Міністерство освіти і науки україни

Національний університет водного господарства та природокористування

Навчально-науковий інститут автоматики, кібернетики та
обчислювальної техніки

Кафедра автоматизації, електротехнічних та
комп'ютерно-інтегрованих технологій

04-03-253

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з навчальної дисципліни
«Електроніка та мікропроцесорні системи»
для здобувачів вищої освіти першого (бакалаврського) рівня за
спеціальністю 015.10 «Професійна освіта. Комп’ютерні технології»
денної та заочної форм навчання

Рекомендовано науково-методичною
комісією зі спеціальності
015.10 «Професійна освіта.
Комп’ютерні технології»
Протокол № 1 від 30.08.2019 р.

Рівне – 2019

Методичні вказівки до лабораторних робіт з навчальної дисципліни «Електроніка та мікропроцесорні системи» для здобувачів вищої освіти першого (бакалаврського) рівня за спеціальністю 015.10 «Професійна освіта. Комп’ютерні технології» денної та заочної форм навчання / Василець С. В., Василець К. С. – Рівне: НУВГП, 2019. – 134 с.

Укладачі:

Василець С. В. професор кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій, доктор технічних наук, доцент;

Василець К. С. асистент кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій.

Відповідальний за випуск:
Древецький В. В., завідувач кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій, докт. техн. наук, професор.

ЗМІСТ

ПЕРЕДМОВА

Метою навчальної дисципліни «Електроніка та мікропроцесорні системи» є оволодіння є формування у здобувачів вищої освіти компетентностей у галузі електроніки та мікропроцесорної техніки.

Методичні вказівки включають опис лабораторних робіт, що виконуються здобувачами вищої освіти першого (бакалаврського) рівня спеціальності 015.10 «Професійна освіта. Комп'ютерні технології».

В результаті виконання лабораторних робіт здобувачі вищої освіти мають набути знань із основних типів аналогових та цифрових мікроелектронних пристроїв, систем числення та алгебри логіки, навчитися формувати схеми «жорсткої» логіки, вивчити структуру мікропроцесорної системи, мікропроцесора та мікроконтролера, призначення, склад та порядок роботи типових вузлів та пристроїв мікропроцесорної техніки, оволодіти основними методами складання структурних, функціональних, а також принципових електричних схем блоків мікропроцесорних систем, уміти сформувати алгоритм роботи мікропроцесорних пристроїв, скласти програму для мікроконтролера, проводити тестування мікропроцесорної системи.

Тематика лабораторних робіт тісно пов'язана з лекційним матеріалом, тому під час підготовки до лабораторної роботи необхідно ґрунтовно вивчити відповідний теоретичний матеріал. Крім того, готуючись до лабораторної роботи, необхідно ознайомитися з методичними вказівками, підготувати необхідні рівняння, структурні схеми, моделі тощо. Лабораторні роботи виконуються з використанням компілятора MASM, програмних пакетів IDE Arduino, Atmel Studio, Proteus.

За результатами виконання кожної лабораторної роботи здобувач вищої освіти готує звіт відповідно до вимог, які наведені в кінці вказівок до кожної роботи. Здобувач має подати викладачеві звіт та захистити його на наступному лабораторному занятті. Під час підготовки до захисту слід орієнтуватися на перелік контрольних запитань.

ЛАБОРАТОРНА РОБОТА №1
Програмні продукти для автоматизованого проектування мікроелектронних схем, розробки програмного забезпечення та програмування мікроконтролерів

Мета: ознайомитися з призначенням, основними функціями та методами використання інтегрованого середовища розробки IDE Arduino, інтегрованого середовища розробки Atmel Studio, пакета програм Proteus, програматора AVRDUDE з графічною оболонкою SinaProg.

1. ІНСТАЛЯЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Підготовка до роботи інтегрованого середовища розробки Arduino IDE. Програма не потребує інсталяції. Достатньо розпакувати архів arduino-1.6.11-windows.zip в папку c:\Program Files\ та запустити файл **arduino.exe** (рис. 1.1).

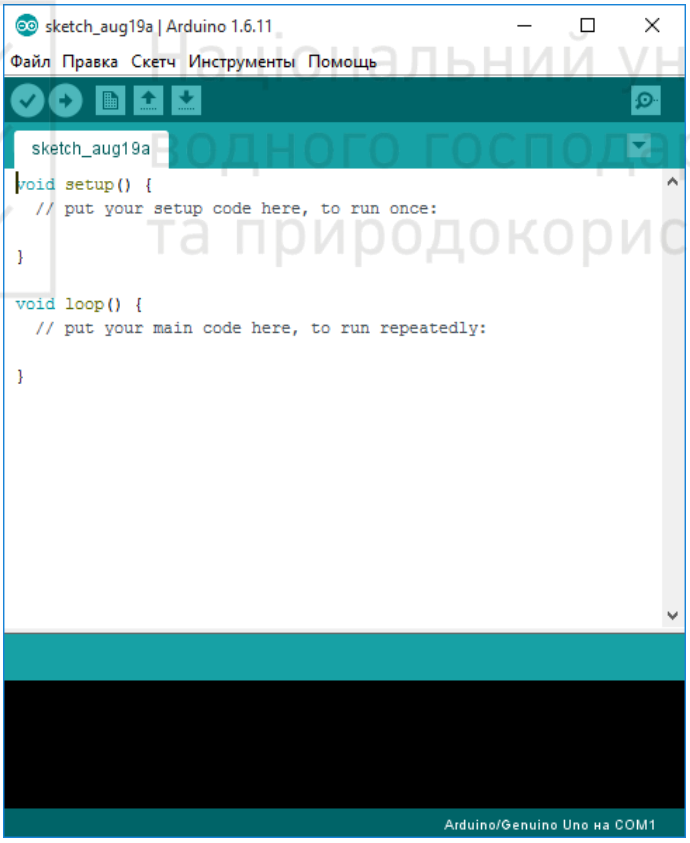


Рисунок 1.1 – Головне вікно програми Arduino IDE

1.2. Інсталяція інтегрованого середовища розробки Atmel Studio. Для інсталяції необхідно запустити файл **as-installer-7.0.1006-full.exe** і виконувати інструкції інсталятора (рис. 1.2, 1.3). Головне вікно Atmel Studio 7.0 зображено на рис. 1.4.

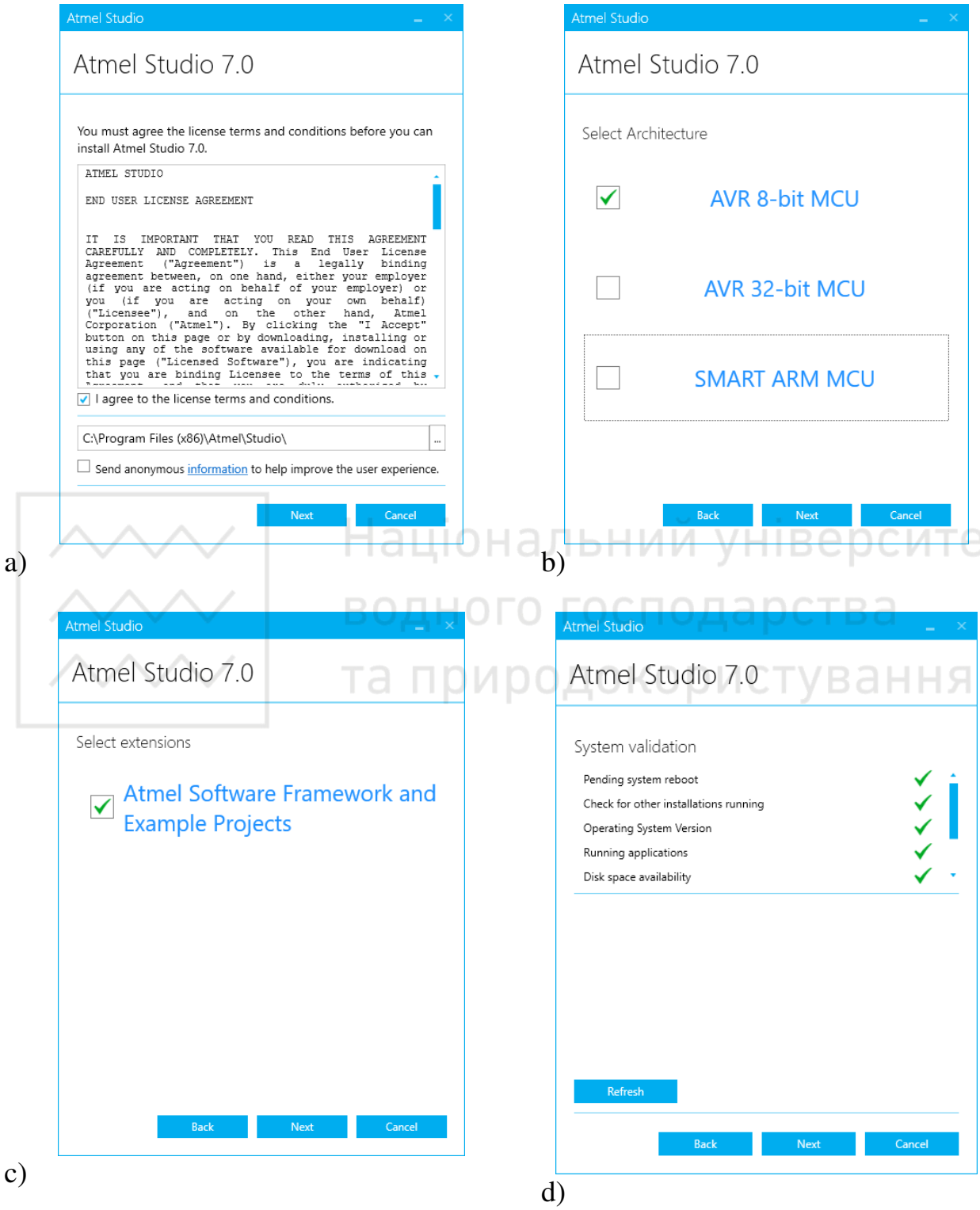


Рисунок 1.2 – Інсталяція інтегрованого середовища розробки Atmel Studio

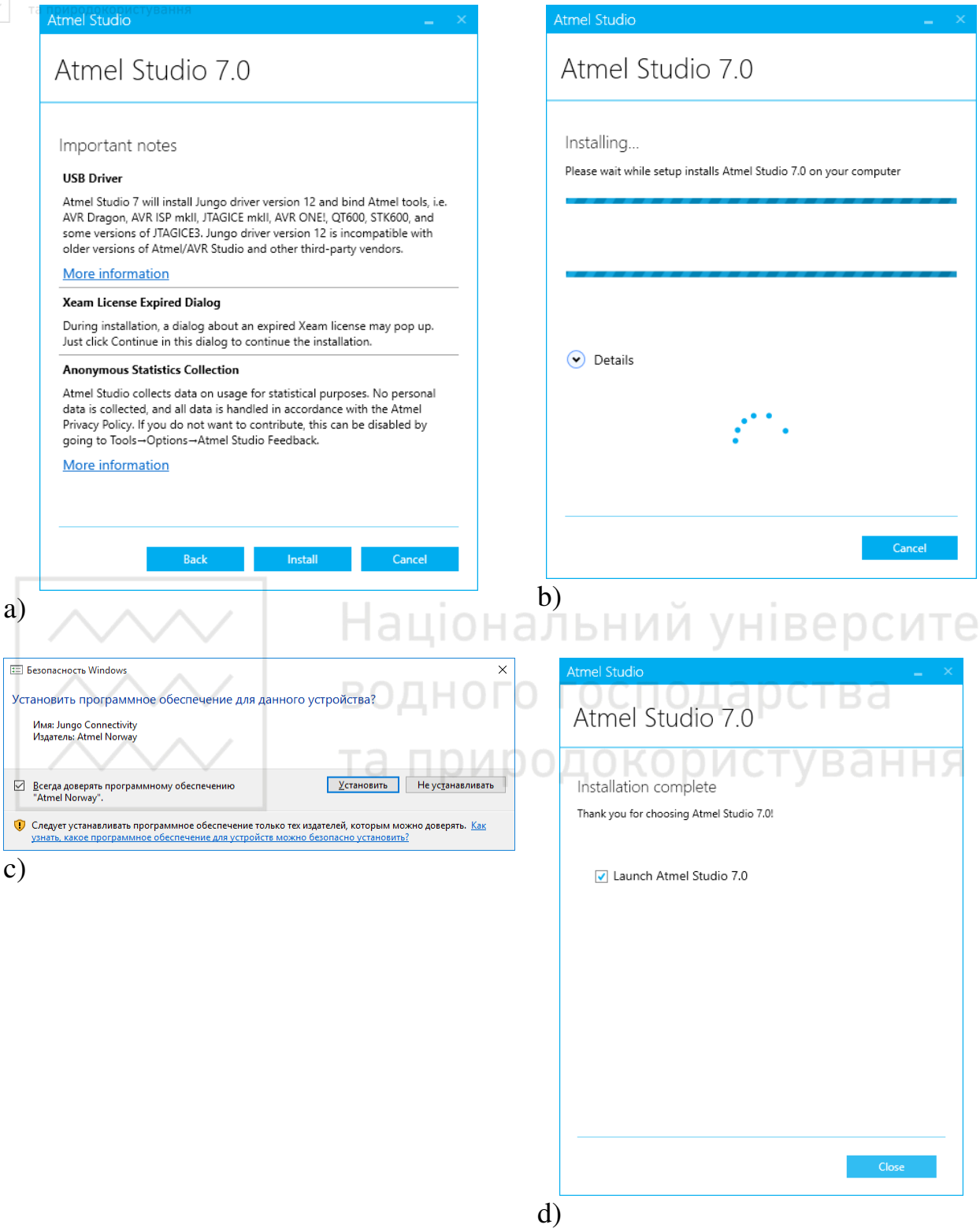


Рисунок 1.3 – Інсталяція інтегрованого середовища розробки Atmel Studio

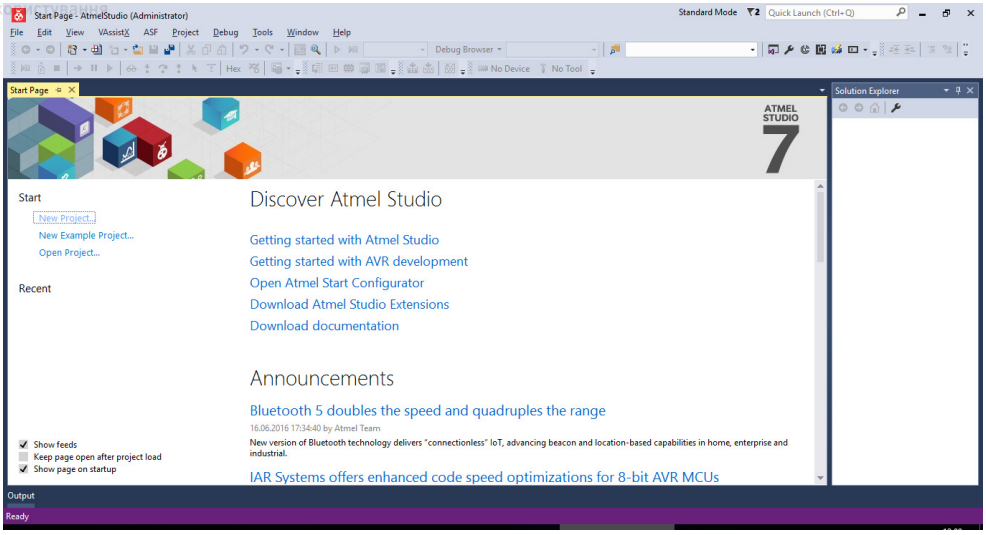


Рисунок 1.4 – Головне вікно Atmel Studio 7.0

1.3. Інсталяція пакета програм Proteus. Необхідно користуватися вказівками з файла «**Proteus install.doc**», що видається на заняттях.

1.4. Підготовка до роботи програматора AVRDUDE з графічною оболонкою SinaProg. Програма не потребує інсталяції. Достатньо розпакувати архів SinaProg2.1.1.zip в папку c:\Program Files\ та запустити файл SinaProg.exe (рис. 1.5).

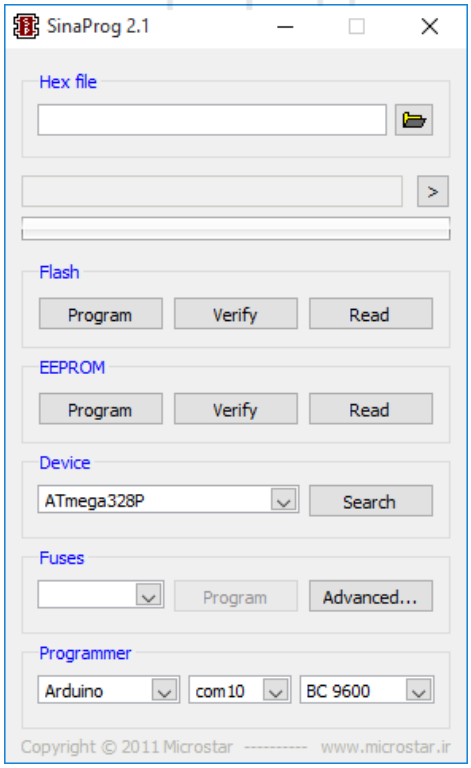


Рисунок 1.5 – Головне вікно програми SinaProg

2. КОРОТКІ ВІДОМОСТІ ЩОДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Інтегроване середовище розробки IDE Arduino

IDE – Integrated Development Environment (інтегроване середовище розробки) – термін, що описує додаток, який об'єднує редактор, засоби побудови додатків, такі як компілятор, і засоби відлагодження, тобто це середовище, де можна писати, редагувати, компілювати і перевіряти програмне забезпечення.

Розробка власних технічних рішень на базі плат, сумісних з архітектурою Arduino, здійснюється в офіційному безкоштовному середовищі програмування Arduino IDE. Середовище призначене для написання, компіляції і завантаження власних програм в пам'ять мікроконтроллера, встановленого на платі Arduino-сумісного пристрою. Основою середовища розробки є **мова Processing/Wiring** – це фактично звичайний C++, доповнений простими і зрозумілими функціями для управління введенням/виводом на контактах. Існують версії середовища для операційних систем Windows, Mac OS і Linux.

Програма, написана в середовищі Arduino, носить назву **скетч**. Скетч пишеться в текстовому редакторові, який має кольорове підсвічування створюваного програмного коду. Під час збереження і експорту проекту в області повідомлень з’являються пояснення і інформація про помилки. Вікно виведення тексту показує повідомлення Arduino, що включають повні звіти про помилки і іншу інформацію. Кнопки панелі інструментів дозволяють перевірити і записати програму, створити, відкрити і зберегти скетч, відкрити моніторинг послідовної шини.

Скетчам, що розробляються, додаткова функціональність може бути додана за допомогою бібліотек. Бібліотеки – спеціальним чином оформ-лений програмний код, що реалізовує визначені функції, який можна підключити до створюваного проекту. Спеціалізованих бібліотек існує багато. Зазвичай бібліотеки пишуться так, щоб спростити рішення тієї або іншої задачі і приховати від розробника деталі програмно-апаратної реалізації. Середовище Arduino IDE поставляється з набором стандартних бібліотек: Serial, EEPROM, SPI, Wire та ін. Вони знаходяться в підкаталозі libraries каталогу установки Arduino. Необхідні бібліотеки можуть бути також завантажені з різних ресурсів. Такі бібліотеки мають бути додані в каталог стандартних бібліотек (підкаталог libraries каталогу установки Arduino). Усередині каталогу з ім'ям бібліотеки знаходяться файли *.cpp, *.h. Багато бібліотек забезпечуються прикладами, що розміщені в examples. Якщо бібліотека встановлена правильно, то вона з'являється в меню Sketch | Import Library. Вибір бібліотеки в меню приведе до додавання в початковий код строчки:

##include <ім'я бібліотеки.h>

Ця директива підключає заголовний файл з описом об'єктів, функцій і констант бібліотеки, які тепер можуть бути використані в проекті. Середовище Arduino компілюватиме створюваний проект разом з вказаною бібліотекою.

Інтерфейс середовища розробки Ардуино містить наступні основні елементи: текстовий редактор для написання коду, область для виведення повідомлень, текстова консоль, панель інструментів з традиційними кнопками і головне меню. Це дозволяє комп'ютеру взаємодіяти з Ардуино як для передачі даних, так і для прошивки коду в контроллер.

Кнопки на панелі інструментів призначені для створення, відкриття, збереження і прошивки програм в пристрій:

***Verify** (Проверить)*

Перевірити код на помилки.

***Upload** (Прошить)*

Скомпілювати програму і "зашити" її в мікроконтроллер Ардуино.
Примітка: щоб прошити мікроконтроллер через зовнішній програматор - треба затиснути клавішу "shift" перед натисненням на цю іконку. При цьому текст біля кнопки зміниться на "Upload using Programmer".

***New** (Создать)*

Створити нову програму.

***Open** (Открыть)*

Команда відкриває меню із списком усіх скетчів, доступних у вашій робочій папці. Після клацання по файлу його вміст відкриється в поточному вікні.

***Save** (Сохранить)*

Зберегти програму

Serial Monitor

Відкрити програму "Serial Monitor" (для роботи з послідовним інтерфейсом).

У середовищі розробки Ардуино використовується принцип організації sketchbook (робочої папки): усі ваші програми (чи скетчі) зберігаються в одному місці. Щоб проглянути їх, необхідно вибрати меню File > Sketchbook або клацнути по кнопці Open на панелі інструментів. Директорія для зберігання ваших програм буде автоматично створена при першому запуску середовища Ардуино. Її місцерозташування завжди можна змінити у вікні налаштувань програми.

Починаючи з версії 1.0, файли скетчів мають розширення .ino. У попередніх версіях використовувалося розширення .pde. У нових версіях програми (1.0 і старше) файли .pde як і раніше можна відкрити, тільки їх розширення буде автоматично змінено на .ino.

2.2. Інтегроване середовище розробки Atmel Studio

Atmel Studio – інтегроване середовище розробки (IDE) від компанії Atmel для розробки додатків під мікроконтроллери ARM Cortex - M і AVR (рис. 1.6).

Програмний пакет AVR Studio розробляється з 2004 року. Починаючи з версії 6.0, програма змінила назву на Atmel Studio. Програма дозволяє працювати як на асемблері, так і на C/C++. Містить в собі майстер проектів, віртуальний симулятор, редактор початкового коду, модуль внутрішньосхемної відладки і інтерфейс командного рядка. Підтримує компілятор GCC і плагин AVR RTOS (операційної системи реального часу). Користувачі можуть вибрати

найбільш оптимальні для їх проекту способи кодування. Візуальні інструменти дозволяють прискорити написання програми. Завдяки зв'язці програмних пакетів Atmel Studio і Proteus від фірми Labcenter Electronics можливе програмування мікроконтроллерів без наявності якої-небудь матеріальної бази. Atmel Studio по праву вважається кращим середовищем створення додатків для контроллерів AVR.

Atmel Studio підтримує усі існуючі на сьогоднішній момент 8-бітові, 32-бітові AVR, SAM3 і SAM4 мікроконтроллери і включає більше 1100 проектів з прикладами. Також доступні старі версії програми.

Інтерфейс повністю англомовний. Програма не розуміє кирилицю, тому назви робіт мають бути написані латинськими буквами!!!

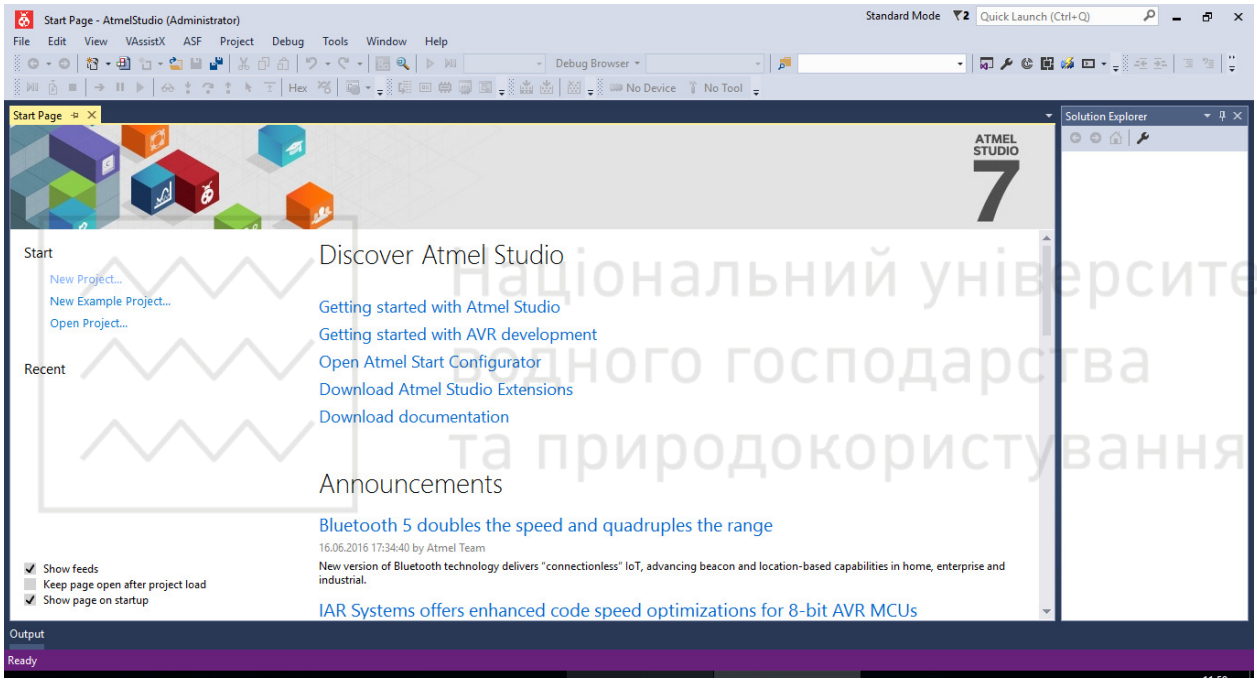


Рисунок 1.6 – Головне вікно Atmel Studio

2.3. Пакет програм Proteus для автоматизованого проектування електронних схем

Середовище віртуального моделювання Proteus – це програмний симулятор, замінюючий реальні радіодеталі і прилади віртуальними моделями. Proteus дозволяє, без зборки реального пристрою, відлагодити роботу схеми, знайти помилки, допущені на стадії проектування, зняти необхідні характеристики тощо. Допускається моделювання аналогових і цифрових пристроїв, а також мікроконтроллерів. Доступна бібліотека моделей елементів. Програма має достатній набір інструментів і функцій, серед яких вольтметр, амперметр, осцилограф, всілякі генератори, можливість відлагоджувати програмне забезпечення мікроконтроллерів та ін.

Proteus складається з двох програм:

1) ISIS - середовище віртуального моделювання;

2) ARES - трасувальник друкованих плат (в даному курсі не

використовується).

При запуску програми ISIS з'являється основне вікно (рис. 1.7), де найбільший простір відведений під вікно редагування EDIT WINDOW, в якому здійснюється створення, редагування і відладка схеми пристрою.

Ліворуч вгорі - маленьке вікно попереднього перегляду Overview Window, з його допомогою можна переміщатися по вікну редагування (клацаючи лівою кнопкою миші по вікну попереднього перегляду, ми переміщаємо вікно редагування за схемою, якщо схема не вміщується у вікно). Переміщати вікно редагування за схемою можна інакше - утримуючи натиснутою кнопку SHIFT рухати курсор миші, не натискаючи її кнопок, по вікну редагування.

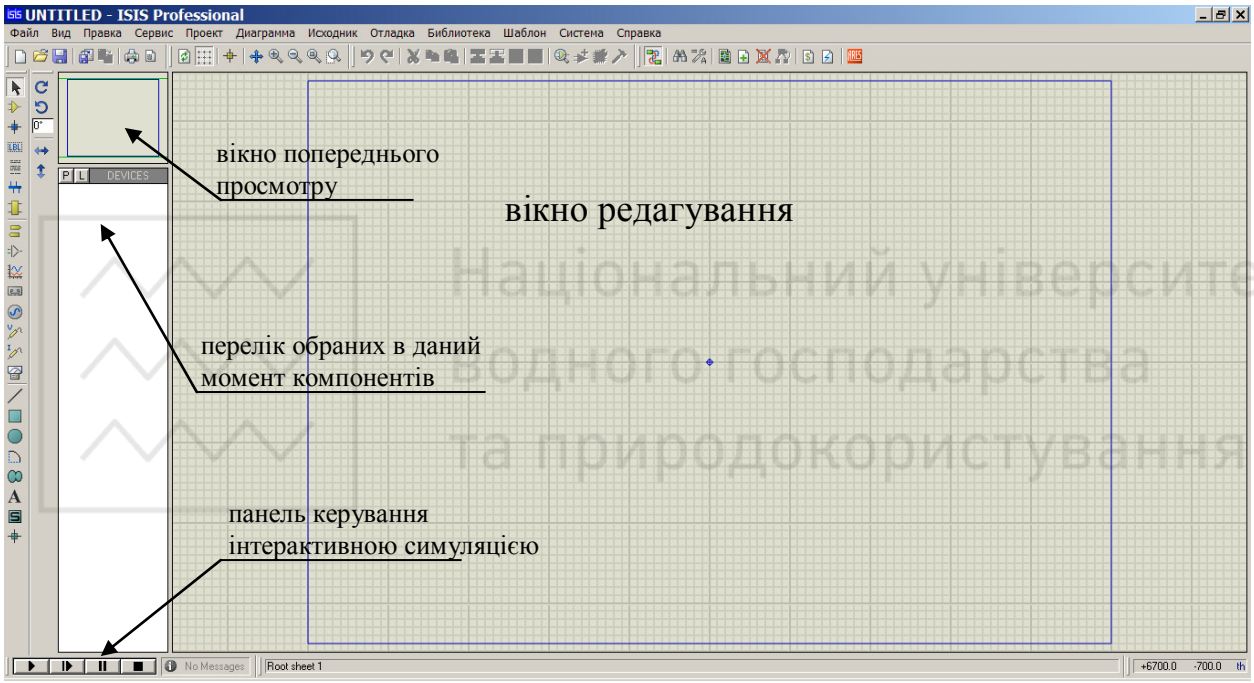


Рисунок 1.7 – Головне вікно ISIS Proteus

Наближати і віддаляти схему у вікні можна відповідно кнопками F6 і F7 або ж колесом миші, F5 центрує схему у вікні а натиснення F8 підганяє розмір схеми під вікно редагування. Під вікном попереднього перегляду знаходиться Object Selector - список вибраних в даний момент компонентів, символів і інших елементів. Виділений в списку об'єкт відображується у вікні попереднього перегляду.

Внизу вікна знаходиться панель управління інтерактивною симуляцією (схожа на магнітофонну панель, функції наступні: пуск - покроковий режим - пауза - стоп).

Приклад складання принципової схеми мікропроцесорного пристрою в редакторі ISIS ілюструє рис. 1.8.

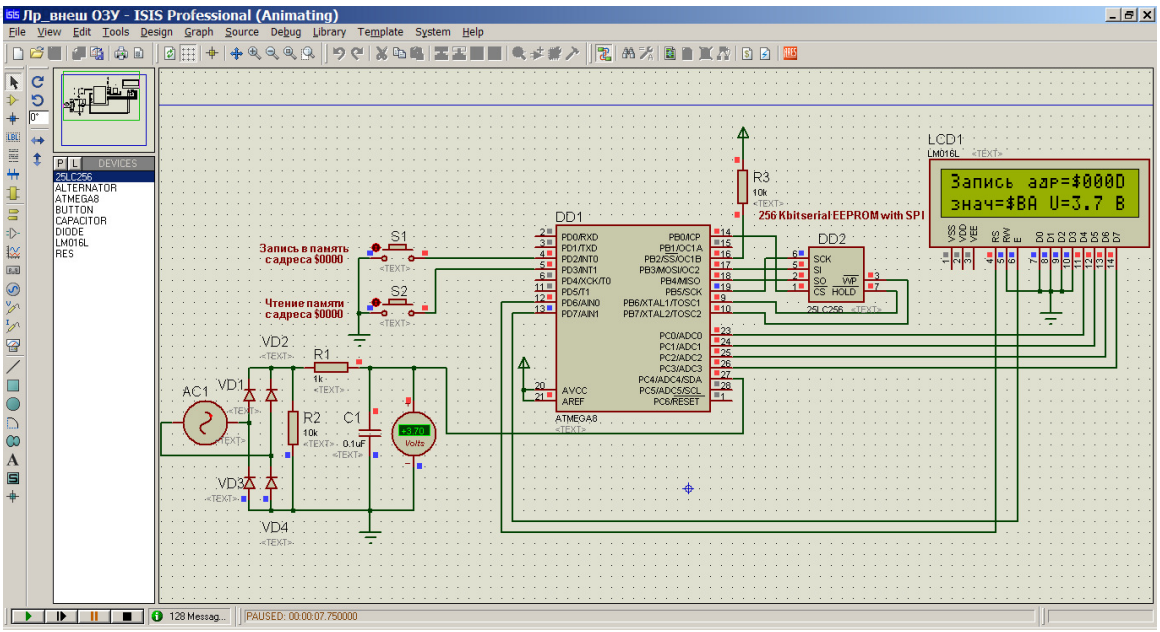


Рисунок 1.8 – Приклад принципової схеми мікропроцесорного пристрою у ISIS Proteus

2.4. Програматор AVRDUDE з графічною оболонкою SinaProg

AVRDUDE – програма для завантаження hex-файлів програм в контроллери фірми ATMEL (сайт програми: http://www.nongnu.org/avrdude/), рис. 1.9. Програматор не має графічного інтерфейсу і працювати з ним можливо тільки в консольному режимі (за допомогою командного рядка). Наприклад, типова команда для прошивки контроллера має вигляд:

```
avrdude - p atmega328p - c arduino - P /dev/ttyACM0 - b 115200 - U flash: w: blink.hex
```

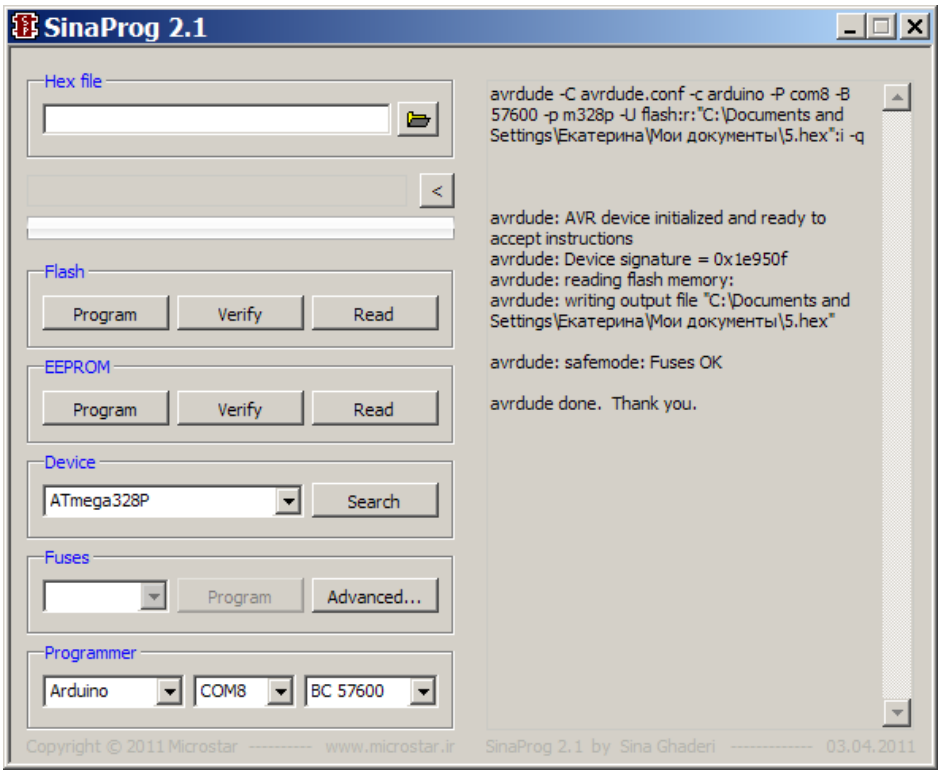


Рисунок 1.9 – Головне вікно SinaProg

Програма підтримує роботу з багатьма типами програматорів, в т.ч. з програматорами, розміщеними на платах Arduino.

Для полегшення програмування розроблена графічна оболонка SinaProg, яка, залежно від налаштувань, формує необхідні команди для AVRDUDE. Програмне забезпечення SinaProg є безкоштовним і вільно поширюється. Ця утиліта портативна, тобто працює з будь-якого місця без установки. Окрім власних файлів вона включає програму AVRdude.

3. Порядок використання програмного забезпечення. Взаємодію між програмами та порядок їх використання під час виконання лабораторних робіт ілюструє схема на рис. 1.10.

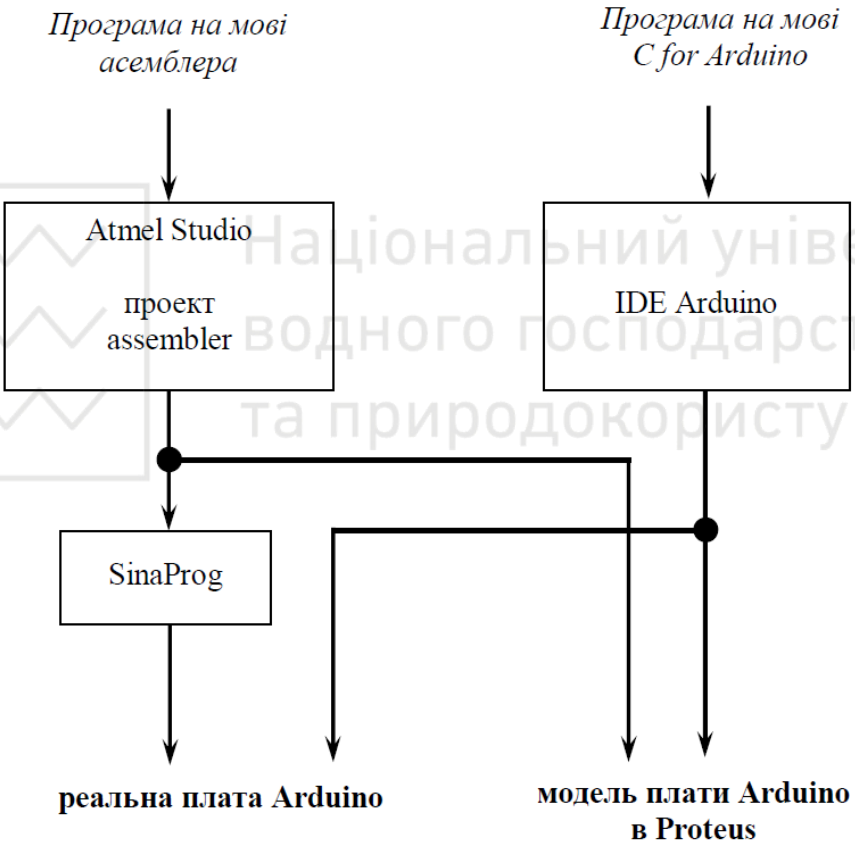


Рисунок 1.10 – Схема для ілюстрації взаємодії програмного забезпечення

КОНТРОЛЬНІ ПИТАННЯ

1. Для чого використовується інтегроване середовище Arduino IDE?
2. Як називається програмний код в середовищі Arduino IDE?
3. Яка мова використовується для написання програм в середовищі Arduino IDE?
4. Назвіть функції середовища розробки Atmel Studio.
5. Для чого призначений Proteus? Розкрити основні можливості.

ЛАБОРАТОРНА РОБОТА №2
Дослідження активно-ємнісних диференціаторів та інтеграторів

Мета: дослідити перехідні процеси в схемах активно-ємнісних диференціаторів та інтеграторів при ступінчатій зміні вхідного сигналу.

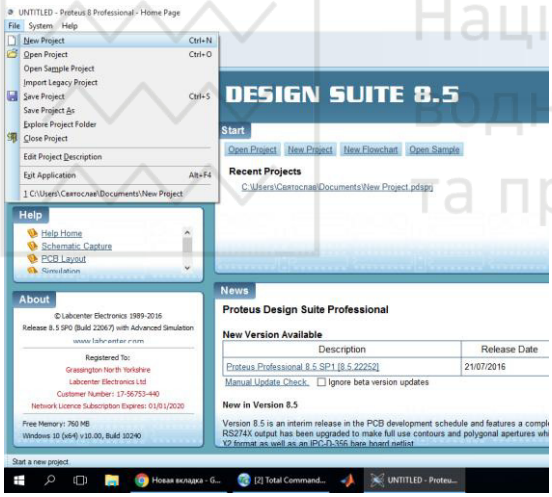
1. ЗАВДАННЯ

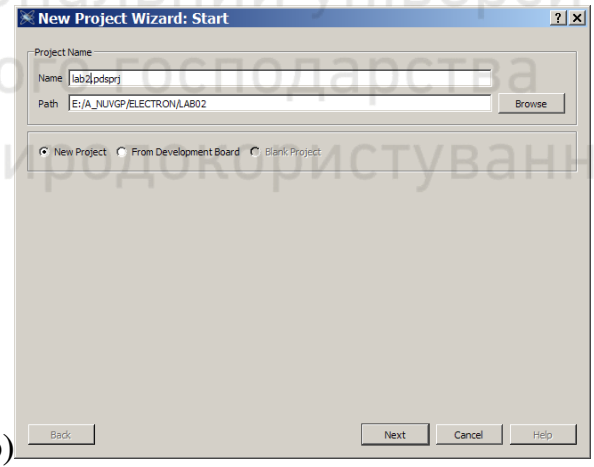
З використанням симулятора Proteus дослідити функціонування схем активно-ємнісних диференціаторів та інтеграторів при подачі на вхід прямокутних імпульсів напруги.

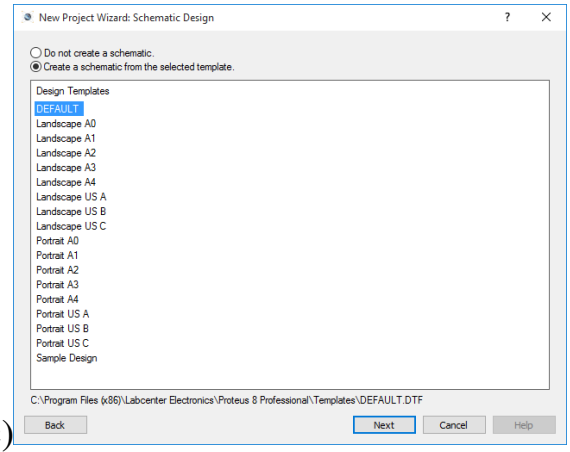
2. ПОРЯДОК ВИКОНАННЯ РОБОТИ

2.1 Створення нового проекту в симуляторі Proteus

Запустити програму Proteus. Створити новий проект (File -> New Project), встановити наступні властивості проекту відповідно до рис. 2.1, 2.2.

a) 

b) 

c) 

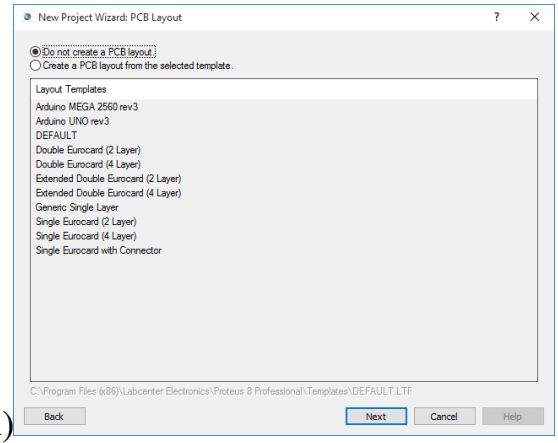
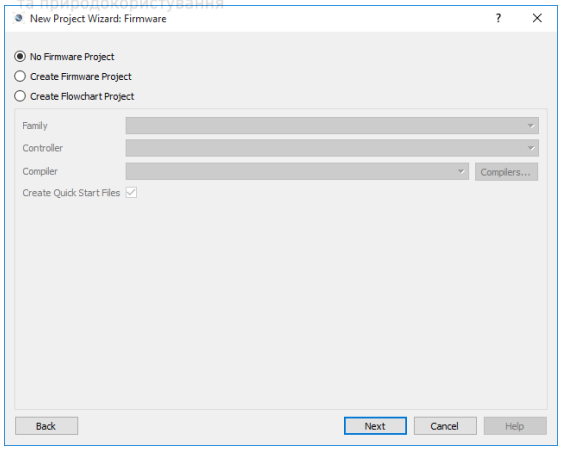
d) 

Рисунок 2.1 – Встановлення основних можливостей проекту Proteus

a) 

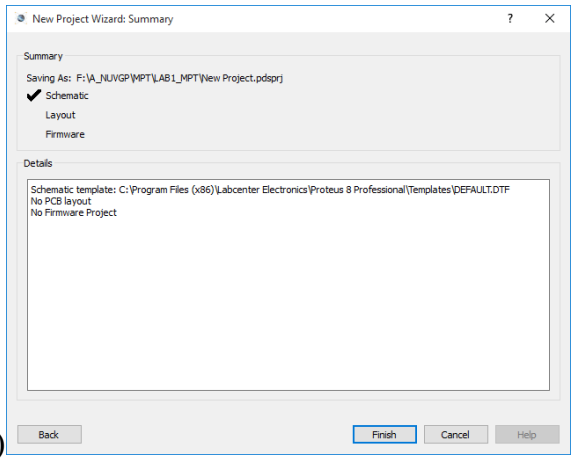
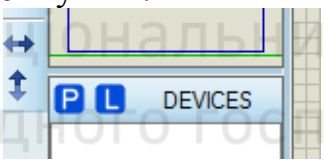
f) 

Рисунок 2.2 – Встановлення основних можливостей проекту Proteus

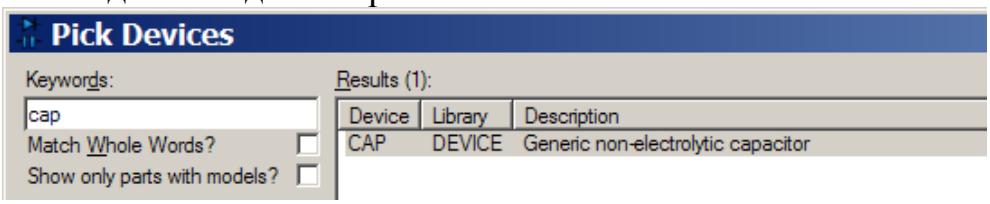
2.2 Вибір необхідних елементів

Після створення нового проекту з'явиться вікно нового документа Schematic. Для вибору необхідних компонентів необхідно відкрити бібліотеки, для чого необхідно натиснути кнопку «P»:

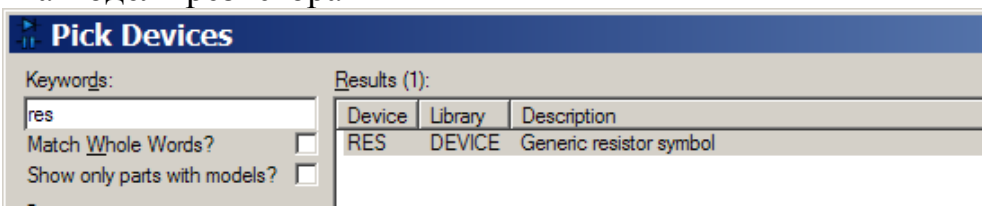


На оперативну панель необхідно додати наступні елементи:

- загальна модель конденсатора

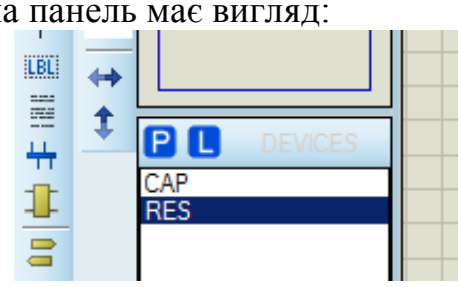


- загальна модель резистора

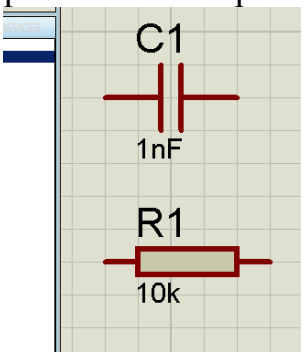


Для додавання необхідного елемента на оперативну панель необхідно на ньому 2 рази клацнути мишею.

Після цього оперативна панель має вигляд:

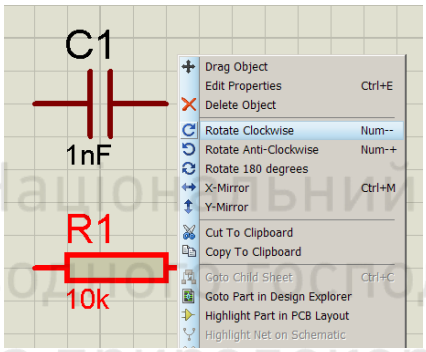


Далі необхідно ці елементи розмістити на робочому полі:

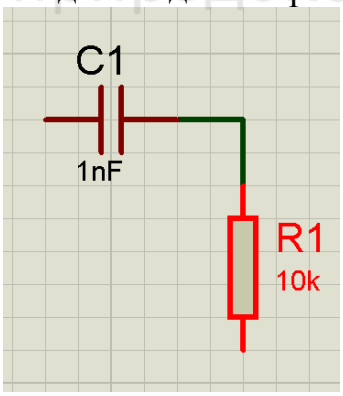


2.3 Створення принципової схеми

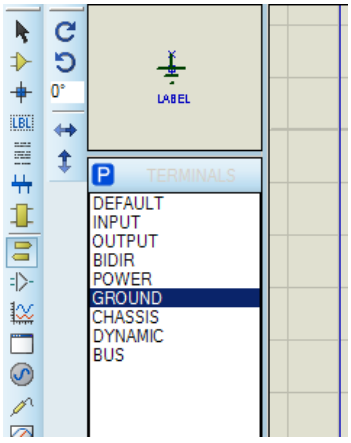
Резистор необхідно розвернути вертикально, для чого слід скористатися командою контекстного меню «Rotate Clockwise»:



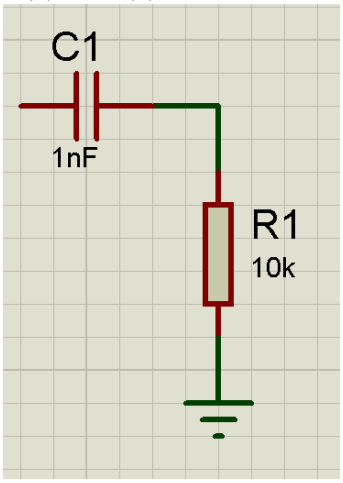
Резистор і конденсатор необхідно з’єднати провідником:



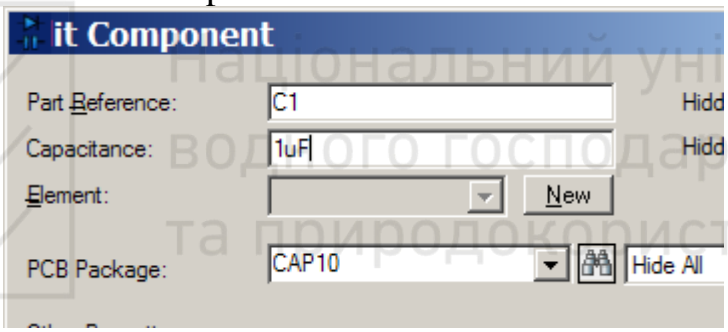
На панелі інструментів «Terminals» обрати елемент «Ground» (загальний вивід):



Цей елемент необхідно приєднати до схеми наступним чином:



Необхідно задати ємність конденсатора. Для цього слід на зображенні конденсатора двічі клацнути і у вікні налаштувань значень параметрів задати необхідне значення в полі «Capacitance»:



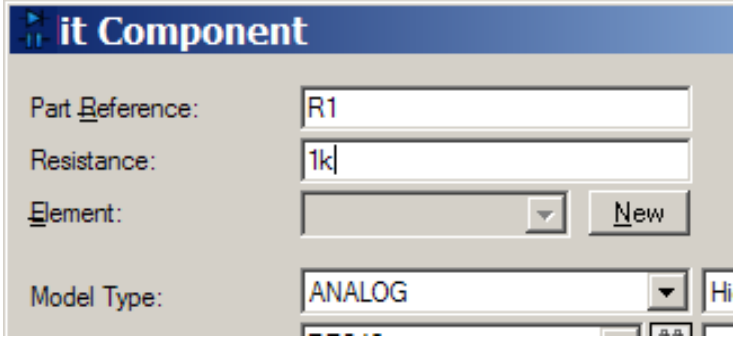
При цьому слід враховувати прийняті позначення одиниць вимірювання та похідних одиниць:

вольт – V, ампер – A, ом – Ohm (або R) , фарад - F, генрі - H, секунда - s, герц – Hz;

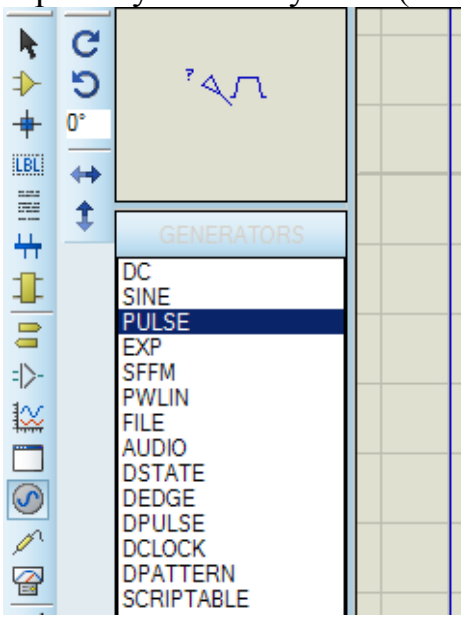
нано – n, мікро – u, мілі – m, кило – k, мега – M, гига – G.

Одиниці вимірювання допускається явно не позначати, наприклад: 5 кОм може бути позначено як «5k».

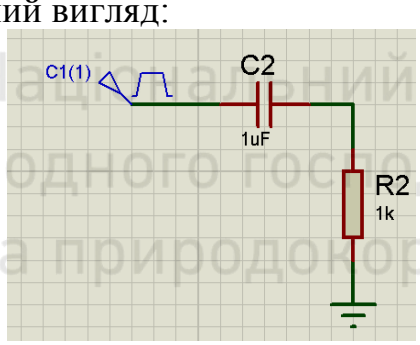
Для резистора необхідне значення опору задається в полі «Resistance»:



З панелі генераторів «Generators» необхідно підключити до входу електричного кола генератор прямокутних імпульсів (елемент PULSE):



Модель матиме наступний вигляд:



Для відображення вікна налаштувань параметрів генератора необхідно на ньому двічі клацнути мишею. У вікні (рис. 2.3) необхідно встановити частоту прямокутних імпульсів 50 Гц – параметр Frequency.

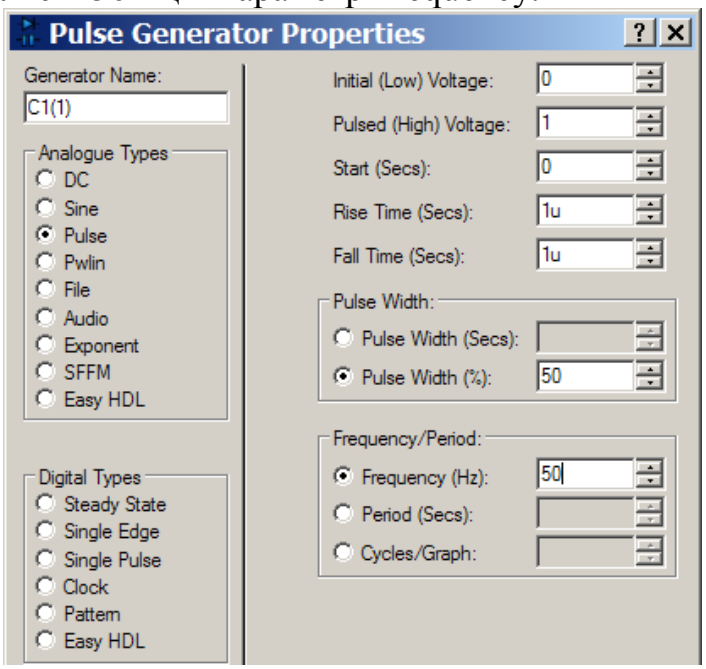
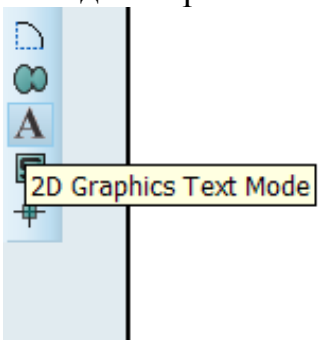
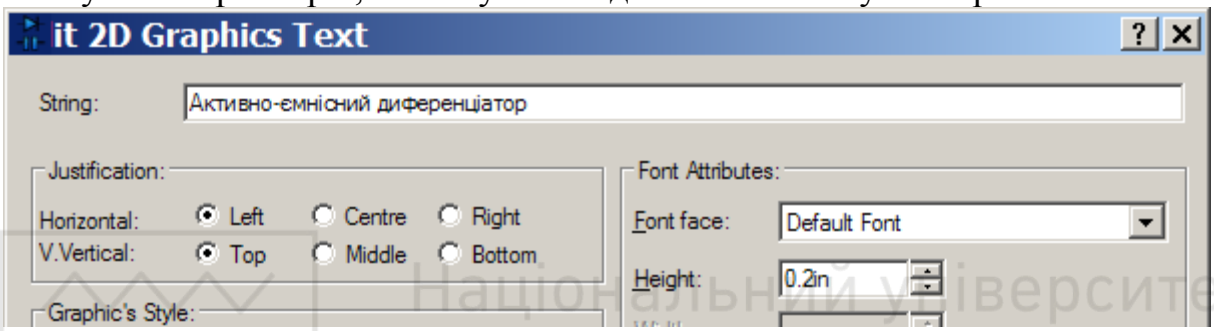


Рисунок 2.3 – Вікно налаштування значень параметрів генератора

Для додавання назви моделі слід використати елемент текстового напису:

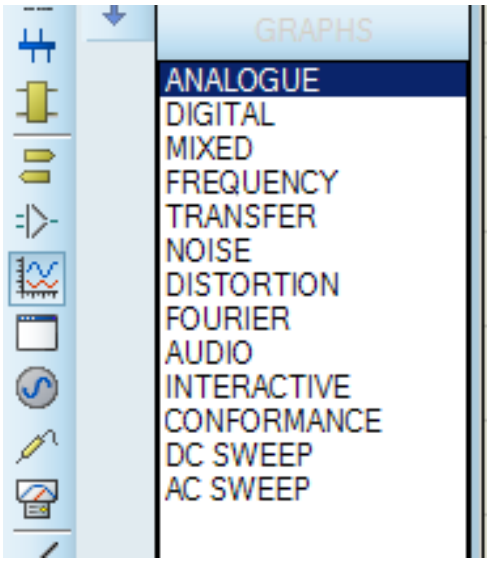


Після додавання вказаного елементу до моделі з’являється вікно налаштувань параметрів, в якому необхідно ввести назву електричного кола:



2.4 Додавання вікна для побудови графіків

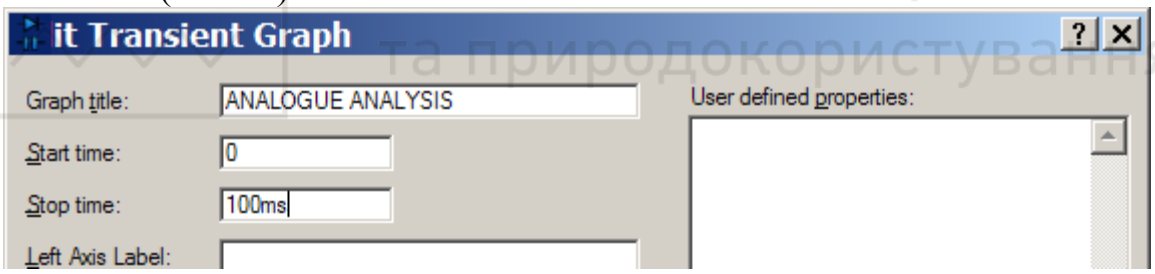
Після цього необхідно додати до моделі вікно для побудови графіків. Для цього необхідно скористатися елементом «ANALOGUE» з панелі інструментів «GRAPHS»:



Вигляд моделі ілюструє рис. 2.4.

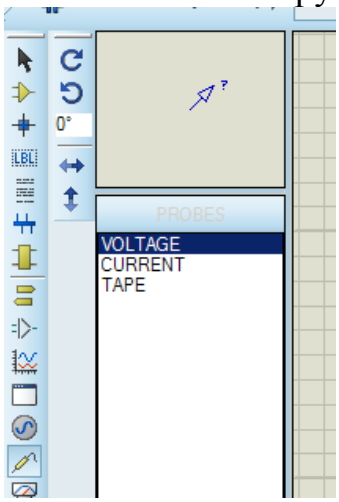
Рисунок 2.4 – Вигляд моделі активно-ємнісного диференціатора у Proteus

У властивостях вікна для побудови графіків необхідно встановити час моделювання (100 мс):



2.5 Вимірювання вихідної напруги

Вихідна напруга кола знімається з резистора R1. Для її вимірювання використовується елемент VOLTAGE з панелі інструментів PROBES:



Елемент VOLTAGE необхідно підключити в точці з'єднання конденсатора з резистором, як показано на рис. 2.5.

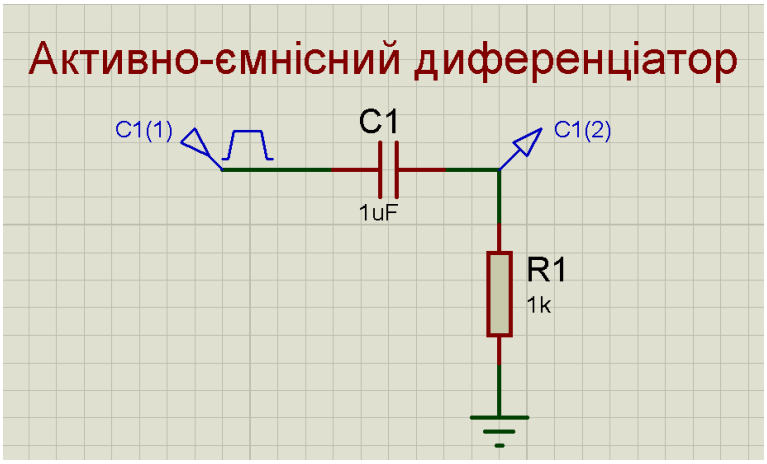
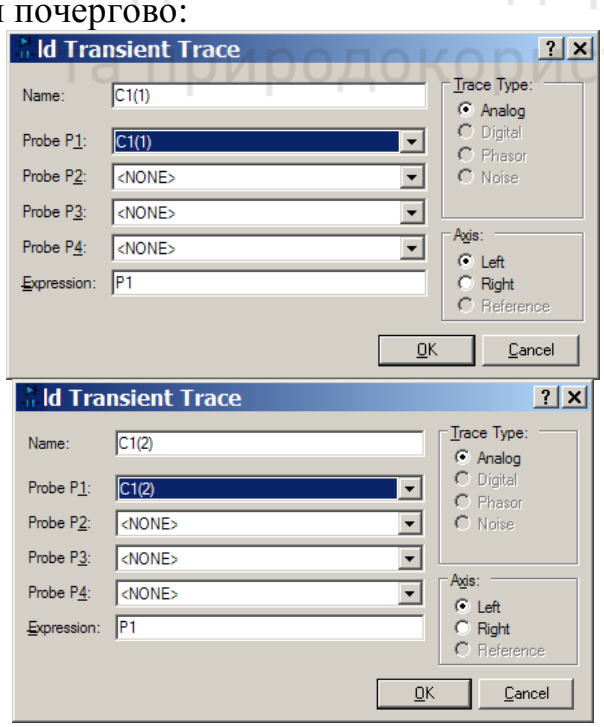


Рисунок 2.5 – Підключення до моделі вимірювального пристрою

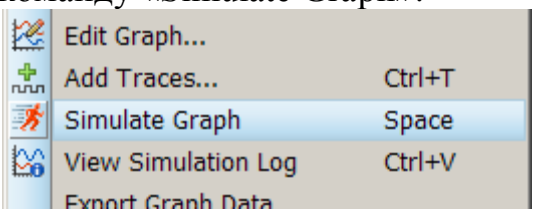
Таким чином, прямокутні імпульси напруги, що подаються на вхід схеми позначені міткою C1(1), а напруга на виході схеми – міткою «C1(2)». Для додавання графіків вказаних величин у вікно для побудови графіків, необхідно у контекстному меню вказаного вікна для побудови графіків для кожної величини обирати команду «Add Traces…» і у вікні налаштування параметрів вказувати дані величини почергово:



В результаті ці величини будуть вказані у лівому верхньому кутку вікна побудови графіків:

2.6 Здійснення моделювання

Для виконання моделювання необхідно з контекстного меню вікна для побудови графіків обрати команду «Simulate Graph»:



В результаті у вікні будуть відображені графіки напруг на вході та виході активно-ємнісного диференціатора, як показано на рис. 2.6

Рисунок 2.6 – Графіки вхідної та вихідної напруги диференціатора

Схема активно-ємнісного інтегратора, що побудована аналогічним чином, наведена на рис. 2.7.

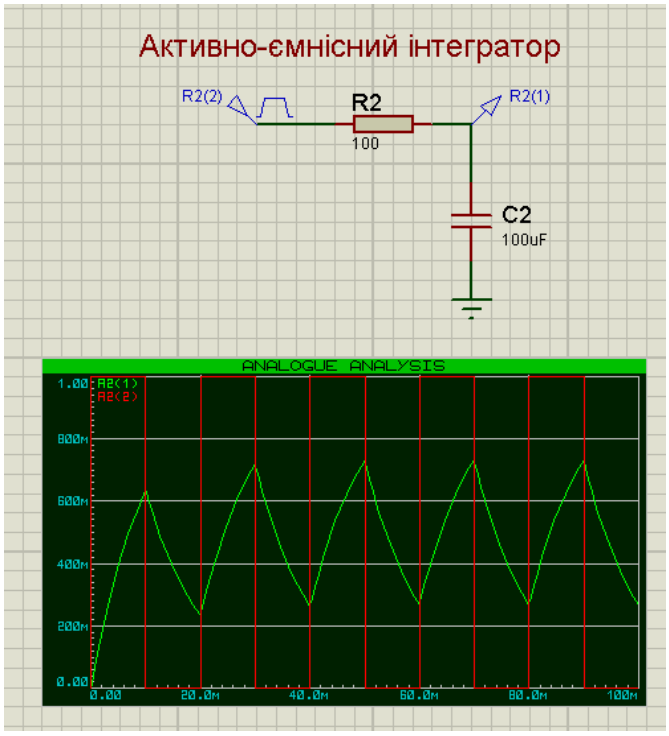


Рисунок 2.7 – Графіки вхідної та вихідної напруги інтегратора

2.7 Дослідження процесів в схемі інтегратора при блокуванні розряду ємності

На вхід схеми інтегратора (рис. 2.7) необхідно додати діод (елемент DIODE) і проаналізувати графік напруги на виході при частоті вхідного сигналу 100 Гц, рис. 2.8.

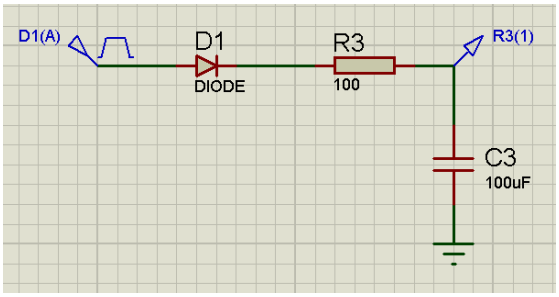


Рисунок 2.8 – Схема інтегратора при блокуванні розряду ємності

ВМІСТ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

1. Тема, мета роботи.
2. Схема активно-ємнісного диференціатора в Proteus, графіки вхідної та вихідної напруги.
3. Схема активно-ємнісного інтегратора в Proteus, графіки вхідної та вихідної напруги.
4. Схема активно-ємнісного інтегратора при блокуванні розряду ємності в Proteus, графіки вхідної та вихідної напруги.
5. Висновки з аналізом отриманих результатів.

КОНТРОЛЬНІ ПИТАННЯ

1. Накреслити принципову схема активно-ємнісного інтегратора.
2. Накреслити принципову схема активно-ємнісного диференціатора.
3. Чим відрізняються схеми активно-ємнісного інтегратора та диференціатора?
4. Який сигнал буде на виході диференціатора, якщо на вхід подаються прямокутні імпульси? Накреслити епюри вхідної та вихідної напруги.
5. Який сигнал буде на виході інтегратора, якщо на вхід подаються прямокутні імпульси? Накреслити епюри вхідної та вихідної напруги.
6. Який сигнал буде на виході диференціатора, якщо на вхід подається незмінний у часі додатний сигнал за напругою? Накреслити епюри вхідної та вихідної напруги.
7. Який сигнал буде на виході інтегратора, якщо на вхід подається незмінний у часі додатний сигнал за напругою? Накреслити епюри вхідної та вихідної напруги.

ЛАБОРАТОРНА РОБОТА №3
Дослідження функціонування мільтивібратора

Мета: дослідити автоколивальний режим функціонування мультивібратора, що побудований на транзисторній елементній базі.

1. ЗАВДАННЯ

З використанням симулятора Proteus отримати графік вихідної напруги транзисторного мультивібратора з колекторно-базовими зв'язками.

2. ПОРЯДОК ВИКОНАННЯ РОБОТИ

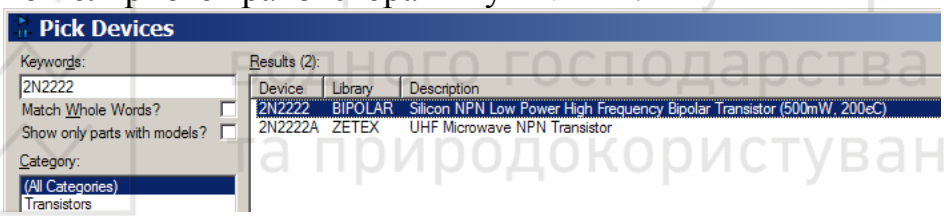
2.1. Створення нового проекту в симуляторі Proteus

Запустити програму Proteus. Створити новий проект (File -> New Project).

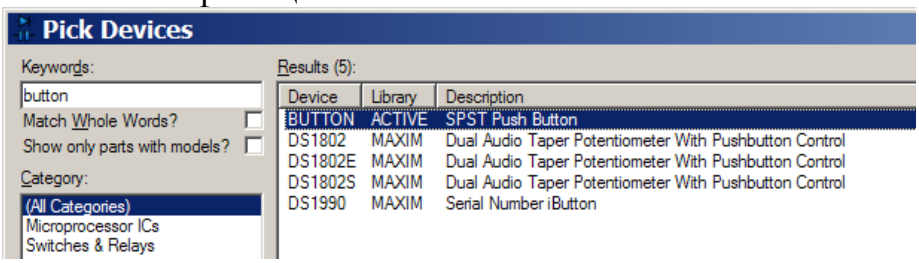
2.2. Вибрати необхідні елементи.

На оперативну панель необхідно додати наступні елементи:

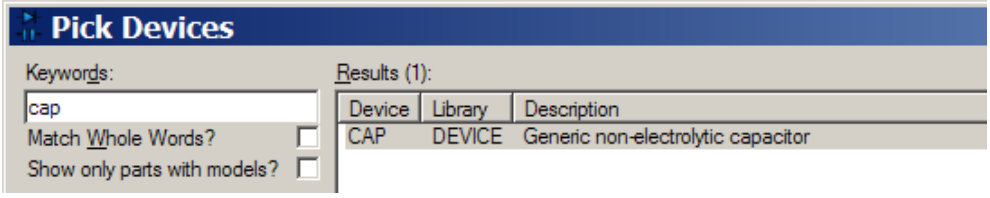
- модель біполярного транзистора типу 2N2222:



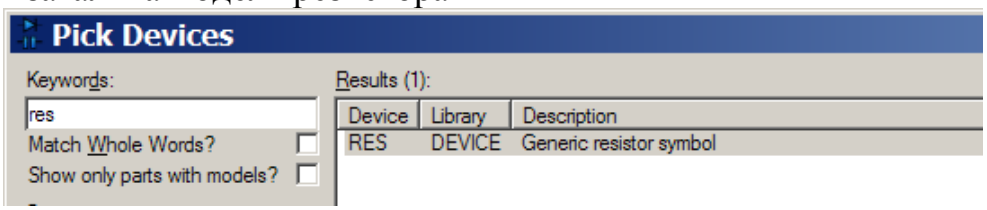
- модель кнопки без фіксації:



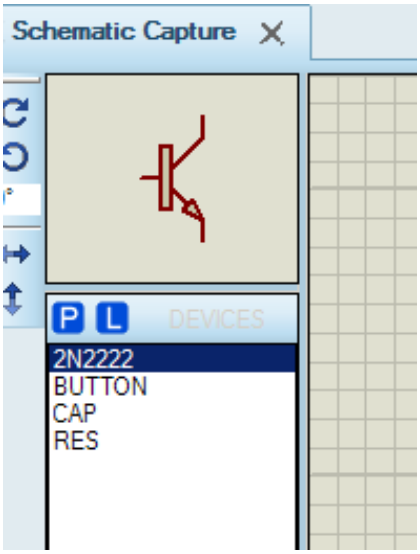
- - загальна модель конденсатора



- загальна модель резистора



Оперативна панель матиме наступний вигляд:



2.3. Формування принципової схеми мультивібратора

Необхідно виконати принципову електричну схему мультивібратора та вказати значення параметрів компонентів, як показано на рис. 3.1.

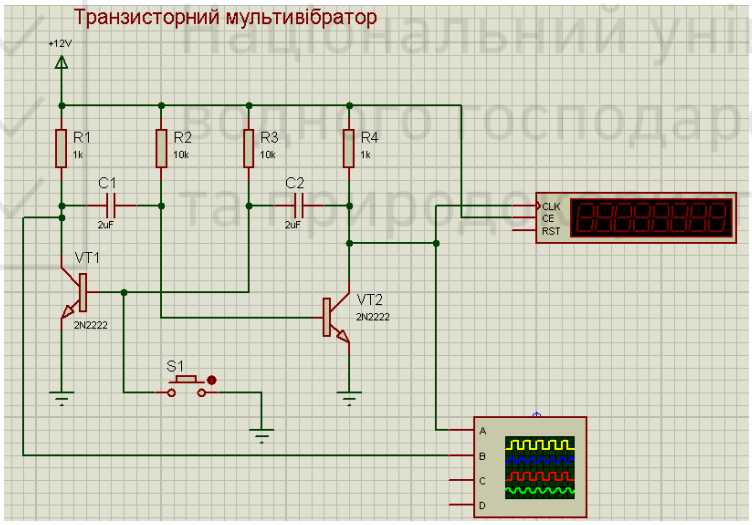
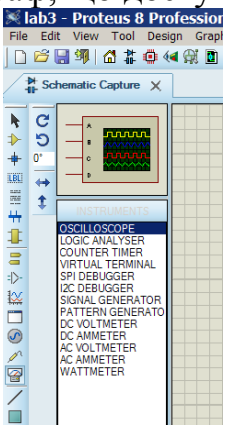
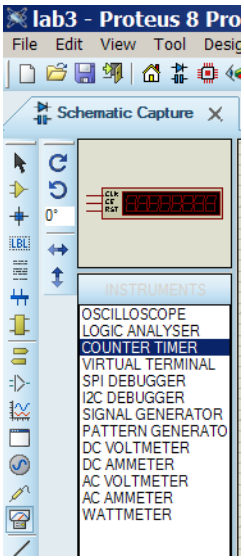


Рисунок 3.1 - Принципова схема мультивібратора

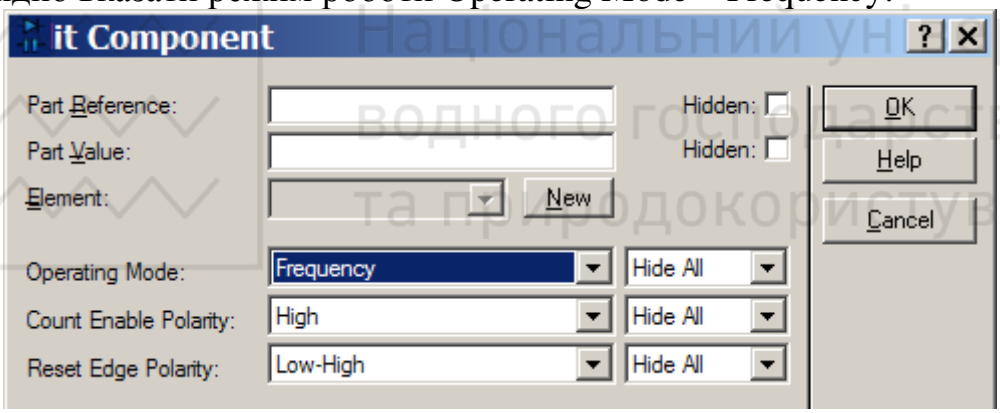
Для отримання осцилограм зміни напруг на колекторах транзисторів VT1 і VT2 використовується осцилограф, що доступний на панелі INSTRUMENTS:



Для вимірювання частоти напруги на виході мільтивібратора використовується частотомір, в якості якого використовується таймер-лічильник Counter Timer, що доступний на панелі INSTRUMENTS:



Після додавання частотоміра до схеми пристрою, в його налаштуваннях необхідно вказати режим роботи Operating Mode – Frequency:



2.4. Запуск мультивібратора

Для запуску моделі мільтивібратора необхідно натиснути кнопку Run the simulation  на панелі керування процесом моделювання:

В реальній схемі мільтивібратора автоколивання виникають при подачі живлення на схему, оскільки параметри елементів схеми не є абсолютно однаковими. Модельні елементи є ідеальними, тому для запуску мільтивібратора необхідно створити несиметрію у схеми, для чого треба натиснути на кнопку S1. У вікні осцилографа можна спостерігати зміну напруг на колекторах транзисторів, які змінюються в протифазі, рис. 3.2.

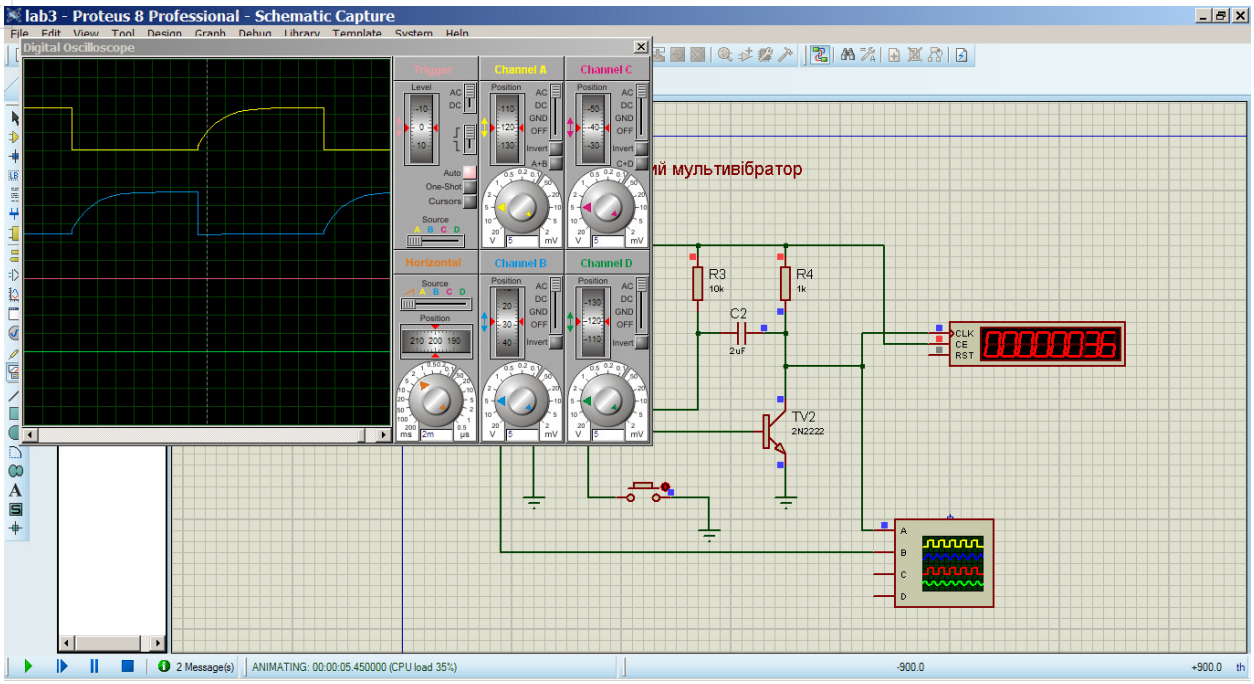


Рисунок 3.2 – Осцилограми напруг мультивібратора

2.5. Визначення частоти автоколивань

Визначимо за графіками напруг частоту автоколивань, рис. 3.3.

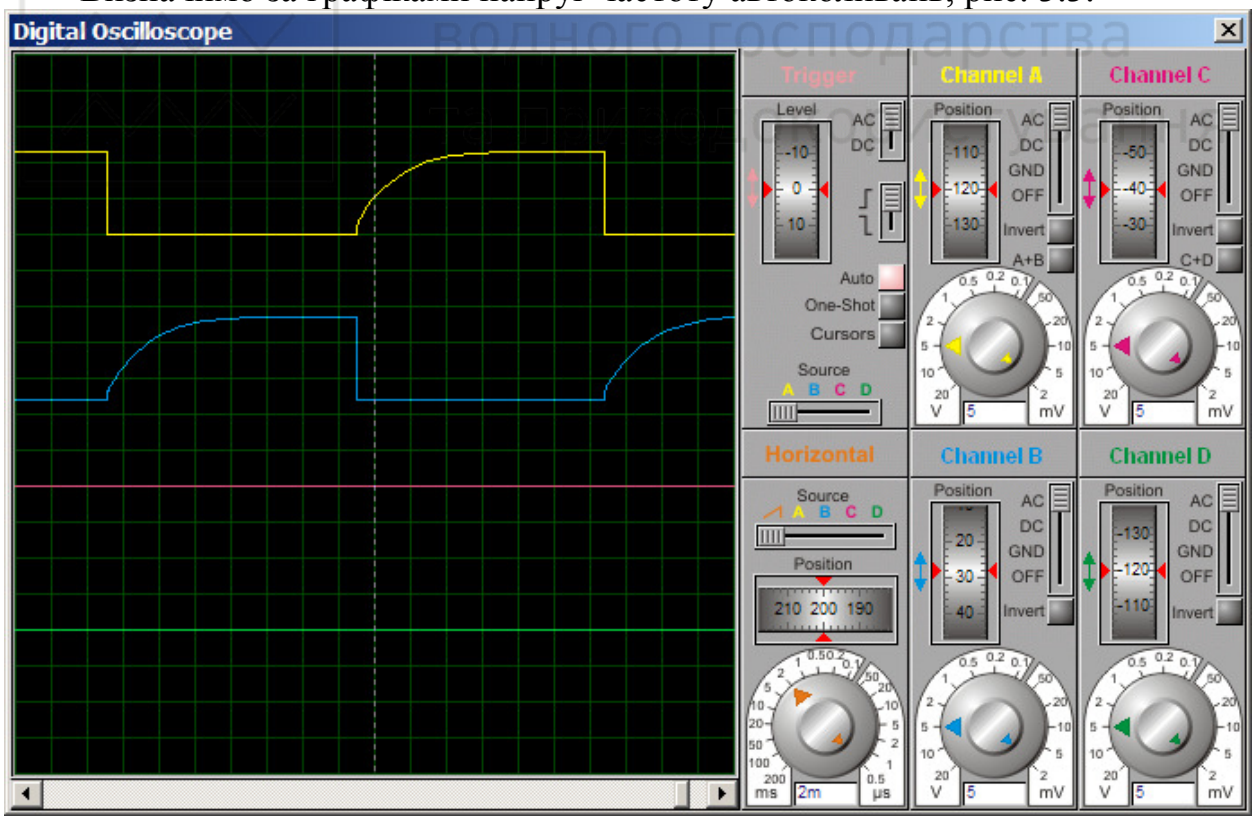


Рисунок 3.3 – Вікно осцилограма з графіками напруг мультивібратора

Як видно з налаштувань осцилографа, масштаб за часом становить 2 мс, тобто одна клітинка за віссю абсцис відповідає 2 мс. З урахуванням цього, період коливань становить T=28 мс. Відповідно, частота коливань складає f=1/T=1/0,028=36 Гц. Таке ж значення частоти показує частотомір, рис. 3.2.

ВМІСТ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

1. Тема, мета роботи.
2. Схема транзисторного мультивібратора з колекторно-базовими зв’язками в симуляторі Proteus.
3. Вікно осцилографа з графіками зміни напруг на колекторах транзисторів.
4. Визначення частоти коливань.
5. Висновки з аналізом отриманих результатів.

КОНТРОЛЬНІ ПИТАННЯ

1. Для чого призначений мультивібратор.
2. Назвіть основні компоненти прицпової схеми мультавібратора.
3. Поясніть, як за осцилограмою вихідної напруги мультивібратора визначити її частоту?
4. Як можна регулювати частоту імпульсів на виході мультивібратора?

ЛАБОРАТОРНА РОБОТА №4
Дослідження основних схем ввімкнення операційного підсилювача

Мета: проаналізувати роботу операційного підсилювача у складі типових схем.

1. ЗАВДАННЯ

З використанням симулятора Proteus дослідити диференційне, інвертуюче, неінвертуюче ввімкнення операційного підсилювача, а також побудову схем підсумовування напруг, інтегратора та суматора з використанням операційного підсилювача.

2. ПОРЯДОК ВИКОНАННЯ РОБОТИ

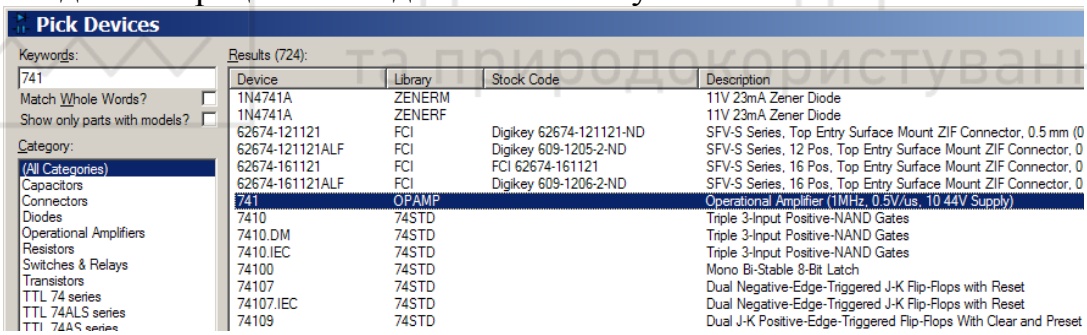
2.1. Створити новий проект в симуляторі Proteus.

Запустити програму Proteus. Створити новий проект (File -> New Project). Для кожної схеми рекомендується створювати новий проект.

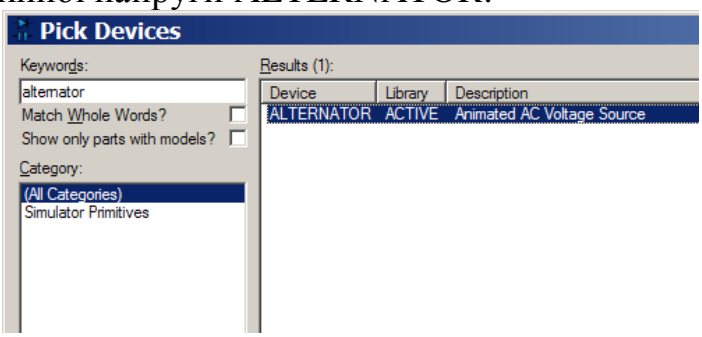
2.2. Вибрати необхідні елементи.

На оперативну панель необхідно додати наступні елементи:

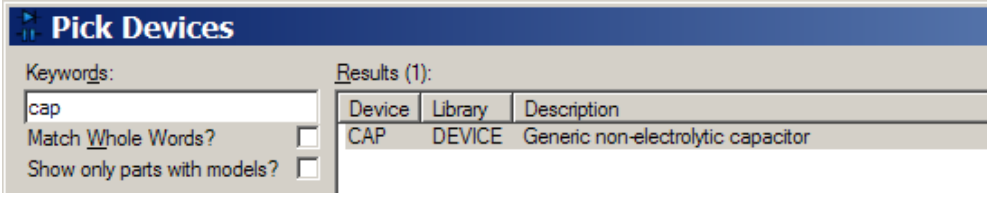
- модель операційного підсилювача типу 741:



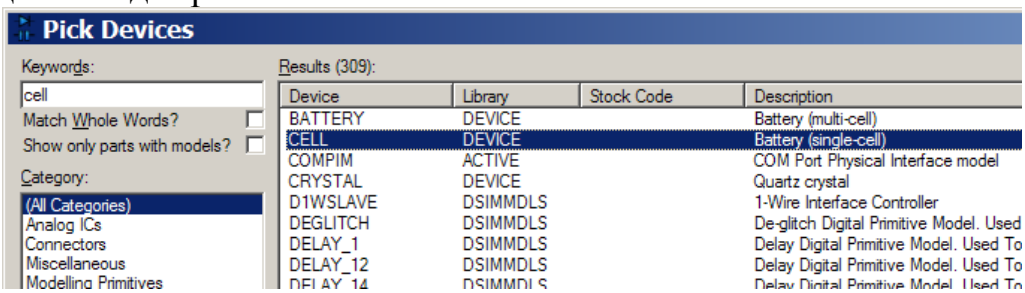
- джерело змінної напруги ALTERNATOR:



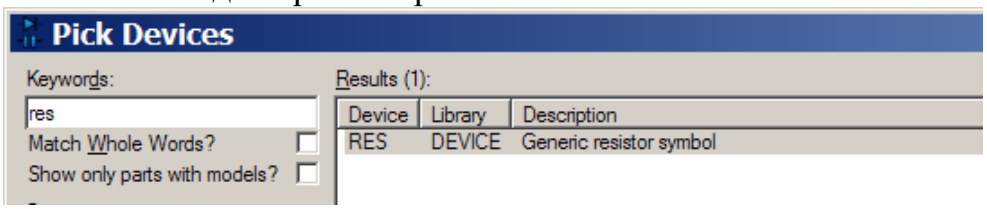
- загальна модель конденсатора



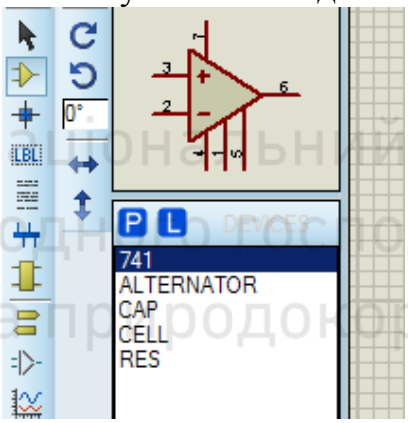
- ідеальне джерело EPC:



- загальна модель резистора



Оперативна панель матиме наступний вигляд:



2.3. Типові схеми ввімкнення операційного підсилювача

2.3.1. Дослідження диференційного ввімкнення операційного підсилювача

Принципова схема диференційного ввімкнення операційного підсилювача наведена на рис. 4.1.

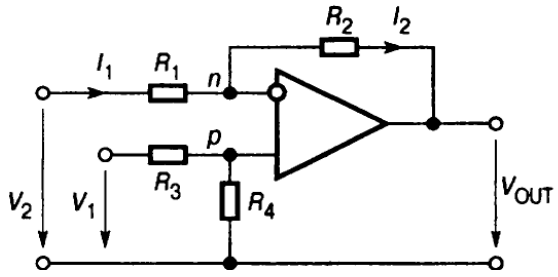


Рисунок 4.1 – Схема диференційного ввімкнення операційного підсилювача

Залежність вихідної напруги U_{out} від вхідних напруг U_1 і U_2 встановлюється співвідношенням:

$$U_{out} = \frac{(R_1 + R_2)R_4}{(R_3 + R_4)R_1} U_1 - \frac{R_2}{R_1} U_2. \qquad (4.1)$$

При виконанні умови $R_1R_4 = R_2R_3$:

$$U_{out} = \left(U_1 - U_2\right)\frac{R_2}{R_1}. \qquad (4.2)$$

Схема диференційного ввімкнення операційного підсилювача в симуляторі Proteus наведена на рис. 4.2. На схемі розглядається функціонування операційного підсилювача 741 за наступних умов:

U_1 = 10 В; U_2 = 3,2 В; $R_1 = R_2 = R_3 = R_4 =$ 10 кОм.

За таких умов вихідна напруга схеми, відповідно до виразу (4.2), дорівнює:

$$U_{out} = \left(0 - 3{,}2\right)\frac{10000}{10000} = 6{,}8 \text{ В}.$$

Отримане значення вихідної напруги показує вольтметр на виході схеми, рис. 4.2.

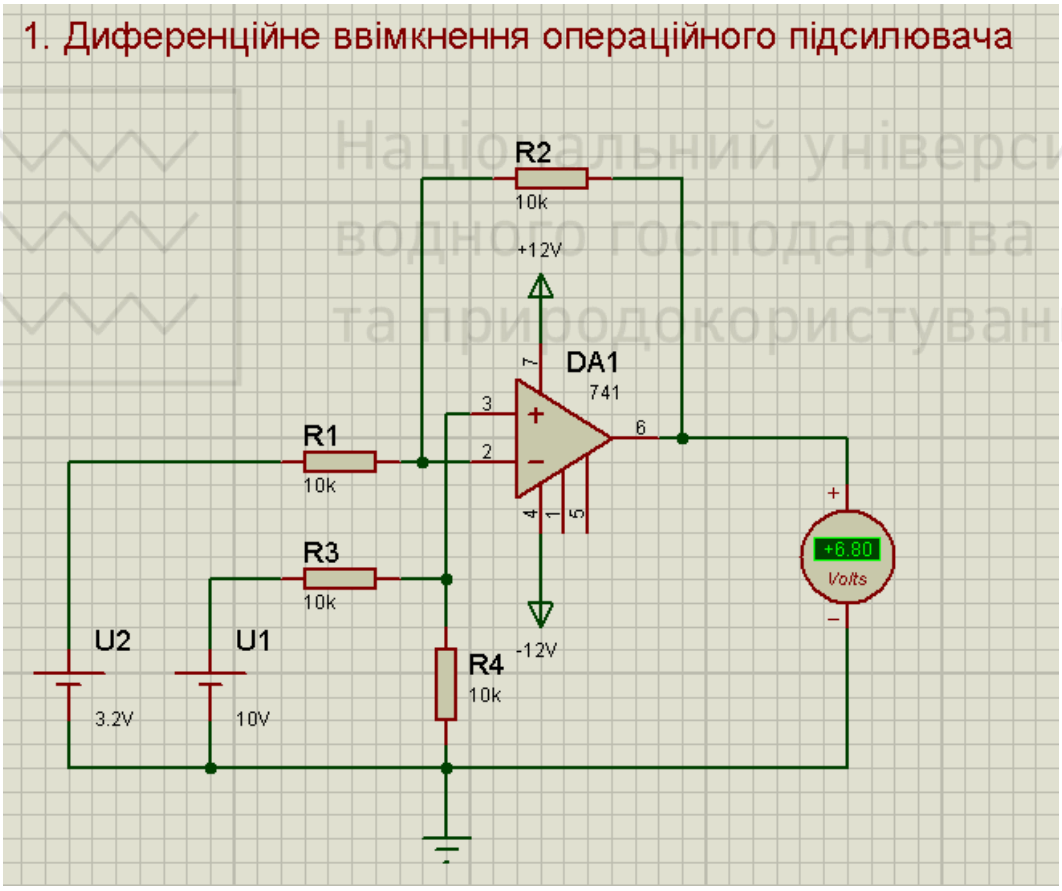


Рисунок 4.2 - Схема диференційного ввімкнення операційного підсилювача в симуляторі Proteus

2.3.2. Дослідження інвертуючого ввімкнення операційного підсилювача

Принципова схема інвертуючого ввімкнення операційного підсилювача наведена на рис. 4.3.

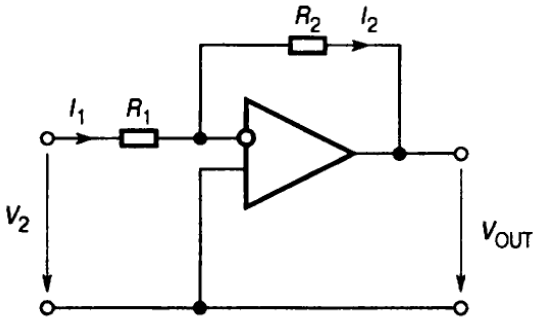


Рисунок 4.3 – Схема інвертуючого ввімкнення операційного підсилювача

Коефіцієнт підсилення такої схеми за напругою:

$$K = \frac{U_{out}}{U_2} = -\frac{R_2}{R_1}. \qquad (4.3)$$

Вихідна напруга підсилювача при інвертуючому ввімкненні знаходиться в протифазі до вхідної напруги. Залежно від опорів резисторів, коефіцієнт підсилення може бути більшим або меншим за одиницю.

Схема інвертуючого ввімкнення операційного підсилювача в симуляторі Proteus наведена на рис. 4.4. На схемі розглядається функціонування операційного підсилювача 741 за наступних умов:

U_2 = 3,2 В; R_1 = 10 кОм; R_2 = 4,7 кОм.

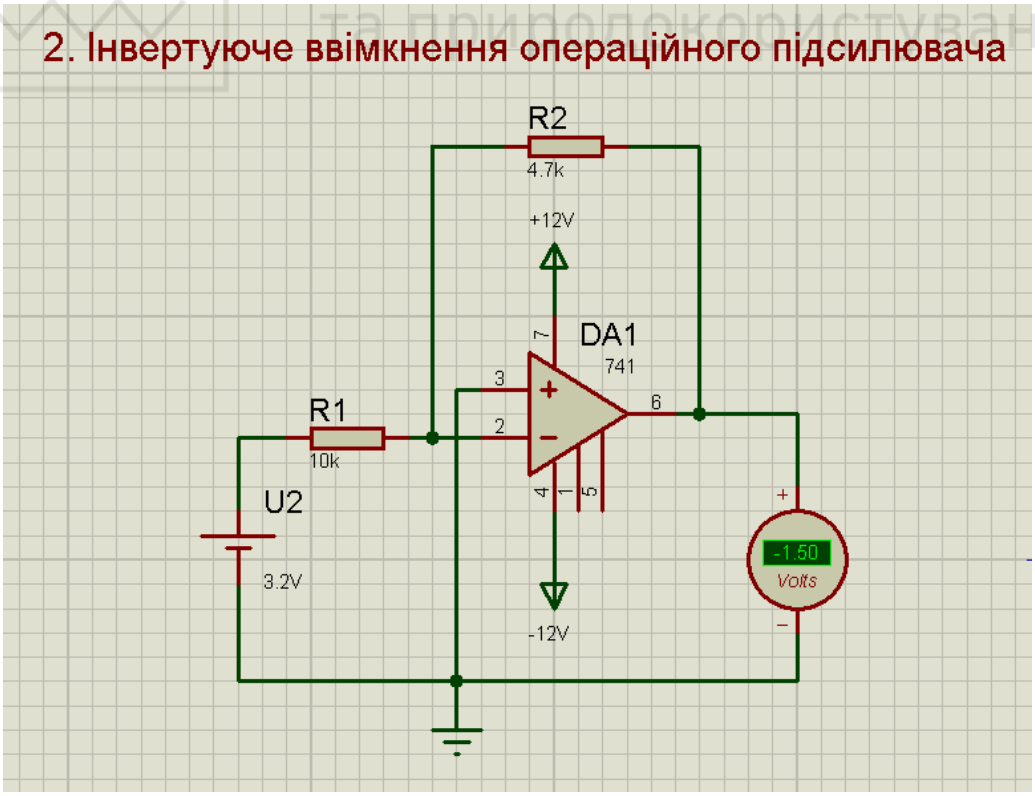


Рисунок 4.4 - Схема інвертуючого ввімкнення операційного підсилювача в симуляторі Proteus

За таких умов вихідна напруга схеми, відповідно до виразу (4.3), дорівнює:

$$U_{out} = -3{,}2 \cdot \frac{4700}{10000} = -1{,}5 \text{ В}.$$

Отримане значення вихідної напруги показує вольтметр на виході схеми, рис. 4.4.

2.3.3. Дослідження неінвертуючого ввімкнення операційного підсилювача

Принципова схема неінвертуючого ввімкнення операційного підсилювача наведена на рис. 4.5.

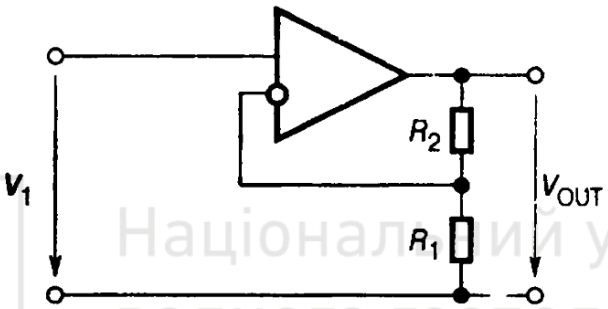


Рисунок 4.5 – Схема неінвертуючого ввімкнення операційного підсилювача

Вихідна напруга підсилювача при неінвертуючому ввімкненні знаходиться в фазі з вхідною напругою. Коефіцієнт підсилення може бути $K \geq 1$:

$$K = \frac{U_{out}}{U_1} = 1 + \frac{R_2}{R_1}. \qquad (4.4)$$

Схема неінвертуючого ввімкнення операційного підсилювача в симуляторі Proteus наведена на рис. 4.6. На схемі розглядається функціонування операційного підсилювача 741 за наступних умов:

$u_1 = U_m \cdot \sin(2\pi f t)$ при U_m=2 В, f = 1 Гц; R_1 = 2,7 кОм; R_2 = 3,9 кОм.

За таких умов коефіцієнт підсилення такої схеми за напругою, відповідно до виразу (4.4), дорівнює:

$$K = \frac{U_{out}}{U_1} = 1 + \frac{3900}{2700} = 2{,}44.$$

Тоді миттєві значення вихідної напруги становлять:

$$u_{out} = K \cdot u_1 = 2{,}44 \cdot 2 \cdot \sin(2\pi \cdot 1 \cdot t) = 4{,}88 \cdot \sin(2\pi t) \text{ В}. \qquad (4.5)$$

Як видно з отриманих графіків, миттєві значення вихідної напруги схеми відповідають отриманому аналітично виразу (4.5).

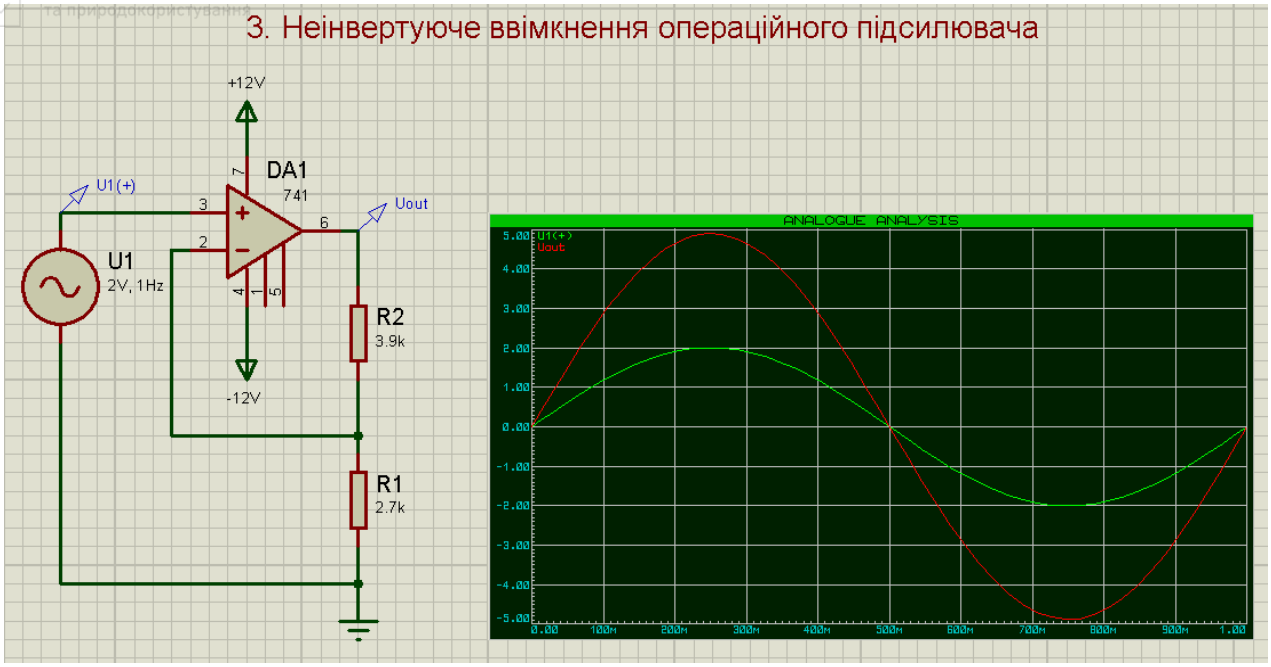


Рисунок 4.6 - Схема неінвертуючого ввімкнення операційного підсилювача в симуляторі Proteus

2.3.4. Дослідження схеми підсумовування напруг

Принципова схема підсумовування напруг з використанням операційного підсилювача наведена на рис. 4.7.

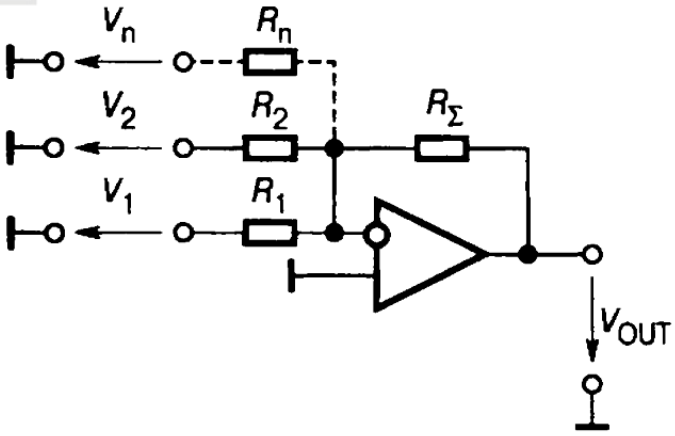


Рисунок 4.7 – Схема підсумовування напруг на операційному підсилювачі

Напруга на виході схеми визначається наступним чином:

$$U_{out} = -R_\Sigma \cdot \left(\frac{U_1}{R_1} + \frac{U_2}{R_2} + \ldots + \frac{U_n}{R_n} \right). \qquad (4.6)$$

Схема підсумовування напруг на операційному підсилювачі в симуляторі Proteus наведена на рис. 4.8. На схемі розглядається функціонування операційного підсилювача 741 за наступних умов:

U_1=1,3 В; U_2=2,2 В; U_3=1,85 В; $R_1 = R_2 = R_3 = R_\Sigma$ = 10 кОм.

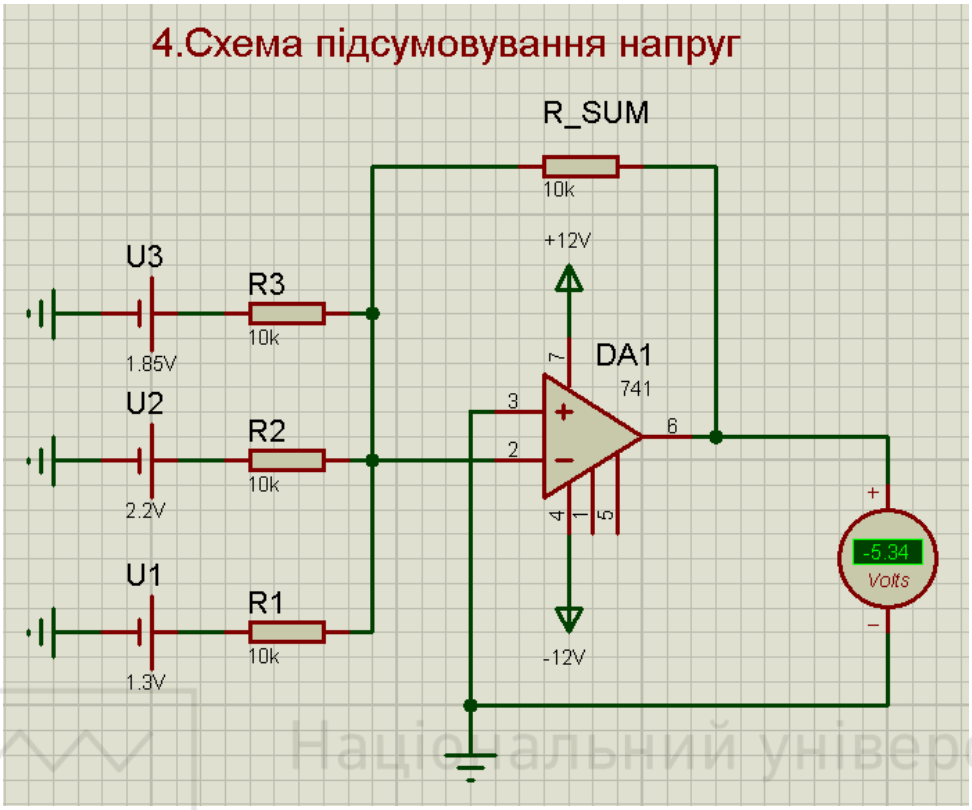


Рисунок 4.8 – Схема підсумовування напруг на операційному підсилювачі в симуляторі Proteus

Тоді вихідна напруга суматора становить:

$$U_{out} = -10000 \cdot \left(\frac{1{,}3}{10000} + \frac{2{,}2}{10000} + \frac{1{,}85}{10000} \right) = -5{,}35 \text{ В}.$$

Розраховане значення $(-5{,}35)$ В відповідає отриманому значенню напруги на виході схеми, рис. 4.8, $(-5{,}34)$ В у межах припустимої похибки.

2.3.5. Дослідження інтегратора на операційному підсилювачі

Принципова схема інтегратора на операційному підсилювачі наведена на рис. 4.9.

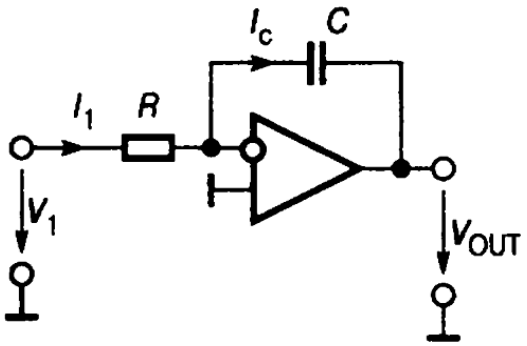


Рисунок 4.9 – Схема інтегратора на операційному підсилювачі

Миттєві значення вихідної напруги інтегратора визначаються виразом:

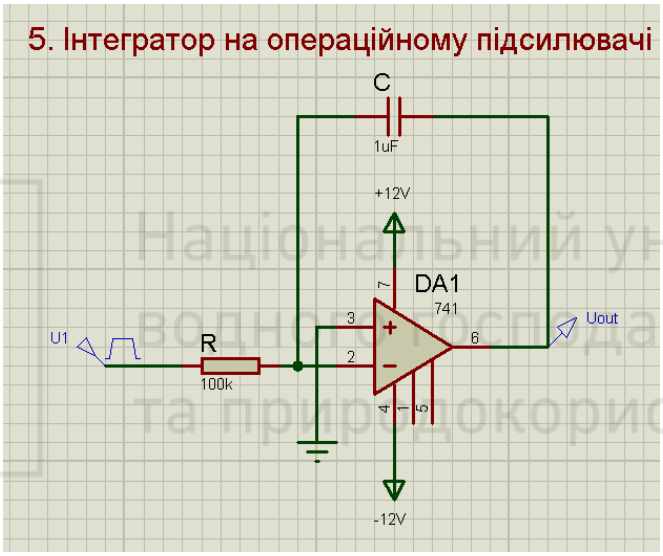
$$u_{out}(t) = u_{out}(0) - \frac{1}{RC}\int_0^t u_1(t)dt, \qquad (4.7)$$

де $u_{out}(0)$ - початкове значення вихідної напруги, В.

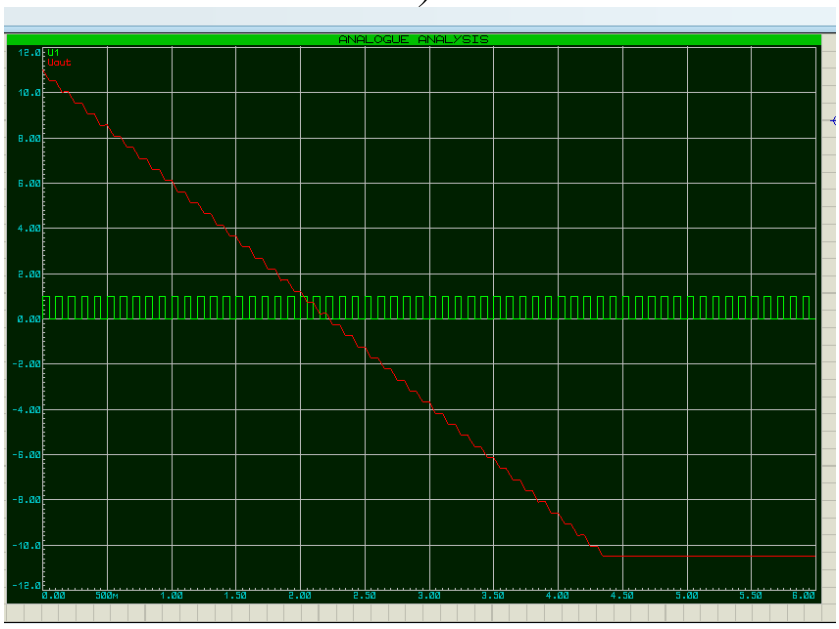
Схема інтегратора на операційному підсилювачі в симуляторі Proteus наведена на рис. 4.10, *а*. На схемі розглядається функціонування операційного підсилювача 741 при подачі на вхід схеми прямокутних імпульсів частотою 10 Гц при шпаруватості 50% і амплітуді 1 В, якщо:

R = 100 кОм; C = 1 мкФ.

Графіки миттєвих значень вхідної та вихідної напруги наведені на рис. 4.10, *б*.



а)



б)

Рисунок 4.10 – Схема інтегратора на операційному підсилювачі в симуляторі Proteus (*а*) та графіки вхідної та вихідної напруг (*б*)

2.3.6. Дослідження диференціатора на операційному підсилювачі

Принципова схема диференціатора на операційному підсилювачі наведена на рис. 4.11.

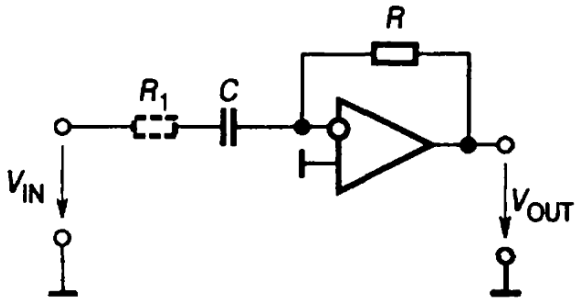
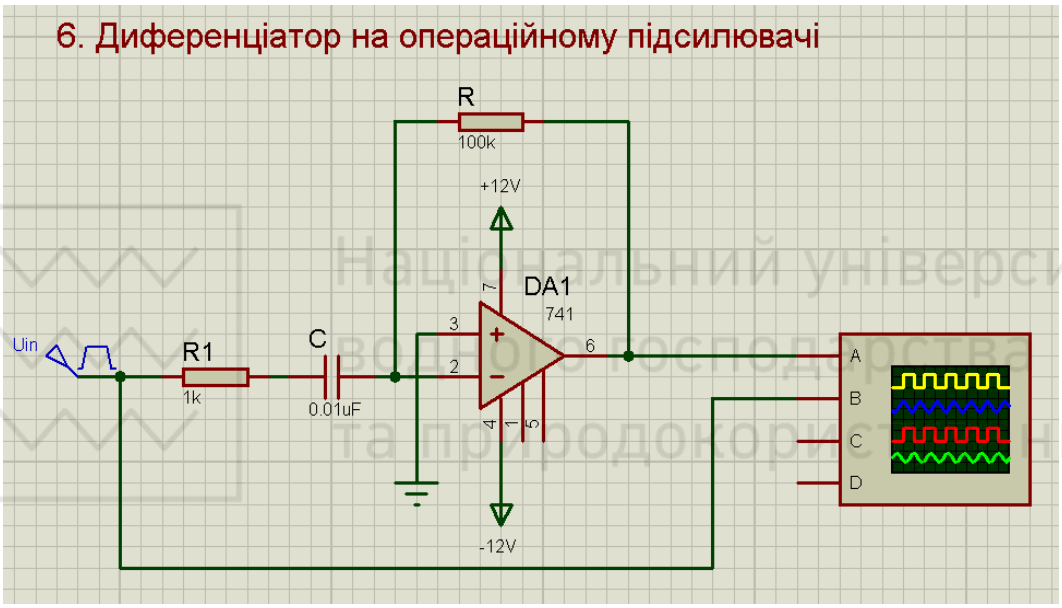
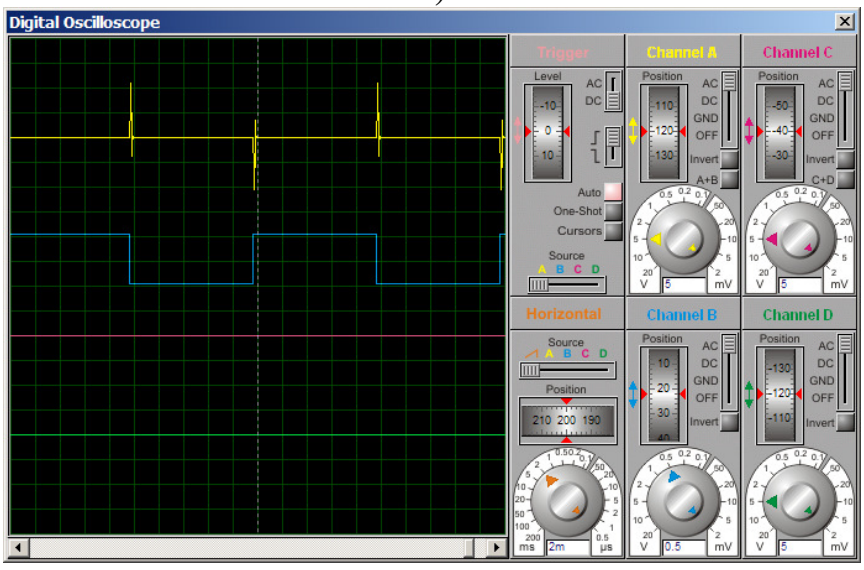


Рисунок 4.11 – Схема диференціатора на операційному підсилювачі



а)



б)

Рисунок 4.12 – Схема диференціатора на операційному підсилювачі в симуляторі Proteus (*а*) та графіки вхідної та вихідної напруг (*б*)

Миттєві значення вихідної напруги диференціатора визначаються виразом:

$$u_{out}(t) = -RC\frac{du_{in}(t)}{dt}. \quad (4.8)$$

Схема диференціатора на операційному підсилювачі в симуляторі Proteus наведена на рис. 4.12, *а*. На схемі розглядається функціонування операційного підсилювача 741 при подачі на вхід схеми прямокутних імпульсів частотою 50 Гц при шпаруватості 50% і амплітуді 1 В, якщо:

R_1 =1 кОм; R = 100 кОм; C = 0,01 мкФ.

Графіки миттєвих значень вхідної та вихідної напруги з осцилографа наведені на рис. 4.12, *б*.

ВМІСТ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

1. Тема, мета роботи.
2. Схема диференційного ввімкнення операційного підсилювача в симуляторі Proteus та результати дослідження функціонування схеми.
3. Схема інвертуючого ввімкнення операційного підсилювача в симуляторі Proteus та результати дослідження функціонування схеми.
4. Схема неінвертуючого ввімкнення операційного підсилювача в симуляторі Proteus та результати дослідження функціонування схеми.
5. Схема підсумовування напруг на операційному підсилювачі в симуляторі Proteus та результати дослідження функціонування схеми.
6. Схема інтегратора на операційному підсилювачі в симуляторі Proteus та результати дослідження функціонування схеми.
7. Схема диференціатора на операційному підсилювачі в симуляторі Proteus та результати дослідження функціонування схеми.
8. Висновки з аналізом отриманих результатів.

КОНТРОЛЬНІ ПИТАННЯ

1. Назвіть основні схеми ввімкнення операційного підсилювача.
2. Накресліть принципову схема диференційного ввімкнення операційного підсилювача. Пояснити призначення елементів.
3. Від чого залежить вихідна напруга операційного підсилювача при диференційній схемі ввімкнення?
4. Накресліть принципову схема інвертуючого ввімкнення операційного підсилювача. Пояснити призначення елементів.
5. Від чого залежить вихідна напруга операційного підсилювача при інвертуючій схемі ввімкнення?
6. Накресліть принципову схема неінвертуючого ввімкнення операційного підсилювача. Пояснити призначення елементів.
7. Від чого залежить вихідна напруга операційного підсилювача при неінвертуючій схемі ввімкнення?

ЛАБОРАТОРНА РОБОТА №5
Переведення чисел між системами числення. Логічні операції

Мета: оволодіти навичками переведення чесел між двійковою, десятковою, шістнадцятковою системами числення, а також навчитися виконувати побітові логічні операції.

1. ПОРЯДОК ВИКОНАННЯ РОБОТИ

1.1. Обрати згідно варіанта вихідні. Для цього необхідно скористатися табл. 5.1, а саме:

- зі стовпчика десяткових чисел dex – три числа (d_1, d_2, d_3);
- зі стовпчика шістнадцяткових чисел hex – три числа (h_1, h_2, h_3);
- зі стовпчика двійкових чисел bin – три числа (b_1, b_2, b_3).

1.2. Виконати перетворення:

а) перевести десяткові числа d_1, d_2, d_3 в двійкову та шістнадцяткову системи числення;

б) перевести шістнадцяткові числа h_1, h_2, h_3 в десяткову та двійкову системи числення;

в) перевести двійкові числа b_1, b_2, b_3 в десяткову та шістнадцяткову системи числення.

Результати занести до табл. 5.2, де замість змінних вписати числа відповідно до варіанта, а в порожні клітинки вписати результати розрахунку.

Таблиця 5.2

Результати розрахунків

Перетворення	dex	hex	bin
а) десяткові числа в двійкову та шістнадцяткову системи числення	d_1		
	d_2		
	d_3		
б) шістнадцяткові числа в десяткову та двійкову системи числення		h_1	
		h_2	
		h_3	
в) двійкові числа в десяткову та шістнадцяткову системи числення			b_1
			b_2
			b_3

Таблиця 5.1

Завдання для виконання

№ вар.	dec			hex			bin		
	d_1	d_2	d_3	h_1	h_2	h_3	b_1	b_2	b_3
-1-	*-2-*	*-3-*	*-4-*	*-5-*	*-6-*	*-7-*	*-8-*	*-9-*	*-10-*
1	12	88	99	2Ch	68h	1Fh	10100101b	01100110b	01110011b
2	18	35	48	3Fh	1Dh	0F8h	10010011b	10001101b	10010011b
3	19	36	125	0F1h	0DDh	0D7h	01110011b	11010101b	10011110b
4	25	39	198	22h	0F1h	0E7h	11101101b	10101010b	10110110b
5	28	38	212	59h	0CCh	89h	10011101b	11110101b	10110111b
6	25	78	126	0CDh	0A1h	12h	10010011b	10101110b	11111011b
7	124	88	198	0DCh	0C2h	45h	10011110b	11110110b	00010000b
8	189	93	75	25h	5Dh	0D5h	10110110b	11110110b	10101111b
9	158	45	23	12h	0E4h	4Dh	10110111b	00010000b	11100011b
10	212	65	14	0FFh	0E6h	7Ah	11111011b	10101111b	11000011b
11	15	63	18	0DCh	9Dh	0A8h	00111001b	11100011b	01010110b
12	25	58	19	0ABh	1Bh	83h	10111011b	11000011b	00111001b
13	89	59	58	0DFh	0B1h	0D8h	10111101b	01010110b	01101001b
14	36	45	69	1Dh	0BBh	12h	11110001b	00111001b	11101010b
15	25	58	98	0D5h	0B4h	18h	00111001b	10101000b	10010101b
16	78	41	42	3Fh	7Ch	0E1h	01101001b	00110011b	01110110b
17	98	39	18	28h	99h	5Dh	11101010b	00001111b	01101011b
18	125	12	19	12h	15h	0D8h	10010101b	01110110b	00110000b
19	156	122	15	0ABh	18h	0F5h	10101010b	01101011b	10101000b
20	98	14	79	0DCh	0D4h	32h	11001010b	00110000b	00110011b

5.3. Виконати побітові логічні операції

3.1) $b_1 + b_2$	3.2) $b_2\, b_3$	3.3) $\bar{b}_1$	3.4) $\overline{b_1 + b_3}$
3.5) $\overline{b_1 b_2 + b_3}$	3.6) $\overline{b_1 b_2 b_3} + b_2$	3.7) $\bar{b}_1 + \bar{b}_3$	3.8) $\bar{h}_1 + \bar{h}_3$
3.9) $b_1 \oplus b_2$	3.10) $h_1\, b_3$	3.11) $h_2\, b_3$	3.12) $d_1 + h_3$

ВМІСТ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

1. Тема, мета роботи.
2. Вихідні дані.
3. Розрахунки для переведення заданих чисел між системами числення.
4. Розрахунки для викоання заданих побітових логічних операцій.
5. Висновки.

КОНТРОЛЬНІ ПИТАННЯ

1. Як перевести число з двійкової до десяткової системи числення?
2. Як перевести число з десяткової до двійкової системи числення?
3. Переведіть у десяткову форму наступні числа: 0A2h, 5Dh, 0F2h, 01011111b, 01110011b.
4. Запишіть десяткове число 128 у зважено-позиційній формі.
5. Запишіть двійкове число 10101100 у зважено-позиційній формі.
6. Як зображують логічний елемент AND ?
7. Як зображують логічний елемент OR ?
8. Наведіть таблицю істиності логічного елементу АБО-НІ.

ЛАБОРАТОРНА РОБОТА №6
Формування логічних схем

Мета: освоїти методи складання логічних схем за рівняннями алгебри логіки.

1. ПОРЯДОК ВИКОНАННЯ РОБОТИ

1.1. Обрати вихідні дані.

Обрати логічну функцію відповідно до варіанта згідно табл. 6.1.

Таблиця 6.1

Вихідні дані

№ вар	Логічна функція	№ вар	Логічна функція
1	$y = x_1 \cdot x_2 + \overline{x_3 + x_4}$	11	$y = \overline{x_1 + x_2} \cdot x_3 + x_2$
2	$y = \overline{\overline{x_1 + x_2} + \overline{x_3 + x_4}} \cdot x_5$	12	$y = \overline{x_1 \cdot x_3} + x_2 \cdot x_1$
3	$y = \overline{\overline{x_1 \cdot x_2 \cdot x_3} + x_4}$	13	$y = \overline{\overline{x_1 + x_2} + x_3 + x_1}$
4	$y = x_1 + x_2 \cdot x_3 + \overline{x_1 + x_2}$	14	$y = x_1 \cdot x_2 \cdot \overline{x_2 + x_1 \cdot x_3}$
5	$y = \overline{x_1 + x_2} \cdot \overline{x_2 \cdot x_3}$	15	$y = \overline{x_1 x_2 x_3} + \overline{x_1 x_4}$
6	$y = \overline{x_1 + x_2 + x_3} \cdot \overline{x_1 \cdot x_2 + x_3}$	16	$y = \overline{x_1 x_2} + x_2 + x_3$
7	$y = x_1 + x_2 \cdot \overline{x_1 + x_3}$	17	$y = x_1 \cdot x_2 + x_3 + \overline{x_1 \cdot x_2}$
8	$y = \overline{x_1 \cdot x_2 + x_3} \cdot x_2$	18	$y = x_1 + x_2 \cdot \overline{x_4 + x_1 \cdot x_3}$
9	$y = \overline{x_1 \cdot x_2 + x_3} + \overline{x_1 \cdot x_3}$	19	$y = \overline{x_1 + x_2 \cdot x_3} + \overline{x_1 + x_3}$
10	$y = x_1 \cdot x_2 + \overline{x_3 + \overline{x_2 \cdot x_3} + x_1 \cdot x_2}$	20	$y = x_1 + x_2 + \overline{x_3 \cdot \overline{x_2 + x_3} + x_1 \cdot x_2}$

1.2. Скласти та реалізувати у Proteus логічну схему, яка реалізує обрану функцію.

Для цього необхідно запустити програму Proteus. Створити новий проект (File -> New Project), встановити властивості проекту, як показано на рис. 6.1.

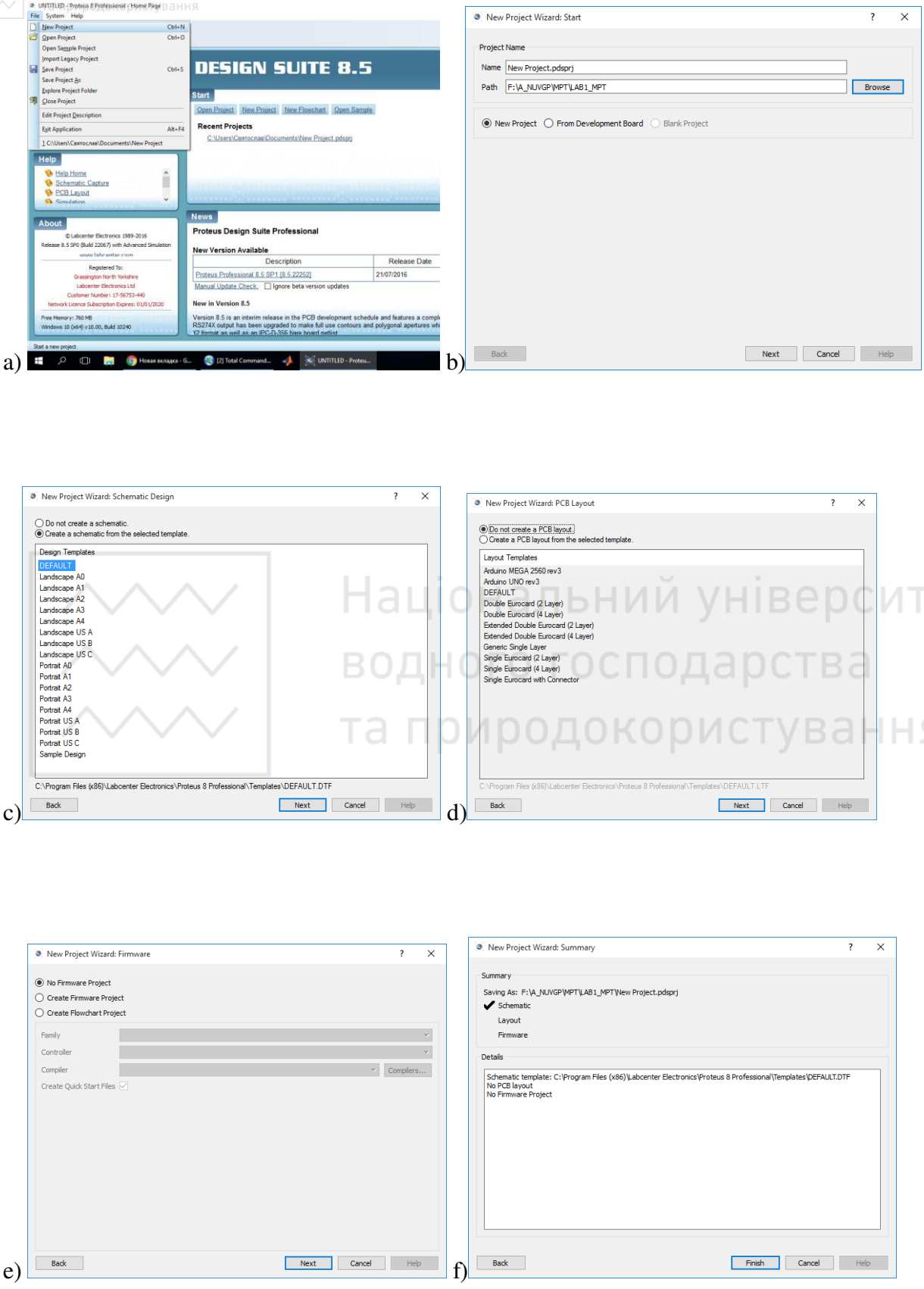
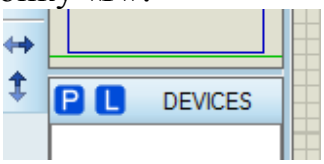


Рисунок 6.1 – Створення нового проекту у Proteus

Після створення нового проекту з’явиться вікно нового документа Schematic. Для вибору необхідних компонентів необхідно відкрити бібліотеки, для чого необхідно натиснути кнопку «P»:



Базові логічні елементи знаходяться в категорії Simulator Primitives. Для додавання необхідного елемента на оперативну панель необхідно на ньому 2 рази клацнути мишею, рис. 6.2.

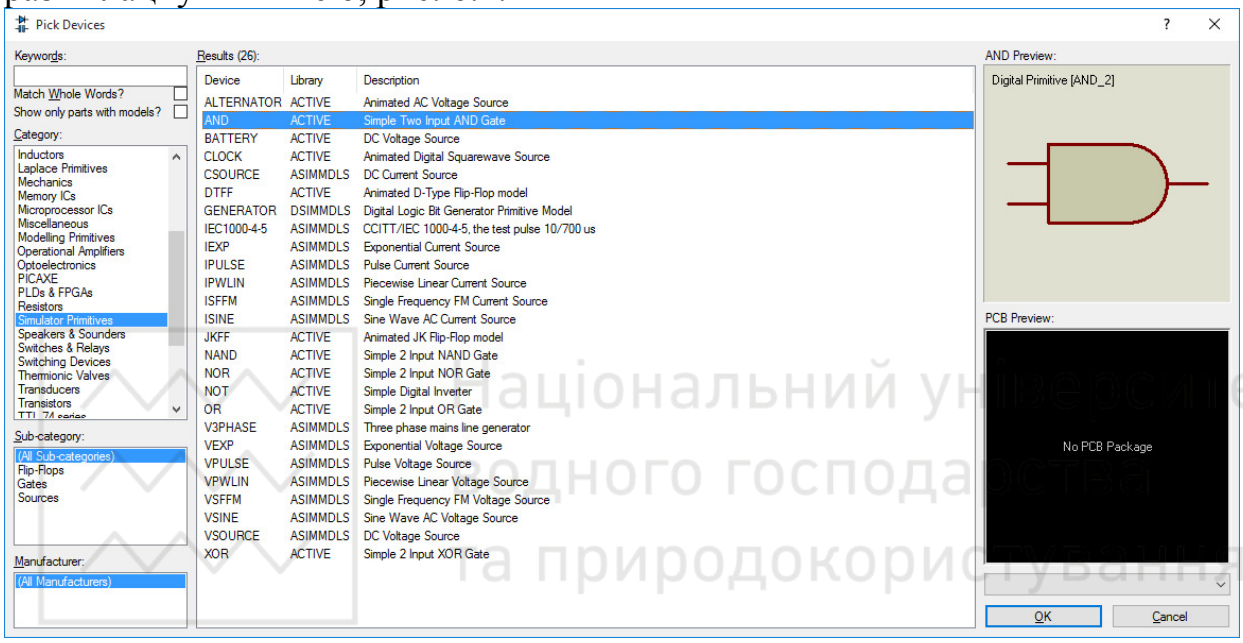


Рисунок 6.2 – Додавання логічних елементів на оперативну панель

Задавачі значень булевих змінних (LOGICTOGGLE) та індикатори значень змінних (LOGICPROBE) знаходяться в категорії Debugging Tools, рис. 6.3.

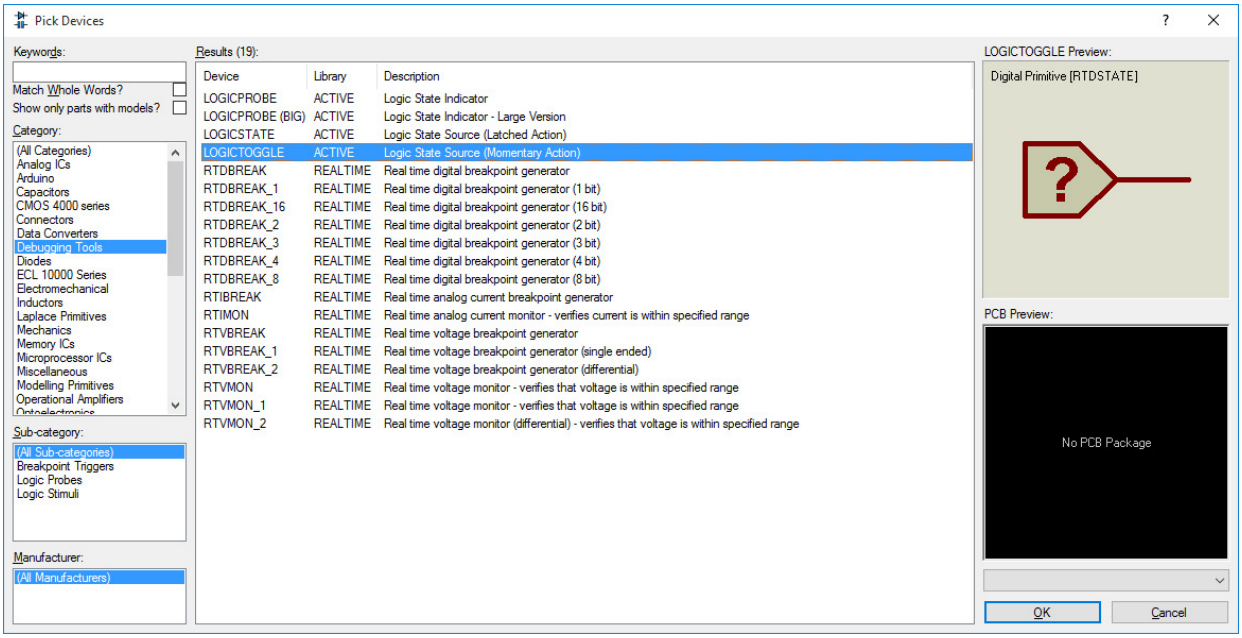


Рисунок 6.3 – Додавання елементів для роботи з булевими змінними

Обрані елементи відображаються у стовпчику Devices. Їх необхідно розставити на робочому просторі та з’єднати лініями, рис. 6.4.

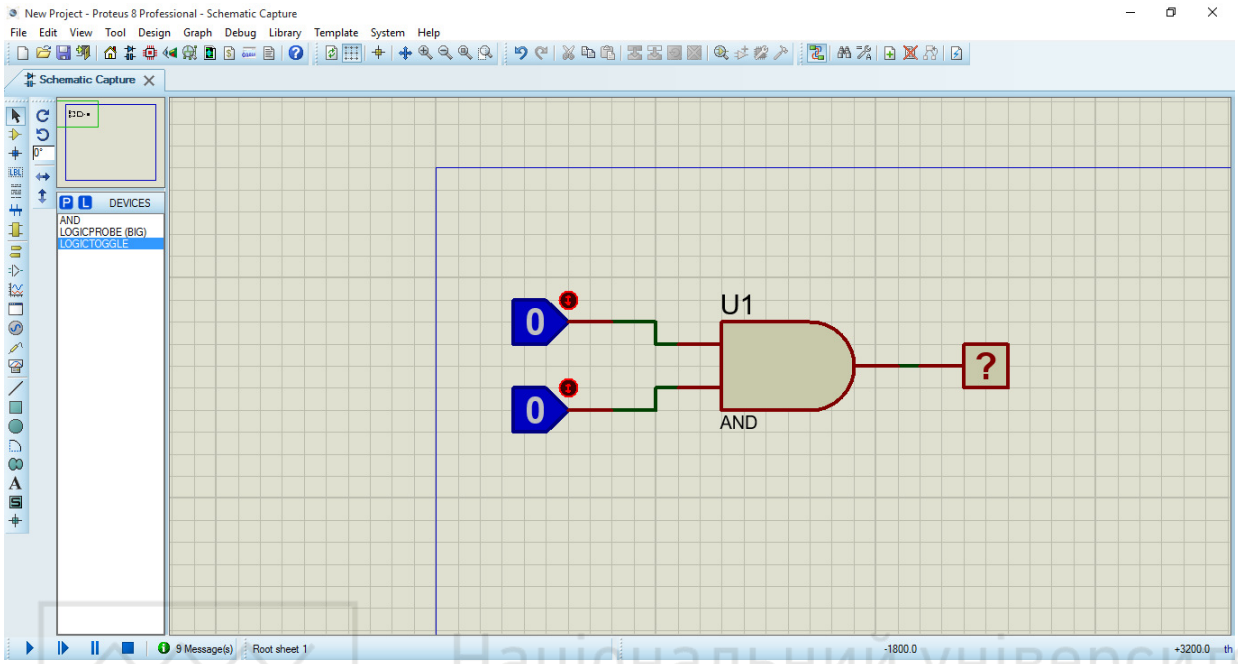


Рисунок 6.4 – Створення логічної схеми

Після цього необхідно вказати позначення вхідних величин. Для цього з контекстного меню задавачів булевих змінних LOGICTOGGLE обрати пункт «Edit Properties», рис. 6.5.

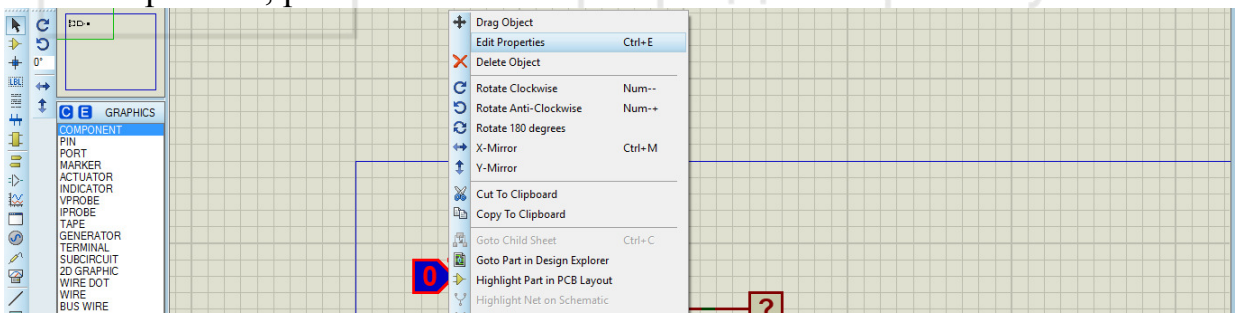
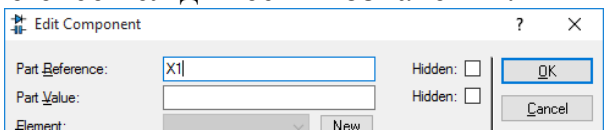


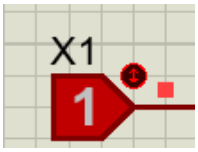
Рисунок 6.5 – Редагування властивостей задавача булевих змінних

В полі «Part Reference» слід ввести позначення:



Для запуску моделювання необхідно натиснути кнопку «Play» в лівому нижньому кутку екрана:

Для зміни значення вхідної змінної слід натиснути на двонаправлену стрілку в кружку:



При цьому червоний колір відповідає лог. 1, синій – лог. 0, рис. 6.6.

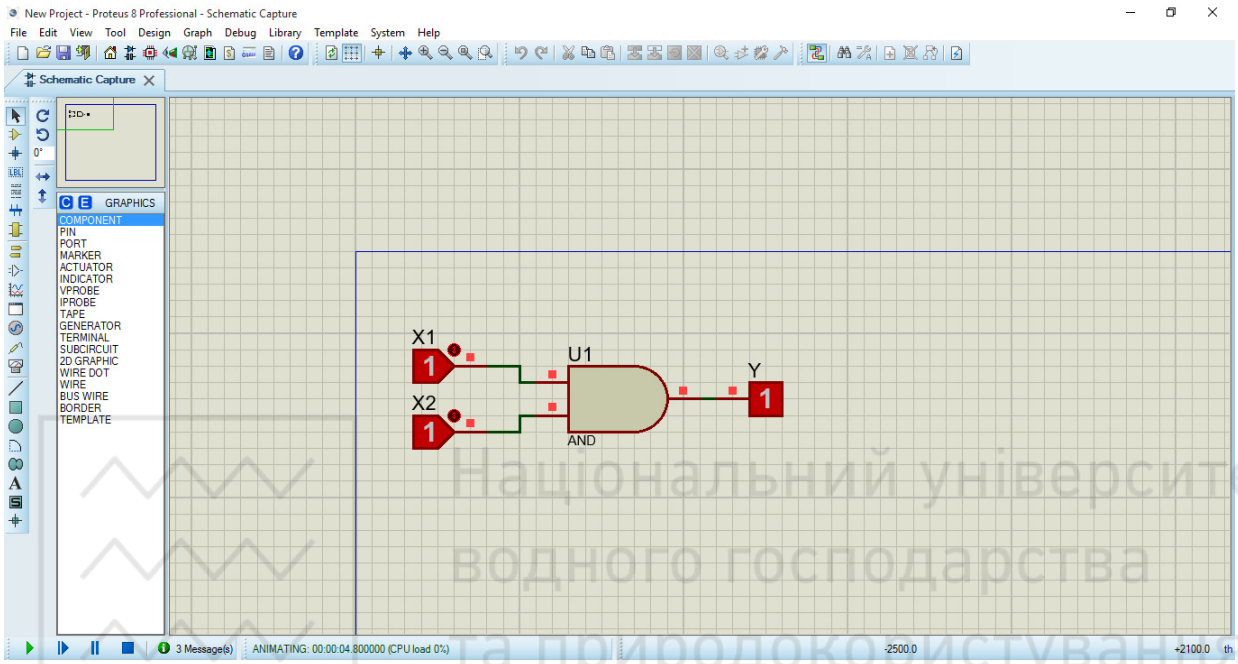


Рисунок 6.6 – Функціонування логічної схеми у Proteus

Нижче, в якості прикладу, наведена логічна схема, що реалізує функцію:

$$y = \overline{x_1 \cdot x_2} + \overline{x_3 \cdot x_4} \qquad (6.1)$$

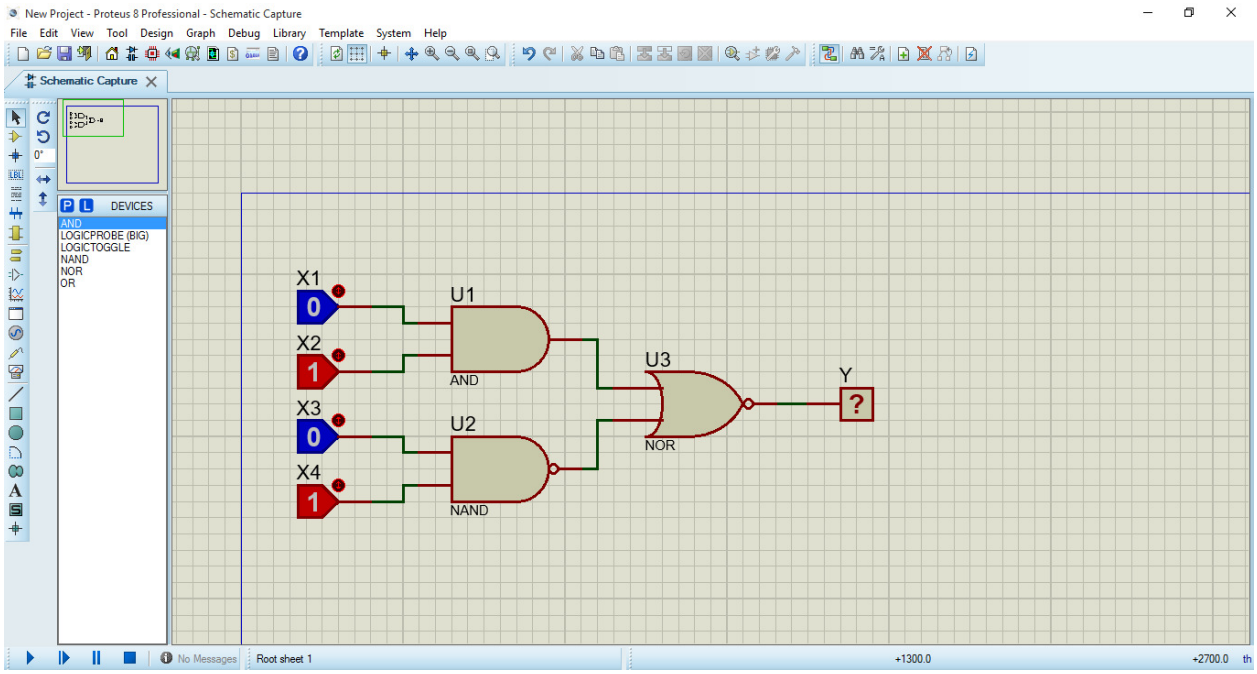


Рисунок 6.1 – Логічна схема, що реалізує функцію (6.1)

1.3. На основі даних моделювання роботи схеми у Proteus скласти таблицю істинності заданої функції.

Наприклад, для попередньої схеми таблиця істинності має вигляд, як показано у табл. 6.1.

Таблиця 6.1

Таблиця істиності логічної функції (6.1)

стан №	x_4	x_3	x_2	x_1	y
1	0	0	0	0	0
2	0	0	0	1	0
3	0	0	1	0	0
4	0	0	1	1	0
5	0	1	0	0	0
6	0	1	0	1	0
7	0	1	1	0	0
8	0	1	1	1	0
9	1	0	0	0	0
10	1	0	0	1	0
11	1	0	1	0	0
12	1	0	1	1	0
13	1	1	0	0	1
14	1	1	0	1	1
15	1	1	1	0	1
16	1	1	1	1	0

ВМІСТ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

1. Тема, мета роботи
2. Рівняння логічної функції у відповідності до варіанту
3. Логічна схема реалізації логічної функції у Proteus.
4. Таблиця істинності логічної функції.
5. Висновки.

КОНТРОЛЬНІ ПИТАННЯ

1. Що таке «алгебра логіки»?
2. Що таке «оператор», «операнд»?
3. Які основні логічні оператори Вам відомі?
4. Наведіть позначення базових логічних елементів?
5. Яким чином позначається інверсія виходу елемента?
6. Що таке «таблиця істинності»?
7. Наведіть таблиці істинності базових логічних елементів.

ЛАБОРАТОРНА РОБОТА №7
Дослідження функціонування тригерів

Мета: дослідити функціонування тригерів найбільш поширених типів.

1. ЗАВДАННЯ

З використанням симулятора Proteus дослідити функціонування RS-тригера, JK-тригера, D-тригера, T-тригер.

2. ПОРЯДОК ВИКОНАННЯ РОБОТИ

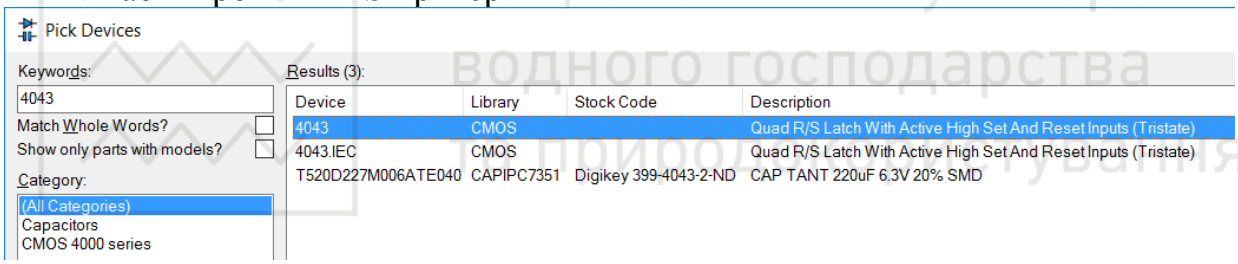
2.1. Створити новий проект в симуляторі Proteus.

Запустити програму Proteus. Створити новий проект (File -> New Project). Для кожної схеми рекомендується створювати новий проект.

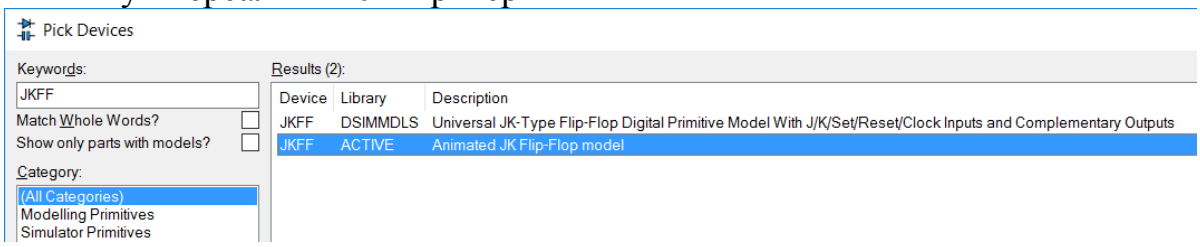
2.2. Вибрати необхідні елементи.

На оперативну панель необхідно додати наступні елементи:

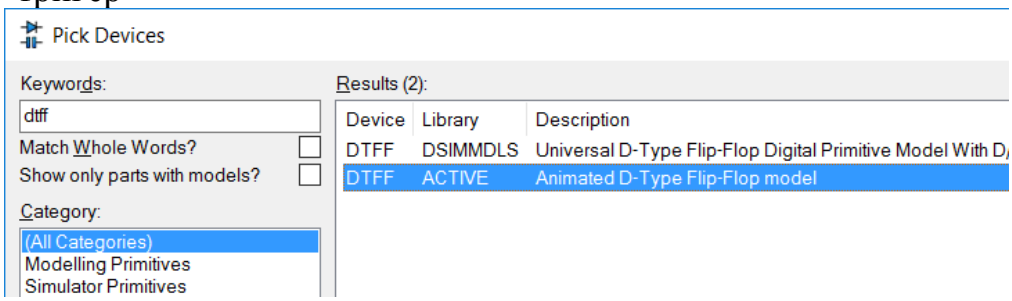
- асинхронний RS-тригер



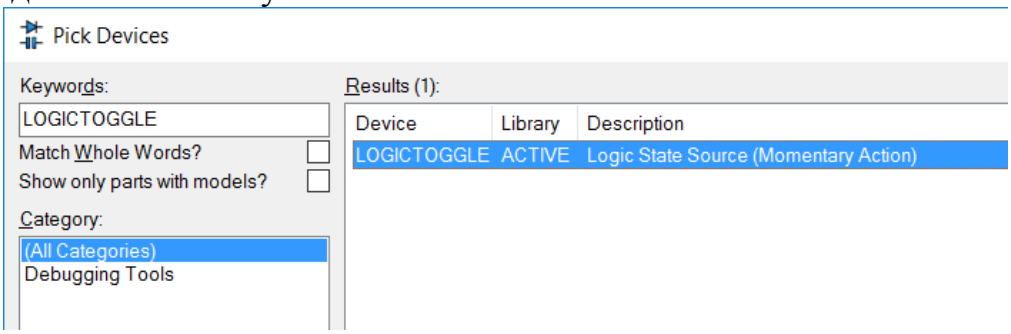
- універсальний JK-тригер



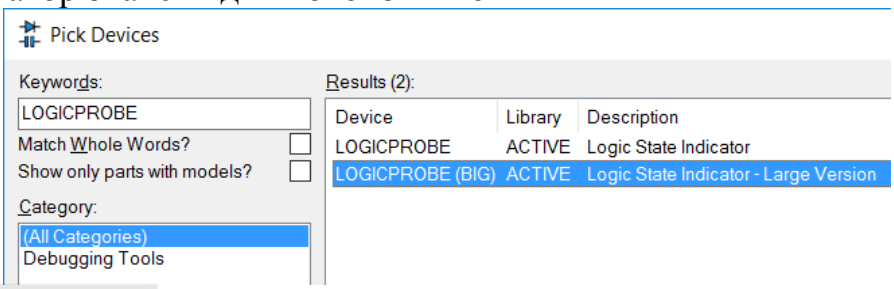
- D-тригер



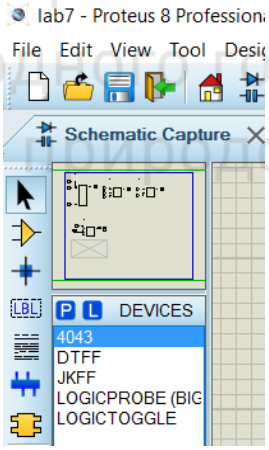
- задавач значень булевих змінних



- індикатор значень двійкової змінної



Оперативна панель матиме наступний вигляд:



2.3. Дослідження функціонування RS-тригера

Схема, що дозволяє досліджувати функціонування асинхронного RS-тригера, наведена на рис. 7.1.

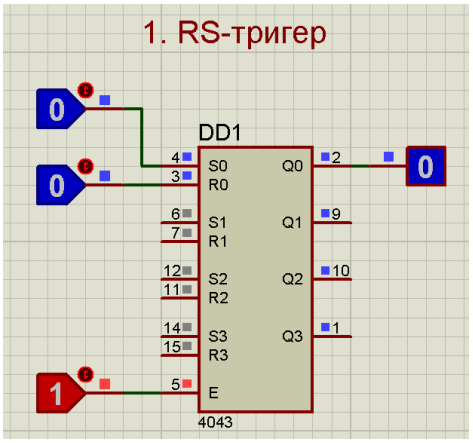


Рисунок 7.1 – Схема моделі RS-тригера

В схемі використовується мікросхема 4043, яка включає чотири однакових RS-тригери з трьома станами. Для демонстрації функціонування використовується RS-тригер №0, входи якого позначені S0, R0, вихід – Q0. До входів тригера підключені задавачі значень булевих змінних, до виходів - індикатор значень двійкової змінної. З використанням запропонованої моделі необхідно скласти таблицю станів RS-тригера (табл. 7.1).

Таблиця 7.1

Таблиця станів RS-тригера

S	R	E	Q_n	Q_{n+1}
0	0	1		
1	0	1		
0	0	1		
0	1	1		
0	0	1		
0	0	0		
1	0	0		
0	0	0		
0	1	0		
0	0	0		

2.4. Дослідження функціонування JK-тригера

Схема, що дозволяє досліджувати функціонування універсально JK-тригера, наведена на рис. 7.2. В схемі використовується модель універсального JK-тригера, яка має вхід синхронізації CLK.

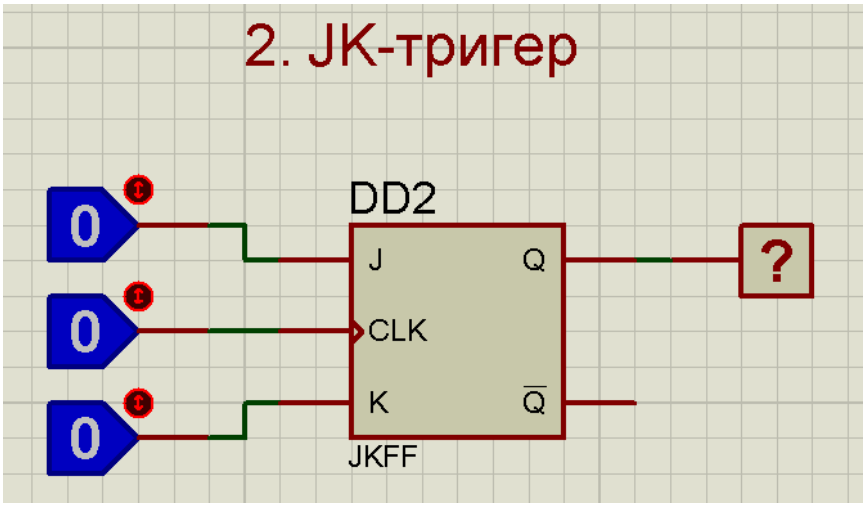


Рисунок 7.2 – Схема моделі JK-тригера

Необхідно проаналізувати роботу JK-тригера та накреслити часові діаграми його роботи в осях, як проказано на рис. 7.3.

Рисунок 7.3 – Шаблон для побудови часових діаграм роботи JK-тригера

2.5. Дослідження функціонування D-тригера

Схема, що дозволяє досліджувати функціонування D-тригера, наведена на рис. 7.4. В схемі використовується модель тригера, яка має вхід синхронізації CLK.

Необхідно проаналізувати роботу D-тригера та накреслити часові діаграми його роботи в осях, як проказано на рис. 7.5.

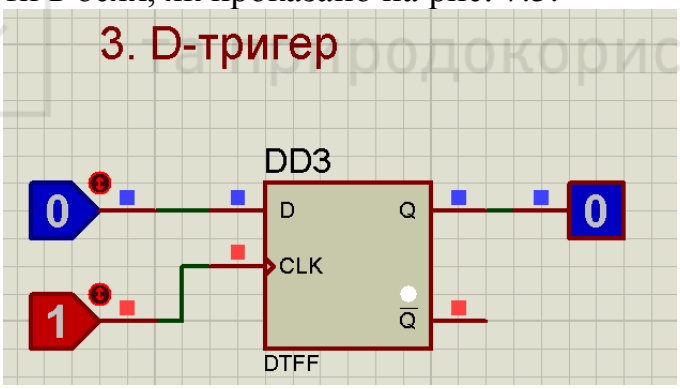


Рисунок 7.4 – Схема моделі D-тригера

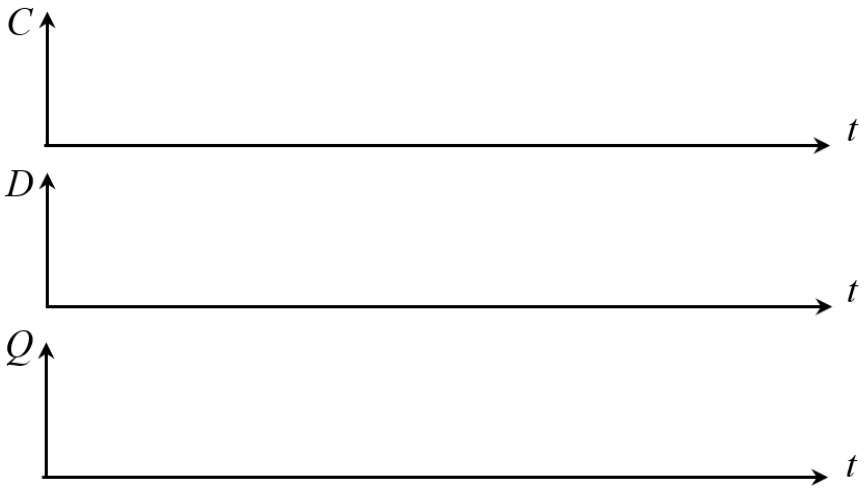


Рисунок 7.5 – Шаблон для побудови часових діаграм роботи D-тригера

2.6. Дослідження функціонування Т-тригер

Схема, що дозволяє досліджувати функціонування Т-тригера, побудована з використанням моделі JK-тригера, що має вхід синхронізації CLK (рис. 7.6). Вхід Т-тригера необхідно підключити до генератора прямокутних імпульсів (Q0), вихідна частота якого становить 1 Гц (рис. 7.7). Вхідний та вихідний сигнали зв схеми Т-тригера необхідно відобразити у вікні цифрового осцилографа DIGITAL ANALYSIS, у вікні завдання параметрів якого слід встановити час моделювання (Stop time) 10 с (рис. 7.8). Необхідно проаналізувати функціонування Т-тригера за діаграмами на цифровому осцилографі і навести їх у звіті.

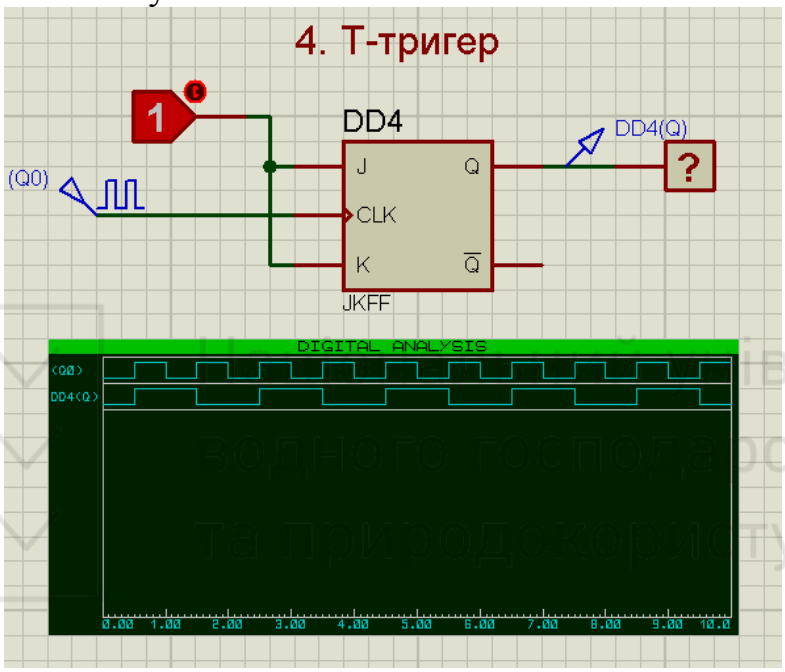


Рисунок 7.6 – Схема моделі Т-тригера

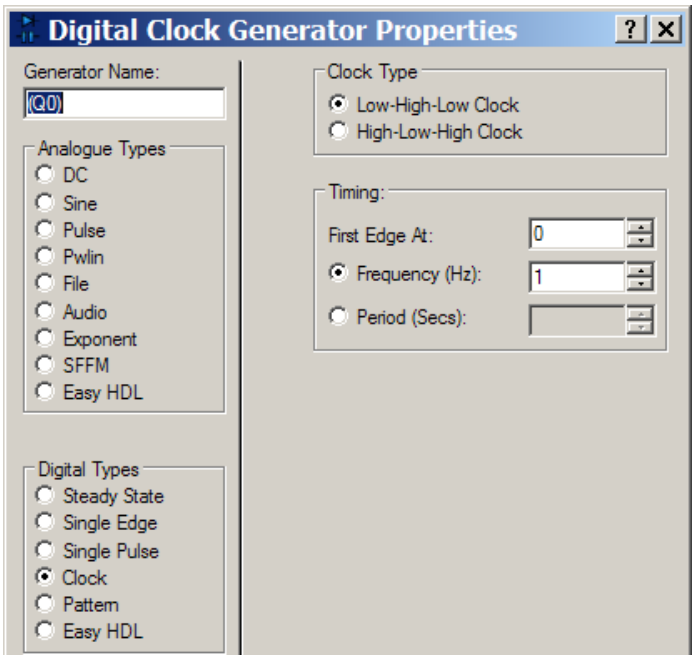


Рисунок 7.7 – Вікно налаштувань параметрів генератора прямокутних імпульсів

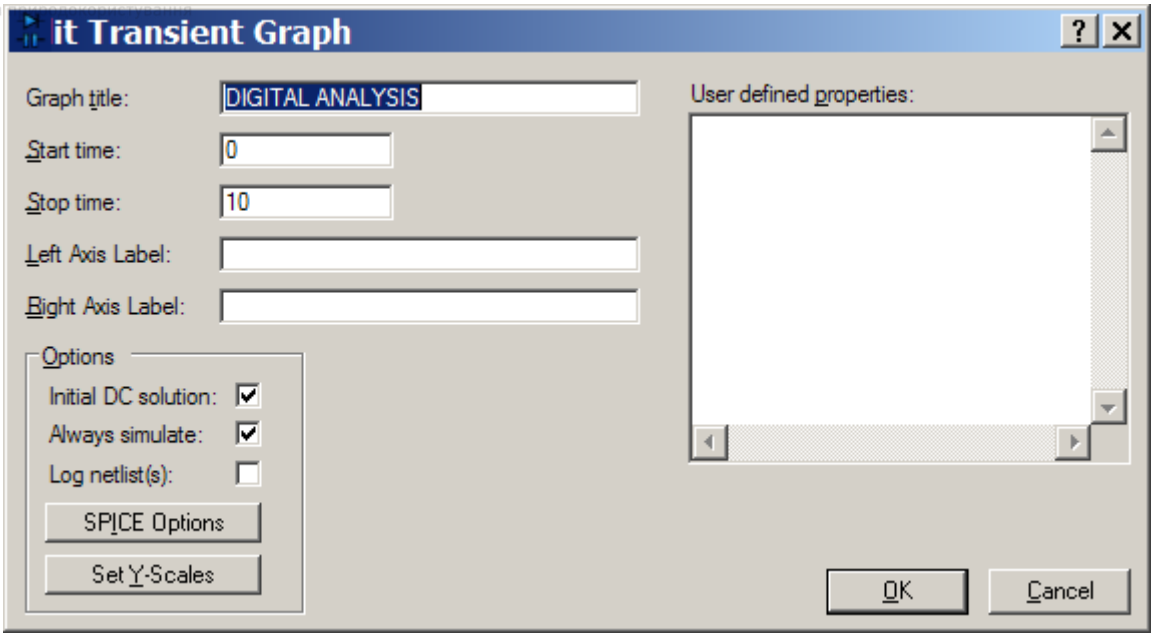


Рисунок 7.8 – Вікно налаштувань параметрів цифрового осцилографа

ВМІСТ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

1. Тема, мета роботи.
2. Схема моделі в симуляторі Proteus і таблиця станів RS-тригера.
3. Схема моделі в симуляторі Proteus та часові діаграми роботи JK-тригера.
4. Схема моделі в симуляторі Proteus та часові діаграми роботи D-тригера.
5. Схема моделі в симуляторі Proteus та вікно цифрового осцилографа з вхідними та вихідними сигналами Т-тригера.
6. Висновки з аналізом отриманих результатів.

КОНТРОЛЬНІ ПИТАННЯ

1. Який пристрій називається тригером?
2. Скільки стійких станів має тригер?
3. Які типи тригерів Вам відомі?
4. Пояснити принцип дії RS-тригера.
5. Яка комбінація вхідних сигналів є забороненою для RS-тригера.

ЛАБОРАТОРНА РОБОТА №8
Дослідження функціонування регістрів

Мета: проаналізувати функціонування послідовного та паралельного регістрів.

1. ЗАВДАННЯ

З використанням симулятора Proteus дослідити функціонування послідовного та паралельного регістрів.

2. ПОРЯДОК ВИКОНАННЯ РОБОТИ

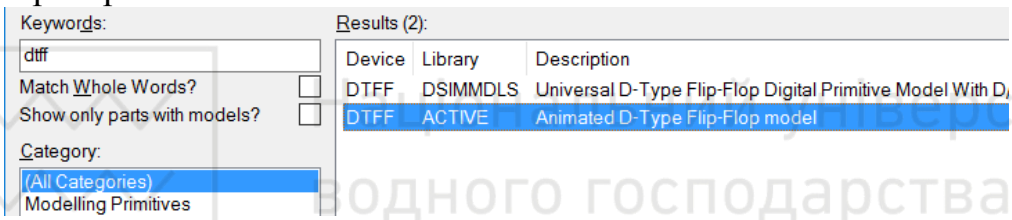
2.1. Створити новий проект в симуляторі Proteus.

Запустити програму Proteus. Створити новий проект (File -> New Project). Для кожної схеми рекомендується створювати новий проект.

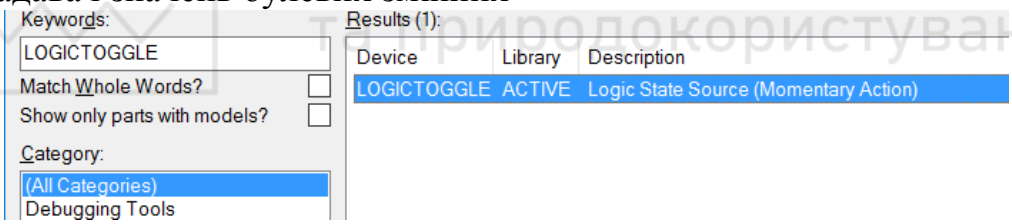
2.2. Вибрати необхідні елементи.

На оперативну панель необхідно додати наступні елементи:

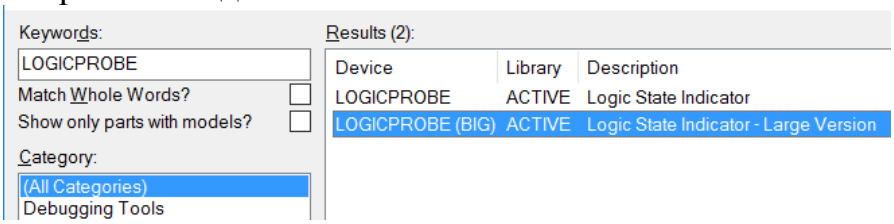
- D-тригер



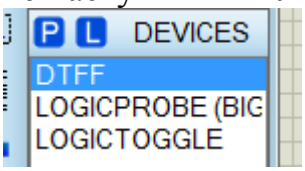
- задавач значень булевих змінних



- індикатор значень двійкової змінної



Оперативна панель матиме наступний вигляд:



2.3. Дослідження функціонування трирозрядного паралельного регістра на D-тригерах

Схема, що дозволяє досліджувати функціонування трирозрядного паралельного регістра на D-тригерах, наведена на рис. 8.1. В схемі використовуються три D-тригери, входи яких відповідають входам розрядів регістра, прямі виходи – відповідають виходам регістра. Входи синхронізації

тригерів об'єднані і відповідають входу синхронізації регістра. Синхронізація динамічного типу за переднім фронтом синхроімпульсу.

З використанням запропонованої схеми необхідно проаналізувати процес збереження трирозрядного двійкового числа до регістра.

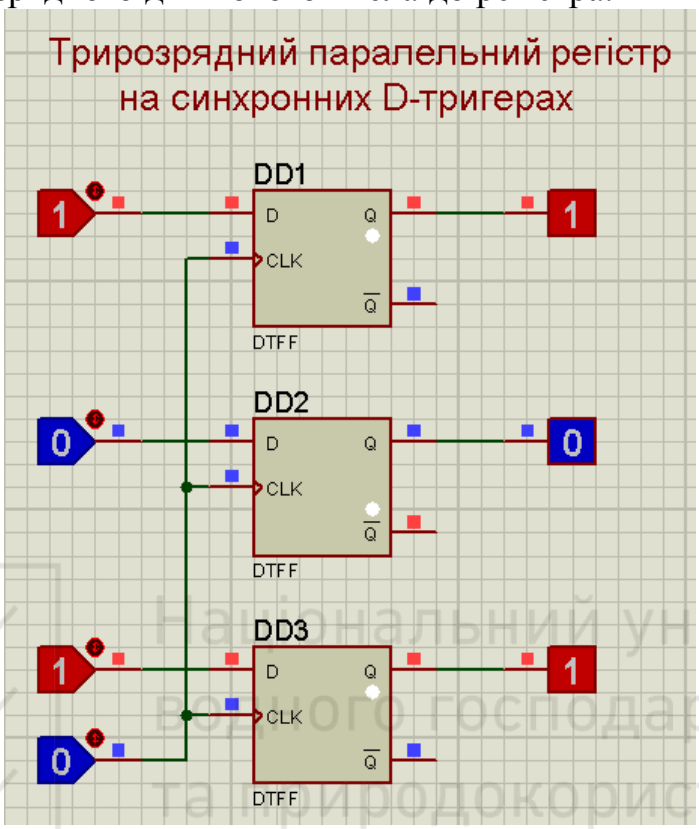


Рисунок 8.1 – Трирозрядний паралельний регістр на D-тригерах

2.4. Дослідження функціонування чотирирозрядного послідовного регістра на D-тригерах

Схема, що дозволяє досліджувати функціонування чотирирозрядного послідовного регістра на D-тригерах, наведена на рис. 8.2. Необхідно проаналізувати роботу наведеного регістра та накреслити часові діаграми його роботи в осях, як проказано на рис. 8.3.

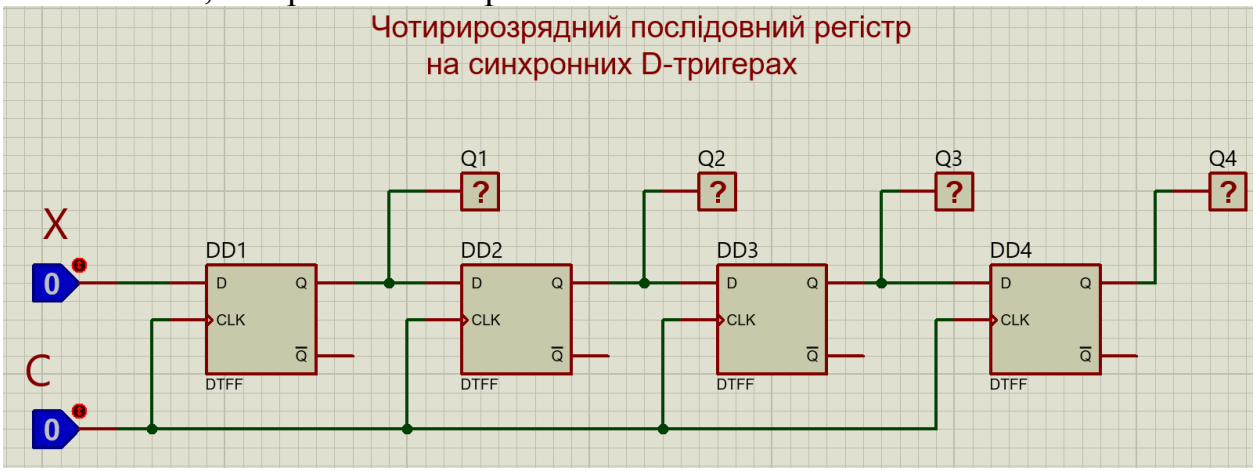


Рисунок 8.2 – Чотирирозрядний послідовний регістр на D-тригерах

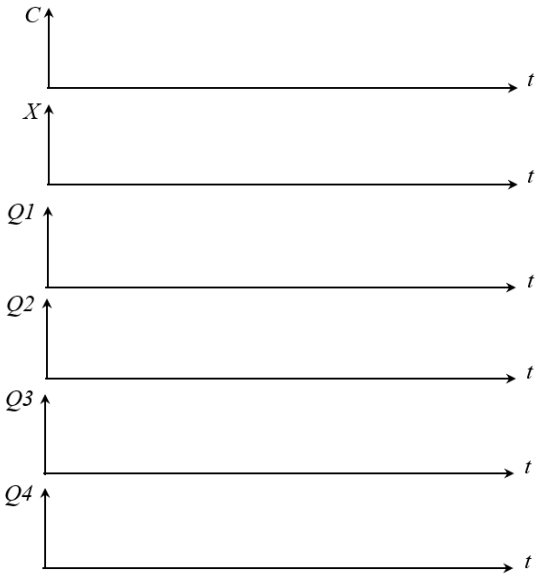


Рисунок 8.3 – Шаблон для побудови часових діаграм роботи чотирирозрядного послідовного регістра на D-тригерах

ВМІСТ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

1. Тема, мета роботи.

2. Схема трирозрядного паралельного регістра на D-тригерах в симуляторі Proteus. Описати процес запису інформації до регістра. Навести у звіті відповіді на питання:

- яким чином відбувається запис інформації до паралельного регістру?
- як записуються біти даних до паралельного регістру: послідовно чи одночасно?
- як довго записані дані зберігаються у паралельному регістрі?

3. Схема чотирирозрядного послідовного регістра на D-тригерах в симуляторі Proteus. Часові діаграми роботи регістра. Описати процес запису інформації до регістра. Навести у звіті відповіді на питання:

- яким чином відбувається запис інформації до послідовного регістру?
- як записуються біти даних до послідовного регістру: послідовно чи одночасно?
- як довго записані дані зберігаються у послідовному регістрі?

4. Висновки з аналізом отриманих результатів.

КОНТРОЛЬНІ ПИТАННЯ

1. Який пристрій називається регістром?
2. В чому відмінність між паралельним та послідовним регістрами?
3. Пояснити процес завантаження інформації до паралельного регістра.
4. Пояснити процес завантаження інформації до послідовного регістра.
5. Для чого призначений вхід CLK?

ЛАБОРАТОРНА РОБОТА №9
Дослідження функціонування аналого-цифрового перетворювача

Мета: проаналізувати процес аналого-цифрового перетворення сигналів.

1. ЗАВДАННЯ

З використанням симулятора Proteus дослідити функціонування вісьмирозрядного аналого-цифрового перетворювача.

2. ПОРЯДОК ВИКОНАННЯ РОБОТИ

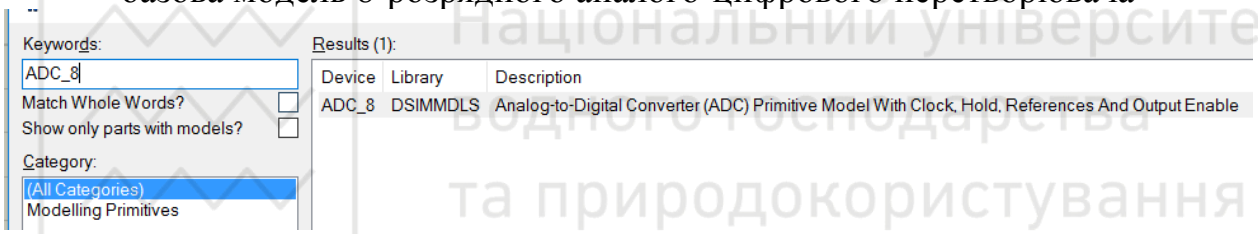
2.1. Створити новий проект в симуляторі Proteus.

Запустити програму Proteus. Створити новий проект (File -> New Project). Для кожної схеми рекомендується створювати новий проект.

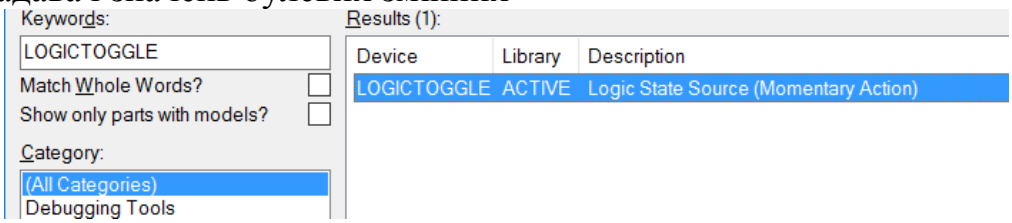
2.2. Вибрати необхідні елементи.

На оперативну панель необхідно додати наступні елементи:

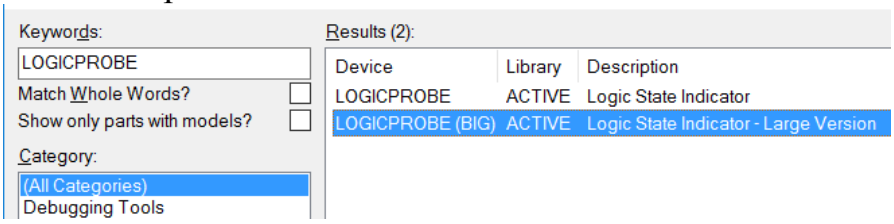
- базова модель 8-розрядного аналого-цифрового перетворювача



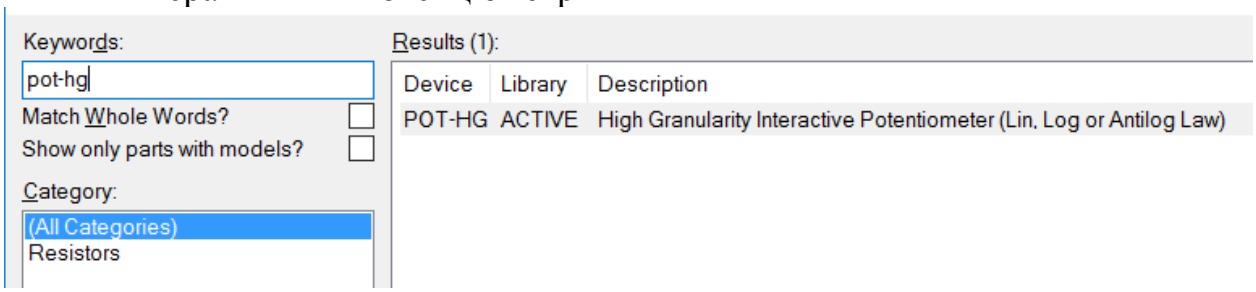
- задавач значень булевих змінних



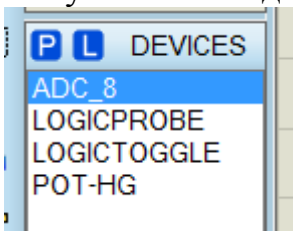
- індикатор значень двійкової змінної



- інтерактивний потенціометр



Оперативна панель матиме наступний вигляд:



2.3. Скласти принципову схему, що дозволяє досліджувати функціонування аналого-цифрового перетворювача.

Схема наведена на рис. 9.1. Модель АЦП (ADC1) має наступні виводи:

аналогові входи:

VIN – вхід аналогового сигналу АЦП;

VREF+ та VREF- - позитивний та негативний виводи опорної напруги;

цифрові входи:

HOLD – вхід дозволу фіксації рівня аналогового сигналу в АЦП; за замовчуванням фіксується за перепадом 0-1 на цьому вході;

CLK – вхід для подачі тактового сигналу, за замовчуванням значення на цифрові виводи видається по перепаду 0-1;

OE – вхід дозволу видачі сигналу на цифрові виходи: 1 – видача дозволена, 0 – заборона видачі (цифрові виходу у високоімпедансному стані);

цифрові виводи:

D0-D7 – шина для видачі результату перетворення.

Для подачі на вхід VIN АЦП у схемі на рис. 9.1 використовується інтерактивний потенціометр, який дозволяє змінювати напругу на VIN від 0 до +5 В. Напруга, яка знімається з потенціометра, вимірюється вольтметром. На вхід VREF+ АЦП подається опорна напруга +5В від джерела живлення. Вхід VREF- підключений до загального виводу джерела живлення. Цифрові входи HOLD і CLK підключені до задавача двійкових значень, який забезпечує запуск процесу перетворення за перепадом сигналу з лог. «0» до лог. «1». На вхід OE подається лог. «1» (+5 В), тобто видача сигналу на цифрові виходи дозволена. Результати перетворення видаються з цифрових виводів АЦП D0-D7 на двійкові індикатори.

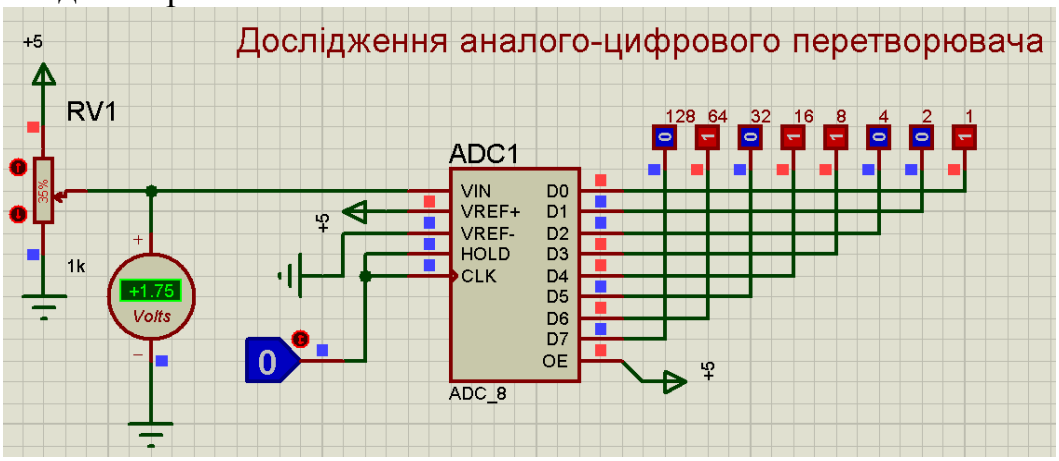


Рисунок 9.1 – Принципова схема для дослідження функціонування 8-розрядного аналого-цифрового перетворювача

2.4. Дослідити функціонування 8-розрядного аналого-цифрового перетворювача.

Для перетворення вхідного аналогового сигналу у двійковий цифровий код необхідно задати за допомогою інтерактивного потенціометра величину вхідного аналогового сигналу, подати на входи HOLD і CLK перепад сигналу 0-1 і зареєструвати показники вихідних двійкових індикаторів.

Наприклад, на рис. на вхід АЦП подається напруга 1,75 В. Після перетворення АЦП видав двійкове значення 01011001b. У десятковій системі числення отримане значення становить:

$$01011001b = 1\cdot64+1\cdot16+1\cdot8+1\cdot1 = 89.$$

Таким чином, напруга 1,75 В відповідає 89 рівням квантування. Перевіримо правильність роботи АЦП.

Діапазон вхідних значень даного АЦП = 0…5 В.

Розрядність АЦП – 8 біт, що відповідає 2^8=256 рівням квантування.

Розрядність АЦП за напругою: (5-0)/256 = 0,0195 В = 19,5 мВ.

Тобто кожному рівню квантування відповідає вхідна напруга 19,5 мВ.

Відповідно 89 рівням квантування відповідає напруга: 89·0,0195=1,74 В, що в межах допустимої похибка відповідає фактичній вхідній напрузі.

Необхідно задатися трьома довільними значенням вхідної напруги і занести до табл. 9.1 результати роботи АЦП.

Таблиця 9.1

Результати роботи АЦП

Вхідна напруга фактична, В	Результат перетворення у двійковому форматі	Результат перетворення у десятковому форматі	Вхідна напруга, обрахована за показами АЦП, В

ВМІСТ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

1. Тема, мета роботи.
2. Принципова схема для дослідження функціонування 8-розрядного аналого-цифрового перетворювача в симуляторі Proteus.
3. Опис призначення виводів АЦП.
4. Аналіз результатів роботи АЦП для трьох значень вхідної напруги.
5. Висновки з аналізом отриманих результатів.

КОНТРОЛЬНІ ПИТАННЯ

1. Для чого використовується АЦП?
2. Назвіть основні характеристики АЦП?
3. Для чого використовуються входи HOLD і CLK АЦП?

ЛАБОРАТОРНА РОБОТА №10
Створення елементарної програми для мікропроцесора Intel на мові асемблера

Мета: освоїти методи написання програми на мові асемблера для мікропроцесора Intel.

1. ЗАВДАННЯ

З використанням мови асемблера створити виконуваний файл типу *.exe, який виводить на екран привітання та ім□я студента.

2. ЗАГАЛЬНІ ВІДОМОСТІ

Для написання програми скористаємося Microsoft Macro Assembler (MASM) - це асемблер для архітектури x86 під операційну систему Microsoft Windows, що використовує синтаксис Intel. MASM існує у двох різновидах: для 16- і 32 бітного коду і для 64-бітного (ML64). MASM випускався як окремий продукт, а також вбудовується в інтегровані середовища розробки: Microsoft Visual Studio, RadASM, WinAsm Studio, EasyCode тощо. Лабораторна робота буде виконуватися з використанням MASM6.11.

Ця версія MASM не може бути використана в операційних системах Win7-10, тому для роботи з цим асемблером скористаємося операційною системою DOS.

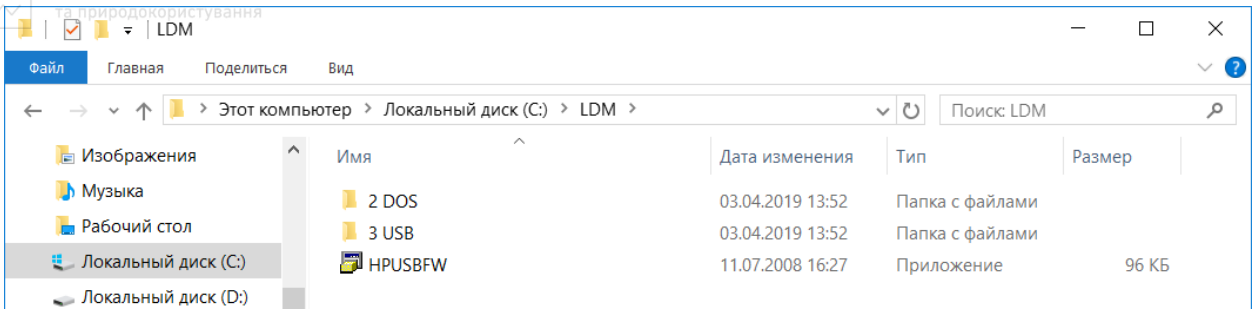
Операційну систему DOS передбачається завантажувати з флеш-накопичувачі. Для розміщенні DOS на накопичувачі використовується програма HP USB Disk Storage Format Tool (HPUSBFW.exe) – це утиліта для форматування та створення завантажувального USB накопичувача. На цьому ж накопичувачі розміщуватиметься і MASM.

Для використання MASM комп□ютер необхідно буде перезавантажити з флеш-накопичувача в операційній системі DOS. При цьому жорсткий диск комп□ютера буде недоступний, а накопичувач буде представляти єдиний диск C. В подальшому з накопичувача необхідно запустити MASM.

Для виконання роботи студентам видається архів LDM.ZIP з необхідним програмним забезпеченням. Також для виконання роботи необхідно мати флеш-накопичувач.

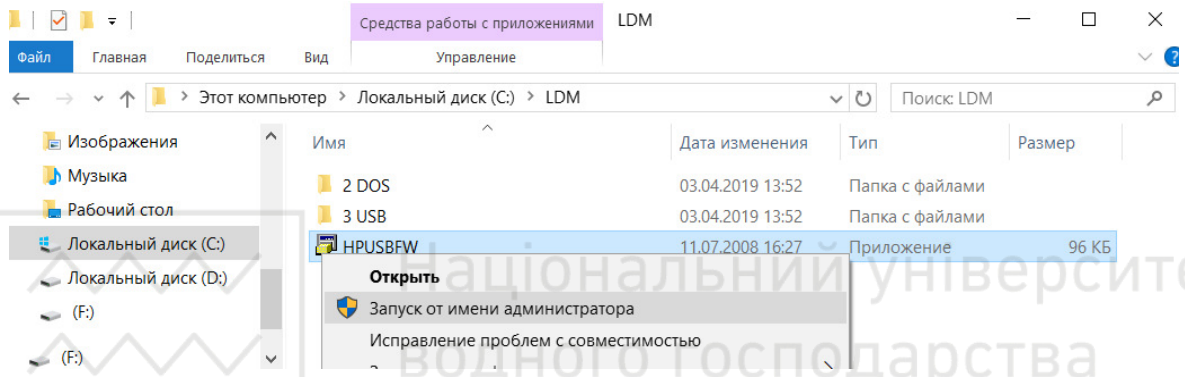
3. ПОРЯДОК ВИКОНАННЯ РОБОТИ

3.1 Скачати архів LDM.ZIP. Видобути з архіву папку LDM. Папку розмістити в корені диску C, тобто адреса папки має бути C:\ LDM. Папка включає наступні елементи:

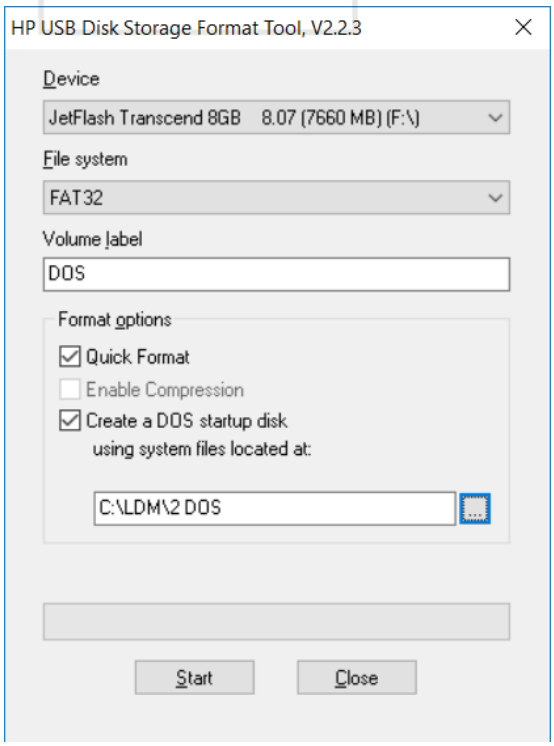


3.2 Підготувати флеш-накопичувач. **УВАГА!** Всі дані з накопичувача будуть видалені!!! Під'єднати флеш-накопичувач до комп'ютера.

3.3 З папки LDM запустити з правами адміністратора файл HPUSBFW.exe.

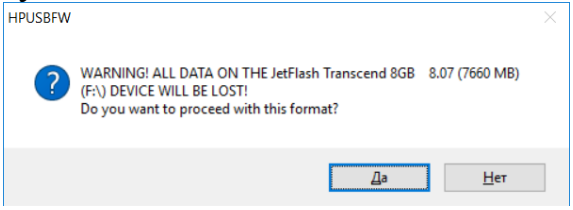


3.4 У вікні налаштувань параметрів утиліти HPUSBFW необхідно:

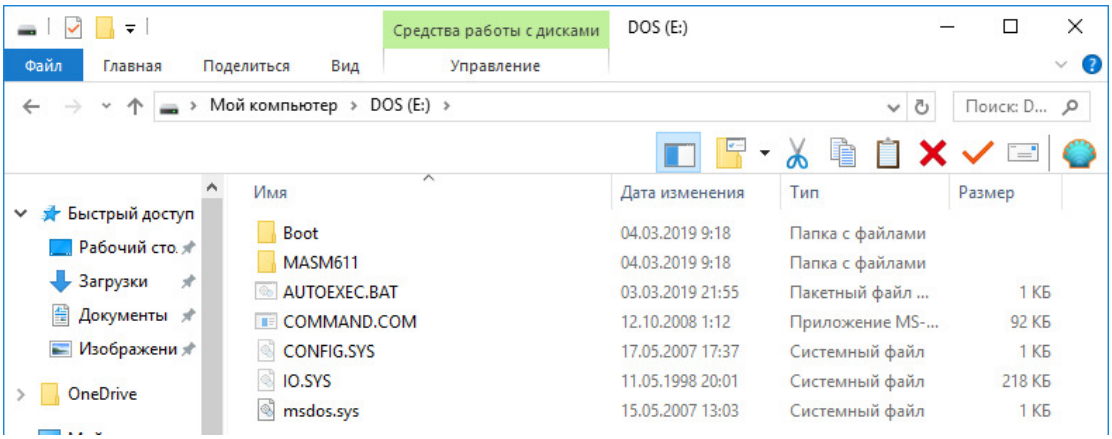


- в полі Device обрати флеш-накопичувач;
- в полі File system обрати тип файлової системи FAT32;
- в полі Volume lable можна задати мітку накопичувача, наприклад DOS;
- відмітити опцію «Quick Format», тобто обрати швидке форматування;
- відмітити опцію «Create a DOS startup disk» та вказати шлях "c:\LDM\2 DOS" до папки, де зберігаються системні файли DOS.

3.5 Натиснути кнопку Start. Програма видасть попередження про те, що всі дані з накопичувача будуть видалені. Треба натиснути «Так».

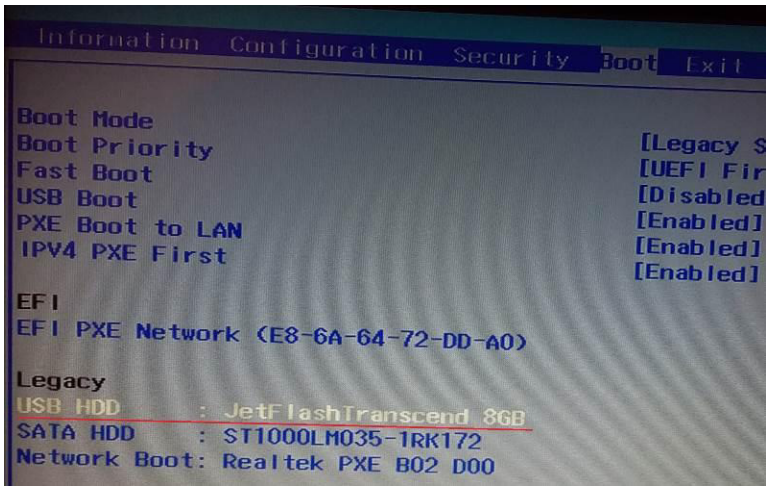


3.6 Після закінчення роботи програми необхідно скопіювати на накопичувач вміст папки 3 USB. Вміст накопичувача буде виглядати наступним чином:



3.7 Необхідно перезавантажити комп□ютер, не від□єднуючи накопичувач. Під час першого етапу завантаження необхідно ввійти до меню налаштувань BIOS, де встановити USB-накопичувач як перший пристрій, з якого має завантажуватися комп□ютер. Перехід до меню налаштувань BIOS у різних комп□ютерах здійснюється під час першого етапу завантаження комп□ютера шляхом натискання спецыальних кнопок. Для стаціонарних комп□ютерів найчастіше це Del або F2. Зазвичай потрібна кнопка вказується під час завантаження комп□ютера у таких повідомленнях: «Press F1 to continue, DEL to enter setup», «Press DEL to run setup» или «Please press DEL or F2 to enter UEFI BIOS settings». Для ноутбуків різних виробників використовуються наступні кнопки: Asus - F2 або Del + F9; Acer - F1, F2, або комбінація Ctrl+Alt+Esc, в серії Acer Aspire може використовуватися Ctrl+F2; Lenovo - F2, або Fn+F2; HP - F10 або Esc, на старих моделях - Del, F1, F11 або F8. Також цю інформацію можна уточнити в документації на ноутбук конкретного типу.

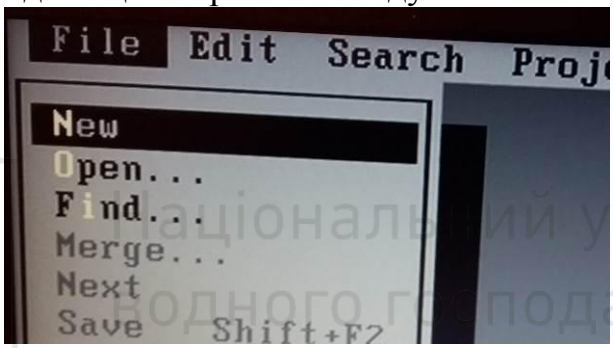
Наприклад, в наведеному меню BIOS пристрій USB HDD встановлений на першій позиції в переліку пристроїв, до яких BIOS звертається за завантаженням. Для виходу з BIOS і збереження налаштувань зазвичай потрібно натиснути F10.



3.8 Після виходу з BIOS комп□ютер перезавантажиться в операційній системі DOS, на екрані з□явиться запрошення DOS:

3.9 Для запуску інтегрованого середовища розробки MASM6.11 необхідно ввести команду pwb.exe і натиснути Enter.

3.10 У вікні середовища вибрати команду File – New

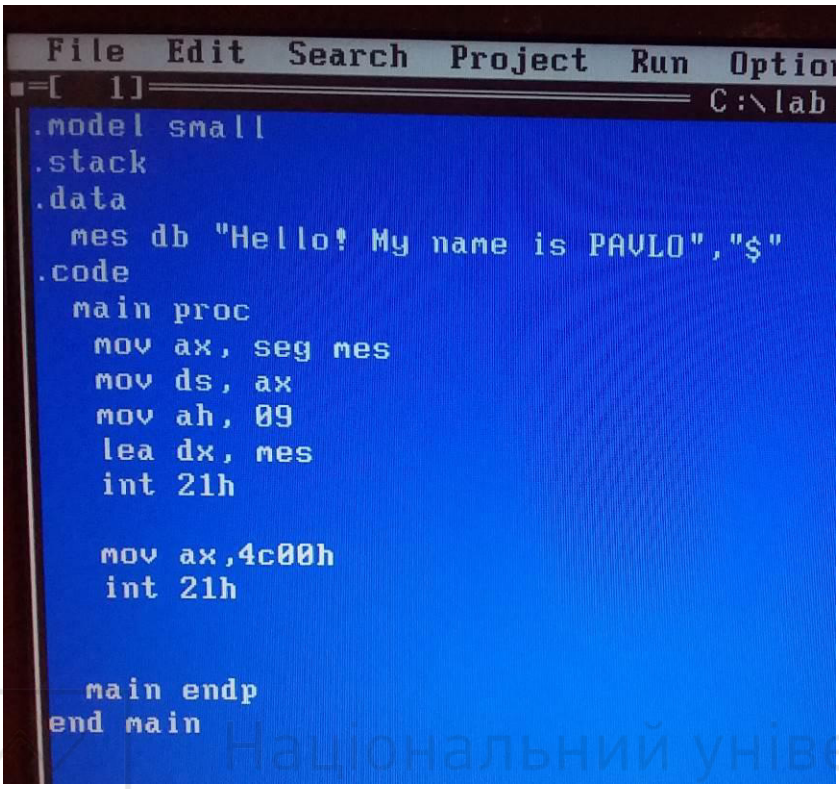


3.11 Зберегти новий файл, для чого вибрати команду File – Save as, вказати ім’я файлу LAB.ASM і зберегти в корені диску C.

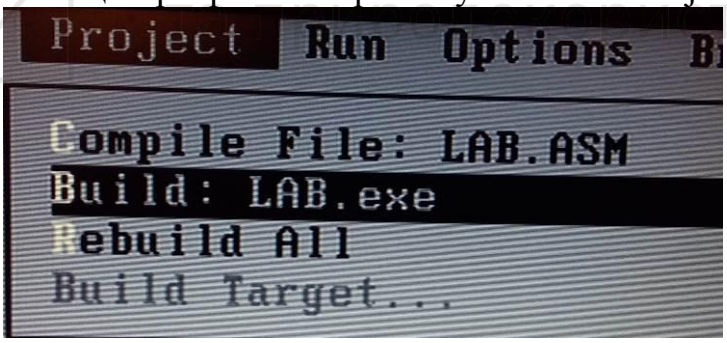
3.12 Набрати текст програми, яка виводить власне ім’я:

```
.model small
.stack
.data
  mes db "Hello! My name is PAVLO","$"
.code
  main proc
   mov ax, seg mes
   mov ds, ax
   mov ah, 09
   lea dx, mes
   int 21h
   mov ax,4c00h
   int 21h
  main endp
end main
```

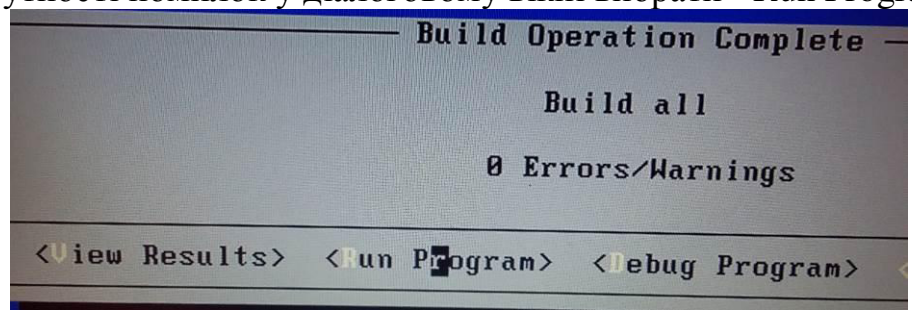
Вікно програми виглядатиме наступним чином:



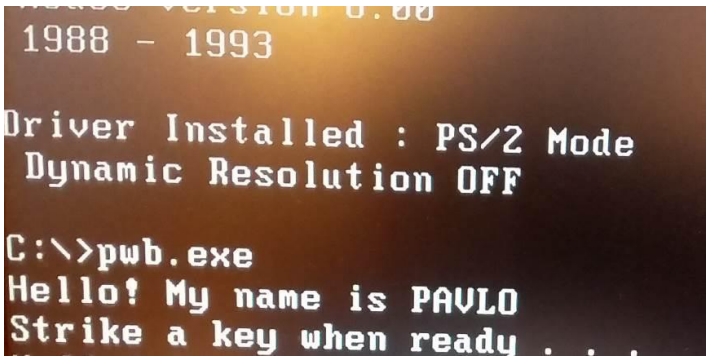
3.13 Зробити фотографію екрана комп□ютера для додавання до звіту.
3.14 Для компіляції програми вибрати пункт меню Project – Build: lab.exe.



При відсутності помилок у діалоговому вікні вибрати <Run Program>:

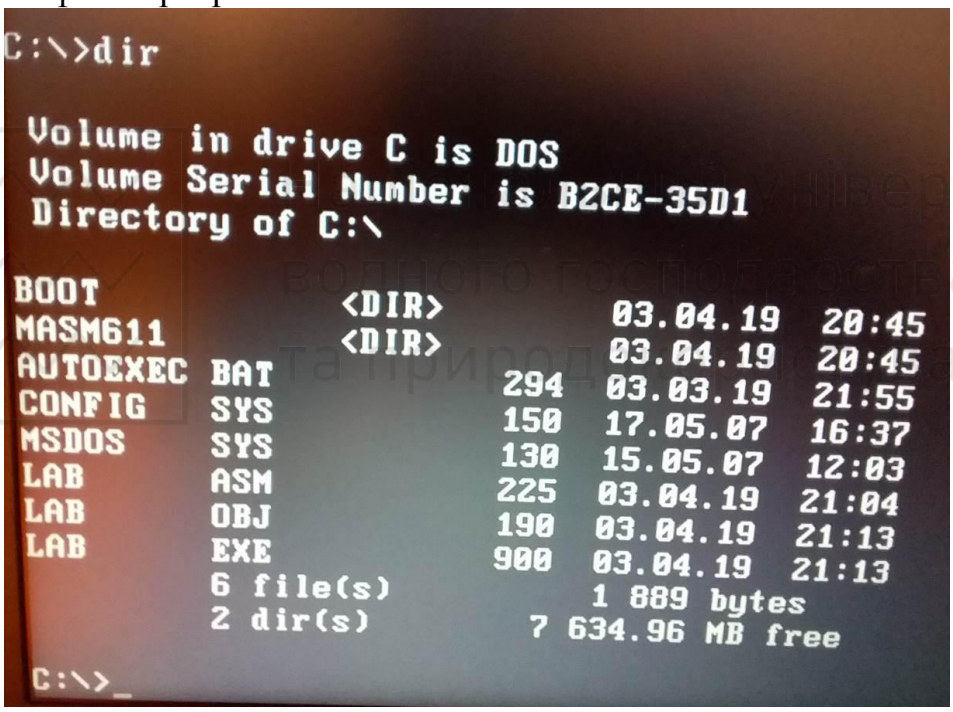


3.15 Ознайомитися з результатом роботи програми, зробити фотографію екрану.

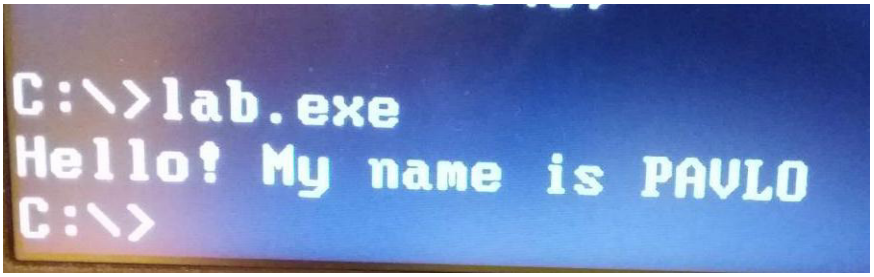


3.16 Вийти з програми, для чого вибрати пункт меню File – Exit.

3.17 Вивести вміст диску C (флеш-накопичувача), для чого виконати команду dir. Видно, що на диску C було створено 3 файли: LAB.ASM – файл з текстом програми на мові асемблера; LAB.OBJ – об□єктний файл; LAB.EXE – виконуваний файл програми.



3.18 Для запуску отриманої програми необхідно запустити файл LAB.EXE.



3.19 Перезавантажити комп□ютер, у BIOS повернути на перше місце завантаження з жорсткого диску комп□ютера.

ВМІСТ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

1. Тема, мета роботи.
2. Текст програми на мові асемблера.
3. Опис кожної команди, що використовується в програмі.
4. Фотографія екрану редактора MASM6.11 з текстом програми.
5. Фотографія результату виконання програми.
6. Висновки з аналізом отриманих результатів.

КОНТРОЛЬНІ ПИТАННЯ

1. Назвіть основні команди операційної системи DOS.
2. Для чого використовуються команди асемблера mov, int ?
3. Як позначається початок сегменту коду?
4. Якою директивою позначається початок сегмента даних?
5. Як позначається початок сегменту стеку?

ЛАБОРАТОРНА РОБОТА №11
Керування дискретними об’єктами за допомогою мікроконтролера

Мета: вивчити можливості мікроконтролерів AVR з керування дискретними об’єктами та формування часових затримок; освоїти методи написання програм для мікроконтролера AVR на мові асемблера (ATMEL Studio) та C for Arduino (IDE Arduino); навчитися симулювати роботу мікропроцесорної системи в програмі Proteus та завантажувати програму в реальну плату Arduino

1. КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Цифрові порти вводу-виводу мікроконтролерів AVR

Кожен порт мікроконтролера складається з певного числа виводів, через яких мікроконтролер може здійснювати прийом і передачу цифрових сигналів (рис. 11.1). Порти позначаються: буква "P" (port) + унікальна буква, що означає порт "A", "B" і так далі. У 8-розрядних контролерів порти мають 8 виводів. Виводи портів позначаються цифрами від 0 до 7. Приклад позначення виводу порту: PC6 - шостий вивід порту C. У мікроконтролерів AVR усі виводи, окрім виводів живлення, за замовчуванням є виводами цифрових портів введення-виводу.

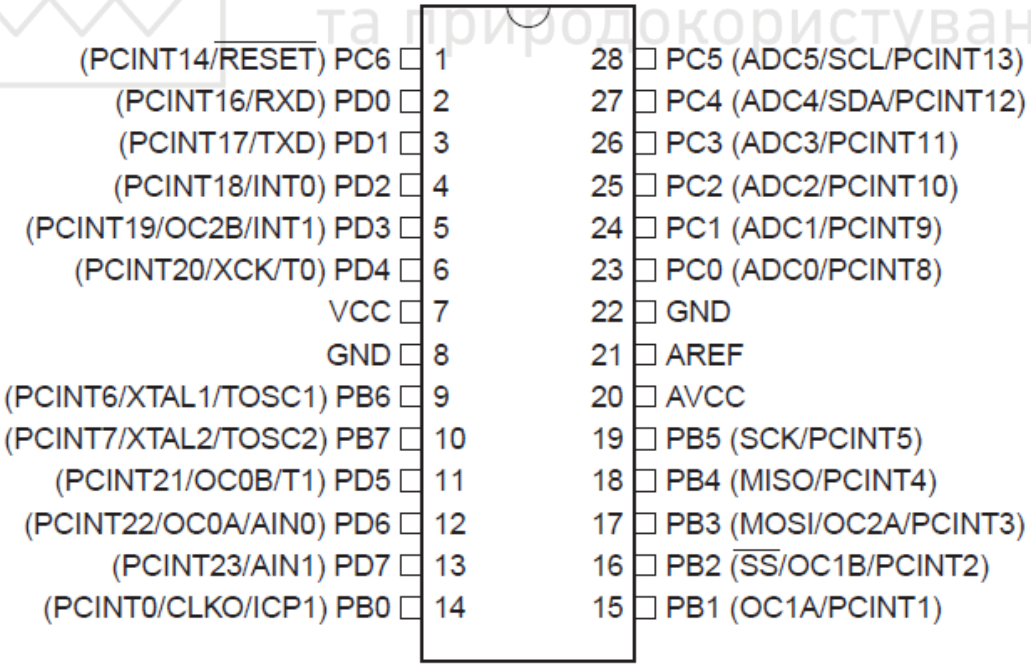


Рис. 11.1. Розміщення виводів мікроконтролера ATMega328p в корпусі PDIP

Звернення до портів здійснюється за допомогою регістрів вводу-виводу. Кожен порт описується програмною моделлю на основі трьох регістрів вводу-виводу (табл 11.1).

Таблиця 11.1

Регістри вводу-виводу, що використовуються для керування цифровими портами

№	Назва регістра	Позначення регістра		Позначення розрядів	
		В загальному вигляді	Приклад	В загальному вигляді	Приклад
1	Регістр напрямку передачі даних	DDR*x*	DDRB	DD*xn*	DDB1
2	Регістр даних	PORT*x*	PORTB	PORT*xn* або P*xn*	PORTB1 або PB1
3	Регістр стану виводів порта	PIN*x*	PINB	PIN*xn*	PINB1

де *x* – им'я порта (A, B, C, D), *n* – номер розряда (0-7)

Розглянемо призначення регістрів.

Регістр напрямку передачі даних DDRx визначає напрямок передачі даних через контакт вводу/виводу.

DD*xn*=1 - *n*-й вивід порта *x* є **виходом**

DD*xn*=0 - *n*-й вивід порта *x* є **входом**

Завдання напряму передачі даних через будь який контакт вводу/виводу може бути здійснено програмно у будь-який момент часу.

Призначення регістра даних PORTx залежить від режиму функціонування виведення мікроконтролера.

а) Якщо вивід конфігурований як **вихід** (у регістрі DDRx розряд DDxn=1), то за допомогою розряду Pxn регістра PORTx здійснюється **управління виводом**:

Pxn=1 - на відповідному виводі встановлюється напруга ВИСОКОГО РІВНЯ;

Pxn=0 - на виводі встановлюється напруга НИЗЬКОГО рівня.

б) Якщо вивід функціонує як **вхід** (у регістрі DDRx розряд DDxn=0), розряд Pxn регістра PORTx визначає **стан внутрішнього підтягуючого резистора** для цього виводу:

Pxn=1 - підтягуючий резистор підключається між виведенням мікроконтролера і шиною живлення Vcc;

Pxn=0 - підтягуючий резистор відключений.

Регістр стану виводів порту PINx дозволяє визначати стан виводу мікроконтролера (незалежно від установок розряду DDxn, тобто не залежно від напряму передачі даних через порт). Регістр PINx доступний тільки для читання.

1.2. Команди мови асемблера для управління цифровими портами вводу-виводу

Для роботи з цифровими портами використовуються наступні основні команди:

SBI A,b — **Встановлення розряду РВВ**. Встановлює розряд b регістра вводу-виводу, який розміщений за адресою A простору вводу-виводу. Команда застосовується тільки до молодших 32-м регістрів (адреси 0...31).

CBI A,b — **Скидання розряду РВВ**. Скидає розряд b регістра вводу-виводу, який розміщений за адресою A простору вводу-виводу. Команда застосовується тільки до молодших 32-м

OUT A,Rr — **Пересилання значення з РЗП до РВВ.** Пересилає вміст регістра загального призначення Rr в регістр вводу-виводу A.

1.3. Функції мови C for Arduino для управління цифровими портами вводу-виводу

Для роботи з цифровими портами використовуються наступні основні команди:

pinMode(pin, mode) — Встановлює **режим роботи** виводу з номером pin плати як входу (mode=INPUT) або як виходу (mode=OUTPUT) .

digitalWrite(pin, value) — **Подає напругу** ВИСОКОГО (value=HIGH) або НИЗЬКОГО (value=LOW) рівня на вивід плати з номером pin.

1.4. Формування часових затримок на мові асемблера

Найбільш простий спосіб формування часових затримок - організація циклу, який виконується задану кількість разів. Оскільки виконання кожної команди циклу вимагає певного часу, при правильному підборі значення лічильника циклу, знаючи частоту тактового генератора мікроконтролера, можливо організувати необхідну затримку. Іноді для збільшення тривалості затримки використовують декілька вкладених циклів.

Для формування часової затримки пропонується використовувати підпрограму DELAY. Вхідним параметром такої підпрограми є тривалість затримки в мілісекундах, значення якої зберігається в регістровій парі X.

```
DELAY:

        ldi YL,low(3997)
        ldi YH,high(3997)
Ld1:        sbiw YL,1
            brne Ld1
        sbiw XL,1
        brne DELAY
        ret
```

Така підпрограма складається з двох циклів: зовнішнього, лічильником якого є регістрова пара X (регістри r27:r26), і внутрішнього, де в якості

лічильника використовується регістрова пара Y (регістри r29:r28). Випишемо в табл. 11.2 число тактів (періодів) генератора мікроконтролера для кожної команди підпрограми затримки.

Таблиця 11.2

Підрахунок тактів для виконання підпрограми затримки

Мітка	*Команда*	*Кількість тактів*
	rcall DELAY	3
	. . . . .	
DELAY:	**ldi** YL,low(3997)	1
	ldi YH,high(3997)	1
Ld1:	**sbiw** YL,1	2
	brne Ld1	2 – якщо перехід на мітку 1 – якщо немає перехода
	sbiw XL,1	2
	brne DELAY	2 - якщо перехід на мітку 1 – якщо немає перехода
	ret	4

Загальна кількість тактів, яка потрібна для виконання внутрішнього цикла, складає:

$$X_1 = \underbrace{(\ 2}_{\textbf{sbiw}\ \texttt{YL,1}} + \underbrace{2\ )}_{\textbf{brne}\ \texttt{Ld1}} \cdot \underbrace{Y}_{\substack{\text{лічильник}\\ \text{внутрішнього}\\ \text{цикла}}} - \underbrace{1}_{\substack{\text{оскільки при виході з}\\ \text{цикла команда}\\ \textbf{brne}\ \texttt{Ld1}\\ \text{виконується не за 2, а}\\ \text{за 1 такт}}} = 4 \cdot Y\text{-}1$$

Загальна кількість тактів, яка потрібна для виконання зовнішнього цикла, становить:

$$X_2 = (\ \underbrace{1}_{\substack{\textbf{ldi}\ \texttt{YL,}\\ \texttt{low(Y)}}} + \underbrace{1}_{\substack{\textbf{ldi}\ \texttt{YH,}\\ \texttt{high(Y)}}} + \underbrace{X_1}_{\substack{\text{Загальна}\\ \text{кількість}\\ \text{тактів}\\ \text{внутрішн}\\ \text{ього}\\ \text{цикла}}} + \underbrace{2}_{\substack{\textbf{sbiw}\\ \texttt{XL,1}}} + \underbrace{2\ )}_{\substack{\textbf{brne}\\ \texttt{DELAY}}} \cdot \underbrace{X}_{\substack{\text{лічильник}\\ \text{зовнішньог}\\ \text{о цикла}\\ \text{(вміст}\\ \text{регістрової}\\ \text{пари } X)}} - \underbrace{1}_{\substack{\text{оскільки при виході з цикла}\\ \text{команда}\ \textbf{brne}\ \texttt{DELAY}\\ \text{виконується не за 2, а за 1}\\ \text{такт}}} = (6 + X_1) \cdot X - 1$$

Підставимо до виразу для X_2 значення X_1:

$$X_2 = (6 + 4Y - 1) \cdot X - 1 = X \cdot (4Y + 5) - 1.$$

Загальна кількість тактів, яка потрібна для виконання підпрограми затримки, становить:

$$N = \underbrace{3}_{\substack{\textbf{rcall}\\ \texttt{DELAY}}} + \underbrace{X_2}_{\substack{\text{Загальна кількість тактів}\\ \text{зовнішнього цикла}}} + \underbrace{4}_{\textbf{ret}} = X \cdot (4Y + 5) + 6$$

Загальний час затримки:

$$T_з = N \cdot T_{MK},$$

де N - загальна кількість тактів підпрограми затримки;

$T_{MK} = \frac{1}{f_{MK}}$ - тривалість такта генератора мікроконтролера;

f_{MK} - частота тактового сигнала.

Тоді загальна тривалість затримки становить:

$$T_з = \frac{N}{f_{MK}} = \frac{X \cdot (4Y + 5) + 6}{f_{MK}}.$$

Мікроконтролер плати Arduino Uno налаштовано на роботу від зовнішнього кварцового резонатора, частота якого f_{MK} = 16 МГц=16·10^6 Гц.

Приймемо, що підпрограма `DELAY` при X=1 формує затримку 1мс=0,001с, тобто нехай при X=1 $T_з = 0{,}001$, тоді:

$$0{,}001 = \frac{1 \cdot (4Y + 5) + 6}{f_{MK}},$$

звідки:

$$Y = \frac{0{,}001 \cdot f_{MK} - 11}{4} = \frac{0{,}001 \cdot 16 \cdot 10^6 - 11}{4} = 3997{,}25 \approx 3997.$$

Отримане значення регістрової пари Y і використано в підпрограмі:

```
DELAY:

        ldi YL,low(3997)
        ldi YH,high(3997)
```

де перша команда заносить до молодшого регістра YL (регістр r28) регістрової пари Y молодший байт числа 3997, друга команда заносить до старшого регістра YL (регістр r29) регістрової пари Y старший байт числа 3997.

1.5. Функції організації часових затримок на мові C for Arduino

delay(ms)	Зупиняє виконання програми на ms мілісекунд
delayMicroseconds(us)	Зупиняє виконання програми на us мікросекунд (рекомендується обирати us>16383)

2. ОПИС ОБЛАДНАННЯ, ЩО ВИКОРИСТОВУЄТЬСЯ

2.1. Експериментальний стенд

Експериментальний стенд для проведення лабораторних робіт (рис. 11.2, 11.3) включає:

- лампу розжарювання HL1, яка управляється оптосимістором V1, для виміру температури патрона лампи використовується термістор R11, для синхронізації з напругою мережі використовується оптопара V2;

- динамік BA1;

- двигун постійного струму M вентилятора; для виміру швидкості обертання використовується оптичний інфрачервоний датчик VD10 - VD11;

- потенціометр R25;

- світлодіоди VD1 - VD4;

- кнопки S1 - S3.

Управляючі виводи всіх перерахованих пристроїв підключені до роз'єму, який знаходиться в лівому нижньому кутку лицьової панелі стенду. За допомогою сполучних провідників необхідні пристрої підключаються до плати Adruino.

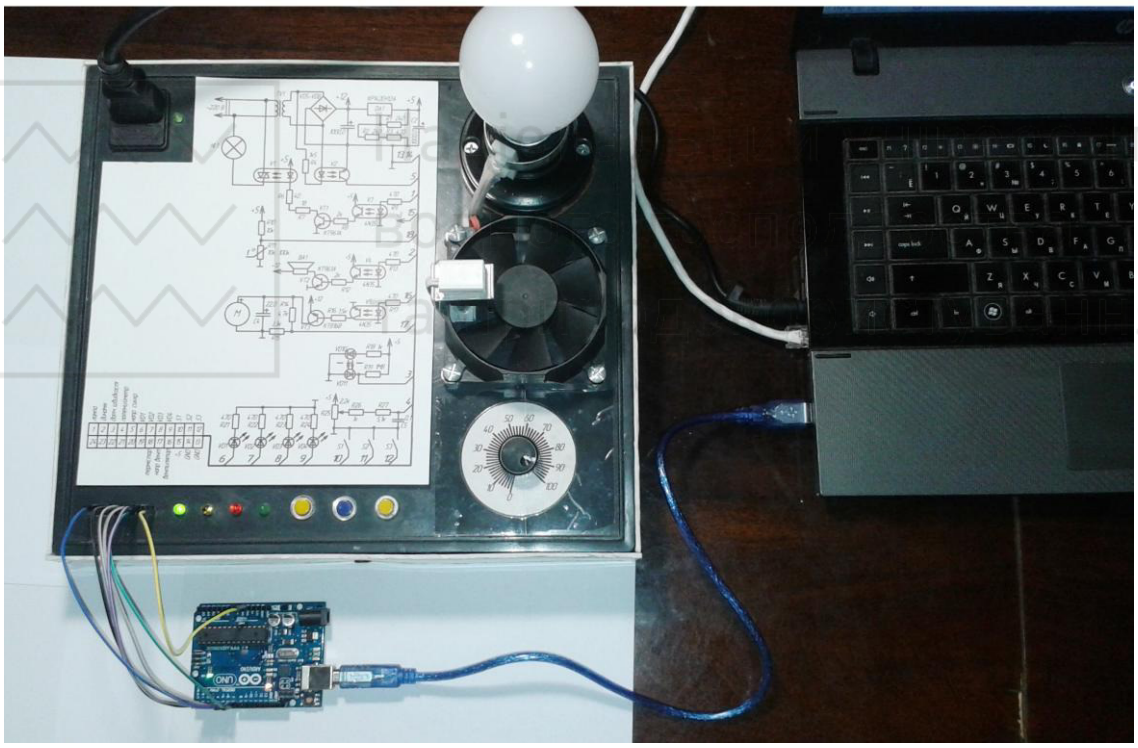


Рисунок 11.2 - Загальний вигляд експериментального стенда з платою Arduino Uno

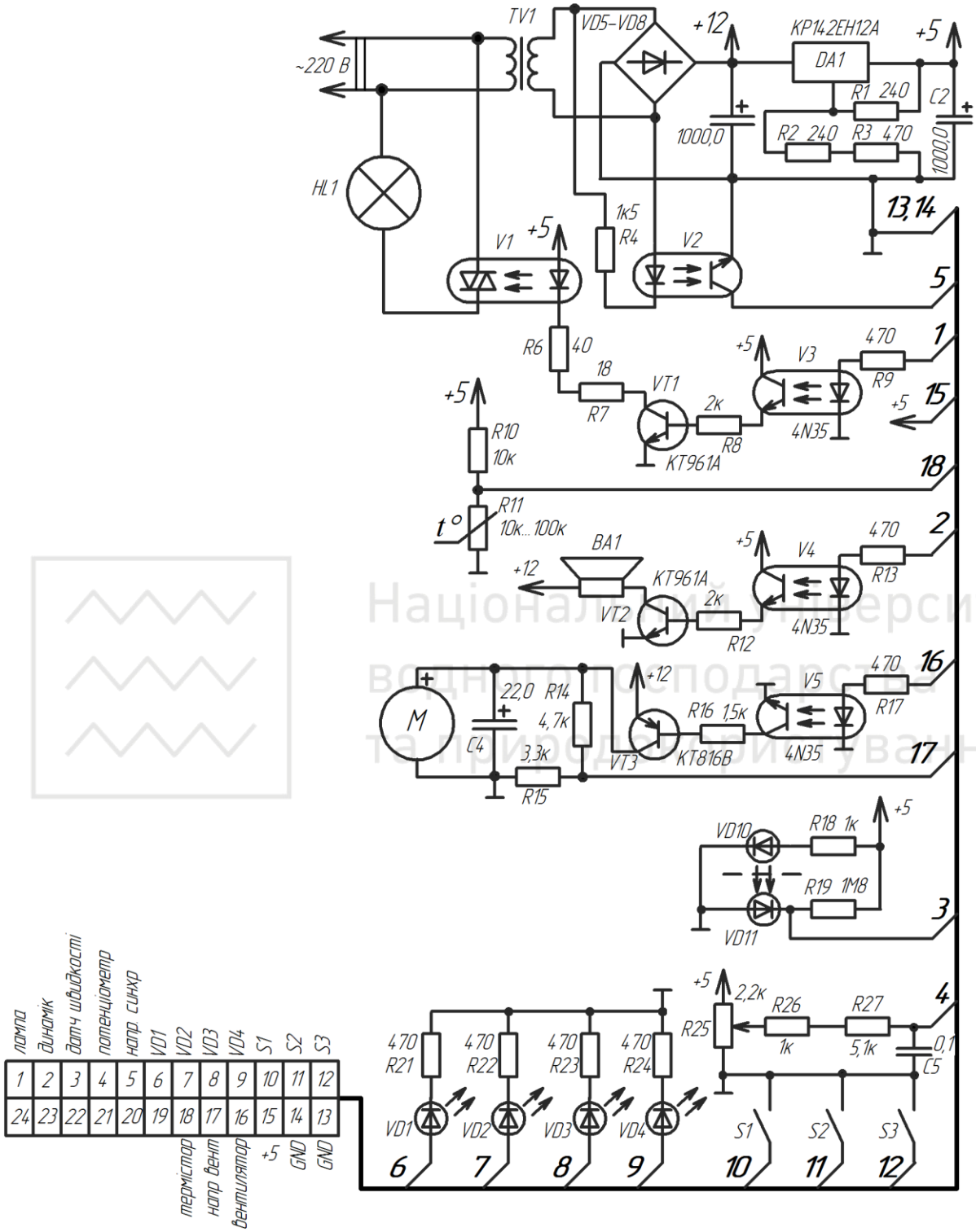


Рисунок 11.3 - Принципова схема експериментального стенда

2.2. Модель експериментального стенда в Proteus

Загальний вигляд моделі стенда в програмі Proteus, що використовується для виконання поточної лабораторної роботи, наведено на рис. 11.4.

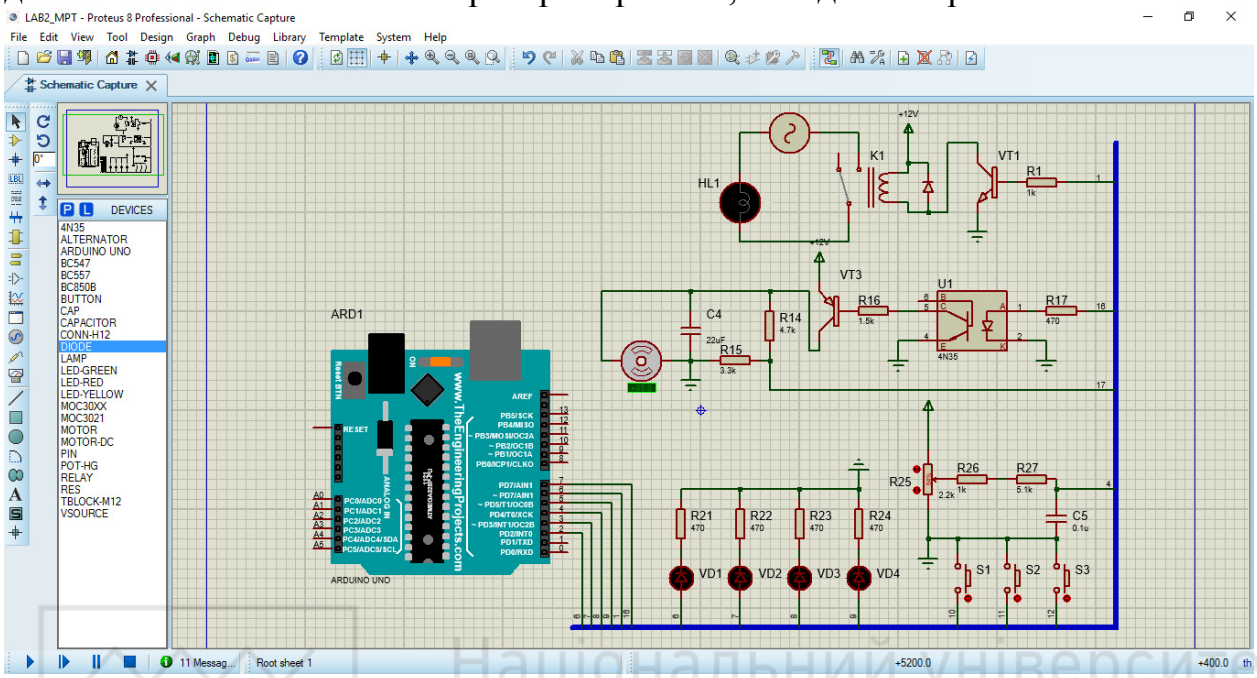


Рисунок 11.4 - Модель лабораторного стенда в Proteus
(конфігурація для виконання даної лабораторної роботи)

3. ЗАВДАННЯ

Скласти програму для контролера плати Arduino UNO, яка виконує наступну послідовність дій:

1) ввімкнення світлодіода VD1;
2) затримка 1с;
3) ввімкнення лампи розжарювання HL1;
4) затримка 2с;
5) ввімкнення вентилятора M;
6) затримка 2с;
7) відключення лампи розжарювання HL1;
8) ввімкнення світлодіодів VD2, VD3, VD4;
9) затримка 3с;
10) відключення вентилятора;
11) затримка 2с;
12) відключення всіх дискретних об’єктів;
13) затримка 4с;
14) повернення до п. 1.

Програму необхідно написати на мовах асемблера та C for Arduino, відлагодити на моделі та реальному стенді.

Плата Arduino Uno підключається до експериментального стенду згідно табл. 11.3.

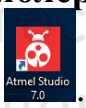
Таблиця 11.3

Підключення стенда до Arduino Uno для
виконання лабораторної роботи

Об’єкт	№ вихідного контакту стенда	Плата Arduino Uno	
		№ порта плати Arduino Uno	порт мікроконтролера ATMega328p
світлодіод VD1	6	2	PD2
світлодіод VD2	7	3	PD3
світлодіод VD3	8	4	PD4
світлодіод VD4	9	5	PD5
лампа розжарювання	1	6	PD6
вентилятор	16	7	PD7

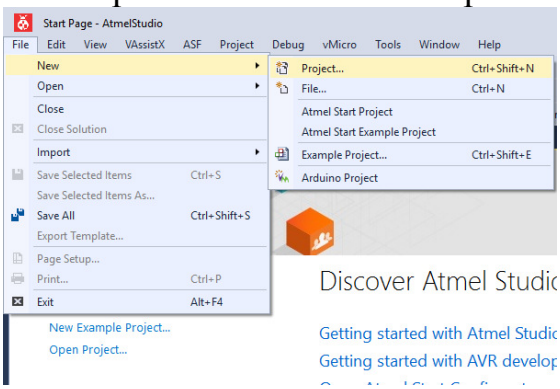
4. ПОРЯДОК ВИКОНАННЯ РОБОТИ

4.1. Написання програми на мові асемблера

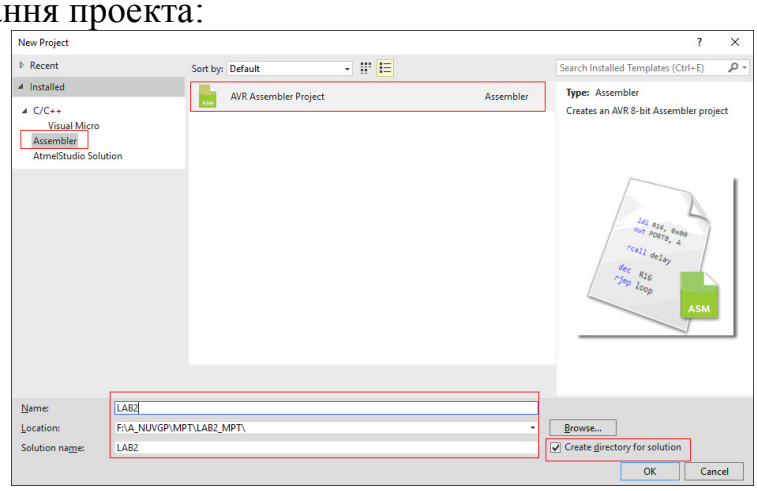
4.1.1. Відкрити програму Atmel Studio 

.

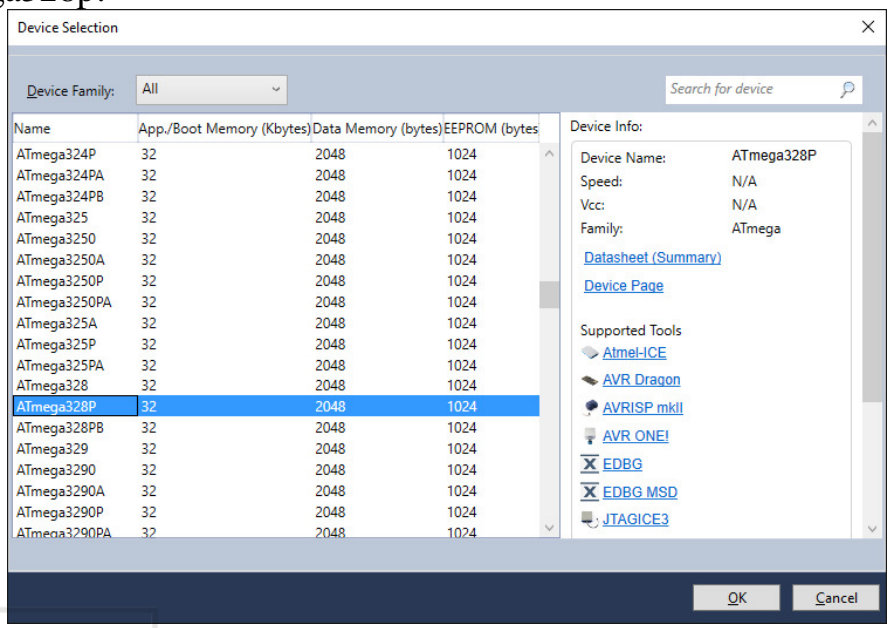
4.1.2. Створити новий проект на мові асемблера:



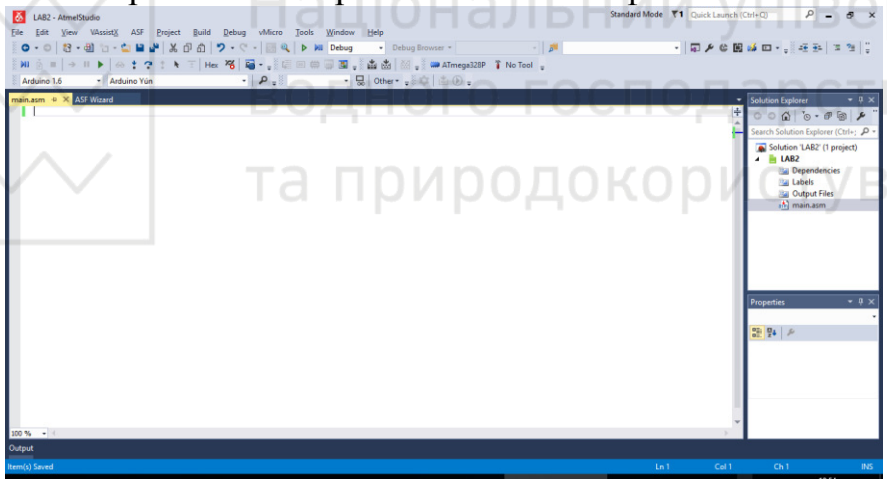
При створенні проекту обрати AVR Assembler Project, вказати ім’я та місце розташування проекта:



Обрати тип мікроконтролера, який встановлено в платі Arduino Uno, а саме ATMega328p:



Програма створить новий проект з головним файлов main.asm:



4.1.3. Набрати наступну програму:

```
.INCLUDE "m328pdef.inc"                 ;підключення бібліотеки
                                        ;визначення символічних імен PBB
                ldi r16,high(RAMEND)    ;ініціалізація стека
                out SPH,r16
                ldi r16,low(RAMEND)
                out SPL,r16
                                        ;конфігурування виводів
                ldi r16,0b11111100      ;PD2-PD7 - виходи
                out DDRD,r16
Lstart:                                 ;основний цикл програми
                sbi PORTD,PD2           ;ввімкнути VD1

                ldi XL,low(1000)        ;тривалість затримки в мс
                ldi XH,high(1000)
                rcall DELAY             ;виклик підпрограми затримки

                sbi PORTD,PD6           ;ввімкнути лампу розжарювання
```

```
            ldi XL,low(2000)        ;тривалість затримки в мс
            ldi XH,high(2000)
            rcall DELAY             ;виклик підпрограми затримки

            sbi PORTD,PD7           ;ввімкнути вентилятор

            ldi XL,low(2000)
            ldi XH,high(2000)
            rcall DELAY

            cbi PORTD,PD6           ;вимкнути лампу розжарювання
            sbi PORTD,PD3           ;ввімкнути VD2
            sbi PORTD,PD4           ;ввімкнути VD3
            sbi PORTD,PD5           ;ввімкнути VD4

            ldi XL,low(3000)
            ldi XH,high(3000)
            rcall DELAY

            cbi PORTD,PD7           ;вимкнути вентилятор

            ldi XL,low(2000)
            ldi XH,high(2000)
            rcall DELAY

            clr r16                 ;відключити все
            out PORTD,r16

            ldi XL,low(4000)
            ldi XH,high(4000)
            rcall DELAY

            rjmp Lstart             ;повернення до початку
                                    ;основного цикла

DELAY:                              ;підпрограма затримки
                                    ;вхід X – тривалість затримки в мс
            ldi YL,low(3997)
            ldi YH,high(3997)
Ld1:        sbiw YL,1
                  brne Ld1
            sbiw XL,1
            brne DELAY
            ret
```

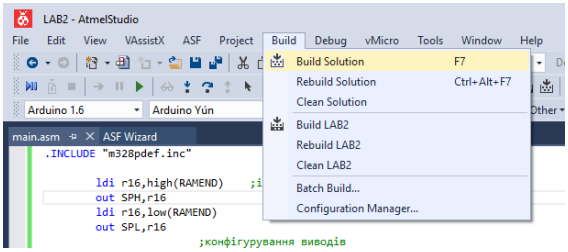

Така програма складається з декількох логічних частин:

а) Ініціалізація стека, конфігурування портів.

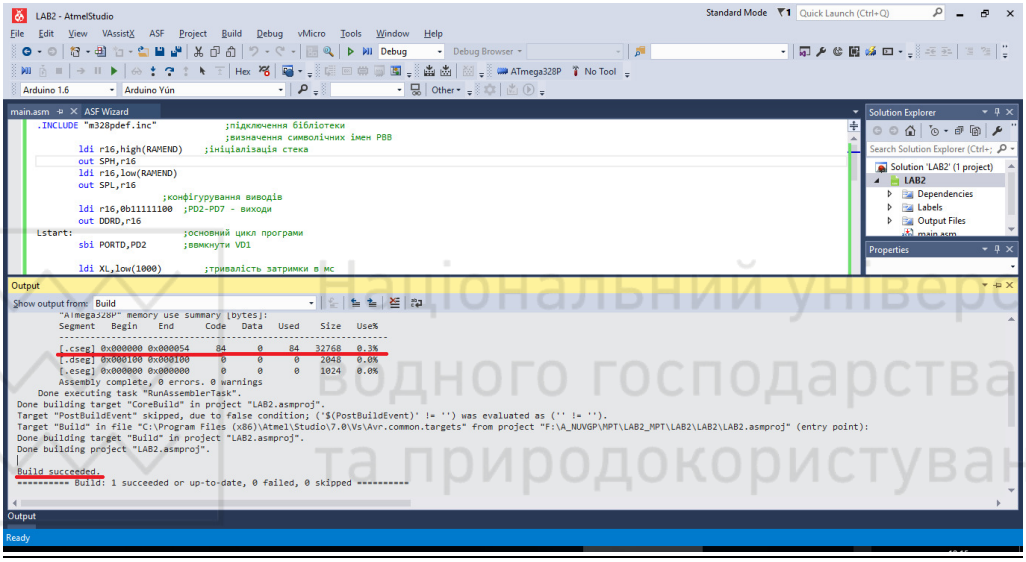
б) Основна програма, яка вмикає та вимикає дискретні об’єкти. Основна програма використовує процедуру затримки `DELAY`, причому параметр, який визначає тривалість затримки, передається в процедуру `DELAY` за допомогою регістрової пари X.

в) Підпрограма затримки `DELAY`.

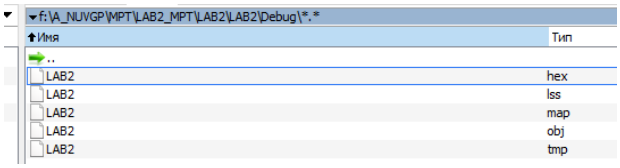
4.1.4. Виконати компіляцію програми, для чого обрати
Build → Build Solution



В разі успішної компіляції у вікні Output з’явиться повідомлення Build suceeded. Також у вікні Output відображається інформація щодо використання пам’яті мікроконтролера. Програма займає 84 байта пам’яті програм, що становить 0,3 % від загального обсягу (32768 байт) пам’яті програм.



Після успішної компіляції у папці DEBUG проекту буде сформований *.hex файл з кодами команд для мікроконтролера. Цей файл необхідно завантажити до пам’яті програм мікроконтролера («прошити» контролер).



Для закриття проекту на мові асемблера в Atmel Studio виконати команду File => Close Solution.

4.2. Завантаження програми на мові асемблера до моделі експериментального стенда

В моделі стенда, що зібрана в програмі Proteus, два рази клацнути мишею на платі Arduino, з’явиться вікно властивостей плати. В полі Program File необхідно вказати шлях до *.hex файла програми, рис. 11.5.

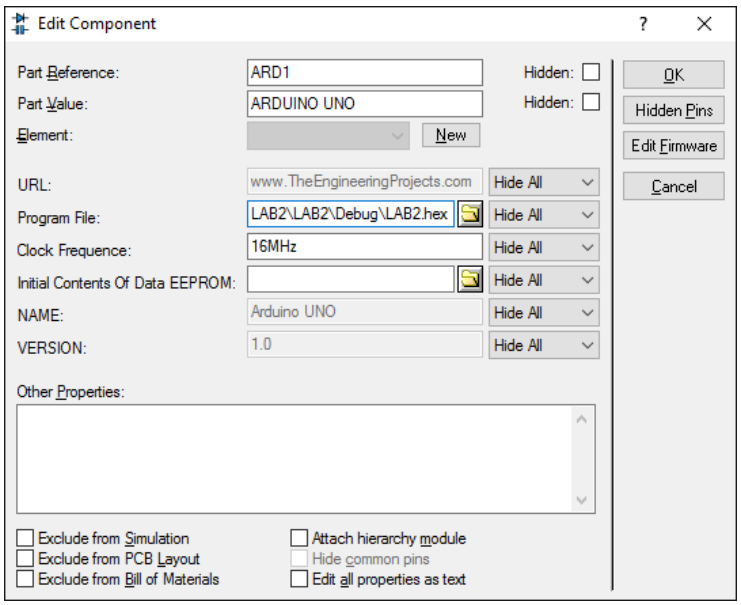


Рисунок 11.5 - Вікно властивостей плати ARDUINO у Proteus

Запустити модель стенда кнопкою Run:

Відслідкувати функціонування стенда. Зберегти Print Screen екрана з працюючою моделлю стенда для звіту.

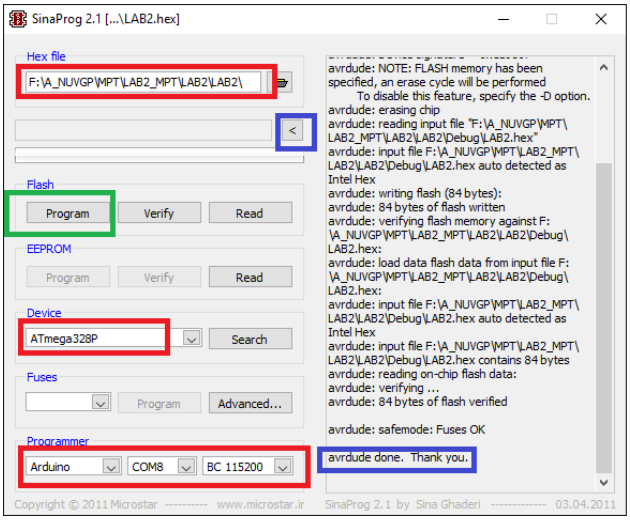
4.3. Завантаження програми на мові асемблера до експериментального стенда

4.3.1. Зібрати схему досліду, показати її викладачеві.

4.3.2. Підключити плату Arduino до комп'ютера.

4.3.3. Запустити програму SinaProg, вказати розміщення hex-файла, налаштувати розділи «Device» і «Programmer».

4.3.4. Для прошивки мікроконтролера натиснути Program в **розділі Flash**. В разі успішної прошивки з'явиться відповідне повідомлення.



4.3.5. Відслідкувати функціонування стенда.

4.4. Написання програми на мові C for Arduino

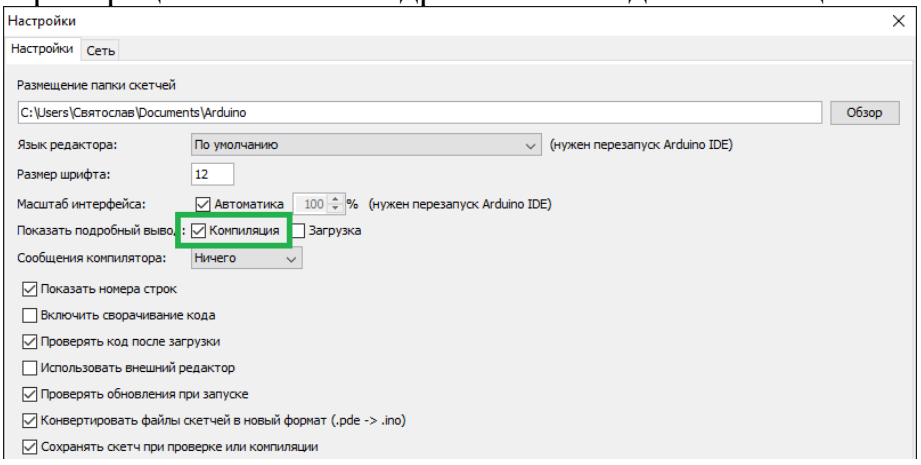
4.4.1. Запустити програму IDE Arduino .

4.4.2. Набрати наступну програму:

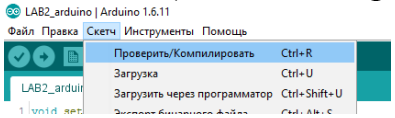
```
void setup()
{    pinMode(2,OUTPUT);       //конфігурація виходів
     pinMode(3,OUTPUT);
     pinMode(4,OUTPUT);
     pinMode(5,OUTPUT);
     pinMode(6,OUTPUT);
     pinMode(7,OUTPUT);
}

void loop()
{    digitalWrite(2,HIGH);    //ввімкнути VD1
     delay(1000);             //затримка 1с
     digitalWrite(6,HIGH);    //ввімкнути лампу
     delay(2000);             //затримка 2с
     digitalWrite(7,HIGH);    //ввімкнути вентилятор
     delay(2000);             //затримка 2с
     digitalWrite(6,LOW);     //вимкнути лампу
     digitalWrite(3,HIGH);    //ввімкнути VD2
     digitalWrite(4,HIGH);    //ввімкнути VD3
     digitalWrite(5,HIGH);    //ввімкнути VD4
     delay(3000);             //затримка 3с
     digitalWrite(7,LOW);     //вимкнути вентилятор
     delay(2000);             //затримка 2с
     digitalWrite(2,LOW);     //вимкнути VD1
     digitalWrite(3,LOW);     //вимкнути VD2
     digitalWrite(4,LOW);     //вимкнути VD3
     digitalWrite(5,LOW);     //вимкнути VD4
     delay(4000);             //затримка 4с
}
```

4.4.3. Зберети папку скетча (Файл → Зберегти як)

4.4.4. Відкрити вікно налаштувань програми (Файл → Налаштування). Встановити прапорець «Показать подробный вывод: компиляция»:



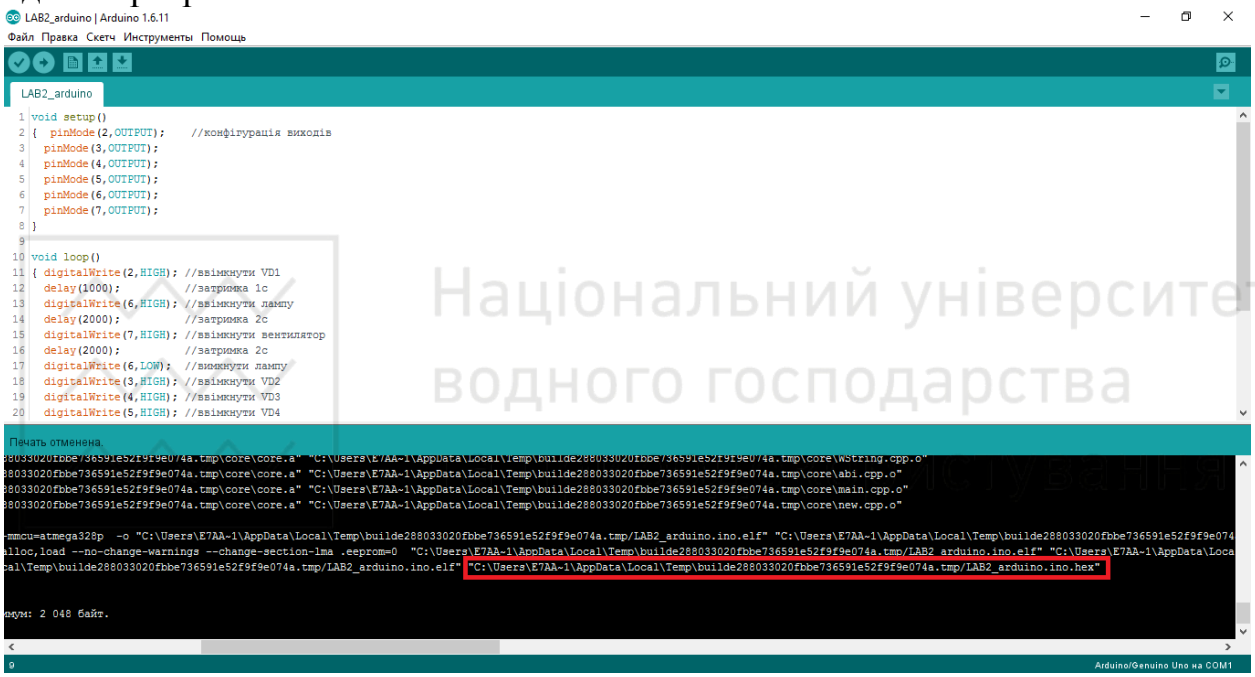
4.4.5. Виконати компіляцію скетча (Скетч → Перевірити/Компілювати):



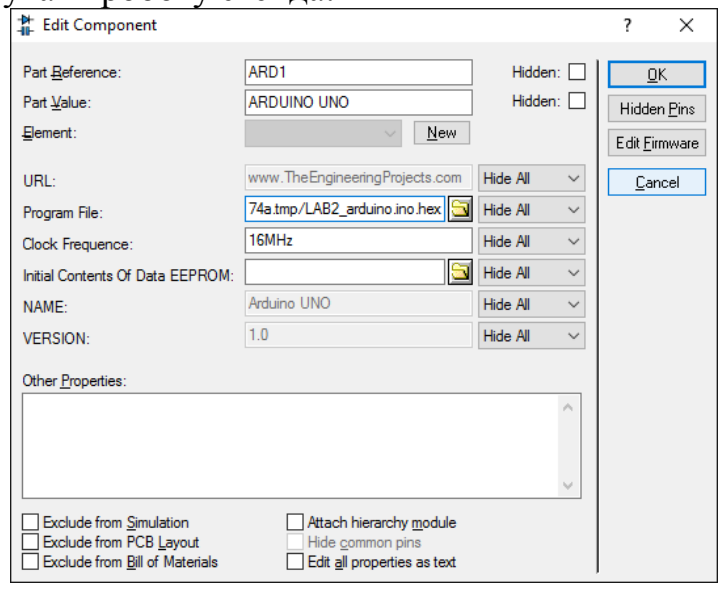
Слід звернути увагу, що компільований скетч використовує 1 134 байт (3%) пам'яті програм пристрою, що у 13 разів більше, ніж аналогічна програма на мові асемблера.

4.5. Завантаження програми на мові C for Arduino до моделі експериментального стенда

З вікна IDE Arduino скопіювати шлях до *.hex файла з шістнадцятковими кодами програми:



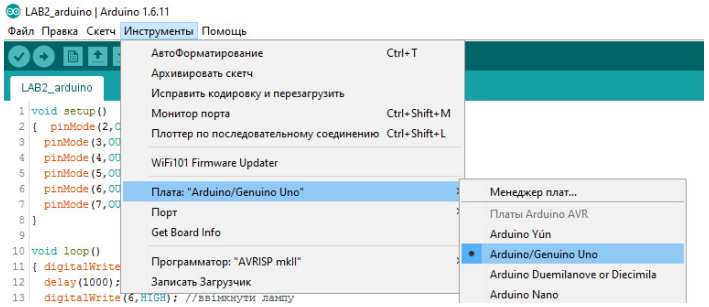
Вставити цей шлях до моделі плати Arduino в Proteus і запустити симуляцію. Відслідкувати роботу стенда.

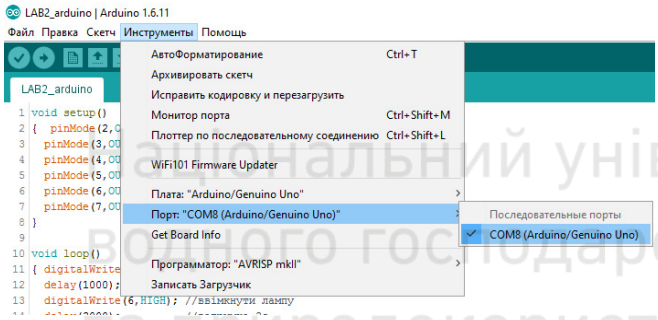


4.6. Завантаження програми на мові C for Arduino до експериментального стенда

4.6.1. Під'єднати плату Arduino до комп'ютера.

4.6.2. В меню «Інструменти» обрати тип плати «Adruino Uno» та послідовний порт. Визначити номер послідовного порта можна за допомогою Диспетчера пристроїв.





4.6.3. Для прошивки плати необхідно натиснути кнопку «Завантажити» на панелі інструментів, або обрати з меню Скетч пункт Завантажити. Після цього виконається скидання плати та почнеться завантаження програми у пам'ять. В процесі завантаження будуть мигати світлодіоди RX і TX.

4.6.4. Відслідкувати функціонування стенда.

ВМІСТ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

1. Тема, мета роботи.
2. Блок-схема алгоритму функціонування мікроконтролера (складається самостійно) з поясненнями.
3. Повний текст програми на мові асемблера та вікно Atmel Studio з програмою на мові асемблера.
4. Опис всіх команд, що використовуються в програмі на мові асемблера, з поясненнями їх синтаксису та призначення в програмі.
5. Програма на мові C for Arduino та вікно IDE Arduino з програмою.
6. Опис всіх команд, що використовуються в програмі на мові C for Arduino, з поясненнями їх синтаксису та призначення в програмі.
7. Модель лабораторного стенда в Proteus.
8. Висновки з аналізом ефективності використання пам'яті програм мікроконтролера при написанні програми на мові асемблера та C for Arduino.

КОНТРОЛЬНІ ПИТАННЯ

1. Які регістри мікроконтролера AVR використовуються для керування дискретними портами введення-виведення?
2. Як налаштувати вивід мікроконтролера для роботи в якості виходa або входа?
3. Які команди мови асемблера використовуються для управління цифровими портами вводу-виводу?
4. Які функції мови C for Arduino застосовуються для управління цифровими портами вводу-виводу?
5. Пояснити принцип формування часових затримок на мові асемблера за допомогою циклів.
6. Які Вам відомі функції організації часових затримок на мові C for Arduino?

ЛАБОРАТОРНА РОБОТА №12
Опитування дискретних органів керування та датчиків за допомогою мікроконтролера

Мета: вивчити можливості мікроконтролерів AVR з опитування дискретних об’єктів.

1. КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Засоби мови асемблера для опитування цифрових портів вводу-виводу

Конфігурація входів

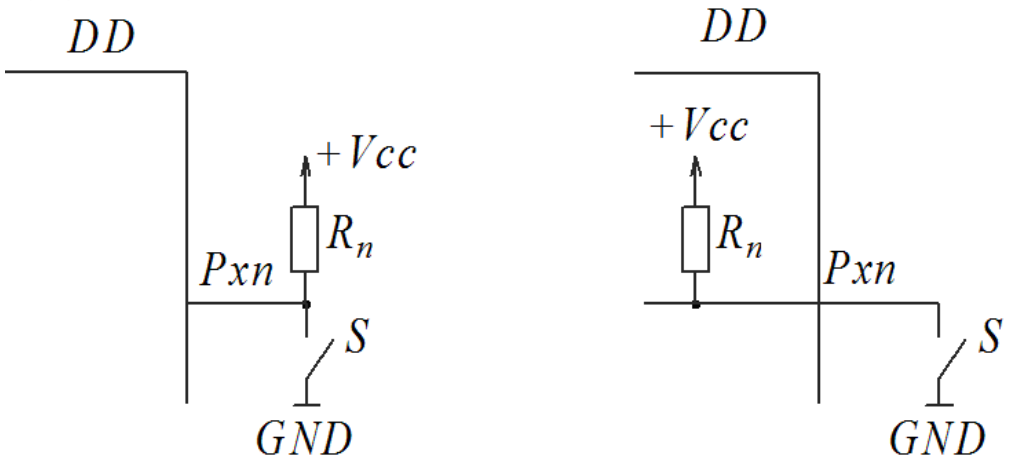
Звернення до портів здійснюється через регістри вводу/виводу. Під кожен порт в адресному просторі вводу/виводу зарезервовано по 3 адреси, за якими розміщені наступні регістри:

- регістр DDRx напряму даних;
- регістр PORTx даних порту;
- регістр PINx виводів порту.

Дійсні назви регістрів отримують підстановкою назви порту замість символу "x", відповідно регістри порту A називаються PORTA, DDRA, PINA, порту B - PORTB, DDRB, PINB і т. д. Оскільки за допомогою регістрів PINx здійснюється доступ до фізичних значень сигналів на виводах порту, вони доступні тільки для читання, тоді як інші два регістри доступні і для читання, і для запису.

Якщо до цифрового порту вводу-виводу мікроконтролера підключений дискретний датчик або кнопка, то для опитування датчика (кнопки) такий вивід порту треба конфігурувати як "вхід". Для цього у відповідний розряд регістра напряму даних DDRx треба записати лог. "0". Оскільки при запуску мікроконтролера в усі розряди регістрів вводу-виводу записуються лог. "0", то за замовчанням усі виводи мікроконтролера є входами.

При натисненні і відпусканні кнопки, яка підключена до виводу мікроконтролера, потенціал цього виводу має бути рівний або потенціалу загального виводу джерела живлення GND або потенціалу позитивного виводу джерела живлення +Vcc. Досягти цього можна шляхом використання підтягуючого резистора R_n, як показано на рис. 12.1, а. Внутрішній опір виходу мікроконтролера в режимі "вхід" оцінюється десятками кОм, опір підтягуючого резистора також вибирається досить великим (до 100 кОм), тому при відпусканні кнопки струм, що протікає через R_n і виведення микроконтролера несуттєво малий і потенціал виводу відповідає потенціалу позитивного виводу джерела живлення +Vcc. При натисненні кнопки потенціал виводу стає тотожним потенціалу загального виводу джерела живлення GND.



а) б)

Рисунок 12.1 - Підключення кнопки до виводу мікроконтролера

З метою зменшення кількості елементів друкованої плати в мікроконтролерах AVR є вбудовані підтягуючи резистори (рис. 12.1, б).

Для використання вбудованого підтягуючого резистора для виводу мікроконтролера, який конфігурований як "вхід", необхідно у відповідних розряд регістра PORTx записати лог. "1".

Наприклад, якщо між виводам PD1 і PD4 контролера і GND підключені кнопки S1 і S2 відповідно, то для використання внутрішніх підтягуючих резисторів необхідно виконати команди (використання будь-якого варіанту приводять до однакового результату):

варіант 1

```
sbi PORTD, PD1
sbi PORTD, PD4
```

варіант 2

```
ldi r16,(1<<PD1)+(1<<PD4)
out PORTD, r16
```

Символ << забезпечує зсув виразу зліва на число біт, що наведено праворуч. Символ «+» (можна використовувати « | ») визначає операцію АБО.

варіант 3

```
ldi r16,0b00010010
out PORTD, r16
```

Замість двійкового числа `0b00010010` можна використовувати десятковий (`18`) або шістнадцятковий (`0x12`) еквівалент.

Опитування входів

Інформація про фізичний рівень сигналу на виводах порту відображується в розрядах регістра виводів порту PINx.

Для перевірки стану кнопок можна використовувати наступні команди:

`sbic A, b -` перевіряє стан розряду b регістра вводу/виводу A. Якщо розряд скинутий, команда, що слідує за "sbic A, b", пропускається.

sbis A, b - перевіряє стан розряду b регістра вводу/виводу A. Якщо розряд встановлений, команда, що слідує за "sbis A, b", пропускається.

У разі натиснутого стану кнопки на схемі (рис. 1) вивід мікроконтролера буде притягнуто до "землі", на виводі буде сигнал НИЗЬКОГО рівня, отже у відповідному розряді регістра PINx буде записаний "0". Якщо кнопка буде відпущена, то через внутрішній підтягуючий резистор на вихід подаватиметься сигнал ВИСОКОГО рівня, отже у відповідному розряді регістра PINx буде записана лог. "1":

кнопка натиснута => з розряду регістра PINx зчитуємо лог. "0"

кнопка відпущена => з розряду регістра PINx зчитуємо лог. "1"

Наприклад, для зчитування стану кнопки S1, підключеної до виводу PD1, допустимі наступні команди:

варіант 1

```
     sbic PIND,PD1              ;пропуск наступної команди, якщо розряд PD1
                                ;регістра PIND скинуто (PIND.PD1=0), тобто
                                ;якщо кнопка натиснута
     rjmp  B_UP                 ;ця команда виконується, якщо PIND.PD1=1,
                                ;тобто якщо кнопка відпущена
          <код, якщо кнопка натиснути>   ;розряд скинуто
     rjmp L1
B_UP:
          <код, якщо кнопка відпущена>   ;розряд встановлено
L1:
```

варіант 2

```
     sbis PIND,PD1              ;пропуск наступної команди, якщо розряд PD1
                                ;регістра PIND встановлено(PIND.PD1=1),
                                ;тобто якщо кнопка відпущена
     rjmp B_DN                  ;ця команда виконується, якщо PIND.PD1=0,
                                ;тобто якщо кнопка натиснута
          <код, якщо кнопка відпущена>  ;розряд встановлено
     rjmp L1
B_DN:
          <код, якщо кнопка натиснути>  ;розряд скинуто
L1:
```

1.2. Команди мови C for Arduino для роботи з цифровими портами вводу-виводу

pinMode(pin, mode)	Встановлює **режим роботи** виводу з номером pin плати як входу (mode=INPUT) або як виходу (mode=OUTPUT) .

digitalWrite(pin, value)	**Подає напругу** ВИСОКОГО (value=HIGH) або НИЗЬКОГО (value=LOW) рівня на вивід плати з номером pin. Якщо вивід плати з номером pin був встановлений функцією pinMode як вихід (mode=OUTPUT), то при value=HIGH на відповідному виводі плати буде встановлена напруга +5В, а при value=LOW 0В (GND). Якщо вивід плати з номером pin був встановлений функцією pinMode як вхід (mode=INPUT), то функція digitalWrite зі значенням value=HIGH буде підключити до вивода внутрішній підтягуючий резистор 20 кОм, подача value=LOW відключає цей резистор.
digitalRead(pin)	Функція **зчитує** значення з заданого виводу плати pin. Повертає значення HIGH або LOW.

2. ОПИС ОБЛАДНАННЯ, ЩО ВИКОРИСТОВУЄТЬСЯ

2.1. Експериментальний стенд

Використовується експериментальний стенд, що був описаний в методичних вказівках до попередньої лабораторної роботи.

2.2. Модель експериментального стенда в Proteus

Використовується модель лабораторного стенда в Proteus, що знаходиться у файлі LAB3_MPT.pdsprj (рис. 12.2). Опис даної моделі наведено у методичних вказівках до попередньої лабораторної роботи.

3. ЗАВДАННЯ

Написати програму для контролера плати Arduino UNO, яка функціонує наступним чином. При натисканні кнопки S1 ввімкнути лампу розжарювання на 3с. При натисканні кнопки S2 ввімкнути світлодіод VD1 на 2с, після чого включити вентилятор на 4с.

Програму необхідно написати на мовах асемблера та C for Arduino, відлагодити на моделі та реальному стенді.

Плата Arduino Uno підключається до експериментального стенду згідно табл. 12.1.

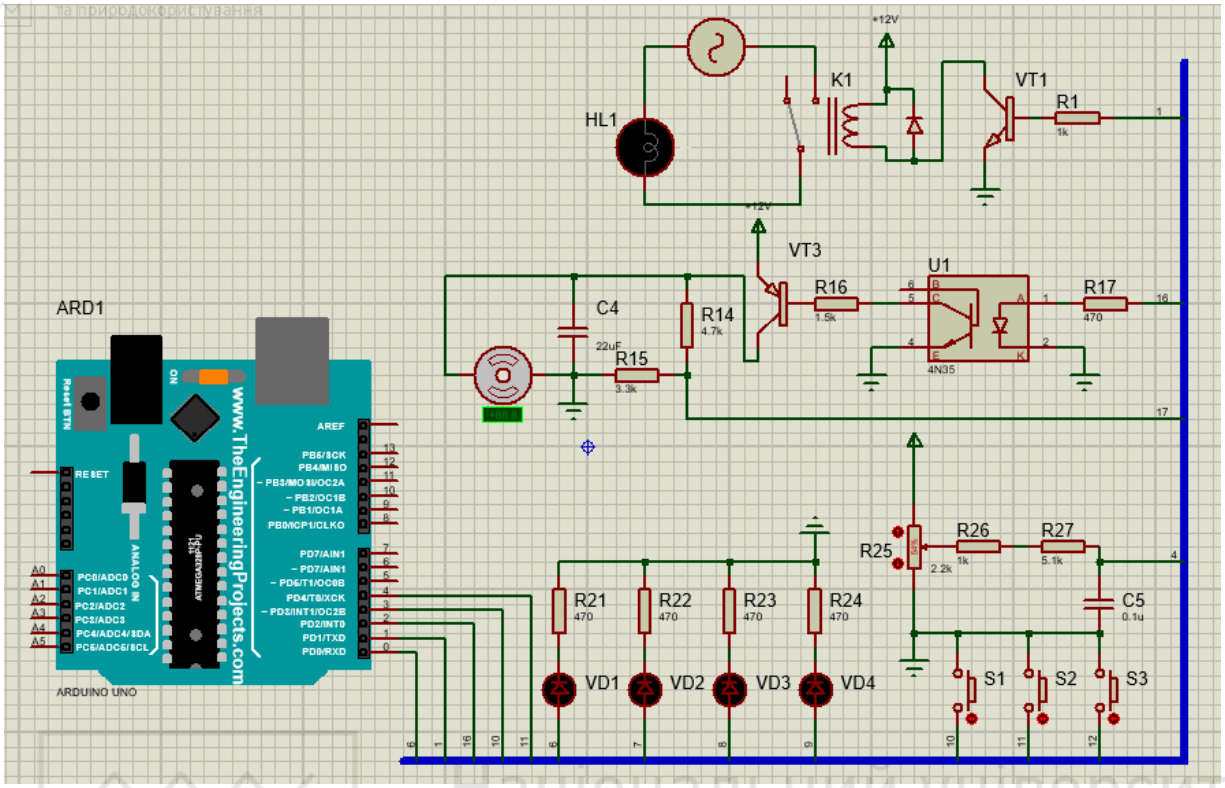


Рисунок 12.2 - Модель експериментального стенда в Proteus

Таблиця 12.1

Підключення стенда до Arduino Uno для виконання лабораторної роботи

Об’єкт	№ вихідного контакту стенда	Плата Arduino Uno	
		№ порта плати Arduino Uno	порт мікроконтролера ATMega328p
світлодіод VD1	6	0	PD0
лампа розжарювання	1	1	PD1
вентилятор	16	2	PD2
кнопка S1	10	3	PD3
кнопка S2	11	4	PD4

4. ПОРЯДОК ВИКОНАННЯ РОБОТИ

4.1. Написання програми на мові асемблера

4.1.1 Відкрити програму Atmel Studio, створити новий проект на мові асемблера. При цьому обрати тип мікроконтролера, який встановлено в платі Arduino Uno, а саме ATMega328p

4.1.2 Набрати наступну програму:

```
.INCLUDE "m328pdef.inc"
      ldi r16,high(RAMEND)     ;ініціалізація стека
      out SPH,r16
      ldi r16,low(RAMEND)
      out SPL,r16
                               ;конфігурування портів
                               ;виходи: PD0 - світлодіод VD1,
                               ;PD1 - лампа, PD2 - вентилятор
      ldi r16,(1<<PD0)+(1<<PD1)+(1<<PD2)
      out DDRD,r16
                               ;входи: PD3 - кнопка S1,
                               ;       PD4 - кнопка S2
                               ;підключити до входів
                               ;підтяг. резистори
      ldi r16,(1<<PD3)+(1<<PD4)
      out PORTD,r16

Lstart:                        ;основний цикл програми

      sbic PIND,PD3            ;перевірити стан кнопки S1
      rjmp B_UP1               ;якщо розряд встановлено
                               ;(кнопка відпущена),
                               ;то йти на мітку B_UP1

      sbi PORTD,PD1            ;включити лампу

      ldi XL,low(2000)         ;тривалість затримки в мс
      ldi XH,high(2000)
      rcall DELAY              ;виклик підпрограми затримки

      cbi PORTD,PD1            ;вимкнути лампу

 B_UP1:

      sbic PIND,PD4            ;перевірити стан кнопки S2
      rjmp Lstart              ;якщо розряд встановлено
                               ;(кнопка відпущена),
                               ;то йти на мітку Lstart

      sbi PORTD,PD0            ;включити VD1

      ldi XL,low(2000)         ;тривалість затримки в мс
      ldi XH,high(2000)
      rcall DELAY              ;виклик підпрограми затримки
```

```
        cbi PORTD,PD0           ;вимкнути VD1

        sbi PORTD,PD2           ;включити вентилятор

        ldi XL,low(2000)        ;тривалість затримки в мс
        ldi XH,high(2000)
        rcall DELAY             ;виклик підпрограми затримки

        cbi PORTD,PD2           ;вимкнути вентилятор

        rjmp Lstart             ;йти до початку основного циклу

DELAY:                          ;підпрограма затримки
                                ;вхід: регістрова пара X -
                                ;тривалість затримки в мс
        ldi YL,low(3999)
        ldi YH,high(3999)
Ld1:     sbiw YL,1
         brne Ld1
        sbiw XL,1
        brne DELAY
        ret
```

4.1.3 Виконати компіляцію програми.

4.2. Завантажити програму на мові асемблера до моделі експериментального стенда. Запустити модель стенда. Відслідкувати функціонування стенда. Зберегти Print Screen екрана з працюючою моделлю стенда для звіту.

4.3. Завантажити програму на мові асемблера до плати. Зібрати схему досліду, показати її викладачеві. Підключити плату Arduino до стенда та відслідкувати функціонування.

4.4. Написання програми на мові C for Arduino

4.4.1 Запустити програму IDE Arduino.

4.4.2 Набрати наступну програму:

```
void setup()
{    pinMode(0,OUTPUT);     //світлодіод VD1
     pinMode(1,OUTPUT);     //лампа розжарювання
     pinMode(2,OUTPUT);     //вентилятор
     pinMode(3,INPUT);      //кнопка S1
     digitalWrite(3,HIGH); //підкл. підтяг. резистор до 3 виходу
     pinMode(4,INPUT);      //кнопка S2
     digitalWrite(4,HIGH); //підкл. підтяг. резистор до 4 виходу
}

void loop()
{
     if (digitalRead(3)==LOW)  //якщо натиснута кнопка S1
     {
       digitalWrite(1,HIGH);   //вкл лампу
```

```
        delay(3000);              //затримка 3с
        digitalWrite(1,LOW);      //вимкнути лампу
 }

     if (digitalRead(4)==LOW)   //якщо натиснута кнопка S2
 {
     digitalWrite(0,HIGH);      //вкл світлодіод VD1
     delay(2000);               //затримка 2с
     digitalWrite(0,LOW);       //вимкнути світлодіод VD1

     digitalWrite(2,HIGH);      //вкл вентилятор
     delay(4000);               //затримка 4с
     digitalWrite(2,LOW);       //вимкнути вентилятор
 }
}
```

4.4.3 Виконати компіляцію скетча

4.4.4 Завантажити програму на мові C for Arduino до моделі експериментального стенда. Відслідкувати функціонування стенда.

4.4.5 Завантажити програму на мові C for Arduino до плати Arduino. Відслідкувати функціонування експериментального стенда.

ВМІСТ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

1. Тема, мета роботи.
2. Блок-схема алгоритму функціонування мікроконтролера (складається самостійно) з поясненнями.
3. Повний текст програми на мові асемблера та вікно Atmel Studio з програмою на мові асемблера.
4. Опис всіх команд, що використовуються в програмі на мові асемблера, з поясненнями їх синтаксису та призначення в програмі.
5. Програма на мові C for Arduino та вікно IDE Arduino з програмою.
6. Опис всіх команд, що використовуються в програмі на мові C for Arduino, з поясненнями їх синтаксису та призначення в програмі.
7. Модель лабораторного стенда в Proteus.
8. Висновки.

КОНТРОЛЬНІ ПИТАННЯ

1. Які регістри мікроконтролера AVR використовуються для опитування цифрових портів?
2. Як налаштувати вивід мікроконтролера для роботи в якості входу або виходу?
3. Які команди мови асемблера використовуються для опитування цифрових портів?
4. Які функції мови C for Arduino застосовуються для опитування цифрових портів?
5. Для чого використовується внутрішній підтягуючий резистор?

ЛАБОРАТОРНА РОБОТА №13
Виведення текстової інформації на рідкокристалічний індикатор

Мета: оволодіти навичками вивода літерно-цифрової інформації на рідкокристалічний індикатор, що підключений до мікроконтролера.

1. КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Контролер зі знакогенератором HD44780

Рідкокристалічний дисплей (РК-дисплей, РКД, *англ.* Liquid crystal display, LCD) – плоский дисплей, в якому використовується властивість рідких кристалів змінювати ступінь прозорості при зміні величини напруги, що прикладається.

Рідкокристалічний індикатор (РК-індикатор, РКІ) – сукупність РК-дисплея та спеціалізованого керуючого контролера, які конструктивно виконані в одному корпусі (рис. 13.1). РК-індикатори можуть бути символьними або графічними.

Розглянемо структурну схему символьного РК-індикатора WH1602A (рис. 13.1, а), який складається з:

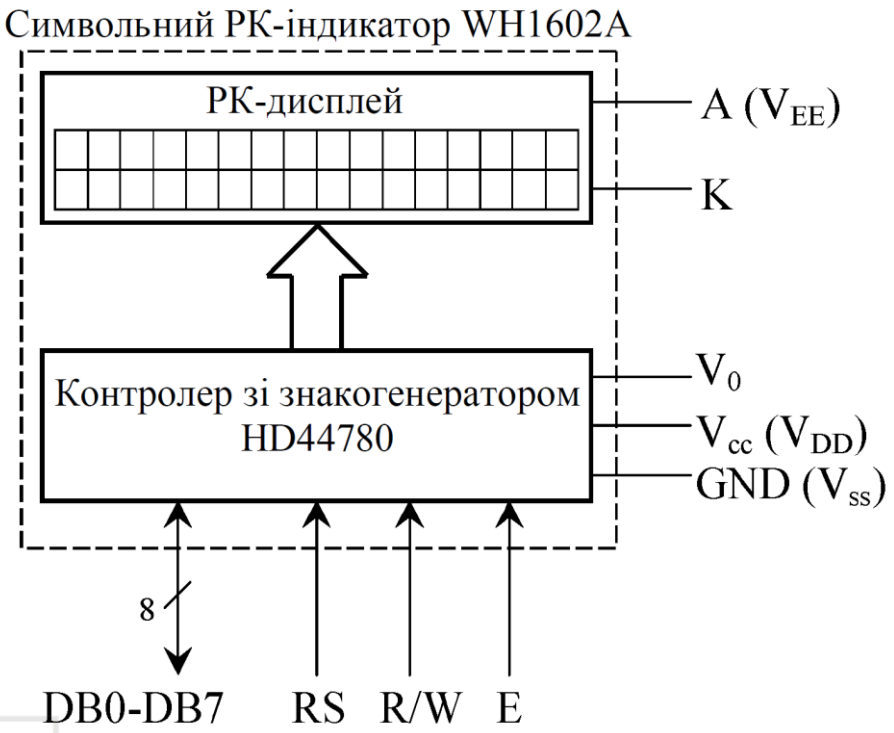
- *РК-дисплея,* побудованого на матриці з рідких кристалів, подаючи напругу на елемент якої ми можемо «засвітити» точку на екрані; матриця складається з 32х знакомісць (розміром 5х8 точок кожне), які розміщені у 2х рядках по 16 знакомісць (рис. 13.1, б);
- *контроллера зі знакогенератором* **HD44780** (аналог - KS0066 виробництва компанії Samsung), який фактично є стандартом на контролери чорно-білих рідкокристалічних знакосинтезирующих дисплеїв з паралельним 4 - або 8-бітовим інтерфейсом.

Призначення виводів даного РК-індикатора представлене в табл. 1.

Контролер HD44780 (рис. 13.2) складається з наступних основних елементів:

- DR - регістр даних;
- IR - регістр команд;
- DDRAM - відеопам'ять;
- CGRAM - ОЗП знакогенератора (таблиця символів користувача);
- CGROM - ПЗП знакогенератора (фіксована таблиця символів);
- AC - лічильник адреси.

а)



б)

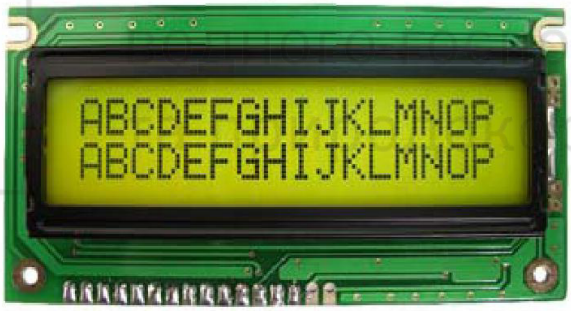


Рисунок 13.1 - Структурна схема символьного РК-індикатора WH1602A виробництва компанії HITACHI (*а*) та його загальний вигляд (*б*)

Таблиця 13.1

Призначення виводів РК-індикатора WH1602A

№ вивода	Назва	Функція
1	GND	Загальний (GND)
2	Vcc	Напруга живлення
3	V_0	Контрасність
4	RS	Команди / дані (1-приймання команд, 0 – даних)
5	R/W	Зчитування / запис (1 – зчиування, 0 - запис)
6	E	Строб (запис здійсн. за перепадом 1/0)
7-14	DB0-DB7	Шина команд / даних
15	A	Анод підсвітки
16	K	Катод підсвітки

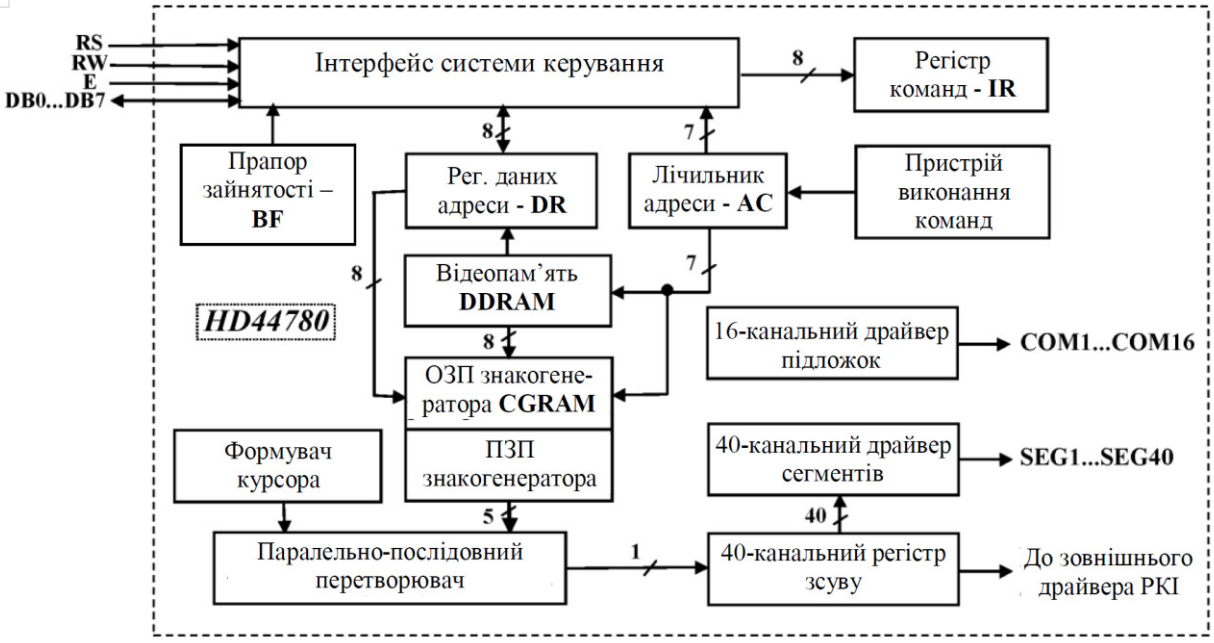


Рисунок 13.2 - Структурна схем контролера зі знакогенератором HD44780

Таблиця 13.2 - Коди символів контролера HD44780, який підтримує кирилицю

старша тетрада

молодша тетрада	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0	@	P	`	p			Б	Ю	ч	.	Д	Ц
1			!	1	A	Q	a	q			Г	Я	ш	ı	Ц	Ъ
2			"	2	B	R	b	r			Ё	б	ъ	ш	Щ	Ч
3			#	3	C	S	c	s			Ж	в	ы	!!	д	Ъ
4			$	4	D	T	d	t			Э	г	ь	˝	ф	Ш
5			%	5	E	U	e	u			И	ё	э	ж	ц	¯
6			&	6	F	V	f	v			Й	ж	ю	ˇ	щ	Ы
7			'	7	G	W	g	w			Л	з	я	I	´	Ь
8			(	8	H	X	h	x			П	и	«	Ш	¨	Э
9			)	9	I	Y	i	y			У	й	»	↑	~	Ю
A			*	:	J	Z	j	z			Ф	к	„	↓	é	Я
B			+	;	K	[	k	10			Ч	л	"	Ж	ç	‡
C			,	<	L	¢	l	12			Ш	м	№	И	ü	Ж
D			–	=	M	]	m	15			Ъ	н	¿	Ж	♣	ß
E			.	>	N	^	n	↵			Ы	п	ƒ	˃	∴	Ц
F			/	?	O	_	o	←			Э	т	€	·	o	■

Відеопам'ять DDRAM складається з однобайтних комірок, в яких знаходяться коди символів, що відображуються на екрані. Відповідність між кодами і изображеним символів встановлює спеціальна таблиця символів (табл. 13.2), вміст якої записаний в ПЗП знакогенератора CGROM. Вміст ПЗП знакогенератора CGROM не можна змінити. Наприклад, якщо необхідно відобразити на екрані цифру 1, то у відеопам'яті має бути записаний код цієї цифри згідно табл. 13.2, тобто $31. Для вибору позиції запису символу існує віртуальний курсор (номер поточного елементу пам'яті, AC), яким можна управляти за допомогою команд. Курсор також можна зробити видимим. За замовчуванням при записі символу в комірку курсор ззувається вперед на одну позицію. Розмір відеопам'яті DDRAM перевищує розмір видимої області екрану: DDRAM містить 80 комірок, згрупованих в 2 рядки, тобто по 40 байт в кожному рядку. В той же час дисплей РК-індикатора WH1602A містить по 16 знакомісць в кожному рядку. Пересуваючи дисплей над відеопам'яттю, можна відображати різні ділянки DDRAM (рис. 13.3).

Дисплей 2x16

Комірки відеопам'яті DDRAM 2x40

$00	$01	$02	. . .	$0F	$10	$11	. . .	$26	$27
$40	$41	$42	. . .	$4F	$50	$51	. . .	$66	$67

Рисунок 13.3 - Відображення вмісту відеопам'яті DDRAM на дисплеї

На відміну від CGROM, ОЗП знакогенератора CGRAM можна змінювати програмно, створюючи нові символи. Розмір CGRAM складає 64 байти, для кодування одного символу потрібно 8 байт, тобто ОЗП знакогенератора може одночасно зберігати 8 нових символів. Окрім пам'яті даних, яка представлена відеопам'яттю і пам'яттю (ОЗП і ПЗП) знакогенератора, у контролера HD44780 є регістр команд IR, в який передається байт команд. Формати команд описані в табл. 3, розшифровка прапорів - в табл. 13.4.

Таблиця 13.3

Система команд контролера зі знакогенератором HD44780

DB7	**DB6**	**DB5**	**DB4**	**DB3**	**DB2**	**DB1**	**DB0**	Значення
0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	Очистити екран. Лічильник адреси на 0 позицію DDRAM
0	**0**	**0**	**0**	**0**	**0**	**1**	-	Адресація на DDRAM, скидання зсувів, лічильник адреси на 0 (початок рядка адресується на початку DDRAM)
0	**0**	**0**	**0**	**0**	**1**	I/D	S	Налаштування зсуву екрана та курсора
0	**0**	**0**	**0**	**1**	D	C	B	Налаштування режима відображення
0	**0**	**0**	**1**	S/C	R/L	-	-	Зсув курсора або екрана
0	**0**	**1**	D/L	N	F	-	-	Вибір кількості рядків, ширини шини та розміру символа
0	**1**	AG	AG	AG	AG	AG	AG	Переключити адресацію на CGRAM та задати адресу в CGRAM
1	AD	AD	AD	AD	AD	AD	AD	Переключити адресацію на DDRAM та задати адресу в DDRAM

Таблиця 13.4

Розшифровка прапорів в таблиці команд контролера HD44780

Опис команд	Прапор	Функції, якщо	
		прапор скинуто (0)	прапор встановлено (1)
Налаштування зсуву екрана і курсора	**I/D**	Декремент лічильника адреси	Інкремент лічильника адреси
	S	При додаванні нового символа екран не зсувається	Екран зсувається відносно вмісту DDRAM при додаванні нового символа. Напрямок зсуву визначається прапором I/D (I/D=0 – вправо, I/D=1 - вліво)
Налаштування режима відображення	**D**	Відключити дисплей	Включити дисплей
	C	Курсор відсутній	Курсор у вигляді прочерка
	B	Курсор відсутній	Курсор в вигляді чорного квадрат
Зсув курсора або екрана	**S/C**	Зсув курсора (напрямок визначається прапором R/L)	Зсув екрана (напрямок визначається прапором R/L)
	R/L	Напрямок зсуву - вліво	Напрямок зсуву - вправо
Вибір числа рядків, ширини шини та розміру символа	**D/L**	4-розрядна шина	8-розрядна шина
	N	Один рядок	Два рядки
	F	Символ 5x8 точок	Символ 5x10 точок (використовується рідко)
Переключити адресацію на SGRAM та задати адресу в SGRAM	**AG**	Адреса в пам’яті CGRAM	
Переключити адресацію на DDRAM і задати адресу в DDRAM	**AD**	Адреса в пам’яті DDRAM	

Байти команд і байти даних передаються по одній шині DB0 - DB7 (рис. 13.1, а). При передачі байта команди на виводі RS має бути встановлений сигнал високого рівня, а при передачі байта даних необхідно забезпечити RS=0. Запис даних або команди в регістр даних або команд (при R/W=0) здійснюється по спадаючому фронту сигналу на вході стробування, тобто при зміні сигналу на вході E від логічної "1" до "0". Пам'ять даних (DDRAM і CGRAM) також можна читати. Для цього необхідно визначити адресу пам'яті шляхом подачі відповідної команди і уставновить R/W=1.

Контроллер HD44780 може функціонувати в режимі 4х розрядної (D/L=0) або 8-розрядної (D/L=1) шини. При використанні 4х розрядної шини задіяні тільки виводи **DB4 - DB7** і байт даних/команди передається по тетрадах: **спочатку - старша тетрада, потім - молодша**. При використанні 8-розрядної шини байт передається одночасно.

Послідовність команд при записі байта в елемент пам'яті, визначений лічильником адреси AC, для 8-розрядної шини виглядає таким чином:

1. Встановити R/W=0 (операція запису).

2. Встановити RS=1 для передачі байта команди або RS=0 для передачі байта даних.

3. Встановити строб E=1 (підготовка до запису).

4. Вивести байт на шину DB0 - DB7.

5. Скинути строб E=0 (здійснити запис).

6. Витримати паузу.

7. Встановити "1" на усіх розрядах шини DB0 - DB7.

Послідовність команд при записі байта в елемент пам'яті, визначений лічильником адреси AC, для 4-розрядної шини наступна:

1. Встановити RS=1 для передачі байта команди або RS=0 для передачі байта даних.

2. Встановити RS=1 для передачі байта команди або RS=0 для передачі байта даних.

3. Встановити строб E=1 (підготовка до запису).

4. Вивести значення старшої тетради байта даних на шині DB4 - DB7.

5. Скинути строб E=0 (здійснити запис).

6. Витримати паузу.

7. Встановити строб E=1 (підготовка до запису).

8. Вивести значення молодшої тетради байта даних на шині DB4 - DB7.

9. Скинути строб E=0 (здійснити запис).

10. Витримати паузу.

11. Встановити "1" на усіх розрядах шини DB4 - DB7.

На обробку кожної команди або на запис отриманих даних контроллеру необхідно певний час, яке звичайно складає близько 40 мкс, але може доходити і до одиниць мілісекунд. Тому в п. 6 приведеного алгоритму витримується пауза. Також про готовність контролера виконувати наступну команду свідчить прапор зайнятості BF, значення якого можна рахувати із старшого розряду регістра IR (читання регістра IR забезпечується при RS=1). Якщо BF=1, то контролер зайнятий, а при BF=0 контролер вільний.

1.2. Підключення LCD дисплея до плати Arduino UNO

Під□єднання дисплея до плати Arduino необхідно виконувати у відповідності до табл. 13.5. Для керування дисплеєм з програми на мові C for Arduino існують бібліотеки LiquidCrystal (або LiquidCrystalRus – підтримує кирилицю) або 4 Bit Arduino LCD.

!!! Для використання бібліотеки **LiquidCrystalRus** необхідно папку LiquidCrystalRus скопіювати в …\arduino-1.6.11\libraries\.

Таблиця 13.5

Підключення LCD дисплея до плати Arduino UNO

Контакт LCD	D4	D5	D6	D7	RS	E	+5	RW	V0	GND
Контакт ARDUINO	4	5	6	7	8	9	5 V	GND	GND	GND

2. ОПИС ОБЛАДНАННЯ, ЩО ВИКОРИСТОВУЄТЬСЯ

2.1. Плата Arduino UNO з LCD дисплеєм зображена на рис. 13.2.

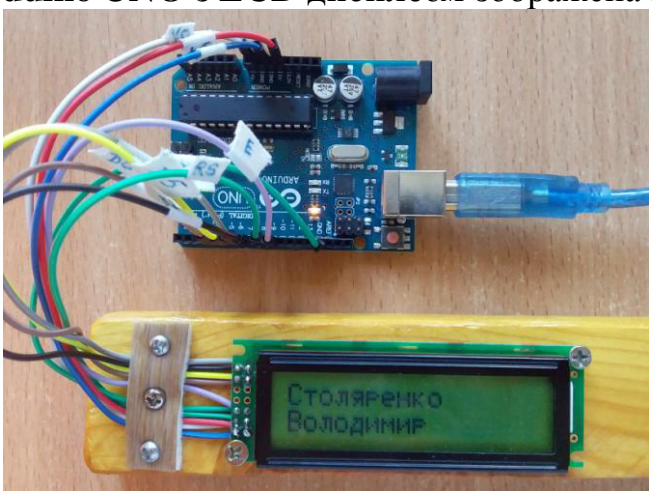


Рисунок 13.2 – Під□єднання дисплея до плати Arduino

2.2. Модель плати Arduino UNO з LCD дисплеєм в Proteus

Модель, що створена у файлі LAB4_MPT.pdsprj, наведена на рис. 13.3.

3. ЗАВДАННЯ

Написати програму для контролера плати Arduino UNO, яка виводить на LCD дисплей два повідомлення, які відображуються почерзі.

Програму необхідно написати на мовах асемблера та C for Arduino, відлагодити на моделі та реальному стенді.

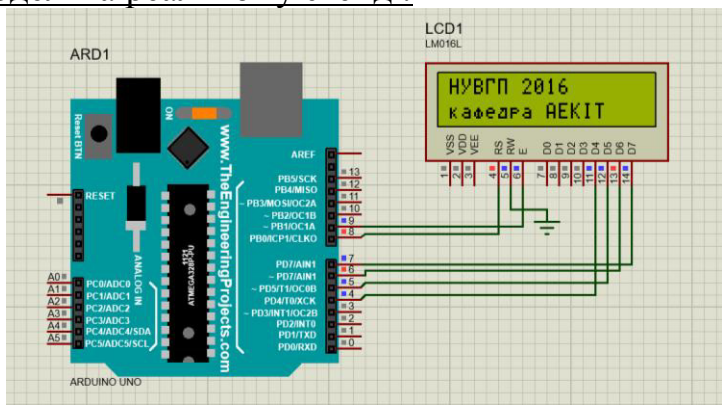


Рисунок 13.3 - Модель плати Arduino UNO з LCD дисплеєм в Proteus

4. ПОРЯДОК ВИКОНАННЯ РОБОТИ

4.1. Написання програми на мові асемблера

4.1.1 Відкрити програму Atmel Studio, створити новий проект на мові асемблера. При цьому обрати тип мікроконтролера, який встановлено в платі Arduino Uno, а саме ATMega328p.

4.1.2 Закодувати такстові повідомлення згідно табл. 13.2. Приклад наведено у табл. 13.5.

Таблиця 13.5

Коди символів текстових повідомлень, що виводяться на дисплей

Екран №1

Адреса DDRAM	$00	$01	$02	$03	$04	$05	$06	$07	$08	$09	$0A	$0B	$0C	$0D	$0E	$0F
Команда*	$80	$81	$82	$83	$84	$85	$86	$87	$88	$89	$8A	$8B	$8C	$8D	$8E	$8F
Row1Scr1	=	Н	У	В	Г	П	=		-	2	0	1	6	-		
Код символа	0x3D	0x48	0xA9	0x42	0xA1	0xA8	0x3D	0x10	0x2D	0x32	0x30	0x31	0x39	0x2D	0x10	0x10
Адреса DDRAM	$40	$41	$42	$43	$44	$45	$46	$47	$48	$49	$4A	$4B	$4C	$4D	$4E	$4F
Команда*	$C0	$C1	$C2	$C3	$C4	$C5	$C6	$C7	$C8	$C9	$CA	$CB	$CC	$CD	$CE	$CF
Row2Scr1	К	А	Ф	Е	Д	Р	А				А	Е	К	І	Т	
Код символа	0x4B	0x41	0xAA	0x45	0xE0	0x50	0x41	0x10	0x10	0x10	0x41	0x45	0x4B	0x49	0x54	0x10

Екран №2

Команда*	$80	$81	$82	$83	$84	$85	$86	$87	$88	$89	$8A	$8B	$8C	$8D	$8E	$8F
Row1Scr2	С	т	о	л	я	р	е	н	к	о						
Код символа	0x43	0xBF	0x6F	0xBB	0xC7	0x70	0x65	0xBD	0xBA	0x6F	0x10	0x10	0x10	0x10	0x10	0x10
Команда*	$C0	$C1	$C2	$C3	$C4	$C5	$C6	$C7	$C8	$C9	$CA	$CB	$CC	$CD	$CE	$CF
Row2Scr2	В	о	л	о	д	и	м	и	р							
Код символа	0x42	0x6F	0xBB	0x6F	0xE3	0xB8	0xBC	0xB8	0x70	0x10	0x10	0x10	0x10	0x10	0x10	0x10

*Команда формується наступним чином: адреса DDRAM + $80 (див. формат команди «Переключити адресацію на DDRAM та задати адресу в DDRAM» в табл. 3)

4.1.3 Набрати наступну програму:

```
.INCLUDE "m328pdef.inc"

.CSEG                           ;ПОЧАТОК ПРОГРАМНОГО СЕГМЕНТА
        ldi r16,high(RAMEND)    ;ініціалізація стека
        out SPH,r16
        ldi r16,low(RAMEND)
        out SPL,r16

        ldi r16,0xF0            ;конфігурування портів
        out DDRD,r16            ;PD4-PD7 - виходи
        ldi r16,(1<<PB0)+(1<<PB1)+(1<<PB3)
        out DDRB,r16            ;PB0, PB1 - виходи

        cbi PORTB,PB3

        rcall INIT              ;ініціалізація дисплея

;================ ОСНОВНА ПРОГРАМА ===============

Lstart:

        ldi r20,0x80            ;відобразити Row1Scr1 в 1 рядку LCD
        ldi ZL,low(Row1Scr1*2)
        ldi ZH,high(Row1Scr1*2)
        rcall LINE

        ldi r20,0xC0            ;відобразити Row2Scr1 в 2 рядку LCD
        ldi ZL,low(Row2Scr1*2)
        ldi ZH,high(Row2Scr1*2)
        rcall LINE

        ldi XL,low(3000)        ;тривалість затримки в мс
        ldi XH,high(3000)
        rcall DELAY

        ldi r20,0x80            ;відобразити Row1Scr2 в 1 рядку LCD
        ldi ZL,low(Row1Scr2*2)
        ldi ZH,high(Row1Scr2*2)
        rcall LINE

        ldi r20,0xC0            ;відобразити Row2Scr2 во 2 рядку LCD
        ldi ZL,low(Row2Scr2*2)
        ldi ZH,high(Row2Scr2*2)
        rcall LINE

        ldi XL,low(3000)        ;тривалість затримки в мс
        ldi XH,high(3000)
        rcall DELAY

        rjmp Lstart

;============ ПІДПРОГРАМИ ==================

                                ;ПІДПРОГРАМА ІНІЦІАЛІЗАЦІЇ LCD
INIT:   cbi PORTB,PB0           ;RS=0 - подаються команди
                                ;перевод LCD в режим 4х розрядної шини
        sbi PORTB,PB1           ;E=1 - встановити строб-сигнал

        ldi r16,0b00100000
        out PORTD,r16
        cbi PORTB,PB1           ;E=0 - скинути строб-сигнал (записати в LCD)

        ldi XL,low(4)           ;тривалість затримки в мс
        ldi XH,high(4)
```

```
        rcall DELAY

        ldi r20,0b00101000          ;а) DL=0 (4-розрядна шина)
        rcall WRITE                 ;   N=1 (2 рядка)
                                    ;   F=0 (знакомісце 5x8)
                                    ;r20 - байт команди/даних

        ldi r20,0b00000110          ;б) I/D=1 (інкремент адреси)
        rcall WRITE                 ;   S=0(екран не пересувається)

                                    ;в) D=1 (включити дисплей),
        ldi r20,0b00001100          ;   C=0 (нема курсора у вигляді прочерка)
        rcall WRITE                 ;   B=0 (нема курсора у вигляді квадрата)

        ldi r20,0b00000001          ;г) очистка екрана, лічильник адреси на
        rcall WRITE                 ;   нульову позицію DDRAM
        ret


                                    ;ПІДПРОГРАМА ЗАПИСУ БАЙТА З r20 В ПОРТ LCD
WRITE:                              ;запис старшої тетради
        sbi PORTB,PB1               ;E=1 - встановити строб-сигнал
        out PORTD,r20
        cbi PORTB,PB1               ;E=0 - скинути строб-сигнал (записати в LCD)

        ldi XL,low(4)               ;тривалість затримки в мс
        ldi XH,high(4)
        rcall DELAY

                                    ;запис молодшої тетради
        sbi PORTB,PB1               ;E=1 - встановити строб-сигнал
        bst r20,3                   ;зберегти   3-й розряд r20 у прапорі T
        bld r16,7                   ;встановити 7-й розряд r16 згідно прапору T
        bst r20,2                   ;зберегти   2-й розряд r20 у прапорі T
        bld r16,6                   ;встановити 6-й розряд r16 згідно прапору T
        bst r20,1                   ;зберегти   1-й розряд r20 у прапорі T
        bld r16,5                   ;встановити 5-й розряд r16 згідно прапору T
        bst r20,0                   ;зберегти   0-й розряд r20 у прапорі T
        bld r16,4                   ;встановити 4-й розряд r16 згідно прапору T
        out PORTD,r16
        nop
        cbi PORTB,PB1               ;E=0 - скинути строб-сигнал (записати в LCD)

        ldi XL,low(4)               ;тривалість затримки в мс
        ldi XH,high(4)
        rcall DELAY

                                    ;встановити шину в "1"
        ldi r16,0xF0                ;встановити старшу тетраду
        out PORTD,r16
        ret


                                    ;ПІДПРОГРАМА ЗАТРИМКИ
DELAY:                              ;вхід: регістрова пара X –
                                    ;тривалість затримки в мс
      ldi YL,low(3999)
      ldi YH,high(3999)
Ld1:    sbiw YL,1
        brne Ld1
      sbiw XL,1
      brne DELAY
      ret

LINE:                               ;ПІДПРОГРАМА ЗАПИСУ РЯДКА В ПАМ'ЯТЬ LCD (DDRAM)
                                    ;Параметри: r20 - початкова адреса DDRAM LCD,
                                    ;               з якої починати писати рядок
                                    ; Z - мітка рядка символів, яку необх. вивести
```

```
                                ;Встановл. адреси DDRAM, з якої починати рядок
        cbi PORTB,PB0           ;RS=0 - подаються КОМАНДИ
                                ;відображати, починаючи з адреси r20 DDRAM
        rcall WRITE             ;підпрограма запису байта з r20 в порт LCD
                                ;посимвольний запис рядка в DDRAM
        sbi PORTB,PB0           ;RS=1 - подаються ДАНІ
                                ;в Z - адреса першого байта рядка

        ldi r19,16              ;лічильник символів (в рядку 16 символів)
L2:     lpm r20,Z+              ;завантажити з пам'яті прогр. (адреса - в Z)
                                ;в r20 код символа для вивода на порт
                                ;після цього Z=Z+1 (підготовка до читання
                                ;наступного символа)
        rcall WRITE             ;підпрограма запису байта з r20 в порт LCD
        dec r19                 ;декр. лічильн. символів (наступний символ)
        brne L2                 ;якщо символи ще не скінчилися,
                                ;то виводити наступний
        ret
                                ;КОДИ СИМВОЛІВ (записуються в пам'ять програм)
                                ;рядки по 16 байт
                                ;=НУВГП= -2016-
Row1Scr1:
.db 0x3D,0x48,0xA9,0x42,0xA1,0xA8,0x3D,0x10, 0x2D,0x32,0x30,0x31,0x36,0x2D,0x10,0x10
                                ;КАФЕДРА АЕКІТ
Row2Scr1:
.db 0x4B,0x41,0xAA,0x45,0xE0,0x50,0x41,0x10,0x10,0x10,0x41,0x45,0x4B,0x49,0x54,0x10
                                ;Столяренко
Row1Scr2:
.db 0x43,0xBF,0x6F,0xBB,0xC7,0x70,0x65,0xBD,0xBA,0x6F,0x10,0x10,0x10,0x10,0x10,0x10
                                ;Володимир
Row2Scr2:
.db 0x42,0x6F,0xBB,0x6F,0xE3,0xB8,0xBC,0xB8,0x70,0x10,0x10,0x10,0x10,0x10,0x10,0x10
```

4.1.3 Виконати компіляцію програми.

4.2. Завантажити програму на мові асемблера до моделі експериментального стенда. Запустити модель стенда. Відслідкувати функціонування стенда. Зберегти Print Screen екрана з працюючою моделлю стенда для звіту.

4.3. Завантажити програму на мові асемблера до плати.

4.4. Написання програми на мові C for Arduino

4.4.1 Запустити програму IDE Arduino.

4.4.2 Набрати наступну програму:

```
    // підключити бібліотеку
#include <LiquidCrystalRus.h>
    // вказати виводи дисплея:
    //8-RS, 9-E, 4-DB4, 5-DB5, 6-DB6, 7-DB7
LiquidCrystalRus lcd(8, 9, 4, 5, 6, 7);
void setup() {
  // встановити кількість символів в
  // рядку (16) та кількість рядків (2)
  lcd.begin(16, 2);
}
void loop() {
```

```
lcd.clear();
lcd.setCursor(0,0);
lcd.print("НУВГП 2016");
lcd.setCursor(0,1);
lcd.print("кафедра АЕКІТ");
delay(2000);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Шевченко");
lcd.setCursor(0,1);
lcd.print("Микола");
delay(2000); }
```

4.4.3 Виконати компіляцію скетча

4.4.4 Завантажити програму на мові C for Arduino до моделі експериментального стенда. Відслідкувати функціонування стенда.

4.4.5 Завантажити програму на мові C for Arduino до плати Arduino. Відслідкувати функціонування експериментального стенда.

ВМІСТ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

1. Тема, мета роботи.
2. Блок-схема алгоритму функціонування мікроконтролера (складається самостійно) з поясненнями.
3. Повний текст програми на мові асемблера та вікно Atmel Studio з програмою на мові асемблера.
4. Програма на мові C for Arduino та вікно IDE Arduino з програмою.
5. Модель лабораторного стенда в Proteus.
6. Висновки.

КОНТРОЛЬНІ ПИТАННЯ

1. Пояснити принцип дії рідкокристалічного індикатора.
2. Яка структура контролера зі знакогенератором HD44780?
3. Яким чином здійснюється кодування символів контролера HD44780?
4. Пояснити принцип дії відеопам’яті DDRAM.
5. Для чого призначений ПЗП знакогенератора CGROM? Чи можна змінити його вміст?
6. Які команди включає система команд контролера зі знакогенератором HD44780?
7. Які Вам відомі способи підключення контролера HD44780 до мікроконтролера?
8. Яка послідовність команд при обміні даними з контролером HD44780?

ЛАБОРАТОРНА РОБОТА №14
Введення аналогових сигналів в мікроконтролер

Мета: оволодіти методами роботи з аналого-цифровим перетворювачем мікроконтролерів сім'ї ATmega.

1. КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Модуль АЦП

Аналого-цифровий перетворювач (АЦП, англ. Analog-to-digital converter, ADC) - пристрій, що перетворює вхідний аналоговий сигнал в дискретний код (цифровий сигнал).

Розрядність АЦП характеризує кількість дискретних значень, які перетворювач може видати на виході. У двійкових АЦП вимірюється в бітах. Наприклад, двійковий 8-ми розрядний АЦП, здатний видати 256 дискретних значень (0…255), оскільки $2^8 = 256$.

До складу мікроконтролера ATmega328p входить 10-розрядний АЦП послідовного наближення. На вході АЦП розташовується 6-канальний аналоговий мультиплексор.

В процесі роботи АЦП може функціонувати в двох режимах:

- *режим поодинокого перетворення* - запуск кожного перетворення ініціюється користувачем;
- *режим безперервного перетворення* - запуск перетворень виконується безперервно через певні інтервали часу.

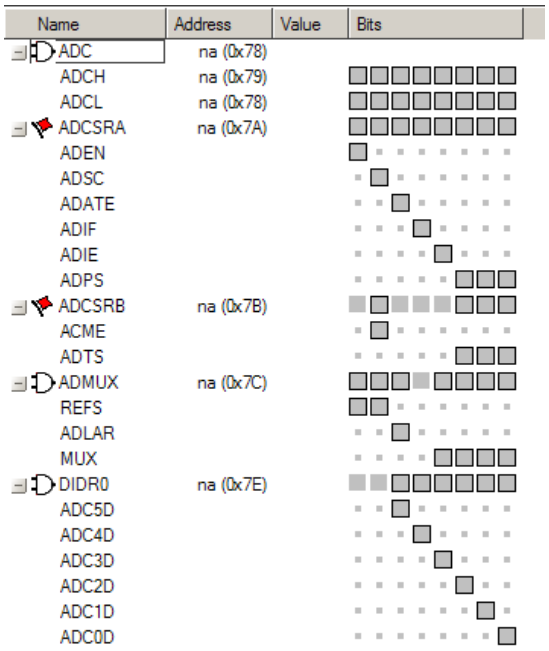


Рисунок 14.1 - Регістри вводу-виводу АЦП у складі мікроконтролера ATMega328p

Управління АЦП здійснюється за допомогою наступних регістрів вводу-виводу (рис. 14.1, 14.2):

- регістри управління і стану (Analog Digital Converter Status Register) АЦП **ADCSRA** та **ADCSRB**;
- регістр управління вхідним мультиплексором і додаткових функцій **ADMUX** (ADC Multiplexer Selection Register);
- регістрова пара **ADCH:ADCL**, що містить результ перетворення;
- регістр відключення цифрових входів **DIDR0** (Digital Input Disable Register 0), дозволяє відключати буфери цифрових портів виводів, що використовуються АЦП, для економії електроспоживання.

Розглянемо формат даних регістрів.

ADCSRA - регістр управління і стану АЦП							
7	6	5	4	3	2	1	0
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

Опис розрядів регістра ADCSRA:

Розряд		Опис розряду
7	ADEN	Дозвіл роботи АЦП (1 – ввімкнено, 0 - вимкнено)
6	ADSC	Запуск перетворення (1 – почати перетворення)
5	ADATE	Вибір режиму работи АЦП. Якщо ADATE=0, то АЦП запускається в режимі поодинокого перетворення, якщо ADATE=1, то функціонування АЦП визначається розрядами ADTS2:ADTS0 регістру ADCSRB
4	ADIF	Прапор переривання від АЦП
3	ADIE	Дозвіл переривання від АЦП
2	ADPS2	Завдання коефіцієнта ділення переддільника АЦП (вибір частоти перетворення)
1	ADPS1	
0	ADPS0	

ADCSRB - регістр управління і стану АЦП							
7	6	5	4	3	2	1	0
-	ACME	-	-	-	ADTS2	ADTS1	ADTS0

Опис розрядів регістра ADCSRB:

Розряд		Опис розряду
6	ACME	Розряд використовується аналоговим компаратором
2	ADTS2	Вибір джерела сигналу для запуску перетворення в режимі безперервного перетворення
1	ADTS1	
0	ADTS0	

ADMUX - регістр управління вхідним мультиплексором і додаткових функцій							
7	6	5	4	3	2	1	0
REFS1	REFS0	ADLAR	-	MUX3	MUX2	MUX1	MUX0

Опис розрядів регістра ADMUX:

Розряд		Опис розряду
7	REFS1	Вибір джерела опорної напруги
6	REFS0	
5	ADLAR	Вирівнювання результату перетворення (1 – по лівому краю, 0 – по правому краю)
4	-	
3	MUX3	Вибір вхідного каналу
2	MUX2	
1	MUX1	
0	MUX0	

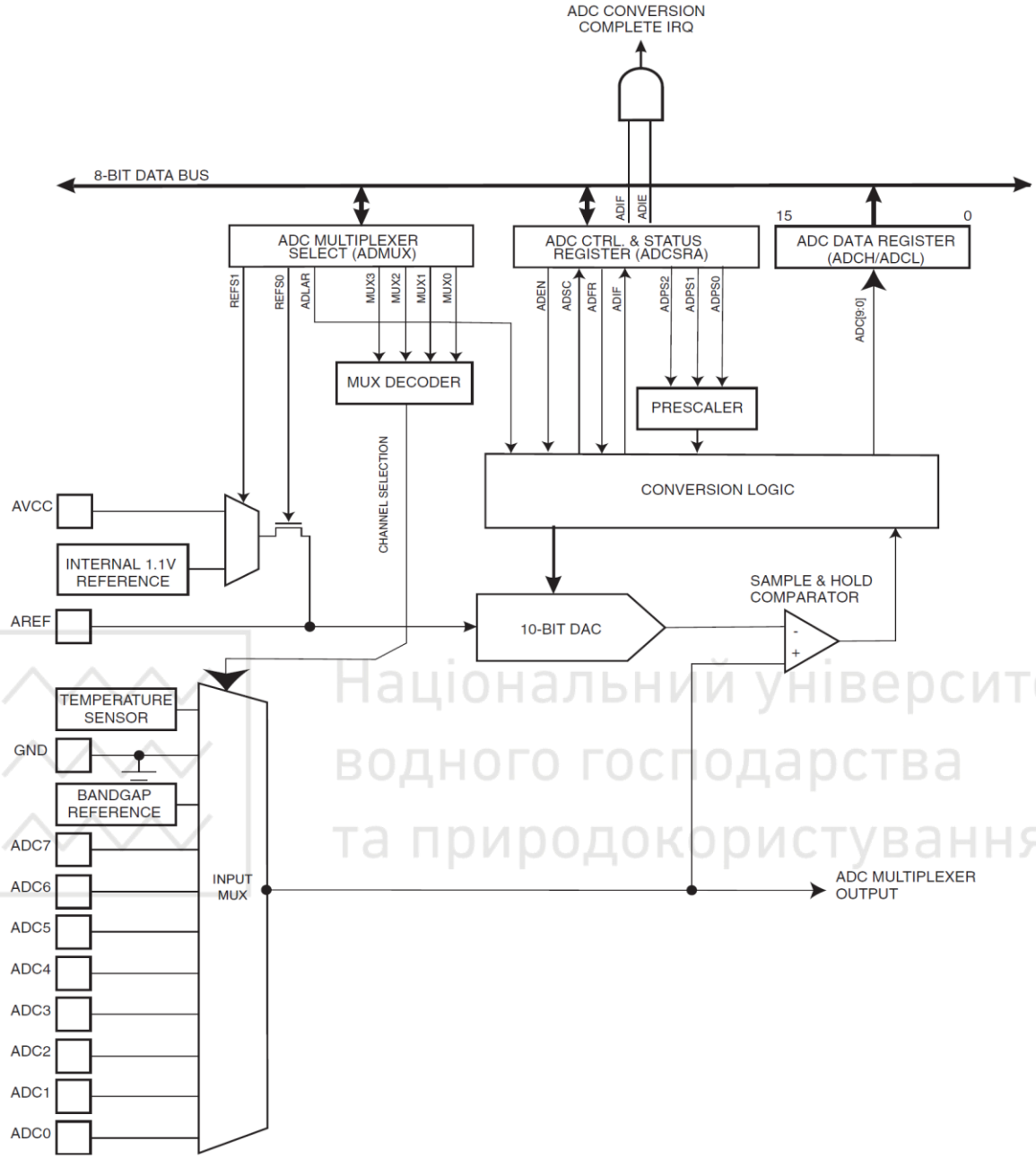


Рисунок 14.2 - Структурна схема модуля АПЦ у складі мікроконтролера ATMega328p

Для дозволу роботи АЦП необхідно записати лог. 1 в розряд ADEN регістра ADCSR, а для відключення - відповідно лог. 0.

Режим роботи АЦП визначається станом розряду ADATE. Якщо він встановлений в "1", АЦП працює в режимі безперервного перетворення. У цьому режимі запуск кожного наступного перетворення здійснюється автоматично після закінчення поточного. Якщо ж розряд ADATE скинутий в "0", АЦП працює в режимі поодинокого перетворення і запуск кожного перетворення здійснюється за командою користувача.

Запуск кожного перетворення в режимі поодинокого перетворення, а також запуск першого перетворення в режимі безперервного перетворення здійснюється установкою в "1" розряду ADSC регістра ADCSRA. Цикл

перетворення починається за першим наростаючим фронтом тактового сигналу після установки цього розряду. Через 13 тактів перетворення завершується, розряд ADSC апаратно скидається в "0" (у режимі поодинокого перетворення) і результат перетворення зберігається в регістрі даних АЦП. Одночасно встановлюється прапор переривання ADIF регістра ADCSRA і генерується запит на переривання. Як і прапори інших переривань, прапор ADIF скидається апаратно при запуску підпрограми обробки переривання від АЦП або програмно, записом в нього лог. 1. Дозвіл переривання здійснюється установкою в "1" розряду ADIE регістра ADCSRA при встановленому прапорі I регістра SREG.

Якщо АЦП працює в режимі безперервного перетворення, новий цикл почнеться відразу ж після запису результату. У режимі поодинокого перетворення нове перетворення може бути запущене відразу ж після скидання розряду ADSC (до збереження результату поточного перетворення).

Тактовим сигналом модуля АЦП є сигнал з переддільника, на вхід якого, у свою чергу, поступає тактовий сигнал мікроконтролера. Коефіцієнт поділу переддільника і, відповідно, тривалість перетворення визначається станом розрядів ADPS2..ADPS0 регістра ADCSRA (табл. 14.1).

Таблиця 14.1

Завдання коефіцієнта поділу переддільника АЦП

ADPS2	ADPS1	ADPS0	Коефіцієнт поділу
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Найбільша точність перетворення досягається, якщо тактова частота модуля АЦП знаходиться в діапазоні 50..200 кГц. Відповідно, коефіцієнт ділення переддільника рекомендується вибирати таким, щоб тактова частота модуля АЦП знаходилася у вказаному діапазоні.

Результат перетворення зберігається в регістрі даних АЦП. Оскільки АЦП 10-розрядне, цей регістр фізично розміщений в двох регістрах вводу/виводу, доступних тільки для читання: ADCH і ADCL. Звернення до цих регістрів (для отримання результату перетворення) повинне виконуватися в певній послідовності: спочатку необхідно прочитати регістр ADCL, а потім ADCH. Ця вимога пов'язана з тим, що після звернення до регістра ADCL процесор блокує доступ до регістрів даних з боку АЦП до тих пір, поки не буде прочитаний регістр ADCH. Завдяки цьому можна бути упевненим, що при читанні регістрів в них знаходитимуться складові одного і того ж результату. Відповідно, якщо чергове перетворення завершиться до звернення до регістра

ADCH, результат перетворення буде втрачений. З іншого боку, якщо для результату перетворення досить точності 8 розрядів, то для його отримання досить прочитати вміст регістра ADCH.

Управління вхідним мультиплексором модуля АЦП, а також додаткове управління модулем АЦП здійснюється за допомогою регістра ADMUX.

Для управління вирівнюванням результату перетворення служить розряд ADLAR регістра ADMUX. Якщо цей розряд встановлено в "1", результат перетворення вирівнюється по лівій межі 16-розрядного слова, якщо скинутий в "0" - по правій межі.

Розряди MUX2..MUX0 цього регістра визначають номер активного каналу (аналоговий вхід, підключений до входу АЦП). Стан розрядів MUX2..MUX0 можна змінити у будь-який момент, проте, якщо це буде зроблено під час циклу перетворення, зміна каналу станеться тільки після завершення перетворення (табл. 14.2). Завдяки цьому в режимі безперервного перетворення можна легко реалізувати сканування каналів. Під цим терміном в даному випадку розуміється послідовне (циклічне або за заданою програмою) перетворення сигналів декількох каналів.

Таблиця 14.2

Управління вхідним мультиплексором АЦП

MUX3	MUX2	MUX1	MUX0	Модуль АЦП підключений до входу
0	0	0	0	ADC0
0	0	0	1	ADC1
0	0	1	0	ADC2
0	0	1	1	ADC3
0	1	0	0	ADC4
0	1	0	1	ADC5
0	1	1	0	ADC6
0	1	1	1	ADC7

Модуль АЦП може використовувати різні джерела опорної напруги. Вибір конкретного джерела опорної напруги здійснюється за допомогою розрядів REFS1 і REFS0 регістра ADMUX (табл. 14.3).

Таблиця 14.3

Вибір джерела опорної напруги

REFS1	REFS0	Джерело опорної напруги
0	0	Зовнішнє джерело опорної напруги, підключене до вивода AREF, внутрішнє джерело опорної напруги відключено
0	1	Напруга живлення AVcc із зовнішнім конденсатором, підключеним до вивода AREF
1	1	Внутрішнє джерело 1,1 В із зовнішнім конденсатором, підключеним до вивода AREF

1.2. Засоби мови асемблера для роботи з АЦП

Регістри вводу-виводу, які використовуються для управління та обміну даними з модулем АЦП, розміщуються в області додаткових регістрів вводу-

виводу пам’яті даних контролера. Це тому до них не можна звертатися за допомогою команд in та out, оскільки зона досяжності цих команд обмежується основними регістрами вводу-виводу і не розповсюджується на додаткові регістри вводу-виводу.

Для звернення до регістрів АЦП використовуються наступні команди:

ST X, Rr – непряме записування до пам’яті даних. Команда зберігає вміст регістра загального призначення Rr до пам’яті даних. Адреса комірки, до якої звертається команда, міститься в індексному регістрі X (регістрова пара R27:R26).

LD Rd, X - непряме зчитування пам’яті даних. Команда завантажує один байт з адресного простору пам’яті даних до регістру загального призначення Rd. Адреса комірки, до якої звертається команда, міститься в індексному регістрі X (регістрова пара R27:R26).

1.3. Засоби мови C for Arduino для роботи з АЦП

Виводи A0-A5 плати Arduino Uno відповідають входам ADC0-ADC5 аналогового мультиплексора АЦП мікроконтролера. Для роботи з АЦП використовуються наступні вбудовані функції:

analogRead(pin)	Зчитує значення з аналогового порта pin, де pin – номер аналогового порта, з якого буде здійснюватися зчитування (0…5). Напруга, що подана на обраний вхід (зазвичай від 0 до 5 В), буде перетворено в цілочисельне значення від 0 до 1023 (10 біт), тобто крок дискретизації за рівнем напруги становить 0,0049 В. Функція повертає значення типу int.
analogReference(type)	Функція визначає опорну напругу, відносно якої здійснюються аналогові вимірювання. Параметр type може приймати наступні значення: DEFAULT – стандартна опорна напруга 5 В; INTERNAL – вбудоване джерело опорної напруги 1,1 В; EXTERNAL – зовнішнє джерело опорної напруги, що підключено до виводу AREF.

2. ОПИС ОБЛАДНАННЯ, ЩО ВИКОРИСТОВУЄТЬСЯ

2.1. Експериментальний стенд

Використовується експериментальний стенд, що був описаний в методичних вказівках до попередніх лабораторних робіт.

2.2. Модель експериментального стенда в Proteus

Використовується модель лабораторного стенда в Proteus, що знаходиться у файлі LAB5_MPT.pdsprj (рис. 14.3). Модель передбачає під'єднання до плати Arduino світлодіодів VD1 ("Напруга перевищує уставку"), VD2 ("Напруга менша від уставки") і потенціометра R25. Повзунок потенціометра підключається до аналогового входу A0 плати, який відповідає входу ADC0 мікроконтролера. Напруга, що знімається з потенціометра та подається на вхід ADC0, додатково вимірюється вольтметром. Положення повзунка потенціометра регулюється шляхом натискання на круглі мітки збоку від його графічного образу.

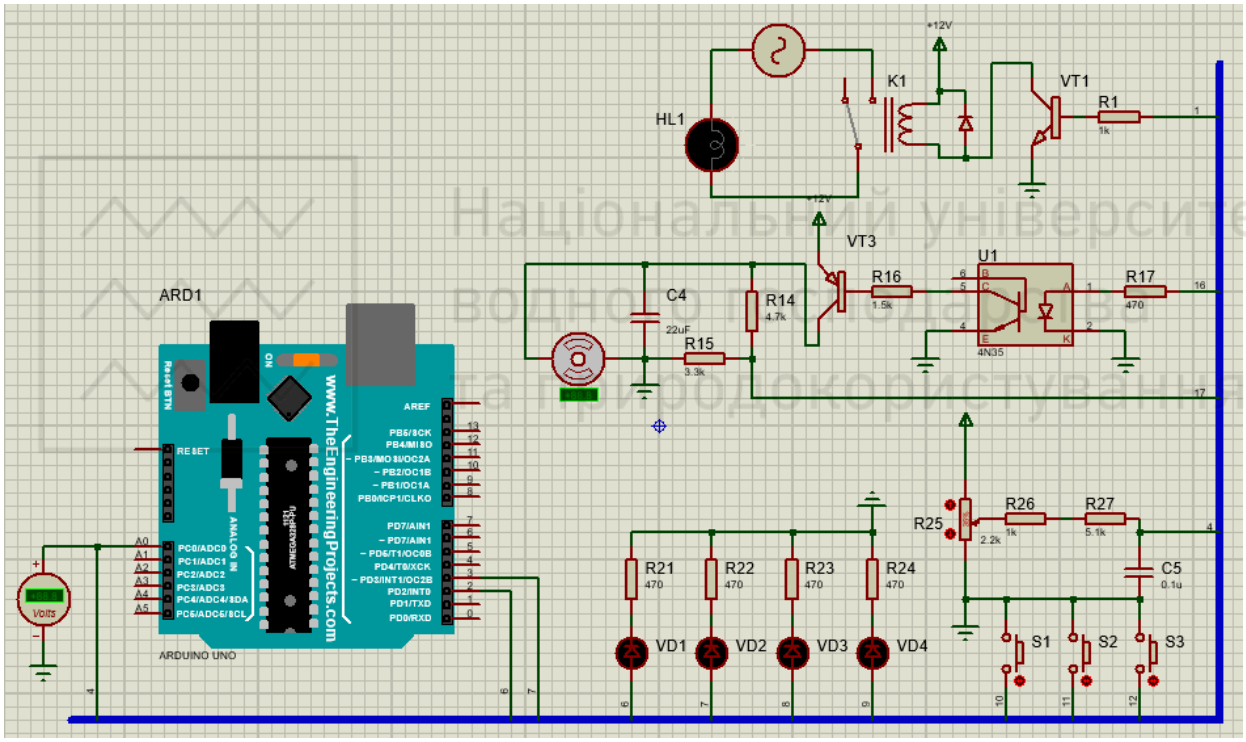


Рисунок 14.3 - Модель експериментального стенда в Proteus

3. ЗАВДАННЯ

Скласти програму для мікроконтролера, яка контролює рівень напруги на аналоговому вході. Якщо фактична напруга більше уставки, треба включити світлодіод VD1. Якщо напруга менше від уставки, то треба включити світлодіод VD2.

Програму необхідно написати на мовах асемблера та C for Arduino, відлагодити на моделі та реальному стенді.

Плата Arduino Uno підключається до експериментального стенду згідно табл. 14.4.

Таблиця 14.4

Підключення стенда до Arduino Uno для виконання

лабораторної роботи

Об’єкт	№ вихідного контакту стенда	Плата Arduino Uno	
		№ порта плати Arduino Uno	порт мікроконтролера ATMega328p
світлодіод VD1	6	2	PD2
світлодіод VD2	7	3	PD3
потенціометр R25	4	A0	ADC0

4. ПОРЯДОК ВИКОНАННЯ РОБОТИ

4.1 Написання програми на мові асемблера

4.1.1 Відкрити програму Atmel Studio, створити новий проект на мові асемблера. При цьому обрати тип мікроконтролера, який встановлено в платі Arduino Uno, а саме ATMega328p

4.1.2 Набрати наступну програму:

```
.INCLUDE "m328pdef.inc"
.CSEG                          ;ПОЧАТОК програмного сегмента
        ldi r16,high(RAMEND)   ;ініціалізація стека
        out SPH,r16
        ldi r16,low(RAMEND)
        out SPL,r16

        ldi r16,(1<<PD2)|(1<<PD3);конфігурування порта B
        out DDRD,r16

        ldi r21,111            ;завантажити значення уставки в регістр r21
                               ;уставка =111 рівнів квантування з 255,
                               ;що становить 111*5B/255=2,18B

        ldi r16,(1<<ADLAR)+(1<<REFS0) ;ADLAR=1 - вирівняти результат
                               ;перетворення по лівому краю
                               ;REFS0=1 - джерелом опорної напруги є
                               ;напруга живлення AVcc

        clr r27                ;очистити старший байт індексного регістра X
        ldi r26,ADMUX          ;завантажити в молодший байт X
                               ;молодший байт адреси

        st X,r16               ;завантажити r16 в регістр ADMUX

Lmain:  ldi r16,(1<<ADEN)+(1<<ADSC)+(1<<ADIF)+(1<<ADPS0)+(1<<ADPS1)
                               ;ADEN=1 – дозволити роботи АЦП,
                               ;ADSC=1 - запуск перетворення,
                               ;ADIF – скидання прапора переривання,
                               ;ADPS0=1 и ADPS1=1 - коеф дільника частоти=8
        ldi r26,ADCSRA         ;завантажити в r26 адресу PBB ADCSRA
        st X,r16               ;завантажити r16 в регістр ADCSRA
Lres:
        ldi r26,ADCSRA
        ld r16,X               ;читати в r16 вміст ADCSRA (адреса - в X)
        sbrs r16,ADIF          ;якщо прапор ADIF встан., то резул. готовий
        rjmp Lres

        ldi r26,ADCH
```

```
        ld r20,X                ;читати результат АЦП в r20

        cp r20,r21              ;порівняти результ (в r20) з уставкою(в r21)

        brsh L1                 ;якщо r20>=r21, то перехід на мітку L1
                                ;r20<r21 (результат менше уставки)
        cbi PORTD,PD2           ;відключити VD1
        sbi PORTD,PD3           ;включити VD2
        rjmp Lmain              ;заново запускати АЦП
L1:                             ;r20>=r21 (результат більше уставки)
        sbi PORTD,PD2           ;включити VD1
        cbi PORTD,PD3           ;відключити VD2
        rjmp Lmain              ;заново запускати АЦП
```

4.1.3 Виконати компіляцію програми.

4.2. Завантажити програму на мові асемблера до моделі експериментального стенда. Запустити модель стенда. Відслідкувати функціонування стенда. Зберегти Print Screen екрана з працюючою моделлю стенда для звіту.

4.3. Завантажити програму на мові асемблера до плати. Зібрати схему досліду, показати її викладачеві. Підключити плату Arduino до стенда та відслідкувати функціонування.

4.4 Написання програми на мові C for Arduino

4.4.1 Запустити програму IDE Arduino.

4.4.2 Набрати наступну програму:

```
int x = 0;                  //фактичне значення
int x_ust=1023/255*111;     //уставка, перерахована на 10 розрядів

void setup() {
   pinMode(2, OUTPUT);    //VD1
   pinMode(3, OUTPUT);    //VD2
}

void loop() {
  x = analogRead(0);
  if (x>x_ust)
      {digitalWrite(2, HIGH);
      digitalWrite(3, LOW);}
  else
      {digitalWrite(2, LOW);
      digitalWrite(3, HIGH);}
}
```

4.4.3 Виконати компіляцію скетча.

4.4.4 Завантажити програму на мові C for Arduino до моделі експериментального стенда. Відслідкувати функціонування стенда.

4.4.5 Завантажити програму на мові C for Arduino до плати Arduino. Відслідкувати функціонування експериментального стенда.

ВМІСТ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

1. Тема, мета роботи.
2. Блок-схема алгоритму функціонування мікроконтролера (складається самостійно) з поясненнями.
3. Повний текст програми на мові асемблера та вікно Atmel Studio з програмою на мові асемблера.
4. Програма на мові C for Arduino та вікно IDE Arduino з програмою.
5. Модель лабораторного стенда в Proteus.
6. Висновки.

КОНТРОЛЬНІ ПИТАННЯ

1. Пояснити призначення аналого-цифрового перетворювача.
2. У мікроконтролера є шість аналогових входів (ADC0-ADC5). Скільки АЦП містить мікроконтролер?
3. Пояснити структура АЦП мікроконтролера.
4. Які регістри вводу-виводу використовуються для керування модулем АЦП та обміну даними?
5. В яких режимах може функціонувати модуль АЦП?
6. Що показує прапор ADIF?
7. Як налаштувати переддільник тактової частоти АЦП?
8. Де зберігається результат перетворення АЦП?
9. Для чого використовується розряд ADLAR регістра ADMUX?
10. Для чого використовується вхідний мультиплексор АЦП?
11. Які команди мови асемблера використовуються для звернення до регістрів АЦП?
12. Які функції мови C for Arduino використовуються дл роботи з АЦП?

ЛАБОРАТОРНА РОБОТА №15
Послідовна передача даних у мікропроцесорній системі

Мета: освоїти методи послідовної передачі даних на прикладі інтерфейсу RS-232.

1. КОРОТКІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Способи послідовної передачі даних

Інтерфейс (від англ. interface - поверхня розділу, перегородка) - сукупність засобів, методів і правил взаємодії (управління, контролю і так далі) між елементами системи.

Для обміну даними між двома мікропроцесорними пристроями використовуються інтерфейси двох основних типів:

- *паралельні*, які передбачають одночасну передачу байта (чи тетради) даних по паралельних провідниках, приклад: LPT-порт комп'ютера, інтерфейс IDE підключення накопичувачів до системної плати;
- *послідовні*, що передбачають побітову передачу даних, тобто в кожен момент часу передається або приймається тільки один біт, приклад: COM -порт комп'ютера, де використовується інтерфейс RS-232C; інтерфейс RS-485; інтерфейс універсальної послідовної шини USB.

Головна перевага послідовних інтерфейсів перед паралельними - зниження числа провідників. Також при високих швидкостях передачі даних, коли на електричні сигнали суттєво впливають затримки в лініях, послідовні інтерфейси є більш прийнятними, ніж паралельних. Крім того, при реалізації послідовного інтерфейсу можна збільшити дальність зв'язку в порівнянні з паралельним інтерфейсом.

У мікроконтролерних пристроях з малими об'ємами даних швидкість передачі не є критичним чинником при виборі типу інтерфейсу. В першу чергу має значення число провідників, що використовуються для зв’язку, тобто число задіяних виводів мікроконтролера. Тому при проектуванні мікропроцесорних пристроїв автоматизації перевагу віддають послідовним інтерфейсам.

Послідовна передача даних може здійснюватися в двох режимах: асинхронному або синхронному.

При **асинхронній передачі** кожному байту передує старт-біт, що сигналізує приймачу про початок посилки, за яким йдуть біти даних і, можливо, біт паритету (парності). Завершується посилка стоп-бітом, що гарантує паузу між посилками (рис. 15.1).

Рисунок 15.1 - Формат асинхронної передачі

Старт-біт наступного байта посилається у будь-який момент після стоп-бита, тобто між передачами можливі паузи довільної тривалості. Старт-біт, що має завжди певне значення (логічний 0), забезпечує простий механізм синхронізації приймача по сигналу від передавача.

Для асинхронного режиму прийнятий ряд стандартних швидкостей обміну: 50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19 200, 38 400, 57600 і 115200 біт/с. Асинхронний обмін в комп’ютері реалізується за допомогою COM-порту з використанням протоколу RS-232C.

Синхронний режим передачі припускає постійну активність каналу зв'язку. Посилка починається з синхробайта, за яким відразу ж йде потік інформаційних біт. У синхронному режимі необхідна зовнішня синхронізація приймача з передавачем, оскільки навіть мале відхилення частот приведе до спотворення даних, що приймаються. Зовнішня синхронізація можлива або за допомогою окремої лінії для передачі сигналу синхронізації, або з використанням кодування даних з самосинхронізацією, при якому на боці приймача з прийнятого сигналу можуть бути виділені імпульси синхронізації.

Існує ряд споріднених міжнародних стандартів: RS-232C, RS-423A, RS-422A і RS-485. Найбільшого поширення в персональному комп’ютері набув найпростіший з перерахованих - стандарт RS-232C, який реалізується COM-портами. У промисловій автоматиці та телемеханіці широко застосовується RS-485. У перерахованих стандартах сигнал представляється потенціалом.

1.2. Послідовний інтерфейс RS-232

Інтерфейс не забезпечує гальванічну розв'язку пристроїв. Стандарт RS-232 описує несиметричні передавачі і приймачі: сигнал передається відносно загального дроту - схемної "землі" (рис. 14.2). Логічній одиниці на вході даних (сигнал RxD) відповідає діапазон напруги від -12 до -3 В; логічному нулю - від +3 до +12 В. Діапазон від -3 до +3 В - зона нечутливості, що обумовлює гістерезис приймача: стан лінії вважається зміненим тільки після перетину порогу (рис. 3). Різниця потенціалів між схемними "землями" (SG) пристроїв, що сполучаються, має бути менше 2 В, при вищій різниці потенціалів можливе невірне сприйняття сигналів.

Для управління потоком даних можуть використовуватися два варіанти протоколу - апаратний і програмний.

Апаратний протокол забезпечує найшвидшу реакцію передавача на стан приймача. Мікросхеми асинхронних приймачів-передавачів мають не менше двох регістрів в приймальній частині – регістра зсуву для прийому чергової посилки, і буферного регістра, з якого прочитується прийнятий байт. Це дозволяє реалізовувати обмін по апаратному протоколу без втрати даних.

Програмний протокол управління потоком XON/XOFF передбачає наявність двоспрямованого каналу передачі даних. Працює протокол таким чином: якщо пристрій, що приймає дані, виявляє причини, з яких не може їх далі приймати, він по зворотному послідовному каналу посилає байт-символ XOFF. Протилежний пристрій, прийнявши цей символ, зупиняє передачу. Коли приймаючий пристрій знову стає готовим до прийому даних, він посилає символ XON, прийнявши який протилежний пристрій поновлює передачу.

Перевага програмного протоколу полягає у відсутності необхідності передачі сигналів інтерфейсу управління - мінімальний кабель для двостороннього обміну може мати тільки 3 провідники. Недоліком, окрім вимоги наявності буфера і більшого часу реакції, є складність реалізації повнодуплексного режиму обміну.

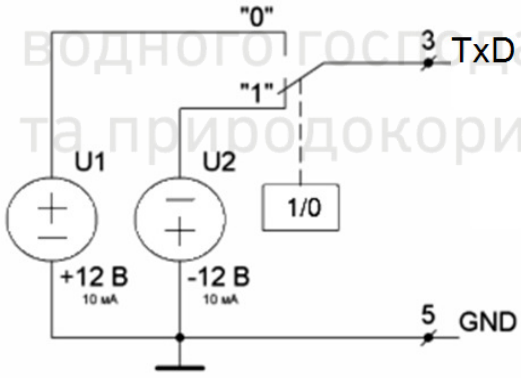


Рисунок 15.2 - Формування сигналів RS-232

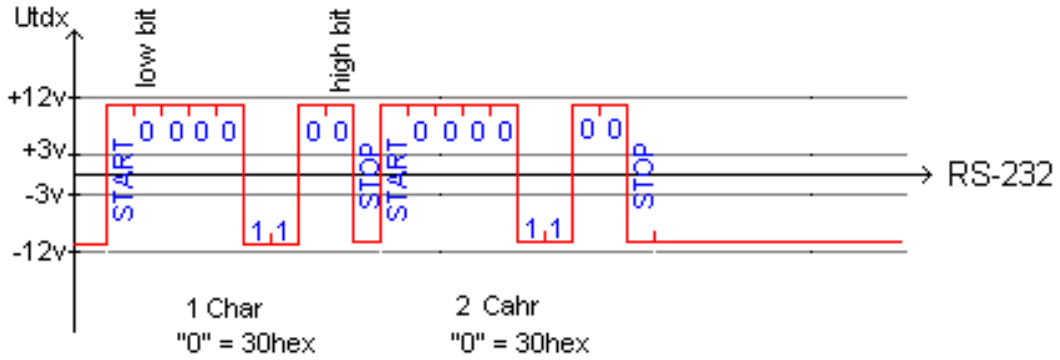


Рисунок 15.3 - Форма сигналів RS-232 при передачі двох символів "0" і "0".

1.3. Модуль USART (UART)

Для забезпечення зв'язку по каналу RS-232 мікроконтролери Atmel сімейства Mega і вище мають у своєму складі апаратні модулі універсального асинхронного приймача UART (Universal Asynchronous Receiver and Transmitter) або універсального синхронно / асинхронний приймач USART.

У асинхронному режимі роботи UART відрізняється від RS-232 тільки логічними рівнями: у UART вони зазвичай відповідають ТТЛ ("0" відповідає 0,4 В, рівень "1" відповідає 2,4 В) рівням, у RS-232 – становлять +12 В для логічного нуля і -12 В для логічної одиниці).

У мікроконтролерах AVR з UART апаратно-програмними засобами підтримуються тільки дві лінії: лінія прийому (RxD) і лінія передачі даних (TxD). Тобто для зв'язку комп'ютера з мікроконтролером використовується тільки загальний провід і дві сигнальні лінії. По одній з них дані передаються від мікроконтролера до комп'ютера, по іншій - від комп'ютера до мікроконтролера. Дані по каналу передаються послідовно, біт за бітом. Інформація передається групами бітів. Для виділення групи перед її початком передається стартовий біт, після закінчення передачі групи - один або два стопових біта, яким може передувати біт парності. І передавальний, і приймаючий пристрої повинні працювати на однаковій швидкості, яка може бути вибрана з передбаченого стандартом ряду.

На лініях RxD і TxD мікроконтролера рівні сигналів відповідають рівням ТТЛ-логіки, тому для підключення мікроконтролера до каналу RS-232 потрібен перетворювач рівнів. В якості останнього найчастіше використовується інтегральна мікросхема MAX232, що перетворює сигнали послідовного порту RS-232 в сигнали, придатні для використання в цифрових схемах на базі ТТЛ-логіки.

У більшості моделей сімейства Mega UART замінений більш функціональним USART (універсальний синхронно-асинхронний приймач-передавач).

До складу мікроконтролера **ATMega328p**, встановленого на платі Arduino UNO, входить модуль USART, який використовує наступні виводи мікроконтролера:

− RXD - вхід USART;

− TXD - вихід USART;

− XCK - вхід/вихід зовнішнього тактового сигналу (використовується тільки в синхронному режимі роботи).

Модуль USART0 використовує такі **регістри вводу-виводу** (рис. 15.4):

1) **UBRR0** - регістрова пара у складі регістрів **UBRR0H:UBRR0L** (USART Baud Rate Registers) - **регістр контролера швидкості передачі**.

Контролер швидкості передачі функціонує як дільник системного тактового сигналу з програмованим коефіцієнтом ділення. Регістр UBRR0 є 12-розрядним і фізично розміщується в двох регістрах введення/виведення UBRR0H:UBRR0L.

При роботі в асинхронному режимі швидкість обміну визначається не лише вмістом регістра UBRR0, але і станом розряду U2X0 регістра UCSR0A.

Якщо цей розряд встановлений в "1", коефіцієнт ділення переддільника зменшується в два рази, а швидкість обміну відповідно подвоюється. При роботі в синхронному режимі цей розряд має бути скинутий.

Значення коефіцієнта ділення UBRR0 контролера швидкості передачі в звичайному режимі (при U2X0="0") обчислюється таким чином:

$$\text{UBRR0} = \frac{f_{clk}}{16 \cdot \text{BAUD}} - 1, \qquad (15.1)$$

де f_{clk} - частота тактового генератора мікроконтролера, Гц;

BAUD - швидкість передачі, біт/с.

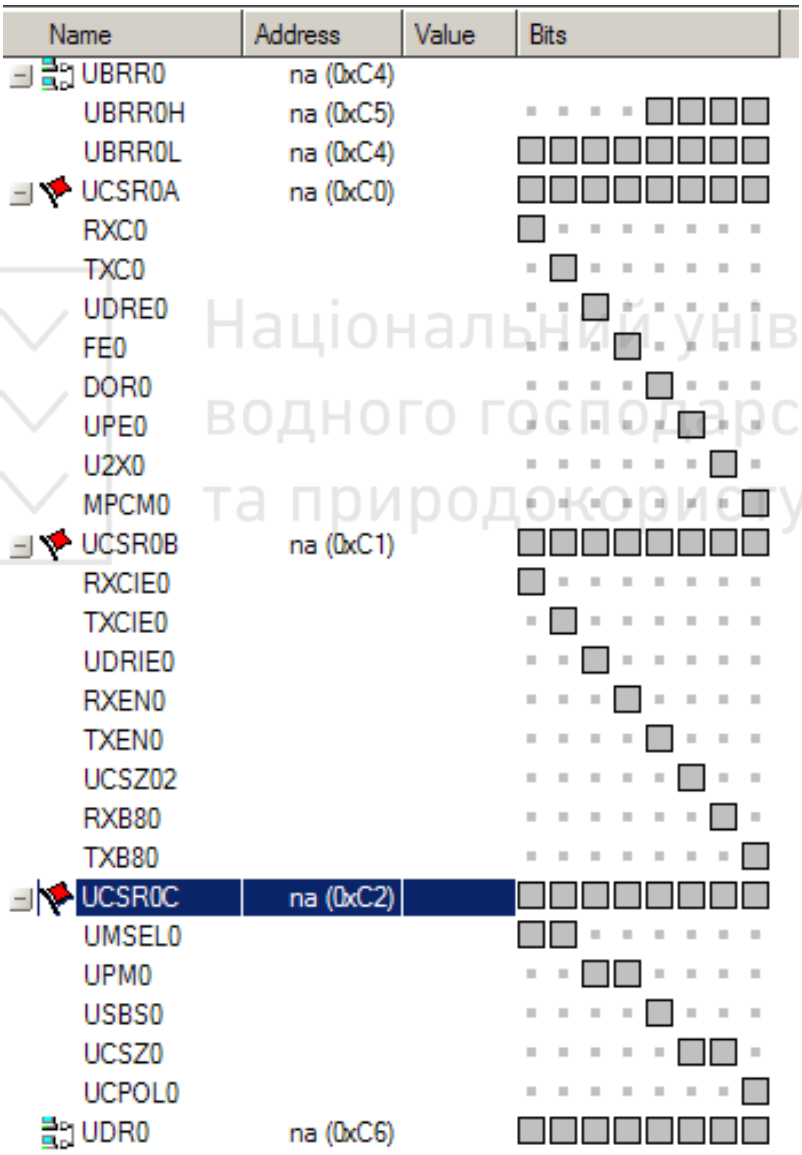


Рисунок 15.4 - Регістри вводу-виводу, що використовуються модулем USART0 мікроконтролера ATMega328p

2) **UCSR0A** - регістр управління А

7	6	5	4	3	2	1	0
RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0

Опис розрядів регістра UCSR0A:

7	RXC0	**Прапор завершення прийому.** Прапор встановлюється в "1" за наявності непрочитаних даних в буфері приймача (регістр даних UDR). Скидається прапор апаратно після спустошення буфера (у UART - після прочитання регістра даних). Якщо розряд RXCIE0 регістра UCSR0B встановлений, то при встановленні прапора генерується запит на переривання "прийом завершений".
6	TXC0	**Прапор завершення передачі.** Прапор встановлюється в "1" після передачі усіх розрядів посилки з регістру зсуву передавача, за умови, що в регістр даних UDR0 не було завантажено нового значення. Якщо розряд TXCIE0 регістра UCSR0B встановлений, то при установці прапора генерується переривання "передачу завершено". Прапор скидається апаратно при виконанні підпрограми обробки переривання або програмно, записом в нього лог. 1.
5	UDRE0	**Прапор спустошення регістра даних.** Цей прапор встановлюється в "1" при порожньому буфері передавача (після пересилки байта з регістра даних UDR0 в регістр зсуву передавача). Встановлений прапор означає, що в регістр даних можна завантажувати нове значення. Якщо розряд UDRIE0 регістра UCSR0B встановлений, генерується запит на переривання "регістр даних порожній". Прапор скидається апаратно при записі в регістр даних.
4	FE0	**Прапор помилки кадрування.** Прапор встановлюється в "1" при виявленні помилки кадрування, тобто якщо перший стоп-біт прийнятої посилки дорівнює "0". Прапор скидається при прийомі стоп-бита, що дорівнює "1".
3	DOR0	**Прапор переповнення.** В USART прапор встановлюється в "1", якщо у момент виявлення нового старт-біта в регістрі зсуву приймача знаходиться останнє прийняте слово, а буфер приймача повний (два значення). У UART прапор встановлюється в "1", якщо новий кадр буде поміщений в регістр зсуву приймача до того, як з регістра даних буде зчитане попереднє слово. Прапор скидається при пересиланні прийнятих даних з регістра зсуву приймача в буфер.
2	UPE0	**Прапор помилки контролю парності.** Прапор встановлюється в "1", якщо в даних, що знаходяться в буфері приймача, виявлена помилка контролю парності. При відключеному контролі парності цей розряд постійно скинутий в "0".
1	U2X0	**Подвоєння швидкості обміну.** Якщо цей розряд встановлений в "1", коефіцієнт ділення переддільника контролера швидкості передачі зменшується з 16 до 8, подвоюючи тим самим швидкість асинхронного обміну по послідовному каналу. У USART розряд U2X0 використовується тільки при асинхронному режимі роботи. У синхронному режимі він має бути скинутий.
0	MPCM0	**Режим мультипроцесорного обміну.** Розряд MPCM0 використовується в режимі мультипроцесорного обміну. Якщо він встановлений в "1", ведений мікроконтролер чекає прийому кадру, що містить адресу. Кадри, що не містять адреси пристрою, ігноруються

3) **UCSR0B** - регістр управління В

7	6	5	4	3	2	1	0

RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80

Опис розрядів регістра UCSR0B:

7	RXCIE0	**Дозвіл переривання після завершення прийому.** Якщо цей розряд встановлений в "1", то при установці прапора RXC0 регістра UCSR0A генерується переривання "прийом завершений" (якщо прапор I регістра SREG встановлений в "1")
6	TXCIE0	**Дозвіл переривання після завершення передачі.** Якщо цей розряд встановлений в "1", то при установці прапора TXC0 регістра UCSR0A генерується переривання "передача завершена" (якщо прапор I регістра SREG встановлений в "1")
5	UDRIE0	**Дозвіл переривання при очищенні регістра даних.** Якщо цей розряд встановлений в "1", то при установці прапора UDRE0 регістра UCSR0A генерується переривання "регістр даних порожній" (якщо прапор I регістра SREG встановлений в "1")
4	RXEN0	**Дозвіл прийому.** При установці цього розряду в "1" дозволяється робота приймача USART/UART і перевизначається функціонування виводу RXD. При скиданні розряду RXEN0 робота приймача забороняється, а його буфер скидається. Значення прапорів TXC0, DOR0 і FE0 при цьому стають недійсними
3	TXEN0	**Дозвіл передачі.** При установці цього розряду в "1" дозволяється робота передавача UART і перевизначається функціонування виводу TXD. Якщо розряд скидається в "0" під час передачі, виключення передавача станеться тільки після завершення передачі даних, що знаходяться в регістрі зсуву і буфері передавача
2	UCSZ02	**Формат посилок.** Цей розряд використовується для завдання розміру слів даних, що передаються по послідовному каналу. У модулях USART він використовується спільно з розрядами UCSZ01:00 регістра UCSR0C.
1	RXB80	**8-й розряд даних, що приймаються.** При використанні 9-розрядних слів даних цей розряд містить значення старшого розряду прийнятого слова. У разі USART вміст цього розряду має бути зчитаний до зчитування регістра даних UDR.
0	TXB80	**8-й розряд даних, що передаються.** При використанні 9-розрядних слів даних, вміст цього розряду є старшим розрядом слова, що передається. Необхідне значення має бути занесене в цей розряд до завантаження байта даних в регістр UDR.

4) **UCSR0C** - регістр управління С

7	6	5	4	3	2	1	0
UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0

Опис розрядів регістра UCSR0C:

7	UMSEL01	0	0	1	**Режим роботи USART**
6	UMSEL00	0	1	1	

		асинхр. режим	синхр. режим	ведучий SPI	
5	UPM01	0	1	1	**Режим роботи схеми контролю та формування парності**
4	UPM00	0	0	1	
		не викор. контр. парності	перевірка на парність (even parity)	перевірка на непарність (edd parity)	
3	USBS0	**Кількість стоп-бітів.** Цей розряд визначає кількість стоп-бітів, що посилаються передавачем. Якщо розряд скинутий в "0", передавач посилає 1 стоп-біт, якщо встановлений в "1", то 2 стоп-бита. Для приймача вміст цього розряду не має значення			

UCSR0B.UCSZ02		0	0	0	0	1	**Формат посилок.** Спільно з розрядом UCSZ02 ці розряди визначають кількість розрядів даних в посилках (розмір слова)
2	UCSZ01	0	0	1	1	1	
1	UCSZ00	0	1	0	1	1	
		5 біт	6 біт	7 біт	8 біт	9 біт	
0	UCPOL0	**Полярність тактового сигналу.** Значення цього розряду визначає момент видачі і зчитування даних на виводах модуля. Розряд використовується тільки при роботі в синхронному режимі. При роботі в асинхронному режимі він має бути скинутий в "0"					

5) **UDR0** - регістр даних (Universal Data Register).

Буферні регістри приймача і передавача розташовуються за однією адресою простору вводу/виводу і позначаються як регістр даних UDR0. У цьому регістрі зберігаються молодші 8 розрядів даних, що приймаються і передаються. При читанні виконується звернення до буферного регістра UDR приймача, при запису - до буферного регістра передавача.

Формат кадру. Під кадром в даному випадку розуміється сукупність одного слова даних і супутньої інформації. Кадр починається із старт-біта, за яким йде молодший розряд слова даних. Після старшого розряду слова даних слідує один або два стоп-бита. Якщо включена схема формування біта парності, він включається між старшим розрядом слова даних і першим стоп-бітом (рис. 15.1).

Вибір кількості стоп-бітів в модулях USART здійснюється за допомогою розряду USBS0 регістра UCSR0C. Якщо цей розряд скинутий в "0", блок передавача формує 1 стоп-біт у кінці посилки. Якщо розряд встановлений в "1", блок передавача формує 2 стоп-бита. Слід зазначити, що приймачем другий стоп-біт ігнорується, і відповідно помилки кадрування виявляються тільки для першого стоп-бита.

Розряди UPM01:UPM00 регістра UCSR0C визначають функціонування схеми контролю парності модулів USART. Значення біта парності вираховується шляхом виконання операції XOR над усіма розрядами слова даних, що передається. Якщо використовується перевірка на непарність (odd parity), отриманий результат інвертується. Якщо контроль парності включений,

біт парності вставляється передавачем між старшим розрядом даних, що передаються, і першим стоп-бітом.

Передача даних. Робота передавача дозволяється установкою в "1" розряду TXEN0 регістра UCSR0B. При установці розряду вивід TXD0 підключається до передавача USART/UART і починає функціонувати як вихід незалежно від установок регістрів управління портом. Якщо використовується синхронний режим роботи (у USART), перевизначається також функціонування виводу XCK.

Передача ініціюється записом даних, що передаються, в буферний регістр передавача - регістр даних UDR. Після цього дані пересилаються з регістра UDR в регістр зсуву передавача. Одночасно, якщо використовуються 9-розрядні дані, значення розряду TXB80 регістра UCSR0B копіюється в 9-й розряд регістра зсуву.

Після пересилки слова даних в регістр зсуву, прапор UDRE0 регістра UCSR0A встановлюється в "1", що означає готовність передавача до отримання нового слова даних. У цьому стані прапор залишається до наступного запису в буфер. Одночасно з пересилкою в регістрі формується службова інформація - старт-біт, можливий біт парності (тільки у USART), а також один або два стоп-біта. Після завантаження регістра зсуву його вміст починає зсуватися вправо і поступати на вивід TXD. Швидкість зсуву визначається налаштуваннями контролера швидкості передачі.

Прийом даних. Робота приймача дозволяється встановленням розряду RXEN0 регістра UCSRB (UCSRnB). При установці розряду вивід RXD (RXDn) підключається до приймача USART/UART і починає функціонувати як вхід незалежно від установок регістрів управління портом. Якщо використовується синхронний режим роботи (у USART), перевизначається також функціонування виводу XCK.

Прийом даних починається відразу ж після виявлення приймачем коректного старт-біта. Кожен розряд вмісту кадру потім прочитується з частотою, яка визначається установками контролера швидкості передачі або тактовим сигналом XCK. Зчитані розряди даних послідовно поміщаються в регістр зсуву приймача до виявлення першого стоп-біта кадру. Після цього вміст регістра зсуву пересилається в буфер приймача, з якого прийняте значення може бути отримане шляхом читання регістра даних модуля. При використанні 9-розрядних слів даних значення старшого розряду може бути визначене за станом прапора RX80 регістра UCSR0B.

Якщо під час прийому кадру була включена схема контролю парності (тільки USART), вона обчислює біт парності для усіх розрядів прийнятого слова даних і порівнює його з прийнятим бітом парності. Результат перевірки запам'ятовується в буфері приймача разом з прийнятим словом даних і стоп-бітами. Наявність або відсутність помилки контролю парності може бути потім визначена за станом прапора UPE0. Цей прапор встановлюється в "1", якщо наступне слово, яке може бути прочитане з буфера, має помилку контролю

парності. При вимкненому контролі парності прапор UPE0 завжди читається як "0".

Регістри вводу-виводу модуля USART мікроконтролера ATMega328p розташовуються в додатковому адресному просторі регістрів вводу-виводу, тому для звернення до них необхідно використовувати команди непрямого запису ST і непрямого читання LD.

1.4. Команди мови C for arduino для послідовної передачі даних

Для роботи з модулем UART використовуються наступні функції бібліотеки Serial.

Serial.begin(speed)	Ініціює послідовне з'єднання та задає швидкість speed передачі даних в біт/с (бод). Для обміну даними з комп'ютером використовуються наступні значення швидкості speed: 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200.
Serial.end()	Закриває послідовне з'єднання.
Serial.available()	Отримує кількість байтів (символів), що доступні для читання з послідовного інтерфейсу зв'язку. Буфер може зберігати до 128 байт.
Serial.read()	Зчитує черговий доступний байт з буфера послідовного з'єднання.
Serial.flush()	Очікує закінчення передачі вихідних даних.
Serial.print(val)	Передає дані val через послідовний порт як текст ASCII.
Serial.println(val)	Передає дані val через послідовний порт як текст ASCII, доповнюючи його символом переносу рядка (символ ASCII 13 або '\r') та символом нового рядка (символ ASCII 10 або '\n').
Serial.write(val) Serial.write(str) Serial.write(buf, len)	Передає дані через послідовний порт як бінарний код. Дані посилаються одиничним байтом (val), серією байтів (str) або масивом байтів buf довжиною len.

1.5. Передача даних з використанням кодування ASCII

При роботі в текстовому режимі, такому як сеанс MS DOS, в сумісних з IBM PC комп'ютерах використовується стандартна таблиця символів ASCII (*American Standard Code for Information Interchange, Американський стандартний код обміну інформацією*). У таблиці ASCII кожному символу призначається стандартний унікальний 7-розрядний двійковий код. Послідовність одного або декількох символів називається рядком. Рядки у форматі ASCII (чи ASCII-рядки) зберігаються в пам'яті комп'ютера у вигляді послідовності байтів, що є ASCII-кодами.

Оскільки в ASCII-кодах використовуються тільки молодші 7 бітів кожного байта, то додатковий 8-й біт може використовуватися на різних

комп'ютерних платформах для підтримки локальної таблиці символів. Наприклад, в сумісних з IBM PC комп'ютерах значення кодів ASCII-таблиці в діапазоні від 128 до 255 використовуються для представлення псевдографічних символів і символів національних алфавітів (табл. 15.1). Щоб знайти в таблиці шістнадцятковий код потрібного символу, слід поглянути на рядок і стовпчик, на перетині яких він розташований в таблиці. Код символу відповідає сумі номера рядка і номера стовпця. В якості прикладу визначимо ASCII-код англійської букви "S". Буква знаходиться на перетині рядка з номером 50h і стовпчика з номером 03h. Код англійської букви "S" складає: 50h+03h=53h.

Таблиця 15.1

ASCII-коди символів *

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80	Ђ 0402	Ѓ 0403	‚ 201A	ѓ 0453	„ 201E	… 2026	† 2020	‡ 2021	€ 20AC	‰ 2030	Љ 0409	‹ 2039	Њ 040A	Ќ 040C	Ћ 040B	Џ 040F
90	ђ 0452	‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014		™ 2122	љ 0459	› 203A	њ 045A	ќ 045C	ћ 045B	џ 045F
A0	NBSP 00A0	Ў 040E	ў 045E	Ј 0408	¤ 00A4	Ґ 0490	¦ 00A6	§ 00A7	Ё 0401	© 00A9	Є 0404	« 00AB	¬ 00AC	- 00AD	® 00AE	Ї 0407
B0	° 00B0	± 00B1	І 0406	і 0456	ґ 0491	µ 00B5	¶ 00B6	· 00B7	ё 0451	№ 2116	є 0454	» 00BB	ј 0458	Ѕ 0405	ѕ 0455	ї 0457
C0	А 0410	Б 0411	В 0412	Г 0413	Д 0414	Е 0415	Ж 0416	З 0417	И 0418	Й 0419	К 041A	Л 041B	М 041C	Н 041D	О 041E	П 041F
D0	Р 0420	С 0421	Т 0422	У 0423	Ф 0424	Х 0425	Ц 0426	Ч 0427	Ш 0428	Щ 0429	Ъ 042A	Ы 042B	Ь 042C	Э 042D	Ю 042E	Я 042F
E0	а 0430	б 0431	в 0432	г 0433	д 0434	е 0435	ж 0436	з 0437	и 0438	й 0439	к 043A	л 043B	м 043C	н 043D	о 043E	п 043F
F0	р 0440	с 0441	т 0442	у 0443	ф 0444	х 0445	ц 0446	ч 0447	ш 0448	щ 0449	ъ 044A	ы 044B	ь 044C	э 044D	ю 044E	я 044F

* - під символом вказаний його номер в UNICODE.

Деякі символи таблиці ASCII використовуються в якості службових. Наприклад:

LF 0x0A	Line Feed (Переведення рядка). Вказує на рух механізму друку або курсору дисплея на початок наступного рядка (на один рядок вниз).
CR	Carriage Return (Переведення каретки). Вказує на рух механізму друку або

0x0D	курсору дисплея до вихідної (крайньою лівою) позиції поточного рядка.
SP 0x20	Space (Пропуск). Недрукований символ для розподілу слів або переміщення механізму друку або курсору дисплея вперед на одну позицію.

2. ОПИС ОБЛАДНАННЯ, ЩО ВИКОРИСТОВУЄТЬСЯ

2.1. Плата Arduino UNO

Плата підключається до персонального комп'ютера. Для зчитування даних, що передаються по інтерфейсу RS-232 від плати до комп'ютера, використовується монітор послідовного порту. Для його відкриття в програмі Arduino IDE необхідно з меню «Інструменти» виконати команду «Монітор порту».

2.2. Модель плати Arduino UNO в Proteus

Використовується модель в Proteus (рис. 15.5), яка включає плату Arduino UNO з підключеним віртуальним терміналом Virtual Terminal. Останній дозволяє обмінюватися даними по інтерфейсу RS-232 з віртуальною платою Arduino. Модель знаходиться у файлі LAB6_MPT.pdsprj.

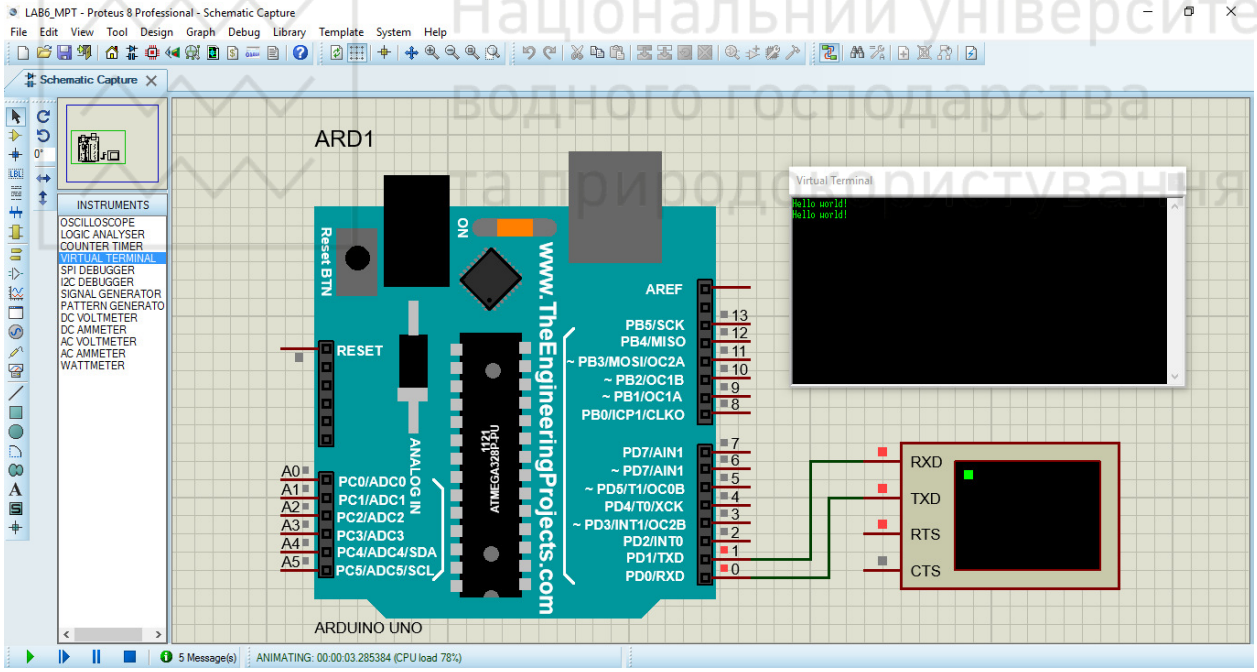


Рисунок 15.5 - Модель в Proteus для дослідження послідовного обміну даними за інтерфейсом RS-232

3. ЗАВДАННЯ

Написати програму, згідно якої мікроконтролер з інтервалом 2с передає в комп'ютер рядок «Hello world!». На комп'ютері це повідомлення відобразити за допомогою монітора послідовного порту.

Завдання виконати за наступних **умов:**

- плата Arduino UNO обмінюється даними з комп'ютером по інтерфейсу RS-232, який зі сторони плати емулюється перетворювачем USB<->USART, а зі сторони комп’ютера – програмно драйвером.

- швидкість обміну 57600 біт/с;
- кадр містить 8 біт даних;
- парність не контролюється;
- кількість стоп-бітів 1.

Програму необхідно написати на мовах асемблера та C for Arduino, відлагодити на моделі та реальному стенді.

4. ПОРЯДОК ВИКОНАННЯ РОБОТИ

4.1. Підключити плату Arduino UNO до комп’ютера. При цьому необхідно налаштувати віртуальний com-порт комп’ютера для роботи при заданих параметрах (рис. 15.6).

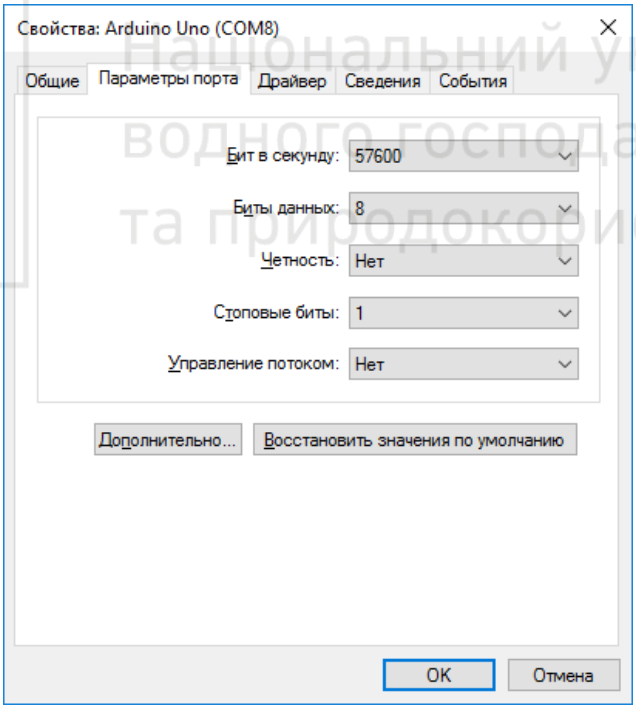


Рисунок 15.6 - Вікно налаштування параметрів віртуального послідовного порту

4.2. Закодувати повідомлення. В якості повідомлення пропонується текст «Hello world!», що кодується за допомогою таблиці ACSII, а саме:

Символ	H	e	l	l	o		w	o	r	l	d	!	CR	LF
ACSII-код, hex	48	65	6C	6C	6F	20	77	6F	72	6C	64	21	0D	0A

В кінці рядка додані службові символи CR і LF закінчення рядка для того, аби наступні повідомлення відображалися в моніторі послідовного порту з нового рядка.

4.3. Написати програму на мові асемблера.

4.3.1 Відкрити програму Atmel Studio, створити новий проект на мові асемблера. При цьому обрати тип мікроконтролера, який встановлено в платі Arduino Uno, а саме ATMega328p.

4.3.2 Набрати наступну програму:

```
.INCLUDE "m328pdef.inc"
.CSEG
    ldi r16,high(RAMEND)      ;ініціалізація стеку
    out SPH,r16
    ldi r16,low(RAMEND)
    out SPL,r16
                              ;формат посилки - 8 біт
    ldi r16,(1<<UCSZ01)+(1<<UCSZ00)
    ldi XL,low(UCSR0C)        ;UCSR0C знаходиться у додатковому
    ldi XH,high(UCSR0C)       ;просторі РВВ (адреса 0xC2), тому
    st X,r16                  ;команда непрямого запису st

    ldi r16,low(16)           ;завантаж. до UBRR0 конст. 16, що
    ldi XL,low(UBRR0L)        ;відповідає швидкості 57600 біт/с
    ldi XH,high(UBRR0L)       ;при f_clk=16 МГц
    st X,r16

    ldi r16,high(16)
    ldi XL,low(UBRR0H)
    ldi XH,high(UBRR0H)
    st X,r16

    ldi r16,(1<<TXEN0)        ;дозволити передачу
    ldi XL,low(UCSR0B)
    ldi XH,high(UCSR0B)
    st X,r16

Ls:    ldi ZL,low(mes*2)      ;завантажити до Z адресу першого
     ldi ZH,high(mes*2)       ;байта рядка
           ;*2 – оскільки адресація по словам (по 2 байти)
    ldi r19,14                ;лічильник символів (14 символів)
L2:    lpm r20,Z+             ;завантажити з пам'яті прогр.
                              ;(адреса комірки - в Z)
                              ;до r20 байт, який кодує символ,
                              ;після цього Z=Z+1 (пдготовка
                              ;до читання наступного символа)
     ldi XL,low(UDR0)         ;передати символ
     ldi XH,high(UDR0)
     st X,r20

Ltr:    ldi XL,low(UCSR0A)    ;чекати на закінчення передачі
```

```
      ldi XH,high(UCSR0A)       ;(тобто перевіряти прапор TXC0)
         ld r16,X
         sbrs r16,TXC0
         rjmp Ltr

         ldi r16,(1<<TXC0)      ;скинути прапор TXC0 шляхом запису
         ldi XL,low(UCSR0A)     ;до нього лог. "1"
         ldi XH,high(UCSR0A)
         st X,r16

    dec r19                     ;декремент лічильника символів
                                ;в рядку (наступний символ)
    brne L2                     ;якщо не всі символи передані, то
                                ;передавати далі

    ldi XL,low(2000)            ;затримка 2с
    ldi XH,high(2000)
    rcall DELAY

    rjmp Ls                     ;знову передавати рядок

DELAY:                          ;підпрограма затримки
                                ;вхід: X – тривалість в мс
    ldi YL,low(3999)
    ldi YH,high(3999)
Ld1:    sbiw YL,1
        brne Ld1
    sbiw XL,1
    brne DELAY
    ret
                    ;Hello world! + символи завершення рядка
                    ;(всього 14 символів)
mes:
.db    0x48,0x65,0x6C,0x6C,0x6F,0x20,0x77,0x6F,0x72,0x6C,0x64,0x21,
0x0D,0x0A           ;символи записуються одним рядком
```

В програмі використано значення 16 коефіцієнта поділу UBRR0 контролера швидкості передачі модуля USART, яке обраховано згідно залежності (1) при швидкості обміну BAUD = 57600 біт/с, в звичайному режимі (при U2X0=«0»), причому f_{clk} = 16 МГц = $16 \cdot 10^6$ Гц - частота тактового генератора мікроконтролера ATMega328p у складі плати Arduino UNO:

$$\text{UBRR0} = \frac{f_{clk}}{16 \cdot \text{BAUD}} - 1 = \frac{16 \cdot 10^6}{16 \cdot 57600} - 1 = 16{,}36 \approx 16.$$

4.3.3 Виконати компіляцію програми.

4.4. Завантажити програму на мові асемблера до моделі плати Arduino UNO. Запустити модель в Proteus. Відслідкувати функціонування моделі. Зберегти Print Screen екрана з працюючою моделлю для звіту.

4.5. Завантажити програму на мові асемблера до плати.

4.6. Запустити монітор послідовного порту (рис. 15.7). Для цього в програмі Arduino IDE необхідно з меню «Інструменти» виконати команду «Монітор порту». Зберегти Print Screen монітора порту.

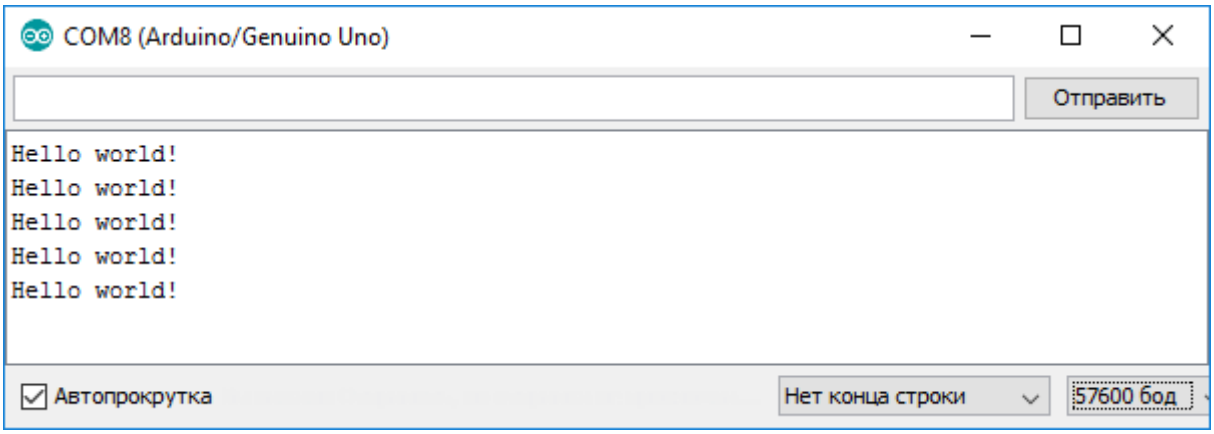


Рисунок 15.7 - Монітор послідовного порту

4.7 Написання програми на мові C for Arduino

4.7.1 Запустити програму IDE Arduino.

4.7.2 Набрати наступну програму:

```
void setup() {
   Serial.begin(57600);
}
void loop() {
   Serial.println("Hello world!");
   delay(2000);
}
```

4.7.3 Виконати компіляцію скетча

4.7.4 Завантажити програму на мові C for Arduino до моделі. Відслідкувати функціонування моделі.

4.7.5 Завантажити програму на мові C for Arduino до плати Arduino. За допомогою монітора послідовного порту відслідкувати передачу даних.

ВМІСТ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

1. Тема, мета роботи.

2. Блок-схема алгоритму функціонування мікроконтролера (складається самостійно) з поясненнями.

3. Повний текст програми на мові асемблера та вікно Atmel Studio з програмою на мові асемблера.

4. Програма на мові C for Arduino та вікно IDE Arduino з програмою.

5. Print Screen екрана з працюючою моделлю в Proteus.
6. Print Screen монітора послідовного порту.
7. Висновки.

КОНТРОЛЬНІ ПИТАННЯ

1. Які стандартні швидкості обміну прийняті для асинхронного режиму послідовної передачі даних?
2. Які передавачі і приймачі описує стандарт RS-232?
3. Який діапазон напруги, згідно стандарту RS-232, відповідає лог. «1» та лог. «0»?
4. Які виводи мікроконтролера використовує модуль USART?
5. Які регістри мікроконтролера AVR використовуються модулем USART?
6. Як налаштувати швидкість обміну даними USART?
7. Який формат кадру використовує USART?
8. Описати порядок передачі даних USART.
9. Описати порядок прийому даних USART.
10. Перелічити команди мови C for arduino для послідовної передачі даних.

ЛІТЕРАТУРА

Базова

1 Колонтаєвський Ю. П., Сосков А. Г. Електроніка та мікросхемо техніка : підручник. 2-е вид. К. : Каравела, 2009. 416 с.
2 Хоровиц П., Хилл У. Искусство схемотехники. Изд. 5-е, перераб. М. : Мир, 1998. 704 с.
3 Мікропроцесорна техніка : підручник / Ю. І. Якименко та ін., за ред. Т. О. Терещенко. 2-ге вид., перероб. та доповн. К. : ІВЦ «Видавництво «Політехніка»; «Кондор», 2004. 440 с.
4 Волович Г. И. Схемотехника аналоговых и аналого-цифровых электронных устройств. М. : Издательский дом «Додэка-XXI», 2005. 528 с.
5 Схемотехніка електронних систем: у 3 кн. Кн. 2. Цифрова схемотехніка: підручник / В. І. Бойко та ін. 2-ге вид., допов. і перероб. К. : Вища шк., 2004. 423 с.
6 Новиков Ю. В. Основы цифровой схемотехники. Базовые элементы и схемы. Методы проектирования. М. : Мир, 2001. 379 с.
7 Електротехніка та електроніка. Теоретичні відомості, розрахунки та дослідження за підтримкою комп'ютерних технологій : навч. посіб. / Щерба А. А. та ін. К. : "Корнійчук", 2007. 488 с.
8 Схемотехника электронных систем. Микропроцессоры и микроконтроллеры / В. И. Бойко и др. СПб.: БХВ-Петербург, 2004. 464 с.
9 Евстифеев А. В. Микроконтроллеры AVR семейств Tiny и Mega фирмы «ATMEL», 5-е изд., стер. М. : Издательский дом «Додэка-XXI», 2008. 560 с.
10 Петин В. А. Проекты с использованием микроконтроллера Arduino. СПб.: БХВ-Петербург, 2014. 400 с.

Допоміжна

1 Угрювом Е. П. Цифровая схемотехника : учеб. пособие для вузов. 2-е изд., перераб и доп. СПб. : БХВ-Петербург, 2005. 800 с.
2 Опадчий Ю. Ф., Глудкин О. П., Гуров А. И. Аналоговая и цифровая электроника (Полный курс): учебник для вузов. М.: Горячая линия – Телеком, 2002. 768 с.
3 Прянишников В. А. Электроника : курс лекций. СПб. : Корона-принт, 1998. 400 с.
4 Ветров И. Л. Электронные устройства на аналоговых и аналого-цифровых интегральных микросхемах : учеб. пособие. Севастополь: Изд-во СевНТУ, 2006. 282 с.
5 Гейтс Эрл. Д. Введение в электронику. Ростов-на-Дону: «Феникс», 1998. 640 с.

6 Лачин В. И., Савелов Н. С. Электроника : учебное пособие. Ростов-на-Дону: «Феникс», 2001. 438 с.
7 Новиков Ю. В. Основы цифровой схемотехники. Базовые элементы и схемы. Методы проектирования. М. : Мир, 2001. 379 с.
8 Ирвин Кип Язык ассемблера для процессоров Intel, 4-е изд.: пер. с англ. М.: Издательский дом «Вильямс», 2005. 912 с.
9 Зубков С. В. Assembler для DOS, Windows и UNIX. 3-е изд., стер. М.: ДМК Пресс; СПб. : Питер, 2005. 608 с.
10 Assembler. Учебник для вузов. 2-е вид. / В. И. Юров. СПб.: Питер, 2003. 637с.
11 Евстифеев А. В. Микроконтроллеры AVR семейства Classic фирмы «ATMEL», 3-е изд., стер. М. : Издательский дом «Додэка-XXI», 2006. 288 с.
12 Болл Стюарт Р. Аналоговые интерфейсы микроконтроллеров. М. : Издательский дом «Додэка-XXI», 2007. 360 с.
13 Редькин П. П. Микроконтроллеры Atmel архитектуры AVR32 семейства AT32UC3. Руководство пользователя. М. : Техносфера, 2010. 784 с.
14 Ревич Ю. В. Практическое программирование микроконтроллеров Atmel AVR на языке ассемблера. СПб. : БХВ-Петербург, 2008. 384 с.
15 Хофманн М. Микроконтроллеры для начинающих: Пер. с нем. СПб. : БХВ-Петербург, 2010. 304 с.
16 Нарышкин А. К. Цифровые устройства и микропроцессоры : учеб. пособие для студ. высш. уч. завед. М.: «Академия», 2006. 320 с.
17 Голубцов М. С. Микроконтроллеры AVR: от простого к сложному. М. : СОЛОН-Пресс, 2003. 288 с.
18 Трамперт В. AVR-RISC микроконтроллеры. Пер. с нем. К. : «МК-Пресс», 2006. 464 с.
19 Мортон Дж. Микроконтроллеры AVR. Вводный курс. / Пер. с англ. М. : Издательский дом «Додэка-XXI», 2006. 272 с.
20 Баранов В. Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы. М. : Издательский дом «Додэка-XXI», 2004. 288 с.
21 Белов А. В. Микроконтроллеры AVR в радиолюбительской практике. СПб. : Наука и Техника, 2007. 352 с.

Інформаційні ресурси

1 Документація на плати Arduino, система команд, навчальні матеріали. Режим доступу: http://arduino.ua/
2 Документація на 8- та 32-х розрядні мікроконтролери PIC та AVR. Режим доступу: https://www.microchip.com/
3 Навчальні матеріали та відеокурси з використання програми Proteus. Режим доступу: https://www.labcenter.com/