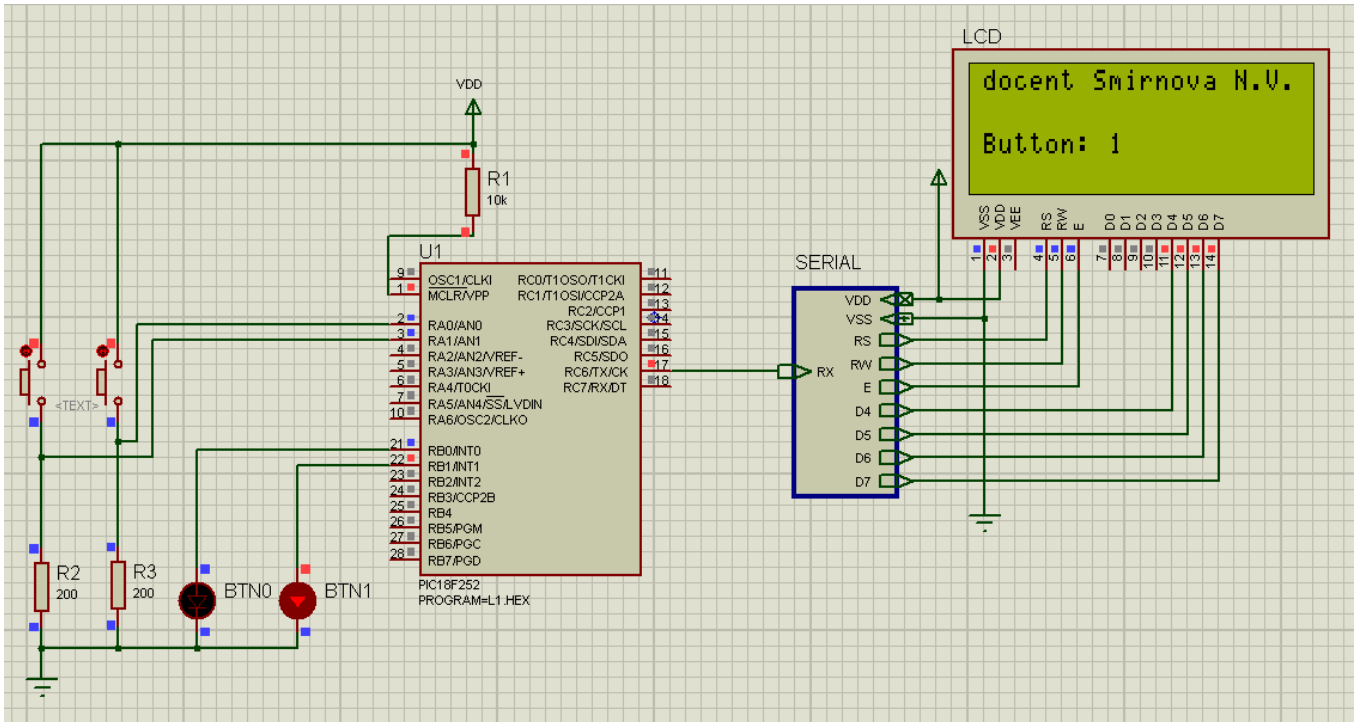


Рисунок 2.2 - Результат виконання лабораторної роботи для варіанту «Proteus»

2. Алгоритм програми для лабораторної роботи №1 повинен мати вигляд:

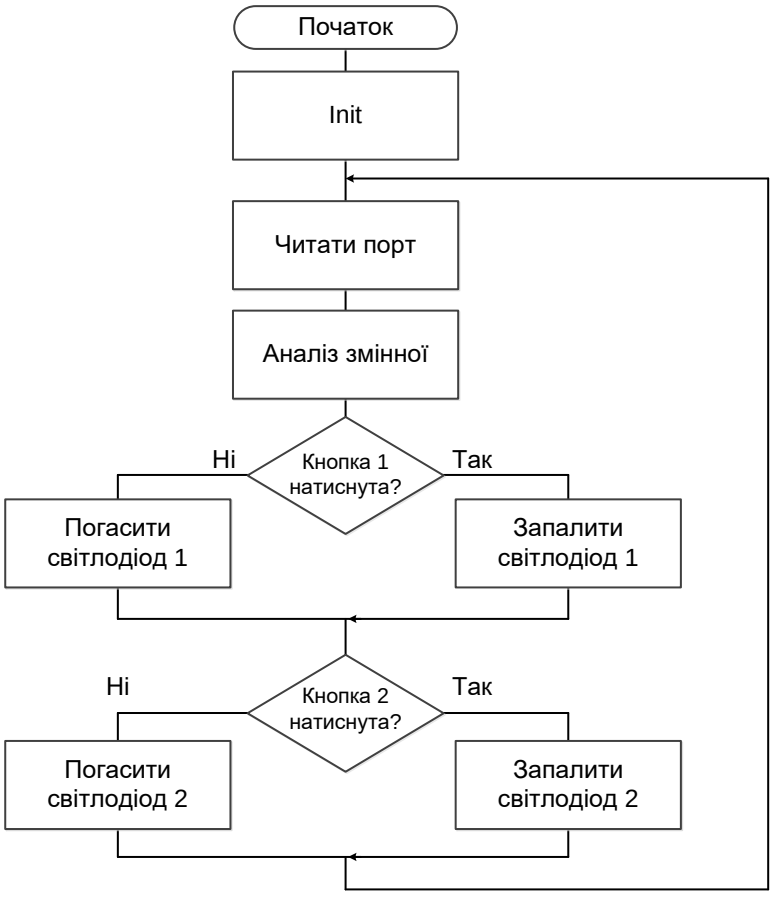


Рисунок 4 – Алгоритм 1 програми лабораторної роботи №1

3. Програма повинна мати вигляд і структуру відповідно до лістингу 1.

Лістинг 1 – Програма реалізації алгоритму 1 (рис. 4) лабораторної роботи №1 для варіанту «Навчальний комплекс»

```
//###################################################################
// Найменування програми
// file:  file_name.
// Copyright (c) Docent Smirnova N.V. Kropyvnyts'kyy
//###################################################################

#include "L1.h"
#include <swc_LCD.h>       // драйвер LCD дисплея
#CASE                      // враховувати регістр символів

//=========================================================================
// Ініціалізація PIC
//=========================================================================
void init()
{
   setup_adc_ports(NO_ANALOGS);
   setup_adc(ADC_OFF);
   setup_spi(FALSE);
   setup_wdt(WDT_OFF);
   setup_timer_0(RTCC_INTERNAL);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_oscillator(FALSE);
}
//=========================================================================
// Запалити світлодіод 1
//=========================================================================
void blink1()
{
   output_high(PIN_B6);
   delay_ms(50);
   output_low(PIN_B6);
   delay_ms(50);
}
//=========================================================================
// Запалити світлодіод 2
//=========================================================================
void blink2()
{
   output_high(PIN_B7);
   delay_ms(100);
   output_low(PIN_B7);
   delay_ms(100);
}
//=========================================================================
// Main
//=========================================================================
void main()
{
   int8 p;
   //int8 test;

   init();
   //====== конфігурація портів
   set_tris_b(0b00111111);
   //====== для lcd дисплея
   delay_ms(1000);
```

```
   //====== виведення заголовка на дисплей
   //lcd_clear();
   lcd_running_string_begin(1); //початок повідомлення в рядку 4
   printf("Лаб. работа N 1");
   lcd_running_string_end();    //кінець повідомлення
   lcd_running_string_start();  //запустити біжучий рядок
   lcd_goto(2,1);
   printf("2014");
   lcd_goto(3,1);
   printf("Docent Smirnova N.V.");
   lcd_goto(4,1);
   printf("Нажата кнопка: ");

   //====== головний цикл
   for(;;){
      //=== читати порт в змінну
      p = input_b();
      //=== перевірити біт 1 на натиснення клавіші (0)
      //test = bit_test(p,1);
      //if(test == 0)
      if(!bit_test(p,1))
      {
         //=== якщо 0 запалити ліхтарик 1
         blink1();
         //=== виведення на дисплей
         lcd_goto(4,17);
         printf("B1");

      }else{
         //=== погасити ліхтарик 1
         output_low(PIN_B6);
      }
      //=== перевірити біт 3 на натиснення клавіші (0)
      //test = bit_test(p,3);
      //if(test == 0)
      if(!bit_test(p,3))
      {
         //=== якщо 0 запалити ліхтарик 2
         blink2();
         //=== виведення на дисплей
         lcd_goto(4,17);
         printf("B3");

      }else{
         //=== погасити ліхтарик 2
         output_low(PIN_B7);
      }
   }
}
```

Лістинг 2 – Програма реалізації алгоритму 1 (рис. 4) лабораторної роботи №1 для варіанту «Proteus»

```
//###################################################################
// Найменування програми
// file:  file_name.c
// Copyright (c) Docent Smirnova N.V. Kropyvnyts'kyy
//###################################################################

#include "L1.h"
#include <swc_LCD.h>       // драйвер LCD дисплея
#CASE                      // враховувати регістр символів

//====================================================================
// Ініціалізація PIC
//====================================================================
void init()
{
   setup_adc_ports(NO_ANALOGS);
   setup_adc(ADC_OFF);
   setup_spi(FALSE);
   setup_wdt(WDT_OFF);
   setup_timer_0(RTCC_INTERNAL);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_oscillator(FALSE);
}
//====================================================================
// Запалити світлодіод 1
//====================================================================
void blink1()
{
   int i;
   output_high(PIN_B6);
   output_low(PIN_B7);
   delay_ms(300);
}
//====================================================================
// Запалити світлодіод 2
//====================================================================
void blink2()
{
   int i;
   output_high(PIN_B7);
   output_low(PIN_B6);
   delay_ms(300);
}
//====================================================================
// Main
//====================================================================
void main()
{
   int8 p;

   init();
   //====== конфігурація портів
   set_tris_b(0b00111111);
```

```
   output_low(PIN_B6);
   output_low(PIN_B7);
   //====== головний цикл
   for(;;){
      lcd_goto(1,1);
      printf("Docent Smirnova N.V.");
      //=== читати порт в змінну
      p = input_b();
      //=== перевірити біт 1 на натиснення клавіші (0)
      if(bit_test(p,1))
      {
         //=== виведення на дисплей
         lcd_goto(3,1);
         printf("Button: 1");
         //=== якщо 0 запалити ліхтарик 1
         blink2();
      }
      //=== перевірити біт 3 на натиснення клавіші (0)
      if(bit_test(p,0))
      {
         //=== виведення на дисплей
         lcd_goto(3,1);
         printf("Button: 0");
         //=== якщо 0 запалити ліхтарик 2
         blink1();
      }
      delay_ms(200);
   }
}
```

Методичні вказівки до виконання лабораторної роботи:

1. Намалювати принципову схему підключень відповідно до варіанта для варіанту «Навчальний комплекс».
2. Зкомутувати кнопки, тумблери і світлодіоди відповідно до варіанта.
3. Створити алгоритм програми
4. Написати програму, що реалізує створений алгоритм.
5. Відкомпілювати програму, записати у мікроконтролер, виконати.

Контрольні питання:

1. Дати коротку характеристику мікроконтролера PIC18F252.
2. Як управляти напрямком введення-виведення даних у мікроконтролері?
3. Які команди здійснюють виведення інформації на LCD?
4. Яким чином здійснюється запис програми у мікроконтролер?
5. У яку пам'ять завантажується код програми?

Зміст звіту

У звіті повинні бути представлені:

- принципова схема з'єднань (роздруківка виводів контролера і від руки намальовані з'єднання для варіанту «Навчальний комплекс»), результат роботи у «Proteus».
- блок-схема алгоритму роботи програми.
- лістинг програми.
- висновки за результатами роботи.

Примітки:

1. Шрифт – `Courier New 10 пт.`
2. Одиночний інтервал.

Варіанти

	Введення								Виведення							
№	A0	A1	A2	A3	B0	B1	B2	B3	B0	B1	B2	B3	B4	B5	B6	B7
1	+	+							+	+						
2		+	+							+	+					
3			+	+							+	+				
4	+			+					+			+				
5					+	+							+	+		
6						+	+							+	+	
7							+	+							+	+
8		+		+						+		+				
9					+		+						+	+		
10						+		+						+		+
11	+			+						+						+
12	+		+								+				+	
13		+		+								+		+		
14					+			+					+		+	
15						+		+							+	+

Лабораторна робота №2

Тема: Робота з перериваннями мікроконтролера

Ціль роботи

Отримання навичок роботи з перериваннями мікроконтролера, програмування обробників переривань мікроконтролерів, написання та налагодження програми обробників переривань порту RS-232, таймера, зовнішніх переривань.

Завдання:

- Намалювати принципову схему підключень відповідно до варіанту.
- Для варіанту «Proteus» використовувати готову схему
- Здійснити виклик переривань із кнопок і тумблерів стенда та обробити їх відповідно до варіанту.
- Здійснити виведення результатів на LCD і на світлодіоди з обробників переривань.
- Написати програму секундоміра в обробнику переривання таймера.
- Здійснити приймання рядка символів з термінальної програми на PC по інтерфейсу RS-232, що містить ПІБ студента.
- Виведення на LCD повинне містити:

Секунди і хвилини в обробнику переривання таймера, номер зовнішнього переривання в обробнику зовнішнього переривання, рядок символів в обробнику переривання порту RS-232.

Групу, прізвище та ініціали студента.

Теоретичні відомості:

Переривання – це виклики певних функцій, які генеруються, головним чином, апаратною частиною мікроконтролера. У результаті переривання виконання програми зупиняється, і відбувається перехід до відповідної до підпрограми обробки переривання.

Переривання є найважливішим інструментом для побудови систем реального

часу. Будь-яка система повинна реагувати на події зовнішнього середовища, уміти розпізнавати їх і виконувати певні дії відповідно до заданого алгоритму. Переривання бувають внутрішніми (програмні переривання, переривання таймерів, CCP, ADC та ін.) і зовнішніми (INT0-INT2, зміна стану порту B).

Мікроконтролери PIC18F252 мають кілька джерел переривань і функцію пріоритетної системи переривань, яка дозволяє для кожного джерела переривань призначити високий або низький пріоритет. При виникненні переривання з високим пріоритетом відбувається перехід по вектору 000008h, а при виникненні переривання з низьким пріоритетом – 000018h. Переривання з високим пріоритетом припиняють обробку переривань із низьким пріоритетом.

Час переходу на обробку переривань від зовнішніх джерел (переривання INT, зміна рівня сигналу на входах PORTB і ін.) становить три-чотири цикли команд. Час переходу не залежить від типу виконуваної команди (однослівна або двохслівна). Прапори переривань встановлюються незалежно від стану битів глобального та індивідуального дозволу переривань.

Для роботи з перериваннями необхідно вказати компілятору, які переривання будуть використані у програмі.

Для роботи з перериванням таймера RTCC необхідно здійснити його ініціалізацію:

```
setup_timer_0(RTCC_INTERNAL|RTCC_DIV_8);
```

Заборонити використання інших таймерів можна директивами:

```
setup_timer_1(T1_DISABLED);
setup_timer_2(T2_DISABLED,0,1);
setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
```

Для роботи з перериваннями їх необхідно дозволити:

```
enable_interrupts(INT_RTCC);  //дозволити переривання таймера RTCC
enable_interrupts(INT_EXT);   //дозволити зовнішнє переривання 0
enable_interrupts(INT_EXT1);  //дозволити зовнішнє переривання 1
enable_interrupts(INT_EXT2);  //дозволити зовнішнє переривання 2
enable_interrupts(INT_RDA);   //дозволити переривання порту RS-232
```

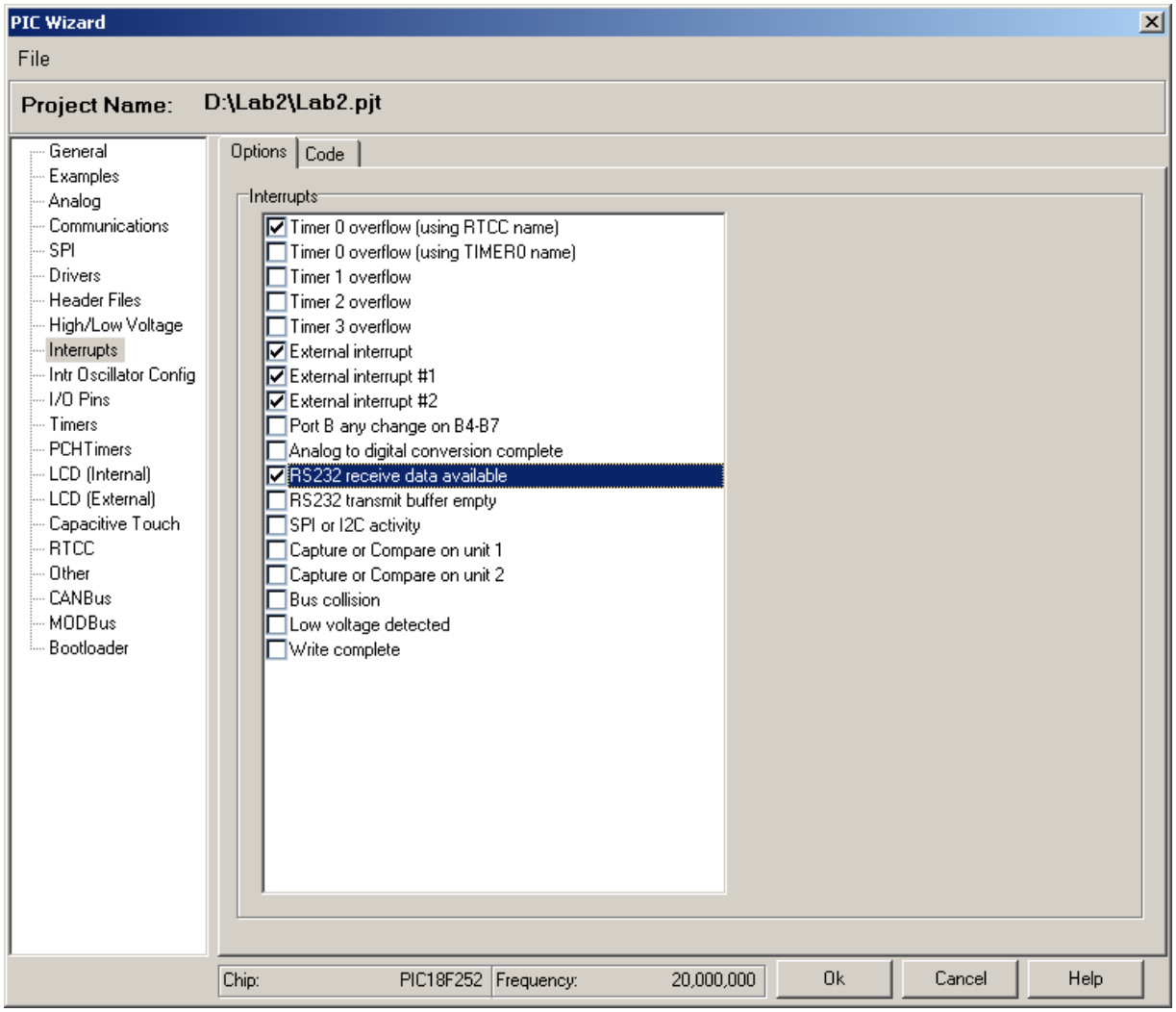
Далі необхідно дозволити всі зазначені переривання:

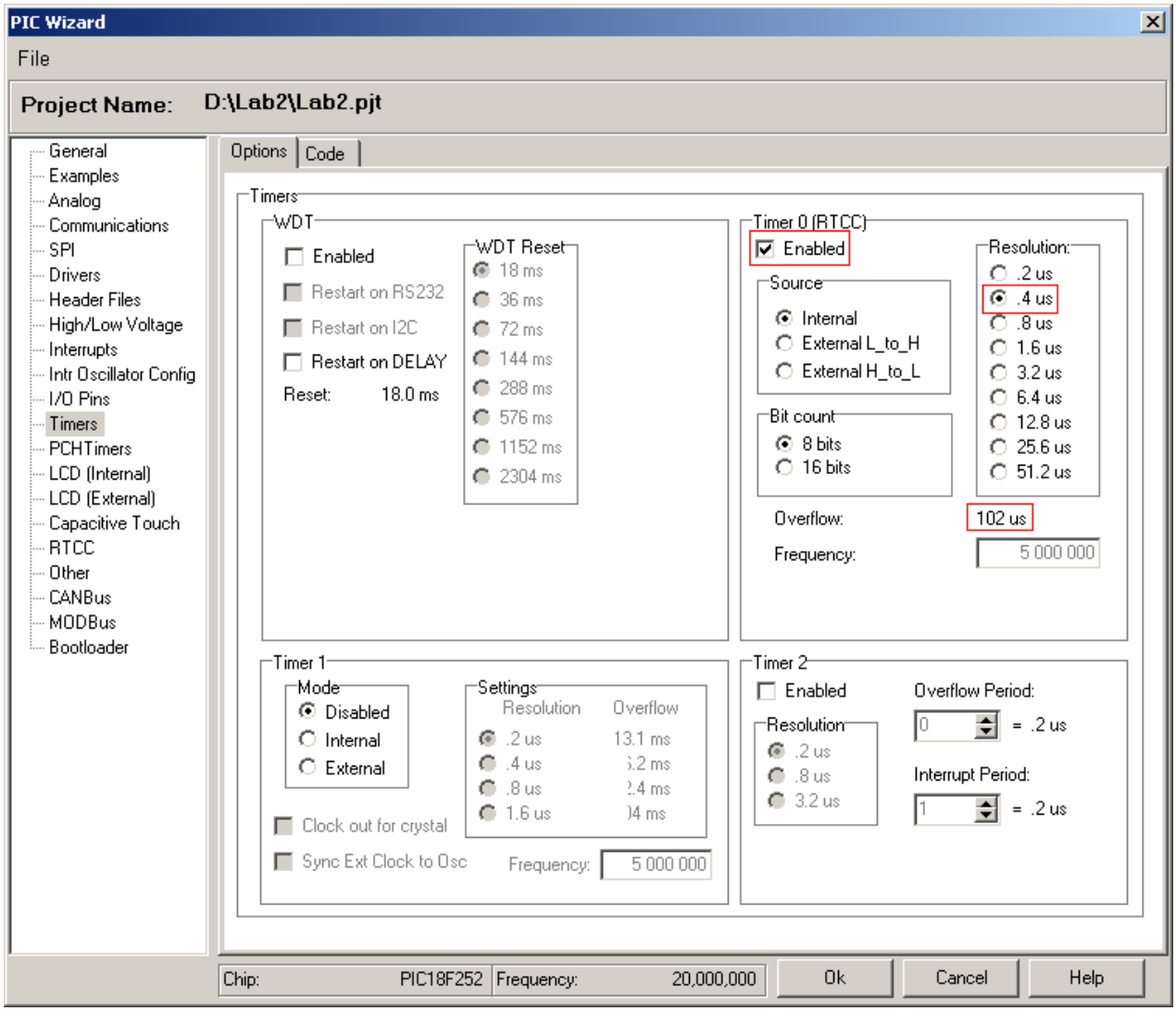
```
enable_interrupts(GLOBAL);    //дозволити всі зазначені переривання
```

Можна в довільному порядку заборонити переривання:

```
disable_interrupts(INT_RTCC); //заборонити переривання таймера RTCC
disable_interrupts(INT_EXT);  //заборонити зовнішнє переривання 0
disable_interrupts(INT_EXT1); //заборонити зовнішнє переривання 1
disable_interrupts(INT_EXT2); //заборонити зовнішнє переривання 2
disable_interrupts(INT_RDA);  //заборонити переривання порту RS-232
disable_interrupts(GLOBAL);   //заборонити всі зазначені переривання
```

Переривання можна встановити з IDE:





Нажати кнопку Ok. Проект готовий, можна приступати до написання програми. Буде згенерированний наступний код:

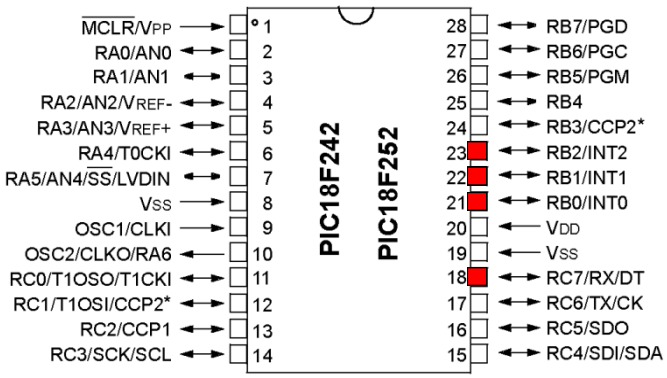
```
#include <Lab2.h>
#int_RTCC
void  RTCC_isr(void)
{

}
#int_EXT
void  EXT_isr(void)
{

}
#int_EXT1
void  EXT1_isr(void)
{

}
#int_EXT2
void  EXT2_isr(void)
{

}
#int_RDA
```

```
void  RDA_isr(void)
{

}

void main()
{
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_2|RTCC_8_bit);      //102 us overflow

   setup_timer_3(T3_DISABLED | T3_DIV_BY_1);

   enable_interrupts(INT_RTCC);
   enable_interrupts(INT_EXT);
   enable_interrupts(INT_EXT1);
   enable_interrupts(INT_EXT2);
   enable_interrupts(INT_RDA);
   enable_interrupts(GLOBAL);

   while(TRUE)
   {
      //TODO: User Code
   }

}
```

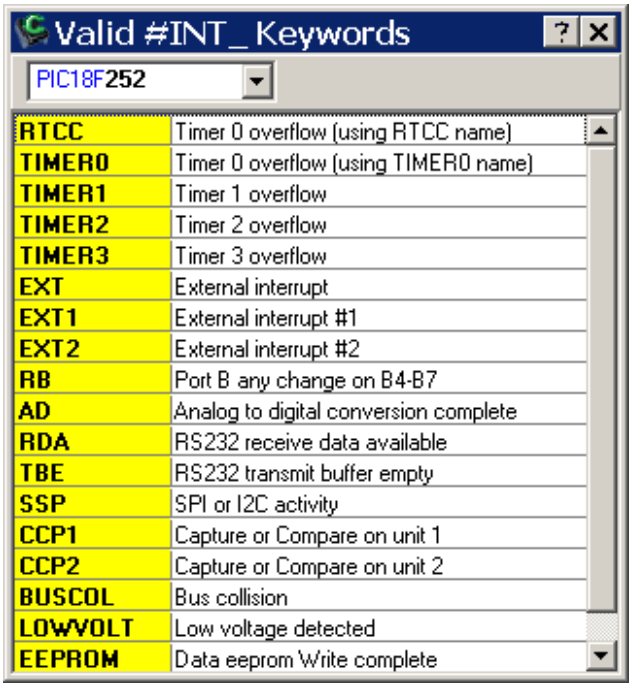
ПРИМІТКА: При роботі із зовнішніми перериваннями порту B контролера необхідно виводи порту B підключити до джерела живлення через резистори, тим самим забезпечивши наявність логічної «1» на виводах порту:

```
port_b_pullups(TRUE);  //входи порту B через резистори підключити до
                       // джерела 5 В
```

Прив'язка переривань до виводів мікроконтролера:



Доступні переривання мікроконтролера:



Порядок роботи для варіанту «Навчальний комплекс»

1. **Методичні вказівки до виконання лабораторної роботи:**

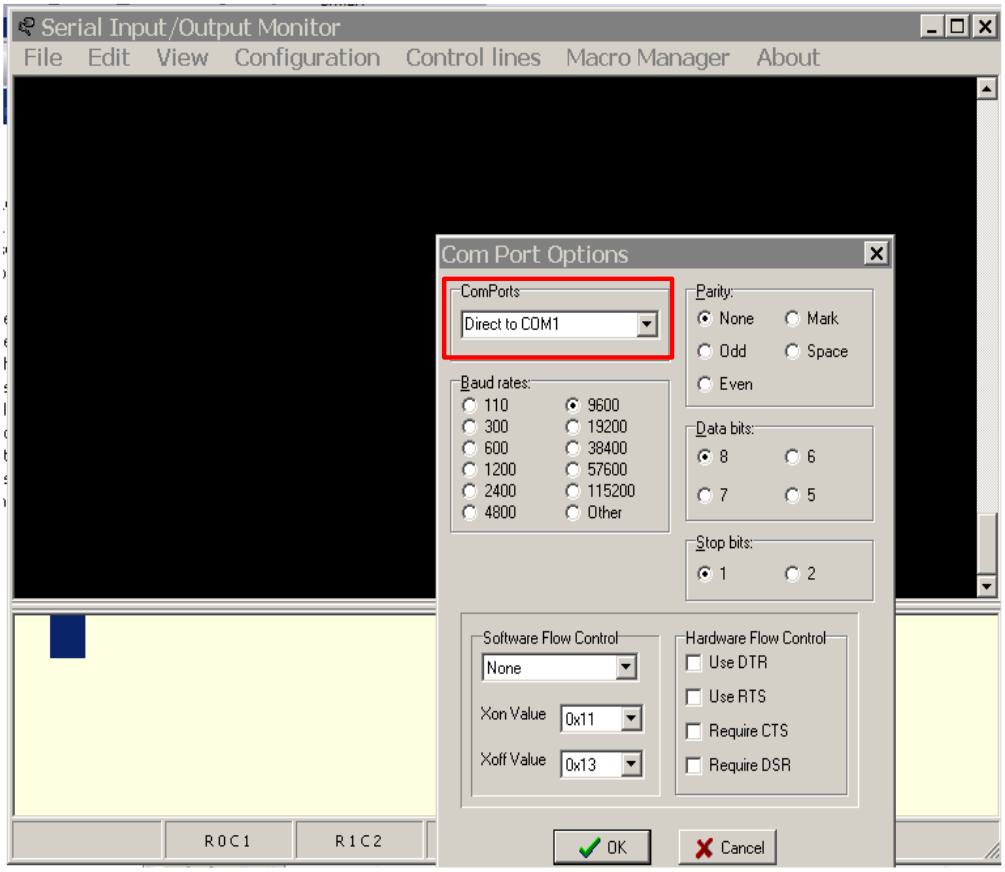
1. Намалювати принципову схему підключень відповідно до варіанту.
2. Зкомутувати кнопки, тумблери і світлодіоди відповідно до варіанту.

   Кнопками здійснювати виклик переривань INT0, INT1, INT2.

   Переривання порту RS-232 здійснювати шляхом посилки рядка символів з терміналу SIOW.exe або Terminal.exe. Вивести простий рядок (прізвище, ім'я) на LCD дисплей.

Програма - термінал SIOW.exe

Для роботи із програмою необхідно вибрати порт і встановити параметри порту:



Потім у меню **File | download Software** вибрати ***hex** – файл.

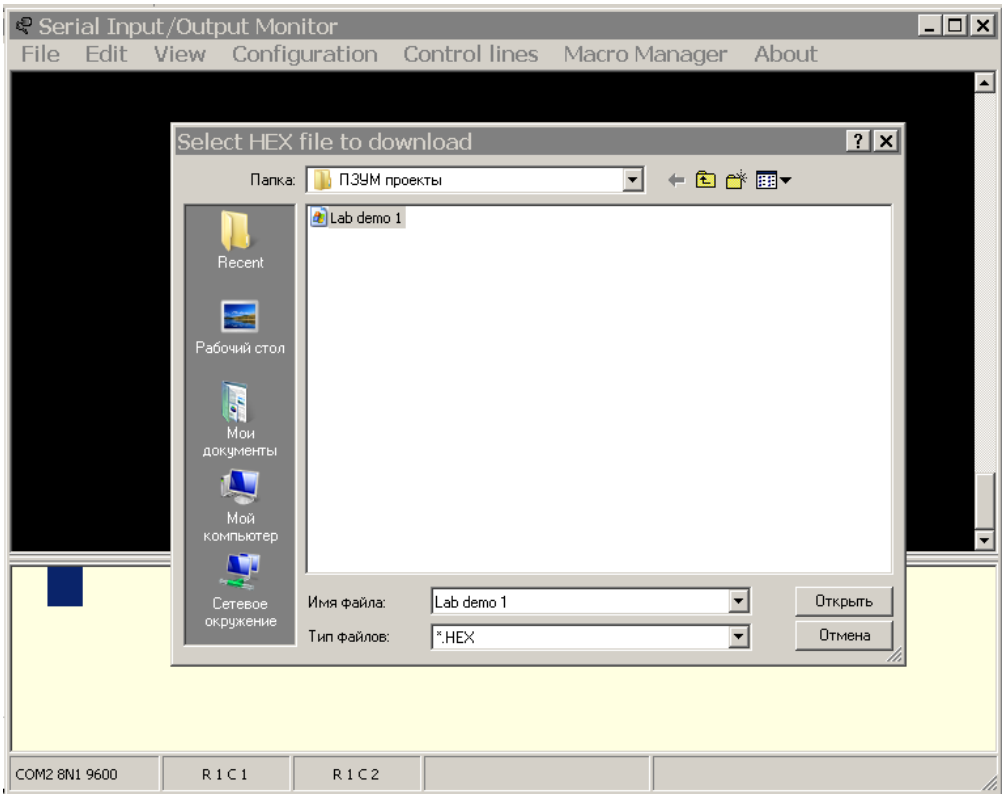


Рис. 1. Термінал SIOW.

Програма - термінал TERMINAL

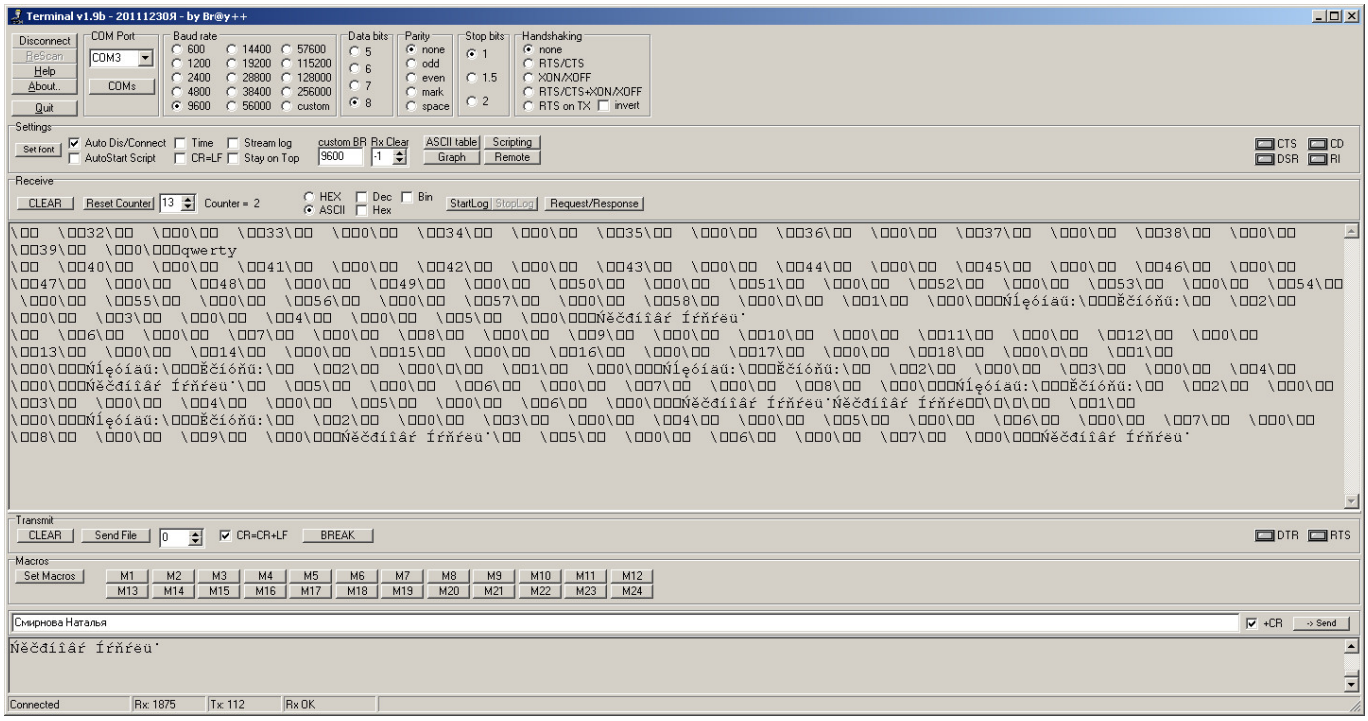


Рис. 2. Термінал TERMINAL.

Примітка: в обробнику переривання порту RS-232 необхідно очистити прийомний буфер, що б уникнути повторних переривань. Очищення буфера здійснюється шляхом його читання функціями: getc(), gets(), get_string().

1. Створити алгоритм програми з обробниками переривань.
2. Створити алгоритм програми з обробниками переривань.
3. Написати програму, що реалізує створений алгоритм в обробнику переривань.
4. Відкомпілювати програму, записати в мікроконтролер, виконати.

Порядок роботи для варіанту «Proteus»:

1. Завантажити з сервера папку «Proteus_students».
2. Відкрити середовище розробки PCWH.
3. У папці «Proteus_students» вибрати папку відповідної лабораторної роботи (LAB1... LAB8).

4. Відкрити файл проекту з розширенням *.Pjt.

5. У редакторі відкрити шаблон файлу програми з розширенням *.C.

6. Відкрити середу моделювання «Proteus».

7. У папці «Proteus_students» вибрати папку відповідної лабораторної роботи (LAB1... LAB8).

8. Відкрити файл проекту з розширенням *. DSN.

9. У середовищі розробки PCWH, використовуючи шаблон програми самостійно написати, відкомпілювати програму у відповідності з завданням.

10. У середовищі розробки «Proteus» виконати програму.

Примітка: Файл демонстрацій виконання лабораторної роботи розташований в папці «Demo».

Принципова схема для виконання лабораторної роботи для варіанту «Proteus»

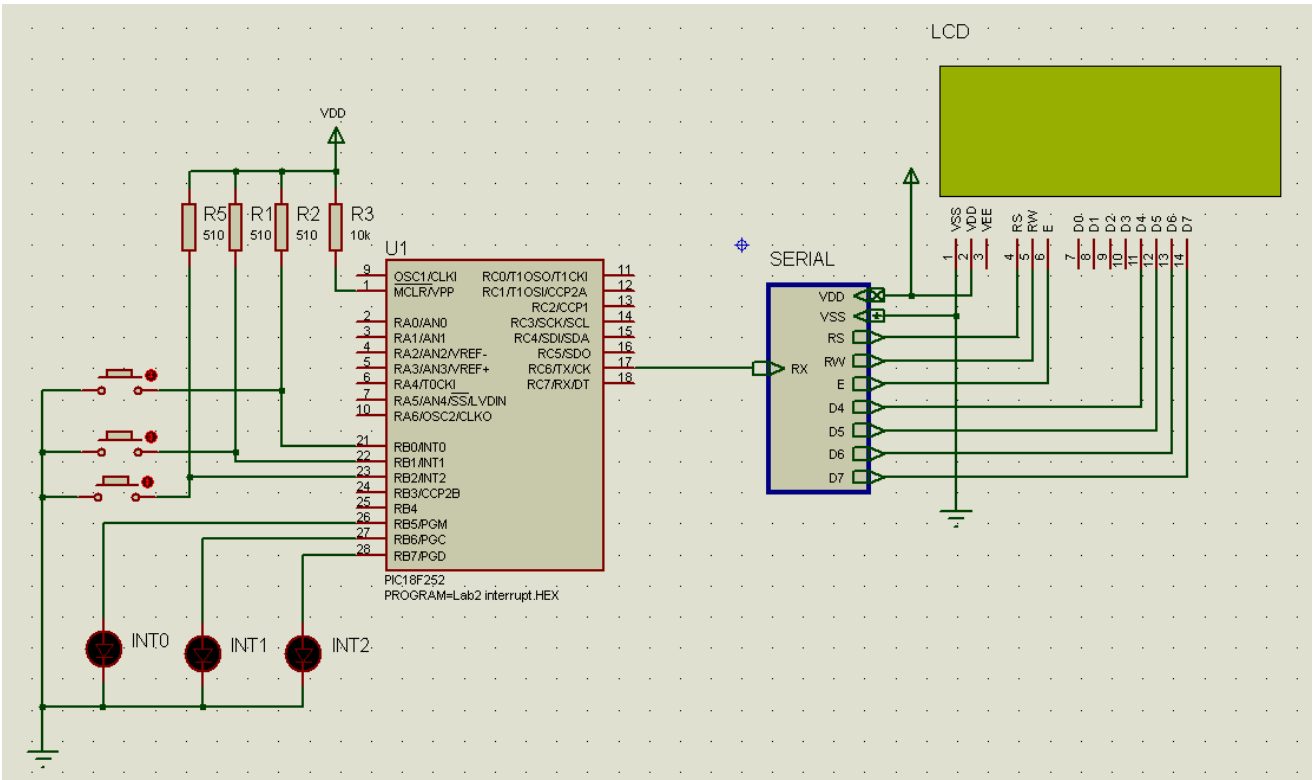


Рисунок 3 - Принципова схема для виконання лабораторної роботи для варіанту «Proteus»

Результат виконання програми:

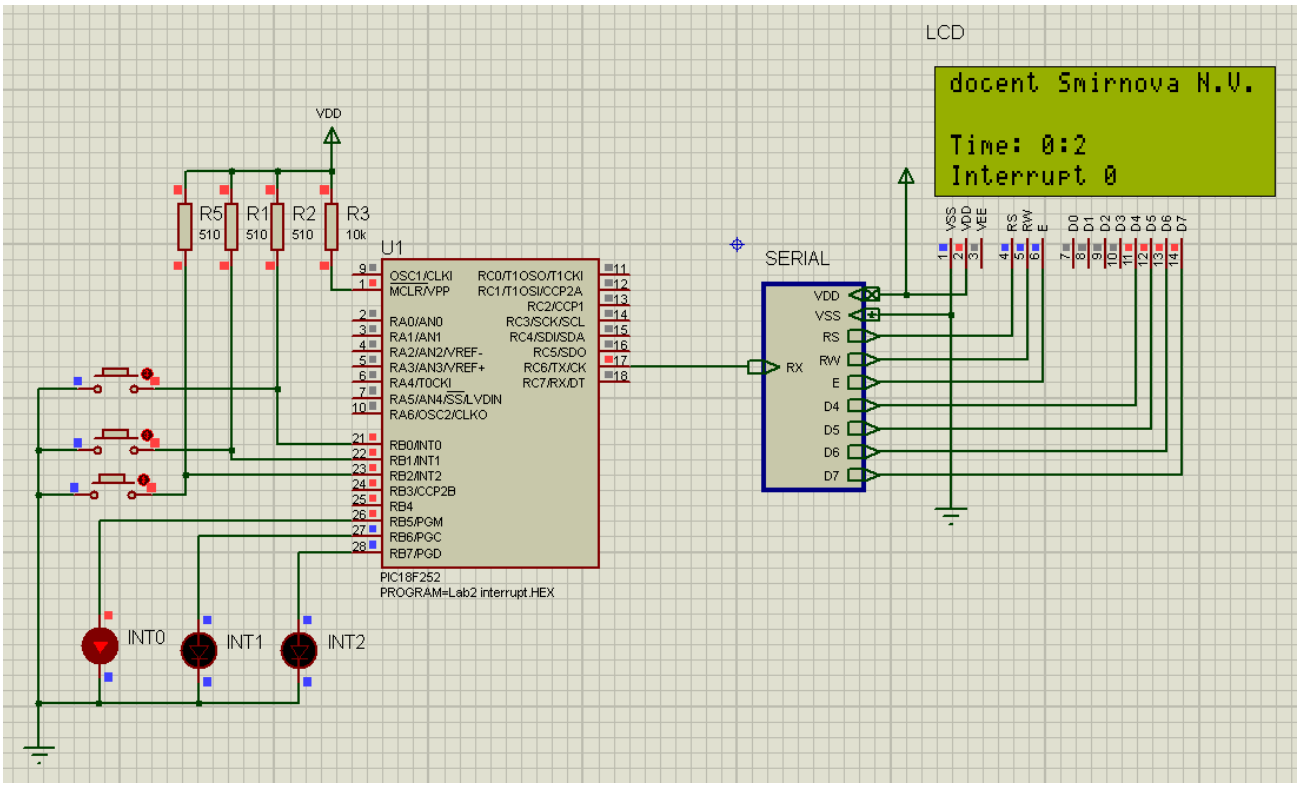


Рисунок 4- Результат виконання лабораторної роботи для варіанту «Proteus»

Контрольні питання:

1. Причини виникнення внутрішніх і зовнішніх переривань мікроконтролера PIC18F252. (Типи переривань).
2. Як управляти (дозволяти, забороняти) перериваннями у мікроконтролері?
3. Чому в перериванні порту RS-232 необхідно очистити прийомний буфер?
4. Якими по відношенню до контролера бувають переривання?
5. Навіщо потрібні переривання в системах керування?

Зміст звіту:

У звіті повинні бути представлені:

- принципова схема з'єднань (роздруківка виводів контролера і від руки намальовані з'єднання для варіанту «Навчальний комплекс»), результат роботи для варіанту «Proteus».
- блок-схема алгоритму роботи програми.
- лістинг програми.
- висновки за результатами роботи.

Варіанти

	Переривання				
№	**RTCC**	**RS-232**	**INT0**	**INT1**	**INT2**
1	+	+	+	+	
2	+	+	+		+
3	+	+		+	+
4	+	+	+		+
5	+	+	+	+	
6	+	+	+		+
7	+	+		+	+
8	+	+	+		+
9	+	+	+	+	
10	+	+	+		+
11	+	+		+	+
12	+	+	+		+
13	+	+	+	+	
14	+	+	+		+
15	+	+		+	+

Лабораторна робота №3

Тема: Робота з АЦП

Ціль роботи

Одержання навичок роботи з АЦП мікроконтролера і написання програм обробки оцифрованих аналогових сигналів

Завдання:

- Намалювати принципову схему підключень відповідно до варіанту.
- Для варіанту «Proteus» використовувати готову схему
- Здійснити зчитування рівня напруги з потенціометра, оцифрувати за допомогою АЦП і перетворити в десятковий формат відповідно до варіанту.
- Відобразити рівень поточної напруги у вольтах на LCD. Показання напруги на LCD повинно відрізнятися від показань вольтметра не більше, ніж на 0,01 В.
- Написати програму визначення натиснутої кнопки на клавіатурі.
- Виведення на LCD повинне містити:
- Зчитані дані з АЦП у HEX форматі, десятковому форматі у вольтах, символ натиснутої клавіші, прізвище та ініціали студента.

Теоретичні відомості:

АЦП (Аналого-Цифровий Перетворювач) або ADC (Analog to Digital Convertor) призначений для перетворення аналогових сигналів у їх цифрове представлення. Якість перетворення залежить від швидкості перетворення, частоти дискретизації і розрядності АЦП.

Існують три основні методи перетворення:

- метод прямого (паралельного) перетворення;
- метод послідовного наближення (поразрядного врівноваження);
- метод інтегрування.

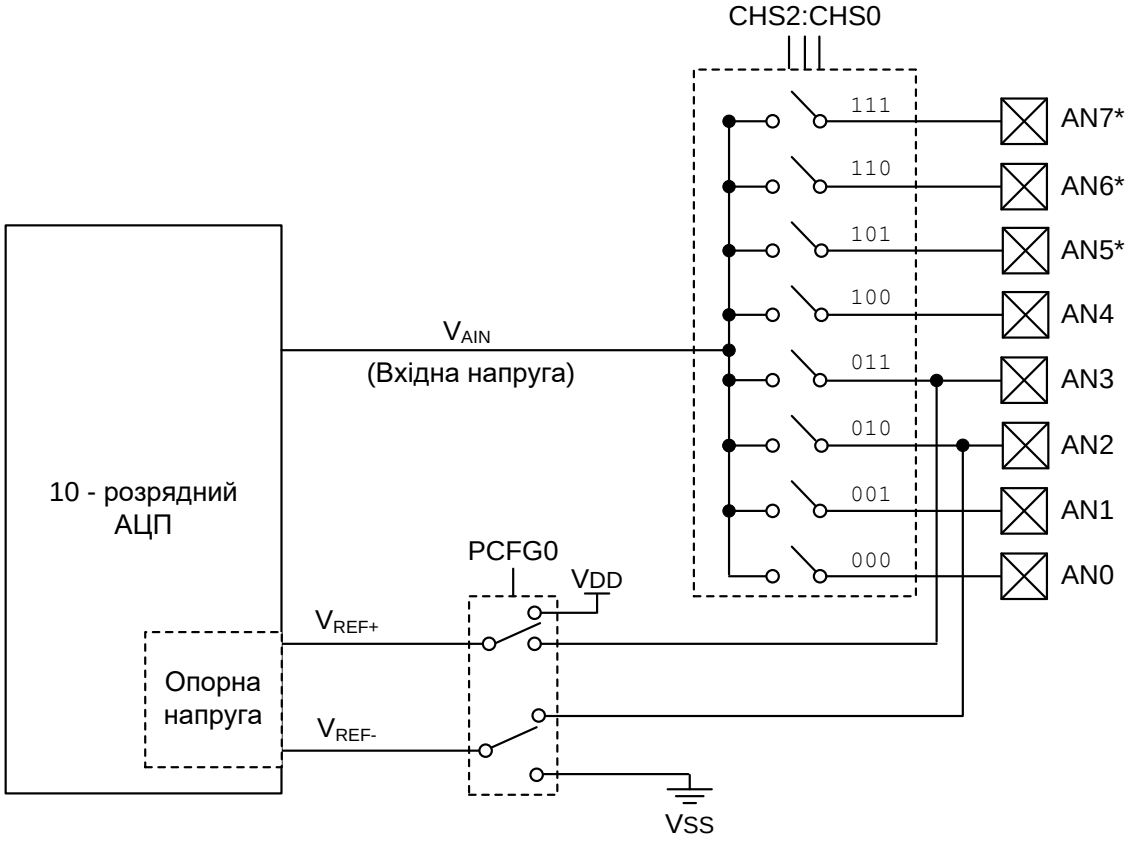


Рисунок 3.1 - Структурна схема модуля АЦП

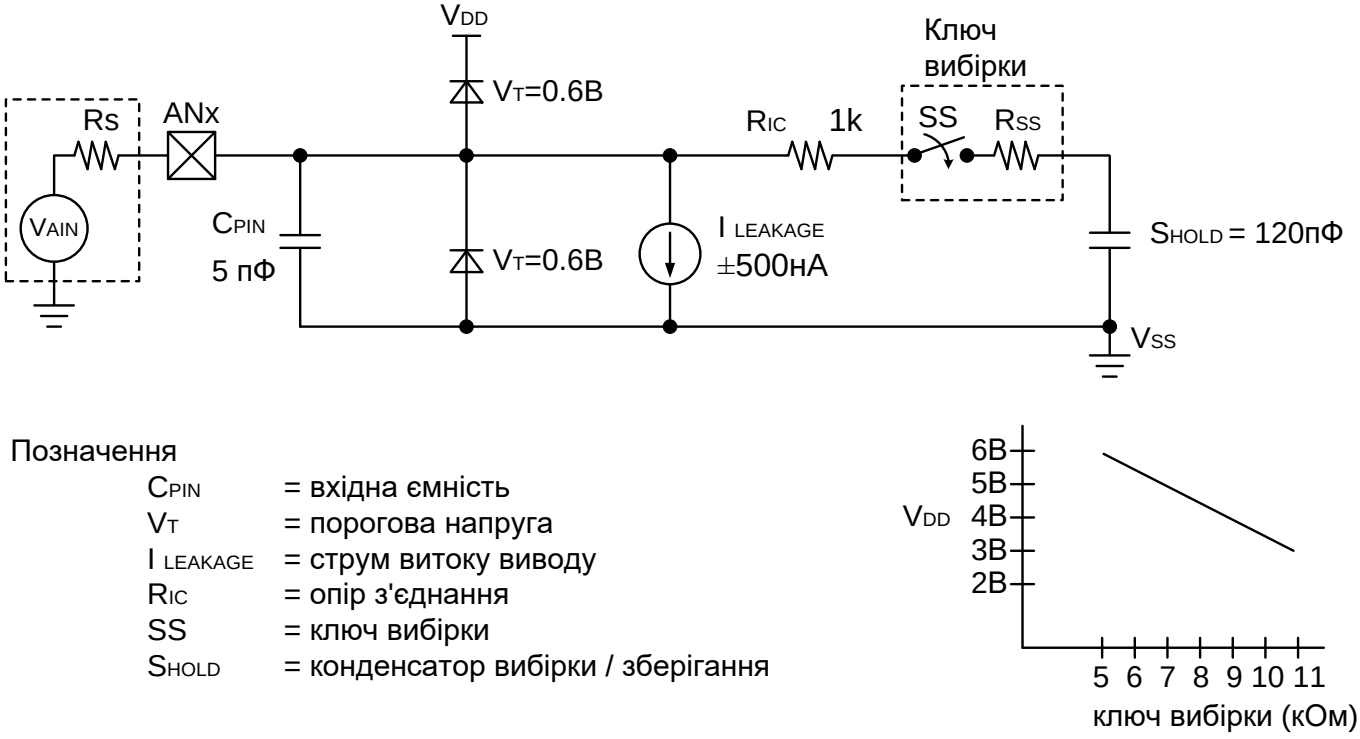


Рисунок 3.2 - Схема аналогового входу АЦП

Модуль АЦП у PIC18F252 має п'ять каналів (рис. 3.1). Кожний канал порту, пов'язаний з модулем АЦП, може бути настроєний як аналоговий вхід (RA3 і RA2 як входи опорної напруги) або цифрового входу/виходу.

Принцип перетворення представлений на рис. 3.2. Вхідний аналоговий сигнал через комутатор каналів заряджає внутрішній конденсатор АЦП $C_{HOLD.}$ Модуль АЦП перетворює напругу, яка утримувалась на конденсаторі C_{HOLD} у відповідний 10-розрядний цифровий код методом послідовного наближення.

При скиданні мікроконтролера значення всіх його регістрів встановлюються за замовчуванням. Скидання виключає модуль АЦП, а також зупиняє процес перетворення, якщо він був початий.

На рисунку 3.3 показані результати перетворення аналогового сигналу з низькою розв'язною здатністю АЦП – 8 розрядів (256 рівнів) і високою частотою дискретизації:

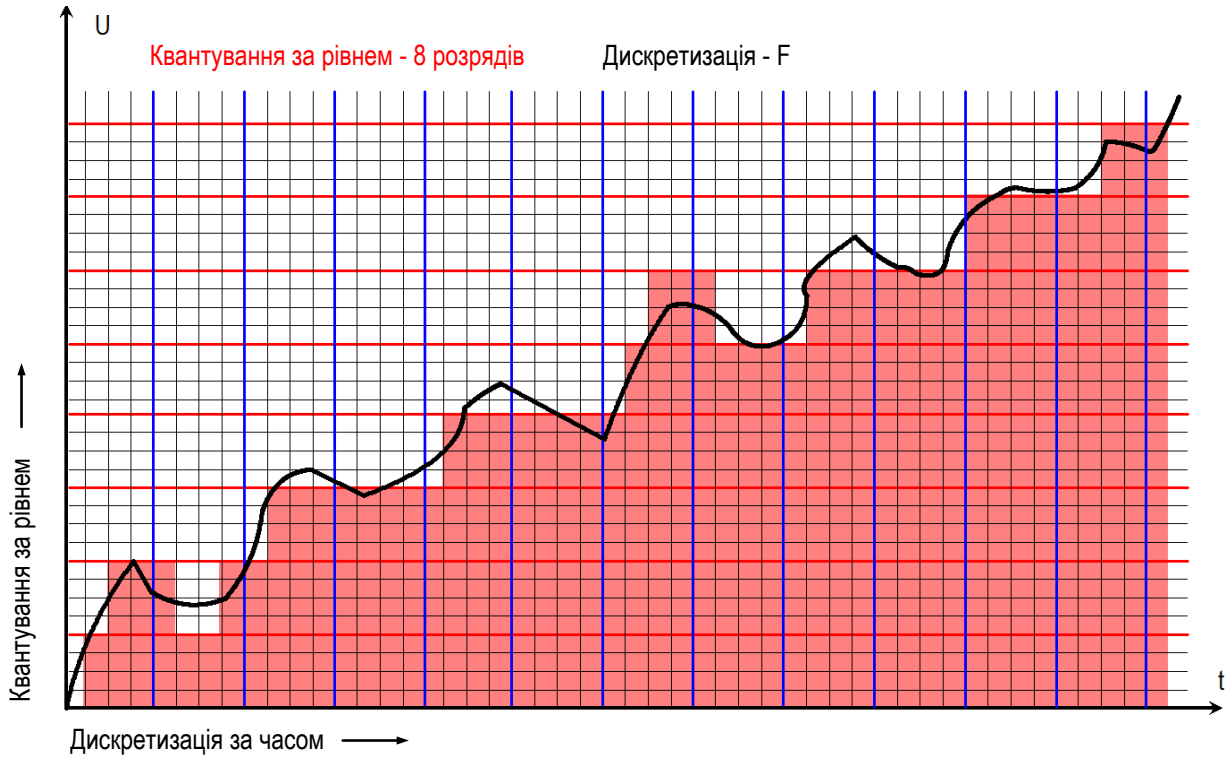


Рисунок 3.3 – Перетворення сигналу частотою дискретизації F і розрядністю = 8

На рисунку 3.4 показані результати перетворення аналогового сигналу з більш високою розв'язною здатністю АЦП – 10 розрядів (1024) рівня і низькою частотою

дискретизації.

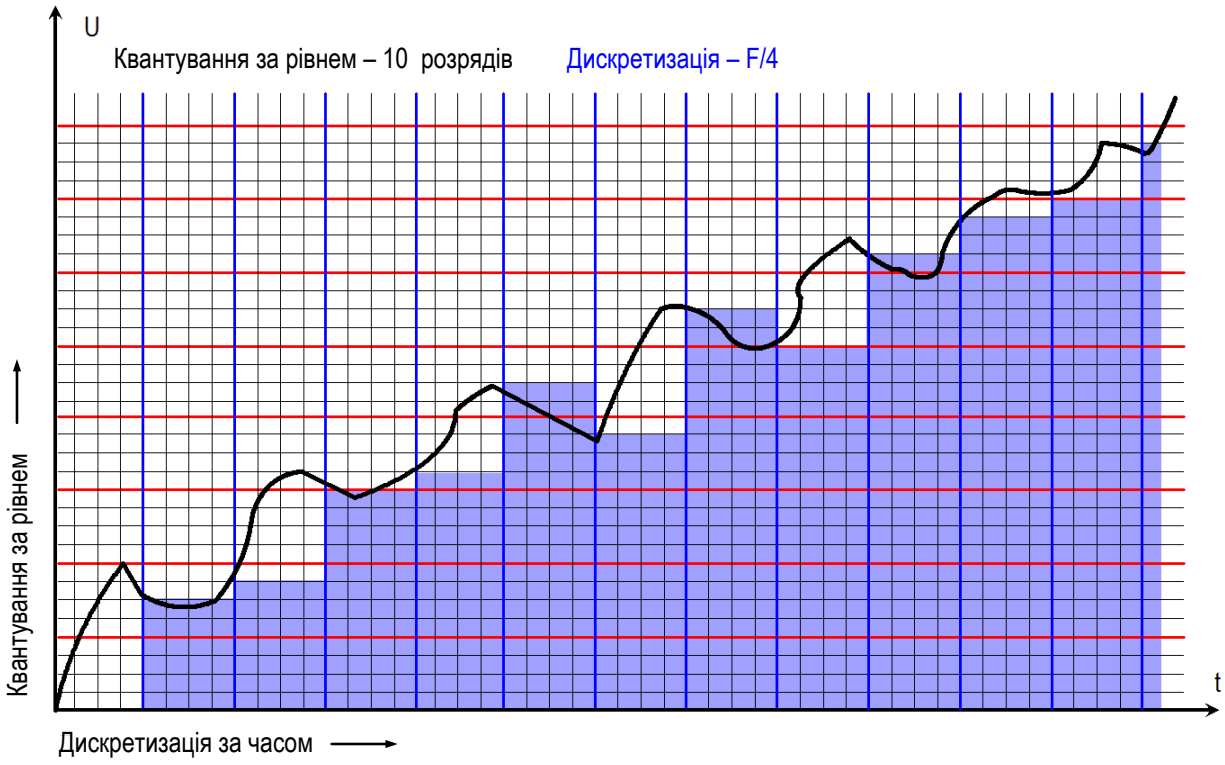


Рисунок 3.4 – Перетворення сигналу частотою дискретизації F/4 і розрядністю = 10

Представлені графіки показують, що низька розв'язна здатність АЦП і низька частота дискретизації не дають можливості якісного перетворення аналогового сигналу. Теорема Котельнікова (Шеннона) визначає мінімальну частоту дискретизації, що як удвічі перевищує найвищу гармоніку в спектрі вихідного сигналу.

Тобто, якщо основний сигнал (перша гармоніка) має частоту 1 кГц, а п'ята (наприклад, найвища) – 32 кГц, то частота дискретизації АЦП повинна бути не менше 64 кГц.

У мікроконтролері PIC18F252 розрядність АЦП – 8 і 10 розрядів. Частота дискретизації визначається програмістом.

Важливим параметром є опорна напруга Vref, відносно якої проводиться відлік шкали перетворення. Наприклад, якщо Vref = 10В, а розрядність АЦП = 8 розрядів (255 рівнів), то ціна (вага) одного розряду складе 10В/255 = 0.039В або 39 мВ.

Якщо Vref = 2В, то ціна одного розряду складе 2В/255 = 0.007В або 7 мВ.

Перевагою високого рівня Vref є більш висока завадостійкість АЦП, а недоліком - необхідність застосування підсилювача, якщо рівень сигналу недостатній для перетворення в повний масштаб шкали.

Перевагою низького рівня Vref є можливість перетворення сигналу з більш низьким рівнем, а недоліком – необхідність захисту від перешкод.

У кожному разі необхідно, щоб максимальний рівень сигналу дорівнював опорній напрузі Vref.

Зазвичай Vref встановлюють рівним напрузі живлення АЦП, тобто, Vref = VDD = 5В.

Прив'язка входів АЦП до виводів мікроконтролера представлена на рис 3.5.

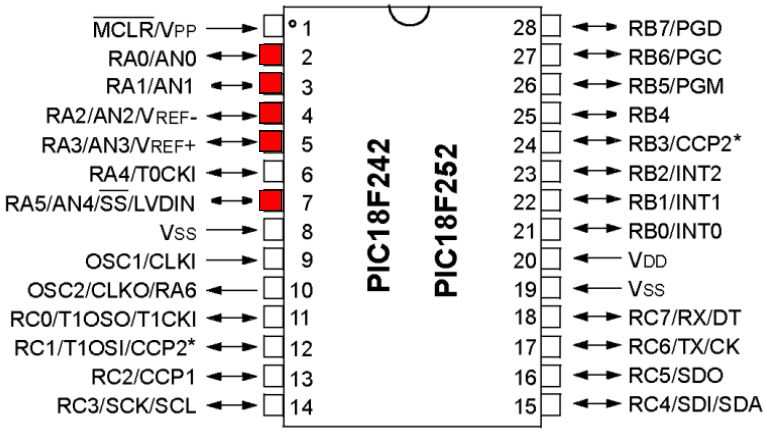


Рисунок 3.5 – Прив'язка входів АЦП до виводів мікроконтролера

Для конфігурування мікроконтролера для робота з АЦП, у заголовному файлі *.h необхідно ввести рядок, що вказує розрядність АЦП:

```
#device adc=8 або #device adc=10
```

а у функції ініціалізації ввести рядки:

```
setup_adc_ports(ALL_ANALOG);
setup_adc(ADC_CLOCK_INTERNAL);
```

Конфігурування можна зробити з IDE при створенні проекту:

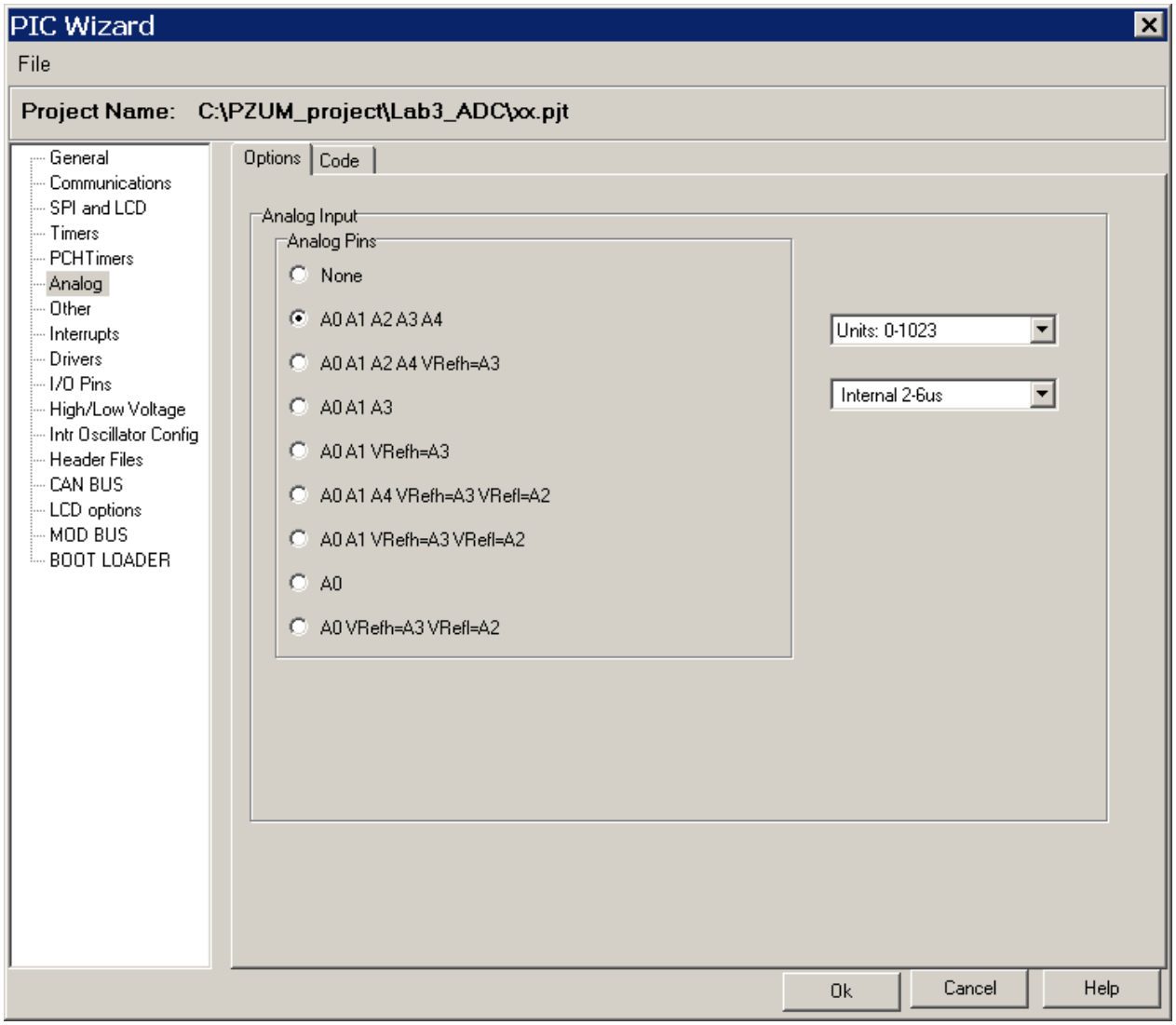


Рисунок 3.6 – Конфігурування АЦП із IDE при створенні проекту

Методичні вказівки до виконання лабораторної роботи

Порядок роботи для варіанту «Proteus»:

1. Завантажити з сервера папку «Proteus_students».
2. Відкрити середовище розробки PCWH.
3. У папці «Proteus_students» вибрати папку відповідної лабораторної роботи (LAB1... LAB8).
4. Відкрити файл проекту з розширенням *. Pjt.
5. У редакторі відкрити шаблон файлу програми з розширенням *. C.
6. Відкрити середу моделювання «Proteus».

7. У папці «Proteus_students» вибрати папку відповідної лабораторної роботи (LAB1... LAB8).

8. Відкрити файл проекту з розширенням *. DSN.

9. У середовищі розробки PCWH , використовуючи шаблон програми самостійно написати, відкомпілювати програму у відповідності з завданням.

10. У середовищі розробки «Proteus» виконати програму.

Примітка: Файл демонстрацій виконання лабораторної роботи розташований в папці «Demo».

Загальна схема процесу розробки для варіанту «Proteus» виглядає наступний чином:

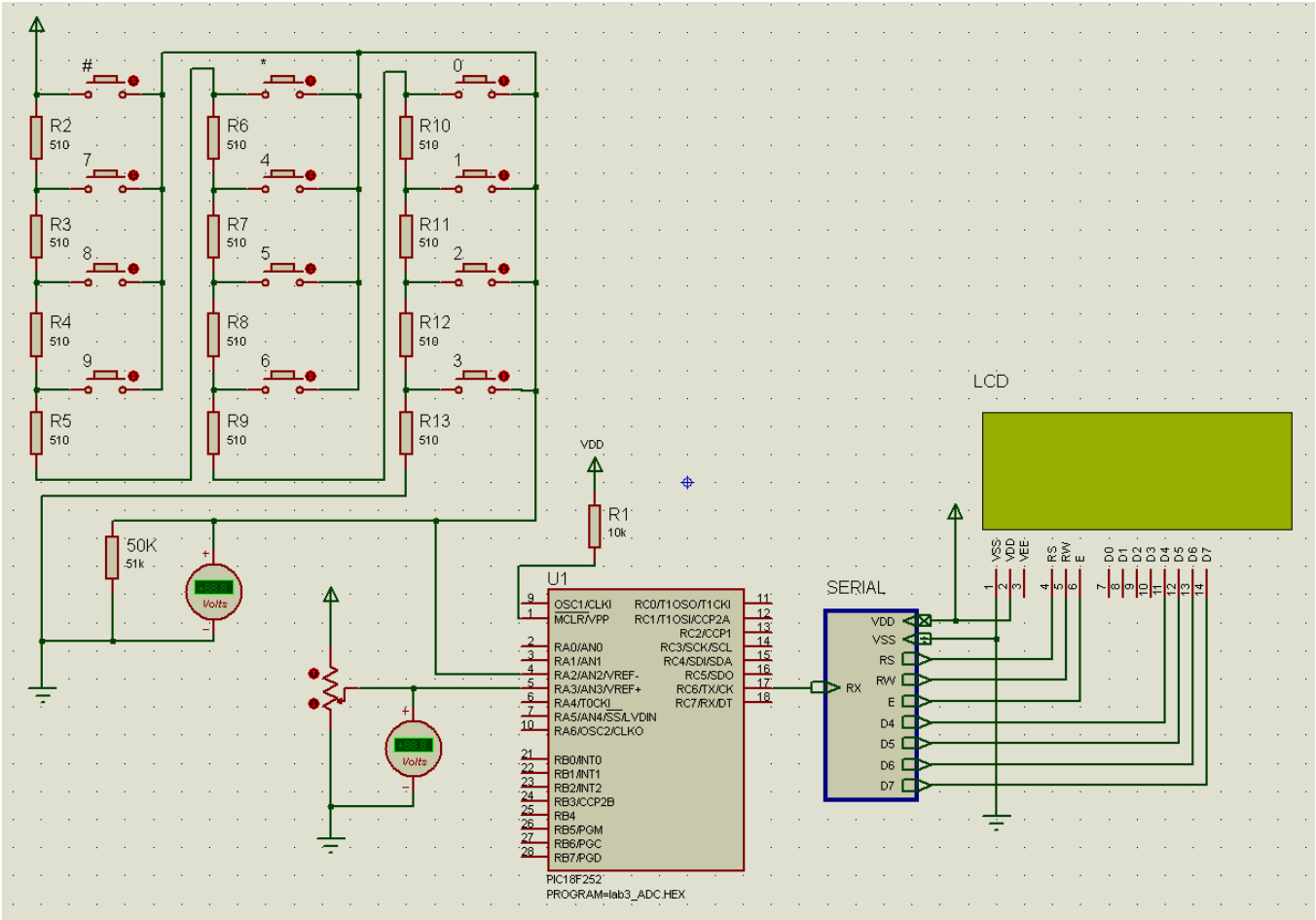


Рисунок 3.7 - Принципова схема для виконання лабораторної роботи для варіанту «Proteus»

Результат виконання програми:

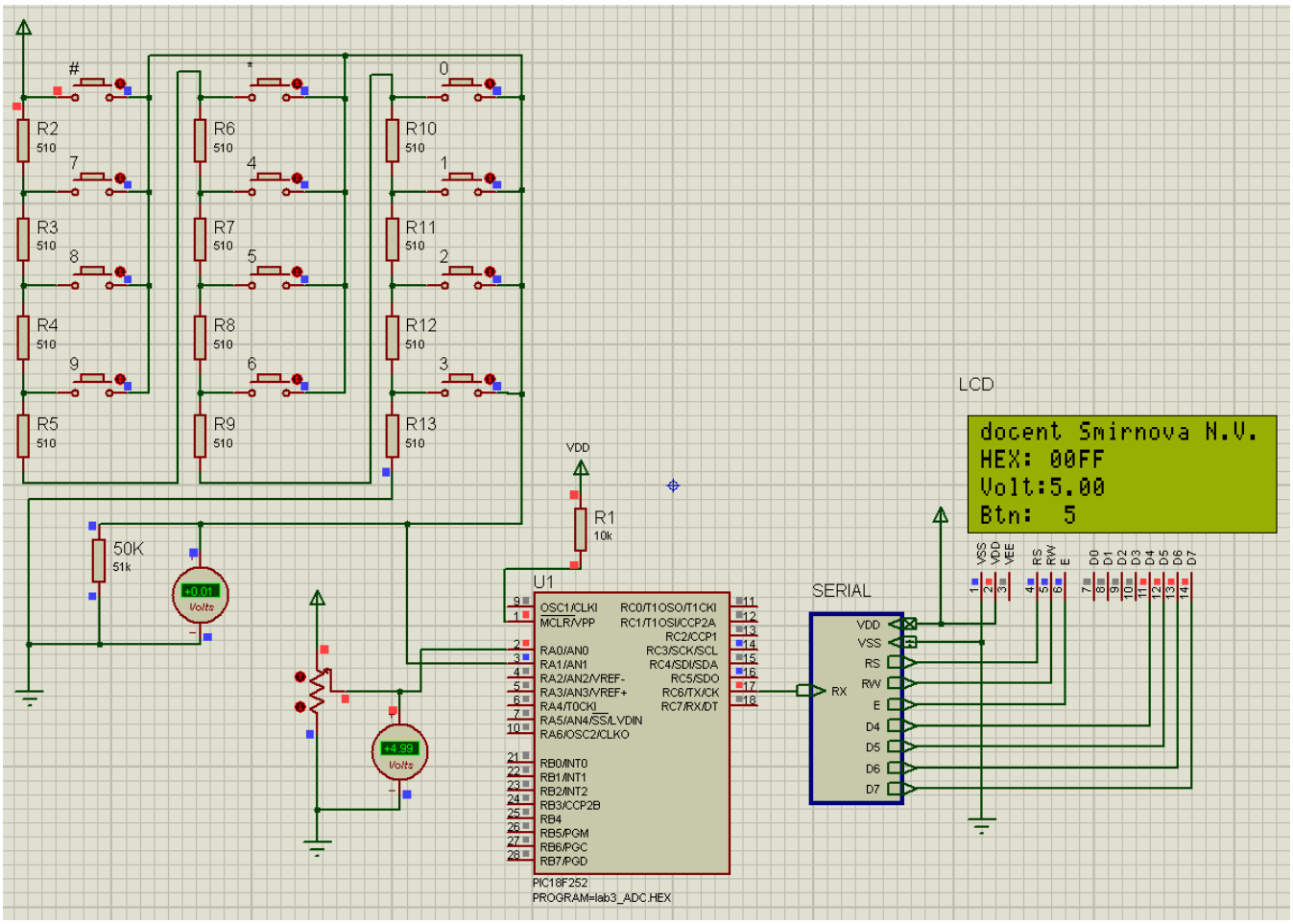


Рисунок 3.8 - Результат виконання лабораторної роботи для варіанту «Proteus»

Порядок роботи для варіанту «Навчальний комплекс»:

1. Намалювати принципову схему підключень відповідно до варіанта.
2. Підключити потенціометр, клавіатуру і вольтметр до АЦП відповідно до варіанта.

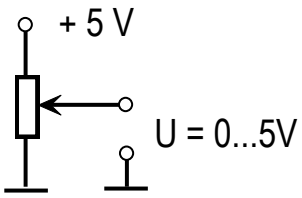


Рисунок 3.8 - Принципова схема потенціометра

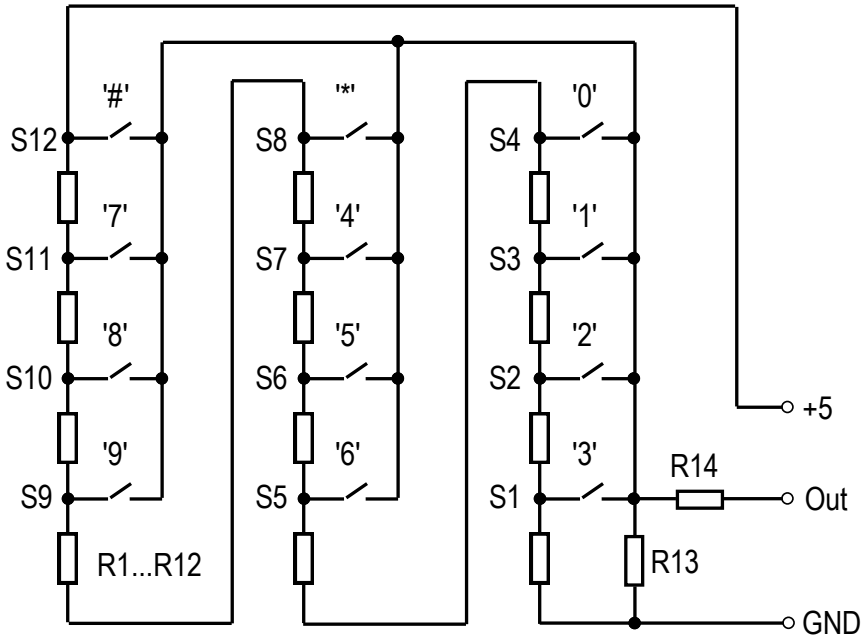


Рисунок 3.9 - Принципова схема клавіатури

3. Створити алгоритм програми роботи з АЦП для перетворення рівня напруги з потенціометра і клавіатури.
4. Написати програму, що реалізує створений алгоритм.
5. Відкомпілювати програму, записати в мікроконтролер, виконати.

Приклад зчитування даних з АЦП з каналу 2 і перетворення у вольти:

```
float volt;
int16 adc_val;

//====== вибрати номер каналу
set_adc_channel(2);
//====== затримка для роботи АЦП
delay_us(50);
//====== прочитати значення
adc_val = read_adc();
//====== визначити вольти
volt = adc_val * ADC_CVANT;
```

Змінна `ADC_CVANT` є ціною ділення шкали АЦП і обчислюється як частка від ділення Vref на розрядність АЦП, виражену у двійкових одиницях:

```
//====== опорна напруга = напруга живлення
const float REF_VDD = 5.00;
const float ADC_8 = 255;
const float ADC_10 = 1023;
```

```
//====== ділення шкали: 5В/шкала (розрядність АЦП)
//=== 8 розрядів
const float  ADC_CVANT = REF_VDD/ADC_8;
//=== 10 розрядів
const float  ADC_CVANT = REF_VDD/ADC_10;
```

Для визначення натиснутої кнопки на клавіатурі необхідно визначити рівень напруги на кожній клавіші:

```
//====== коди клавіш у вольтах відповідно до розведення плати
const float KBD_CVANT = REF_VDD/12;      //усього 12 кнопок

//=== кнопки за схемою відповідно до розведення плати
const float ADC_BTN_0 = KBD_CVANT*4;
const float ADC_BTN_1 = KBD_CVANT*3;

...

const float ADC_BTN_STAR = KBD_CVANT*8;
const float ADC_BTN_DIEZ = KBD_CVANT*12;
//=== не натиснута жодна кнопка
const float ADC_BTN_NOT = KBD_CVANT/2;
```

Потім порівнювати значення АЦП у вольтах при натисканні клавіші з рівнями напруги, визначеним для кожної клавіші. При збігу значень здійснюється ідентифікація натиснутою клавіші:

```
if((pushed_button > ADC_BTN_STAR-LUFT)&&(pushed_button < ADC_BTN_STAR+LUFT)){
      char_button = '*';
 }
 if((pushed_button > ADC_BTN_DIEZ-LUFT)&&(pushed_button < ADC_BTN_DIEZ+LUFT)){
      char_button = '#';
 }
   //====== не натиснута
   if((pushed_button < ADC_BTN_NOT)){
      char_button = 0;
 }
```

Примітка: після виводу значень на LCD необхідно ввести затримку часу індикації:

```
delay_ms(100);
```

Контрольні питання:

1. Призначення АЦП і методи перетворення.
2. Конфігурування мікроконтролера для роботи з АЦП.
3. Вплив розрядності АЦП на точність перетворення.
4. Опорна напруга Vref і його вплив на параметри АЦП.
5. Як визначається ціна ділення (вага) шкали АЦП?

Зміст звіту:

У звіті повинні бути представлені:

- принципова схема з'єднань (роздруківка виводів контролера і від руки намальовані з'єднання для варіанту «Навчальний комплекс»), результат роботи для варіанту «Proteus».
- блок-схема алгоритму роботи програми.
- лістинг програми.
- висновки за результатами роботи.

Варіанти

	Канал АЦП								Розрядність АЦП	
	Потенціометр				Клавіатура					
№	A0	A1	A2	A3	A0	A1	A2	A3	8	10
1	+					+			+	
2		+					+			+
3			+					+	+	
4				+	+					+
5				+		+			+	
6			+			+				+
7	+							+	+	
8	+						+			+
9		+						+	+	
10			+		+					+
11				+	+				+	
12	+					+				+
13		+			+				+	
14			+					+		+
15				+			+		+	

Лабораторна робота № 4

Тема: Робота з ЦАП

Ціль роботи

Одержання навичок роботи із ЦАП MCP4921 і написання драйвера послідовного інтерфейсу SPI.

Завдання:

- Намалювати принципову схему підключень відповідно до варіанта.
- Для варіанту «Proteus» використовувати готову схему
- Написати програму драйвера послідовного інтерфейсу SPI.
- Здійснити зчитування рівня напруги з потенціометра, оцифрувати за допомогою АЦП відповідно до варіанта.
- Відобразити рівень поточної напруги у вольтах на LCD.
- Вивести цифрове значення напруги на ЦАП через інтерфейс SPI.
- Здійснити зчитування рівня напруги з виходу ЦАП, оцифрувати за допомогою АЦП.
- Відобразити рівень напруги з виходу ЦАП у вольтах на LCD.
- Виведення на LCD повинне містити:

Зчитані дані з АЦП у вольтах, зчитані дані з ЦАП у вольтах, групу, прізвище та ініціали студента.

Теоретичні відомості:

ЦАП (Цифро - Аналоговий Перетворювач) або DAC (Digital to Analog Convertor) призначений для перетворення цифрового сигналу, заданого у вигляді двійкового N-розрядного коду, у відповідну напругу або струм. Якість перетворення залежить від швидкості перетворення і розрядності ЦАП.

Характеристики і класифікація ЦАП

Залежність вихідної напруги $U_{вих}$ від вхідного цифрового сигналу D що змінюється від 0 до 2^{N-1} називають характеристикою перетворення ЦАП. Ця характеристика може бути представлена у вигляді східчастої кривої. Величина сходинки відповідає одиниці молодшого значущого розряду (МЗР).

Під час відсутності апаратних погрішностей середні точки сходів розташовані на прямій 1, якій відповідає ідеальна характеристика перетворення (рис.1.).

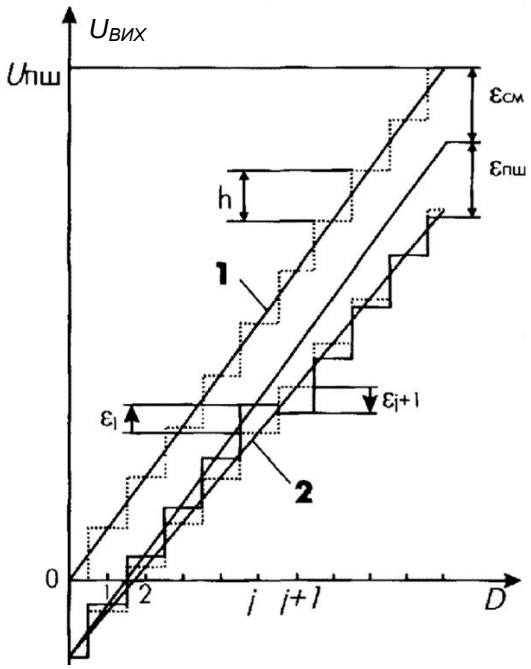


Рисунок 1 - Характеристика перетворення ЦАП

Реальна характеристика перетворення може відрізнятися від ідеальної розташуванням, розмірами і формою сходів. Для кількісного опису цих відмінностей використовуються характеристики, що називаються статичними. Зміни вихідного сигналу в часі при стрибкоподібній зміні вхідного коду від мінімуму («усі нулі») до максимуму («усі одиниці») описуються динамічними характеристиками ЦАП.

Статичні характеристики ЦАП

1. Розрядність. Число розрядів (двійкових символів) N, що передають кодований вхідний сигнал. Йому відповідає число рівнів квантування 2^N.

2. Число каналів. Число аналогових виходів ЦАП, на яких формується вихідна напруга. Залежно від схеми побудови цифровий код для кожного з каналів подається або на різні цифрові входи ЦАП, або на той самий з поділом за часом. Як правило, частота перетворення і напруга повної шкали для всіх каналів ЦАП рівні.

Однак у деяких моделях ИМС ЦАП напруга повної шкали для кожного каналу задається окремо і залежить від опорної напруги.

3. Розв'язна здатність. Збільшення $U_{вих}$ при збільшенні коду на одиницю називається кроком квантування. Номінальне значення кроку квантування становить:

$$h = \frac{U_{ref}}{2^N - 1} \quad (1)$$

де U_{ref} - номінальна максимальна вихідна напруга ЦАП (напруга повної шкали),

N - розрядність ЦАП. Чим більше розрядність перетворювача, тем вище його розв'язна здатність (менше h).

4. Погрішність повної шкали. Відносна різниця між реальним і ідеальним значеннями межі шкали перетворення при відсутності зсуву нуля:

$$\delta_{ref} = \frac{\varepsilon_{ref}}{U_{ref}} 100\% \quad (2)$$

Вона є мультиплікативною складовою повною погрішності. Іноді виражається відповідним числом МЗР.

Динамічні характеристики ЦАП

1. Частота перетворення. Максимальна частота, з якою ЦАП може формувати вихідну напругу або струм, відповідний до вхідного коду. Частота перетворення - це основний параметр, що характеризує швидкодію ЦАП.

2. Час встановлення. Під цією величиною розуміють інтервал часу від

моменту зміни вхідного коду (t = 0) до моменту, коли востаннє виконується рівність (d - допуск на вихідну напругу або струм ЦАП):

$$|U_{вих} - U_{ref}| = \frac{d}{2} \quad (3)$$

Зазвичай цей допуск дорівнює половині кроку квантування h.

3. Швидкість наростання. Це максимальна швидкість зміни $U_{вих}$ під час перехідного процесу, яка визначається як відношення збільшення $\Delta U_{вих}$ до часу At, за який відбулося це збільшення (рис. 2). Зазвичай вказується в технічних характеристиках ЦАП з вихідним сигналом у вигляді напруги.

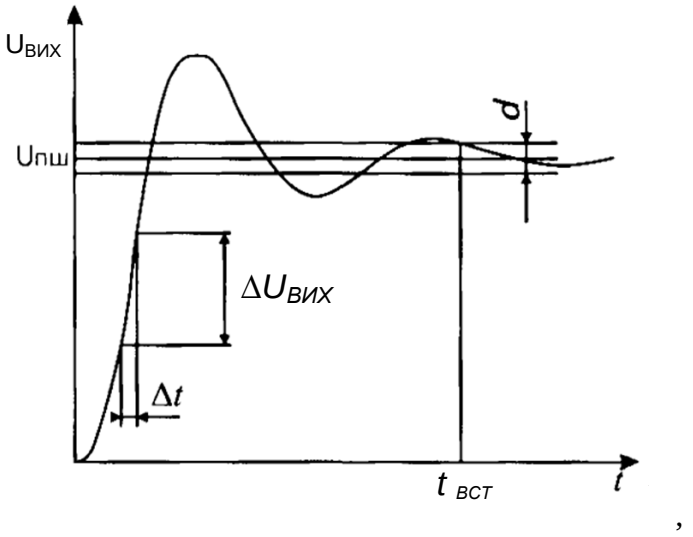


Рисунок 2 - Визначення часу встановлення і швидкості наростання

4. Шуми ЦАП. Шум на виході ЦАП може з'являтися по різних причинах, що викликаються фізичними процесами, які відбуваються в напівпровідникових пристроях. Для оцінки якості ЦАП з високою розв'язною здатністю прийнято використовувати поняття середньоквадратичного значення шуму. Спектральну щільність шуму вимірюють зазвичай в заданій полосі частот.

5. Викиди (імпульсні перешкоди). Викиди являють собою круті короткі сплески чи провали у вихідній напрузі, що виникають під час зміни значень вихідного коду за рахунок не синхронності розмикання і замикання аналогових ключів у різних розрядах ЦАП.

Наприклад, якщо при переході від значення коду 011...111 до значення 100...000 ключ самого старшого розряду ЦАП з підсумовуванням вагових струмів відкриється пізніше, чим закриються ключі молодших розрядів, то на виході ЦАП якийсь час буде існувати сигнал, відповідний до коду 000...000.

Класифікація ЦАП

Класифікація ЦАП за схемотехнічними ознаками представлена на рис. 3. За принципом роботи є дві основні групи ЦАП: послідовні і паралельні. Перевагою послідовних ЦАП є простота схемотехнічної реалізації, недоліком - невисока швидкодія.

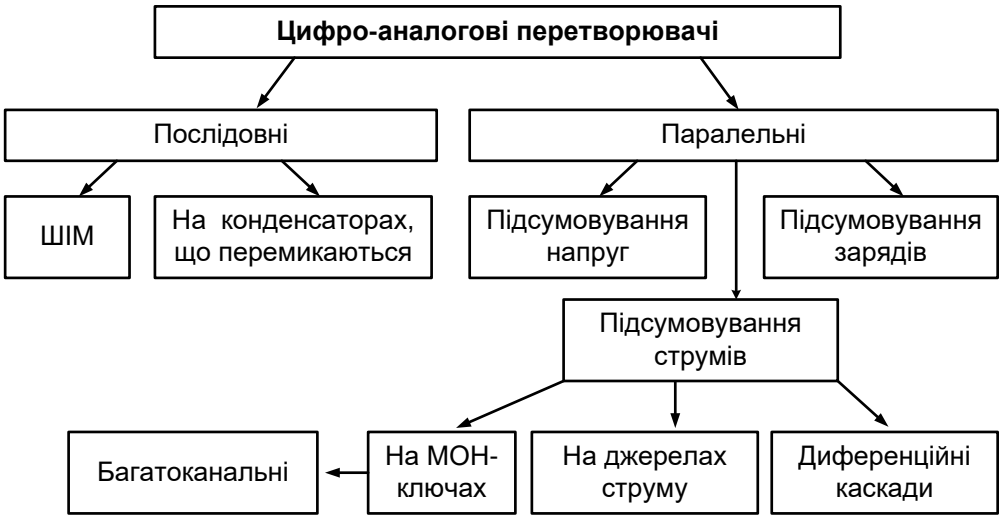


Рисунок 3 – Класифікація ЦАП по схемотехнічним ознакам

Паралельні ЦАП забезпечують максимально можливу швидкодію, але це досягається за рахунок значного ускладнення схеми в порівнянні з послідовними ЦАП.

ЦАП можуть також класифікуватися по:

- вигляду вихідного сигналу: зі струмовим виходом; з виходом у вигляді напруги;
- полярності вихідного сигналу: однополярні; біполярні;
- характеру опорного сигналу: з постійним опорним сигналом; з мінливим опорним сигналом; що множать;
- швидкодії: помірна швидкодія; висока швидкодія;

- типу цифрового інтерфейсу (введення вихідного коду): з послідовним введенням; з паралельним введенням;
- числу ЦАП на кристалі: одноканальні; багатоканальні.

Відлік шкали перетворення проводиться відносно опорної напруги Uref. Наприклад, якщо Uref = 10В, а розрядність ЦАП = 8 розрядів (255 рівнів), то ціна (вага) одного розряду складе 10В/255 = 0.039В або 39 мВ.

Зазвичай Uref встановлюють рівним напрузі живлення ЦАП, тобто, Uref = VDD = 5В.

Двійковий код в еквіваленті вихідної напруги ЦАП знаходять за формулою:

$$HEX = U_{вих} / DAC\_CVANT,$$

де: $DAC\_CVANT = Uref / 2^{N-1}$

Внутрішня структура ЦАП MCP 4921 представлена на рис. 4.

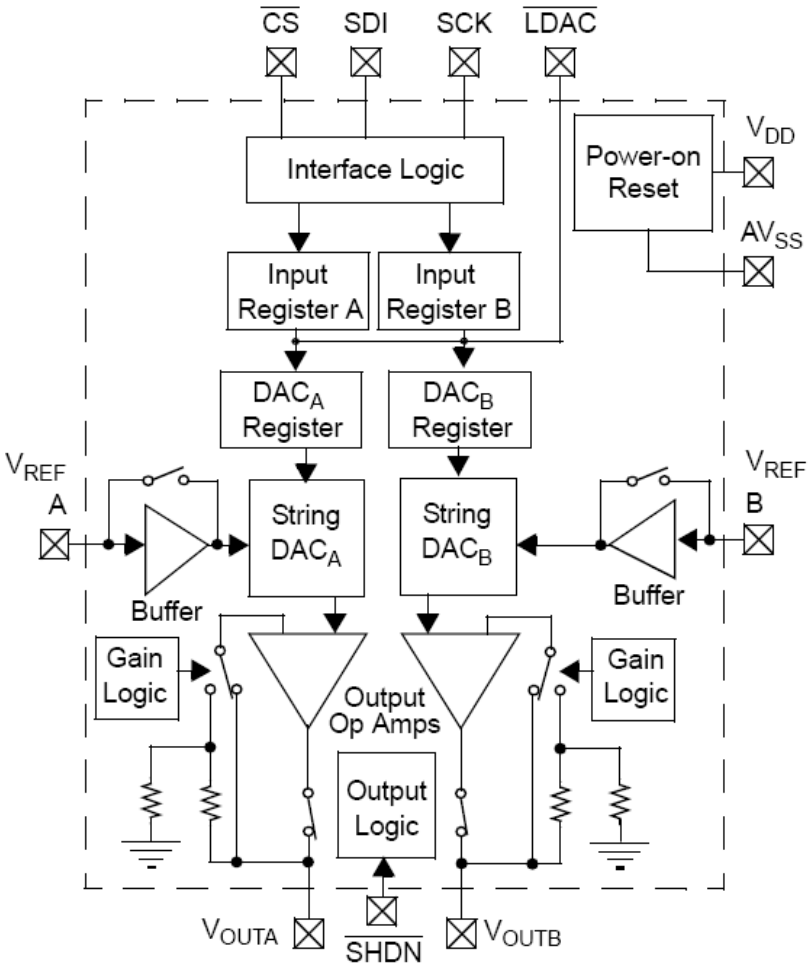


Рисунок 4 – Внутрішня структура ЦАП MCP 4921

Розташування виводів ЦАП MCP 4921 представлено на рис. 5.

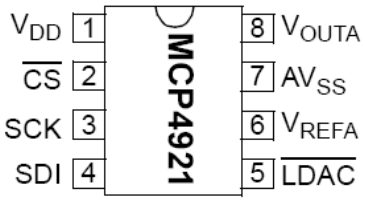


Рисунок 5 – ЦАП MCP 4921

Таблиця 1 – Призначення виводів ЦАП MCP 4921

MCP4921 Pin No.	MCP4922 Pin No.	Symbol	Function
1	1	V_{DD}	Positive Power Supply Input (2.7V to 5.5V)
—	2	NC	No Connection
2	3	$\overline{CS}$	Chip Select Input
3	4	SCK	Serial Clock Input
4	5	SDI	Serial Data Input
—	6	NC	No Connection
—	7	NC	No Connection
5	8	$\overline{LDAC}$	Syncronization input used to transfer DAC settings from serial latches to the output latches.
—	9	$\overline{SHDN}$	Hardware Shutdown Input
—	10	V_{OUTB}	DAC_B Output
—	11	V_{REFB}	DAC_B Voltage Input (AV_{SS} to V_{DD})
7	12	AV_{SS}	Analog ground
6	13	V_{REFA}	DAC_A Voltage Input (AV_{SS} to V_{DD})
8	14	V_{OUTA}	DAC_A Output

Підключення ЦАП до мікроконтролера по інтерфейсу SPI представлено на рис.6.

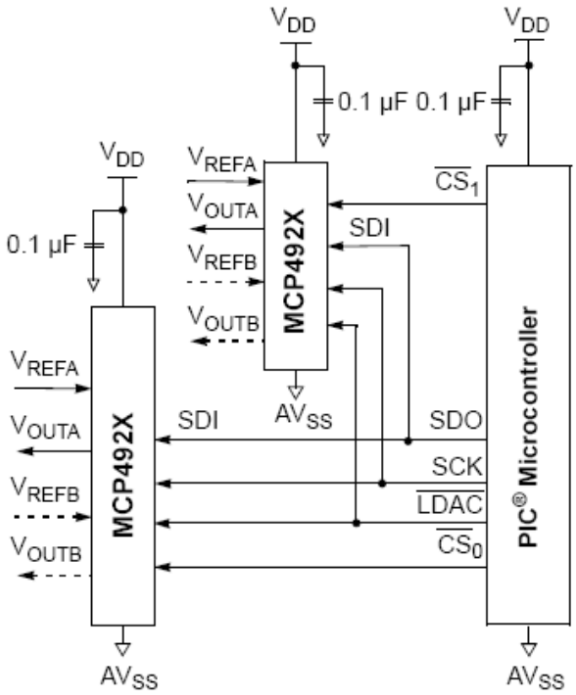


Рисунок 6 – Підключення ЦАП до мікроконтролера по інтерфейсу SPI

Тимчасові діаграми роботи інтерфейсу SPI представлені на рис.7.

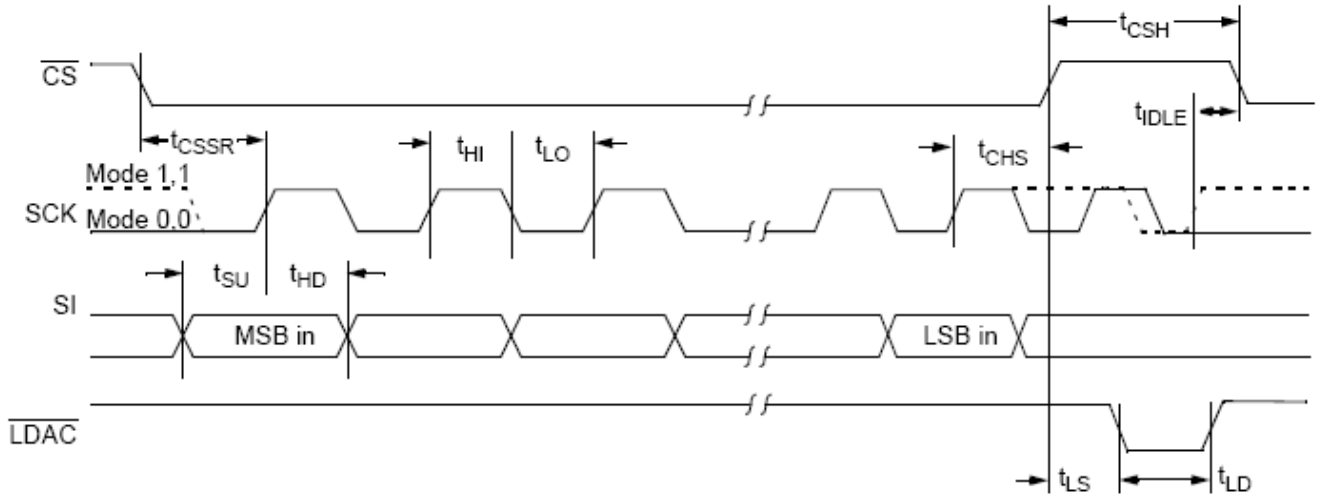


Рисунок 7 – Тимчасові діаграми роботи інтерфейсу SPI

Таблиця 2 – Тимчасові характеристики роботи інтерфейсу SPI

Electrical Specifications: Unless otherwise indicated, V_{DD}= 2.7V – 5.5V, T_A= -40 to +125°C. Typical values are at +25°C.						
Parameters	**Sym**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
Schmitt Trigger High-Level Input Voltage (All digital input pins)	V_{IH}	0.7 V_{DD}	—	—	V	
Schmitt Trigger Low-Level Input Voltage (All digital input pins)	V_{IL}	—	—	0.2 V_{DD}	V	
Hysteresis of Schmitt Trigger Inputs	V_{HYS}	—	0.05 V_{DD}	—		
Input Leakage Current	$I_{LEAKAGE}$	-1	—	1	µA	$\overline{SHDN}$ = $\overline{LDAC}$ = $\overline{CS}$ = SDI = SCK + V_{REF} = V_{DD} or AV_{SS}
Digital Pin Capacitance (All inputs/outputs)	C_{IN}, C_{OUT}	—	10	—	pF	V_{DD} = 5.0V, T_A = +25°C, f_{cLK} = 1 MHz
Clock Frequency	F_{CLK}	—	—	20	MHz	T_A = +25°C
Clock High Time	t_{HI}	15	—	—	ns	
Clock Low Time	t_{LO}	15	—	—	ns	
$\overline{CS}$ Fall to First Rising CLK Edge	t_{CSSR}	40	—	—	ns	Applies only when $\overline{CS}$ falls with CLK high.
Data Input Setup Time	t_{SU}	15	—	—	ns	
Data Input Hold Time	t_{HD}	10	—	—	ns	
SCK Rise to $\overline{CS}$ Rise Hold Time	t_{CHS}	15	—	—	ns	
$\overline{CS}$ High Time	t_{CSH}	15	—	—	ns	
$\overline{LDAC}$ Pulse Width	t_{LD}	100	—	—	ns	
$\overline{LDAC}$ Setup Time	t_{LS}	40	—	—	ns	
SCK Idle Time before $\overline{CS}$ Fall	t_{IDLE}	40	—	—	ns	

Процес пересилання пакета даних від контролера до ЦАП по інтерфейсу SPI представлений на рис. 8.

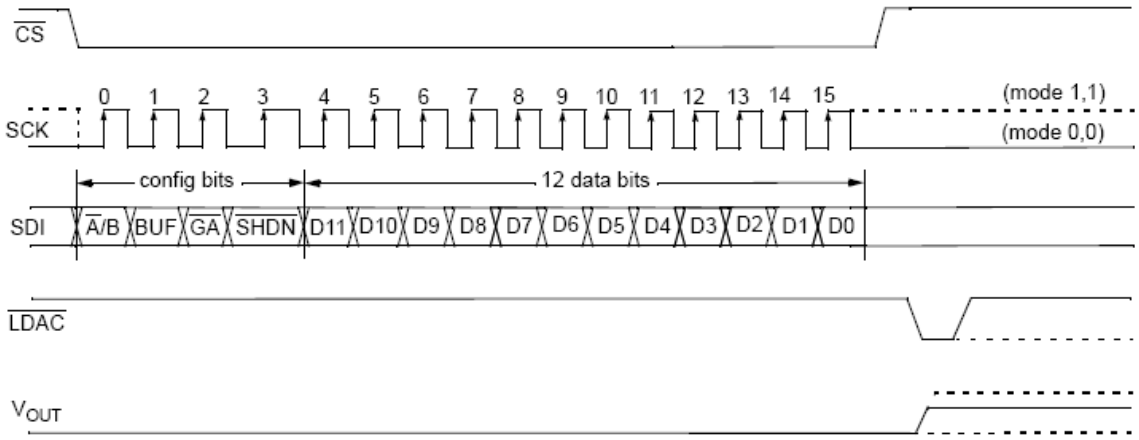
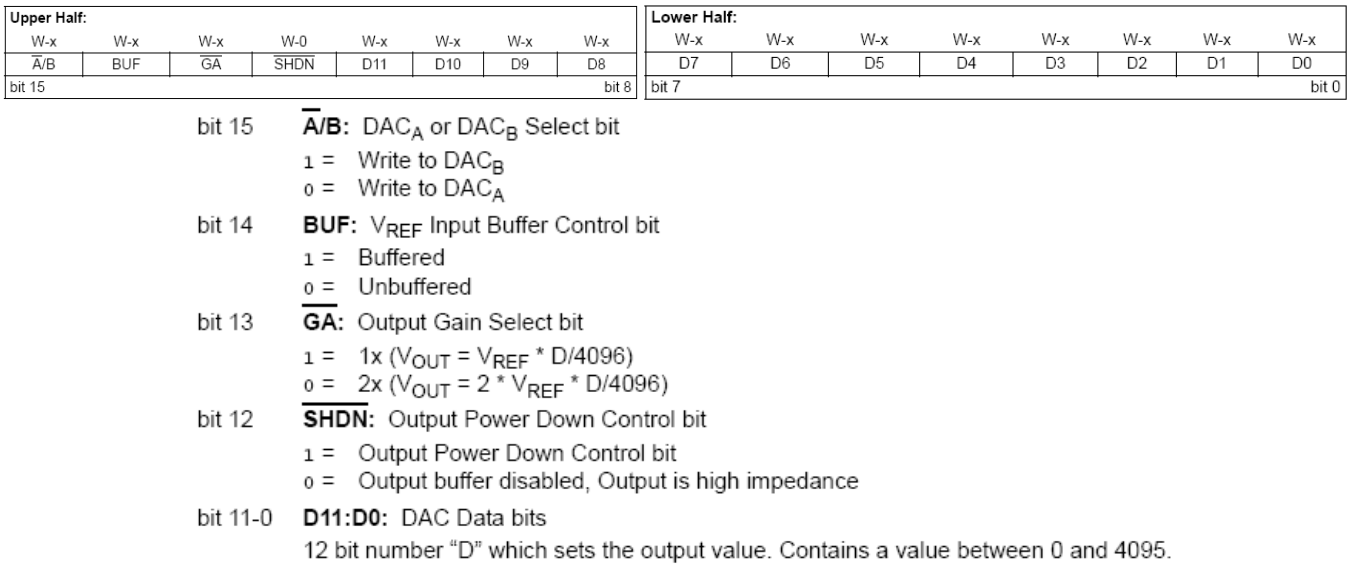


Рисунок 8 – Процес пересилання пакета даних від контролера до ЦАП по інтерфейсу SPI

Формат пакета даних представлений на рис. 9.



Upper Half:							
W-x	W-x	W-x	W-0	W-x	W-x	W-x	W-x
$\overline{A}$/B	BUF	$\overline{GA}$	$\overline{SHDN}$	D11	D10	D9	D8
bit 15							bit 8

Lower Half:							
W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
D7	D6	D5	D4	D3	D2	D1	D0
bit 7							bit 0

bit 15 **$\overline{A}$/B:** DAC_A or DAC_B Select bit
1 = Write to DAC_B
0 = Write to DAC_A

bit 14 **BUF:** V_{REF} Input Buffer Control bit
1 = Buffered
0 = Unbuffered

bit 13 **$\overline{GA}$:** Output Gain Select bit
1 = 1x ($V_{OUT} = V_{REF} * D/4096$)
0 = 2x ($V_{OUT} = 2 * V_{REF} * D/4096$)

bit 12 **$\overline{SHDN}$:** Output Power Down Control bit
1 = Output Power Down Control bit
0 = Output buffer disabled, Output is high impedance

bit 11-0 **D11:D0:** DAC Data bits
12 bit number "D" which sets the output value. Contains a value between 0 and 4095.

Рисунок 9 – Формат пакета даних ЦАП

Конфігурування мікроконтролера для роботи із ЦАП по інтерфейсу SPI можна зробити з IDE при створенні проекту або у функції ініціалізації ввести рядок:

```
setup_spi(SPI_MASTER|SPI_L_TO_H|SPI_CLK_DIV_4);
```

Для даної лабораторної роботи це робити не рекомендується з тієї причини, що у компіляторі підтримується робота з інтерфейсу SPI тільки в однобайтовому режимі, що добре при роботі із зовнішньою пам'яттю та іншими пристроями, але для ЦАП MCP 4921 потрібно двохбайтова передача пакета.

Тому написання драйвера інтерфейсу SPI для роботи у двохбайтовому режимі є завданням лабораторної роботи.

Методичні вказівки:

Порядок роботи для варіанту «Навчальний комплекс»

1. Намалювати принципову схему підключень у відповідності зі структурною і функціональною схемою (рис. 10,11).
2. Підключити потенціометр, вольтметр АЦП і ЦАП відповідно до варіанта.

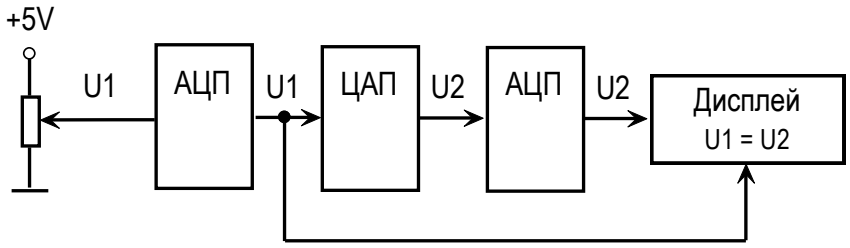


Рисунок 10 - Структурна схема підключень

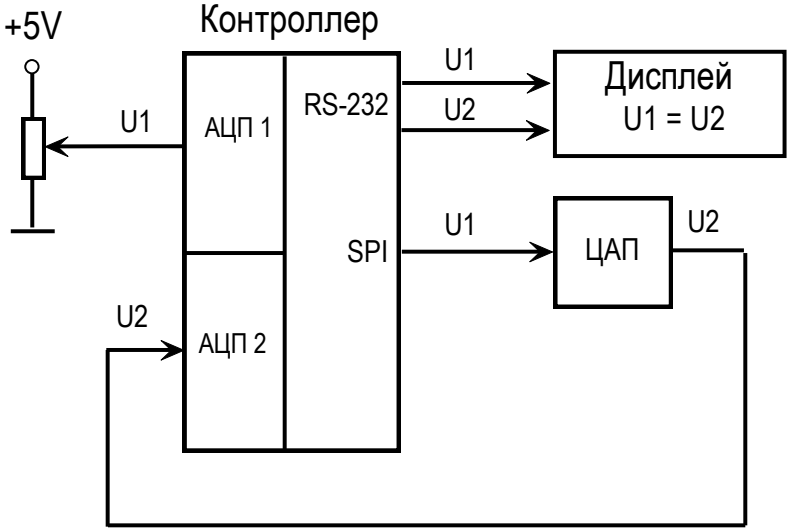


Рисунок 11 - Функціональна схема підключень

3. Створити алгоритм програми роботи з ЦАП по інтерфейсу SPI.
4. Написати програму, що реалізує створений алгоритм.
5. Відкомпілювати програму, записати в мікроконтролер, виконати.

Порядок роботи для варіанту «Proteus»:

1. Завантажити з сервера папку «Proteus_students».
2. Відкрити середовище розробки PCWH.
3. У папці «Proteus_students» вибрати папку відповідної лабораторної роботи (LAB1... LAB8).
4. Відкрити файл проекту з розширенням *. Pjt.
5. У редакторі відкрити шаблон файлу програми з розширенням *. C.
6. Відкрити середу моделювання «Proteus».
7. У папці «Proteus_students» вибрати папку відповідної лабораторної роботи (LAB1... LAB8).
8. Відкрити файл проекту з розширенням *. DSN.
9. У середовищі розробки PCWH , використовуючи шаблон програми самостійно написати, відкомпілювати програму у відповідності з завданням.
10. У середовищі розробки «Proteus» виконати програму.

Примітка: Файл демонстрацій виконання лабораторної роботи розташований в папці «Demo».

Принципова схема для виконання лабораторної роботи для варіанту «Proteus»

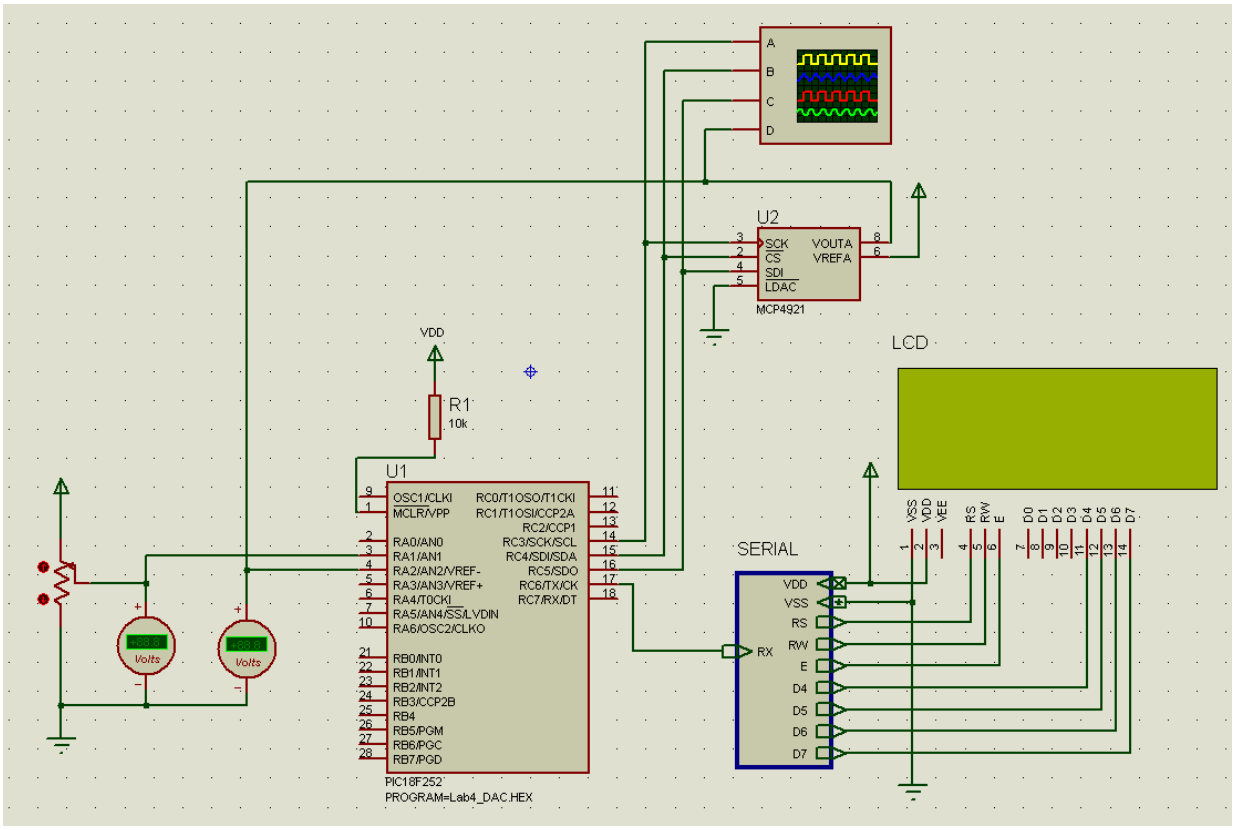


Рисунок 12 - Принципова схема для виконання лабораторної роботи для варіанту «Proteus»

Результат виконання програми:

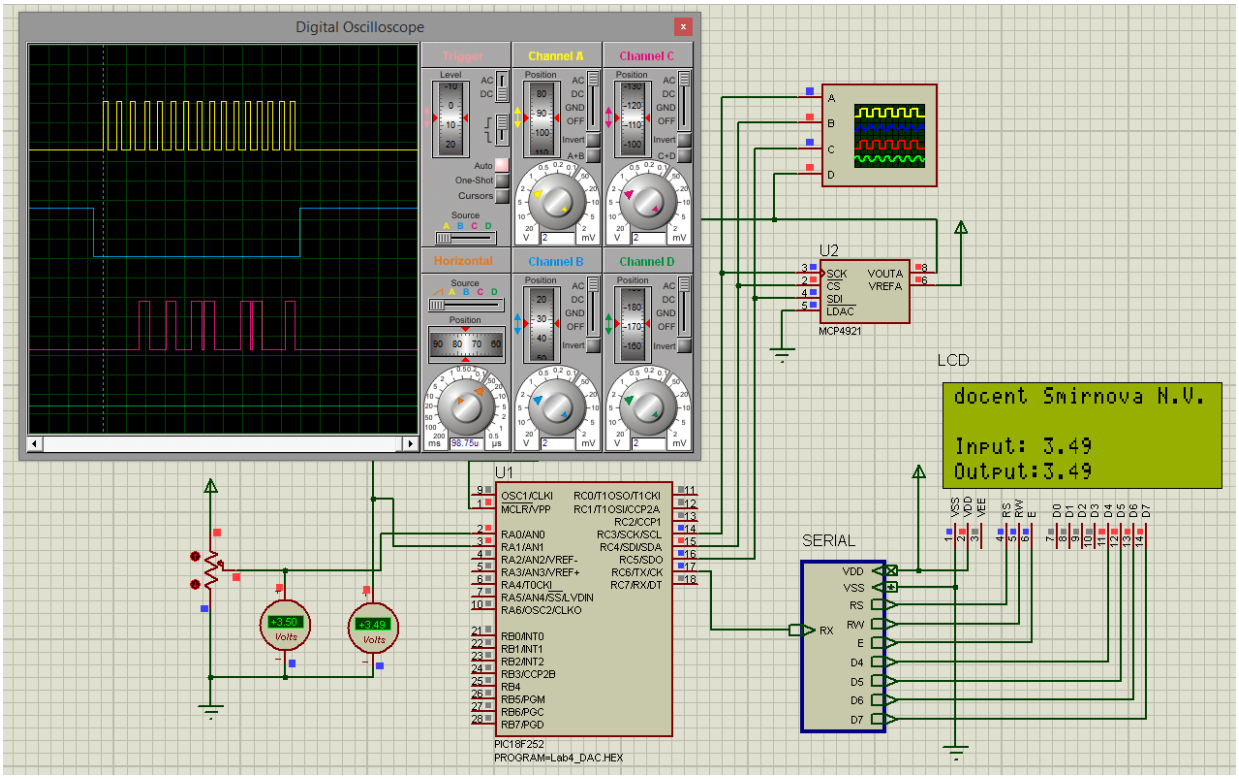


Рисунок 13 - Результат виконання лабораторної роботи для варіанту «Proteus»

Приклад зчитування даних з АЦП з каналу 1 і передача даних у ЦАП:

```
float volt_in;          //вихід потенціометра
float volt_out;         //вихід ЦАП
int16 dac_hex;          //вхід ЦАП

//====== послати значення в ЦАП
dac_hex = volt_in/DAC_CVANT;  //двійковий еквівалент напруги
SPI_send(dac_hex);            //послати дані в ЦАП
```

Змінна `DAC_CVANT` є ціною розподілу шкали ЦАП і обчислюється як частка від ділення Uref на розрядність АЦП, виражену у двійкових одиницях:

```
//====== опорна напруга = напруга живлення
const float REF_VDD = 5.00;
const float DAC_12 = 4095;

//====== розподіл шкали: 5В/шкала (розрядність ЦАП)
const float  DAC_CVANT = REF_VDD/DAC_12;
```

Визначення двійкового значення величини `dac_hex` напруги Uвих, що встановлюється:

```
dac_hex = volt_in/DAC_CVANT;
```

Примітка: після виведення значень на LCD необхідно ввести затримку часу індикації: `delay_ms(100);`

Послідовність побудови інтерфейсу SPI:

1. Встановити виводи контролера, що беруть участь в роботі з SPI у режим передачі даних.
2. Визначити інтерфейс:

```
#define CLK  PIN_C3
#define CS   PIN_C4
#define SDO  PIN_C5
```

3. Об′явити змінні для лічильника розрядів і затримки. Затримку прийняти рівною 25 мкс.
4. У пакет даних, що передається, вставити біт режиму `0x2000;`
5. Перед початком передачі даних встановити сигнал CS в ‘0’.
6. Почати передачу пакета даних у циклі зі зсувом вліво. При передачі послідовно виставляти дані, затримки і строб відповідно до тимчасової діаграми на рис. 7.

```
//=== виставити біт
if(bit_test(val,c)){    //c – номер біта 0-15
   output_high(SDO);    //SDO = 1
}else{
   output_low(SDO);     //SDO = 0
}
```

7. Після відправлення останнього біта встановити сигнал CS в ‘1’ і виконати затримку.

На виході ЦАП з'явиться рівень напруги пропорційний заданому двійковому значенню.

Прототип функції: `void SPI_send(int16 val);`

Контрольні питання:

1. Призначення ЦАП і методи перетворення.
2. Статичні і динамічні характеристики ЦАП.
3. Призначення ліній інтерфейсу SPI.
4. Структура пакета даних ЦАП MCP 4921 (SPI).
5. Як визначається величина двійкового еквівалента необхідної вихідної напруги ЦАП?

Зміст звіту:

У звіті повинні бути представлені:

- принципова схема з'єднань (роздруківка виводів контролера і від руки намальовані з'єднання для варіанту «Навчальний комплекс»), результат роботи для варіанту «Proteus».
- блок-схема алгоритму роботи програми.
- лістинг програми.
- висновки за результатами роботи.

Варіанти

	Канал АЦП							
	Потенціометр				ЦАП			
№	A0	A1	A2	A3	A0	A1	A2	A3
1	+					+		
2		+					+	
3			+					+
4				+	+			
5		+			+			
6			+			+		
7	+						+	
8	+							+
9		+					+	
10			+		+			
11				+	+			
12	+					+		
13		+			+			
14			+					+
15				+			+	

Лабораторна робота № 5

Тема: Динамічна індикація. Виведення інформації на LED – дисплей

Ціль роботи

Одержання навичок роботи із багаторозрядними LED - дисплеями і написання драйвера динамічної індикації для LED - дисплея.

Завдання:

- Намалювати принципову схему підключень відповідно до варіанта.
- Для варіанту «Proteus» використовувати готову схему
- Написати програму драйвера динамічної індикації.
- Здійснити виведення 3-х значного числа лічильника секунд на LED - дисплей відповідно до варіанта.

Виведення на LCD повинне містити:

- Групу, прізвище та ініціали студента.

Теоретичні відомості:

У мікроконтролерних системах керування одним з обов'язкових вузлів є підсистема індикації. Вона може складатися з одного світлодіода або декількох динамічних 7- сегментних дисплеїв.

У кожному разі необхідно виконати розрахунки допустимих струмів, що протікають через світлодіоди індикатору.

Розрахункова схема підключення світлодіода представлена на рис. 1.

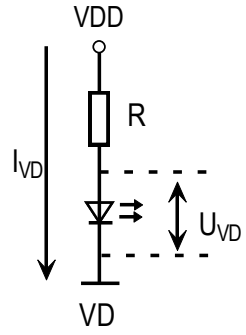


Рисунок 1 - Розрахункова схема підключення світлодіода

Зазвичай V_{DD} становить 5 В, а спадання напруги на світлодіоді Uvd – 2,5 В. Робочий струм світлодіода становить у середньому 5 мА.

Опір резистора R1 визначається виразом:

$$R = (VDD-U_{vd})/I_{vd} \quad (1)$$

Для V_{DD} = 5В опір R буде рівним:

$$К = (5-2,5)/0.005 = 500 \text{ Ом}$$

Найближчий номінал резистора – 510 ом.

Приклади підключення світлодіодів до мікроконтролера представлені на рис.2.

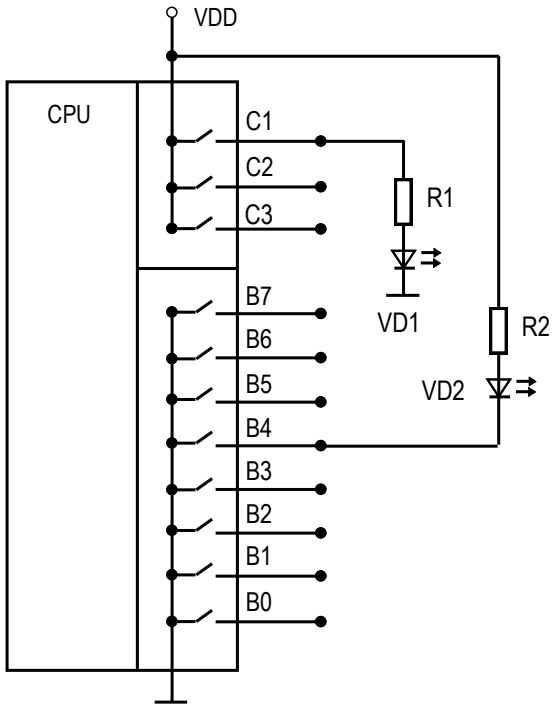


Рисунок 2 - Приклади підключення світлодіодів до мікроконтролера

LED – дисплей являє собою матрицю знакомісць, у кожному з них розташовано 8 світлодіодів – 7 інформаційних і 1 – десяткова крапка (рис 3).

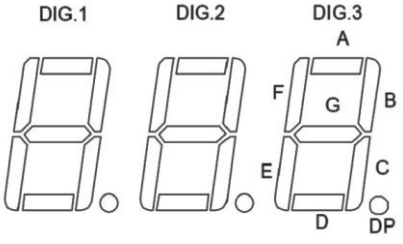


Рисунок 3 – LED – дисплей

Світлодіоди у дисплеях можуть бути включені за схемою із загальним катодом (рис.4) або із загальним анодом (рис.5)

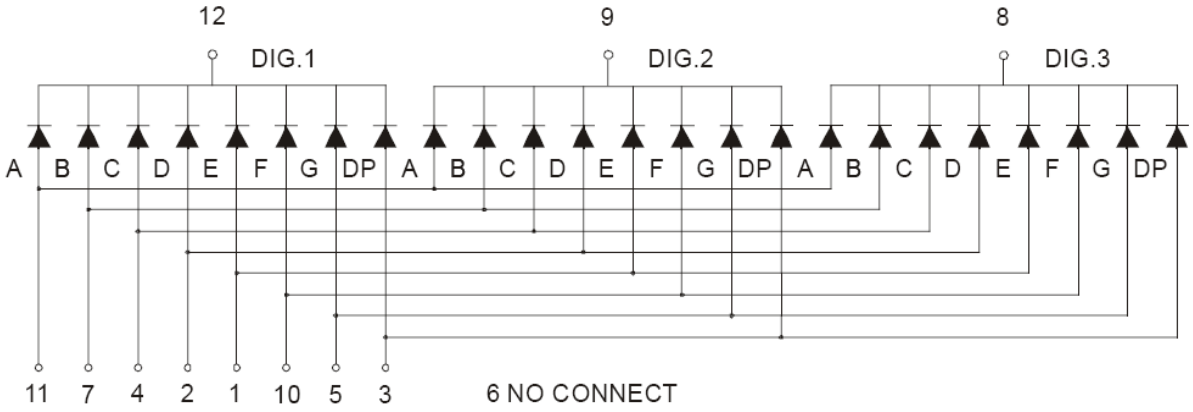


Рисунок 4 – Схема із загальним катодом

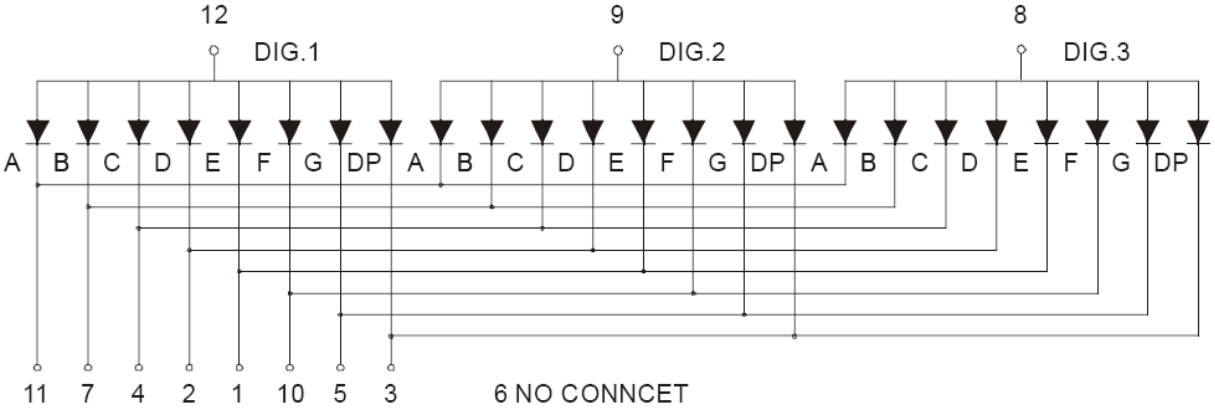


Рисунок 5 – Схема із загальним анодом

Схема підключення LED - дисплея до контролера представлена на рис. 6.

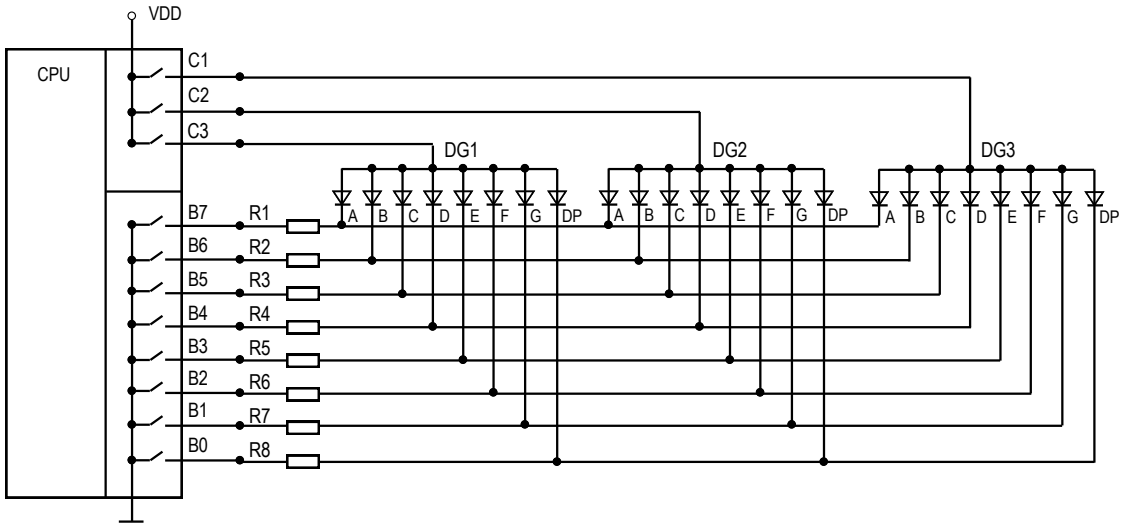


Рисунок 6 – Підключення LED - дисплея до контролера

Тимчасова діаграма роботи дисплея у динамічному режимі представлена на рис.7.

У цьому режимі послідовно змінюються дані на інформаційних входах дисплея (A..G) і кожне знакомісце активується подачею логічної «1» на керуючий вхід (DG1..DG3).

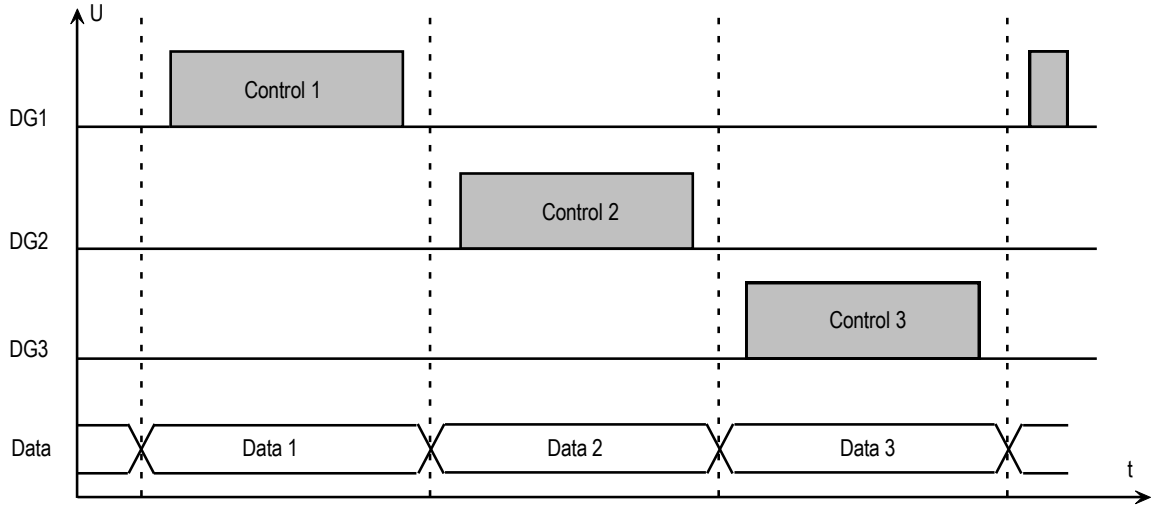


Рисунок 7 – Тимчасова діаграма роботи дисплея у динамічному режимі

Методичні вказівки до виконання лабораторної роботи:

1. Намалювати принципову схему підключень (рис. 6) відповідно до варіанта
2. Заповнити таблицю 1 кодами цифр у 7-сегментному представленні

Таблиця 1 - 7-сегментне представлення десяткових цифр

Цифра	A	B	C	D	E	F	G	DP
0	0	0	0	0	0	0	1	1
1	1	0	0	1	1	1	1	1
…								
8	0	0	0	0	0	0	1	1
9	0	0	0	0	1	0	0	1
DP	1	1	1	1	1	1	1	0

3. Вивести на LED – дисплей значення лічильника в інтервалі N_1 – N_2 з періодом в 1 сек.

Значення N_1 і N_2:

N_1 = номер за списком * 5;

N_2 = 150 – номер за списком.

4. Створити алгоритм програми роботи з LED - дисплеєм.
5. Написати програму, що реалізує створений алгоритм.
6. Відкомпілювати програму, записати в мікроконтролер, виконати.

Примітка: Відображення цифр на LED - дисплеї можна здійснювати декількома варіантами:

a. конструкцією if-then;
b. конструкцією switch;
c. табличним методом і т.д.

Порядок роботи для варіанту «Proteus»:

1. Завантажити з сервера папку «Proteus_students».

2. Відкрити середовище розробки PCWH.

3. У папці «Proteus_students» вибрати папку відповідної лабораторної роботи (LAB1... LAB8).

4. Відкрити файл проекту з розширенням *. Pjt.

5. У редакторі відкрити шаблон файлу програми з розширенням *. C.

6. Відкрити середу моделювання «Proteus».

7. У папці «Proteus_students» вибрати папку відповідної лабораторної роботи (LAB1... LAB8).

8. Відкрити файл проекту з розширенням *. DSN.

9. У середовищі розробки PCWH , використовуючи шаблон програми самостійно написати, відкомпілювати програму у відповідності з завданням.

10. У середовищі розробки «Proteus» виконати програму.

Примітка: Файл демонстрацій виконання лабораторної роботи розташований в папці «Demo».

Принципова схема для виконання лабораторної роботи для варіанту «Proteus»

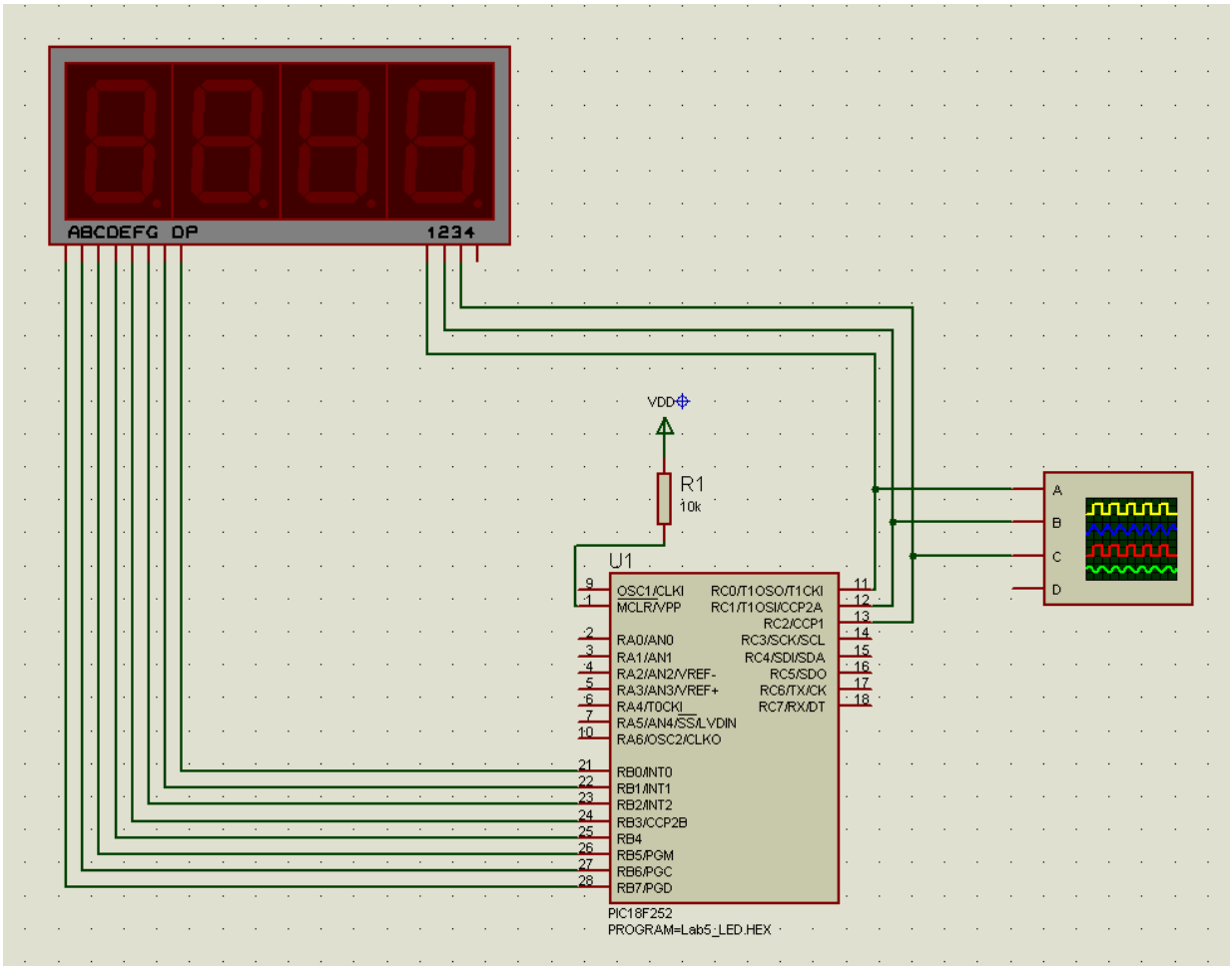


Рисунок 8 - Принципова схема для виконання лабораторної роботи для варіанту «Proteus»

Результат виконання програми:

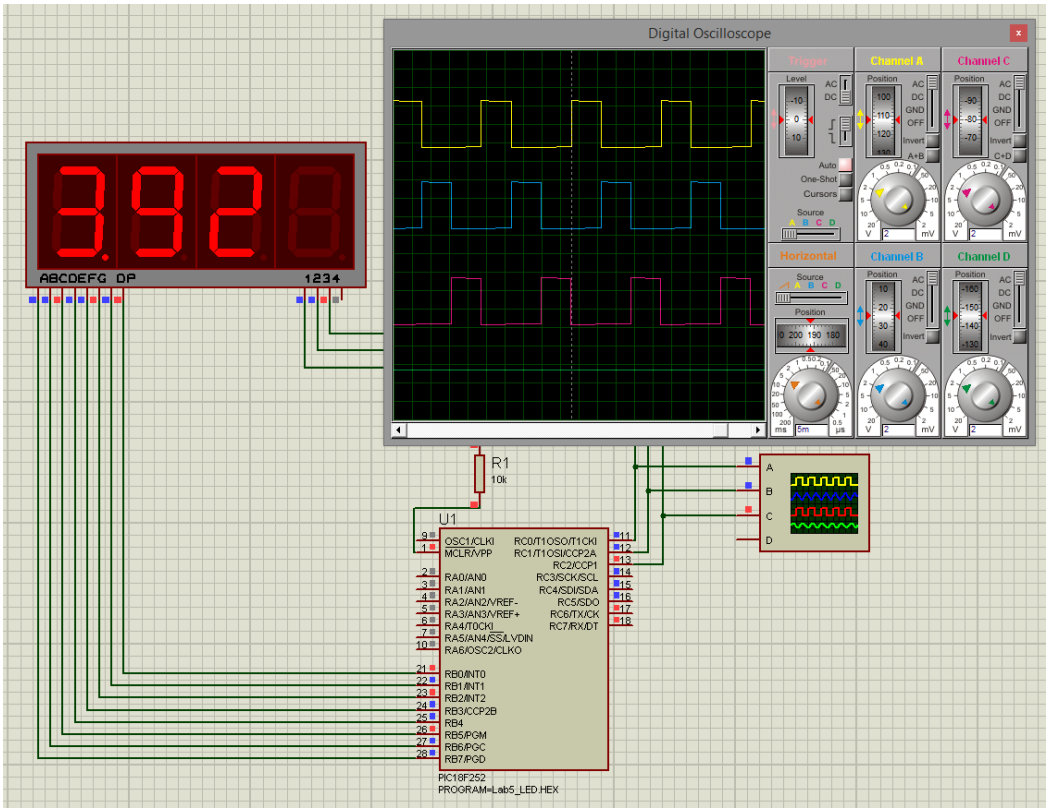


Рисунок 9- Результат виконання лабораторної роботи для варіанту «Proteus»

Приклад застосування конструкції if-then:

```
//=== відобразити 2-е знакомісце
   output_c(D_2);                    //включити знакомісце

 if(digit == '1'){
      output_b(LED_digit);           //відобразити значення
 }
```

Приклад застосування конструкції switch:

```
switch(digit){
      ...
//=== відобразити 2-е знакомісце
output_c(D_2);                       //включити знакомісце

case '1':
      output_b(LED_digit);           //відобразити значення
break;
      ...
```

Приклад застосування конструкції табличного методу:

```
//=== індекс масиву 7 сегментних кодів
idx = atoi(str_idx);
//=== читати з масиву 7 сегментний код числа
digit_out = LED[idx];

//=== відобразити знакомісце
output_c(D_2);              //включити знакомісце
  output_b(digit_out);       //відобразити значення
  //=== час відображення
  delay_ms(delay);
```

Алгоритм відображення цифр на дисплеї:

1. Зчитати із заданого числа один розряд;
2. Перетворити в 7-сегментний код;
3. Виключити попереднє знакомісце;
4. Вивести 7- сегментний код;
5. Включити поточне знакомісце;
6. Включити затримку для індикації знакомісця;
7. Перейти до п. 1.

Для виключення мерехтіння затримку слід вибирати в межах 3-10 мс.

Контрольні питання:

1. Схеми включення світлодіодів в LED-дисплеї.
2. Розрахункова формула для визначення опору резистора в ланцюзі живлення світлодіода.
3. Призначення виводів LED-дисплея.
4. Принцип динамічної індикації.
5. Перетворення десяткової цифри у 7-сегментний код.

Зміст звіту:

У звіті повинні бути представлені:

- принципова схема з'єднань (роздруківка виводів контролера і від руки

намальовані з'єднання для варіанту «Навчальний комплекс»), результат роботи для варіанту «Proteus».

- блок-схема алгоритму роботи програми.
- лістинг програми.
- висновки за результатами роботи.

Варіанти

	Число	**Керування**				**Дані**							
№	**N_1 і N_2**	**C0**	**C1**	**C2**	**C3**	**B0**	**B1**	**B2**	**B3**	**B4**	**B5**	**B6**	**B7**
1	$N_1 = № \cdot 5$ $N_2 = 150 - №$	D1	D2	D3		A	B	C	D	E	F	G	DP
2			D1	D2	D3	A	B	C	D	E	F	G	DP
3		D1	D2	D3		DP	G	F	E	D	C	B	A
4			D1	D2	D3	DP	G	F	E	D	C	B	A
5		D1	D2	D3		A	B	C	D	E	F	G	DP
6			D1	D2	D3	A	B	C	D	E	F	G	DP
7		D1	D2	D3		DP	G	F	E	D	C	B	A
8			D1	D2	D3	DP	G	F	E	D	C	B	A
9		D1	D2	D3		A	B	C	D	E	F	G	DP
10			D1	D2	D3	A	B	C	D	E	F	G	DP
11		D1	D2	D3		DP	G	F	E	D	C	B	A
12			D1	D2	D3	DP	G	F	E	D	C	B	A
13		D1	D2	D3		A	B	C	D	E	F	G	DP
14			D1	D2	D3	A	B	C	D	E	F	G	DP
15		D1	D2	D3		DP	G	F	E	D	C	B	A

Лабораторна робота № 6

Тема: Модуль CCP. Керування двигуном постійного струму

Ціль роботи

Одержання навичок роботи з ШІМ і керування навантаженням за допомогою ШІМ.

Завдання:

- Намалювати принципову схему підключень відповідно до варіанта.
- Для варіанту «Proteus» використовувати готову схему
- Написати програму роботи ШІМ модуля.
- Здійснити регулювання яскравості світіння світлодіода.
- Здійснити регулювання напрямку і швидкості обертання електродвигуна постійного струму.

Виведення на LCD повинне містити:

- Поточний режим роботи двигуна, значення скважності у форматі ##.## %.
- Групу, прізвище та ініціали студента.

Теоретичні відомості:

Модуль CCP (Capture, Compare, PWM - Pulse Width Modulation) дозволяє виконувати завдання виміру тимчасових параметрів вхідних сигналів, фіксації часу зміни фронтів сигналів, керування навантаженням та ін.

Розташування виводів CCP контролера представлене на рис. 1.

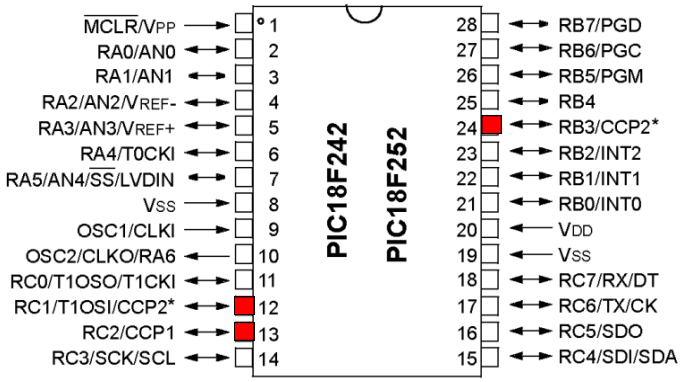


Рисунок 1 - Розташування виводів CCP контролера

При аналоговому способі керування, для зміни струму через навантаження необхідно змінити різницю потенціалів (напруга) на навантаженні. При такому способі регулювання на регуляторі виділяється значна теплова потужність, що обмежує область застосування таких регуляторів.

ШІМ – керування вільне від зазначеного недоліку, оскільки регулятор працює в ключовому режимі, тобто ключ перебуває в положенні «включене» або «виключене». У першому режимі струм, що протікає через ключ, не створює спадання напруги на відкритому ключі, отже, на ключі не виділяється теплова потужність.

У другому режимі струм через ключ не протікає, що визначає нульову потужність, яка споживається ключем.

Суть ШІМ – керування навантаженням полягає в наступному. На виході ШІМ – модуля формується прямокутний сигнал з постійним періодом проходження T (постійною частотою), як показано на рис. 2.

У кожному періоді ширина імпульсу τ може змінюватися від 0 до T, що відповідає скважності (коефіцієнту заповнення) від 0% до 100%.

Скважність імпульсів визначається по формулі:

$$S = \frac{100\% * \tau}{T} \tag{1}$$

Результуюча напруга $U_{вих}$ при інтегруванні, є лінійною функцією значення скважності S, і визначається формулою:

$$Uвих = \frac{Vdd * T}{\tau} \tag{2}$$

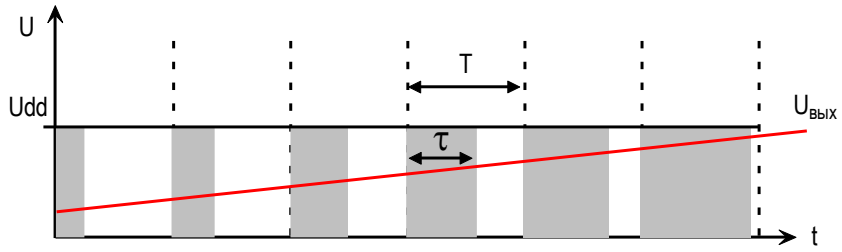


Рисунок 2 – Формування ШІМ і вихідної напруги

Принцип ШІМ-керування навантаженням (яскравістю світіння світлодіода) представлений на рис. 3. При подачі ШІМ-сигналу транзистор VT1 працює в ключовому режимі, тобто, відкривається при надходженні імпульсу і закривається

при його відсутності (транзистор включений за схемою з загальним емітером, тому на колекторі сигнал інверсний по відношенню до вхідного сигналу). Через світлодіод VD1 протікає струм, пропорційний тривалості імпульсу τ протягом періоду T. Оскільки частота ШІМ досить висока (1.2 кГц), то мерехтіння світлодіода непомітно.

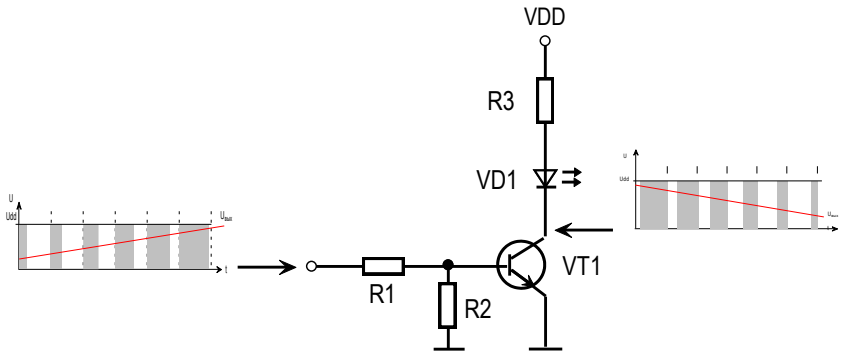


Рисунок 3 - Принцип ШІМ-керування навантаженням

Аналогічно здійснюється керування електродвигуном постійного струму. Оскільки крім швидкості обертання необхідно змінювати і напрямок обертання, то електродвигун включається в мостову схему, як показано на рис. 4.

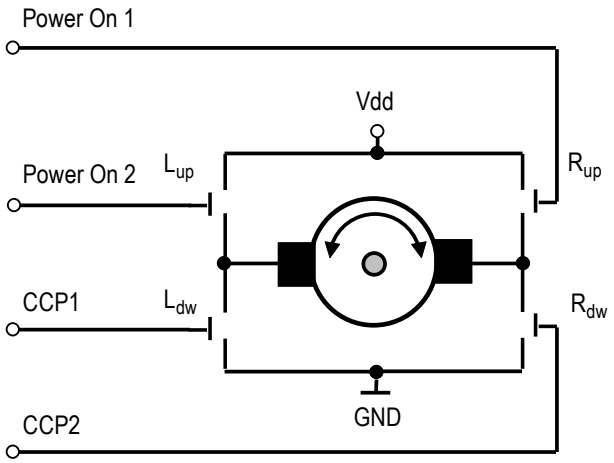


Рисунок 4 – Включення двигуна постійного струму в мостову схему

Керування напрямком обертання двигуна здійснюється в такий спосіб:

Обертання вліво:

- замкнути контакти в плечі моста Lup (подати сигнал керування Power On 2).
- замкнути контакти в плечі моста Rdw (подати сигнал керування CCP2).

Обертання вправо:

- замкнути контакти в плечі моста Rup (подати сигнал керування Power On 1).
- замкнути контакти в плечі моста Ldw (подати сигнал керування CCP1).

Швидкість обертання двигуна визначається тривалістю імпульсу τ протягом періоду T.

Конфігурування контролера для роботи з ШІМ

Вмикання модуля CCP:

```
setup_ccp1(CCP_PWM);
setup_ccp2(CCP_PWM);
```

Вимикання модуля CCP:

```
setup_ccp1(CCP_OFF);
setup_ccp2(CCP_OFF);
```

Модуль ШІМ працює з таймером timer_2. Параметри вихідного сигналу на виходах CCP1 і CCP2 контролера визначаються установками таймера timer_2 (рис.5).

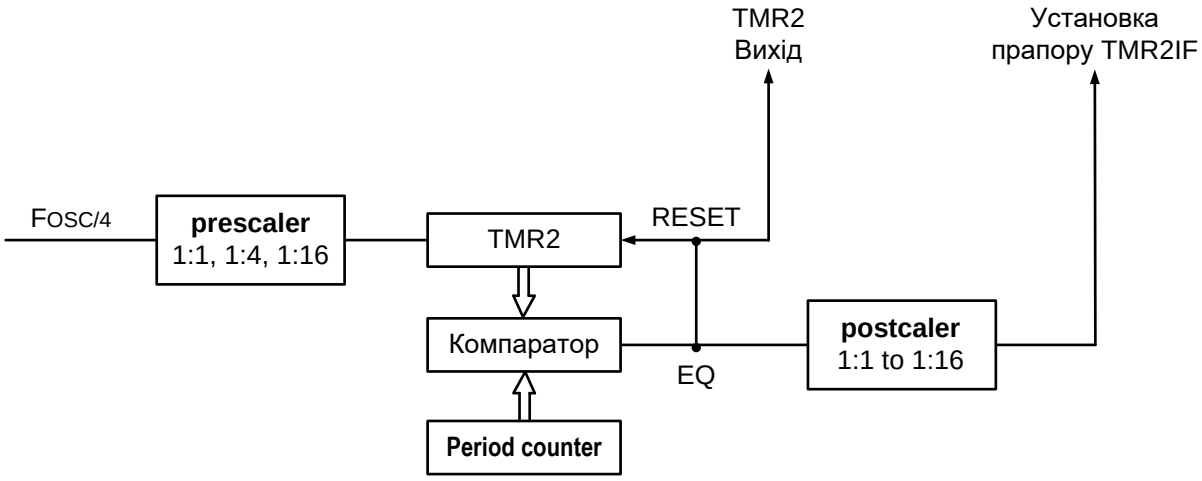


Рисунок 5 – Структурна схема таймера 2

Установка таймера:

Формат: `setup_timer_2(prescaler, period, postscaler)`, наприклад:

```
setup_timer_2(T2_DIV_BY_16,255,1);
```

Перший параметр пропорційно змінює період проходження T і тривалість імпульсу τ.

T2_DIV_BY_1 - Частота ШІМ = 19.5 khz, період = 51.2 us

T2_DIV_BY_4 - Частота ШІМ = 4.9 khz, період = 204.8 us

T2_DIV_BY_16 - Частота ШІМ = 1.2 khz, період = 833.3 us

Залежність параметрів вихідного сигналу CCP_X від значення параметра 1 представлена на рис.6.

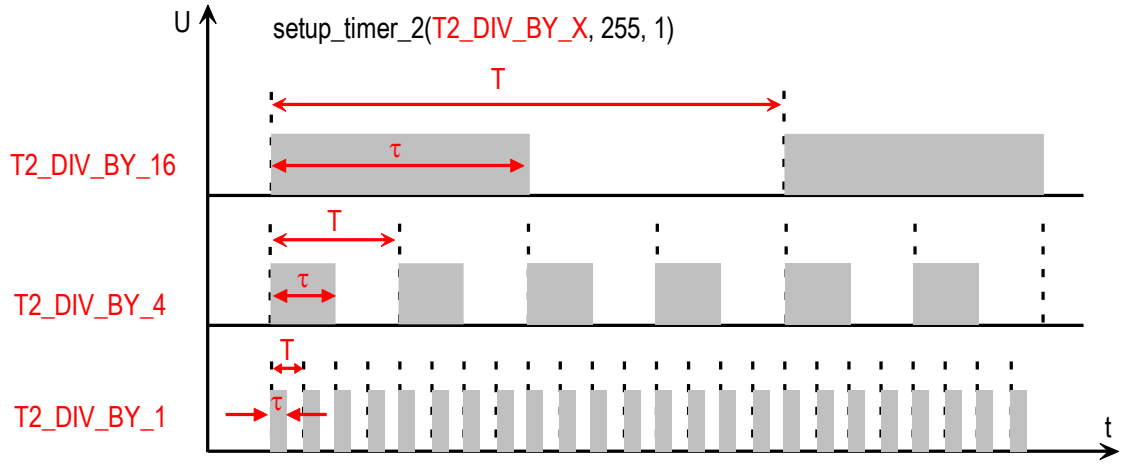


Рисунок 6 – Залежність параметрів вихідного сигналу від значення параметра 1

Другий параметр є лічильником періоду Т. Коефіцієнт приймає значення від 1 до 255.

Залежність періоду Т вихідного сигналу CCP_X від значення параметра 2 представлена на рис.7.

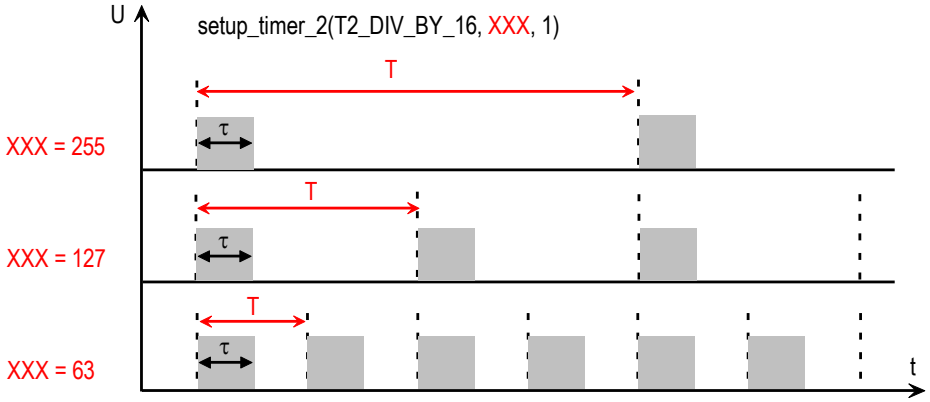


Рисунок 7 – Залежність параметрів вихідного сигналу від значення параметра 2

Примітка: Якщо параметр 1 пропорційно змінює і період проходження Т і тривалість імпульсу τ (скважність S = const), то параметр 2 змінює тільки період

проходження T. Наслідком є зміна скважності S. Змінювати скважність у такий спосіб не рекомендується. Для виконання лабораторної роботи використовувати наступні параметри:

```
setup_timer_2(T2_DIV_BY_16,255,1);
```

Керування скважністю імпульсів

Керування скважністю імпульсів здійснюється шляхом установки тривалості імпульсу τ у періоді проходження T.

```
set_pwm1_duty(long pulse_width);
```

Величина аргументу параметра `pulse_width` може перебувати в діапазоні 1..1023, що відповідає значенню скважності S у межах від 0.1% до 99.9%.

При цьому:

- Тривалість імпульсу τ не залежить від періоду проходження T. Якщо тривалість імпульсу буде більше періоду проходження, то на виході контролера CCP_X буде постійно перебувати логічна '1'.
- Період проходження T не залежить від тривалості імпульсу τ.

Величина аргументу параметра `pulse_width` який задає тривалість імпульсу τ у періоді T обчислюється по формулі:

$$pulse\_width = \tau / (1/Fosc) * t2\_div, \quad (3)$$

Тривалість імпульсу τ при відомій величині `pulse_width` визначається виразом:

$$\tau = pulse\_width * (1/Fosc) * t2div \quad (4)$$

Приклад 1: Визначити величину аргументу параметра `pulse_width`.

Частота генератора - 20 мГц = 20 000 000 Гц.

Частота $F_{ШІМ}$ = 1.2 кГц = 1200 Гц

Період = $1/F_{ШІМ}$ = 1/1200 = 833.3 мкс. = 0.0008333 сек.

Предделитель: T2_DIV_BY_16 = 16

Скважність 50% = 833.3 мкс./2 = 416 мкс. = 0.000416 сек.

Звідси:

pulse_width = 0.000416 /(1/20000000) * 16 = 520

Приклад 2: Визначити тривалість імпульсу τ і скважність S при відомому значенні аргументу параметра `pulse_width`.

Частота генератора - 20 мГц = 20 000 000 Гц.

Частота $F_{ШІМ}$ = 1.2 кГц = 1200 Гц

Період = $1/F_{ШІМ}$ = 1/1200 = 833.3 мкс. = 0.0008333 сек.

Предделитель: T2_DIV_BY_16 = 16

pulse_width = 260

Звідси:

τ = 260 * (1/20000000) * 16 = 0.000208 сек.

S знаходимо по формулі (1): S = (0.000208 *100)/0.0008333 = 24,96% або 25%

Методичні вказівки до виконання лабораторної роботи:

Порядок роботи для варіанту «Навчальний комплекс»

1. Намалювати принципову схему підключень світлодіода і двигуна постійного струму (рис. 4) відповідно до варіанта.
2. Створити алгоритм роботи програми керування яскравістю світіння світлодіода.
3. Написати програму, що реалізує створений алгоритм.
4. Створити алгоритм роботи програми керування напрямком і швидкістю обертання двигуна постійного струму.
5. Написати програму, що реалізує створений алгоритм.
6. Відкомпілювати програму, записати в мікроконтролер, виконати.

Порядок роботи для варіанту «Proteus»:

1. Завантажити з сервера папку «Proteus_students».
2. Відкрити середовище розробки PCWH.
3. У папці «Proteus_students» вибрати папку відповідної лабораторної роботи (LAB1... LAB8).
4. Відкрити файл проекту з розширенням *. Pjt.
5. У редакторі відкрити шаблон файлу програми з розширенням *. C.
6. Відкрити середу моделювання «Proteus».
7. У папці «Proteus_students» вибрати папку відповідної лабораторної роботи (LAB1... LAB8).
8. Відкрити файл проекту з розширенням *. DSN.
9. У середовищі розробки PCWH , використовуючи шаблон програми самостійно написати, відкомпілювати програму у відповідності з завданням.
10. У середовищі розробки «Proteus» виконати програму.

Примітка: Файл демонстрацій виконання лабораторної роботи розташований в папці «Demo».

Режими роботи: «Вправо», «Вліво», «Стоп». Реалізувати дискретними кнопками.

Швидкість обертання регулювати за допомогою потенціометра.

Керування режимами доцільно здійснювати через переривання INT0-INT2.

Принципова схема для виконання лабораторної роботи для варіанту «Proteus»

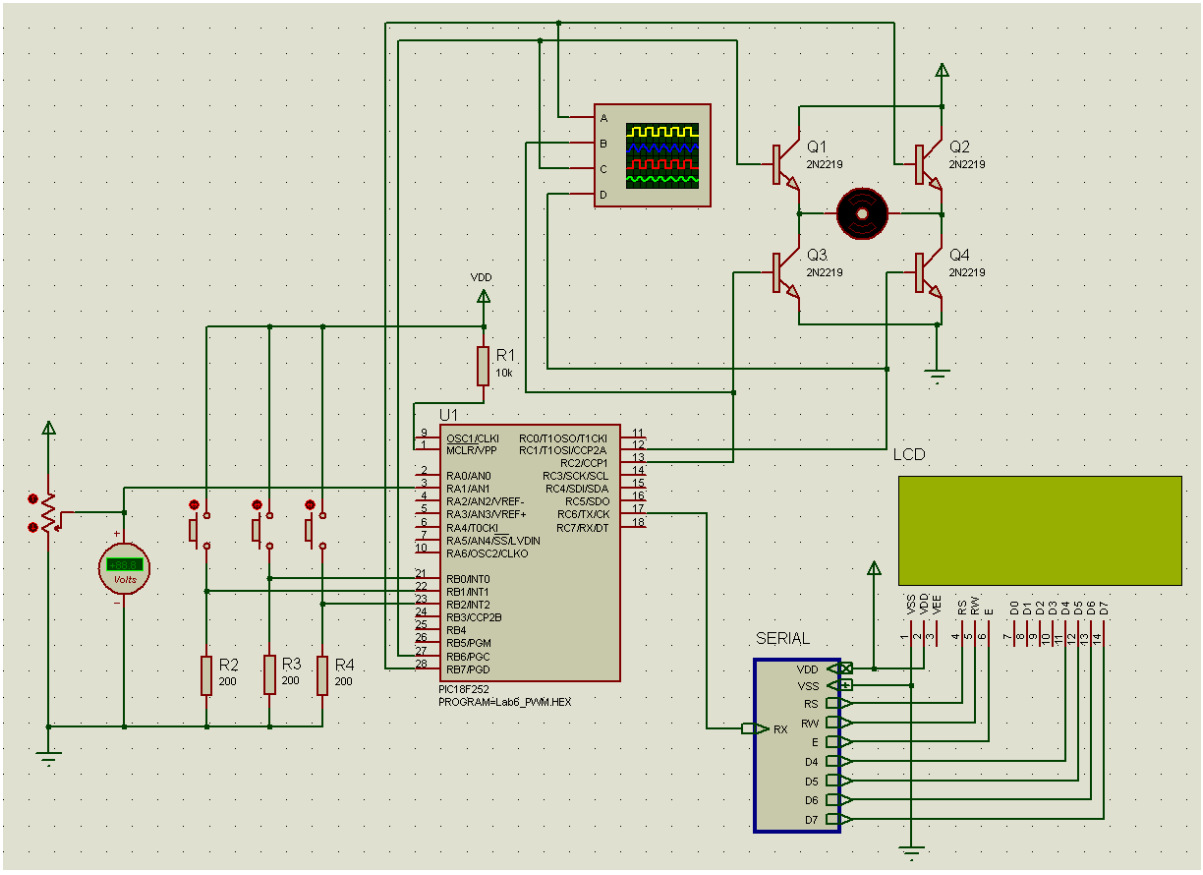


Рисунок 8 - Принципова схема для виконання лабораторної роботи для варіанту «Proteus»

Результат виконання програми:

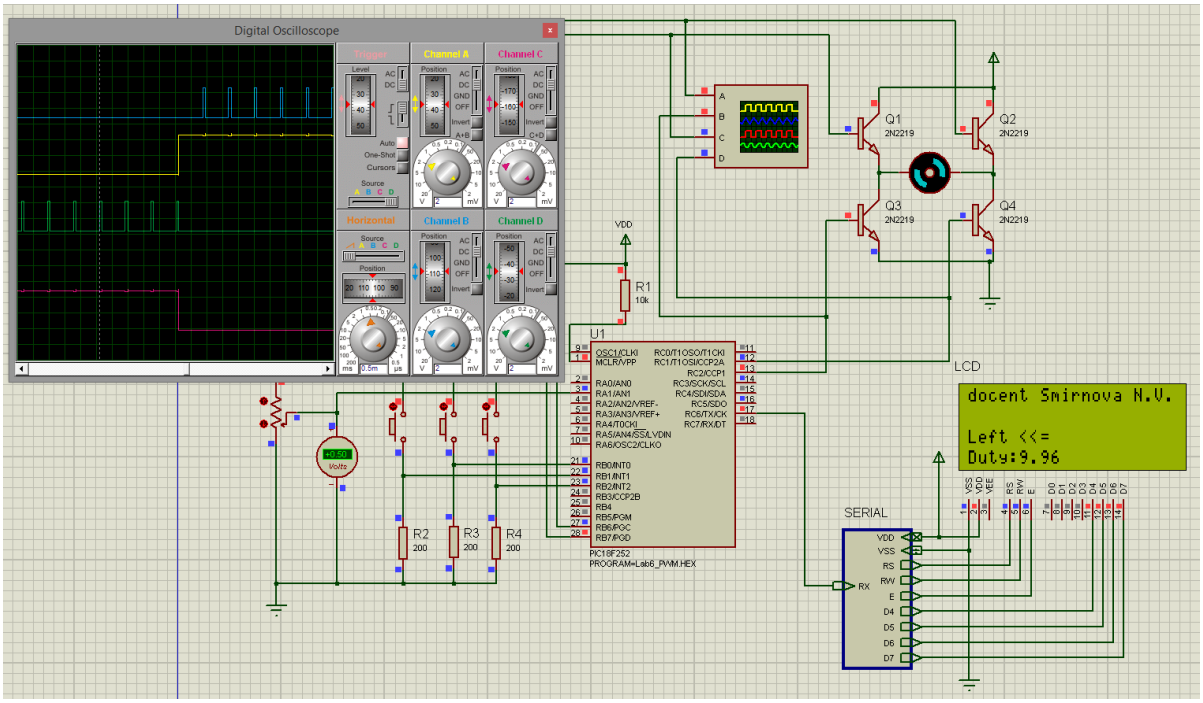


Рисунок 9 - Результат виконання лабораторної роботи для варіанту «Proteus»

Контрольні питання:

1. Призначення ШІМ і принцип роботи.
2. Загальна формула визначення скважності імпульсів і частна для завдання тривалості імпульсу.
3. Призначення параметрів ініціалізації таймера.
4. Яка залежність існує між скважністю імпульсів і формованою напругою в інтегруючому ланцюзі?
5. Принцип керування напрямком і швидкістю обертання двигуна постійного струму.

Зміст звіту:

У звіті повинні бути представлені:

- принципова схема з'єднань (роздруківка виводів контролера і від руки намальовані з'єднання для варіанту «Навчальний комплекс»), результат роботи для варіанту «Proteus».
- блок-схема алгоритму роботи програми.
- лістинг програми.
- Висновки за результатами роботи.

Варіанти

	Керування режимами						
№	**Вліво**	**Вправо**	**Стоп**	**Lup**	**Rup**	**Ldw**	**Rdw**
1	Int 0	Int 1	Int 2	B7	B6	CCP 1	CCP 2
2	Int 1	Int 2	Int 0	B6	B5	CCP 2	CCP 1
3	Int 2	Int 0	Int 1	B5	B4	CCP 1	CCP 2
4	Int 0	Int 1	Int 2	B4	B7	CCP 2	CCP 1
5	Int 1	Int 2	Int 0	B7	B6	CCP 1	CCP 2
6	Int 2	Int 0	Int 1	B6	B5	CCP 2	CCP 1
7	Int 0	Int 1	Int 2	B5	B4	CCP 1	CCP 2
8	Int 1	Int 2	Int 0	B4	B7	CCP 2	CCP 1
9	Int 2	Int 0	Int 1	B7	B6	CCP 1	CCP 2
10	Int 0	Int 1	Int 2	B6	B5	CCP 2	CCP 1
11	Int 1	Int 2	Int 0	B5	B4	CCP 1	CCP 2
12	Int 2	Int 0	Int 1	B4	B7	CCP 2	CCP 1
13	Int 0	Int 1	Int 2	B7	B6	CCP 1	CCP 2
14	Int 1	Int 2	Int 0	B6	B5	CCP 2	CCP 1
15	Int 2	Int 0	Int 1	B5	B4	CCP 1	CCP 2

Лабораторна робота № 7

Тема: Робота із зовнішньою EEPROM пам'яттю по інтерфейсу I^2C

Ціль роботи

Одержання навичок роботи з зовнішньою EEPROM – пам'яттю по інтерфейсу I^2C.

Завдання:

- Намалювати принципову схему підключень відповідно до варіанта.
- Для варіанту «Proteus» використовувати готову схему
- Написати програму роботи з зовнішньою пам'яттю (запис і читання даних по заданій адресі).
- Здійснити запис даних типу int8, int16, int32, рядок символів.
- Здійснити запис і читання даних типу int8, int16, int32, рядка символів.

Виведення на LCD повинне містити:

- Записані дані і прочитані дані.

Теоретичні відомості:

Шина I^2C широко використовується в побутовій електроніці, передачі даних і промисловій електроніці. Розроблена фірмою Philips двонаправлена 2-провідна шина призначена для приймання – передачі даних для мікроконтролерів, ЖКІ-індикаторів, аналого-цифрових і цифро-аналогових перетворювачах, ланцюгах цифрового настроювання, DTMF кодерів і декодерів, годин реального часу і т.д.

Шина I^2C являє собою концепцію, яка вирішує багато проблем послідовного інтерфейсу, і має наступні переваги:

- Тільки дві лінії - послідовна лінія даних (SDA) і послідовна лінія синхронізації (SCL).
- Кожний елемент, з'єднаний із шиною є програмно-адресуємим своєю унікальною адресою. При цьому відношення між ними можуть бути побудовані відповідно до архітектури Master-Slave або Multi-Master.
- Це справжня шина, з можливістю роботи в Multi-Master середовищі,

включаючи перевірку на пересічення та арбітраж.

- Швидкість передачі даних становить від 0 до 100 kbit/s у стандартному режимі і до 1000 kbit/s у швидкому режимі.

- Фільтрація сигналів всередині мікросхем забезпечує нечутливість до викидів на лінії шини даних.

- Число пристроїв, які можуть бути з'єднані однієї шиною, обмежене тільки максимальною ємністю шини 400 pf.

- 2-провідний послідовний інтерфейс мінімізує з'єднання, що зменшує кількість виводів;

- Повністю інтегрований I²C протокол шини усуває потребу в декодерах адреси і т.п.;

Розташування виводів шини I²C контролера представлено на рис. 1.

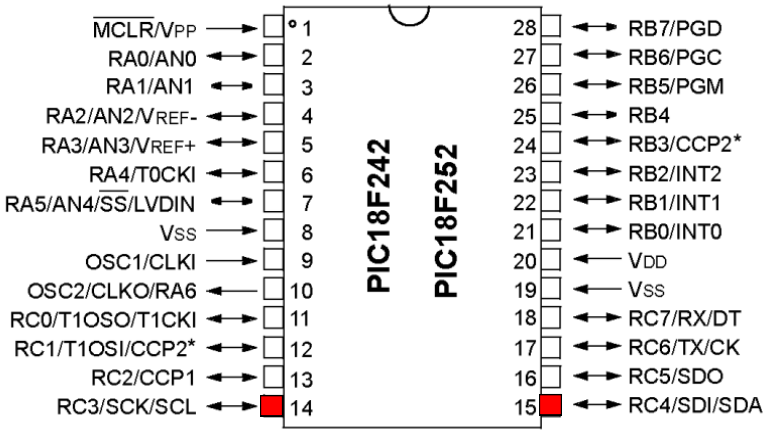


Рисунок 1 - Розташування виводів шини I²C контролера

Концепція шини I²C

Усі операції по шині I²C здійснюються за допомогою двох ліній:

- лінія послідовних даних (SDA):
- лінія синхронізації (SCL).

SDA і SCL - двонаправлені лінії з відкритим стоком (колектором), з'єднані з позитивною живлячою напругою через резистор (рис. 2). Коли шина вільна, обидві лінії знаходяться у стані "1".

Кожний елемент визначається своєю унікальною адресою, у яку входить група приладів і номер конкретного приладу. Група визначає, чи є елемент

мікроконтролером, LCD-індикатором, пам'яттю т.д. Наприклад, усі пристрої пам'яті мають код 0x0A, таймери і годинник реального часу - 0x0D, пристрою телетексту - 0x02 і т.д.

Будь-який елемент, що ініціює передачу, є майстром (Master), будь-який адресуємий елемент є підлеглим (Slave). У системах з декількома майстрами, той самий елемент може в різний час виступати або як майстер або як підлеглий.

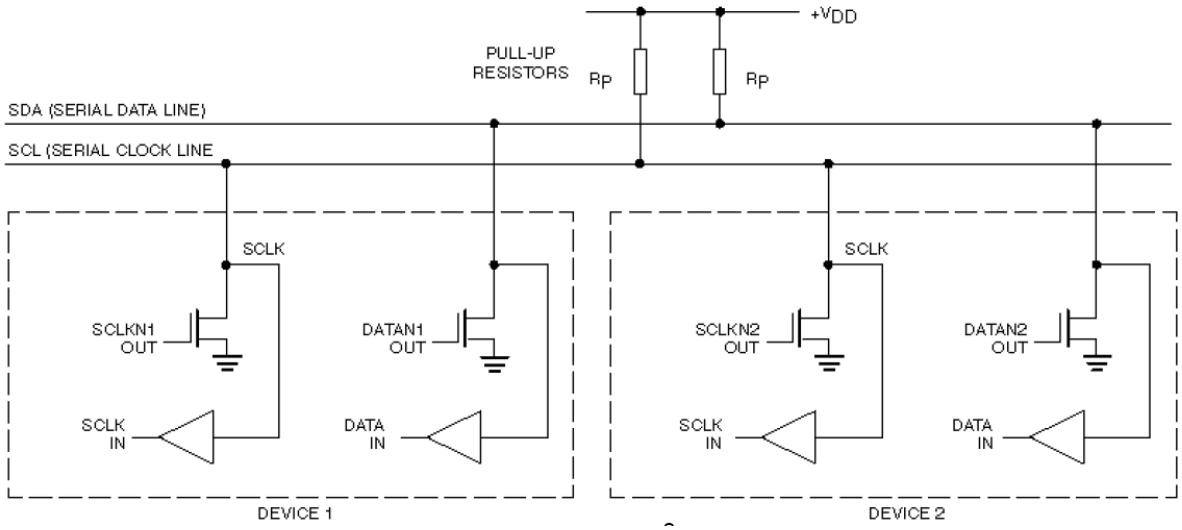


Рисунок 2 – Шина I^2C

Передача біта

Для передачі одного біта даних використовується один імпульс сигналу синхронізації на лінії SCL, при цьому рівень на лінії SDA повинен бути незмінним протягом високого рівня на лінії SCL, і може змінюватися тільки при низькому рівні на SCL (рис. 3). Виключеннями служать два особливі стани - Start і Stop.

Стани Start і Stop

Існують два особливі стани шини I^2C - Start і Stop, які служать для індикації початку і кінця передачі і відповідно, переходу шини в неактивний стан. Доти, поки не встановлений стан Start, сигнали на лініях SDA і SCL можуть бути довільними (рис. 3). Це дозволяє використовувати одну лінію SDA і кілька ліній SCL.

Стан START - "1" на лінії SCL - перехід від "1" до "0" на лінії SDA.

Стан STOP - "1" на лінії SCL - перехід від "0" до "1" на лінії SDA.

Ці два стани завжди генеруються майстром.

Детектування станів Start і Stop зазвичай проводиться апаратно.

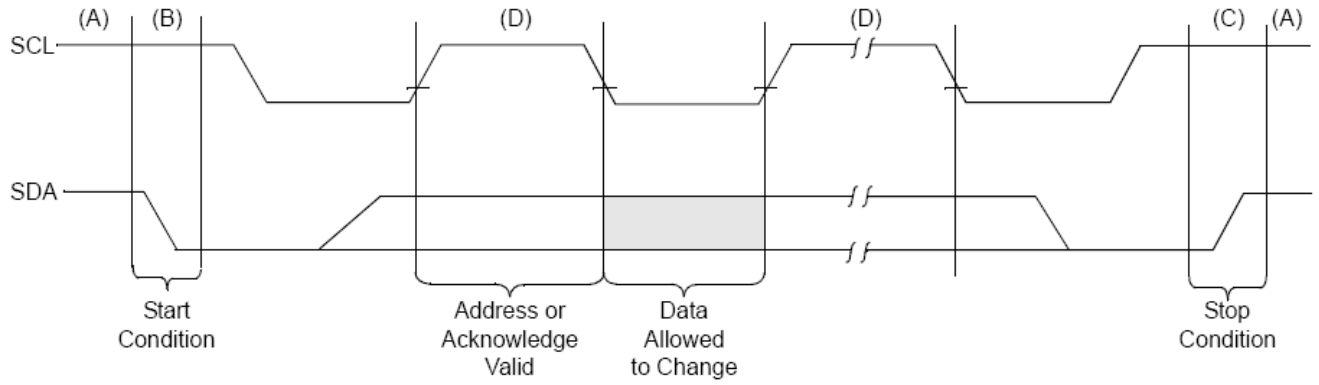


Рисунок 3 – Стани Start і Stop

Передача даних

Усі передачі проводяться 8-розрядними байтами. Число байтів, які можуть бути передані за одну передачу обмежене ємністю прийомного буфера приймача. Кожний байт повинен супроводжуватися бітом підтвердження (АСК). Дані передаються починаючи зі старшого біта (рис. 4).

Квитирування

Обмін даними по шині I^2C повинен бути детермінований. Для виконання цієї вимоги існує механізм квитирування. Для підтвердження передачі байта передавач встановлює лінію SDA в "1" протягом синхронізуючого імпульсу. Приймач при цьому повинен виставити квитанцію АСК - "0" на лінії SDA (рис. 4).

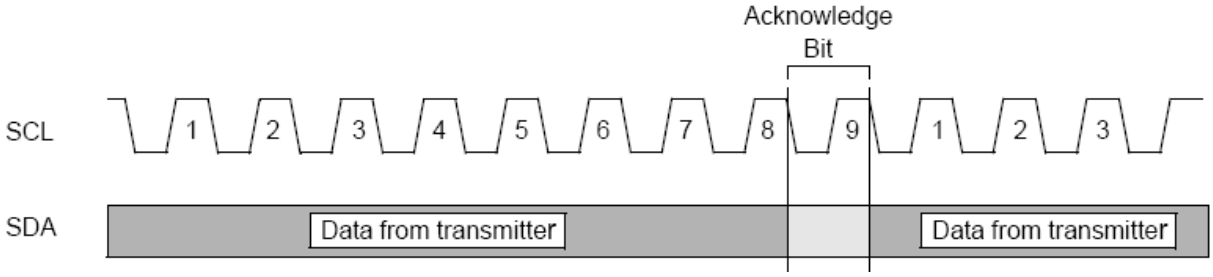


Рисунок 4 – Передача даних і квитирування

Приймач повинен генерувати сигнал АСК після одержання кожного байта.

Якщо приймач не підтверджує підлеглу адресу (наприклад, пристрій не готовий, тому що виконує деяку внутрішню функцію), лінія SDA даних повинна бути залишена в "1". Майстер встановлює стан Stop, щоб перервати передачу.

Після передачі останнього байта майстер виставляє стан Stop, при цьому приймач повинен звільнити лінію даних.

EEPROM – пам'ять 24LC256

EEPROM – пам'ять 24LC256 має об'єм 32 kb. Інтерфейс передачі - I^2C. Структура пам'яті 24LC256 і розташування виводів представлене на рис. 5 і 5.1.

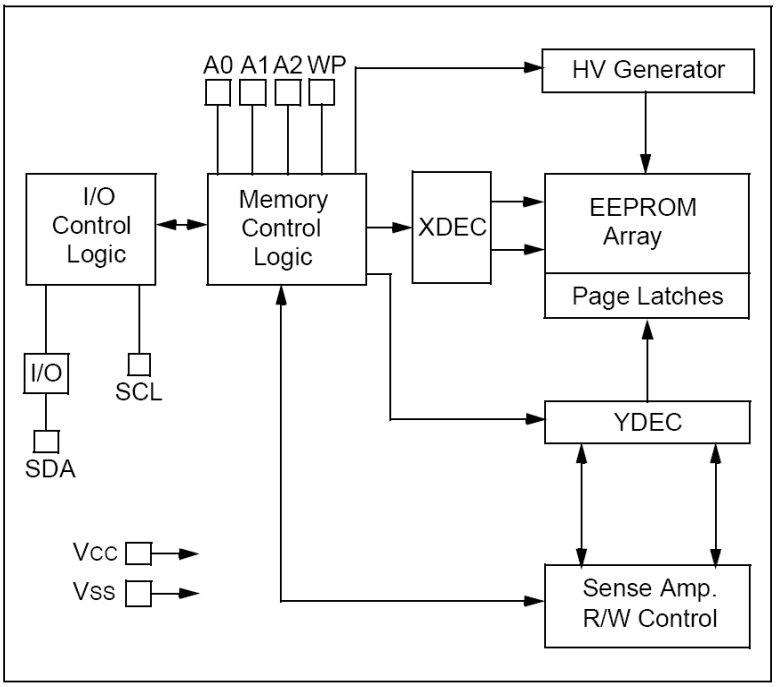


Рисунок 5 – Структура пам'яті 24LC256

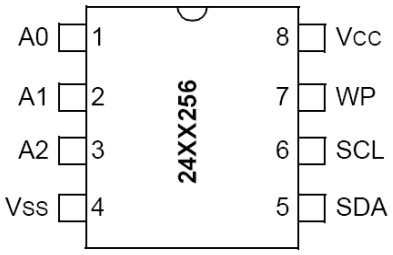


Рисунок 5.1 – Розташування виводів 24LC256

Призначення виводів 24LC256 представлено у табл.1.

Таблиця 1 – Призначення виводів 24LC256

Name	8-pin PDIP	8-pin SOIC	8-pin TSSOP	8-pin MSOP	8-pin DFN	Function
A0	1	1	1	—	1	User Configurable Chip Select
A1	2	2	2	—	2	User Configurable Chip Select
(NC)	—	—	—	1, 2	—	Not Connected
A2	3	3	3	3	3	User Configurable Chip Select
Vss	4	4	4	4	4	Ground
SDA	5	5	5	5	5	Serial Data
SCL	6	6	6	6	6	Serial Clock
(NC)	—	—	—	—	—	Not Connected
WP	7	7	7	7	7	Write-Protect Input
Vcc	8	8	8	8	8	+1.8V to 5.5V (24AA256) +2.5V to 5.5V (24LC256) +1.8V to 5.5V (24FC256)

Робота з пам'яттю

Перед виконанням будь-якої операції з пам'яттю, на шину I^2C необхідно виставити керуючий байт. Формат керуючого байта представлений на рис.6.

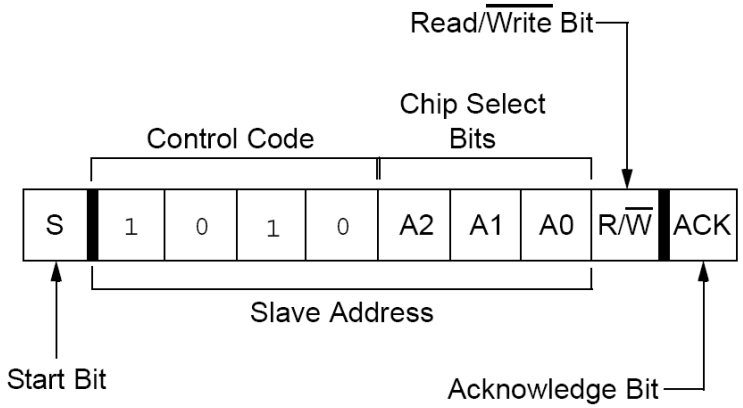


Рисунок 6 – Формат керуючого байта

Старші біти 7-4 керуючого байта містять код 1010 – ідентифікатор типу пристрою. Біти 3-2 містять адресу пристрою в системі і можуть приймати значення 000 –111. Таким чином, у системі може знаходитися 8 чипів пам'яті.

Біт 0 визначає операцію з пам'яттю: 0 - запис в пам'ять, 1 – читання з пам'яті.

На рисунку 7 представлений формат адресації чипа і адреси комірки пам'яті.

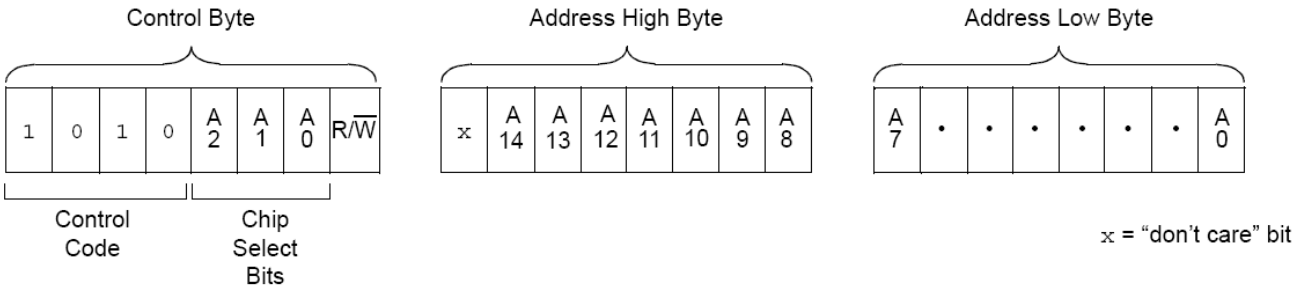


Рисунок 7 – Формат адресації чипа і адреси комірки пам'яті

Запис байта

Запис байта з контролера в пам’ять здійснюється у такий спосіб (рис.8):

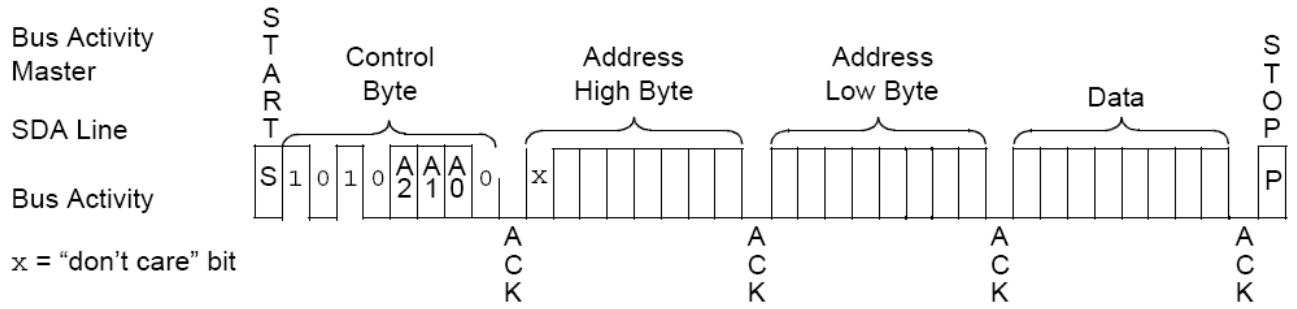


Рисунок 8 – Запис байта

- Активізується шина I^2C – на лінії SDA встановлюється стан Start (0).
- Посилається байт керування, у якому біт R/W = ‘0’.
- Очікується квитанція АСК на протязі 8-го біта синхронізації на лінії SCL.
- Якщо квитанція надходить (0 на лінії SDA), то посилається старший байт адреси комірки пам'яті.
- Очікується квитанція АСК.
- Якщо квитанція надходить, то посилається молодший байт адреси комірки пам'яті.
- Очікується квитанція АСК.
- Якщо квитанція надходить, то посилається байт даних.
- Очікується квитанція АСК.
- Якщо квитанція надходить, то на лінії SDA встановлюється стан Stop (1).

Стан Stop встановлюється контролером у всіх випадках, якщо квитанція АСК не надходить вчасно.

Запис послідовності байт

При записі побайтно послідовності байт, на кожний байт даних доводиться три службові байти, що в кілька раз знижує швидкість обміну даними з пам'яттю.

Тому існує режим запису, який дозволяє записати в пам’ять послідовність байтів даних. Дані при цьому містяться в прийомний буфер.

При записі байта даних розряди 0-6 молодшого байта адреси автоматично інкрементується, тому немає необхідності адресації для кожного байта.

Процес запису послідовності байтів представлений на рис. 9.

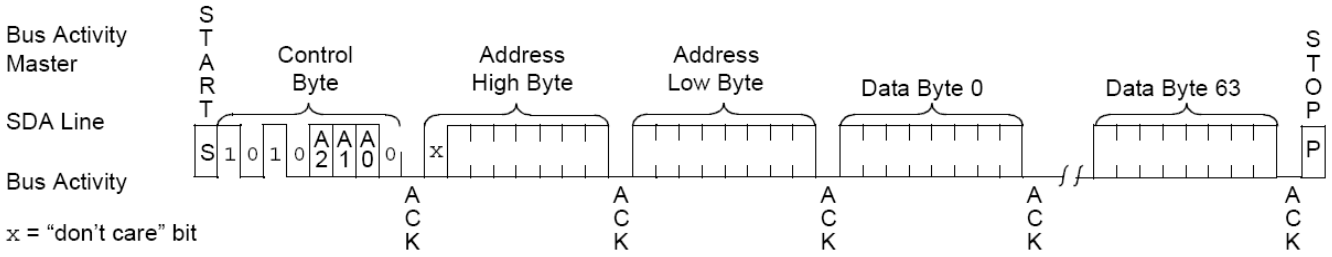


Рисунок 9 – Запис послідовності байтів

При послідовному записі байтів даних вони поміщаються в буфер пам'яті, адресуємий молодшим байтом адреси, а при надходженні стану Stop переписуються у комірки пам'яті.

Розмір буфера становить 64 байта. Якщо кількість байтів перевищить це значення, то в молодшому байті адреси виникне переповнення і дані в буфері будуть перезаписані.

Перевірка готовності

При роботі з пам'яттю завжди необхідно переконатися, що пам'ять знаходиться у стані готовності. Після прийому послідовності байт у прийомний буфер і одержання стану Stop, пам'ять переходить у режим запису вмісту буфера в комірки пам'яті. Оскільки запис байта в комірку пам'яті забирає тривалий час (5 мс), у це пам'ять зайнята і квитанція ACK не виставляється.

Для одержання стану готовності необхідно:

1. Після виконання попередньої операції завершити її станом Stop.
2. Виставити стан Start.
3. Послати керуючий байт, у якому біт R/W = ‘0’.

4. Повірити стан шини SDA на ‘0’.
5. Якщо квитанція не поступила, перейти до пункту 3.

Блок-схема алгоритму перевірки готовності пам'яті представлена на рис. 10.

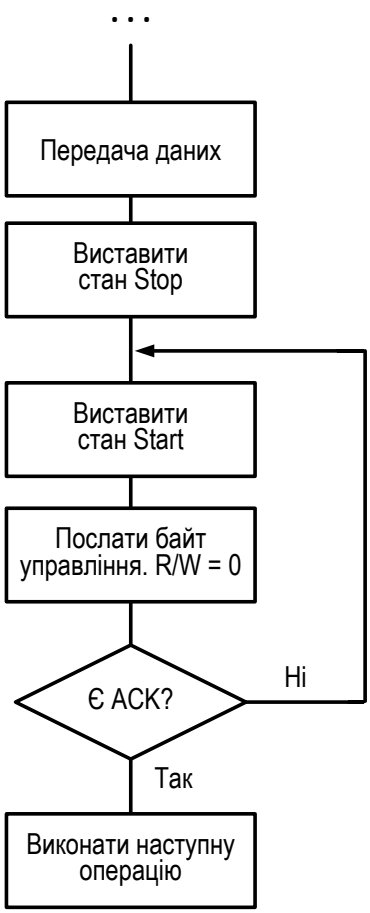


Рисунок 10 – Блок-схема алгоритму перевірки готовності пам'яті

Читання байта по поточній адресі

Після виконання операції читання байта, внутрішній лічильник адреси 24LC256 інкрементується, тому при читанні наступного байта, його можна не адресувати (рис. 11).

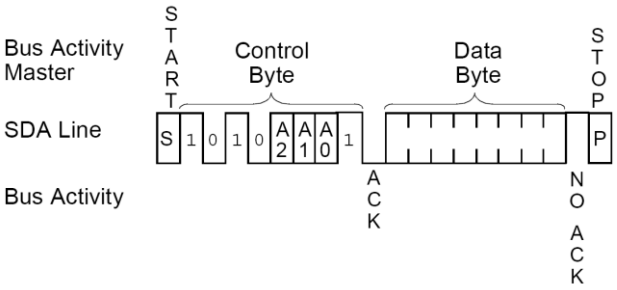


Рисунок 11 – Читання байта по поточній адресі

Читання байта по довільній адресі

Процедура читання байта по довільній адресі представлена на рис. 12.

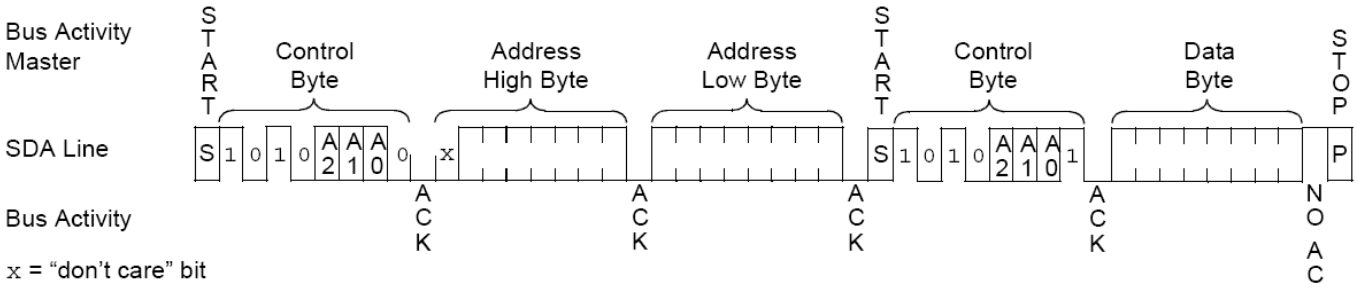


Рисунок 12 – Читання байта по довільній адресі

Процедура читання байта виконується у наступній послідовності:

- Активізується шина I^2C – на лінії SDA встановлюється стан Start.
- Посилається байт керування, у якому біт R/W = ‘0’.
- Очікується квитанція АСК на протязі 8-го біта синхронізації на лінії SCL.
- Якщо квитанція надходить (0 на лінії SDA), то посилається старший байт адреси комірки пам'яті.
- Очікується квитанція АСК.
- Якщо квитанція надходить, то посилається молодший байт адреси комірки пам'яті.
- Очікується квитанція АСК.
- Встановлюється стан Start.
- Посилається байт керування, у якому біт R/W = ‘1’ (читання).
- Якщо квитанція надходить, то зчитується байт даних.

- Квитанція після прочитаного байта не очікується.
- На лінії SDA встановлюється стан Stop.

Читання послідовності байт по довільній адресі

Процедура читання послідовності байтів аналогічна процедурі читання одного байта, за винятком того, що виключається адресація кожного байта, що читається,

оскільки внутрішній лічильник адреси пам'яті інкрементується автоматично.

Процедура послідовності байт по довільній адресі представлена на рис. 13.

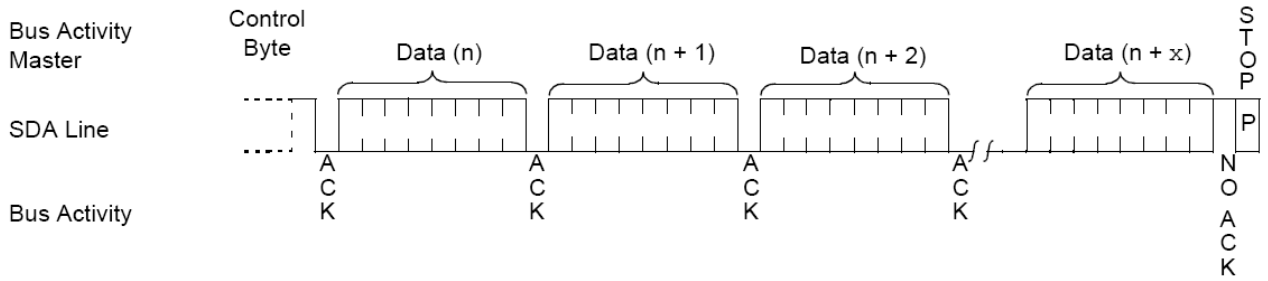


Рисунок 13 – Читання послідовності байтів по довільній адресі

Об'єм зчитаних даних обмежується об'ємом пам'яті.

Конфігурування контролера для роботи з I²C

У тексті програми або в заголовному файлі вставити стоку:

```
#use i2c(Master,Slow,sda=PIN_C4,scl=PIN_C3);
```

Цей рядок призначає контролер Майстром, швидкість обміну – низька, дані прив'язані до виводу контролера `PIN_C4`, синхронізація - до виводу контролера `PIN_C3`.

У функцію ініціалізації вставити рядки:

```
output_float(PIN_C3);
output_float(PIN_C4);
```

Ці команди встановлюють виводи шини в режим роботи з відкритим стоком.

Керування шиною I²C

Керування шиною I²C здійснюється функціями:

```
i2c_start();                //Стан Start
i2c_stop();                 //Стан Stop
```

Посилка управляючого байта:

```
i2c_write(control_byte);      //ID пам'яті, адреса пам'яті, операція
```

Посилка адреси:

```
i2c_write(make8(address,1));  //старший байт адреси
i2c_write(make8(address,0));  //молодший байт адреси
```

Запис байта в пам’ять:

```
i2c_write(data);              //записати байт
```

Читання байта з пам'яті:

```
Data = i2c_read();            //читати байт
```

Перевірка готовності пам'яті

Перевірка готовності пам'яті здійснюється наступною функцією:

```
int8 EEP_ready()
{
   int1 ack;                        //квитанція

   i2c_start();                     //якщо команда одержить квитанцію,
   ack = i2c_write(ctrl_write);     //значить пристрій готовий
   i2c_stop();
   return !ack;
}
```

Приклад реалізації функції записи байта в пам’ять

```
EEP_write_byte(int16 address, int8 data)
{
   while(!EEP_ready());          //чекати готовності пам'яті
   i2c_start();                  //старт
   i2c_write(ctrl_write);        //ID пам'яті
   i2c_write(make8(address,1));  //старший байт адреси
   i2c_write(make8(address,0));  //молодший байт адреси
   //====== записати байт
   i2c_write(data);              //записати байт
   i2c_stop();                   //стоп
}
```

Методичні вказівки до виконання лабораторної роботи:

Порядок роботи для варіанту «Навчальний комплекс»

1. Намалювати принципову схему підключень контролера й пам'яті до шини I^2C.
2. Створити алгоритм роботи програми роботи с зовнішньою пам'яттю (запис і читання даних по заданій адресі):
   - запис даних типу int8, int16, int32, рядка символів;

- читання даних типу int8, int16, int32, рядка символів.

3. Написати програму, що реалізує створений алгоритм.
4. Відкомпілювати програму, записати в мікроконтролер, виконати.

Примітка: робота із завданням може бути організована за допомогою меню або з демонстрацією роботи в послідовності виконання завдань із інтервалом в 2-5 сек.

Порядок роботи для варіанту «Proteus»:

1. Завантажити з сервера папку «Proteus_students».
2. Відкрити середовище розробки PCWH.
3. У папці «Proteus_students» вибрати папку відповідної лабораторної роботи (LAB1... LAB8).
4. Відкрити файл проекту з розширенням *. Pjt.
5. У редакторі відкрити шаблон файлу програми з розширенням *. C.
6. Відкрити середу моделювання «Proteus».
7. У папці «Proteus_students» вибрати папку відповідної лабораторної роботи (LAB1... LAB8).
8. Відкрити файл проекту з розширенням *. DSN.
9. У середовищі розробки PCWH , використовуючи шаблон програми самостійно написати, відкомпілювати програму у відповідності з завданням.
10. У середовищі розробки «Proteus» виконати програму.

Примітка: Файл демонстрацій виконання лабораторної роботи розташований в папці «Demo».

Принципова схема для виконання лабораторної роботи для варіанту «Proteus»

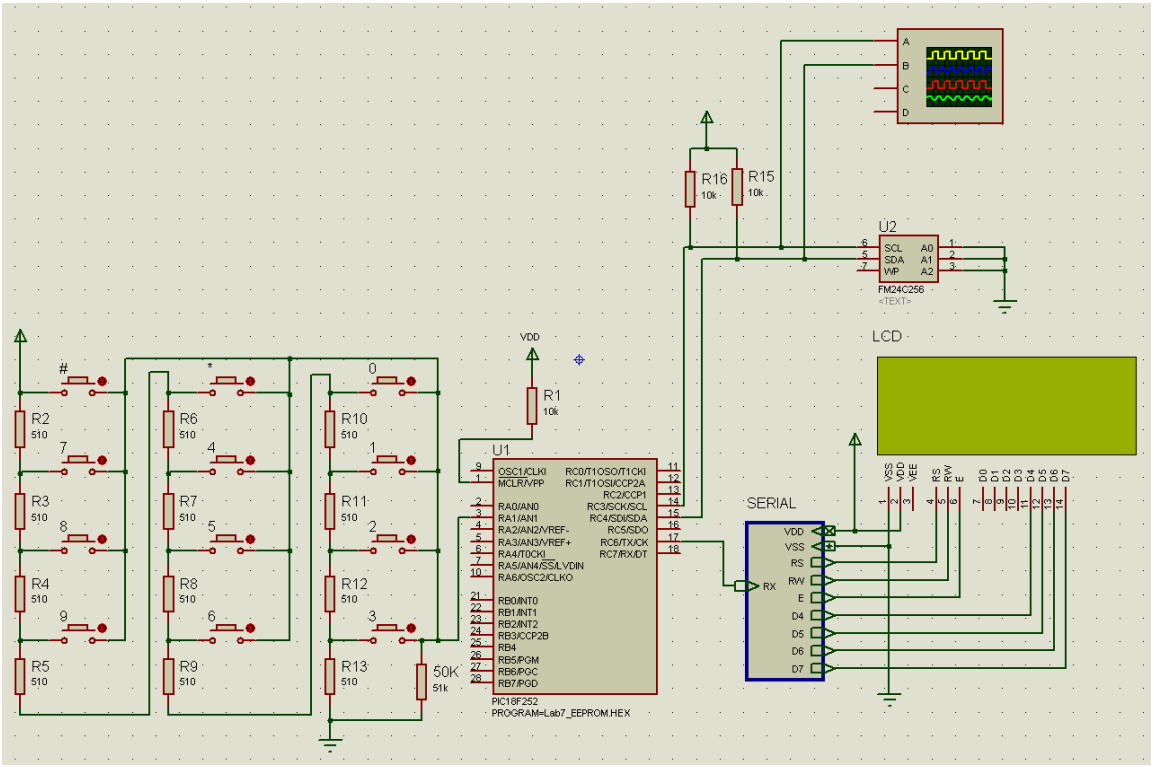


Рисунок 14 - Принципова схема для виконання лабораторної роботи для варіанту «Proteus»

Результат виконання програми:

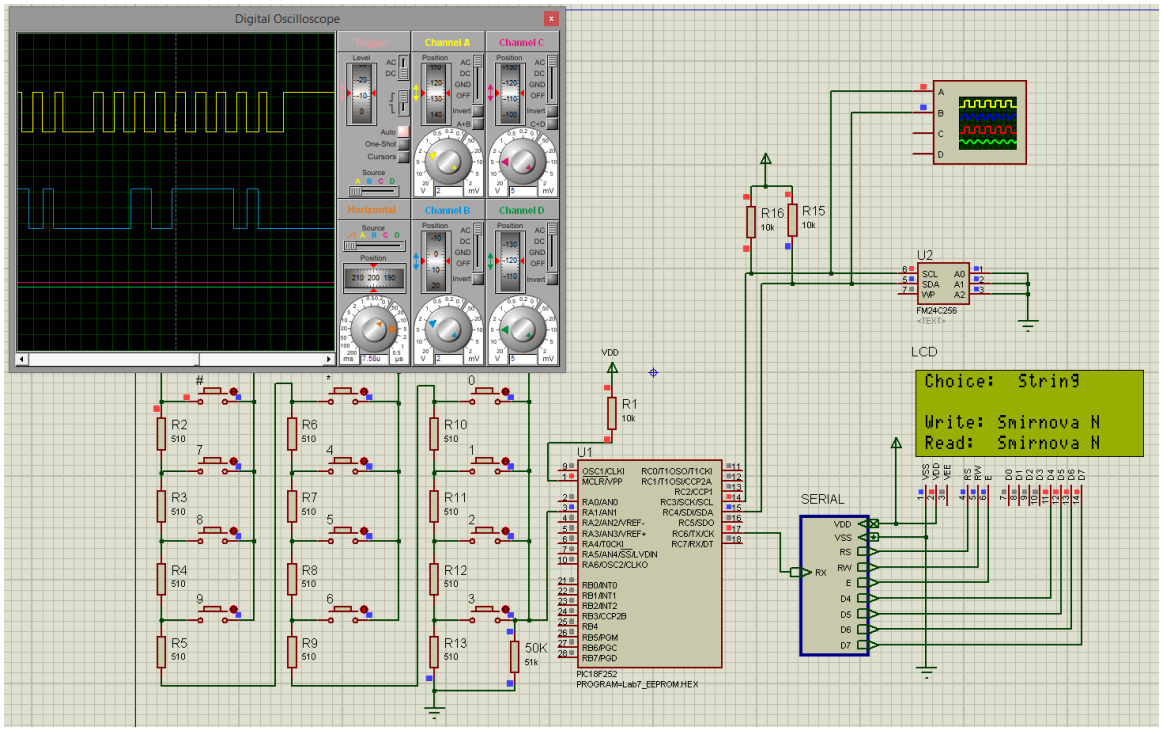


Рисунок 15 - Результат виконання лабораторної роботи для варіанту «Proteus»

Контрольні питання:

1. Призначення шини I^2C і принцип роботи.
2. Порядок ініціалізації контролера для роботи із шиною I^2C і призначення параметрів команд.
3. Процедура запису даних в пам’ять по шині I^2C.
4. Процедура читання даних в пам’ять по шині I^2C.
5. Алгоритм визначення готовності пам'яті.

Зміст звіту:

У звіті повинні бути представлені:

- принципова схема з'єднань (роздруківка виводів контролера і від руки намальовані з'єднання для варіанту «Навчальний комплекс»), результат роботи для варіанту «Proteus».
- блок-схема алгоритму роботи програми.
- лістинг програми.
- висновки за результатами роботи.

Варіанти

	Адреса		**Дані**			
№	**Intxx**	**Рядок**	**Int8**	**Int16**	**Int32**	**Рядок**
1	10	20	11	1111	11111111	Прізвище, ініціали
2	20	30	12	1212	12121212	Прізвище, ініціали
3	30	40	13	1313	13131313	Прізвище, ініціали
4	40	50	14	1414	14141414	Прізвище, ініціали
5	50	60	15	1515	15151515	Прізвище, ініціали
6	60	70	16	1616	16161616	Прізвище, ініціали
7	70	80	17	1717	17171717	Прізвище, ініціали
8	80	90	18	1818	18181818	Прізвище, ініціали
9	90	100	19	1919	19191919	Прізвище, ініціали
10	100	110	20	2020	20202020	Прізвище, ініціали
11	110	120	21	2121	21212121	Прізвище, ініціали
12	120	130	22	2222	22222222	Прізвище, ініціали
13	130	140	23	2323	23232323	Прізвище, ініціали
14	140	150	24	2424	24242424	Прізвище, ініціали
15	150	160	25	2525	25252525	Прізвище, ініціали

Лабораторна робота №8

Тема: Сервоприводи

Ціль роботи

Одержання навичок роботи керування сервоприводом.

Завдання:

- Намалювати принципову схему підключень.
- Для варіанту «Proteus» використовувати готову схему
- Здійснити керування валом сервопривода в межах 180 градусів за допомогою потенціометра і АЦП шляхом перетворення напруги на движку потенціометра у задаючий вплив, (тривалість керуючого імпульсу) для сервопривода.
- Здійснити виведення тривалості керуючого імпульсу на LCD.

Виведення на LCD повинне містити:

- Групу, прізвище та ініціали студента.

Теоретичні відомості:

Сервопривод – це спеціальний електричний двигун з негативним зворотним зв'язком, призначений для використання в системах керування різними об'єктами. Сервоприводи мають високі швидкісні характеристики, і точність позиціонування.

Сучасні Сервоприводи здатні розвивати великі швидкості обертання при досить високій потужності. Великий діапазон регулювання обертання вала сервопривода засобами спеціального програмного забезпечення, робить їх незамінними для пристроїв автоматики.

Сервоприводи поєднують у собі велику потужність і компактність. Вони можуть працювати тільки при наявності електронного блоку. Сукупність сервомотора і керуючого електронного модуля називається сервоприводом. Одна з переваг сервоприводов перед кроковими двигунами - це плавністю ходу. Наявність зворотного зв'язку створює умову для точного позиціонування положення і швидкістю обертання вала сервопривода.

Сервоприводи в цей час застосовуються у високопродуктивному обладнанні багатьох галузей промисловості.

Сервоприводи обертального руху використовуються в:

- промислових роботах;
- приводах верстатів ЧПУ;
- поліграфічних верстатах;
- пакувальних верстатах;
- приладах;
- керованих моделях.

У сучасному виробництві до сервоприводу пред'являються високі вимоги по наступних параметрах:

- точність позиціонування;
- діапазон регулювання;
- нерівномірність частоти обертання;
- перевантажувальна здатність;
- висока динаміка.

Склад сервоприводу

- **привод** - наприклад, електромотор з редуктором, або пневмоциліндр,
- **датчик зворотного зв'язку** - наприклад, датчик кута повороту вихідного вала редуктора (енкодер),
- **блок керування** (він же перетворювач частоти і сервопідсилювач).
- **блок перетворення, задаючого впливу**

Потужність двигунів: від 0,05 до 15 кВт.

Моменти обертання (номінальні): від 0,15 до 500 кг/см і більше.

Живлення і цоколівка роз'ємів:

У лабораторній роботі буде використовуватися сервопривод для радіокерованих моделей.

Ці Сервоприводи мають 3 дроти:

1. Сигнальний дріт, по якому надходить керуючий імпульс (зазвичай білого або оранжевого кольору).

2. Провід живлення, від 4,8 В до 6 В (зазвичай червоного кольору).

3. Земля (зазвичай провід чорного або коричневого кольору).

У лабораторній роботі сервопривод буде підключений до роз'єму.

Основні характеристики сервоприводів

Базовими технічними характеристиками сервоприводів є обертовий момент на вихідному валу і швидкість повороту качалки.

Обертовий момент вимірюють у кг/см. Момент в 3 кг/см означає, що сервопривод буде утримувати вантаж, закріплений на важелі в 1 сантиметр від його осі, із силою в 3 кг. Тобто, добуток сили F у кг на плечі важеля P у см - це і є обертовий момент M.

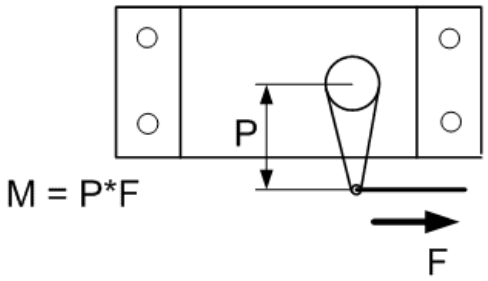


Рис. 1 - Розрахунок обертового моменту

Швидкість повороту вала вказує за який час у секундах, сервопривод повертає вал на 60°.

Керуюча електроніка споживає незначний струм: 8-10 мА.

Габарити:

По габаритах сервоприводи діляться на 3 основні розміри: мікро, стандартні і великі.

Функціональна схема сервопривода:

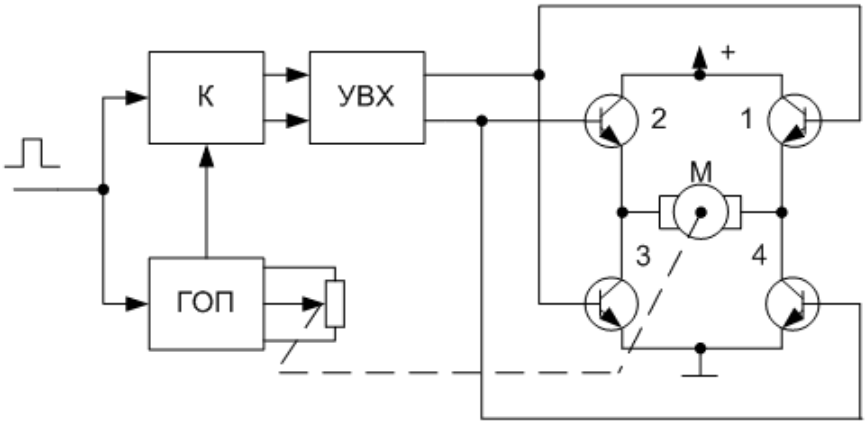


Рис. 2 - Функціональна схема сервопривода

Схема складається з генератора опорного імпульсу (ГОП), до якого підключений потенціометр зворотного зв'язку, компаратора (К), пристрою вибірки-зберігання (УВХ) і силового моста, у діагональ якого включений електромотор (М). (Тут бази транзисторів-ключів об'єднані умовно).

Керуючий імпульс від приймача приходить на компаратор і одночасно запускає генератор опорного імпульсу.

Тривалість опорного імпульсу залежить від положення потенціометра зворотнього зв'язка, механічно з'єднаного з вихідним валом.

Керуючий і опорний імпульси порівнюються компаратором по тривалості. Різницевий імпульс з'являється на верхньому, або нижньому виходах компаратора, залежно від того, який з порівнюваних імпульсів довше.

Довжина різницевого імпульсу визначає величину неузгодженості (сигналу помилки) між необхідним і поточним положенням вала сервопривода.

Ця величина виміряється і запам'ятовується у вигляді постійного потенціалу на час циклу керуючого імпульсу в пристрої вибірки-зберігання (УВХ).

Виходи УВХ управляють ключами моста і обертають електродвигун сервопривода (як у лабораторній роботі із ШИМ).

Керування кутом поворотом вала аналогового сервопривода

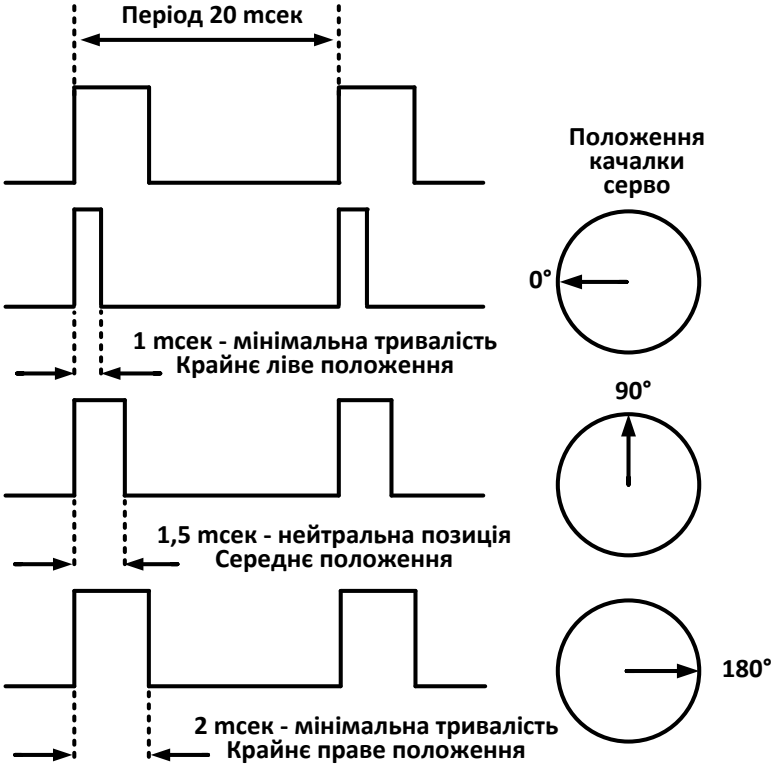


Рис. 3 - Керування кутом поворотом вала аналогового сервопривода

Для керування кутом повороту вала сервопривода використовуються прямокутні імпульси з постійним періодом проходження і зміною тривалістю.

Період повторення імпульсів є постійним і становить 20 миллисекунд. Тривалість імпульсів у періоді змінюється в діапазоні від 1 до 2 мілісекунд.

Тривалість вхідного імпульсу є задаючим впливом і визначає кут повороту вихідного вала.

Наприклад, тривалість вхідного імпульсу в 1 миллисекунду являє собою команду сервоприводу повернути вал у крайнє ліве положення.

Вхідний імпульс тривалістю в 1.5 мс дає команду сервоприводу повернути вал у середнє положення, а 2 мс – у крайнє праве. Тривалість керуючого імпульсу може різнитися.

Таким чином, змінюючи тривалість вхідного імпульсу 50 раз у секунду (період – 20 мс) від 1 до 2 мс, можна управляти кутом повороту вихідного вала сервопривода.

Методичні вказівки до виконання лабораторної роботи:

Порядок роботи для варіанту «Навчальний комплекс»

1. Намалювати принципову схему підключень.
2. Створити алгоритм програми
3. Написати програму, що реалізує створений алгоритм.
4. Відкомпілювати програму, записати в мікроконтролер, виконати.

Порядок роботи для варіанту «Proteus»:

1. Завантажити з сервера папку «Proteus_students».
2. Відкрити середовище розробки PCWH.
3. У папці «Proteus_students» вибрати папку відповідної лабораторної роботи (LAB1... LAB8).
4. Відкрити файл проекту з розширенням *. Pjt.
5. У редакторі відкрити шаблон файлу програми з розширенням *. C.
6. Відкрити середу моделювання «Proteus».
7. У папці «Proteus_students» вибрати папку відповідної лабораторної роботи (LAB1... LAB8).
8. Відкрити файл проекту з розширенням *. DSN.
9. У середовищі розробки PCWH, використовуючи шаблон програми самостійно написати, відкомпілювати програму у відповідності з завданням.
10. У середовищі розробки «Proteus» виконати програму.

Примітка: Файл демонстрацій виконання лабораторної роботи розташований в папці «Demo».

Принципова схема для виконання лабораторної роботи для варіанту «Proteus»

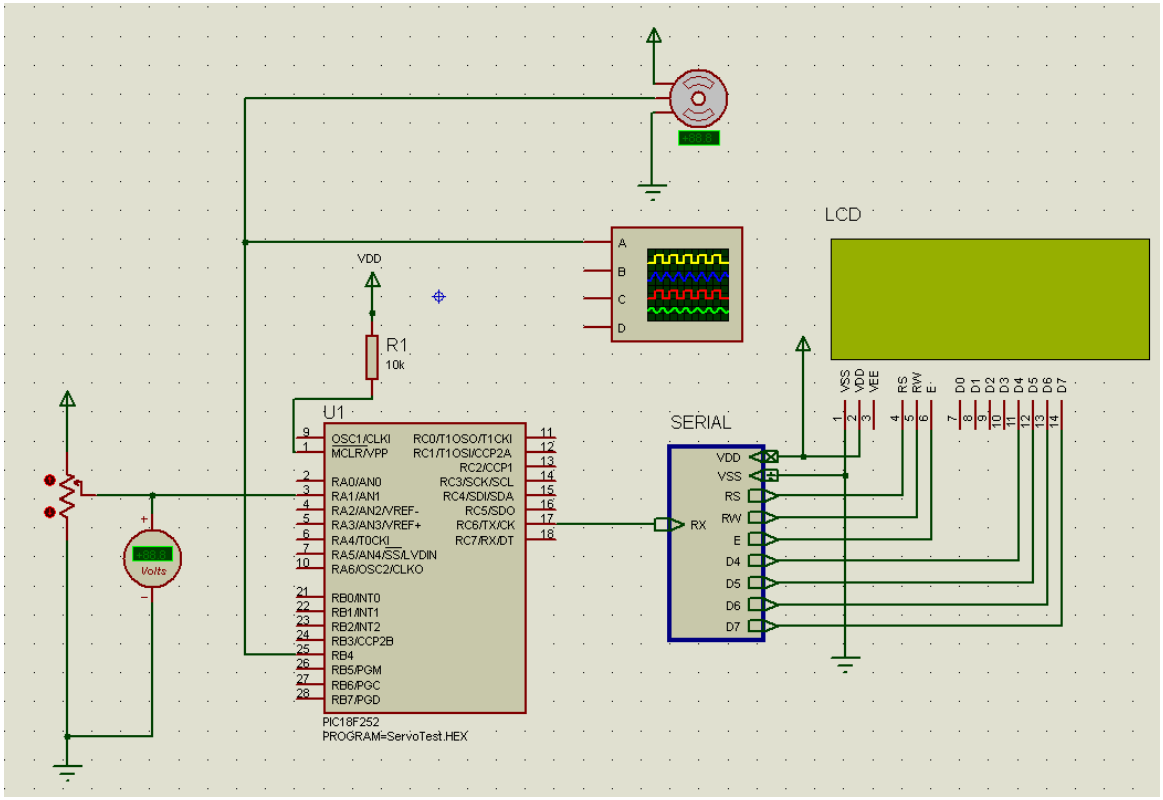


Рисунок 4 - Принципова схема для виконання лабораторної роботи для варіанту «Proteus»

Результат виконання програми:

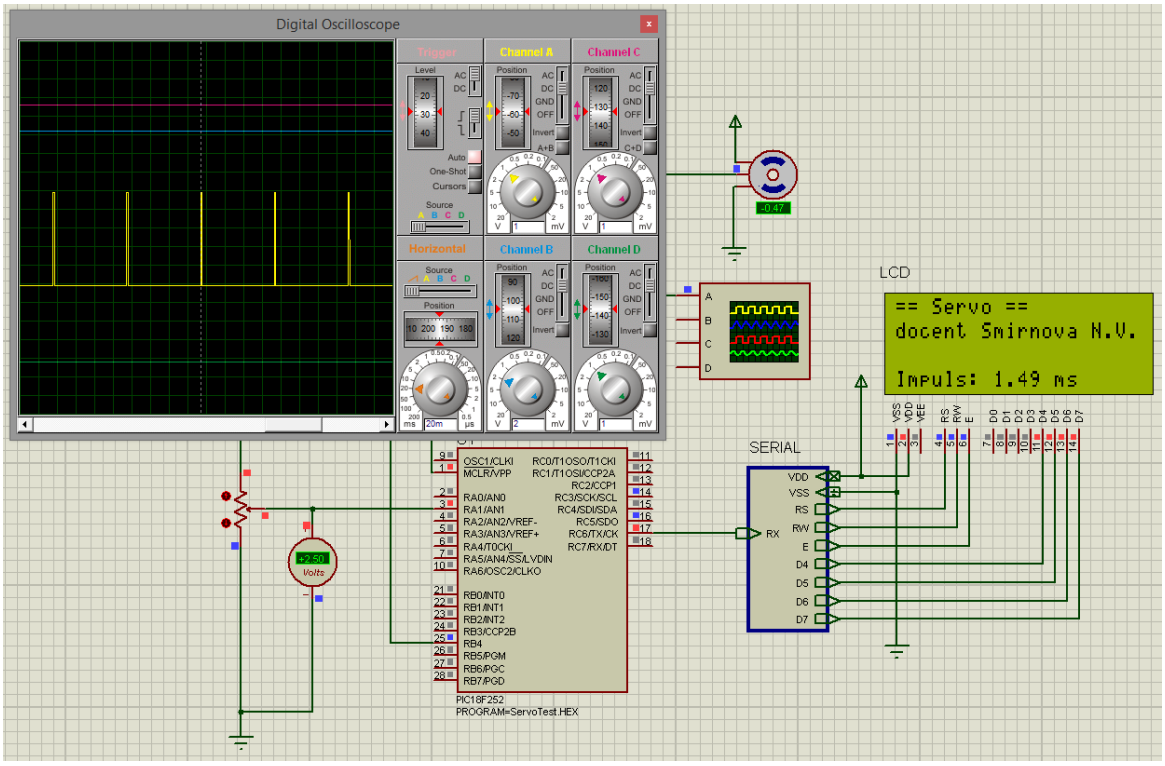


Рисунок 5 - Результат виконання лабораторної роботи для варіанту «Proteus»

Контрольні питання:

- Призначення сервопривода;
- Область застосування сервопривода;
- Принцип роботи і керування сервоприводом;
- Тривалість керуючого імпульсу для даного сервопривода;
- Відмінність цифрового і аналового сервопривода;

Зміст звіту

У звіті повинні бути представлені:

- принципова схема з'єднань (роздруківка виводів контролера і від руки намальовані з'єднання для варіанту «Навчальний комплекс»), результат роботи для варіанту «Proteus».
- блок-схема алгоритму роботи програми.
- лістинг програми.
- Висновки за результатами роботи.

Тестові завдання для самоконтролю

1) Коротка характеристика мікроконтролера PIC18F252.
a) Периферійний інтерфейсний контролер
b) Універсальний мікропроцесор
c) Універсальний сигнальний процесор
d) Зв'язувальний контролер
e) Керуючий модуль
a.

2) Як управляти напрямком введення - виведення даних у мікроконтролері?
a) Командою PUSH
b) Командою DIRECT
c) Командою DIRECTION
d) Командою TRIS
e) Командою POP
d.

3) Які команди здійснюють виведення інформації на LCD?
a) out_LCD
b) send_LCD
c) printf
d) print_LCD
e) plot
c.

4) Яким чином здійснюється запис програми у мікроконтролер?
a) ручкою
b) лістингом
c) файлом
d) програматором
e) записувачем
d.

5) У яку пам'ять завантажується код програми?
a) У внутрішню
b) В EEPROM
c) У пам'ять даних
d) У пам'ять програм
e) У пам'ять коду
d.

6) Зовнішні переривання мікроконтролера PIC18F252
a) CCP
b) USART
c) Таймер

d) ШІМ
e) ЦАП
b.

7) Як управляти (дозволяти, забороняти) перериваннями в мікроконтролері?
a) Вмиканням живлення
b) Вимиканням живлення
c) Командою YES
d) Командою NOT
e) Командою ENABLE/DISABLE
e.

8) Чому в перериванні порту RS-232 необхідно очистити прийомний буфер?
a) Щоб звільнити місце
b) Для запису відповіді
c) Щоб зняти переривання
d) Щоб переривання не скинулося
e) Щоб прочитати символ
e.

9) Якими по відношенню контролера бувають переривання?
a) Вищерозміщені
b) Центральні
c) Правобічні
d) Лівосторонні
e) Зовнішні
e.

10) Призначення АЦП.
a) Для перемикання каналів
b) Для швидкої реакції на переривання
c) Перетворення цифрового значення в аналоговий
d) Захист від зависання
e) Перетворення аналогового сигналу в цифровий
e.

11) Методи перетворення АЦП послідовного наближення?
a) Прямий
b) Зворотний
c) Послідовний
d) Паралельного наближення
e) Послідовного наближення
e.

12) Конфігурування мікроконтролера для роботи з АЦП.
a) Командою SET_float

b) Командою RESET
c) Командою SET_DIGIT
d) Командою RESET_DIGIT
e) Командою SET_ANALOG
e.

13) Вплив розрядності АЦП на точність перетворення.
a) Більше - краще
b) Менше - краще
c) Більше - гірше
d) Не впливає
e) Впливає незначно
a.

14) Опорна напруга Vref і його вплив на параметри АЦП.
a) Не потрібно
b) Не впливає
c) Впливає незначно
d) Визначає шкалу перетворення
e) Збільшує перешкодостійкість
d.

15) Як визначається ціна розподілу (вага) шкали АЦП?
a) Зліва на право
b) З права на ліво
c) Множенням Vref на розрядність
d) Поділом Vref на розрядність
e) Вирахуванням Vref з VDD
d.

16) Призначення ЦАП.
a) Цифро-алфавітне перетворення
b) Цифро-послідовне перетворення
c) Цифро-паралельне перетворення
d) Цифро -аналогове перетворення
e) Цифро-аналогічне перетворення
d.

17) Інтерфейс SPI.
a) Послідовно-паралельний
b) Тільки паралельний
c) Тільки послідовний
d) Режим паралельності задається
e) Паралельно-послідовний
c.

18) Призначення лінії CS інтерфейсу SPI.
a) Скидання даних
b) Передача даних
c) Прийом даних
d) Очікування
e) Вибір чипа
e.

19) Призначення лінії SDI інтерфейсу SPI
a) Скидання даних
b) Передача даних
c) Прийм даних
d) Вмикання
e) Вимикання
c.

20) Призначення лінії SDO інтерфейсу SPI
a) Скидання даних
b) Передача даних
c) Приймання даних
d) Вмикання
e) Вимикання
b.

21) Схеми включення світлодіодів у LED - дисплеї.
a) Послідовно
b) Паралельно
c) Паралельно - послідовно
d) Із загальним анодом
e) Порозрядно
d.

22) Зовнішні переривання мікроконтролера PIC18F252
a) CCP
b) USART
c) Таймер
d) ШІМ
e) ЦАП
b.

23) 1 – розрядний LED - дисплей має 9 виводів. 3 - розрядний LED - дисплей має виводів?
a) 27
b) 25
c) 12
d) 15

e) 11
e.

24) 3 – розрядний LED - дисплей із загальним катодом має виводів?
a) 11
b) 13
c) 14
d) 15
e) 16
a.

25) 3 – розрядний LED - дисплей із загальним анодом має виводів?
a) 11
b) 13
c) 14
d) 15
e) 16
a.

26) Призначення ШІМ.
a) Створювати шум
b) Демпфірувати шум
c) Інтерфейсна модуляція
d) Модулювати аналоговий сигнал
e) Управляти навантаженням
e.

27) Чому в перериванні порту RS-232 необхідно очистити прийомний буфер?
a) Щоб звільнити місце
b) Для запису відповіді
c) Щоб зняти переривання
d) Щоб переривання не скинулося
e) Щоб прочитати символ
e.

28) Архітектура мікроконтролера PIC18F252
a) Фон - Неймана
b) Неймана-Пірсона
c) Гарварда
d) Гарвардська
e) Неймана – Гарварда
d.

29) Яка залежність існує між скважністю імпульсів і формованою напругою в інтегруючому ланцюзі?
a) Логарифмічна

b) Квадратурна
c) Білінійна
d) Лінійна
e) Уніполярна
d.

30) Задати напрямок обертання двигуна постійного струму в мостовій схемі - вправо.
a) Відкрити верхні ключі
b) Відкрити нижні ключі
c) Відкрити лівий верхній і правий нижній ключ
d) Відкрити правий нижній і верхній ключ
e) Відкрити всі ключі
c.

31) Призначення шини I^2C.
a) Послідовна передача даних
b) Паралельний прийом даних
c) Паралельно-послідовна передача даних
d) Паралельно-послідовний прийом даних
e) Паралельно-лінійна передача даних
a.

32) Кількість провідників у шині I^2C.
a) 1
b) 2
c) 3
d) 4
e) 8
b.

33) Відношення обладнань у концепції шини I^2C.
a) Master -Slave
b) Slave-SubSlave
c) Master- SubSlave
d) Slave -Slave
e) Master-Slave- Master
a.

34) Ключові елементи шини I^2C включені за схемою:
a) Загальний емітер
b) Загальний колектор
c) Загальна база
d) Загальний істок
e) Загальний сток
d.

35) Біти даних у шині I^2C передаються по провіднику:
a) SCL
b) SDA
c) SDB
d) DSA
e) CLS
b.

36) Кількість виводів у мікроконтролера PIC18F252
a) 16
b) 18
c) 22
d) 28
e) 34
d.

37) Об’єм пам'яті даних у мікроконтролера PIC18F252
a) 128байт
b) 256байт
c) 1,5К
d) 2К
e) 4К
c.

38) Об’єм пам'яті програм у мікроконтролера PIC18F252
a) 1К
b) 2К
c) 4К
d) 16К
e) 32К
e.

39) Об’єм пам'яті EEPROM у мікроконтролера PIC18F252
a) 128байт
b) 256байт
c) 1k
d) 2k
e) 4k
b.

40) Кількість паралельних портів у мікроконтролера PIC18F252
a) 0
b) 1
c) 2
d) 4

e) 8
a.

41) Кількість портів введення - вивдення у мікроконтролера PIC18F252
a) 1
b) 2
c) 3
d) 4
e) 6
c.

42) Кількість таймерів у мікроконтролера PIC18F252
a) 1
b) 2
c) 3
d) 4
e) 6
d.

43) Кількість CCP – модулів у мікроконтролера PIC18F252
a) 1
b) 2
c) 3
d) 4
e) 5
b.

44) Кількість USART – модулів у мікроконтролера PIC18F252
a) 1
b) 2
c) 3
d) 4
e) 5
a.

45) Кількість USB – модулів у мікроконтролера PIC18F252
a) 1
b) 2
c) 3
d) 4
e) 0
e.

46) Кількість SPI – модулів у мікроконтролера PIC18F252
a) 1
b) 2

c) 3
d) 4
e) 5
a.

47) Кількість I^2C – модулів у мікроконтролера PIC18F252
a) 1
b) 2
c) 3
d) 4
e) 5
a.

48) Кількість каналів ЦАП MCP4921
a) 1
b) 2
c) 3
d) 4
e) 5
a.

49) Кількість каналів АЦП у мікроконтролера PIC18F252
a) 1
b) 2
c) 3
d) 4
e) 5
e.

50) Архітектура мікроконтролера PIC18F252
a) Фон - Неймана
b) Неймана-Пірсона
c) Гарварда
d) Гарвардська
e) Неймана – Гарварда
d.

51) Біти синхронізації в шині I^2C передаються по провіднику:
a) SCL
b) SDA
c) SDB
d) DSA
e) CLS
a.

52) Кількість зовнішніх переривань типу INT у мікроконтролері PIC18F252
a) 1
b) 2
c) 3
d) 4
e) 5
c.

53) Зовнішні переривання типу INT у мікроконтролері PIC18F252 перебувають на порту:
a) A
b) B
c) C
d) D
e) E
b.

54) Ініціалізація таймера RTCC здійснюється командою
a) setup_timer_4(RTCC_INTERNAL|RTCC_DIV_8);
b) setup_timer_3(RTCC_INTERNAL|RTCC_DIV_8);
c) setup_timer_2(RTCC_INTERNAL|RTCC_DIV_8);
d) setup_timer_1(RTCC_INTERNAL|RTCC_DIV_8);
e) setup_timer_0 (RTCC_INTERNAL|RTCC_DIV_8);
e.

55) Дозвіл роботи переривання від таймера RTCC здійснюється командою
a) enable_interrupts (INT_RTCC);
b) enable_interrupts(EXT_RTCC);
c) enable_interrupts(RTCC_INT);
d) enable_interrupts(RTCC_EXT);
e) enable_interrupts(INT_RTTC);
a.

56) Розрядність АЦП у мікроконтролері PIC18F252 можна встановити:
a) 4 розрядів
b) 5 розрядів
c) 6 розрядів
d) 7 розрядів
e) 8 розрядів
e.

57) Кількість виводів у мікроконтролера PIC18F252
a) 16
b) 18
c) 22
d) 28

e) 34
d.

58) Функція i2c_write (*data*) виконує
a) запис байта
b) запис біта
c) приймання байта
d) приймання біта
e) скидання байта
a.

59) По інтерфейсу SPI у мікроконтролері PIC18F252 можна в одній посилці передати максимум байт:
a) 2
b) 3
c) 4
d) 8
e) 16
a.

60) Модуль ШІМ працює з таймером:
a) timer_0
b) timer_1
c) timer_2
d) timer_3
e) timer_4
d.

61) Функція i2c_start() виконує
a) стартову посилку
b) стопову посилку
c) передачу даних
d) приймання даних
e) кінець процедури приймання/передачі
a.

62) Функція value = getc() виконує
a) передачу символу
b) приймання символу
c) передачу рядка
d) приймання рядка
e) приймання блоку
b.

63) Функція i2c_stop() виконує
a) стартову посилку

b) стопову посилку
c) передачу даних
d) приймання даних
e) кінець процедури приймання/передачі
e.

64) Функція bit_clear(*var*, *bit*) виконує
a) скидання біта
b) установка біта
c) читання біта
d) скидання байта
e) установка байта
a.

65) Функція bit_set(*var*, *bit*) виконує
a) скидання біта
b) установка біта
c) читання біта
d) скидання байта
e) установка байта
b.

66) Функція bit_test (*var*, *bit*) виконує
a) скидання біта
b) установка біта
c) перевірка біта
d) скидання байта
e) установка байта
c.

67) Функція delay_ms (*time*) виконує
a) затримка виконання більше 1 мкс
b) затримка виконання більше 10 мкс
c) затримка виконання більше 100 мкс
d) затримка виконання більше 1000 мкс
e) затримка виконання більше 10000 мкс
d.

68) Функція delay_us (*time*) виконує
a) затримка виконання більше 1 мкс
b) затримка виконання більше 10 мкс
c) затримка виконання більше 100 мкс
d) затримка виконання більше 1000 мкс
e) затримка виконання більше 10000 мкс
a.

69) Функція disable_interrupts (*level*) виконує
a) дозволити переривання
b) продовжити переривання
c) заборонити переривання
d) призупинити переривання
e) скасувати переривання
c.

70) Функція value = input_x() виконує
a) приймання біта
b) передачу біта
c) приймання байта
d) передачу байта
e) скидання байта
c.

71) Функція i2c_write (*data*) виконує
a) запис байта
b) запис біта
c) приймання байта
d) приймання біта
e) скидання байта
a.

72) Функція i8 = MAKE8(*var*, *offset*) виконує
a) об'єднати дані і адресу
b) роз'єднати дані і адресу
c) з'єднати декілька байт в одну змінну
d) зі змінної одержати один байт
e) записати байт у змінну
d.

73) Функція i16 = MAKE16(*varhigh*, *varlow*) виконує
a) об'єднати дані і адресу
b) роз'єднати дані і адресу
c) з'єднати декілька байт в одну змінну
d) зі змінної одержати один байт
e) зі змінної одержати два байти
c.

74) Функція i32 = MAKE32(*var1*, *var2*, *var3*, *var4*) виконує
a) об'єднати дані і адресу
b) роз'єднати дані і адресу
c) з'єднати декілька байт в одну змінну
d) зі змінної одержати один байт
e) зі змінної одержати два байти

c.

75) Функція output_x (*value*) виконує
a) запис байта
b) запис біта
c) приймання байта
d) приймання біта
e) скидання байта
a.

76) Функція output_high (*pin*) виконує
a) установка біта в «2»
b) установка біта в «1»
c) установка біта в «3»
d) установка біта в «0»
e) установка байта
b.

77) Функція output_low (*pin*) виконує
a) установка біта в «2»
b) установка біта в «1»
c) установка біта в «3»
d) установка біта в «0»
e) установка байта
d.

78) Функція i3c_stop() виконує
a) стартову посилку
b) нічого
c) передачу даних
d) приймання даних
e) кінець процедури приймання / передачі
b.

79) Функція puts (*string*) виконує
a) Варіант відповіді:
b) запис байта
c) запис біта
d) запис рядка
e) приймання біта
f) скидання рядка
c.

80) Функція value = read_adc() виконує
a) запис значення в АЦП
b) запис біта в АЦП

c) запис рядка в АЦП
d) приймання біта з АЦП
e) читання значення з АЦП
e.

81) Функція Функція data = i2c_read() виконує
a) запис байта
b) запис біта
c) запис рядка
d) приймання біта
e) приймання байта
e.

82) Функція value = spi_read (*data*) виконує
a) запис байта
b) запис біта
c) запис рядка
d) приймання біта
e) приймання байта
e.

83) Функція spi_write (*value*) виконує
a) запис байта
b) запис біта
c) запис рядка
d) приймання біта
e) приймання байта
a.

84) Функція lcd_putc(int8 ch) виконує
a) запис байта
b) запис біта
c) запис рядка
d) приймання біта
e) приймання байта
a.

85) Функція lcd_init() виконує
a) скидання lcd
b) ініціалізацію lcd
c) завантаження даних в lcd
d) запис біта в lcd
e) приймання байта з lcd
b.

86) Функція lcd_goto(int8 line,int8 pos) виконує
a) скидання lcd
b) ініціалізацію lcd
c) завантаження даних в lcd
d) запис біта в lcd
e) позиціонування курсору lcd
e.

87) Функція lcd_putc(int8 line, int8 pos, int8 ch) виконує
a) запис даних в lcd у координатах
b) ініціалізацію lcd
c) завантаження даних в lcd
d) запис біта в lcd
e) позиціонування курсору lcd
a.

88) Функція lcd_clear_line(int8 line) виконує
a) запис даних в lcd у координатах
b) очищення рядка в lcd
c) завантаження даних в lcd
d) запис біта в lcd
e) позиціонування курсору lcd
b.

89) Функція lcd_clear() виконує
a) скидання lcd
b) ініціалізацію lcd
c) завантаження даних в lcd
d) запис біта в lcd
e) позиціонування курсору lcd
a.

90) Розрядність типу long:
Варіант відповіді:
a) 4
b) 8
c) 16
d) 32
e) 64
e.

91) Розрядність типу float:
Варіант відповіді:
a) 4
b) 8
c) 16

d) 32
e) 64
d.

92) Розрядність типу double:
Варіант відповіді:
a) 4
b) 8
c) 16
d) 32
e) 64
e.

93) Розрядність типу char:
Варіант відповіді:
a) 4
b) 8
c) 16
d) 32
e) 64
b.

94) Правильний варіант приведення типів до byte b:
Варіант відповіді:
a) b = (int)i;
b) b = (byte)i;
c) b = int i;
d) b = int(i);
e) b = byte(i);
b.

95) byte b = 3. Виникне помилка часу виконання
Варіант відповіді:
a) b *= 2;
b) b *= 4;
c) b *= 8;
d) b *= 100;
e) b += 200;
d.

96) Операція инкремента:
Варіант відповіді:
a) ++
b) --
c) +=
d) *=

e) /=
a.

97) Операція декремента:
Варіант відповіді:
a) ++
b) --
c) +=
d) *=
e) /=
b.

98) Операція додавання
Варіант відповіді:
a) ++
b) --
c) +=
d) *=
e) /=
c.

99) Операція розподілу
Варіант відповіді:
a) ++
b) --
c) +=
d) *=
e) /=
e.

100) Операція множення
Варіант відповіді:
a) ++
b) --
c) +=
d) *=
e) /=
d.

Рекомендована література

1. Смірнов В.В., Смірнова Н.В., Пархоменко Ю.М. Архітектура та програмування периферійних інтерфейсних контролерів: підручник. Кропивницький : ЦНТУ, 2020. 278 с.
   URL: http://dspace.kntu.kr.ua/jspui/handle/123456789/10313
2. Смірнов В.В., Смірнова Н.В., Пархоменко Ю.М. Програмне забезпечення управляючих мікро - ЕОМ : навчальний посібник ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2021. – 217 с.
   URL: http://dspace.kntu.kr.ua/jspui/handle/123456789/10811
3. Смірнов В.В., Смірнова Н.В., Пархоменко Ю.М. Програмування мікроконтролерних систем : навчальний посібник ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2021. – 262 с.
   URL: http://dspace.kntu.kr.ua/jspui/handle/123456789/10812
4. Смірнов В.В., Смірнова Н.В. Архітектура адаптивної бездротової локальної мережі для управління об'єктами і пристроями. Загальнодержавний міжвідомчий науково-технічний збірник. Конструювання, виробництво та експлуатація сільськогосподарських машин. Кропівницький: ЦНТУ, 2020. Вип. 50. С. 219-229 URL: http://dspace.kntu.kr.ua/jspui/handle/123456789/11003
5. Сайт кафедри програмування комп'ютерних систем і мереж.
   URL: http://pksm.kntu.kr.ua/DEVELOPMENTS_2.html
6. Смірнов В.В. Конспект лекцій з дисципліни «Програмування мікроконтролерних систем» / В.В. Смірнов, Н.В. Смірнова. – Кропивницький : ЦНТУ.
7. Програмування мікроконтролерних систем: Методичні вказівки до виконання самостійних робіт для студентів денної форми навчання за спеціальністю 123 «Комп’ютерна інженерія / Н.В. Смірнова, В.В. Смірнов - Кропивницький : ЦНТУ.
8. PIC 18FXX2. Data Sheet. High-Performance, Enhanced Flash Microcontrollers with 10-Bit A/D / Microchip Technology Inc. - Microchip Technology Inc. – 2006 p. (www.microchip.com)

9. Програмна реалізація I2C інтерфейсу (режим провідного) / ТОВ "Мікро-Чіп" - М.: 2001. - 8 с. (www.microchip.com)
10. Модуль 10-розрядного АЦП у мікроконтролерах PIC 18FXX2 / ТОВ «Мікро-Чіп» - М.: 2001. - 10 с. (www.microchip.com)
11. Богатирьов Є. А. Енциклопедія електронних компонентів. Том 1. Великі інтегральні схеми / Є. А. Богатирьов, В. Ю. Ларін, А. Є. Лякін. – К.: МК-Прес, 2006. – 246 с.
12. Стюарт Болл Р. Аналогові інтерфейси мікроконтролерів / Стюарт Болл Р.; пров. з англ. С. Щербінін. - К.: МК-Прес, 2007. - 362 с.
13. Сід Катцен. PIC-мікроконтролери. Повне керівництво / Сід Катцен.; пров. з англ. А. Євстифєєв. - К.: МК-Прес, 2010. - 656 с.
14. Дітер Кохц. Вимірювання, керування та регулювання за допомогою PIC мікроконтролерів / Дітер Кохц.; пров. з англ. Ю. Шпак. - К.: МК-Прес, 2006. - 304 с.
15. Майкл Предко. PIC-мікроконтролери. Архітектура та програмування / Майкл Предко.; пров. з англ. Ю.В. Міщенко. - К.: МК-Прес, 2009. - 512 с.
16. Крістіан Таверньє. PIC-мікроконтролери. Практика застосування / Крістіан Таверньє.; пров. із фр. В.А. Марченка. - К.: МК-Прес, 2004. - 272 с.
17. Майкл Предко. Довідник з PIC-мікроконтролерів / Майкл Предко.; пров. з англ. Ю.В. Міщенко. - К.: МК-Прес, 2004. - 512 с.
18. Мухаммед Алі Мазіді. Мікроконтролери PIC та вбудовані системи. Застосування асемблера та C для PIC18 / Мухаммед Алі Мазіді, Ролін Д. МакКінлі, Денні Кусей.; пров. з англ. В.Литвин. - К.: МК-Прес, 2009. - 784 с.
19. Тім Вілмсхерст. Розробка вбудованих систем за допомогою мікроконтролерів PIC. Принципи та практичні приклади / Тім Вілмсхерст.; пров. з англ. В. Стаценка, В. Литвин, Юрій Шпак. – К.: МК-Прес, 2008. – 544 с.
20. Баррі Брей. Застосування мікроконтролерів PIC18. Архітектура, програмування та побудова інтерфейсів із застосуванням C та асемблера / Баррі Брей.; пров. з англ. В.Литвин. - К.: МК-Прес, 2008. - 576 с.
21. Манфред Кеніг. Повний посібник з PIC-мікроконтролерів / Ганна та Манфред Кеніг.; пров. з англ. В. Кириченко, Юрій Шпак. - К.: МК-Прес, 2007. - 256 с.

22.22. Юрій Шпак. Програмування мовою С для AVR та PIC мікроконтролерів / Юрій Шпак. - К.: МК-Прес, 2011. - 544 с.

23.John Morton. The PIC Microcontroller: Your Personal Introductory Course, Third Edition / John Morton. – Newnes, 2005. – 320 p.

24.Tim Wilmshurst. Designing Embedded Systems with PIC Microcontrollers / Tim Wilmshurst. – Newnes, 2009. – 750 p.

25.Hassan Parchizadeh. PIC Projects: A Practical Approach / Hassan Parchizadeh, Branislay Vuksanovic. – Wiley, 2009. – 210 p.

26.Lucio Di Jasio. PIC Microcontrollers: Know It All / [Lucio Di Jasio, Tim Wilmshurst, Dogan Ibrahim]. – Newnes, 2007. – 928 p.

27.Harprit Sandhu. Making PIC Microcontroller Instruments and Controllers / Harprit Sandhu. – McGraw-Hill/TAB Electronics, 2008. – 372 p.

28.Dogan Ibrahim. Advanced PIC Microcontroller Projects in C: From USB to RTOS with the PIC 18F Series / Dogan Ibrahim. – Newnes, 2008. – 560 p.

29.Martin P. Bates. Programming 8-bit PIC Microcontrollers in C: with Interactive Hardware Simulation / Martin P. Bates. – Newnes, 2008. – 304 p.

30.Martin P. Bates. PIC Microcontrollers, Third Edition: An Introduction to Microelectronics / Martin P. Bates. – Newnes, 2011. – 456 p.