Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи візуалізації універсальної діагностики послідовного порту передачі даних”

Виконав здобувач вищої освіти
IV курсу, групи КІ-19
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Ламекін Н.В.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Коваленко А.С.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Кафедра_КбПЗ_2023рік

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*"*Інформаційні технології*"*_____

Спеціальність *123 "*Комп'ютерна інженерія*"*

Освітньо-професійна (освітньо-наукова) програма _*"*Комп'ютерна інженерія*"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Ламекіну Нікіті Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи візуалізації універсальної діагностики послідовного порту передачі даних*

2. Керівник роботи *Коваленко Анна Степанівна, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 7-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи візуалізації універсальної діагностики послідовного порту передачі даних*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

Кафедра КБПЗ – 2023рік

7. Дата видачі завдання « 17 » січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Коваленко А.С.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Ламекін Н.В.
(прізвище та ініціали)

Кафедра_КбПЗ_2023рік

АНОТАЦІЯ

Ламекін Н.В. Програмне забезпечення системи візуалізації універсальної діагностики послідовного порту передачі даних. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи візуалізації універсальної діагностики послідовного порту передачі даних.

Метою розробки є програмне забезпечення системи візуалізації універсальної діагностики послідовного порту передачі даних.

Результат роботи – програмна реалізація системи візуалізації універсальної діагностики послідовного порту передачі даних.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Builder C++.

Ключові слова: комп'ютерна інженерія, послідовний порт передачі даних

Кафедра_КОПЗ_2023рік

ABSTRACT

Lamekin N.V. Universal serial port diagnostics visualization system software. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for the visualization system of universal diagnostics of the serial data port.

The purpose of the development is the software of the visualization system of the universal diagnosis of the serial data transfer port.

The result of the work is the software implementation of the visualization system of universal diagnostics of the serial data transmission port.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Builder C++ environment.

Keywords: computer engineering, serial data transfer port

Кафедра_КОПЗ_2023рік

ЗМІСТ

Кафедра_КбПЗ_2023рік

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БА	–	базова адреса
ЕОМ	–	електронно-обчислювальна машина
ЛОМ	–	локальна обчислювальна мережа
ЗЗК	–	загальновійськовий захисний комплект
ЗФО	–	захисний фільтруючий одяг
ПЗ	–	програмне забезпечення
ЦО	–	цивільна оборона
BREAK	–	лінія в стані розриву зв'язку
BIOS	–	базова система введення/виведення
COM	–	послідовний порт
DCE	–	пристрій зв'язку
DTE	–	термінальний пристрій
IRQ	–	рівень переривання
MARK	–	вихідний стан лінії, рівень логічної 1
MMB	–	алгоритм шифрування
POST	–	Power On Self Testing
RS-232	–	стандарт послідовної передачі даних
SPACE	–	порожній стан лінії
START	–	стартовий біт
UART	–	універсальний асинхронний прийомопередавач
USB	–	універсальна послідовна шина
VCL	–	бібліотека візуальних компонентів

Кафедра_КБПЗ_2023рік

ВСТУП

Актуальність теми. Переважна більшість контролерів, що випускаються в наш час, мають вбудований послідовний порт, повністю сумісний по формату даних з COM-портом. Як і багато інтерфейсів, даний порт піддається небезпеці виходу з ладу в результаті некоректних дій оператора. Причиною несправності може послужити розряд статичної електрики. Небезпека пробою особливо зростає в зимовий період, коли повітря стає сухим від морозу. Іншою причиною може бути неакуратне з'єднання між собою двох незаземлених комп'ютерів, що мають різну напругу на корпусах.

У результаті перерахованих несприятливих впливів порт перестає нормально функціонувати, але все-таки піддається "лікуванню". Звичайно, як правило, виходить із ладу буферна мікросхема, що забезпечує перетворення рівнів сигналів ТТЛ у сигнали стандарту RS-232, які повинні перебувати в діапазоні ± 3...25 В. І буває дуже жаль втраченого часу, коли після довгого пошуку причини непрацездатності зовнішніх пристроїв, виявляється, що несправним був саме COM-порт комп'ютера.

За допомогою розробленого програмного забезпечення візуалізації універсальної діагностики послідовного порту передачі даних студенти легко зможуть навчитися робити діагностику послідовного порту та розробляти програмне забезпечення для його перевірки.

Розвиток засобів обчислювальної техніки відбувається в багатьох напрямках, що розширюють сферу застосування ЕОМ і підвищують ефективність їхнього використання. Але разом з тим збільшується обсяг знань, якими повинен володіти системний програміст.

Використання віртуальних лабораторних стендів дозволяє студентові краще зрозуміти принципи роботи та методи програмування цифрових пристроїв. До того ж на відміну від фізичних стендів, вони значно дешевші.

Кафедра_КОПЗ_2023рік

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи візуалізації універсальної діагностики послідовного порту передачі даних.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем візуалізації універсальної діагностики послідовного порту передачі даних.

– Дослідження системи візуалізації універсальної діагностики послідовного порту передачі даних.

– Програмна реалізація системи візуалізації універсальної діагностики послідовного порту передачі даних.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі візуалізації універсальної діагностики послідовного порту передачі даних.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи візуалізації універсальної діагностики послідовного порту передачі даних, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

Кафедра_КОПЗ_2023рік

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Послідовний порт або COM-порт (від англ. COMmunication port) – двунаправлений послідовний інтерфейс, призначений для обміну байтовою інформацією (рисунок 1.1). Послідовний тому, що інформація через нього передається по одному біту, біт за бітом (на відміну від паралельного порту). Найбільш часто для послідовного порту персональних комп'ютерів використовується стандарт RS-232C. Раніше послідовний порт використовувався для підключення термінала, пізніше для модему або миші. Зараз він використовується для з'єднання із джерелами безперебійного живлення, для зв'язку з апаратними засобами розробки вбудованих обчислювальних систем.

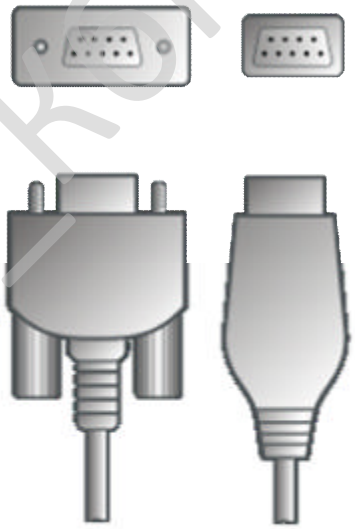

Рисунок 1.1 – Варіанти роз'ємів COM-порту типу DB-9F

Хоча деякі інші інтерфейси комп'ютера – такі як Ethernet, FireWire і USB – також використовують послідовний спосіб обміну, назва «послідовний порт» закріпилося за портом, що має стандарт RS-232C, і початково призначеним для

Кафедра КПЗ 2023рік

обміну інформацією з модемом.

За допомогою COM-порту можна з'єднати два комп'ютери, використовуючи так званий «нуль-модемний кабель».

Прийняте умовне позначення послідовного порту: 

Найбільш часто використовуються Д-подібні роз'єми: 9– та 25-контактні, DB-9 і DB-25 відповідно. Раніше використовувалися також DB-31 і круглі восьмиконтактні DIN-8.

Старі комп'ютери використовують 25-ти контактні роз'єми, але тільки 9 контактів реально задіяно на сьогоднішній день. За винятком двох проводів для передачі й прийому даних, інші використовуються для контролю й заземлення. Напруга на кожному з контактів і дротів виміряється щодо сигнальної землі. Тому мінімальна кількість дротів для двонаправленої передачі даних – 3. У рідких випадках для роботи може вистачити і двох дротів (без сигнальної землі), однак це може призвести до низької продуктивності, а іноді й до помилок при передачі даних.

Залишається ще декілька дротів, які призначені тільки для керування (контролю) і не використовуються для передачі даних. Всі ці сигнали могли б передаватися по одній лінії, але замість цього, для них виділені окремі дроти. Деякі (або усі разом) ці сигнальні лінії називаються "лінії стану модему". Лінії стану можуть перебувати в одному із двох станів встановленому (включений) +12 Вольт або скинутому (виключений) –12 Вольт. Одні із цих проводів сигналізують комп'ютеру про те, що потрібно припинити передачу даних через послідовний порт. Інші у свою чергу сигналізують пристрою, підключеному до послідовного порту, припинити передачу даних у комп'ютер. Якщо підключений пристрій – модем, то лінії, що залишилися, можуть вказувати модему на те, що потрібно зайняти телефонну лінію або сигналізують комп'ютеру про те, що з'єднання було встановлено або що є дзвінок на телефонній лінії (значить хтось намагається з'єднатися з комп'ютером). Максимальна швидкість передачі по послідовному порту звичайно становить 115200 біт/с.

Кафедра_КОПЗ_2023рік

1.2 Область застосування

Розроблене програмне забезпечення можна використовувати для навчання студентів системному програмуванню та діагностиці послідовного порту передачі даних. Даний емулятор можна використовувати для виконання лабораторних робіт по дисциплінам: «Архітектура ЕОМ» та «Системне програмування».

Якість навчання, одержаного студентом, визначається не тільки рівнем теоретичної підготовки, але й умінням використовувати отримані знання на практиці. Практичні навики студент здобуває через лабораторні роботи, та семінарські заняття. Ефекти, що досягаються при цьому: скорочення часу на освоєння матеріалу, підвищення рівня розуміння теоретичних положень. Для підвищення рівня практичної підготовки наукових і технічних фахівців в останні роки все більше застосування знаходять інформаційні технології, зокрема, при організації лабораторних практикумів.

Лабораторні стенди, що припускають фізичне моделювання технологічного процесу навіть у спрощеному вигляді, означають серйозні фінансові витрати на їхнє створення й підтримку в працездатному стані. При такому підході необхідно розробити й реалізувати не тільки модель технологічного процесу, але також оснастити її датчиками, виконавчими органами й забезпечити сполучення з комп'ютером через модулі вводу-виводу. Вартість створення такого стенда часто дуже висока. Стенди громіздкі, займають багато місця, а помилки, що допускаються студентами при роботі з ними, найчастіше призводять до виходу стенда з ладу. У цих умовах найбільш перспективний спосіб організації практичних занять для студентів повинен бути заснований на використанні програмних імітаторів – віртуальних лабораторних стендів. У порівнянні з фізичними моделями, емулятори мають ряд очевидних переваг, що дають істотне скорочення матеріальних і часових витрат на створення, тиражування й супровід лабораторних стендів.

Кафедра_КОПЗ_2023рік

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи візуалізації універсальної діагностики послідовного порту передачі даних, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

Кафедра_КбПЗ_2023рік

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Серед існуючих програм найбільш близьких до теми даної бакалаврської роботи є емулятори послідовного порту та програми діагностики реального COM-порту, але немає програм, що ємулюють тестування. Розглянемо існуючі програми.

Test-rs – програма для тестування послідовних портів з 9 і 25-контактними роз'ємами. Працює під керуванням операційної системи Windows9X/ME.

Після запуску програма робить пошук наявних у комп'ютері COM-портів і вибирає для роботи перший знайдений один за одним. Якщо порти в комп'ютері не виявлені, буде видано відповідне повідомлення й програма завершить свою роботу. У процесі роботи програми можна вибрати інший доступний комп'ютеру порт для тестування. Змінити параметри налаштування порту можна стандартними засобами по шляху: Мій комп'ютер−>Властивості−>Пристрої−> COM та LPT-порти−>Послідовний порт−>Властивості−>Налаштування порту.

Після запуску програми на екрані монітора з'явиться вікно, представлене на рисунку 2.1.

Кафедра_КОПЗ_2023рік

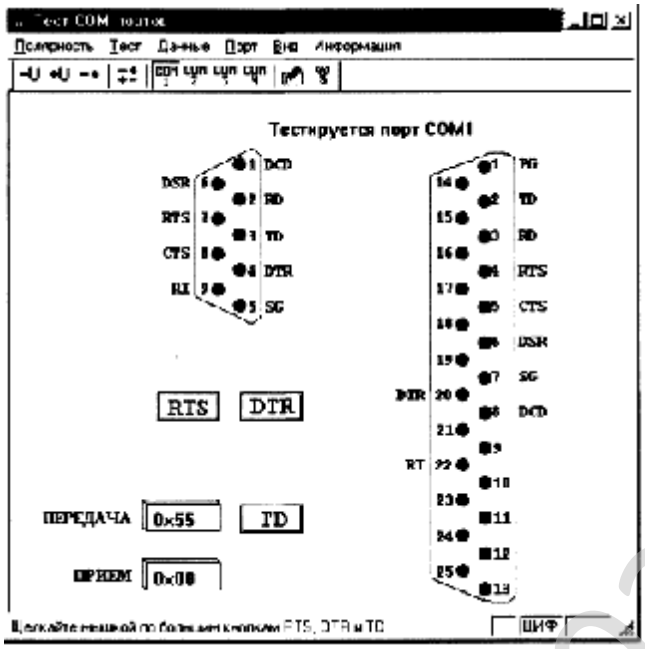

Рисунок 2.1 – Програма Test-rs

У центрі вікна програми зображені роз'єми комп'ютера із призначенням контактів і сигналів COM-порту. Для зручності тестування приводяться два малюнки, для 9 і 25-контактних роз'ємів відповідно.

Номера контактів роз'ємів, їхній тип і призначення сигналів відповідають стандартним і загальноприйнятим для даних портів. Надалі при роботі програми контакти роз'ємів будуть пофарбовані в червоний або зелений колір, що відповідає позитивному або негативному сигналу на них.

Нижче роз'ємів намальовані дві кнопки для сигналів RTS і DTR. Клацаючи лівою клавішею мишки по цих кнопках можна міняти рівень відповідних вихідних сигналів порту на протилежний. Дані операції будуть супроводжуватися автоматичною зміною кольорів контактів на роз'ємах.

Третя кнопка TD призначена для початку передачі даних через порт. Вміст переданих даних представлений в полі "Передача" і може бути змінений після клацання лівою кнопкою миші по цьому полю або за допомогою кнопок, розташованих у меню програми. Прийняті по порту дані відображаються в полі "Прийом".

Кафедра_КбПЗ_2023рік

Для перевірки працездатності порту можна скористатися звичайним мультиметром і контролювати по черзі тестируємі сигнали. Але це не зовсім зручно, тим більше що роз'єми розташовані на тильній стороні комп'ютера, контакти являють собою штирьки, а не гнізда, і можливе їхнє випадкове замикання один з одним.

Тому пропонується використовувати для перевірки невеликий й дуже простий пристрій – пробник. Схема цього пристрою наведена на рисунку 2.2.

Зі схеми пробника видно, що всі сигнали об'єднані в три різні групи. У кожній із груп присутній тільки один вихідний сигнал і від одного до трьох вхідних сигналів.

Пробник складається з одного роз'єму-розетки DB-9F і декількох світлодіодів і резисторів, розпаяних на невеликій макетній платі, з'єднаних між собою шлейфом довжиною біля метра.

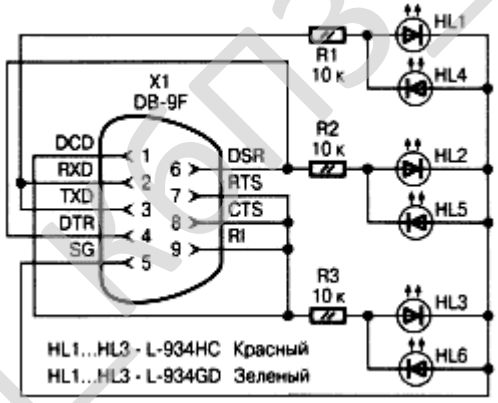

Рисунок 2.2 – Схема пристрою для перевірки COM-порту

Пробник після підключення можна покласти на стіл поруч із оператором і забезпечити тим самим зручність спостереження за процесом тестування порту. Стан будь-якого сигналу видно по світлодіоду, що світиться. Колір світодіоду вказує на полярність сигналу.

Порівнюючи ці сигнали з відображуваними сигналами в програмі, можна легко визначити несправну лінію. Ланцюги передавача й приймача порту замикаються пробником між собою, тому передані дані будуть прийматися цим

Кафедра ЕОПЗ 2023рік

же портом. Порівнюючи передані й прийняті дані, можна перевірити прийомопередатчик.

Код виконуючої програми складається з одного файлу test-rs.exe і не вимагає попередньої установки. З метою зменшення розміру даного файлу програма транслювалася в режимі Shared DLL, тобто стандартні бібліотечні файли DLL не включалися в тіло програми, а використовувалися після запуску програми з доступного системного каталогу C:\WINDOWS\SYSTEM.

Програма test-rs використовує наступні стандартні бібліотечні файли: MFC42.DLL, KERNEL32.DLL, GDI32.DLL, USER32.DLL і MSVCRT.DLL. Тому у випадку, якщо програма видасть повідомлення про відсутність деяких з перерахованих бібліотечних файлів DLL, необхідно помістити їх у зазначений вище каталог.

Ці файли є стандартними й встановлюються самою операційною системою Windows9X. Їх можна також знайти в Інтернеті.

COM Port Stress Test – інструмент тестування COM/RS232 портів. Програма генерує високошвидкісний потік зі змінними характеристиками з'єднання. Головне вікно програми зображене на рисунку 2.3.

COM Port Stress Test це інструмент для "твердого" екстремального тестування COM і RS232 портів (пристроїв), що формує потік даних з випадковими параметрами зв'язку й даними. Даний тест дозволяє відмінно перевірити роботу пристроїв на стійкість і обробку ними помилок, що виникають під великим навантаженням, які могли б ніколи не виявитися в нормальних умовах у додатках користувача і системах. Ціль цього тесту це перевірка того, що додаток або пристрій не "впаде" або перестане працювати в умовах високого навантаження й випадкових даних.

Кафедра_КБПЗ_2023рік

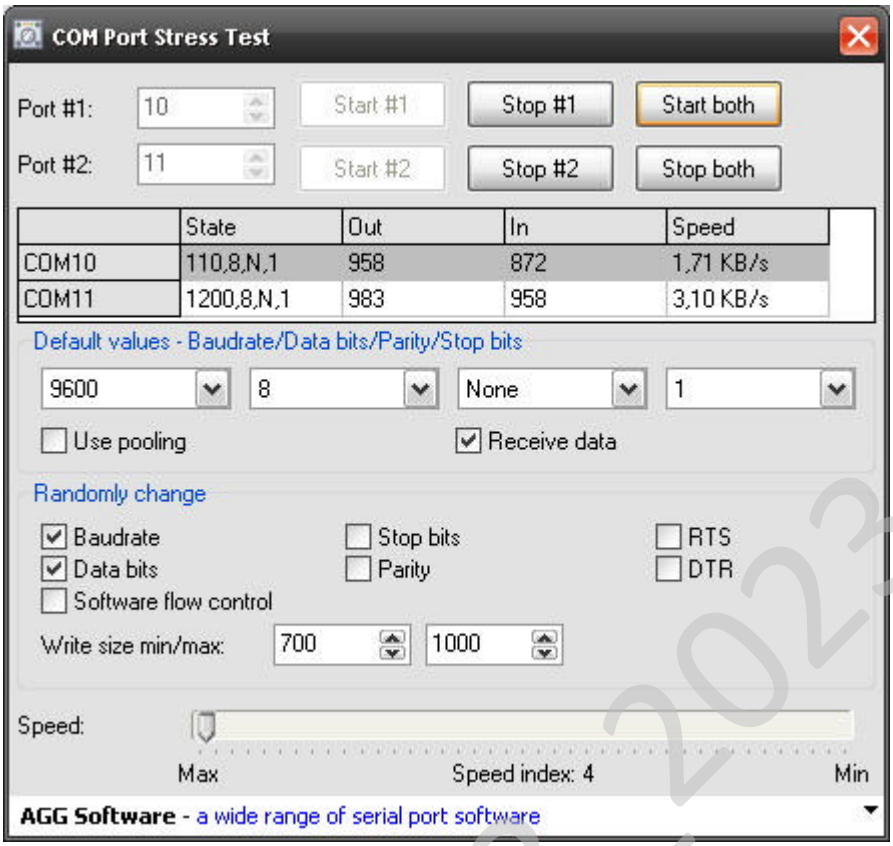

Рисунок 2.3 – Програма COM Port Stress Test

COM Port Stress Test може допомогти розроблювачам, експертам і обслуговуючому персоналу перевірити устаткування, програми або цілі системи в екстремальних умовах. Ця програма генерує дуже щільний потік даних, довільно міняє швидкість передачі, параметри контролю передачі даних, стан сигналів RTS і DTR та інші параметри зв'язку.

Дана програма ідеально підходить для рішення широкого кола завдань у різноманітних областях діяльності: розробка й тестування програмного забезпечення, служб технічної підтримки, тестування телекомунікаційних додатків, системна інтеграція. Скрізь, де існує ймовірність роботи програм або систем із критичними навантаженнями або в позаштатних ситуаціях.

Увага! Цей тест може викликати крах системи. Тому, будь ласка, закрийте всі програми й зробіть резервну копію важливих файлів перед її запуском.

Після установки програми можна знайти іконку програми на робочому

Кафедра_КбПЗ_2023рік

столі або в меню Пуск->Програми, і запустити COM Port Stress Test. Потім слід вибрати номер одного або двох COM-портів, параметри зв'язку за замовчуванням, налагодити параметри тестування й натиснути кнопку "Старт".

Ключові особливості:

– Універсальність. COM Port Stress Test підтримує порти COM, RS232, RS485 (з конвертором);

– Віртуальні COM порти. Програма може працювати з віртуальними портами, usb COM-портами, драйверами віртуальних послідовних портів;

– До двох портів. Ця програма може тестувати два RS232 або COM-порти одночасно;

– Читання/запис. COM Port Stress Test може як читати дані з порту, так і робити запис у послідовний порт;

– Гнучке налаштування. Програма може випадковим чином міняти параметри зв'язку (швидкість передачі, кількість біт даних, кількість стопових біт, парність, контроль передачі даних);

– RTS/DTR. Програма також може випадковим чином міняти стан сигналів RTS або DTR;

– Розмір буфера запису. Можна встановити розмір записуваних даних в одиницю часу;

– Регульоване навантаження. Можна регулювати рівень завантаження портів під час тесту;

– Візуалізація. Програма відображає швидкість прийому й передачі даних для обох портів.

COM Port Data Emulator – це інструмент для емуляції пристрою, підключеного до послідовного порту або мережі Ethernet, що генерує який-небудь потік послідовних даних. Програма може формувати потік даних, перетворювати потік даних у пакети даних для портів RS232, TCP/IP або UDP і відправляти їх через обраний порт. Приклад роботи програми зображений на рисунку 2.4.

Кафедра_КБПЗ_2023рік

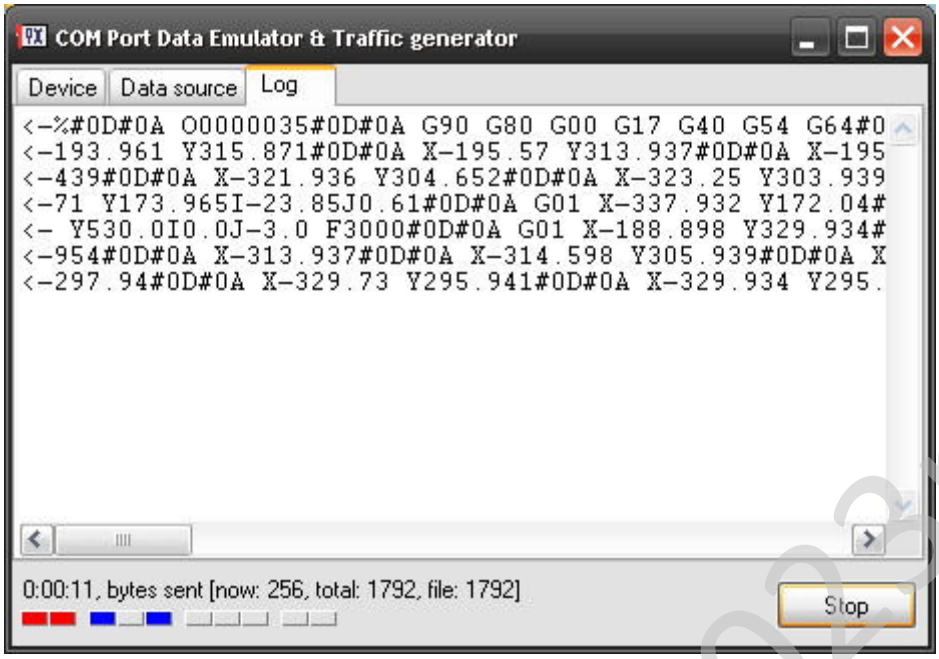

Рисунок 2.4 – Програма COM Port Data Emulator

COM Port Data Emulator може допомогти розроблювачам програмного забезпечення в тестуванні їхніх програм. Ця програма може замінити деякі рутинні операції, які, можливо, системний адміністратор виконує щодня.

Ця програма може читати потік даних з текстового або двійкового файлу або створювати довільний потік даних. Можна відсилати дані один раз або через заданий інтервал.

COM Port Data Emulator ідеально підходить для рішення широкого кола завдань у різноманітних областях діяльності: розробка й тестування програмного забезпечення, мережних адміністраторів інженерів, служб технічної підтримки, користувачів SCADA і телеметрії, тестування телекомунікаційних додатків, системна інтеграція. За допомогою цієї програми можна відтворювати передачу даних реальних пристроїв з лог-файлу й тестувати програмне забезпечення без підключення реального пристрою до комп'ютера. Також можна генерувати випадковий потік даних і тестувати стійкість системи та ПЗ в даній стресовій ситуації.

Після установки програми можна знайти іконку програми на робочому

Кафедра_КоПЗ_2023рік

столі або в меню Пуск->Програми, і запустити COM Port Data Emulator. Потім необхідно вибрати номер COM-порту, параметри зв'язку, настроїти джерело даних і натиснути кнопку "Старт". Після цього програма почне генерувати потік даних.

Ключові особливості:

– Універсальність. COM Port Data Emulator підтримує порти COM, RS232, RS485 (через конвертор), TCP/IP, UDP;

– Підтримка повного дуплекса. Програма може одночасно відправляти й приймати дані (крім RS485);

– Гнучка конфігурація. Емулятор підтримує довільні швидкості передачі (до 115200), кількість біт даних, кількість стопових біт, різні типи парності, контроль передачі даних і т.п.;

– Контроль передачі даних. Підтримується програмний і апаратний контроль передачі даних COM і RS232 портів;

– Клієнт/сервер. Ця програма може працювати як клієнт і як сервер у мережі TCP/IP;

– Довільне джерело даних. Емулятор може читати потік даних з текстового або двійкового файлу;

– Протокол. COM Port Data Emulator може створювати протокол даних, що відсилаються, для подальшого їхнього порівняння на приймаючій стороні;

– Візуалізація. Програма відображає стан лінії (для COM і RS232) і помилки передачі даних, відображає відправлені й прийняті дані.

Virtual null modem. Призначення нуль-модемного кабелю – дозволити двом RS-232 пристроям типу "DTE" обмінюватися даними між собою без використання додаткового устаткування й модемів (тобто, пристроїв типу "DCE"). **Virtual Null Modem** це утиліта, що емулює одну або декілька пар портів RS232, з'єднаних між собою нуль-модемним кабелем. Іншими словами, можна створити безліч віртуальних послідовних портів, які практично не будуть відрізнятися від справжніх портів (наприклад, COM10, COM11, COM127 і т.д.), з'єднані попарно

Кафедра_КбПЗ_2023рік

віртуальним нуль-модемним кабелем. Приклад роботи програми зображений на рисунку 2.5.

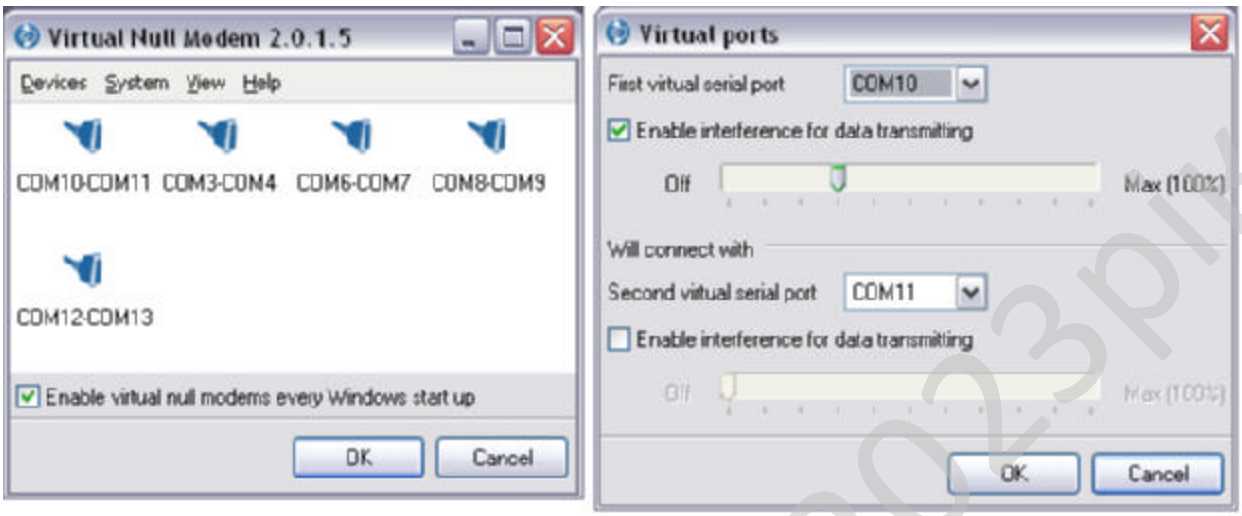

Рисунок 2.5 – Програма Virtual null modem

Припустимо, що у Вас є дві програми, які Ви хочете з'єднати між собою. Звичайно необхідно два справжніх послідовних порти, з'єднаних нуль-модемним кабелем. Сучасні комп'ютери часто мають у наявності тільки один порт, а ноутбуки не мають жодного. При використанні програми **Virtual Null Modem** можна емулювати нуль-модемне з'єднання, таким чином що дві програми на одному комп'ютері можуть спілкуватися між собою без додаткового устаткування й заліза.

Наприклад, Ви можете створити віртуальні порти: COM10, COM12 і використовувати їх у двох різних програмах. Усе, що відішле програма #1 у порт COM10, буде прийнято програмою #2 з COM12, і навпаки.

Після інсталяції програми Virtual Null Modem, її слід запустити за допомогою іконки в меню "Пуск–>Програми". Програма запропонує створити новий пристрій віртуального нуль-модему. Треба відповісти "Так" і вибрати номери портів у діалоговому вікні. Потім натиснути на кнопці "OK" у діалоговому вікні й головному вікні програми. Тепер програма встановить і настроїть віртуальний пристрій, і можна користуватися ним.

Кафедра_КОПЗ_2023рік

Ключові особливості:

– Не потрібне перезавантаження. Можна створювати, встановлювати й видаляти віртуальні послідовні порти в будь-який час без перезавантаження комп'ютера.

– Варіанти. Можна створювати безліч віртуальних пар портів, які можуть бути з'єднані за різними схемами (з повним контролем передачі, без контролю передачі або частковим контролем передачі).

– Незалежні віртуальні порти. Не потрібне ніяке залізо, ніякі апаратні ресурси не будуть використовуватися при створенні віртуальних портів. Навіть взагалі можна не мати реальних послідовних портів у системі;

– 100 % емуляція. Інші програми ніколи не побачать розходження між віртуальним або реальним послідовним портом, створеними засобами Virtual Null Modem;

– Продуктивність. Передача даних між двома віртуальними портами набагато швидша й залежить тільки від продуктивності комп'ютера;

– Надійніше, ніж залізо. Віртуальний зв'язок більш надійний та стійкий, ніж при використанні фізичної лінії;

– Надійне ПЗ. Драйвери віртуальних портів протестовані з різноманітними операційними системами, при використанні засобів тестування, які були надані компанією Microsoft (HCT 12.10 і verifier);

– Високотехнологічність. Virtual Null Modem використовує драйвер ядра й підтримує: WDM, WMI, Power Management, PNP і т.п.;

– Просто у використанні. Virtual Null Modem має простий і зрозумілий інтерфейс для створення й редагування параметрів віртуальних пристроїв;

– Додаткові можливості. Якщо Ви розроблювач програмного забезпечення, то Ви можете включити симуляцію втрати даних при передачі, що дозволить протестувати вашу програму в умовах, наближених до реальності;

– Розширюваність та інтегрованість. Можна додавати й видаляти віртуальні порти, викликом програми з командного рядка;

Кафедра_КОПЗ_2023рік

– Контекстна довідка. Можна використовувати F1 у будь-якому місці цієї програми й одержати вичерпну інформацію з використання програми.

Eltima Virtual Serial Port Driver – потужний програмний пакет для створення будь-якої кількості віртуальних пар послідовних портів, з'єднаних через віртуальний нуль-модемний кабель без використання реальних COM-портів (рисунок 2.6).

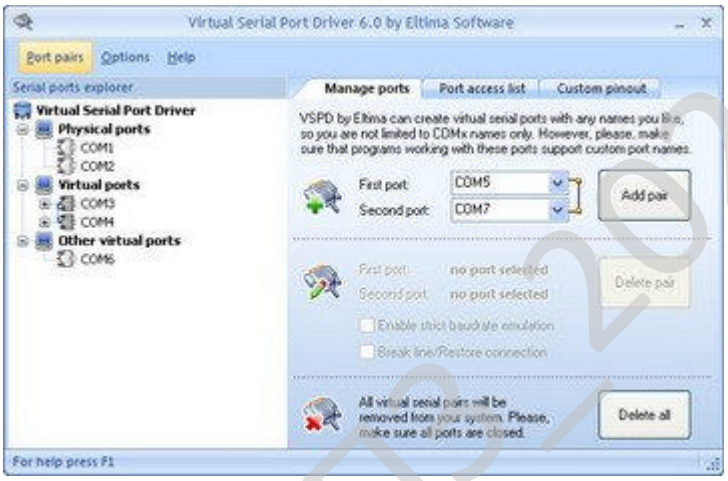

Рисунок 2.6 – Програма Eltima Virtual Serial Port Driver

Наприклад, можна створити віртуальну пару портів: COM5 і COM6 і з'єднати два різних додатки через них і здійснювати обмін даними між ними використовуючи віртуальні COM5 і COM6 порти.

Virtual Serial Port визначається будь-якою операційною системою або додатком як «стандартний» реальний послідовний порт.

Advanced Virtual COM Port – це перша програма, що включає як локальні, так і мережні функції віртуального порту COM.

Кафедра_КПТ_2023рік

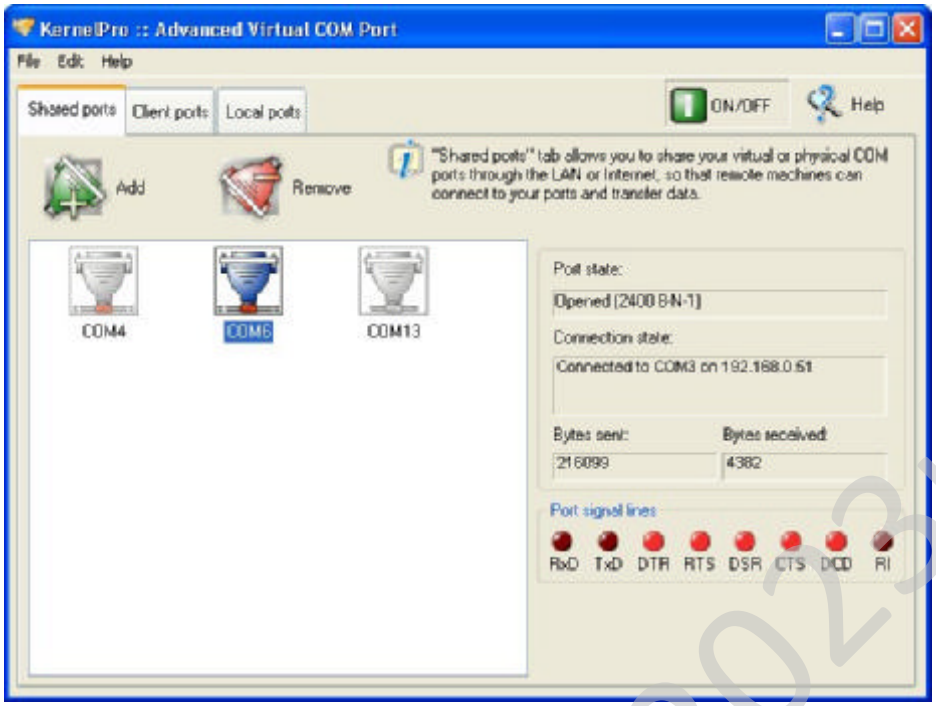

Рисунок 2.7 – Програма Virtual COM Port

Програма може працювати з реальними COM-портами або створювати віртуальні COM-порти і приєднуватися до них за допомогою віртуального кабелю модему або через мережу TCP/IP або через Інтернет.

Можна приєднуватися до розподіленого порту й використовувати його, начебто він є на Вашій машині. Віртуальні серійні порти, що створюються програмою виглядають і працюють так само, як і реальні.

Основні характеристики програми:

– віртуальні COM-порти з'являються в системі й додатках, як реальні порти;

– віртуальні COM-порти працюють як реальні;

– сумісність із 64-бітним процесом (AMD64);

– «гаряче» створення віртуального COM-порту і його видалення (немає необхідності перезапускати комп'ютер);

– створені віртуальні COM-порти існують доти, поки їх не видалять;

– немає необхідності мати фізичні серійні порти;

Кафедра_КБПЗ_2023рік

– більш висока швидкість передачі даних, ніж у серійних портів;

– можна створити до 255 віртуальних COM-портів;

– підтримка Pn, WMI;

– підтримка емулювання потоку;

– простота в моніторингу стану порту;

– можливість тимчасово видаляти всі порти, не втрачаючи їхніх настройок.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Оскільки потрібно розробити просту та легку у користуванні програму, яка б виконувалась під операційною системою Windows, то для її реалізації я обрав Builder C++. Існує велике число бібліотек написаних під Builder C++, тому це одна з важливих причин вибору мови програмування. Середовище Builder C++ досить просте в користуванні, його вихідний код значно менше по об'єму в порівнянні з Delphi чи деякими іншими програмами такого типу. Досить легко організувати взаємодію між модулями програм, об'єктно-орієнтований підхід дає можливість значно скоротити код програми, а отже і час його виконання.

На заміну старого розробленого набору елементів управління у Builder C++ інтегрована бібліотека візуальних компонентів VCL, представлених на палітрі компонентів. Після переносу на форму методом перетягування (drag-and-drop) компоненти відразу становляться діючими об'єктами вашої програми. Окрім типізованих інтерфейсних елементів Windows (кнопки, смуги прокручування, редагуємі текстові області, прості та комбіновані списки, та інше) у бібліотеку включені елементи підтримки діалогових вікон, обслуговування баз даних та багато іншого. Можливо не тільки модифікувати поведінку існуючих компонентів, але і будувати нові.

Кафедра_КОПЗ_2023рік

Builder C++ підтримує останні розширення стандарту мови C++ та забезпечує швидку компіляцію та складання 32-розрядних програм для Windows. Результуючі програми оптимізовані з точки зору швидкості виконання програм та затрат пам’яті. Зручний відладгоджувальник (з асемблерним вікном, можливістю крокового виконання, завдання точок зупинки, трасування та інше) повністю інтегрований у систему проектування. Дизайнер форм, редактор коду, інспектор об’єктів та інші інструменти зостаються доступними під час виконання програми, саме через це вносити зміни до коду можна прямо у процесі відлагодження.

Дизайнер форм, Інспектор об'єктів і інші засоби залишаються доступними під час роботи програми, тому вносити зміни можна в процесі відлагодження.

Builder C++ поставляється в трьох варіантах: Standard (стандартний), Professional (для професіоналів розробників, орієнтованих на мережеву архітектуру) і Client/Server Suite (для розробки систем в архітектурі клієнт/сервер). Останні два варіанти доповнюють стандартний початковими текстами візуальних компонентів, різномасштабним словником даних, новими функціями мови запитів SQL для бази даних, пакетом підтримки систем Internet, службою моніторингу програм, а також рядом інших засобів.

Builder C++ підтримує зв'язок з різними базами даних 3-х видів: dBASE і Paradox: Sybase, Oracle, InterBase і Informix; Excel, Access, FoxPro і Btrieve. Механізм BDE (Borland Database Engine) додає обслуговуванню зв'язків з базами даних дивовижну простоту і прозорість. Провідник Database Explorer дозволяє зображати зв'язки і об'єкти баз даних графічно. Використовуючи компоненти баз даних, я побудував електронний записник згідно таблиці dBASE за півгодини роботи на комп'ютері. Спадкоємство готових форм і їх "підгонка" під специфічні вимоги помітно скорочують тимчасові витрати на вирішення подібних завдань.

Довідкова служба Builder C++ надавала мені допомогу в цій і багатьох інших подібних ситуаціях. Є повний опис кожного управляємого компонента, включаючи списки властивостей і методів, а також численні приклади. Виклад

Кафедра_КОПЗ_2023рік

матеріалу в книзі був значно покращуваний і систематизований завдяки відомостям, почерпнутим мною з довідкової служби.

Завдяки засобам управління проектами, двосторонній інтеграції застосунку і синхронізації між засобами візуального і текстового редагування, а також вбудованому відладнику (з асемблерним вікном прокрутки, покрокового виконання, точок останову, трасуванням і тому подібне) – Builder C++ корпорації Borland надає собою вражаюче середовище розробки, яка, мабуть, витримає конкурентну боротьбу з такими модними продуктами як Developer Studio фірми Microsoft.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи візуалізації універсальної діагностики послідовного порту передачі даних.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

Кафедра_КБПЗ_2023рік

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

Кафедра_КбПЗ_2023рік

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Опис послідовного порту. Стандарт RS-232C

Комп'ютер може мати до чотирьох портів послідовної передачі даних. Ці порти розташовані або на материнській платі, або на окремій платі, що вставляється в слоти розширення материнської плати. Бувають також плати, що містять вісім портів послідовної передачі даних. Їх часто використовують для підключення декількох комп'ютерів або терміналів до одного, центрального комп'ютера. Ці плати мають назву "мультипорт". В основі послідовного порту передачі даних лежить мікросхема Intel 8250 або її сучасні аналоги – Intel 16450, 16550, 16550A. Ця мікросхема є універсальним асинхронним прийомопередавачем (UART – Universal Asynchronous Receiver Transmitter). Мікросхема містить кілька внутрішніх регістрів, доступних через команди введення/виведення. Мікросхема 8250 містить регістри передавача й приймача даних. При передачі байта він записується в буферний регістр передавача, звідки потім переписується у зсувовий регістр передавача. Байт "висувається" зі зсувового регістра по бітах. Аналогічно є зсувовий і буферний регістри приймача.

Програма має доступ тільки до буферних регістрів, копіювання інформації в зсувові регістри й процес зсуву виконується мікросхемою UART автоматично. До зовнішніх пристроїв асинхронний послідовний порт підключається через спеціальний роз'єм. Існує два стандарти на роз'ємів інтерфейсу RS-232C, це DB25 і DB9. Перший роз'єм має 25, а другий 9 виводів (рисунок 3.1).

Рисунок 3.1 – Роз'єми послідовного порту DB25 та DB9

Кафедра_КОПЗ_2023рік

Призначення контактів паралельного порту наведені у таблицях 3.1 та 3.2.

Таблиця 3.1 – Призначення контактів послідовного порту стандарта DB25

Номер контакту	Скорочення	Призначення контакту	Вхід або вихід комп'ютера
1	FG	Захисне заземлення	-
2	TD (TXD)	Передані дані	Вихід
3	RD (RXD)	Прийняті дані	Вхід
4	RTS	Запит для передачі	Вихід
5	CTS	Скидання для передачі	Вхід
6	DSR	Готовність даних	Вхід
7	SG	Сигнальне заземлення	-
8	DCD	Детектор прийнятого з лінії сигналу	Вхід
9	–	Позитивна контрольна напруга	Вхід
10	–	Негативна контрольна напруга	Вхід
11	QM	Режим вирівнювання	Вхід
12	SDCD	Виявлення несучих вторинних даних	Вхід
13	SCTS	Вторинне скидання передачі	Вхід
14	STD	Вторинні передані дані	Вихід
15	TC	Синхронізація передавача	Вхід
16	SRD	Вторинні прийняті дані	Вхід
17	RC	Синхронізація приймача	Вхід
18	DCR	Розділена синхронізація приймача	Вихід
19	SRTS	Вторинний запит передачі	Вихід
20	DTR	Готовність вихідних даних	Вихід
21	SQ	Якість сигналу	Вхід
22	RI	Індикатор виклику	Вхід
23	–	Селектор швидкості даних	-
24	TC	Зовнішня синхронізація передавача	Вихід
25	–	Зайнятість	Вихід

Кафедра_КОПЗ_2023рік

Таблиця 3.2 – Призначення контактів послідовного порту стандарта DB9

Номер контакту	Скорочення	Призначення контакту	Вхід або вихід комп'ютера
1	DCD	Детектор прийнятого з лінії сигналу	Вхід
2	RD	Прийняті дані	Вхід
3	TD	Передані дані	Вихід
4	DTR	Готовність вихідних даних	Вихід
5	SG	Сигнальне заземлення	-
6	DSR	Готовність даних	Вхід
7	RTS	Запит для передачі	Вихід
8	CTS	Скидання для передачі	Вхід
9	RI	Індикатор виклику	Вхід

Тільки два виводи цих роз'ємів використовуються для передачі й прийому даних. Інші передають різні допоміжні й керуючі сигнали. На практиці для приєднання того або іншого пристрою може знадобитися різна кількість сигналів. Інтерфейс RS-232C визначає обмін між пристроями двох типів: **DTE** (Data Terminal Equipment – термінальний пристрій) і **DCE** (Data Communication Equipment – пристрій зв'язку). У більшості випадків, але не завжди, комп'ютер є термінальним пристроєм. Модеми, принтери, графобудівники завжди є пристроями зв'язку.

Сигнали інтерфейсу RS-232C підрозділяються на наступні класи:

1. Послідовні дані (наприклад, *TXD, RXD*). Інтерфейс RS-232C забезпечує два незалежних послідовних канали даних: первинний (головний) і вторинний (допоміжний). Обидва канали можуть працювати в дуплексному режимі, тобто одночасно здійснюють передачу й прийом інформації.

2. Керуючі сигнали квитування (наприклад, *RTS, CTS*). Сигнали квитування – засіб, за допомогою якого обмін сигналами дозволяє *DTE* почати

Кафедра_КОПЗ_2023рік

діалог з *DCE* до фактичної передачі або прийому даних по послідовній лінії зв'язку.

3. Сигнали синхронізації (наприклад, *TC*, *RC*). У синхронному режимі (на відміну від більш розповсюдженого асинхронного) між пристроями необхідно передавати сигнали синхронізації, які спрощують синхронізм прийнятого сигналу з метою його декодування.

На практиці допоміжний канал RS-232C застосовується рідко, і в асинхронному режимі замість 25 ліній використовуються тільки 9 ліній.

Базові адресі контролерів послідовного інтерфейсу залежно від номера контролера розташовуються у сегменті даних BIOS і наведена у таблиці 3.3.

Таблиця 3.3 – Базові адреси контролерів послідовного інтерфейсу

Номер контролера	Адреса в сегменті BIOS	Номер переривання
COM1	0040:0000	IRQ4 (INT 0Ch)
COM2	0040:0002	IRQ3 (INT 0Bh)
COM3	0040:0004	Не фіксований
COM4	0040:0006	Не фіксований

Базові адреси контролерів заносяться в сегмент даних BIOS програмою POST (Power On Self Testing) при перевірці після включення електроживлення. Програма POST поміщає базові адреси контролерів послідовно один за іншим. Це означає, що між значущими полями не може бути нульового поля.

Розглянемо докладно призначення й вміст регістрів контролера послідовного інтерфейсу.

До складу контролера послідовного інтерфейсу входять наступні регістри:

1. Регістри буферів приймача й передавача.
2. Регістри дозволу й ідентифікації переривань.
3. Регістри керування й стану лінії.
4. Регістри керування й стану модему.
5. Регістри буфера дільника генератора.

Кафедра_КОПЗ_2023рік

У таблиці 3.4 наведені адреси всіх програмно доступних регістрів. Адреси в цій таблиці дані щодо базової адреси контролера.

Також у таблиці наведене значення біта DLAB регістра LCR, що управляє адресацією регістрів. Саме цей біт уможливлює доступ до різних регістрів контролера через порти з одною адресою. У наведеній таблиці в графі «DLAB» стоїть символ «X», якщо для адресації відповідного регістра стан даного біта несуттєвий.

Таблиця 3.4 – Адреси регістрів контролера послідовного інтерфейсу

Адреса	Операція	Регістр	DLAB
0	W	Буфер передавача (THR)	0
0	R	Буфер приймача (RBR)	0
0	R\W	Молодший байт буфера дільника (Division Latch LSB)	1
1	R\W	Старший байт буфера дільника (Division Latch MSB)	1
1	R\W	Регістр дозволу переривання (IER)	0
2	R	Регістр ідентифікації переривання(IIR)	X
3	R\W	Регістр керування лінією (LCR)	X
4	R\W	Регістр керування модемом (MCR)	X
5	R	Регістр стану лінії (LSR)	X
6	R	Регістр стану модему (MSR)	X
7	R\W	Невикористовуваний регістр (Scratch Register)	X

Регістр буфера передавача (THR). Має адресу 0 щодо базової адреси контролера. Даний регістр доступний тільки під час запису й при значенні біта дозволу доступу до дільника (DLAB) у регістрі керування лінією (LCR), рівному 0. Регістр THR містить вісім біт даних (біт 0 є молодшим значущим розрядом і посилається першим у канал передачі).

Регістр буфера приймача (RBR). Має адресу 0 щодо базової адреси контролера. Цей регістр доступний під час читання (IN) і при значенні біта дозволу доступу до дільника (DLAB) у регістрі керування лінією (LCR), рівному

Кафедра_КОТЗ_2023рік

0. Регістр RBR містить вісім біт даних (біт 0 є молодшим значущим розрядом і приймається першим з каналу передачі).

Регістр буфера молодшого байта дільника (Divisor Latch LSB). Регістр має адресу 0 щодо базової адреси контролера. Цей регістр доступний і під час читання, і під час запису тільки при значенні біта дозволу доступу до дільника (DLAB) у регістрі керування лінією (LCR), рівному 1. При записі в цей регістр нового значення дільник перезавантажується негайно.

Регістр буфера старшого байта дільника (Divisor Latch MSB). Регістр має адресу 1 щодо базової адреси контролера. Цей регістр доступний по читанню й запису тільки при значенні біта дозволу доступу до дільника (DLAB) у регістрі керування лінією (LCR), рівному 1. При записі в цей регістр нового значення дільник відразу перезавантажується.

Регістр дозволу переривань (IER). Має адресу 1 щодо базової адреси контролера. Цей регістр доступний по читанню й запису, але тільки при значенні біта дозволу доступу до дільника (DLAB) у регістрі керування лінією (LCR), рівному 0. Цей регістр дозволяє управляти чотирма типами переривань, породжуваними контролером послідовного інтерфейсу. Формат регістра наведений на рисунку 3.2.

7	6	5	4	3	2	1	0
0	0	0	0	**ICM**	**ICL**	**IFB**	**IDA**

Рисунок 3.2 – Формат регістра дозволу переривань (IER)

ICM задає переривання при зміні стану модема:

1 – переривання виробляється;

0 – переривання заборонене.

ICL визначає переривання при зміні стану лінії приймача:

1 – переривання виробляється;

0 – переривання заборонене.

Кафедра_КОПЗ_2023рік

IFB задає переривання при звільненні регістра буфера прийнятих даних:

1 – переривання виробляється;

0 – переривання заборонене.

IDA визначає переривання при доступності прийнятих даних:

1 – переривання виробляється;

0 – переривання заборонене.

Біти 7-4 не використовуються й повинні приймати значення 0.

Регістр ідентифікації переривання (IIR). Регістр має адресу 2 щодо базової адреси контролера. Цей регістр доступний тільки по читанню й дозволяє одержати інформацію від контролера про очікуюче переривання. Значення біт регістра наведено на рисунку 3.3.

7	6	5	4	3	2	1	0
0	0	0	0	0	**I Type**		**II**

Рисунок 3.3 – Формат регістра ідентифікації переривання (IIR)

Біти **I Type** визначають тип очікуючого переривання, якщо воно зберігається контролером (що визначається бітом **II**):

11 – змінився стан лінії приймача;

10 – прийняті дані доступні;

01 – звільнений регістр буфера;

00 – змінився стан модему.

Біт II є індикатором очікуючого переривання:

0 – контролер послідовного інтерфейсу зберігає переривання;

1 – немає переривань, що очікують обробки.

Біти 7-3 регістра не використовуються й повинні приймати значення 0.

Більш докладна інформація про пріоритети переривань, умови появи й умови скидання стану переривання наведена в таблиці 3.5.

Кафедра_КбПЗ_2023рік

Таблиця 3.5 – Інформація про очікуюче переривання

I Type	Пріоритет	Тип	Умова появи	Умови скидання
11	1	Стан лінії приймача	Помилка переповнення, парності, посилки або пауза	Операція читання LSR
10	2	Доступність прийнятих даних	Доступність прийнятих даних	Операція читання RBR
01	3	Звільнення регістра буфера передавача	Звільнення THR	Операція читання IIR або запис в THR
00	4	Стан модему	Clear To Send, Data Set Ready, Ring Indicator або Data Carrier Detect	Операція читання MSR

Регістр керування лінією (LCR). Регістр має адресу 3 щодо базової адреси контролера. Цей регістр доступний і під час читання, і під час запису.

Значення даного регістра визначає формат переданих даних у лінію передачі даних контролером послідовного інтерфейсу. Значення біт регістра показане на рисунку 3.4.

7	6	5	4	3	2	1	0
DLAB	**SB**	**SP**	**EPS**	**PA**	**NSB**	**WLS**	

Рисунок 3.4 – Формат регістра керування лінією (LCR)

DLAB управляє доступом до регістрів буфера дільника. Якщо біт дорівнює 1, операція читання й запису по адресі 1 щодо базової адреси виконуються з

Кафедра_КБПЗ_2023рік

регістрами буфера дільника програмувального генератора. Для доступу до регістрів RBR, THR і IER біт повинен мати нульове значення.

SB встановлює стан «пауза», коли дорівнює 1. У цьому стані на виході контролера послідовного інтерфейсу встановлюється значення 0, що не може бути змінене ніякими іншими діями, крім як переустановкою біта в 0.

SP управляє установкою режиму незмінного біта контролю парності. Значення біта 1 задає режим, а значення 0 – скасовує. При встановленні біта SP в 1 повинен встановлюватися в 1 і біт PA, так як ці два біти зв'язані. Коли значення біта EPS дорівнює 0, посилається й контролюється значення біта контролю парності, рівне 1 (Mark Parity). При одиничному значенні біта EPS посилається й контролюється значення біта контролю парності, рівне 0 (Space Parity).

EPS задає вибір режиму контролю парності. Якщо біт встановлений в 0 і біт PA установлений в 1, генерується й перевіряється парна кількість одиничних біт символу посилки й біта контролю парності. Якщо біт установлений в 1 і біт PA встановлений в 1, генерується й перевіряється непарна кількість одиничних біт символу посилки й біта контролю парності.

PA є бітом дозволу контролю парності. Якщо біт установлений в 1, то генерується біт контролю парності між останнім бітом переданого символу й стоп-бітом.

NSB визначає кількість стоп-біт у кожному символі, переданому контролером послідовного інтерфейсу, і пов'язаний з довжиною слова обміну (біти WLS). Якщо цей біт встановлений в 0, то генерується й перевіряється один стоп-біт при будь-якій довжині слова обміну. Якщо цей біт встановлений в 1, то при довжині слова обміну в 5 біт генерується й перевіряється 1.5 стоп-біта, а при будь-якій іншій довжині слова обміну генерується й перевіряється 2 стоп-біта. При асинхронній передачі поняття біта нерозривно зв'язане з тривалістю сигналу, тому цілком можлива посилка нецілого числа стоп-біт. Це може знадобитися, якщо підключений до комп'ютера пристрій не програмується, а суворо налагоджений на аналіз стоп-біт заданої довжини.

Кафедра_КОПЗ_2023рік

Біти **WLS** визначають довжину слова обміну:

00 – 5 біт;

01 – 6 біт;

10 – 7 біт;

11 – 8 біт.

Регістр керування модемом (MCR). Регістр керування модемом має адресу 4 щодо базової адреси контролера. Цей регістр доступний по читанню й запису. За допомогою регістра можна управляти роботою модему. Формат регістра показаний на рисунку 3.5.

7	6	5	4	3	2	1	0
0	0	0	**LB**	**Out2**	**Out1**	**RTS**	**DTR**

Рисунок 3.5 – Формат регістра керування модемом (MCR)

LB задає режим «шлейфа» (Loopback) для діагностичних цілей. Якщо цей біт дорівнює 1, відбувається наступне:

1) вихід передавача (SOUT) встановлюється в активний стан;

2) вхід приймача (SIN) від'єднується;

3) вихід зсувового регістра передавача приєднується до зсувового регістра приймача;

4) чотири вхідні керуючі сигнали модему (CTS, DSR, DCD і RI) від'єднуються;

5) чотири вихідні керуючі сигнали модему (DTR, RTS, Out1 і Out2) приєднуються до чотирьох керуючих входів модему;

6) керуючі ланцюги модему примусово встановлюються в неактивний стан.

У діагностичному режимі передані дані відразу ж приймаються. При цьому повністю забезпечується переривання приймача й передавача. Керування перериваннями так само управляється регістром IER, однак джерелами переривань у цьому випадку є чотири молодших біти регістра MCR замість

Кафедра_КОПЗ_2023рік

чотирьох керуючих входів модему. Система керування перериваннями може бути перевірена в режимі «шлейф» записом у молодші 6 біт регістра LSR і молодші 4 біти регістра MSR. При установці кожного із цих біт в одиницю, виробляються відповідні переривання (якщо воно дозволено в регістрі IER). Умови скидання стану переривання повністю відповідають нормальному режиму роботи.

Для повернення до нормального режиму роботи необхідно спочатку перепрограмувати регістри для цього режиму роботи, а потім встановити біт LB регістра MCR у значення 0.

Out2 управляє сигналом Out2. При одиничному значенні біта сигнал Out2 встановлюється рівним 1. Сигнал Out2 управляє генерацією переривань контролера послідовного інтерфейсу. При одиничному знанні сигнал контролер генерує переривання у відповідності зі значенням регістра IER. При нульовому значенні сигналу Out2 контролер не генерує переривань незалежно від значення регістра IER.

Out1 управляє сигналом Out1. Якщо біт встановлений в 1, сигнал Out1 встановлюється 1. При завданні значення 0 сигнал встановлюється в нульовий рівень.

RTS управляє сигналом «запит на передачу» (Request to Send). При значенні цього біта, рівному 1, сигнал «запит на передачу» встановлюється рівним 1. При завданні значення 0 сигнал встановлюється в нульовий рівень.

DTR задає рівень сигналу «готовність термінала» (Data Terminal Ready). Якщо біт встановлений в 1, сигнал «готовність термінала» встановлюється рівним 1. При задані значення 0 сигнал встановлюватися в нульовий рівень.

Біти 7-5 не використовуються й завжди встановлюються в 0.

Регістр стану лінії (LSR). Регістр стану лінії має адресу 5 щодо базової адреси контролера й доступний тільки по читанню. Регістр LSR надає інформацію про стан обміну даних.

Формат регістра показаний на рисунку 3.6.

Кафедра_КБПЗ_2023рік

7	6	5	4	3	2	1	0
0	**TEMT**	**THRE**	**BI**	**FE**	**PE**	**OR**	**DR**

Рисунок 3.6 – Формат регістра стану лінії (LSR)

TEMT являється індикатором звільнення передавача. Встановка цього біта в 1 означає, що як регістр THR, так і регістр TSR вільний. Цей біт встановлюється в значення 0, якщо кожний з регістрів THR і TSR містить символ.

THRE є індикатором звільнення регістра THR. Установка цього біта в 1 означає, що з регістра THR символ переданий у зсувовий регістр передавача (TSR) і регістра THR готовий прийняти наступний байт. Якщо в регістрі IER дозволене переривання по звільненню регістра THR, то при установці цього біта в значенні 1 відбувається також переривання по звільненню регістра THR.

BI є індикатором стану «пауза» (Break Interrupt). Стан «пауза» фіксується в тому випадку, якщо рівень прийнятого сигналу встановлений в 0 на час прийому повної посилки, тобто загальний час стартового біта, біт даних, біта контролю парності й стоп-біта. Цей біт приймає значення 0 після операції читання регістра LSR. Біти 4-1 є індикаторами помилки й установка кожного із цих біт у значення 1 проводить до породження переривання по стану ліній приймача.

FE є індикатором «помилки стоп-біт» (Framing Error). Помилка стоп-бита фіксується в тому випадку, коли в прийнятому символі не виявлено коректного стоп-біта, тобто біт, що слідує за останнім бітом даних або за бітом контролю парності (у випадку контролю парності), має значення 0. Цей біт приймає значення 0 після операції читання регістра LSR.

PE є індикатором «помилки парності» (Parity Error). Помилка парності фіксується, якщо в прийнятому символі виявлене некоректне значення біта контролю парності. Цей біт приймає значення 0 після читання регістра LSR.

OR є індикатором «помилки переповнення» (Overrun Error). Помилка переповнення фіксується в тому випадку, якщо при записі чергового символу в

Кафедра_КОПЗ_2023рік

регістр RBR виявлено, що попередній вміст цього регістра не зчитане й, таким чином, воно загублено. Цей біт приймає значення 0 після операції читання регістра LSR.

DR індикатор доступності прийнятих даних. Цей біт завжди встановлюється в 1, коли приймачем повністю прийняті символ і поміщений у регістр RBR. Біт приймає значення 0 після операції читання з регістра RBR.

Біт 7 завжди встановлюється в значення 0.

Регістр стану модему (MSR). Регістр має адресу 6 щодо базової адреси контролера й доступний тільки по читанню. Регістр надає інформацію про стан керуючих ліній модему. Крім того, цей регістр містить 4 біти, які відображають зміну стану модему й встановлюються в значення 0 після операції читання з регістра MSR.

7	6	5	4	3	2	1	0
DCD	**RI**	**DSR**	**CTS**	**DDCD**	**TERI**	**DDSR**	**DCTS**

Рисунок 3.7 – Формат регістра стану модему (MSR)

DCD є інвертованим сигналом Data Carrier Detect (DCD). При встановленому режимі «шлейфа» (біт LB регістра MCR має значення 1) цей біт еквівалентний біту Out2 регістри MCR.

RI є інвертованим сигналом Ring Indicator (RI). При встановленому режимі «шлейфа» (біт LB регістра MCR має значення 1) еквівалентний біту Out1 регістра MCR.

DSR є інвертованим сигналом Data Set Ready (DSR). У режимі «шлейфа» (біт LB регістра MCR має значення 1) еквівалентний біту DTR регістра MCR.

CTS – інвертований сигнал Clear to Send (CTS). При встановленому режимі «шлейфа» (біт LB регістра MCR має значення 1) цей біт еквівалентний біту RTS регістра MCR.

Кафедра_КбПЗ_2023рік

Біти DDCD, TERI, DDSR і DCTS є індикаторами зміни стану модему й установка кожного із цих біт у значення 1 приводить до породження переривання по стану модему, якщо воно дозволено в регістрі IER.

DDCD є індикатором зміни сигналу Data Carrier Detect (DCD). Цей біт приймає значення 1 при зміні сигналу DCD після останньої операції читання регістра MSR.

TERI є індикатором заднього фронту сигналу RI. Цей біт приймає значення 1 при зміні сигналу RI з рівня логічної 1 на рівень логічного нуля.

DDSR є індикатором зміни сигналу Data Set Ready (DSR). Цей біт приймає значення 1 при зміні сигналу DSR після останньої операції читання регістра MSR.

DTCS є індикатором зміни сигналу Clear to Send (CTS). Цей біт приймає значення 1 при зміні сигналу CTS після останньої операції читання регістра MSR.

Невикористовуваний регістр (Scratch Register). Має адресу 7 щодо базової адреси контролера й доступний по читанню й запису. Регістр не управляє контролерам і може бути використаний як робочий регістр для зберігання яких-небудь даних.

Для послідовного введення потрібен засіб перетворення послідовних вхідних даних у паралельні дані, які можна помістити на шину. З іншого боку, для послідовного виводу необхідні засоби перетворення паралельних даних, представлених на шині, у послідовні вихідні дані. У першому випадку перетворення здійснюється регістром зсуву з послідовним входом і паралельним виходом (SIPO), а в другому – регістром зсуву з паралельним входом і послідовним виходом (PISO).

Програмуємий генератор служить для установки частоти контролера послідовного інтерфейсу. Частота проходження визначається як відношення частоти задаючого генератора до дільника частоти. Частота задаючого генератора дорівнює 1.8432Мгц. Дільник частоти являє собою 16-ти бітове число, молодший і старший байт якого завантажуються окремо через регістри буфера дільника. Після операції запису кожного з регістрів дільника дільник перезавантажується

Кафедра_КБПЗ_2023рік

відразу ж. У таблиці 3.6 наведені необхідні значення дільника для одержання потрібної частоти проходження.

Таблиця 3.6 – Значення дільника частоти для одержання потрібної частоти проходження

Необхідна частота проходження (у бодах)	Значення дільника для одержання необхідної частоти проходження	
	У десятковій системі	У шістнадцятковій системі
50	2304	0900h
75	1536	0600h
150	1536	0600h
300	384	0180h
600	192	00C0h
1200	96	0060h
1800	64	0040h
2400	48	0030h
3600	32	0020h
4800	24	0018h
7200	16	0010h
9600	12	000Ch
19200	6	0006h
38400	3	0003h
57600	2	0002h
115200	1	0001h

Протокол послідовного зв'язку

Послідовна передача даних означає, що дані передаються по єдиній лінії. При цьому біти байта даних передаються по черзі з використанням одного дроту. Для синхронізації перед групою біт даних звичайно йде спеціальний ***стартовий біт***, а після групи біт слідує ***біт перевірки на парність*** і один або два ***стопових***

Кафедра_КБПЗ_2023рік

біт. Іноді біт перевірки на парність може бути відсутнім. Протокол послідовної передачі даних показаний на рисунку 3.8.

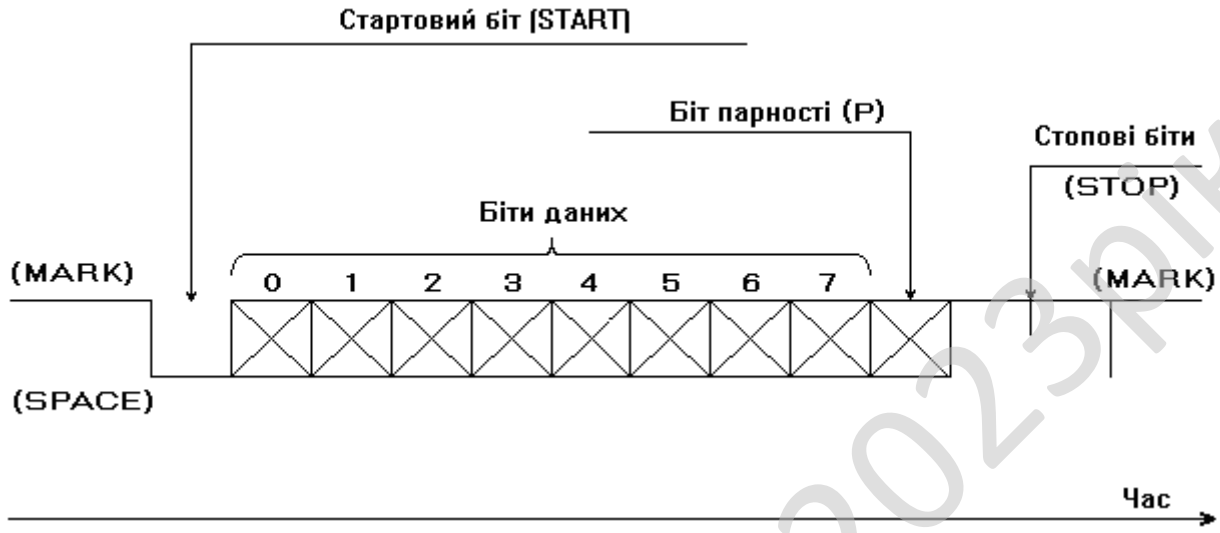

Рисунок 3.8 – Послідовна передача даних

З рисунку видно, що вихідний стан лінії послідовної передачі даних – рівень логічної 1. Це стан лінії називають відміченим – MARK. Коли починається передача даних, рівень лінії переходить в 0. Цей стан лінії називають порожнім – SPACE. Якщо лінія перебуває в такому стані більше певного часу, вважається, що лінія перейшла в стан розриву зв'язку – BREAK. Стартовий біт START сигналізує про початок передачі даних. Далі передаються біти даних, спочатку молодші, потім старші.

Якщо використовується біт парності P, то передається й він. Біт парності має таке значення, щоб у пакеті біт загальна кількість одиниць (або нулів) була парною або непарною, залежно від установки регістрів порту. Цей біт служить для виявлення помилок, які можуть виникнути при передачі даних через перешкоди на лінії. Прийомний пристрій заново обчислює парність даних і порівнює результат із прийнятим бітом парності. Якщо парність не збіглася, то вважається, що дані передані з помилкою. Звичайно, такий алгоритм не дає стовідсоткової гарантії виявлення помилок. Так, якщо при передачі даних

Кафедра_КбПЗ_2023рік

змінилося парне число біт, то парність зберігається й помилка не буде виявлена. Тому на практиці застосовують більш складні методи виявлення помилок.

У самому кінці передаються один або два стопових біти STOP, що завершують передачу байта. Потім до приходу наступного стартового біта лінія знову переходить у стан MARK. Використання біта парності, стартових і стопових біт визначають формат передачі даних. Очевидно, що передавач і приймач повинні використовувати той самий формат даних, інакше обмін буде неможливий. Інша важлива характеристика – швидкість передачі даних. Вона також повинна бути однаковою для передавача й приймача.

Швидкість передачі даних звичайно вимірюється в бодах. Боди визначають кількість переданих біт у секунду. При цьому враховуються й стартові/стопові біти, а також біт парності. Іноді використовується інший термін – біти в секунду (bps). Тут мається на увазі ефективна швидкість передачі даних, без врахування службових біт.

Послідовні дані передаються в синхронному або асинхронному режимах.

У синхронному режимі всі передачі здійснюються під керуванням спільного сигналу синхронізації, що повинен бути присутнім на обох кінцях лінії зв'язку.

Асинхронна передача має на увазі передачу даних пакетами. Кожний пакет містить необхідну інформацію, що вимагається для декодування даних, які містяться у ньому. Звичайно, другий режим складніше, але в нього є серйозна перевага: не потрібний окремий сигнал синхронізації.

Спроба встановити послідовний обмін інформацією буде марною, якщо один із пристроїв відключений. Без приймаючого пристрою передана інформація буде безвісти зникати в каналі. На щастя RS-232 у своїх специфікаціях виділяє 2 дроти для визначення підключення до кожного кінця послідовного каналу пристрою і його стану (чи включений пристрій).

Сигнал, що передається по 20 контакту називається сигналом готовності термінала (Data Terminal ready – DTR). Він має позитивну форму з DTE –

Кафедра_КОПЗ_2023рік

пристрою для повідомлення про те, що воно підключено, забезпечене живлення й готове почати сеанс зв'язку.

Аналогічно сигнал надходить на контакт 6. Він називається сигналом готовності набору даних (Data set ready – DSR). Цей сигнал так само представляється в позитивному виді й говорить про те, що DCE – пристрій включений і готовий до роботи.

У нормальному каналі RS-232 обоє ці сигналу повинні з'явитися перш ніж відбудеться що-небудь. Пристрій DTE посилає сигнал DTR пристрою DSE, і DSE посилає сигнал DSR пристрою DTE. Тепер обоє пристроїв знають, що інший пристрій готовий до роботи.

Звичайно апаратне квитування модему реалізується за допомогою двох різних дротів. Пристрій DCE встановлює позитивну напругу в 5 лінії, що говорить про готовність до прийому (Clear to send – CTS). Пристрій DTE сприймає цей сигнал як «шлях вільний». З іншої сторони каналу пристрій DTE встановлює позитивну напругу на 4-му контакті. Цей сигнал називається запит на передачу (Request to Send – RTS). Він говорить про те, що DCE повинний одержати інформацію.

Важливе правило говорить, якщо обидва сигнали й CTS, RTS не представлені позитивною напругою, інформація не буде передаватися в жодному напрямку. Якщо позитивна напруга відсутня на контакті CTS, пристрій DTE не передасть інформацію на DCE. Якщо ж позитивна напруга відсутня в лінії RTS, DCE не передасть інформацію DTE.

Ще один сигнал породжується DCE, що необхідний для початку передачі інформації. Це сигнал визначення передачі інформації (Carrier Detect або Data Carrier Detect – CD або DCD). Позитивна напруга в цій лінії вказує, що модем DCE одержав несучий сигнал з модему з іншого кінця лінії. Якщо ж цей сигнал не виявлений, то послідовність імпульсів може бути тільки шумами в лінії. Сигнали CD допомагаю DTE довідатися, коли варто побоюватися перешкод. У деяких випадках коли CD не мають позитивного потенціалу, DTE буде ігнорувати вхідну

Кафедра_КОПЗ_2023рік

інформацію.

Сигнал контакту 22 називається індикатором виклику (Ring Indicator – RI). Він використовується модемом DCE для індикації терміналу DTE, до якого він підключений, що їм визначена напруга виклику в телефонній лінії. Інакше кажучи, позитивна напруга RI активує термінал, повідомляючи йому, що хтось тривожить модем. У більшості послідовних систем зв'язку цей сигнал може вважатися параметричним сигналом, тому що його відсутність не перешкодить передачі інформації.

Номінально, передачі інформації в послідовних каналах передує дуже жорсткий протокол. Перш ніж вона відбудеться, апаратура на обох кінцях каналах повинна бути включена й готова до роботи. DTE, комп'ютер підтвердять сигнал DTR і DCE. Модем підтвердить свій DCR. Коли телефонний виклик розбудить модем він відправить RI до комп'ютера, що може видати повідомлення на екран. Якщо модем веде переговори з іншим модемом на іншому кінці каналу, звідки надійшов виклик, місцевий модем сформує CD сигнал своєму комп'ютеру. Якщо вони не були включені під час очікування до виклику, комп'ютер підтвердить RTS, а модем – CTS.

Введіть інформацію із клавіатури для посилки її через модем або пошліть інформацію з файлу. Якщо модем може передати інформацію досить швидко, він установить сигнал CTS, просячи PC трохи почекати. Коли сигнал CTS знову стане позитивним, комп'ютер сприймає це як запрошення до передачі інформації.

Якщо під час передачі інформації в комп'ютера з'явиться необхідність виконати яку-небудь важливу функцію, наприклад, збереже частину прийнятої інформації на диск, сигнал RTS буде прибраний, і модем припинить передачу інформації. Коли комп'ютер звільниться, сигнал RTS буде знову встановлена й інформація буде знову передаватися через модем.

Розглянемо більш детальніше порядок взаємодії комп'ютера й модему, а також двох комп'ютерів безпосередньо з'єднаних один з одним. Спочатку подивимося, як відбувається з'єднання комп'ютера з модемом. Входи TD і RD

Кафедра_КОПЗ_2023рік

використовуються пристроями DTE і DCE по-різному. Пристрій DTE використовує вхід TD для передачі даних, а вхід RD для прийому даних. І навпаки, пристрій DCE використовує вхід TD для прийому, а вхід RD для передачі даних. Тому для з'єднання термінального пристрою й пристрою зв'язку виводи їхніх роз'ємів необхідно з'єднати прямо (рисунок 3.9).

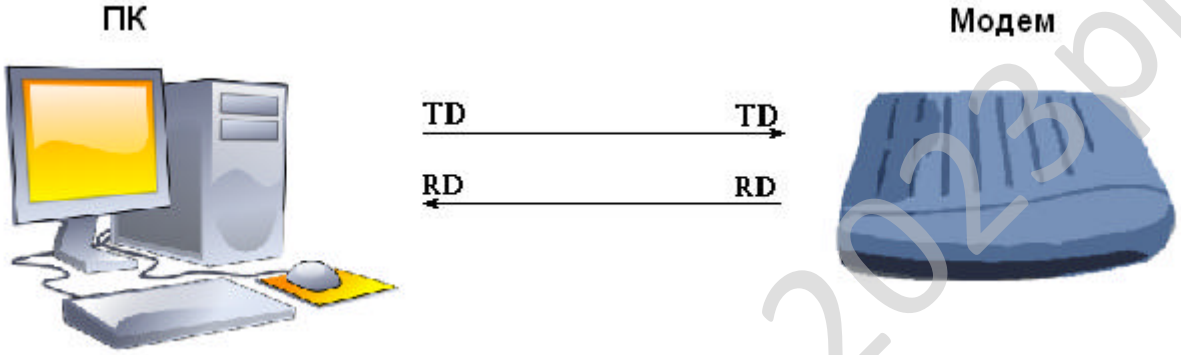

Рисунок 3.9 – Порядок взаємодії комп'ютера й модему по послідовному протоколу зв'язку

З'єднання інших ліній комп'ютера й модему показані на рисунку 3.10.

Рисунок 3.10 – З'єднання ліній послідовного порту комп'ютера та модему

Кафедра_КбПЗ_2023рік

Розглянемо процес підтвердження зв'язку між комп'ютером і модемом. На початку сеансу зв'язку комп'ютер повинен упевнитися, що модем може зробити виклик (перебуває в робочому стані). Потім, після виклику абонента, модем повинен повідомити комп'ютер, що він зробив з'єднання з віддаленою системою. Докладніше це відбувається в такий спосіб. Комп'ютер подає сигнал по лінії DTR, щоб показати модему, що він готовий до проведення сеансу зв'язку. У відповідь модем подає сигнал по лінії DSR.

Коли модем зробив з'єднання з іншим, віддаленим модемом, він подає сигнал по лінії DCD, щоб сповістити про це комп'ютеру. Якщо напруга на лінії DTR падає, це повідомляє модем, що комп'ютер не може далі продовжувати сеанс зв'язку, наприклад через те що виключено живлення комп'ютера. У цьому випадку модем перерве зв'язок. Якщо напруга на лінії DCD падає, це повідомляє комп'ютер, що модем втратив зв'язок і не може більше продовжувати з'єднання. В обох випадках ці сигнали дають відповідь на наявність зв'язку між модемом і комп'ютером.

Розглянута взаємодія є найнижчим рівнем керування зв'язку – підтвердження зв'язку. Існує більш високий рівень, що використовується для керування швидкістю обміну даними, але він також реалізується апаратно. Практично керування швидкістю обміну даними (керування потоком) необхідне, якщо виробляється передача великих обсягів даних з високою швидкістю.

Коли одна система намагається передати дані з більшою швидкістю, ніж вони можуть бути оброблені приймаючою системою, результатом може стати втрата частини переданих даних.

Щоб запобігти передачі більшого числа даних, ніж те, що може бути оброблене, використовують керування зв'язком – "керування потоком" (flow-controll handshake).

Стандарт RS-232C визначає можливість керування потоком тільки для напівдуплексного з'єднання. Напівдуплексним називається з'єднання, при якому в кожний момент часу дані можуть передаватися тільки в одну сторону. Однак

Кафедра_КОПЗ_2023рік

фактично цей механізм використовується й для дуплексних з'єднань, коли дані передаються по лінії зв'язку одночасно у двох напрямках.

У напівдуплексних з'єднаннях пристрій DTE подає сигнал RTS, коли воно бажає передати дані. DCE відповідає сигналом по лінії CTS, коли воно готово, і DTE починає передачу даних. Доти, поки обидва сигнали RTS і CTS не приймуть активний стан, тільки DCE може передавати дані.

При дуплексних з'єднаннях сигнали RTS/CTS мають протилежні значення в порівнянні з тими, які вони мали для напівдуплексних з'єднань.

Коли DTE може прийняти дані, він подає сигнал по лінії RTS. Якщо при цьому DCE готово для прийняття даних, воно повертає сигнал CTS. Якщо напруга на лініях RTS або CTS падає, то це повідомляє передавальній системі, що одержуючи система не готова для прийому даних.

На рисунку 3.11 приведений уривок діалогу між комп'ютером і модемом, що відбувається при обміні даними.

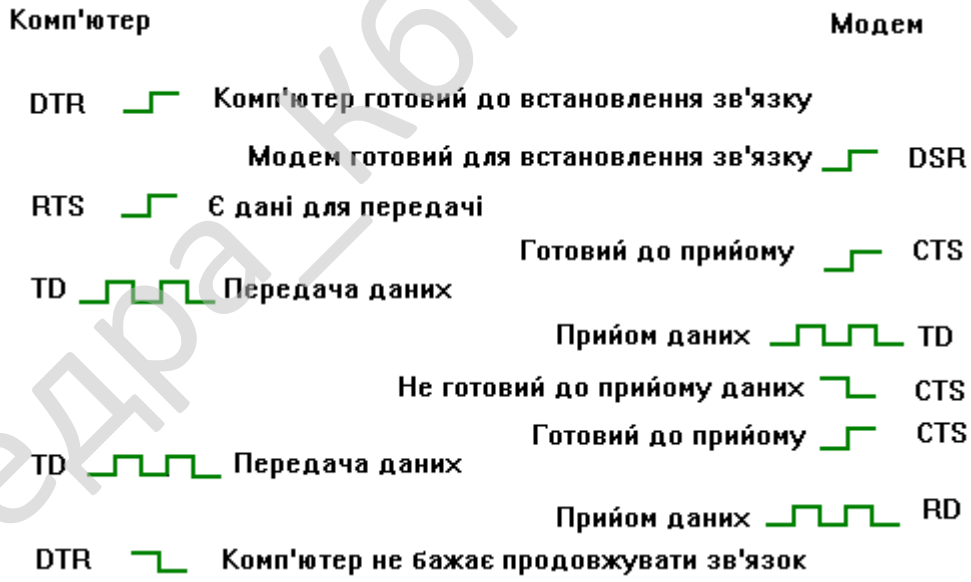

Рисунок 3.11 – Обмін даними між комп'ютером та модемом по COM-порту

Якщо треба зв'язати разом два комп'ютери за допомогою такого ж кабелю, як і для зв'язку модему й комп'ютера, то виникнуть деякі проблеми. Для з'єднання

Кафедра_КбПЗ_2023рік

двох термінальних пристроїв (двох комп'ютерів) як мінімум необхідно перехресне з'єднання ліній TR і RD (рисунок 3.12).

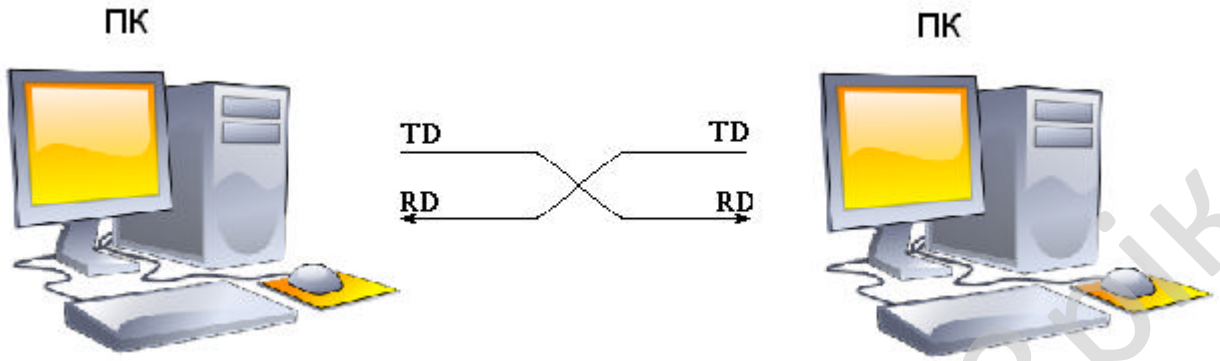

Рисунок 3.12 – Порядок взаємодії двох комп'ютерів по послідовному протоколу зв'язку

Однак у більшості випадків цього недостатньо, тому що для пристроїв DTE і DCE функції, виконувані лініями DSR, DTR, DCD, CTS і RTS, асиметричні. Пристрій DTE подає сигнал DTR і очікує одержання сигналів DSR і DCD. У свою чергу, пристрій DCE подає сигнали DSR, DCD і очікує одержання сигналу DTR. Таким чином, якщо з'єднати разом два пристрої DTE кабелем, що використовується для з'єднання пристроїв DTE і DCE, то вони не зможуть домовитися один з одним. Не виконається процес підтвердження зв'язку.

Тепер перейдемо до сигналів RTS і CTS, керування потоком даних. Іноді для з'єднання двох пристроїв DTE ці лінії з'єднують разом на кожному кінці кабелю. У результаті одержуємо те, що другий пристрій завжди готовий для одержання даних.

Тому, якщо при великій швидкості передачі приймаючий пристрій не встигає приймати і обробляти дані, можлива втрата даних.

Щоб вирішити всі ці проблеми для з'єднання двох пристроїв типу DTE використовується спеціальний кабель, що у побуті називається "нуль-модемом" (рисунок 3.13).

Кафедра_КоПЗ_2023рік

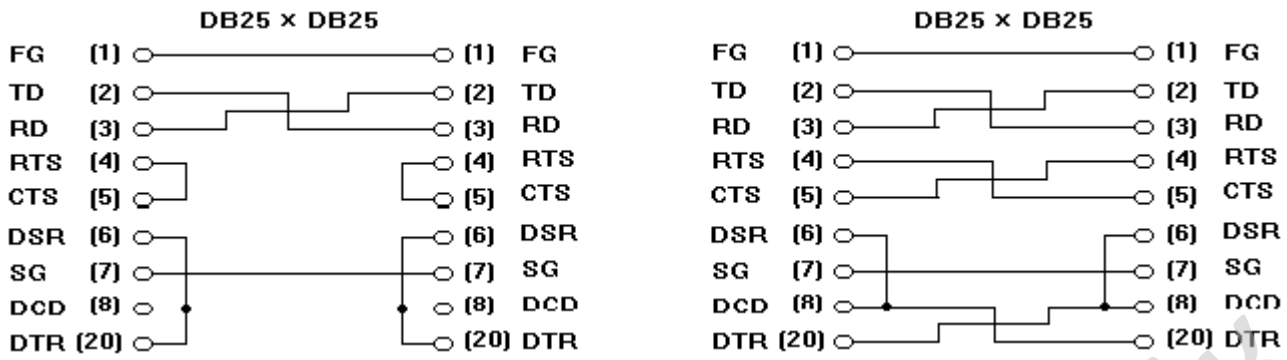

Рисунок 3.10 – Схема нуль-модемного кабелю

Для повноти картини розглянемо ще один аспект, пов'язаний з механічним з'єднанням портів RS-232C. Через наявність двох типів роз'ємів – DB25 і DB9 – часто бувають потрібні перехідники з одного типу роз'ємів на інші. Наприклад, Ви можете використовувати такий переходник для з'єднання COM-порту комп'ютера й кабелю нуль-модему, якщо на комп'ютері встановлений роз'єм DB25, а кабель кінчається роз'ємом DB9. Схема такого перехідника наведена на рисунку 3.14.

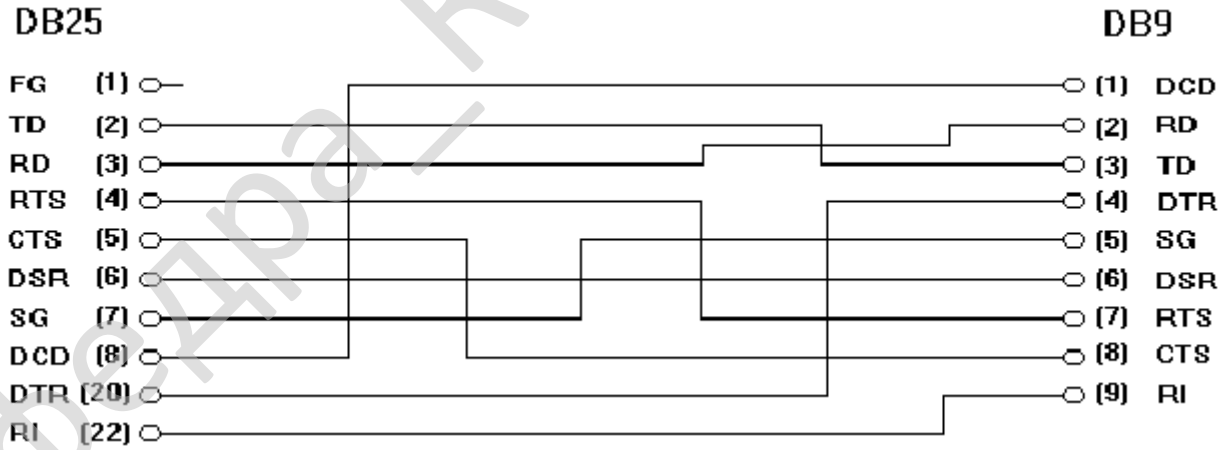

Рисунок 3.10 – Схема перехідника з роз'єму DB25 на роз'єм DB9

Помітимо, що багато пристроїв (такі, як термінали й модеми) дозволяють управляти станом окремих ліній RS-232C за допомогою внутрішніх перемикачів

Кафедра_КБПЗ_2023рік

(DIP-switches). Ці перемикачі можуть міняти своє значення на різних моделях модемів. Тому для їхнього використання варто вивчити документацію модему. Наприклад, для hayes-сумісних модемів, якщо перемикач 1 перебуває в положенні "виключений" (down), це означає, що модем не буде перевіряти наявність сигналу DTR. У результаті модем може відповідати на вхідні дзвінки, навіть якщо комп'ютер і не запитує в модему встановлення зв'язку.

Типові несправності при роботі послідовного порту та їх діагностика

Часто причиною поламок є проблеми з експлуатацією кабелів. До речі, стандартним для IBM PC є рішення, коли блоковим з'єднанням служить розетка (male), а на кабелі – розетка (female). У позначенні кабелю типу M/M або M/F перша буква говорить про вид з'єднувача з боку комп'ютера, друга – з боку пристрою (male-male, male-female).

Звичайно, чим вище швидкість обміну, тим вище вимоги до використовуваного сполучного кабелю, щоб уникнути помилок при прийомі або передачі. Звичайно при обміні даними на великих відстанях необхідно використовувати спеціальні кабелі з низькою питомою ємністю, крученими парами товстих дротів, поліетиленовою ізоляцією й одним або двома екранами.

Особливо часто користувачі попадають у скрутний стан, коли з конфігурації системи (для PC/AT) слідує, що послідовний порт (або порти) у системі є, але пристрій, підключений до одного із цих портів, не працює. От і починається гадання на кавовій гущі, що ж несправне насправді кабель, порт або пристрій, що підключається до нього. Для того щоб однозначно відповісти на це питання, простіше, як правило, спочатку перевірити справність кабелю й послідовного порту й тільки після цього робити якісь припущення щодо підключеного пристрою, оскільки це, як правило, трохи складніше.

У будь-якого IBM PC-сумісного комп'ютера в ROM BIOS убудовані функції, які призначені для роботи з послідовними портами (ініціалізація, прийом і передача символу, а також одержання статусу порту). При виконанні процедури POST, як правило, здійснюються ініціалізація й тестування внутрішніх регістрів

Кафедра_КОПЗ_2023рік

послідовних портів, включених у конфігурацію системи. Іншими словами, частина роботи з перевірки працездатності послідовних інтерфейсів комп'ютер бере на себе.

Зрозуміло, що тестування вихідних ланцюгів мікросхем UART, на базі яких, як правило, реалізований послідовний інтерфейс, без додаткової комутації вхідних і вихідних сигналів виконати неможливо, і ця частина роботи у функції POST не включена. Використовуючи численні тестові програми, можна досить точно локалізувати несправність або, принаймні, переконатися в непрацездатності (або працездатності) послідовного порту. Але спочатку для цього необхідно виготовити «заглушку», що забезпечує необхідну комутацію сигналів прийому-передачі й сигналів квитування при перевірці порту.

Залежно від того, який роз'єм в комп'ютері використовується для послідовного порту, заглушка може бути виготовлена на базі роз'єму-розетки female DB-25, або female DB-9. Для 25-контактного роз'єму необхідно запаяти перемички між контактами 2-3, 4-5, 8-20-22, а для перевірки 9-контактного – використовувати перехідник 9pm – 25pm або запаяти перемички 2-3, 7-8, 1-4-9.

Встановивши «заглушку» безпосередньо на роз'єм перевіряє мого порту, потрібно запустити яку-небудь тестову програму, наприклад, CHECKIT У тому випадку, якщо перевіряється справний порт, виконуються відповідні тести регістрів (даних статусу й т. п.) і тест передачі й прийому даних на різних швидкостях.

Повністю переконавшись у справності відповідного порту, «заглушку» можна підключити до інтерфейсного кабелю й перевірити його разом з портом. При наявності справного кабелю результати цієї перевірки не повинні відрізнятися від попередньої.

Кафедра_КОПЗ_2023рік

3.4 Розробка структурної схеми

Структурна схема розробленої системи зображена на рисунку 3.11. На ньому показано структуру обраного роз'єму та контролера послідовного порту.

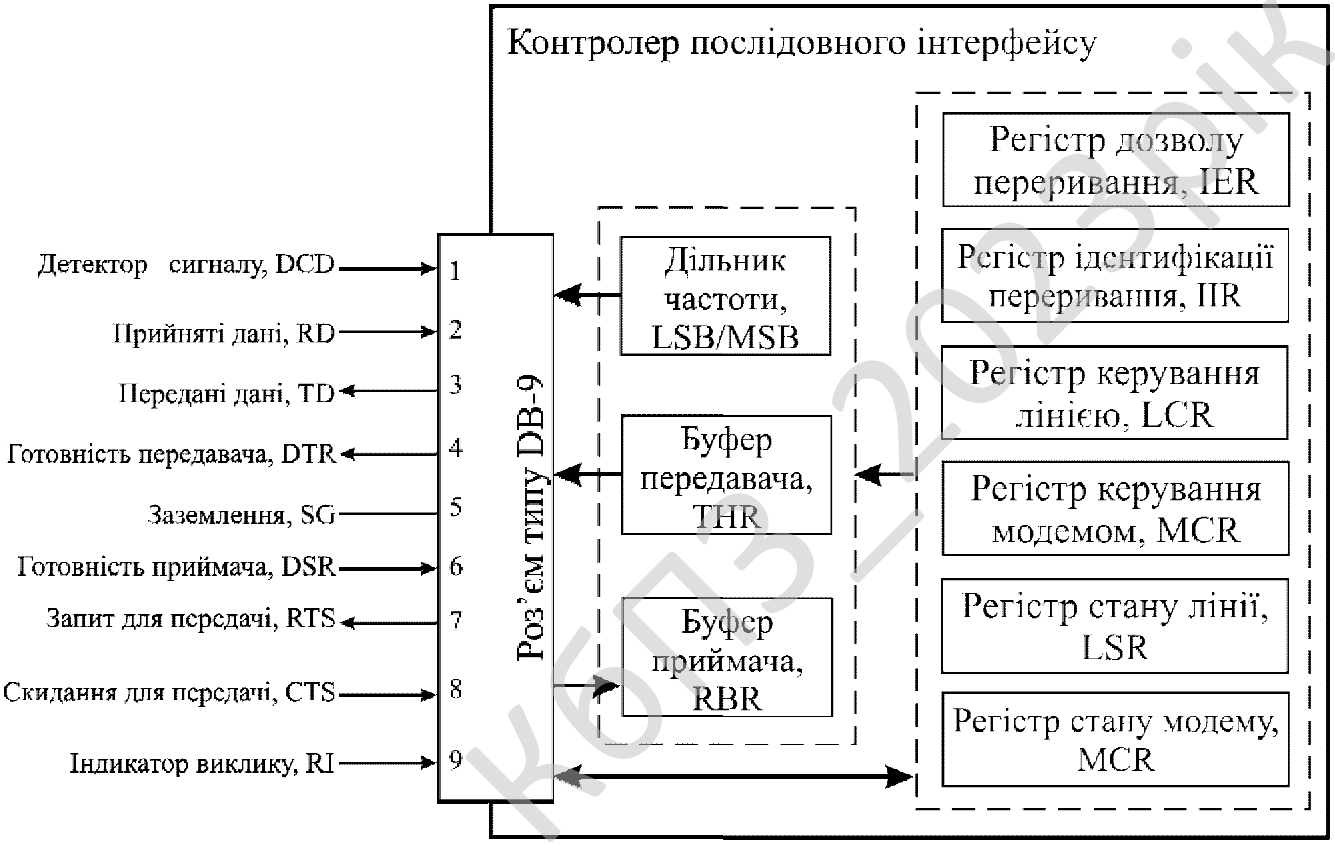

Рисунок 3.11 – Структурна схема розробленої системи

З рисунку видно, що для емуляції послідовного порту обрано роз'єм типу DB-9, що містить наступні контакти:

1. Детектор сигналу, DCD.
2. Прийняті дані, RD.
3. Передані дані, TD.
4. Готовність передавача, DTR.
5. Заземлення, SG.
6. Готовність приймача, DSR.

Кафедра_КОПЗ_2023ПК

7. Запит для передачі, RTS.
8. Скидання для передачі, CTS.
9. Індикатор виклику, RI.

Сигнали послідовного інтерфейсу підрозділяються на наступні: послідовні дані, керуючі сигнали квитування та сигнали синхронізації.

Послідовні дані (TD, RD) – це два незалежних послідовних канали даних: первинний і вторинний. Обидва канали можуть працювати в дуплексному режимі, тобто одночасно здійснюють передачу й прийом інформації.

Керуючі сигнали квитування (RTS, CTS) – засіб, за допомогою якого обмін сигналами дозволяє комп'ютеру почати діалог з модемом до фактичної передачі або прийому даних по послідовній лінії зв'язку.

Сигнали синхронізації (DCD, DTR, DSR, RI). У синхронному режимі (на відміну від більш розповсюдженого асинхронного) між пристроями необхідно передавати сигнали синхронізації, які спрощують синхронізм прийнятого сигналу з метою його декодування.

Контролер послідовного інтерфейсу містить:

- Дільник частоти, LSB/MSB.
- Буфер передавача, THR.
- Буфер приймача, RBR.
- Регістр дозволу переривання, IER.
- Регістр ідентифікації переривання, IIR.
- Регістр керування лінією, LCR.
- Регістр керування модемом, MCR.
- Регістр стану лінії, LSR.
- Регістр стану модему, MCR.

Значення дільника частоти міститься в двох регістрах: регістру буфера молодшого байта дільника (LSB) та регістру буфера старшого байта дільника (MSB). Ці регістри доступні по читанню й запису тільки при значенні біта дозволу доступу до дільника (DLAB) у регістрі керування лінією (LCR), рівному

Кафедра_КОПЗ_2023рік

1. При записі в цей регістр нового значення дільник відразу перезавантажується.

Регістр буфера передавача (THR) доступний тільки під час запису й при значенні біта дозволу доступу до дільника (DLAB) у регістрі керування лінією (LCR), рівному 0.

Регістр буфера приймача (RBR) доступний під час читання (IN) і при значенні біта дозволу доступу до дільника (DLAB) у регістрі керування лінією (LCR), рівному 0.

Регістр дозволу переривань (IER) доступний по читанню й запису, але тільки при значенні біта дозволу доступу до дільника (DLAB) у регістрі керування лінією (LCR), рівному 0. Цей регістр дозволяє управляти чотирма типами переривань, породжуваними контролером послідовного інтерфейсу.

Регістр ідентифікації переривання (IIR) доступний тільки по читанню й дозволяє одержати інформацію від контролера про очікуюче переривання.

Регістр керування лінією (LCR) доступний і під час читання, і під час запису. Значення даного регістра визначає формат переданих даних у лінію передачі даних контролером послідовного інтерфейсу.

Регістр керування модемом (MCR) доступний по читанню й запису. За допомогою регістра можна управляти роботою модему.

Регістр стану лінії (LSR) надає інформацію про стан обміну даних. Доступний тільки по читанню.

Регістр стану модему (MSR) надає інформацію про стан керуючих ліній модему. Крім того, цей регістр містить 4 біти, які відображають зміну стану модему й встановлюються в значення 0 після операції читання з регістра MSR.

3.5 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.12.

З рисунку видно, що розроблена система складається з наступних частин:

– блок емуляції;

Кафедра_КБПЗ_2023рік

– блок діагностики;

– блок графічного інтерфейсу;

– блок довідки.

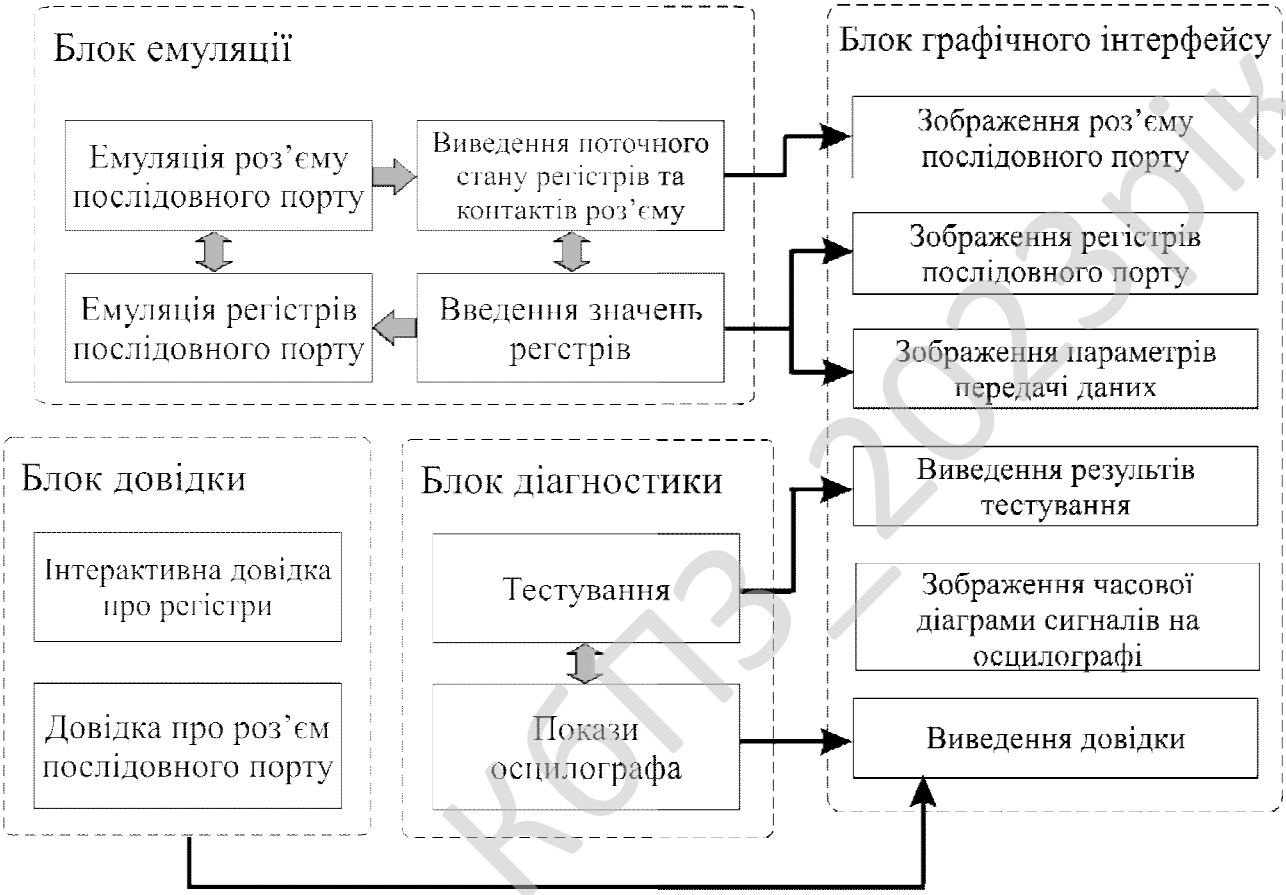

Рисунок 3.12 – Функціональна схема розробленої системи

Розглянемо більш детально кожний з блоків розробленої програми.

Блок емуляції містить функції, що реалізують логіку роботи послідовного порту, його регістрів та роз'єму. Він складається з наступних частин:

– Емуляція роз'єму послідовного порту.

– Емуляція регістрів послідовного порту.

– Виведення поточного стану регістрів та контактів роз'єму.

– Введення значень регістрів.

Блок діагностики дозволяє проводити тестування віртуального

Кафедра_КПЗ_2023рік

послідовного порту та виводити результати тестів у вигляді текстових повідомлень та показів осцилографа. Даний блок містить:

– Тестування.
– Покази осцилографа.

Блок довідки розроблений для того, щоб навіть непідготовлені користувачі, які не мають базових знань стосовно роботи послідовного порту, могли ознайомитися з необхідною для роботи з програмою інформацією. Блок містить наступні складові:

– Інтерактивна довідка про регістри.
– Довідка про роз’єм послідовного порту.

Блок графічного інтерфейсу дозволяє виводити на екран результати роботи програми у графічному вигляді, наглядному та зручному для розуміння. Даний блок містить:

– Зображення роз’єму послідовного порту.
– Зображення регістрів послідовного порту.
– Зображення параметрів передачі даних.
– Виведення результів тестування.
– Зображення часової діаграми сигналів на осцилографі.
– Виведення довідки.

Основними блоками є блок емуляції та діагностики, саме вони реалізують логіку роботи і системного програмування діагностики послідовного порту передачі даних. Блок довідки дозволяє більш повно зрозуміти процеси, що відбуваються під час роботи та тестування порту.

Важливу роль відіграє графічний інтерфейс, адже користувач взаємодіє з усіма блоками програми саме через нього. Тому велика увага приділялася тому, як саме буде візуалізовано процес діагностики. Було вирішено показувати зображення та поточний стан роз'єму, регістрів, осцилографа та виводити параметри передачі даних. Це дозволяє отримати якомога повнішу уяву про роботу та діагностику послідовного порту.

Кафедра_КПЗ_2023рік

3.6 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.13.

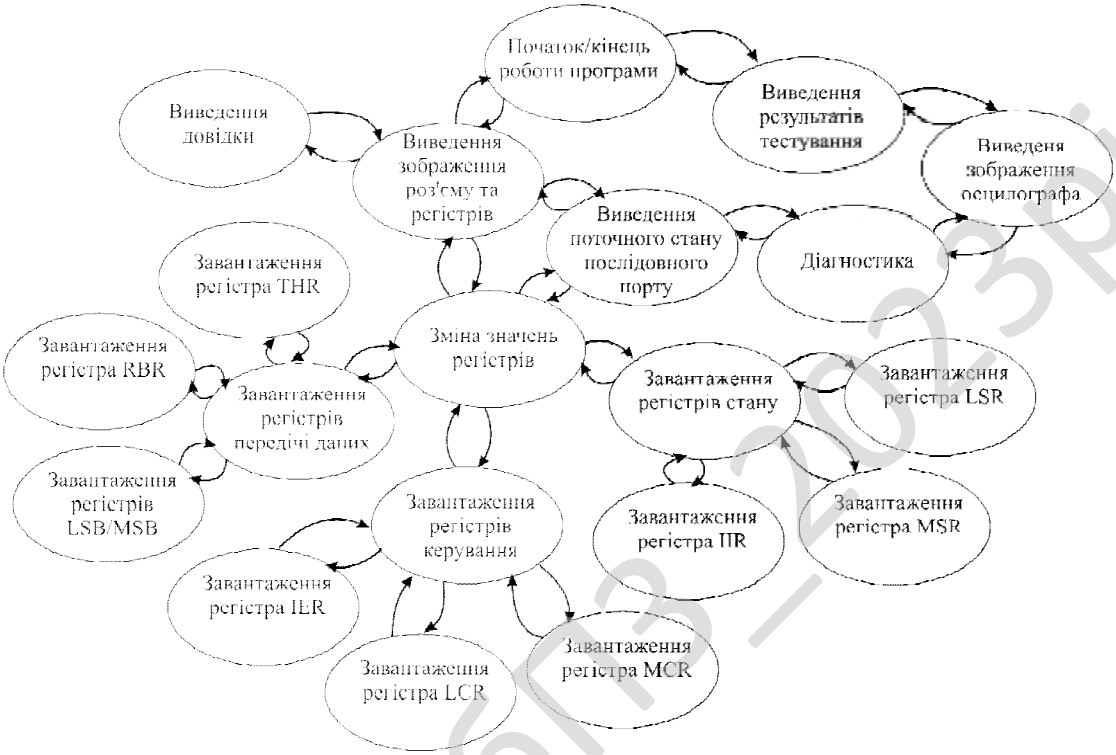

Рисунок 3.13 – Діаграма процесів розробленої системи

На початку роботи системи викликається процес виведення зображення роз'єму та регістрів, він тісно пов'язаний з наступним процесом – виведення поточного стану послідовного порту. У будь який момент часу користувач може викликати один з наступних процесів: виведення довідки, зміна значень регістрів, діагностика. Процес зміни значень регістрів складається з: завантаження регістрів передічі даних, завантаження регістрів керування та завантаження регістрів стану. Після діагностики запускаються процеси виведення зображення осцилографа та виведення результатів тестування.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

Кафедра_КБПЗ_2023рік

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1.

З рисунку видно, що після запуску програми спочатку відбувається вивід основного вікна програми. Потім здійснюється ініціалізація регістрів COM-порту та виведення початкових значень регістрів.

Користувач може запустити процес тестування, змінити значення регістрів та включити осцилограф.

Тестування здійснюється наступним чином:

1. Встановлення та обнулення біт регістрів контролера послідовного порту.
2. Читання регістрів та порівняння їх значень зі значенням відповідних контактів COM-порту.
3. Якщо прочитане з порту значення збігається з записаним у нього, то тестування пройшло успішно.
4. Якщо прочитане і записане у порт значення різні, то порт несправний і на екран виводяться назви несправних контактів, що не змінили своє значення після встановлення відповідних регістрів.

Якщо користувач хоче переглянути часову діаграму значень сигналів на контактах послідовного порту, то він повинен ввімкнути осцилограф. Після ввімкнення відкриється вікно з показами осцилографа для всіх контактів порту крім контакту, що використовується для заземлення.

Якщо користувач змінює значення регістрів, то відбувається зміна режиму роботи, параметрів передачі даних та рівня напруг на контактах COM-порту відповідно до встановлених у регістрах. А потім виводиться новий стан порту.

Кафедра_КБПВ_2023рік

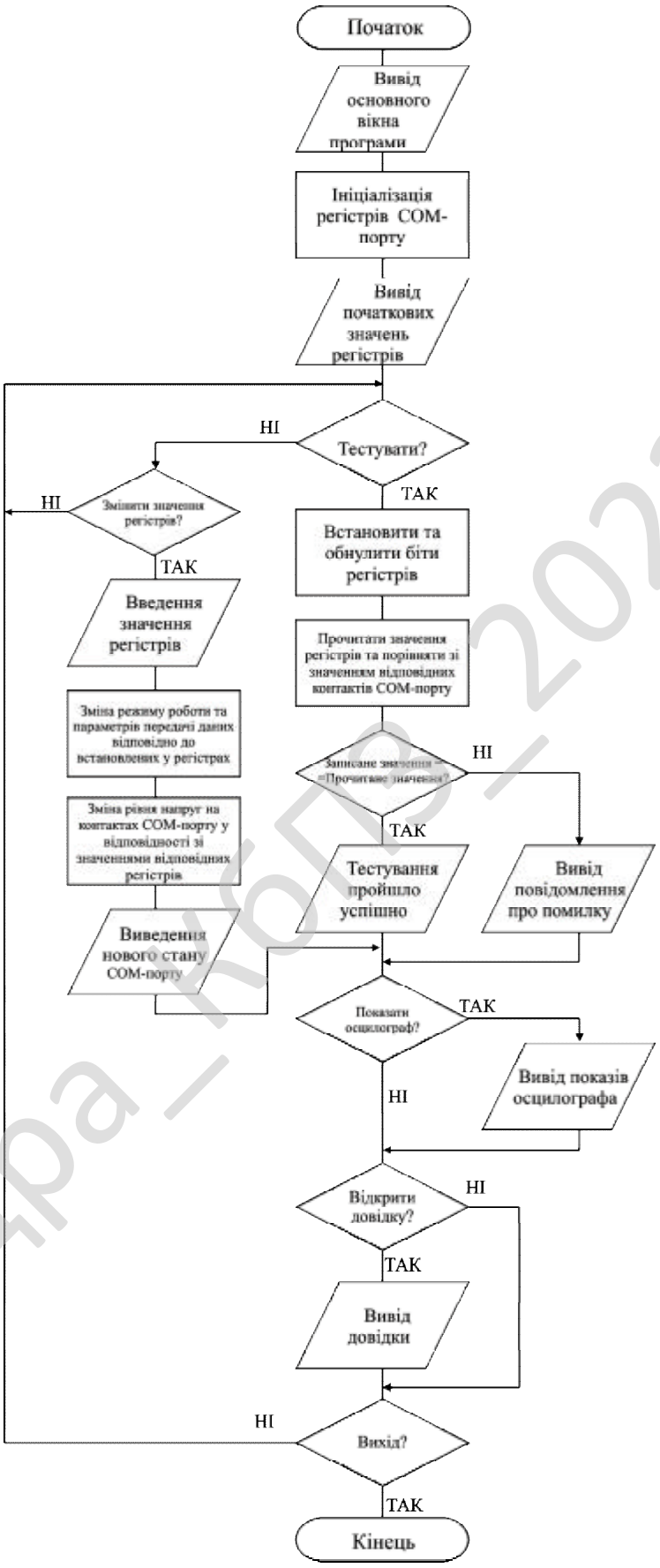

Рисунок 4.1 – Блок-схема алгоритму роботи основної програми

Кафедра_КОПЗ_2023рік

Перед першим використанням адаптера асинхронного послідовного зв'язку необхідно зробити його ініціалізацію, указавши йому наступні параметри: частоту передачі, кількість інформаційних біт, кількість стоп-бітов і наявність біта парності в кадрі, а для більш просунутих контролерів (16550 UART) можна також визначити розмір буфера FIFO і дозволити/заборонити DMA. Цю операцію проводить BIOS при запуску комп'ютера, ми ж будемо емалювати цей процес. Детально це виглядає так:

0) Контроль помилок

1) Встановлюємо регістр керування лінією (БА+3) у такий спосіб:

– біти 0,1 – довжина слова;

– біт 2 – число стоп-біт;

– біти 3,4,5 – спосіб контролю парності;

– біт 6 – 0 = звичайне функціонування, 1 = сигнал BREAK;

– біт 7 – 0 = режим передачі/прийому даних, 1 = режим вибору дільника частоти.

Під час ініціалізації біт 7 встановлюємо в 1, у біті 6 – 0.

2) Пишемо в БА+0 і в БА+1 молодший і старший байти дільника частоти. Дільник частоти – це число, одержуване розподілом магічного числа 1843200 на бажану швидкість, помножену на 16. Магічне число – це частота внутрішнього джерела синхронізації в 1843200 Гц. Дільник частоти, рівний одиниці відповідає максимальній частоті в 115200 біт/с, тому що 1843200/(115200*16) = 1.3.

Встановлюємо біт 7 регістра БА+3 у нуль. Якщо пропустити цей пункт, то програма не зможе передавати/приймати дані через інтерфейс і дозволяти/забороняти переривання, тому що буде вважати ці дані дільником частоти.

4) При необхідності можна також включити режим буферизації (FIFO) і прямого доступу до пам'яті (DMA) для мікросхеми 16550 UART шляхом зміни регістра БА+2.

Блок-схема алгоритму ініціалізації показана на рисунку 4.2.

Кафедра_КПЗ_2023рік

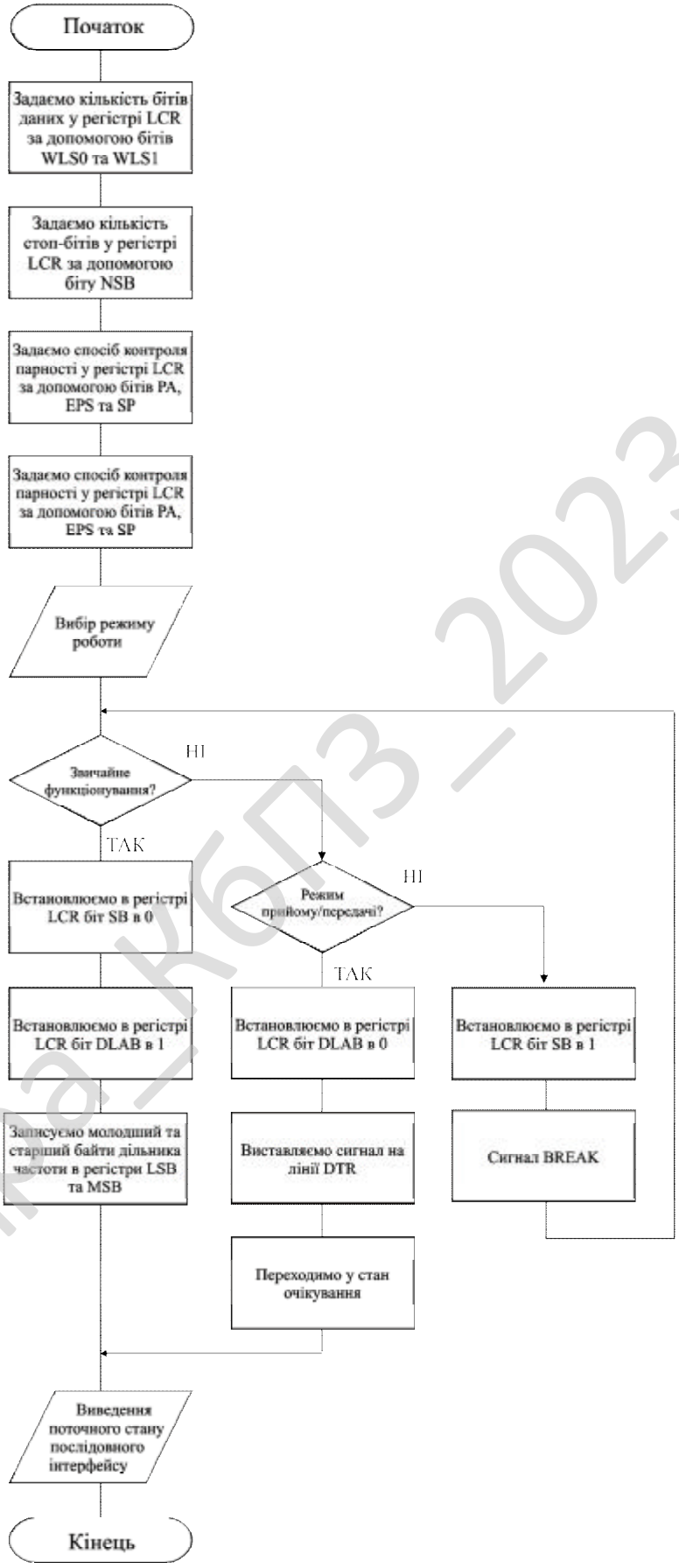

Рисунок 4.2 – Блок-схема алгоритму ініціалізації послідовного інтерфейсу

Кафедра_КбПЗ_2023рік

Програмно це виглядає наступним чином:

```
#define MAX_RS232_IF 5 // Максимальна кількість адаптерів RS-232
#define _ADAPTER_FREQ 1843200L // Частота джерела синхронізації, константа
#define _16550_UART_FIFO // Вмикання/вимикання буферизації
unsigned long RS232_base_ports [MAX_RS232_IF]; // У цьому масиві ми будемо зберігати базові адреси для кожного інтерфейсу
/* Використання такого масиву дозволить вказати базову адресу тільки один раз, у функції RS232_init(), а в інших функціях указувати лише номер адаптера. */
char RS232_init(unsigned char num, unsigned short base_port, unsigned char parity,unsigned char stop_bits, unsigned char word_length, unsigned long baud_rate)
// Функція ініціалізації RS-232
// num – номер інтерфейсу (від 0 до MAX_RS232_IF), base_port – БА, parity – контроль по парності
// stop_bits – кількість стоп-біт, word_length – довжина слова, baud_rate – бажана швидкість передачі/прийому
{
char format_byte;
union
{
int baud_divisor;
char bytes[2];
}word;
// Контроль помилок
if ((num > MAX_RS232_ IF-1)||(word_length > 3)||(stop_bits > 1)||(parity > 7)||(!baud_rate)) return RS232_INVALID_PARAM;
RS232_base_ports[num] = base_port; // Запам'ятали БА для даного інтерфейсу, він знадобиться в інших функціях
format_byte = 0x3F & ((word_length)|(stop_bits << 2)|(parity << 3));
word.baud_divisor = (_ADAPTER_FREQ / baud_rate) >> 4;
outportb(base_port+3, format_byte | 0x80); // (Крок 1)
outportb(base_port+0, word.bytes[0]); // (Крок 2)
outportb(base_port+1, word.bytes[1]);
outportb(base_port+3, format_byte); // (Крок 3)
#ifdef _16550_UART_FIFO // Якщо ми дозволили FIFO для мікросхеми 16550 UART
outportb(base_port+2, 0x7); // Вмикаємо режим FIFO і встановлюємо розмір буфера в 16 байт (Крок 4)
#endif
return RS232_NOERROR;
}
```

Опишемо, функцію перевірки ініціалізації RS-232. Ця функція допоможе

Кафедра_КбПЗ_2023рік

встановити, чи був ініціалізований контролер:

```
char isRS232_init(unsigned char num)
{ // Перевірка ініціалізації адаптера (чи була використана функція RS232_init()?)
return (RS232_base_ports[num] != 0)?RS232_NOERROR:RS232_NOT_INIT;
}
```

Після цього дані, записані в регістр БА+0 будуть передаватися через інтерфейс до пристрою, а дані, прочитані із цього регістра будуть прийматися з інтерфейсу RS-232. Але не все так просто. Не варто забувати, що при роботі з яким-небудь пристроєм нам спочатку потрібно його ініціалізувати (попросту говорячи, включити й підготувати), а також про те, що перед будь-якою передачею/прийомом даних нам потрібно чекати апаратного підтвердження готовності.

Для того, що б перевірити готовність інтерфейсу передати дані драйверу на обробку й готовність пристрою одержати дані від драйвера через інтерфейс RS-232 необхідно прочитати біти готовності з відповідних портів введення/виведення (адаптер повинен бути ініціалізований функцією RS232_init() перед цим). Якщо потрібна функція повертає значення RS232_NOERROR – можна передавати наступний байт.

```
char RS232_receive_ready(unsigned char num)
{ // Функція повертає ознаку готовності приймача (перевіряє біт 0 у БА+5)
return (inportb(RS232_base_ports[num]+5) & 0x01)?RS232_NOERROR:RS232_NOT_READY;
}
char RS232_send_ready(unsigned char num)
{ // Функція повертає ознаку готовності передавача (перевіряє біт 4 у БА+6)
Return (inportb(RS232_base_ports[num]+6) & 0x10)?RS232_NOERROR:RS232_NOT_READY;
}
```

А тепер – найважливіші функції: функції відправлення й одержання байта в/з адаптера RS-232. Звертаю увагу на те, що в них відсутня перевірка правильності ініціалізації. RS232_receivebyte() і RS232_sendbyte() використовують масив RS232_base_ports[], у якому зберігаються базові адреси для ініціалізованих функцією RS232_init() адаптерів.

Кафедра_КОПЗ_2023рік

```
char RS232_receivebyte(unsigned char num)
{ // Читання байта з інтерфейсу
return inportb(RS232_base_ports[num]);
}
void RS232_sendbyte(unsigned char num, char byte)
{ // Відправлення байта в інтерфейс
outportb(RS232_base_ports[num], byte);
}
```

У принципі, написаних функцій уже буде досить для обміну інформацією з ліній інтерфейсу RS-232, але перед тим, як ми почнемо роботу із пристроєм (наприклад, з модемом або з мишкою), не погано б було зробити його ініціалізацію, та й просто проконтролювати його працездатність. З погляду програмування буває 2 типи пристроїв для RS-232: прості та "розумні". Різниця між ними в тому, що розумні пристрої (наприклад, модеми) можуть апаратно підтримувати своє вмикання й готовність, виставляючи сигнали на відповідних лініях. Прості (наприклад, мишки) – не можуть.

Для ініціалізації простого пристрою можна скористатися схемою:

1. Виставляємо сигнал на лініях DTR і RTS.
2. Якщо треба, виставляємо сигнал на лінії OUT2 для дозволу генерації IRQ.

Схема ініціалізації розумного, такого як модем, пристрою виглядає трохи складніше:

1. Виставляємо сигнал на лінії DTR.
2. Переходимо у стан очікування.
3. Перевіряємо сигнал на лінії DSR: якщо він відсутній – модем не відповідає (або його просто немає), тоді виходимо з помилкою 1.
4. Виставляємо сигнал на лінії DTR і RTS одночасно.
5. Переходимо у стан очікування.
6. Якщо модем не відповів сигналом на лінії CTS – живлення включене, але він не готовий до обміну інформацією, тоді виходимо з помилкою 2.
7. Якщо треба, виставляємо сигнал на лінії OUT2.

Виходячи із цих двох алгоритмів, напишемо ще чотири функції

Кафедра_КОПЗ_2023рік

ініціалізації й деініціалізації різних пристроїв, що підключаються до RS232 (вони необхідні для встановлення й перевірки відповідних сигналів на лініях):

```
char RS232_modem_on(unsigned char num)
{ // Ініціалізація модему
char mcr;
unsigned long wait;
if(num>MAX_RS232_ IF-1) return RS232_INVALID_PARAM;
mcr = inportb(RS232_base_ports[num]+4);
outportb(RS232_base_ports[num]+4, mcr | 0x01);
// Виставляємо DTR (Data Terminal Ready)
wait = _WAIT;
while(wait --);//Затримка
if(!(inportb(RS232_base_ports[num]+6) & 0x20))
{
// Якщо сигнал DSR (Data Set Ready) не надійшов – модем не відповідає
outportb(RS232_base_ports[num]+4, mcr); // Відновлюємо регістр
return RS232_NO_MODEM;
}
// Виставляємо DTR і RTS (Request To Send)
outportb(RS232_base_ports[num]+4, mcr | 0x03);
wait = _WAIT;
while(wait --); //Затримка
// Якщо немає сигналу CTS (Clear To Send) – модем є, але не готовий
if(!(inportb(RS232_base_ports[num]+6) & 0x10))
{
outportb(RS232_base_ports[num]+4, mcr); // Відновлюємо регістр
return RS232_MODEM_NOT_READY;
}
// Виставляємо сигнали DTR, RTS і OUT2
outportb(RS232_base_ports[num]+4, mcr | 0x0B);
// Очищаємо всі можливі події апаратного переривання:
inportb(RS232_base_ports[num]+0);
inportb(RS232_base_ports[num]+5);
inportb(RS232_base_ports[num]+6);
return RS232_NOERROR;
}
void RS232_modem_off(unsigned char num)
{ // Деініціалізація (вимикання) модему
RS232_device_off(num); // Викликаємо функцію RS232_device_off(), описану
нижче
}
```

Кафедра_КБПЗ_2023рік

```
char RS232_device_on(unsigned char num)
{ // Ініціалізація простого пристрою
char mcr;
if(num>MAX_RS232_ IF-1) return RS232_INVALID_PARAM;
mcr = inportb(RS232_base_ports[num]+4);
outportb(RS232_base_ports[num]+4, mcr | 0x03); // Встановлюємо DTR і RTS
return RS232_NOERROR;
}
void RS232_device_off(unsigned char num)
{ // Деініціалізація розумного пристрою
if(num>MAX_RS232_ IF-1) return;
outportb(RS232_base_ports[num]+4, 0x00); // Очищаємо всі біти керування
// Очищаємо всі можливі події апаратного переривання:
inportb(RS232_base_ports[num]+0);
inportb(RS232_base_ports[num]+5);
inportb(RS232_base_ports[num]+6);
}
```

_WAIT – це час затримки поки модем (чи інший "розумний" пристрій) обробляє сигнал від комп'ютера і відповідає на нього. Якщо значення _WAIT занадто мале, пристрій може не встигнути відповісти на запит, і функція поверне помилку. Якщо занадто велике – буде відчутна пауза при роботі драйвера, що також не припустимо. Задавати його краще в такий спосіб:

```
#define _WAIT 0x0FF
```

При ініціалізації іншого пристрою (не модему), затримки не потрібні, оскільки відповіді в кожному разі не буде.

У деяких випадках (особливо, якщо ми використовуємо оброблювач апаратного переривання) нам буде необхідно контролювати помилки передачі даних по інтерфейсу. Для цього скористаємося досить примітивними функціями виявлення помилки кадрування RS232_framing_error() і помилки паритету, тобто контролю по парності/непарності RS232_parity_error():

```
char RS232_framing_error(unsigned char num)
{ // Визначаємо наявність помилки кадрування
Return (inportb(RS232_base_ports[num]+5) & 0x08)?RS232_FRAMING_ERROR:RS232_
NOERROR;
}
char RS232_parity_error(unsigned char num)
{ // Визначаємо наявність помилки паритету
```

Кафедра_КОПЗ_2023рік

```
	Return (inportb(RS232_base_ports[num]+5) & 0x04)?RS232_PARITY_ERROR:RS232_N
OERROR;
	}
```

Наступна функція також є дуже цікавою для програміста. З технічної сторони, вона всього лише визначає наявність сигналу RI на лінії інтерфейсу. При підключенні до адаптера RS-232 модему, сигнал буде генеруватися останнім у випадку дзвінка на телефонній лінії. Це дозволить нам програмно зреагувати на вхідний дзвінок (наприклад, зняти трубку й установити з'єднання з віддаленим модемом).

```
	char RS232_modem_ring(unsigned char num)
	{
	if(num>MAX_RS232_ IF-1) return RS232_INVALID_PARAM;
	return (inportb(RS232_base_ports[num]+6) & 0x40)?RS232_NOERROR:RS232_MODEM_
NO_RING;
	}
```

На рисунку 4.3 зображена блок-схема алгоритму прийому/передачі через послідовний порт.

У протоколі RS-232 існують два методи керування обміном даних: апаратний і програмний, а також два режими передачі: синхронний і асинхронний. Протокол дозволяє використовувати будь-який з методів керування разом з будь-яким режимом передачі. Також допускається робота без керування потоком, що має на увазі постійну готовність хоста й пристрою до прийому даних, коли зв'язок встановлений (сигнали DTR і DSR установлені).

Апаратний метод керування реалізується за допомогою сигналів RTS і CTS. Для передачі даних хост (комп'ютер) встановлює сигнал RTS і чекає сигналу CTS від пристрою, після чого починає передачу даних доти, поки сигнал CTS встановлений. Сигнал CTS перевіряється хостом безпосередньо перед початком передачі чергового байта, тому байт, що вже почав передаватися, буде переданий повністю незалежно від значення CTS. У напівдуплексному режимі обміну даними (пристрій і хост передають дані по черзі, у повнодуплексному режимі вони можуть робити це одночасно) зняття сигналу RTS хостом означає його перехід у режим прийому.

Кафедра_КОПЗ_2023рік

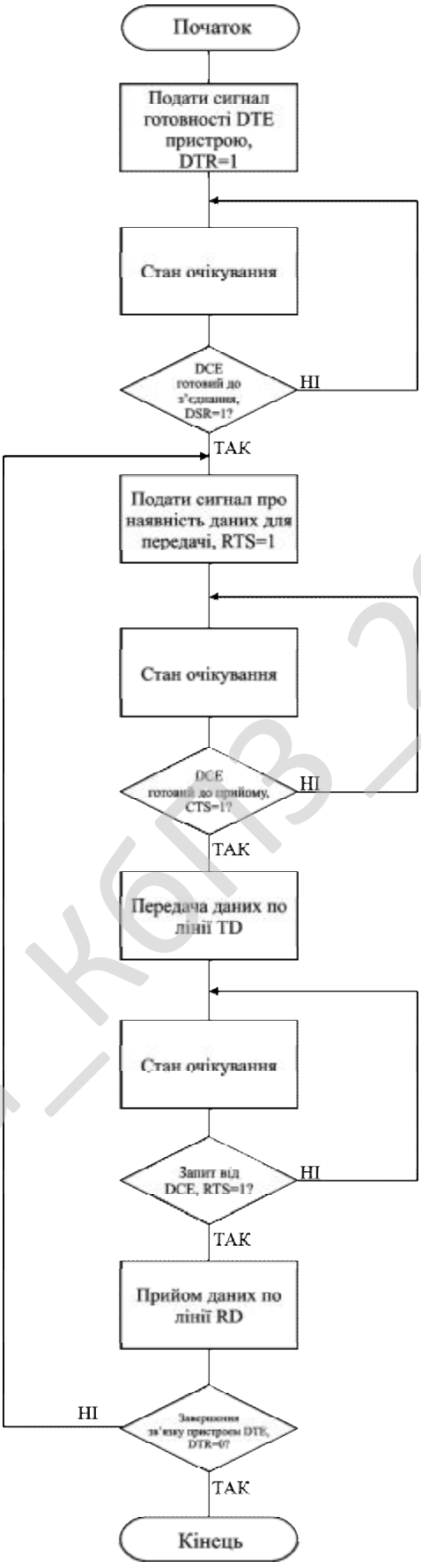

Рисунок 4.3 – Блок-схема алгоритму прийому/передачі через послідовний порт

Кафедра_КОПЗ_2023рік

Програмний метод керування полягає в передачі приймаючою стороною спеціальних символів зупинки (символ з кодом 0x13, що називається XOFF) і поновлення (символ з кодом 0x11 – XON) передачі. При одержанні даних символів передавальна сторона повинна відповідно зупинити передачу або відновити її (при наявності даних, що очікують передачі). Цей метод простіше з погляду реалізації апаратури, однак забезпечує більш повільну реакцію й відповідно вимагає завчасного повідомлення передавача при зменшенні вільного місця в прийомному буфері до певної межі.

Синхронний режим передачі має на увазі безперервний обмін даними, коли біти слідують один за іншим без додаткових пауз із заданою швидкістю. Цей режим COM-портом не підтримується.

Асинхронний режим передачі полягає в тому, що кожний байт даних (і біт контролю парності, у випадку його наявності) "обертається" синхронізуючою послідовністю з старт-бітом й одним або декількома стоп-бітами.

З'єднання пристроїв DTE та DCE для встановлення зв'язку відбувається наступним чином:

1. Встановлення сигналу готовності DTE пристрою, DTR=1.
2. Перехід DTE у стан очікування.
3. Якщо DSR=1, то DCE готовий до з’єднання.

Алгоритм передачі даних:

1. DTE подає сигнал про наявність даних для передачі, RTS=1.
2. Перехід DTE у стан очікування.
3. Якщо CTS=1 – DCE готовий до прийому.
4. Передача даних по лінії TD.

Алгоритм прийому даних:

1. DTE у стані очікування.
2. Запит від DCE, RTS=1.
3. Прийом даних по лінії RD

Для завершення зв'язку пристрій DTE подає сигнал DTR=0.

Кафедра_КБПЗ_2023рік

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму Camellia – блоковий шифр на основі мережі Фейстеля. У криптографії, Camellia – це симетричний ключ блоковий шифр із розміром блоку 128 біт і розмірами ключа 128, 192 і 256 біт. Він був розроблений спільно Mitsubishi Electric і NTT з Японії. Шифр був схвалений для використання ISO / IEC, проектом Європейського Союзу NESSIE і Японським CRYPTREC проект. шифр має рівні безпеки й можливості обробки, порівнянні з Advanced Encryption Standard.

Шифр був розроблений, щоб підходити як для програмних, так і для апаратних реалізацій, від недорогих смарт-карти для високошвидкісних мережних систем. Він є частиною криптографічного протоколу Transport Layer Security (TLS), призначеного для забезпечення безпеки зв'язки в комп'ютерній мережі, такий як Інтернет

Camellia – це шифр Фейстеля з 18 раундами (при використанні 128-бітних ключів) або 24 раундами (при використанні 192- або 256-бітних ключів). Кожні шість раундів застосовується шар логічного перетворення: так звана «FL-функція» або її зворотна. Camellia використовує чотири 8 × 8-бітних S-блоку із вхідними й вихідними афінними перетвореннями й логічними операціями. Шифр також використовує введення й вивід відбілювання клавіш. Шар дифузія використовує лінійне перетворення на основі матриці з номером галузей 5.

Аналіз безпеки

Камелія вважається сучасним надійним шифром. Навіть при використанні параметра меншого розміру ключа (128 біт) уважається неможливим зламати його за допомогою атаки грубої сили на ключі за допомогою сучасних технологій. Немає відомих успішних атак, що значно послабляють шифр. Шифр був схвалений для використання ISO / IEC, проектом Європейського Союзу

Кафедра_КОПЗ_2023рік

NESSIE і Японським CRYPTREC проект. Японський шифр має рівні безпеки й можливості обробки, порівнянні із шифром AES/Rijndael.

Camellia – це блоковий шифр, який може бути повністю визначені мінімальними системами багатомірних багаточленів:

– Камелія (а також AES) S-блоки можуть бути описані системою 23 квадратних рівнянь в 80 членах.

– Розклад ключів можна описати 1120 рівняннями в 768 змінні з використанням 3328 лінійних і квадратичних членів.

– Увесь блоковий шифр можна описати 5104 рівняннями в 2816 змінні з використанням 14 592 лінійних і квадратичних членів.

– Усього потрібно 6224 рівняння з 3584 змінними з використанням 17 920 лінійних і квадратичних членів.

– Кількість вільних членів становить 11 696, що приблизно таке ж число, що й для AES.

Теоретично, такі властивості можуть дозволити зламати Camellia (і AES) за допомогою алгебраїчної атаки, такий як розширена розріджена лінеаризація, у т Майбутнє за умови, що атака стане можливою.

Хоча Camellia запатентована, вона доступна за безоплатною ліцензією. Це дозволило шифру Camellia стати частиною проекту OpenSSL під ліцензією з відкритим вихідним кодом з листопада 2006 року. Це також дозволило йому стати частиною Mozilla Модуль NSS (Служби мережної безпеки).

Підтримка Camellia була додана в остаточний випуск Mozilla Firefox 3 в 2008 році (за замовчуванням відключене починаючи з Firefox 33 в 2014 році в дусі «Пропозиції по зміні стандартних наборів шифрів TLS, пропонованих браузерами», який був виключено з версії 37 в 2015 році). Pale Moon, відгалуження Mozilla / Firefox, продовжує пропонувати Camellia і розширив свою підтримку, включивши в нього набори Galois / Counter mode (GCM) із шифром, але вилучив GCM знову у випуску 27.2.0, пославшись на очевидну відсутність інтересу до них.

Кафедра_КОПЗ_2023рік

Пізніше, в 2008 році, група розробки релізу FreeBSD оголосила, що цей шифр також був включений в FreeBSD 6.4. Крім того, Йошисато Янагисава додав підтримку шифру Camellia у дисковий клас зберігання geli FreeBSD.

У вересні 2009 року GNU Privacy Guard додала підтримку Camellia у версії 1.4.10.

Veracrypt (відгалуження Truecrypt) включав Camellia як один з підтримуваних алгоритмів шифрування.

Крім того, різні популярні бібліотеки безпеки, такі як Crypto ++, Gnutls, mbed TLS і Openssl також включають підтримку Camellia.

26 березня 2013 р. було оголошено, що Camellia була знову обрана для включення в новий список рекомендованих шифрів для електронного уряду Японії як єдиний 128-бітний алгоритм блокового шифрування, розроблений у Японії. Це збігається з тим, що список CRYPTREC обновляється вперше за 10 років. Вибір був заснований на високій репутації Camellia у плані простоти придбання, а також характеристик безпеки й продуктивності, порівнянних з такими з Advanced Encryption Standard (AES). Камелія залишається незмінною у своєму повному втіленні. Неможлива диференціальна атака на Camellia з 12 раундами без шарів FL / FL дійсно існує.

Продуктивність

S-блоки, використовувані Camellia, мають структуру, аналогічну S-блоку AES. У результаті можна прискорити реалізацію програмного забезпечення Camellia за допомогою наборів команд ЦП, розроблених для AES, таких як x86 AES-NI.

Кафедра_КБПЗ_2023рік

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1

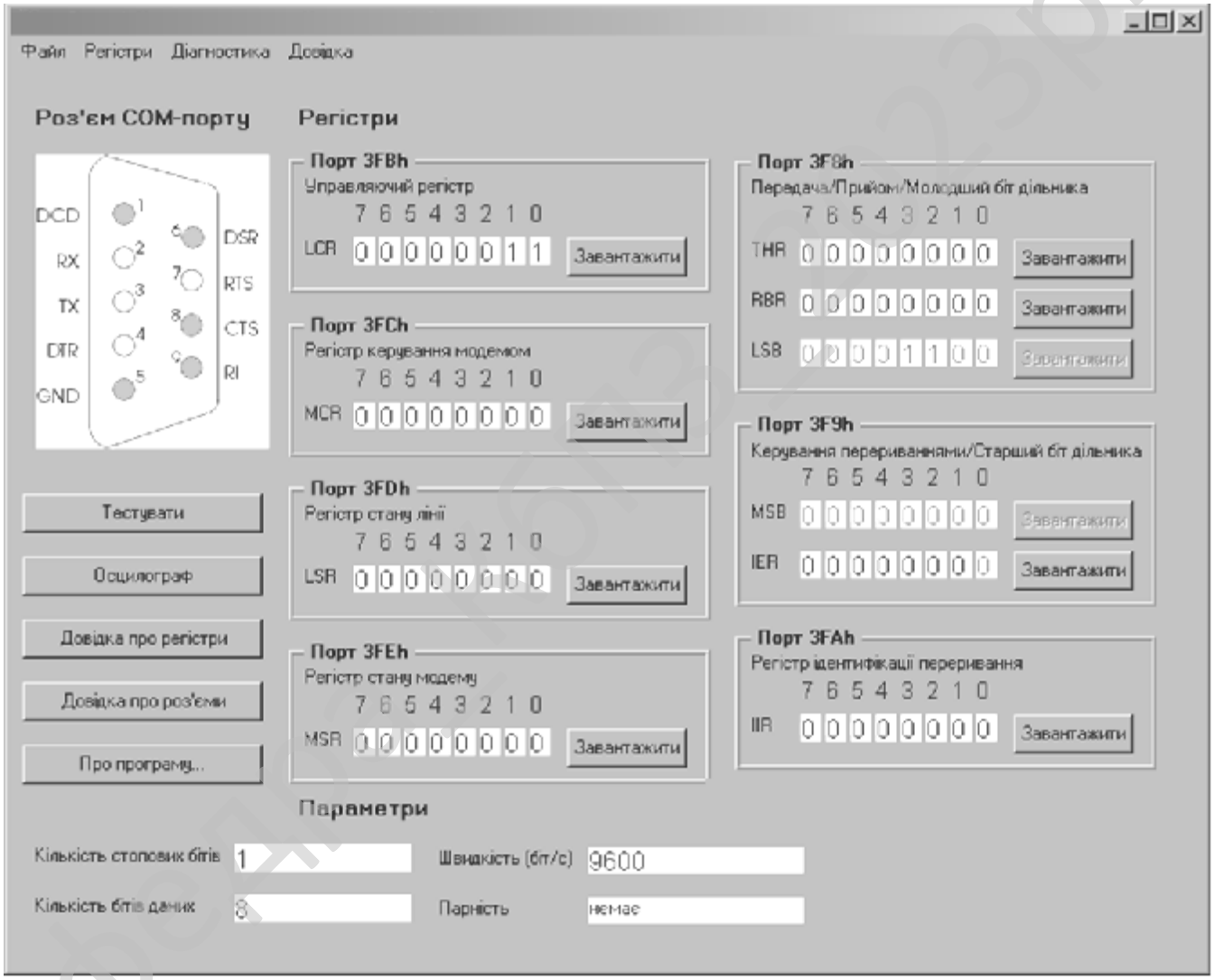

Рисунок 5.1 – Головне вікно програми

Програма складається з таких частин:

– Емуляція роз'єму послідовного порту.

– Емуляція регістрів.
– Покази осцилографа.
– Тестування.
– Інтерактивна довідка.

У верхній правій частині головного вікна програми розміщене зображення стандартного роз'єму послідовного порту з назвами контактів. Якщо на контакті високий рівень сигналу, то його колір стає рожевим, у протилежному випадку – білим.

Також на головному вікні зображені всі регістри послідовного порту, їх назви, адреси та поточне значення. Існує можливість записати в них нове значення за допомогою кнопок "Завантажити" навпроти кожного регістра. При натисненні кнопки навпроти потрібного регістру, з'являється діалогове вікно, що зображене на рисунку 5.2.

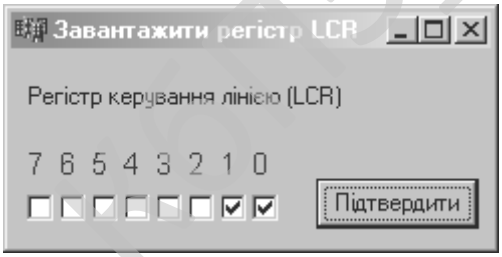

Рисунок 5.2 – Приклад діалогового вікна завантаження регістрів послідовного порту

Запис необхідного значення у регістр здійснюється за допомогою восьми чекбоксів, кожен з яких відповідає одному з восьми біт регістра. Порожній чекбокс відповідає логічному нулю, а з встановленим прапорцем – логічній одиниці. Після встановлення прапорців, для запису отриманого слова у регістр необхідно натиснути кнопку "Підтвердити".

В нижній частині головного вікна програми можна побачити обрані за допомогою управляючого регістра параметри передачі даних, в які входять:

– Швидкість передачі даних.

Кафедра_КПЗ_2023рік

– Парність.

– Кількість стопових біт.

– Кількість біт даних.

Часову діаграму сигналів на контактах роз'єму послідовного порту можна переглянути за допомогою осцилографа (рисунок 5.3), вікно якого відкривається за допомогою кнопки "Осцилограф", або пункту меню Діагностика–>Осцилограф.

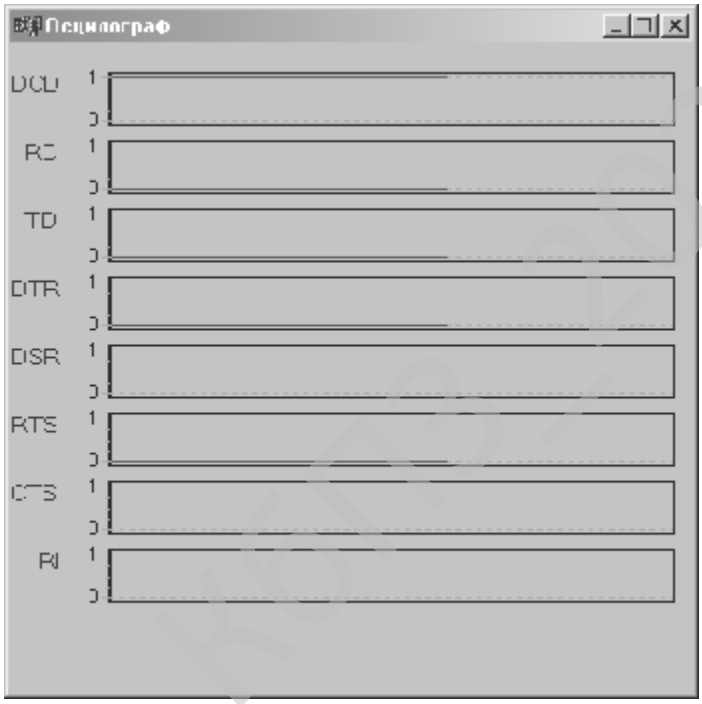

Рисунок 5.3 – Вікно осцилографа

Для того, щоб здійснити тестування послідовного порту слід натиснути кнопку "Тестувати", або пункт меню Діагностика–>Тестувати. Після цього відбудеться перевірка всіх контактів порту та видача результатів.

Якщо тестування виявить помилки, до з'явиться повідомлення з інформацію про те, які саме контакти дали збій (рисунок 5.4).

Кафедра_КОТС_2023рік

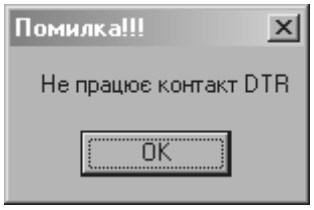

Рисунок 5.4 – Приклад інформаційного повідомлення при виявленні помилки під час тестування

Якщо ж тестування пройде успішно, то з'явиться інформаційне повідомлення зображене на рисунку 5.5.

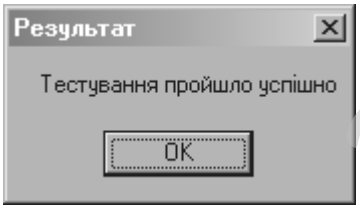

Рисунок 5.5 – Інформаційне повідомлення, що з'являється після успішного тестування послідовного порту

Для зручності роботи з програмою були створені інтерактивна довідка про регістри (рисунок 5.6) та довідка про роз'єм послідовного порту (рисунок 5.7).

Кафедра_КбПЗ_2023рік

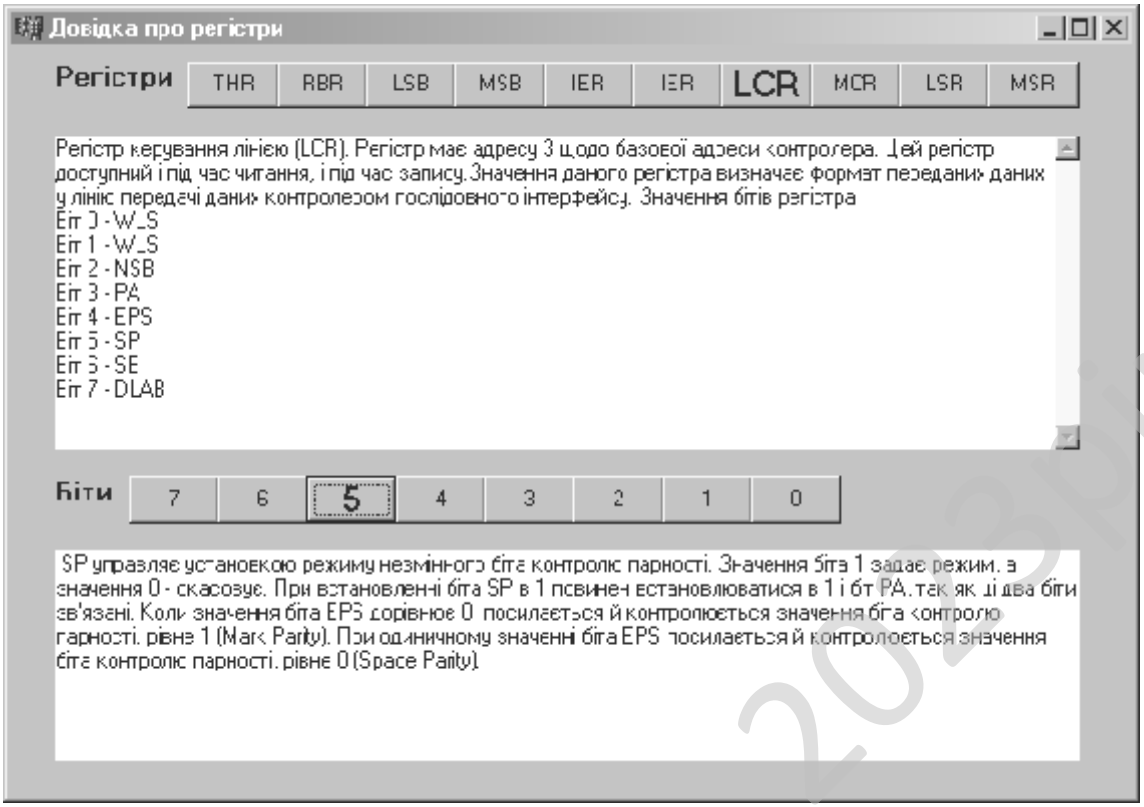

Рисунок 5.6 – Інтерактивна довідка про регістри послідовного порту

Довідка про регістри містить два поля, в першому розміщується інформація про обраний регістр, у другому інформація про обраний біт даного регістра. Вибір регістрів та їх біт здійснюється за допомогою кнопок, що містять назви регістрів та номери біт.

Довідка про роз'єм містить назви та призначення контактів послідовного порту.

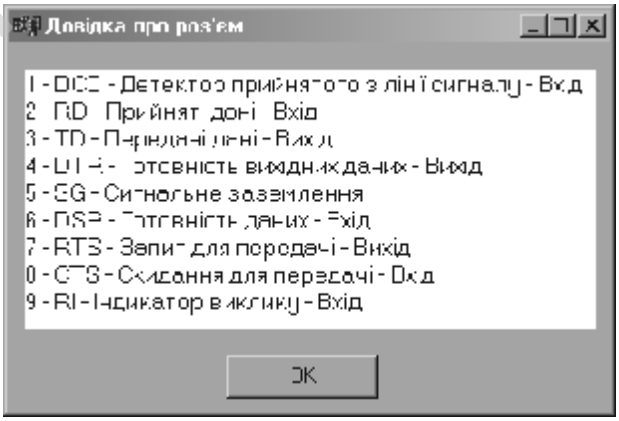

Рисунок 5.7 – Довідка про контакти послідовного порту

Кафедра_КПЗ_2023ІК

Коротку довідку про розроблену систему можна переглянути, натиснувши кнопку "Про програму…" (рисунок 5.8).

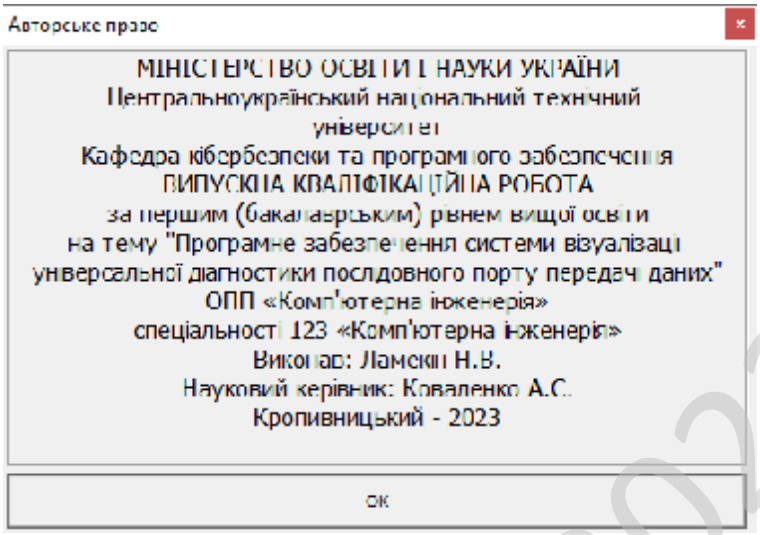

Рисунок 5.8 – Вікно "Про програму…"

Кафедра_КбПЗ_2023рік

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи візуалізації універсальної діагностики послідовного порту передачі даних.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем візуалізації універсальної діагностики послідовного порту передачі даних.

– Досліджена система візуалізації універсальної діагностики послідовного порту передачі даних.

– На основі отриманих результатів досліджень створена програмна реалізація системи візуалізації універсальної діагностики послідовного порту передачі даних.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання візуалізації універсальної діагностики послідовного порту передачі даних.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Builder C++. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

Кафедра_КОПЗ_2023рік

системи візуалізації універсальної діагностики послідовного порту передачі даних. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Camellia.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Кафедра_КБПЗ_2023рік

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаційні технології. Взаємозв'язок відкритих систем. Базова еталонна модель. Частина 1. Базова модель (ISO/IEC 7498-1:1994, IDT):ДСТУ ISO/IEC 7498-1:2004 – [Чинний від 2006–01–01]. – Київ: Держспоживстандарт України, 2007. – 67 с. – (Національний стандарт України).

2. Карманов И.Н. Измерения, испытания, контроль. Метрология и метрологическое обеспечение: учеб. пособ. / И.Н. Карманов, Н.А. Мещеряков, О.К. Ушаков. – Новосибирск: СГГА, 2006. – 184 с.

3. Каспина Т.В. Экономика и управление приборостроительным произвоздством: учебн. пособ. для высших учебных заведений / Т.В. Каспина, Н.Н. Лямина. – М.: ИЦ "Академия", 2008. – 240 с.

4. Клюев В.В. Неразрушающий контроль и диагностика. Справочник, 2-е изд., перераб. и доп. / В.В. Клюев – М: Машиностроение, 2003. – 656 с.

5. Клюня В.Л. Основы экономической теории / В.Л. Клюня, Н.В. Черченко. – Минск: Минск, 2006. – 238 с.

6. Коваленко А.С. Разработка структуры базы данных интегрированной информационной системы / А.С. Коваленко, А.В. Коваленко // Информационные технологии и защита информации в информационно-коммуникационных системах: монографія / Под редакцией профессора В.С. Пономаренко. – Х.: Вид-во ТОВ «Щедра садиба плюс», 2015. – С. 54-64.

7. Кожанова А.С. Обґрунтування необхідності створення систем технічної діагностики інтегрованих інформаційних систем / О.А. Смірнов, А.С. Кожанова, О.В. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2013. – Вип. 6(113). – С. 255-257.

8. Коваленко А.С. Задачи распознавания ситуаций в ERP системах / А.В. Коваленко ., А.А. Смірнов, А.С. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2014. – Вип. 4(120). – С. 161-164.

Кафедра_КБПЗ_2023рік

9. Коваленко А.С. Підсистема технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах / А.С. Коваленко , О.А.Смірнов, О.В. Коваленко // Системи озброєння і військова техніка.– Х.: ХУПС, 2014. – № 1(37). – С. 126-129.

10. Коваленко А.С. Анализ эффективности использования экспертной системы технической диагностики с традиционной структурой / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2014. – № 2(38). – С. 106-108.

11. Коваленко А.С. Разработка структуры экспертной системы технической диагностики интегрированной информационной системы / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Наука і техніка Повітряних Сил Збройних Сил України. – Харків: ХУПС, 2014. – № 2(15). – С.154-157.

12. Коваленко А.С. Разработка структуры экспертной системы технической диагностики интегрированной информационной системы / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Наука і техніка Повітряних Сил Збройних Сил України. – Харків: ХУПС, 2014. – № 2(15). – С.154-157.

13. Коваленко А.С. Структура системи технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Збірник наукових праць Кіровоградського національного технічного університету / техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Кіровоград: Вид-во КНТУ, 2014. – Вип. 27. – С. 245-251.

14. Коваленко А.С. Дослідження будови інтегрованої інформаційної системи та її елементів / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2014. – № 4(40). – С. 85-88.

15. Коваленко А.С. Розробка структури бази даних для обліку технічного стану елементів інтегрованої інформаційної системи з урахуванням вимог споживачів інформації / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2015. – Вип. 1(126). – С. 75-79.

Кафедра_КБПЗ_2023рік

16. Коваленко А.С. Обґрунтування набору даних для оцінки технічного стану інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2015. – Вип. 1(42). – С.39-41.

17. Коваленко А.С. Експертна система технічного діагностування інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2015. – № 1(41). – С. 106-111.

18. Коваленко А.С. Удосконалення методу технічного обслуговування об'єктів інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко, О.П. Доренський // Системи озброєння і військова техніка. – Х.: ХУПС, 2016. – № 2(46). – С. 109-114.

19. Коваленко А.С. Метод визначення оптимального комплексу робіт з відновлення працездатності інтегрованої системи технічної діагностики в умовах ресурсних обмежень / А.С. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2016. – Вип. 3(140). – С. 69-72.

20. Kovalenko A.S. Information model and its element for displaying information on technical condition of objects of integrated information system / A.S. Kovalenko, A.A. Smirnov, A.V. Kovalenko, A.P. Dorensky // International Journal of Computational Engineering Research (IJCER). – India: Delhi, 2016. – Volume 6, Issue 1. – P. 21-27.

21. Кожанова А.С. Система технічної діагностики інтегрованих інформаційних систем – обґрунтування необхідності створення, визначення понятійного апарату та напрямів досліджень / А.С. Кожанова, О.А. Смірнов, М.П. Савченко, Д.М. Ізосімов, В.В. Мороз // Створення та модернізація озброєння і військової техніки в сучасних умовах: Тринадцята наук.-техн. конф., 5-6 вер. 2013 р., м. Феодосія: тези доп. – Феодосія: ДНВЦ, 2013. – С. 187-188.

22. Кожанова А.С. Визначення основних напрямків досліджень щодо створення системи технічної діагностики інтегрованих інформаційних систем

Кафедра_КБПЗ_2023рік

/ А.С. Кожанова, О.А. Смірнов, А.В. Челпанов // Проблемні питання розвитку озброєння та військової техніки Збройних Сил України: IV наук.-техн. конф., 16-20 груд. 2013 р., м. Київ: зб. тез. – Київ: ЦНДІ ОВТ ЗСУ, 2013. – С. 293.

23. Коваленко А.С. Обґрунтування необхідності створення систем технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Інформатика та системні науки : V Всеукр. наук.-практ. конф., 13–15 бер. 2014 р., м. Полтава : зб. тез. – Полтава: ПУЕТ, 2014. – С. 292-294.

24. Коваленко А.С. Задачи распознавания ситуаций в системах организационной стратегии интеграции производства и операций / А.С. Коваленко, А.В. Коваленко // Комбінаторні конфігурації та їх застосування: XVI міжнар. наук.-практ. сем., 11-12 квіт. 2014 р., м. Кіровоград: зб. тез. – Кіровоград: КНТУ, 2014. – С. 53-55.

25. Коваленко А.С. Створення систем технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Проблеми і перспективи розвитку ІТ-індустрії: VI між нар. наук.-практ. конф., 17-18 квіт. 2014 р., м. Харків: зб. тез. – Харків: ХНЕУ, 2014. – С. 241.

26. Коваленко А.С. Визначення понятійного апарату та напрямів досліджень для синтезу систем технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Комп'ютерне моделювання у наукоємних технологіях (КМНТ-2014): наук.-техн. конф. з міжнар. участю, 28-31 трав. 2014 р., м. Харків: зб. наук. праць. – Харків: ХНУ, 2014. – С. 190-193.

27. Коваленко А.С. Основні складові та функції системи технічної діагностики інтегрованих інформаційних систем / Коваленко А.С. // Інформаційні технології та комп'ютерна інженерія: наук.-практ. конф., 4 груд. 2014 р., м. Кіровоград: зб. тез доп. – Кіровоград: КНТУ, 2014. – С. 236.

Кафедра_КОПЗ_2023рік

28. Коваленко А.С. Розробка структури бази даних інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Проблеми і перспективи розвитку ІТ-індустрії: VII міжнар. наук.-практ. конф., 17-18 квіт. 2015 р., м. Харків: зб. тез. – Харків: ХНЕУ, 2015. – С. 15.

29. Коваленко А.С. Дослідження елементів інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Комбінаторні конфігурації та їх застосування: XVII між нар. наук.-практ. сем., 17-18 квіт. 2015 р., м. Кіровоград: зб. тез – Кіровоград: КНТУ, 2015. – С. 5.

30. Коваленко А.С. Метод автоматизованої перевірки результатів вимірювання параметрів об'єкті в інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Стратегія якості у промисловості і освіті: XI міжнар. конф., 1 – 5 черв. 2015 р., м. Варна, Болгарія.: зб. матер. – Варна: ТУВ, 2015. – С. 423-426.

31. Коваленко А.С. Обґрунтування необхідності створення розподіленої бази даних для забезпечення захисту рухомих повітряних об`єктів / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Перспективні напрями захисту інформації: І всеукр. наук.-практ. конф., 07 вер. 2015 р., м. Одеса: зб. тез доп. – Одеса: ОНАЗ, 2015. – С. 35-39.

32. Коваленко А.С. Розробка інформаційної моделі автоматизованої оцінки технічного стану інтегральної інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Інформаційні технології та взаємодії (IT & I): II між нар. наук.-практ. конф., 3-5 лист. 2015 р., м. Київ: тези доп. – Київ: КНУ ім. Т. Шевченка, 2015. – С. 41-42.

33. Коваленко А.С. Разработка метода усовершенствования технического обслуживания интегрированной информационной системы / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Информационные и телекоммуникационные технологии: образование, наука, практика: II междунар. научн.-практ. конф., 3-4 дек. 2015 г., г. Алматы, Казахстан: сб. труд. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – Т.2. – С. 423-427.

Кафедра_КОПЗ – 2023рік

34. Королюк Н.А. Оценка временных интервалов работы лица, принимающего решение, на автоматизированном командном пункте / Н.А. Королюк, А.И. Тимочко // Системи обробки інформації. – Х.: ХУПС, 2005. – Вип. 8 (48). – С. 51-54.

35. Костерев В.В. Надёжность технических систем и управление риском: учебн. пособ. / В.В. Костерев. – М.: МИФИ, 2008. – 280 с.

36. Костюков А.В. Підвищення операційної ефективності підприємств на основі моніторингу в реальному часі. / А.В. Костюков, В.М. Костюков. – М.: Машинобудування, 2009. – 192 с.

37. Лазарев А.А. Выбор показателя затрат для анализа сравнительной экономической эффективности техники конечного потребления / А.А. Лазарев, М.В. Бейлин // Сборник научных трудов ХГПУ.– Х.: ХГПУ, 1999. – Вып. 74. – С. 27-29.

38. Ланецкий Б.Н. Основы теории надежности, технического обслуживания и ремонта вооружения и военной техники: Справочные материалы, часть 1. / Б.Н. Ланецкий, А.А. Посудевский. – Харьков: ХВУ, 1993. – 308 с.

39. Ланецкий Б.Н. Основы теории надежности, технического обслуживания и ремонта вооружения и военной техники: Справочные материалы, часть 2. / Б.Н. Ланецкий, А.А. Посудевский. – Харьков: ХВУ, 1993. – 208 с.

40. Лапсарь А.П. Метод оценки состояния сложных технических объектов для синтеза быстродействующих прогнозирующих систем / А.П. Лапсарь // Измерительная техника. – 2004. – № 2. – С. 7-10.

41. Линейные задачи оптимизации: Учеб. пособие / С.В. Лутманов. – Пермь: ЛИТЕР-А, 2004. – Ч.1. – Линейное программирование. – 128 с.

42. Литвак Б.Г. Экспертные технологии в управлении. Учебное пособие / Б.Г. Литвак. – М.: Дело, 2014. – 318 с.

43. Локазюк В.М. Надійність, контроль, діагностика і модернізація ПК: Посібн. / В.М. Локазюк, Ю.Г. Савченко. – К.: Видавничий центр «Академія», 2004. – 376 с.

Кафедра_КБПЗ_2023рік

44. Лопатников Л. И. Экономико-математический словарь: Словарь современной экономической науки / Л.И. Лопатников. – М.: Дело, 2003. – 520 с.

45. Манухина С.Ю. Инженерная психология и эргономика: хрестоматия / С.Ю Манухина. – М.: Изд. центр ЕАОИ, 2009. –224 с.

46. Мартыненко М.В. Человекомашинные процедуры поддержки организационно–управленческих решений: учеб. пособие СПбГЭТУ / М.В. Мартыненко, О.И. Шеховцов. – СПб, 2012. – 250 с.

47. Мунипов О.В. Эргономика: человекоориентированное проектирование техники, программных средств и среды: Учебник / О.В. Мунипов, В.П. Зинченко. – М.: Логос, 2001. – 356 с.

48. Надеев А.И. Математическая модель эксплуатационной надежности интеллектуальных датчиков / А.И. Надеев, Р.А. Юсупов, Ю.К. Свечников, Д.Р. Юсупов // Измерительная техника. – М: Стандартинформ, 2004. – № 1. – С. 8-11.

49. Надійність техніки. Аналіз надійності. Основні положення: ДСТУ 2861-94 – [Чинний від 1997–01–01]. – Київ: Держстандарт України, 1995. – 33 с. – (Національний стандарт України).

50. Надійність техніки. Терміни та визначення: ДСТУ 2860-94 – [Чинний від 1996–01–01]. – Київ: Держстандарт України, 1994. – 36 с. – (Національний стандарт України).

51. Нейлор К. Как построить свою экспертную систему / К. Нейлор. – М.: Энегроатомиздат, 2007. – 242 с.

КафеДра_КбПЗ_2023рік

Додаток А
(обов'язковий)

Технічне завдання

Зміст

Кафедра_КОПЗ_2023рік

					ВКРБ-123.23.0004.00.00.ТЗ	
Вим.	Арк.	№ документа	Підпис	Дата		
Розробив		Ламекін Н.В.			Програмне забезпечення системи візуалізації універсальної діагностики послідовного порту передачі даних	Літ. Б; Аркуш 1; Аркушів 6
Перевірив		Коваленко А.С.				
Н. Контр.		Гермак В.С.				ЦНТУ КІ-19
Затв.		Смірнов О.А.				

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи візуалізації універсальної діагностики послідовного порту передачі даних.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 7-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи візуалізації універсальної діагностики послідовного порту передачі даних.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

Кафедра_КБПЗ_2023рік

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи візуалізації універсальної діагностики послідовного порту передачі даних;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

Кафедра_КОПЗ_2023рік

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп'ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14" 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Builder C++.

Кафедра_КБПЗ_2023рік

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 86 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

Кафедра_КБПЗ_2023рік

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2023 р.

11.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 1.06.2023 р.

Кафедра_КБПЗ_2023рік

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Коваленко А.С.

Програмне забезпечення системи візуалізації універсальної діагностики послідовного порту передачі даних

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 57

Літера: РП

Кропивницький – 2023 року

Кафедра_КбПЗ_2023рік

Основна програма

Файл Project1.cpp основної програми

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------
USEFORM("main.cpp", Form1);
USEFORM("graf.cpp", Form2);
USEFORM("about.cpp", Form3);
USEFORM("help.cpp", Form4);
USEFORM("help2.cpp", Form5);
USEFORM("write1.cpp", Form6);
USEFORM("write2.cpp", Form7);
USEFORM("write3.cpp", Form8);
USEFORM("write4.cpp", Form9);
USEFORM("write5.cpp", Form10);
USEFORM("write6.cpp", Form11);
USEFORM("write7.cpp", Form12);
USEFORM("write8.cpp", Form13);
USEFORM("write9.cpp", Form14);
USEFORM("write10.cpp", Form15);
//---------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
        try
        {
             Application->Initialize();
             Application->CreateForm(__classid(TForm1), &Form1);
             Application->CreateForm(__classid(TForm2), &Form2);
             Application->CreateForm(__classid(TForm3), &Form3);
             Application->CreateForm(__classid(TForm4), &Form4);
             Application->CreateForm(__classid(TForm5), &Form5);
             Application->CreateForm(__classid(TForm6), &Form6);
             Application->CreateForm(__classid(TForm7), &Form7);
             Application->CreateForm(__classid(TForm8), &Form8);
             Application->CreateForm(__classid(TForm9), &Form9);
             Application->CreateForm(__classid(TForm10), &Form10);
             Application->CreateForm(__classid(TForm11), &Form11);
             Application->CreateForm(__classid(TForm12), &Form12);
             Application->CreateForm(__classid(TForm13), &Form13);
             Application->CreateForm(__classid(TForm14), &Form14);
             Application->CreateForm(__classid(TForm15), &Form15);
             Application->Run();
        }

         catch (Exception &exception)
        {
             Application->ShowException(&exception);
        }
        catch (...)
        {
                 try
                 {
                         throw Exception("");
                 }
                 catch (Exception &exception)
                 {
                     Application->ShowException(&exception);
                 }
        }
        return 0;
}
//---------------------------------------------------------------------
```

Кафедра КОПЗ_2023рік

Кафедра_КбПЗ_2023рік

Файл Main.cpp основної програми

```
//---------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "main.h"
#include "graf.h"
#include "about.h"
#include "help.h"
#include "help2.h"
#include "write1.h"
#include "write2.h"
#include "write3.h"
#include "write4.h"
#include "write5.h"
#include "write6.h"
#include "write7.h"
#include "write8.h"
#include "write9.h"
#include "write10.h"


//---------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
int d1,d2,d3,d4,d5,d6,d7,d8;
//---------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------

//Процедура, що обробляє натиснення кнопки вмикання/вимикання осцилографа

void __fastcall TForm1::Button12Click(TObject *Sender)
{
if(Form2->Visible==false) Form2->Show();
else if(Form2->Visible==true) Form2->Close();
}
//---------------------------------------------------------------

//Відкриття вікна довідки про роз'єми

void __fastcall TForm1::Button15Click(TObject *Sender)
{
Form5->Show();
}
//---------------------------------------------------------------



void __fastcall TForm1::N19Click(TObject *Sender)
{
Form5->Show();
}
//---------------------------------------------------------------

//Відкриття вікна "Про програму..."

void __fastcall TForm1::N15Click(TObject *Sender)
{
Form3->Show();
}
//---------------------------------------------------------------
```

Кафедра_КБПЗ_2023рік

```
void __fastcall TForm1::Button11Click(TObject *Sender)
{
Form3->Show();
}
//---------------------------------------------------------------------

//Відкриття вікна інтерактивної довідки про регістри

void __fastcall TForm1::Button14Click(TObject *Sender)
{
Form4->Show();
}
//---------------------------------------------------------------------

void __fastcall TForm1::N18Click(TObject *Sender)
{
Form4->Show();
}
//---------------------------------------------------------------------

//Відкриття вікна з показами осцилографа через відповідний пункт меню програми

void __fastcall TForm1::N17Click(TObject *Sender)
{
Form2->Show();
}
//---------------------------------------------------------------------

//Відкриття діалогового вікна запису у регістр THR

void __fastcall TForm1::Button1Click(TObject *Sender)
{
Form6->Show();
}
//---------------------------------------------------------------------

void __fastcall TForm1::N4Click(TObject *Sender)
{
Form6->Show();
}
//---------------------------------------------------------------------

//Відкриття діалогового вікна запису у регістр RBR

void __fastcall TForm1::Button2Click(TObject *Sender)
{
Form7->Show();
}
//---------------------------------------------------------------------

void __fastcall TForm1::N5Click(TObject *Sender)
{
Form7->Show();
}
//---------------------------------------------------------------------

//Відкриття діалогового вікна запису у регістр LSB

void __fastcall TForm1::Button3Click(TObject *Sender)
{
Form8->Show();
}

//---------------------------------------------------------------------
void __fastcall TForm1::N6Click(TObject *Sender)
{
Form8->Show();
}
```

Кафедра_КБПЗ_2023рік

```
//---------------------------------------------------------------

//Відкриття діалогового вікна запису у регістр MSB

void __fastcall TForm1::Button4Click(TObject *Sender)
{
Form9->Show();
}
//---------------------------------------------------------------

void __fastcall TForm1::N7Click(TObject *Sender)
{
Form9->Show();
}
//---------------------------------------------------------------

//Відкриття діалогового вікна запису у регістр IER

void __fastcall TForm1::Button5Click(TObject *Sender)
{
Form10->Show();
}

//---------------------------------------------------------------
void __fastcall TForm1::N8Click(TObject *Sender)
{
Form10->Show();
}
//---------------------------------------------------------------


//Відкриття діалогового вікна запису у регістр IIR

void __fastcall TForm1::Button6Click(TObject *Sender)
{
Form11->Show();
}

//---------------------------------------------------------------
void __fastcall TForm1::N9Click(TObject *Sender)
{
Form11->Show();
}
//---------------------------------------------------------------

//Відкриття діалогового вікна запису у регістр LCR

void __fastcall TForm1::Button7Click(TObject *Sender)
{
Form12->Show();
}

//---------------------------------------------------------------
void __fastcall TForm1::N10Click(TObject *Sender)
{
Form12->Show();
}
//---------------------------------------------------------------

//Відкриття діалогового вікна запису у регістр MCR

void __fastcall TForm1::Button8Click(TObject *Sender)
{
Form13->Show();
}
//---------------------------------------------------------------

void __fastcall TForm1::N11Click(TObject *Sender)
{
```

Кафедра_КбПЗ_2023рік

```
Form13->Show();
}
//---------------------------------------------------------------

//Відкриття діалогового вікна запису у регістр LSR

void __fastcall TForm1::Button9Click(TObject *Sender)
{
Form14->Show();
}
//---------------------------------------------------------------

void __fastcall TForm1::N12Click(TObject *Sender)
{
Form14->Show();
}
//---------------------------------------------------------------

//Відкриття діалогового вікна запису у регістр MSR

void __fastcall TForm1::Button13Click(TObject *Sender)
{
Form15->Show();
}
//---------------------------------------------------------------

void __fastcall TForm1::N14Click(TObject *Sender)
{
Form16->Show();
}
//---------------------------------------------------------------

//Виведення поточного стану послідовного порту

void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
int x;

if(MSR7->Text=='1') ImageDCD->Visible=false; else if(MSR7->Text=='0') ImageDCD-
>Visible=true;
if(MCR0->Text=='1') ImageDTR->Visible=true; else if(MCR0->Text=='0') ImageDTR-
>Visible=false;
if(MSR5->Text=='1') ImageDSR->Visible=false; else if(MSR5->Text=='0') ImageDSR-
>Visible=true;
if(MCR1->Text=='1') ImageRTS->Visible=true; else if(MCR1->Text=='0') ImageRTS-
>Visible=false;
if(MSR4->Text=='1') ImageCTS->Visible=false; else if(MSR4->Text=='0') ImageCTS-
>Visible=true;
if(MSR6->Text=='1') ImageRI->Visible=false; else if(MSR6->Text=='0') ImageRI-
>Visible=true;
if(LCR7->Text=='1') {
Button1->Enabled=false;
Button2->Enabled=false;
Button3->Enabled=true;
Button4->Enabled=true;
Button5->Enabled=false;
THR0->Enabled=false;
THR1->Enabled=false;
THR2->Enabled=false;
THR3->Enabled=false;
THR4->Enabled=false;
THR5->Enabled=false;
THR6->Enabled=false;
THR7->Enabled=false;
RBR0->Enabled=false;
RBR1->Enabled=false;
RBR2->Enabled=false;
RBR3->Enabled=false;
RBR4->Enabled=false;
```

Кафедра_КБПЗ_2023рік

```
RBR5->Enabled=false;
RBR6->Enabled=false;
RBR7->Enabled=false;
LSB0->Enabled=true;
LSB1->Enabled=true;
LSB2->Enabled=true;
LSB3->Enabled=true;
LSB4->Enabled=true;
LSB5->Enabled=true;
LSB6->Enabled=true;
LSB7->Enabled=true;

IER1->Enabled=false;
IER2->Enabled=false;
IER3->Enabled=false;
IER4->Enabled=false;
IER5->Enabled=false;
IER6->Enabled=false;
IER7->Enabled=false;
MSB0->Enabled=true;
MSB1->Enabled=true;
MSB2->Enabled=true;
MSB3->Enabled=true;
MSB4->Enabled=true;
MSB5->Enabled=true;
MSB6->Enabled=true;
MSB7->Enabled=true;
}
else if(LCR7->Text=='0')  {
Button1->Enabled=true;
Button2->Enabled=true;
Button3->Enabled=false;
Button4->Enabled=false;
Button5->Enabled=true;
THR0->Enabled=true;
THR1->Enabled=true;
THR2->Enabled=true;
THR3->Enabled=true;
THR4->Enabled=true;
THR5->Enabled=true;
THR6->Enabled=true;
THR7->Enabled=true;
RBR0->Enabled=true;
RBR1->Enabled=true;
RBR2->Enabled=true;
RBR3->Enabled=true;
RBR4->Enabled=true;
RBR5->Enabled=true;
RBR6->Enabled=true;
RBR7->Enabled=true;
LSB0->Enabled=false;
LSB1->Enabled=false;
LSB2->Enabled=false;
LSB3->Enabled=false;
LSB4->Enabled=false;
LSB5->Enabled=false;
LSB6->Enabled=false;
LSB7->Enabled=false;

IER1->Enabled=true;
IER2->Enabled=true;
IER3->Enabled=true;
IER4->Enabled=true;
IER5->Enabled=true;
IER6->Enabled=true;
IER7->Enabled=true;
MSB0->Enabled=false;
MSB1->Enabled=false;
MSB2->Enabled=false;
```

Кафедра_КбПЗ_2023рік

```
MSB3->Enabled=false;
MSB4->Enabled=false;
MSB5->Enabled=false;
MSB6->Enabled=false;
MSB7->Enabled=false;
}

x=StrToInt(LCR1->Text)*10+StrToInt(LCR0->Text);
if(x==0) DataBit->Text=5;
if(x==1) DataBit->Text=6;
if(x==10) DataBit->Text=7;
if(x==11) DataBit->Text=8;

x=StrToInt(LCR2->Text);
if(x==0) StopBit->Text=1;
if(x==1) StopBit->Text=2;

x=StrToInt(LCR4->Text)*10+StrToInt(LCR3->Text);
if((x==0)|(x==10)) Parn->Text="немає";
if(x==1) Parn->Text="контроль на непарність";
if(x==11) Parn->Text="контроль на парність";

}
//---------------------------------------------------------------------

void __fastcall TForm1::N2Click(TObject *Sender)
{
Form1->Close();
}
//---------------------------------------------------------------------

// Тестування контактів послідовного порту

void __fastcall TForm1::Button10Click(TObject *Sender)
{

MSR7->Text='0';
MCR0->Text='1';
MSR5->Text='0';
MCR1->Text='1';
MSR4->Text='0';
MSR6->Text='0';

if((ImageDCD->Visible==false)|(ImageDCD->Visible==false)|(ImageDTR-
>Visible==false)|(ImageDSR->Visible==false)|(ImageRTS-
>Visible==false)|(ImageCTS->Visible==false)|(ImageRI->Visible==false))
{
if(ImageDCD->Visible==false) Application->MessageBox("Не працює контакт DCD",
"Помилка!!!", MB_OK);

if(ImageDTR->Visible==false) Application->MessageBox("Не працює контакт DTR",
"Помилка!!!", MB_OK);

if(ImageDSR->Visible==false) Application->MessageBox("Не працює контакт DSR",
"Помилка!!!", MB_OK);

if(ImageRTS->Visible==false) Application->MessageBox("Не працює контакт RTS",
"Помилка!!!", MB_OK);

if(ImageCTS->Visible==false) Application->MessageBox("Не працює контакт CTS",
"Помилка!!!", MB_OK);

if(ImageRI->Visible==false) Application->MessageBox("Не працює контакт RI",
"Помилка!!!", MB_OK);
}
else Application->MessageBox("Тестування пройшло успішно", "Результат", MB_OK);
}
//---------------------------------------------------------------------
```

Кафедра_КбПЗ_2023рік

Кафедра_КбПЗ_2023рік

Файл Main.h – бібліотека для файлу Main.cpp

```
//---------------------------------------------------------------

#ifndef mainH
#define mainH
//---------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
#include <Buttons.hpp>
#include <Menus.hpp>
#include <Chart.hpp>
#include <Series.hpp>
#include <TeEngine.hpp>
#include <TeeProcs.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
        TImage *Image1;
        TImage *ImageDCD;
        TImage *Image3RX;
        TImage *ImageTX;
        TImage *ImageDTR;
        TImage *ImageGND;
        TImage *ImageDSR;
        TImage *ImageRTS;
        TImage *ImageCTS;
        TImage *ImageRI;
        TLabel *Label1;
        TLabel *Label2;
        TEdit *THR7;
        TEdit *THR6;
        TEdit *THR5;
        TEdit *THR4;
        TEdit *THR3;
        TEdit *THR2;
        TEdit *THR1;
        TEdit *THR0;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TLabel *Label9;
        TLabel *Label10;
        TEdit *RBR7;
        TEdit *RBR6;
        TEdit *RBR5;
        TEdit *RBR4;
        TEdit *RBR3;
        TEdit *RBR2;
        TEdit *RBR1;
        TEdit *RBR0;
        TEdit *LSB7;
        TEdit *LSB6;
        TEdit *LSB5;
        TEdit *LSB4;
        TEdit *LSB3;
        TEdit *LSB2;
        TEdit *LSB1;
        TEdit *LSB0;
```

Кафедра_КбПЗ_2023рік

```
TButton *Button1;
TButton *Button2;
TButton *Button3;
TEdit *MSB7;
TEdit *MSB6;
TEdit *MSB5;
TEdit *MSB4;
TEdit *MSB3;
TEdit *MSB2;
TEdit *MSB1;
TEdit *MSB0;
TLabel *Label13;
TLabel *Label14;
TLabel *Label15;
TLabel *Label16;
TLabel *Label17;
TLabel *Label18;
TLabel *Label19;
TLabel *Label20;
TEdit *IER7;
TEdit *IER6;
TEdit *IER5;
TEdit *IER4;
TEdit *IER3;
TEdit *IER2;
TEdit *IER1;
TEdit *IER0;
TButton *Button4;
TButton *Button5;
TEdit *IIR7;
TEdit *IIR6;
TEdit *IIR5;
TEdit *IIR4;
TEdit *IIR3;
TEdit *IIR2;
TEdit *IIR1;
TEdit *IIR0;
TLabel *Label22;
TLabel *Label23;
TLabel *Label24;
TLabel *Label25;
TLabel *Label26;
TLabel *Label27;
TLabel *Label28;
TLabel *Label29;
TButton *Button6;
TEdit *LCR7;
TEdit *LCR6;
TEdit *LCR5;
TEdit *LCR4;
TEdit *LCR3;
TEdit *LCR2;
TEdit *LCR1;
TEdit *LCR0;
TLabel *Label31;
TLabel *Label32;
TLabel *Label33;
TLabel *Label34;
TLabel *Label35;
TLabel *Label36;
TLabel *Label37;
TLabel *Label38;
TButton *Button7;
TEdit *MCR7;
TEdit *MCR6;
TEdit *MCR5;
TEdit *MCR4;
TEdit *MCR3;
TEdit *MCR2;
```

Кафедра_КбПЗ_2023рік

```
TEdit *MCR1;
TEdit *MCR0;
TLabel *Label40;
TLabel *Label41;
TLabel *Label42;
TLabel *Label43;
TLabel *Label44;
TLabel *Label45;
TLabel *Label46;
TLabel *Label47;
TButton *Button8;
TEdit *LSR7;
TEdit *LSR6;
TEdit *LSR5;
TEdit *LSR4;
TEdit *LSR3;
TEdit *LSR2;
TEdit *LSR1;
TEdit *LSR0;
TLabel *Label49;
TLabel *Label50;
TLabel *Label51;
TLabel *Label52;
TLabel *Label53;
TLabel *Label54;
TLabel *Label55;
TLabel *Label56;
TButton *Button9;
TMainMenu *MainMenu1;
TMenuItem *N1;
TMenuItem *N2;
TMenuItem *N3;
TMenuItem *N4;
TMenuItem *N5;
TMenuItem *N6;
TMenuItem *N7;
TMenuItem *N8;
TMenuItem *N9;
TMenuItem *N10;
TMenuItem *N11;
TMenuItem *N12;
TMenuItem *N13;
TMenuItem *N14;
TMenuItem *N15;
TButton *Button10;
TButton *Button12;
TBevel *Bevel1;
TBevel *Bevel2;
TBevel *Bevel3;
TBevel *Bevel4;
TBevel *Bevel5;
TBevel *Bevel6;
TBevel *Bevel7;
TBevel *Bevel8;
TLabel *Label11;
TLabel *Label12;
TBevel *Bevel9;
TLabel *Label21;
TBevel *Bevel10;
TLabel *Label30;
TBevel *Bevel11;
TBevel *Bevel12;
TLabel *Label39;
TLabel *Label48;
TLabel *Label57;
TLabel *Label58;
TLabel *Label59;
TLabel *Label60;
TLabel *Label61;
```

Кафедра_КбПЗ_2023рік

```
TLabel *Label62;
TLabel *Label63;
TLabel *Label64;
TLabel *Label65;
TLabel *Label66;
TLabel *Label67;
TLabel *Label68;
TLabel *Label69;
TLabel *Label70;
TLabel *Label71;
TLabel *Label72;
TLabel *Label73;
TEdit *MSR7;
TEdit *MSR6;
TEdit *MSR5;
TEdit *MSR4;
TEdit *MSR3;
TEdit *MSR2;
TEdit *MSR11;
TEdit *MSR0;
TButton *Button13;
TBevel *Bevel13;
TBevel *Bevel14;
TLabel *Label74;
TLabel *Label75;
TMenuItem *N16;
TMenuItem *N17;
TMenuItem *N18;
TMenuItem *N19;
TLabel *Label76;
TEdit *Edit81;
TLabel *Label77;
TEdit *DataBit;
TLabel *Label78;
TEdit *Parn;
TLabel *Label79;
TEdit *StopBit;
TLabel *Label80;
TButton *Button11;
TButton *Button15;
TButton *Button14;
TMenuItem *MSR1;
TTimer *Timer1;
TLabel *Label81;
TLabel *Label82;
TLabel *Label83;
TLabel *Label84;
TLabel *Label85;
TLabel *Label86;
TLabel *Label87;
void __fastcall Button12Click(TObject *Sender);
void __fastcall Button15Click(TObject *Sender);
void __fastcall N15Click(TObject *Sender);
void __fastcall Label57Click(TObject *Sender);
void __fastcall Button11Click(TObject *Sender);
void __fastcall Button14Click(TObject *Sender);
void __fastcall N19Click(TObject *Sender);
void __fastcall N18Click(TObject *Sender);
void __fastcall N17Click(TObject *Sender);
void __fastcall Button1Click(TObject *Sender);
void __fastcall Button2Click(TObject *Sender);
void __fastcall Button3Click(TObject *Sender);
void __fastcall Button4Click(TObject *Sender);
void __fastcall Button5Click(TObject *Sender);
void __fastcall Button6Click(TObject *Sender);
void __fastcall Button7Click(TObject *Sender);
void __fastcall Button8Click(TObject *Sender);
void __fastcall Button9Click(TObject *Sender);
void __fastcall Button13Click(TObject *Sender);
```

Кафедра_КбПЗ_2023рік

```
        void __fastcall Timer1Timer(TObject *Sender);
        void __fastcall FormCreate(TObject *Sender);
        void __fastcall N5Click(TObject *Sender);
        void __fastcall N6Click(TObject *Sender);
        void __fastcall N4Click(TObject *Sender);
        void __fastcall N7Click(TObject *Sender);
        void __fastcall N8Click(TObject *Sender);
        void __fastcall N9Click(TObject *Sender);
        void __fastcall N10Click(TObject *Sender);
        void __fastcall N11Click(TObject *Sender);
        void __fastcall MSR1Click(TObject *Sender);
        void __fastcall N12Click(TObject *Sender);
        void __fastcall N2Click(TObject *Sender);
        void __fastcall Button10Click(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Кафедра_КбПЗ_2023рік

Файл graf.cpp – виведення на екран показів осцилографа

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
#include "graf.h"
#include "main.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
int t1,t2,t3,t4,t5,t6,t7,t8;
//---------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------

//Часова діаграма сигналу на контакті DCD

void __fastcall TForm2::Timer1Timer(TObject *Sender)
{
int d;
d=(StrToInt(Form1->MSR7->Text)+1)%2;
Series1->Add(d,t1,clRed);
t1++;
if(t1==20) {t1=0; Series1->Clear();}
}
//---------------------------------------------------------------------

//Часова діаграма сигналу на контакті RD

void __fastcall TForm2::Timer2Timer(TObject *Sender)
{
Series2->Add(0,t2,clRed);
t2++;
if(t2==20) {t2=0; Series2->Clear();}
}
//---------------------------------------------------------------------

//Часова діаграма сигналу на контакті TD

void __fastcall TForm2::Timer3Timer(TObject *Sender)
{

Series3->Add(0,t3,clRed);
t3++;
if(t3==20) {t3=0; Series3->Clear();}
}
//---------------------------------------------------------------------



//Часова діаграма сигналу на контакті DTR

void __fastcall TForm2::Timer4Timer(TObject *Sender)
{
int d;
d=StrToInt(Form1->MCR0->Text);
Series4->Add(d,t4,clRed);
t4++;
if(t4==20) {t4=0; Series4->Clear();}
}
//---------------------------------------------------------------------
```

Кафедра_КБПЗ_2023рік

```
//Часова діаграма сигналу на контакті DSR

void __fastcall TForm2::Timer5Timer(TObject *Sender)
{
int d;
d=(StrToInt(Form1->MSR5->Text)+1)%2;
Series5->Add(d,t5,clRed);
t5++;
if(t5==20) {t5=0; Series5->Clear();}
}
//---------------------------------------------------------------

//Часова діаграма сигналу на контакті RTS

void __fastcall TForm2::Timer6Timer(TObject *Sender)
{
int d;
d=StrToInt(Form1->MCR1->Text);
Series6->Add(d,t6,clRed);
t6++;
if(t6==20) {t6=0; Series6->Clear();}
}
//---------------------------------------------------------------

//Часова діаграма сигналу на контакті CTS

void __fastcall TForm2::Timer7Timer(TObject *Sender)
{
int d;
d=(StrToInt(Form1->MSR4->Text)+1)%2;
Series7->Add(d,t7,clRed);
t7++;
if(t7==20) {t7=0; Series7->Clear();}
}
//---------------------------------------------------------------
//Часова діаграма сигналу на контакті RI

void __fastcall TForm2::Timer8Timer(TObject *Sender)
{
int d;
d=(StrToInt(Form1->MSR6->Text)+1)%2;
Series8->Add(d,t8,clRed);
t8++;
if(t8==20) {t8=0; Series8->Clear();}
}
```

Кафедра_КбПЗ_2023рік

Файл graf.h – бібліотека для файлу graf.cpp

```
//---------------------------------------------------------------------

#ifndef grafH
#define grafH
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Chart.hpp>
#include <ExtCtrls.hpp>
#include <Series.hpp>
#include <TeEngine.hpp>
#include <TeeProcs.hpp>
//---------------------------------------------------------------------
class TForm2 : public TForm
{
__published:      // IDE-managed Components
        TChart *Chart1;
        TLineSeries *Series1;
        TLabel *Label1;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TLabel *Label9;
        TChart *Chart2;
        TLineSeries *Series2;
        TChart *Chart3;
        TLineSeries *Series3;
        TChart *Chart4;
        TLineSeries *Series4;
        TChart *Chart5;
        TLineSeries *Series5;
        TChart *Chart6;
        TLineSeries *Series6;
        TChart *Chart7;
        TLineSeries *Series7;
        TChart *Chart8;
        TLineSeries *Series8;
        TTimer *Timer1;
        TTimer *Timer2;
        TTimer *Timer3;
        TTimer *Timer4;
        TTimer *Timer5;
        TTimer *Timer6;
        TTimer *Timer7;
        TTimer *Timer8;
        TLabel *Label2;
        void __fastcall Timer1Timer(TObject *Sender);
        void __fastcall Timer2Timer(TObject *Sender);
        void __fastcall Timer3Timer(TObject *Sender);
        void __fastcall Timer4Timer(TObject *Sender);
        void __fastcall Timer5Timer(TObject *Sender);
        void __fastcall Timer6Timer(TObject *Sender);
        void __fastcall Timer7Timer(TObject *Sender);
        void __fastcall Timer8Timer(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm2(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm2 *Form2;
//---------------------------------------------------------------------
```

Кафедра_КбПЗ_2023рік

```
#endif
```

Кафедра_КбПЗ_2023рік

Файл write1.cpp – діалогове вікно запису у регістр THR

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "write1.h"
#include "main.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm6 *Form6;
//---------------------------------------------------------------------
__fastcall TForm6::TForm6(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm6::Button1Click(TObject *Sender)
{
if(CheckBox1->State==cbChecked) Form1->THR0->Text=1; else Form1->THR0->Text=0;
if(CheckBox2->State==cbChecked) Form1->THR1->Text=1; else Form1->THR1->Text=0;
if(CheckBox3->State==cbChecked) Form1->THR2->Text=1; else Form1->THR2->Text=0;
if(CheckBox4->State==cbChecked) Form1->THR3->Text=1; else Form1->THR3->Text=0;
if(CheckBox5->State==cbChecked) Form1->THR4->Text=1; else Form1->THR4->Text=0;
if(CheckBox6->State==cbChecked) Form1->THR5->Text=1; else Form1->THR5->Text=0;
if(CheckBox7->State==cbChecked) Form1->THR6->Text=1; else Form1->THR6->Text=0;
if(CheckBox8->State==cbChecked) Form1->THR7->Text=1; else Form1->THR7->Text=0;
Form6->Close();
}
//---------------------------------------------------------------------
```

Кафедра_КБПЗ_2023рік

Файл write1.h – бібліотека для файлу write1.cpp

```
//---------------------------------------------------------------------

#ifndef write1H
#define write1H
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------
class TForm6 : public TForm
{
__published:      // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TCheckBox *CheckBox2;
        TCheckBox *CheckBox1;
        TCheckBox *CheckBox3;
        TCheckBox *CheckBox4;
        TCheckBox *CheckBox5;
        TCheckBox *CheckBox6;
        TCheckBox *CheckBox7;
        TCheckBox *CheckBox8;
        TLabel *Label9;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private:  // User declarations
public:     // User declarations
        __fastcall TForm6(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm6 *Form6;
//---------------------------------------------------------------------
#endif
```

Кафедра_КбПЗ_2023рік

Файл write2.cpp – діалогове вікно запису у регістр RBR

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "write2.h"
#include "main.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm7 *Form7;
//---------------------------------------------------------------------
__fastcall TForm7::TForm7(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------

void __fastcall TForm7::Button1Click(TObject *Sender)
{
if(CheckBox1->State==cbChecked) Form1->RBR0->Text=1; else Form1->RBR0->Text=0;
if(CheckBox2->State==cbChecked) Form1->RBR1->Text=1; else Form1->RBR1->Text=0;
if(CheckBox3->State==cbChecked) Form1->RBR2->Text=1; else Form1->RBR2->Text=0;
if(CheckBox4->State==cbChecked) Form1->RBR3->Text=1; else Form1->RBR3->Text=0;
if(CheckBox5->State==cbChecked) Form1->RBR4->Text=1; else Form1->RBR4->Text=0;
if(CheckBox6->State==cbChecked) Form1->RBR5->Text=1; else Form1->RBR5->Text=0;
if(CheckBox7->State==cbChecked) Form1->RBR6->Text=1; else Form1->RBR6->Text=0;
if(CheckBox8->State==cbChecked) Form1->RBR7->Text=1; else Form1->RBR7->Text=0;
Form7->Close();
}
//---------------------------------------------------------------------
```

Кафедра_КБПЗ_2023рік

Файл write2.h – бібліотека для файлу write2.cpp

```
//---------------------------------------------------------------------

#ifndef write2H
#define write2H
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------
class TForm7 : public TForm
{
__published:      // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TCheckBox *CheckBox2;
        TCheckBox *CheckBox1;
        TCheckBox *CheckBox3;
        TCheckBox *CheckBox4;
        TCheckBox *CheckBox5;
        TCheckBox *CheckBox6;
        TCheckBox *CheckBox7;
        TCheckBox *CheckBox8;
        TLabel *Label9;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private: // User declarations
public:     // User declarations
        __fastcall TForm7(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm7 *Form7;
//---------------------------------------------------------------------
#endif
```

Кафедра_КбПЗ_2023рік

Файл write3.cpp – діалогове вікно запису у регістр LSB

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "write3.h"
#include "main.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm8 *Form8;
//---------------------------------------------------------------------
__fastcall TForm8::TForm8(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm8::Button1Click(TObject *Sender)
{
if(CheckBox1->State==cbChecked) Form1->LSB0->Text=1; else Form1->LSB0->Text=0;
if(CheckBox2->State==cbChecked) Form1->LSB1->Text=1; else Form1->LSB1->Text=0;
if(CheckBox3->State==cbChecked) Form1->LSB2->Text=1; else Form1->LSB2->Text=0;
if(CheckBox4->State==cbChecked) Form1->LSB3->Text=1; else Form1->LSB3->Text=0;
if(CheckBox5->State==cbChecked) Form1->LSB4->Text=1; else Form1->LSB4->Text=0;
if(CheckBox6->State==cbChecked) Form1->LSB5->Text=1; else Form1->LSB5->Text=0;
if(CheckBox7->State==cbChecked) Form1->LSB6->Text=1; else Form1->LSB6->Text=0;
if(CheckBox8->State==cbChecked) Form1->LSB7->Text=1; else Form1->LSB7->Text=0;
Form8->Close();
}
//---------------------------------------------------------------------
```

Кафедра_КБПЗ_2023рік

Файл write3.h – бібліотека для файлу write3.cpp

```
//---------------------------------------------------------------

#ifndef write3H
#define write3H
//---------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------
class TForm8 : public TForm
{
__published:      // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TCheckBox *CheckBox2;
        TCheckBox *CheckBox1;
        TCheckBox *CheckBox3;
        TCheckBox *CheckBox4;
        TCheckBox *CheckBox5;
        TCheckBox *CheckBox6;
        TCheckBox *CheckBox7;
        TCheckBox *CheckBox8;
        TLabel *Label9;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm8(TComponent* Owner);
};
//---------------------------------------------------------------
extern PACKAGE TForm8 *Form8;
//---------------------------------------------------------------
#endif
```

Кафедра_КбПЗ_2023рік

Файл write4.cpp – діалогове вікно запису у регістр MSB

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "write4.h"
#include "main.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm9 *Form9;
//---------------------------------------------------------------------
__fastcall TForm9::TForm9(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm9::Button1Click(TObject *Sender)
{
if(CheckBox1->State==cbChecked) Form1->MSB0->Text=1; else Form1->MSB0->Text=0;
if(CheckBox2->State==cbChecked) Form1->MSB1->Text=1; else Form1->MSB1->Text=0;
if(CheckBox3->State==cbChecked) Form1->MSB2->Text=1; else Form1->MSB2->Text=0;
if(CheckBox4->State==cbChecked) Form1->MSB3->Text=1; else Form1->MSB3->Text=0;
if(CheckBox5->State==cbChecked) Form1->MSB4->Text=1; else Form1->MSB4->Text=0;
if(CheckBox6->State==cbChecked) Form1->MSB5->Text=1; else Form1->MSB5->Text=0;
if(CheckBox7->State==cbChecked) Form1->MSB6->Text=1; else Form1->MSB6->Text=0;
if(CheckBox8->State==cbChecked) Form1->MSB7->Text=1; else Form1->MSB7->Text=0;
Form9->Close();

}
//---------------------------------------------------------------------
```

Кафедра_КБПЗ_2023рік

Файл write4.h – бібліотека для файлу write4.cpp

```
//---------------------------------------------------------------------

#ifndef write4H
#define write4H
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------
class TForm9 : public TForm
{
__published:      // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TCheckBox *CheckBox2;
        TCheckBox *CheckBox1;
        TCheckBox *CheckBox3;
        TCheckBox *CheckBox4;
        TCheckBox *CheckBox5;
        TCheckBox *CheckBox6;
        TCheckBox *CheckBox7;
        TCheckBox *CheckBox8;
        TLabel *Label9;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private: // User declarations
public:     // User declarations
        __fastcall TForm9(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm9 *Form9;
//---------------------------------------------------------------------
#endif
```

Кафедра_КбПЗ_2023рік

Файл write5.cpp – діалогове вікно запису у регістр IER

```
//---------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "write5.h"
#include "main.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm10 *Form10;
//---------------------------------------------------------------------
__fastcall TForm10::TForm10(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm10::Button1Click(TObject *Sender)
{
if(CheckBox1->State==cbChecked) Form1->IER0->Text=1; else Form1->IER0->Text=0;
if(CheckBox2->State==cbChecked) Form1->IER1->Text=1; else Form1->IER1->Text=0;
if(CheckBox3->State==cbChecked) Form1->IER2->Text=1; else Form1->IER2->Text=0;
if(CheckBox4->State==cbChecked) Form1->IER3->Text=1; else Form1->IER3->Text=0;
if(CheckBox5->State==cbChecked) Form1->IER4->Text=1; else Form1->IER4->Text=0;
if(CheckBox6->State==cbChecked) Form1->IER5->Text=1; else Form1->IER5->Text=0;
if(CheckBox7->State==cbChecked) Form1->IER6->Text=1; else Form1->IER6->Text=0;
if(CheckBox8->State==cbChecked) Form1->IER7->Text=1; else Form1->IER7->Text=0;
Form10->Close();
}
//---------------------------------------------------------------------
```

Кафедра_КБПЗ_2023рік

Файл write5.h – бібліотека для файлу write5.cpp

```
//---------------------------------------------------------------------

#ifndef write5H
#define write5H
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------
class TForm10 : public TForm
{
__published:      // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TCheckBox *CheckBox2;
        TCheckBox *CheckBox1;
        TCheckBox *CheckBox3;
        TCheckBox *CheckBox4;
        TCheckBox *CheckBox5;
        TCheckBox *CheckBox6;
        TCheckBox *CheckBox7;
        TCheckBox *CheckBox8;
        TLabel *Label9;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private:// User declarations
public:     // User declarations
        __fastcall TForm10(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm10 *Form10;
//---------------------------------------------------------------------
#endif
```

Кафедра_КбПЗ_2023рік

Файл write6.cpp – діалогове вікно запису у регістр IIR

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "write6.h"
#include "main.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm11 *Form11;
//---------------------------------------------------------------------
__fastcall TForm11::TForm11(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm11::Button1Click(TObject *Sender)
{
if(CheckBox1->State==cbChecked) Form1->IIR0->Text=1; else Form1->IIR0->Text=0;
if(CheckBox2->State==cbChecked) Form1->IIR1->Text=1; else Form1->IIR1->Text=0;
if(CheckBox3->State==cbChecked) Form1->IIR2->Text=1; else Form1->IIR2->Text=0;
if(CheckBox4->State==cbChecked) Form1->IIR3->Text=1; else Form1->IIR3->Text=0;
if(CheckBox5->State==cbChecked) Form1->IIR4->Text=1; else Form1->IIR4->Text=0;
if(CheckBox6->State==cbChecked) Form1->IIR5->Text=1; else Form1->IIR5->Text=0;
if(CheckBox7->State==cbChecked) Form1->IIR6->Text=1; else Form1->IIR6->Text=0;
if(CheckBox8->State==cbChecked) Form1->IIR7->Text=1; else Form1->IIR7->Text=0;
Form11->Close();

}
//---------------------------------------------------------------------
```

Кафедра_КБПЗ_2023рік

Файл write6.h – бібліотека для файлу write6.cpp

```
//---------------------------------------------------------------------

#ifndef write6H
#define write6H
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------
class TForm11 : public TForm
{
__published:      // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TCheckBox *CheckBox2;
        TCheckBox *CheckBox1;
        TCheckBox *CheckBox3;
        TCheckBox *CheckBox4;
        TCheckBox *CheckBox5;
        TCheckBox *CheckBox6;
        TCheckBox *CheckBox7;
        TCheckBox *CheckBox8;
        TLabel *Label9;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private:  // User declarations
public:     // User declarations
        __fastcall TForm11(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm11 *Form11;
//---------------------------------------------------------------------
#endif
```

Кафедра_КбПЗ_2023рік

Файл write7.cpp – діалогове вікно запису у регістр LCR

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "write7.h"
#include "main.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm12 *Form12;
//---------------------------------------------------------------------
__fastcall TForm12::TForm12(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm12::Button1Click(TObject *Sender)
{
if(CheckBox1->State==cbChecked) Form1->LCR0->Text=1; else Form1->LCR0->Text=0;
if(CheckBox2->State==cbChecked) Form1->LCR1->Text=1; else Form1->LCR1->Text=0;
if(CheckBox3->State==cbChecked) Form1->LCR2->Text=1; else Form1->LCR2->Text=0;
if(CheckBox4->State==cbChecked) Form1->LCR3->Text=1; else Form1->LCR3->Text=0;
if(CheckBox5->State==cbChecked) Form1->LCR4->Text=1; else Form1->LCR4->Text=0;
if(CheckBox6->State==cbChecked) Form1->LCR5->Text=1; else Form1->LCR5->Text=0;
if(CheckBox7->State==cbChecked) Form1->LCR6->Text=1; else Form1->LCR6->Text=0;
if(CheckBox8->State==cbChecked) Form1->LCR7->Text=1; else Form1->LCR7->Text=0;
Form12->Close();

}
//---------------------------------------------------------------------
```

Кафедра_КБПЗ_2023рік

Файл write7.h – бібліотека для файлу write7.cpp

```
//---------------------------------------------------------------------

#ifndef write7H
#define write7H
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------
class TForm12 : public TForm
{
__published: // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TCheckBox *CheckBox2;
        TCheckBox *CheckBox1;
        TCheckBox *CheckBox3;
        TCheckBox *CheckBox4;
        TCheckBox *CheckBox5;
        TCheckBox *CheckBox6;
        TCheckBox *CheckBox7;
        TCheckBox *CheckBox8;
        TLabel *Label9;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private:     // User declarations
public:           // User declarations
        __fastcall TForm12(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm12 *Form12;
//---------------------------------------------------------------------
#endif
```

Кафедра_КбПЗ_2023рік

Файл write8.cpp – діалогове вікно запису у регістр MCR

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "write8.h"
#include "main.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm13 *Form13;
//---------------------------------------------------------------------
__fastcall TForm13::TForm13(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm13::Button1Click(TObject *Sender)
{
if(CheckBox1->State==cbChecked) Form1->MCR0->Text=1; else Form1->MCR0->Text=0;
if(CheckBox2->State==cbChecked) Form1->MCR1->Text=1; else Form1->MCR1->Text=0;
if(CheckBox3->State==cbChecked) Form1->MCR2->Text=1; else Form1->MCR2->Text=0;
if(CheckBox4->State==cbChecked) Form1->MCR3->Text=1; else Form1->MCR3->Text=0;
if(CheckBox5->State==cbChecked) Form1->MCR4->Text=1; else Form1->MCR4->Text=0;
if(CheckBox6->State==cbChecked) Form1->MCR5->Text=1; else Form1->MCR5->Text=0;
if(CheckBox7->State==cbChecked) Form1->MCR6->Text=1; else Form1->MCR6->Text=0;
if(CheckBox8->State==cbChecked) Form1->MCR7->Text=1; else Form1->MCR7->Text=0;
Form13->Close();

}
//---------------------------------------------------------------------
```

Кафедра_КБПЗ_2023рік

Файл write8.h – бібліотека для файлу write8.cpp

```
//---------------------------------------------------------------------

#ifndef write8H
#define write8H
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------
class TForm13 : public TForm
{
__published:      // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TCheckBox *CheckBox2;
        TCheckBox *CheckBox1;
        TCheckBox *CheckBox3;
        TCheckBox *CheckBox4;
        TCheckBox *CheckBox5;
        TCheckBox *CheckBox6;
        TCheckBox *CheckBox7;
        TCheckBox *CheckBox8;
        TLabel *Label9;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm13(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm13 *Form13;
//---------------------------------------------------------------------
#endif
```

Кафедра_КбПЗ_2023рік

Файл write19.cpp – діалогове вікно запису у регістр LSR

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "write9.h"
#include "main.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm14 *Form14;
//---------------------------------------------------------------------
__fastcall TForm14::TForm14(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm14::Button1Click(TObject *Sender)
{
if(CheckBox1->State==cbChecked) Form1->LSR0->Text=1; else Form1->LSR0->Text=0;
if(CheckBox2->State==cbChecked) Form1->LSR1->Text=1; else Form1->LSR1->Text=0;
if(CheckBox3->State==cbChecked) Form1->LSR2->Text=1; else Form1->LSR2->Text=0;
if(CheckBox4->State==cbChecked) Form1->LSR3->Text=1; else Form1->LSR3->Text=0;
if(CheckBox5->State==cbChecked) Form1->LSR4->Text=1; else Form1->LSR4->Text=0;
if(CheckBox6->State==cbChecked) Form1->LSR5->Text=1; else Form1->LSR5->Text=0;
if(CheckBox7->State==cbChecked) Form1->LSR6->Text=1; else Form1->LSR6->Text=0;
if(CheckBox8->State==cbChecked) Form1->LSR7->Text=1; else Form1->LSR7->Text=0;
Form14->Close();

}
//---------------------------------------------------------------------
```

Кафедра_КОПЗ_2023рік

Файл write9.h – бібліотека для файлу write9.cpp

```
//---------------------------------------------------------------------

#ifndef write9H
#define write9H
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------
class TForm14 : public TForm
{
__published:      // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TCheckBox *CheckBox2;
        TCheckBox *CheckBox1;
        TCheckBox *CheckBox3;
        TCheckBox *CheckBox4;
        TCheckBox *CheckBox5;
        TCheckBox *CheckBox6;
        TCheckBox *CheckBox7;
        TCheckBox *CheckBox8;
        TLabel *Label9;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private:   // User declarations
public:     // User declarations
        __fastcall TForm14(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm14 *Form14;
//---------------------------------------------------------------------
#endif
```

Кафедра_КбПЗ_2023рік

Файл write110.cpp – діалогове вікно запису у регістр MSR

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "write10.h"
#include "main.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm15 *Form15;
//---------------------------------------------------------------------
__fastcall TForm15::TForm15(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm15::Button1Click(TObject *Sender)
{
if(CheckBox1->State==cbChecked) Form1->MSR0->Text=1; else Form1->MSR0->Text=0;
if(CheckBox2->State==cbChecked) Form1->MSR11->Text=1; else Form1->MSR11->Text=0;
if(CheckBox3->State==cbChecked) Form1->MSR2->Text=1; else Form1->MSR2->Text=0;
if(CheckBox4->State==cbChecked) Form1->MSR3->Text=1; else Form1->MSR3->Text=0;
if(CheckBox5->State==cbChecked) Form1->MSR4->Text=1; else Form1->MSR4->Text=0;
if(CheckBox6->State==cbChecked) Form1->MSR5->Text=1; else Form1->MSR5->Text=0;
if(CheckBox7->State==cbChecked) Form1->MSR6->Text=1; else Form1->MSR6->Text=0;
if(CheckBox8->State==cbChecked) Form1->MSR7->Text=1; else Form1->MSR7->Text=0;
Form15->Close();

}
//---------------------------------------------------------------------
```

Кафедра_КБПЗ_2023рік

Файл write10.h – бібліотека для файлу write10.cpp

```
//---------------------------------------------------------------------

#ifndef write10H
#define write10H
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------
class TForm15 : public TForm
{
__published:      // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TCheckBox *CheckBox2;
        TCheckBox *CheckBox1;
        TCheckBox *CheckBox3;
        TCheckBox *CheckBox4;
        TCheckBox *CheckBox5;
        TCheckBox *CheckBox6;
        TCheckBox *CheckBox7;
        TCheckBox *CheckBox8;
        TLabel *Label9;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm15(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm15 *Form15;
//---------------------------------------------------------------------
#endif
```

Кафедра_КбПЗ_2023рік

Файл help.cpp – інтерактивна довідка про регістри послідовного порту

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "help.h"
//---------------------------------------------------------------------#pragma
package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
int reg=1;
//---------------------------------------------------------------------
__fastcall TForm4::TForm4(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm4::Button1Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Регістр буфера передавача (THR). Має адресу 0 щодо базової
адреси контролера. Даний регістр доступний тільки під час запису й при значенні
біта дозволу доступу до дільника (DLAB) у регістрі керування лінією (LCR),
рівному 0. Регістр THR містить вісім бітів даних (біт 0 є молодшим значущим
розрядом і посилається першим у канал передачі).");
Button1->Font->Size=14;
Button2->Font->Size=8;
Button3->Font->Size=8;
Button4->Font->Size=8;
Button5->Font->Size=8;
Button6->Font->Size=8;
Button7->Font->Size=8;
Button8->Font->Size=8;
Button9->Font->Size=8;
Button10->Font->Size=8;
reg=1;
}
//---------------------------------------------------------------------

void __fastcall TForm4::Button2Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Регістр буфера приймача (RBR). Має адресу 0 щодо базової
адреси контролера. Цей регістр доступний під час читання (IN) і при значенні
біта дозволу доступу до дільника (DLAB) у регістрі керування лінією (LCR),
рівному 0. Регістр RBR містить вісім бітів даних (біт 0 є молодшим значущим
розрядом і приймається першим з каналу передачі).");

Button1->Font->Size=8;
Button2->Font->Size=14;
Button3->Font->Size=8;
Button4->Font->Size=8;
Button5->Font->Size=8;
Button6->Font->Size=8;
Button7->Font->Size=8;
Button8->Font->Size=8;
Button9->Font->Size=8;
Button10->Font->Size=8;
reg=2;
}
//---------------------------------------------------------------------

void __fastcall TForm4::Button3Click(TObject *Sender)
{
```

Кафедра_КбПЗ_2023рік

```
Memo1->Clear();
Memo1->Lines->Add("Регістр буфера молодшого байта дільника (Divisor Latch LSB). Регістр має адресу 0 щодо базової адреси контролера. Цей регістр доступний і під час читання, і під час запису тільки при значенні біта дозволу доступу до дільника (DLAB) у регістрі керування лінією (LCR), рівному 1. При записі в цей регістр нового значення дільник перезавантажується негайно.");
Memo1->Lines->Add("Дільник     Швидкість передачі в бодах");
Memo1->Lines->Add("1040        110");
Memo1->Lines->Add("768          150");
Memo1->Lines->Add("384          300 ");
Memo1->Lines->Add("192          600 ");
Memo1->Lines->Add("96           1200 ");
Memo1->Lines->Add("48           2400 ");
Memo1->Lines->Add("24           4800 ");
Memo1->Lines->Add("12           9600  ");
Memo1->Lines->Add("6            19200 ");
Memo1->Lines->Add("3            38400");
Memo1->Lines->Add("2            57600  ");
Memo1->Lines->Add("1            115200  ");
Button1->Font->Size=8;
Button2->Font->Size=8;
Button3->Font->Size=14;
Button4->Font->Size=8;
Button5->Font->Size=8;
Button6->Font->Size=8;
Button7->Font->Size=8;
Button8->Font->Size=8;
Button9->Font->Size=8;
Button10->Font->Size=8;
reg=3;
}
//---------------------------------------------------------------
void __fastcall TForm4::Button4Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Регістр буфера старшого байта дільника (Divisor Latch MSB). Регістр має адресу 1 щодо базової адреси контролера. Цей регістр доступний по читанню й запису тільки при значенні біта дозволу доступу до дільника (DLAB) у регістрі керування лінією (LCR), рівному 1. При записі в цей регістр нового значення дільник відразу перезавантажується.");
Button1->Font->Size=8;
Button2->Font->Size=8;
Button3->Font->Size=8;
Button4->Font->Size=14;
Button5->Font->Size=8;
Button6->Font->Size=8;
Button7->Font->Size=8;
Button8->Font->Size=8;
Button9->Font->Size=8;
Button10->Font->Size=8;
reg=4;
}
//---------------------------------------------------------------
void __fastcall TForm4::Button5Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Регістр дозволу переривань (IER). Має адресу 1 щодо базової адреси контролера. Цей регістр доступний по читанню й запису, але тільки при значенні біта дозволу доступу до дільника (DLAB) у регістрі керування лінією (LCR), рівному 0. Цей регістр дозволяє управляти чотирма типами переривань, породжуваними контролером послідовного інтерфейсу. Формат регістра:");
Memo1->Lines->Add("Біт 0 - IDA ");
Memo1->Lines->Add("Біт 1 - IFB ");
Memo1->Lines->Add("Біт 2 - ICL ");
Memo1->Lines->Add("Біт 3 - ICM ");
Memo1->Lines->Add("Біт 4 - зарезервований, =0");
Memo1->Lines->Add("Біт 5 - зарезервований, =0");
Memo1->Lines->Add("Біт 6 - зарезервований, =0");
Memo1->Lines->Add("Біт 7 - зарезервований, =0");
```

Кафедра_КБПЗ_2023рік

```
Button1->Font->Size=8;
Button2->Font->Size=8;
Button3->Font->Size=8;
Button4->Font->Size=8;
Button5->Font->Size=14;
Button6->Font->Size=8;
Button7->Font->Size=8;
Button8->Font->Size=8;
Button9->Font->Size=8;
Button10->Font->Size=8;
reg=5;
}
//---------------------------------------------------------------------

void __fastcall TForm4::Button6Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Регістр ідентифікації переривання (IIR). Регістр має адресу 2
щодо базової адреси контролера. Цей регістр доступний тільки по читанню й
дозволяє одержати інформацію від контролера про очікуюче переривання. Значення
бітів регістра:");
Memo1->Lines->Add("Біт 0 - II ");
Memo1->Lines->Add("Біт 1 - I Type ");
Memo1->Lines->Add("Біт 2 - I Type ");
Memo1->Lines->Add("Біт 3 - зарезервований, =0 ");
Memo1->Lines->Add("Біт 4 - зарезервований, =0");
Memo1->Lines->Add("Біт 5 - зарезервований, =0");
Memo1->Lines->Add("Біт 6 - зарезервований, =0");
Memo1->Lines->Add("Біт 7 - зарезервований, =0");
Button1->Font->Size=8;
Button2->Font->Size=8;
Button3->Font->Size=8;
Button4->Font->Size=8;
Button5->Font->Size=8;
Button6->Font->Size=14;
Button7->Font->Size=8;
Button8->Font->Size=8;
Button9->Font->Size=8;
Button10->Font->Size=8;
reg=6;
}
//---------------------------------------------------------------------

void __fastcall TForm4::Button7Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Регістр керування лінією (LCR). Регістр має адресу 3 щодо
базової адреси контролера. Цей регістр доступний і під час читання, і під час
запису.Значення даного регістра визначає формат переданих даних у лінію передачі
даних контролером послідовного інтерфейсу. Значення бітів регістра");
Memo1->Lines->Add("Біт 0 - WLS ");
Memo1->Lines->Add("Біт 1 - WLS ");
Memo1->Lines->Add("Біт 2 - NSB ");
Memo1->Lines->Add("Біт 3 - PA ");
Memo1->Lines->Add("Біт 4 - EPS ");
Memo1->Lines->Add("Біт 5 - SP ");
Memo1->Lines->Add("Біт 6 - SB ");
Memo1->Lines->Add("Біт 7 - DLAB ");
Button1->Font->Size=8;
Button2->Font->Size=8;
Button3->Font->Size=8;
Button4->Font->Size=8;
Button5->Font->Size=8;
Button6->Font->Size=8;
Button7->Font->Size=14;
Button8->Font->Size=8;
Button9->Font->Size=8;
Button10->Font->Size=8;
reg=7;
```

Кафедра_КбПЗ_2023рік

```
}
//---------------------------------------------------------------------

void __fastcall TForm4::Button8Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Регістр керування модемом (MCR). Регістр керування модемом
має адресу 4 щодо базової адреси контролера. Цей регістр доступний по читанню й
запису. За допомогою регістра можна управляти роботою модему. Формат
регістра:");
Memo1->Lines->Add("Біт 0 - DTR ");
Memo1->Lines->Add("Біт 1 - RTS ");
Memo1->Lines->Add("Біт 2 - Out1 ");
Memo1->Lines->Add("Біт 3 - Out2 ");
Memo1->Lines->Add("Біт 4 - LB ");
Memo1->Lines->Add("Біт 5 - зарезервований, =0 ");
Memo1->Lines->Add("Біт 6 - зарезервований, =0 ");
Memo1->Lines->Add("Біт 7 - зарезервований, =0 ");
Button1->Font->Size=8;
Button2->Font->Size=8;
Button3->Font->Size=8;
Button4->Font->Size=8;
Button5->Font->Size=8;
Button6->Font->Size=8;
Button7->Font->Size=8;
Button8->Font->Size=14;
Button9->Font->Size=8;
Button10->Font->Size=8;
reg=8;
}
//---------------------------------------------------------------------

void __fastcall TForm4::Button9Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Регістр стану лінії (LSR). Регістр стану лінії має адресу 5
щодо базової адреси контролера й доступний тільки по читанню. Регістр LSR надає
інформацію про стан обміну даних. Формат регістра:");
Memo1->Lines->Add("Біт 0 - DR ");
Memo1->Lines->Add("Біт 1 - OR ");
Memo1->Lines->Add("Біт 2 - PE ");
Memo1->Lines->Add("Біт 3 - FE ");
Memo1->Lines->Add("Біт 4 - BI ");
Memo1->Lines->Add("Біт 5 - THRE ");
Memo1->Lines->Add("Біт 6 - TEMT ");
Memo1->Lines->Add("Біт 7 - зарезервований, =0 ");
Button1->Font->Size=8;
Button2->Font->Size=8;
Button3->Font->Size=8;
Button4->Font->Size=8;
Button5->Font->Size=8;
Button6->Font->Size=8;
Button7->Font->Size=8;
Button8->Font->Size=8;
Button9->Font->Size=14;
Button10->Font->Size=8;
reg=9;
}
//---------------------------------------------------------------------

void __fastcall TForm4::Button10Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Регістр стану модему (MSR). Регістр має адресу 6 щодо базової
адреси контролера й доступний тільки по читанню. Регістр надає інформацію про
стан керуючих ліній модему. Крім того, цей регістр містить 4 біти, які
відображають зміну стану модему й встановлюються в значення 0 після операції
читання з регістра MSR.");
Memo1->Lines->Add("Біт 0 - DCTS ");
```

Кафедра КБПЗ_2023рік

```
Memo1->Lines->Add("Біт 1 - DDSR ");
Memo1->Lines->Add("Біт 2 - TERI ");
Memo1->Lines->Add("Біт 3 - DDCD ");
Memo1->Lines->Add("Біт 4 - CTS ");
Memo1->Lines->Add("Біт 5 - DSR ");
Memo1->Lines->Add("Біт 6 - RI ");
Memo1->Lines->Add("Біт 7 - DCD ");
Button1->Font->Size=8;
Button2->Font->Size=8;
Button3->Font->Size=8;
Button4->Font->Size=8;
Button5->Font->Size=8;
Button6->Font->Size=8;
Button7->Font->Size=8;
Button8->Font->Size=8;
Button9->Font->Size=8;
Button10->Font->Size=14;
reg=10;
}
//---------------------------------------------------------------------

void __fastcall TForm4::Button11Click(TObject *Sender)
{

Button11->Font->Size=14;
Button12->Font->Size=8;
Button13->Font->Size=8;
Button14->Font->Size=8;
Button15->Font->Size=8;
Button16->Font->Size=8;
Button17->Font->Size=8;
Button18->Font->Size=8;
if(reg==1){
Memo2->Clear();
Memo2->Lines->Add("біт даних для передачі");
}
else if(reg==2){
Memo2->Clear();
Memo2->Lines->Add("прийнятий біт даних");
}
else if(reg==3){
Memo2->Clear();
Memo2->Lines->Add("один з бітів дільника");
}
else if(reg==4){
Memo2->Clear();
Memo2->Lines->Add("один з бітів дільника");
}
else if(reg==5){
Memo2->Clear();
Memo2->Lines->Add("Зарезервований, =0");
}
else if(reg==6){
Memo2->Clear();
Memo2->Lines->Add("Зарезервований, =0");
}
else if(reg==7){
Memo2->Clear();
Memo2->Lines->Add("DLAB управляє доступом до регістрів буфера дільника. Якщо біт
дорівнює 1, операція читання й запису по адресі  1 щодо базової адреси
виконуються з регістрами буфера дільника програмувального генератора. Для
доступу до регістрів RBR, THR і IER біт повинен мати нульове значення.");
}
else if(reg==8){
Memo2->Clear();
Memo2->Lines->Add("Зарезервований, =0");
}
else if(reg==9){
Memo2->Clear();
```

Кафедра_КБПЗ_2023рік

```
Memo2->Lines->Add("Зарезервований, =0");
}
else if(reg==10){
Memo2->Clear();
Memo2->Lines->Add("DCD є інвертованим сигналом Data Carrier Detect (DCD). При
встановленому режимі 'шлейфа' (біт LB регістра MCR має значення 1) цей біт
еквівалентний біту Out2 регістри MCR.");
}

}
//---------------------------------------------------------------------
void __fastcall TForm4::Button12Click(TObject *Sender)
{
Button11->Font->Size=8;
Button12->Font->Size=14;
Button13->Font->Size=8;
Button14->Font->Size=8;
Button15->Font->Size=8;
Button16->Font->Size=8;
Button17->Font->Size=8;
Button18->Font->Size=8;
if(reg==1){
Memo2->Clear();
Memo2->Lines->Add("біт даних для передачі");
}
else if(reg==2){
Memo2->Clear();
Memo2->Lines->Add("прийнятий біт даних");
}
else if(reg==3){
Memo2->Clear();
Memo2->Lines->Add("один з бітів дільника");
}
else if(reg==4){
Memo2->Clear();
Memo2->Lines->Add("один з бітів дільника");
}
else if(reg==5){
Memo2->Clear();
Memo2->Lines->Add("Зарезервований, =0");
}
else if(reg==6){
Memo2->Clear();
Memo2->Lines->Add("Зарезервований, =0");
}
else if(reg==7){
Memo2->Clear();
Memo2->Lines->Add(" SB встановлює стан 'пауза', коли дорівнює 1. У цьому стані
на виході контролера послідовного інтерфейсу встановлюється значення 0, що не
може бути змінене ніякими іншими діями, крім як переустановкою біта в 0.");
}
else if(reg==8){
Memo2->Clear();
Memo2->Lines->Add("Зарезервований, =0");
}
else if(reg==9){
Memo2->Clear();
Memo2->Lines->Add("TEMT являється індикатором звільнення передавача. Встановка
цього біта в 1 означає, що як регістр THR, так і регістр TSR вільний. Цей біт
встановлюється в значення 0, якщо кожний з регістрів THR і TSR містить
символ.");
}
else if(reg==10){
Memo2->Clear();
Memo2->Lines->Add("RI є інвертованим сигналом Ring Indicator (RI). При
встановленому режимі 'шлейфа' (біт LB регістра MCR має значення 1) еквівалентний
біту Out1 регістра MCR.");
}
}
```

Кафедра_КбПЗ_2023рік

```
//---------------------------------------------------------------------

void __fastcall TForm4::Button13Click(TObject *Sender)
{
Button11->Font->Size=8;
Button12->Font->Size=8;
Button13->Font->Size=14;
Button14->Font->Size=8;
Button15->Font->Size=8;
Button16->Font->Size=8;
Button17->Font->Size=8;
Button18->Font->Size=8;
if(reg==1){
Memo2->Clear();
Memo2->Lines->Add("біт даних для передачі");
}
else if(reg==2){
Memo2->Clear();
Memo2->Lines->Add("прийнятий біт даних");
}
else if(reg==3){
Memo2->Clear();
Memo2->Lines->Add("один з бітів дільника");
}
else if(reg==4){
Memo2->Clear();
Memo2->Lines->Add("один з бітів дільника");
}
else if(reg==5){
Memo2->Clear();
Memo2->Lines->Add("Зарезервований, =0");
}
else if(reg==6){
Memo2->Clear();
Memo2->Lines->Add("Зарезервований, =0");
}
else if(reg==7){
Memo2->Clear();
Memo2->Lines->Add(" SP управляє установкою режиму незмінного біта контролю
парності. Значення біта 1 задає режим, а значення 0 - скасовує. При встановленні
біта SP в 1 повинен встановлюватися в 1 і біт PA, так як ці два біти зв'язані.
Коли значення біта EPS дорівнює 0, посилається й контролюється значення біта
контролю парності, рівне 1 (Mark Parity). При одиничному значенні біта EPS
посилається й контролюється значення біта контролю парності, рівне 0 (Space
Parity).");
}
else if(reg==8){
Memo2->Clear();
Memo2->Lines->Add("Зарезервований, =0");
}
else if(reg==9){
Memo2->Clear();
Memo2->Lines->Add("THRE є індикатором звільнення регістра THR. Установка цього
біта в 1 означає, що з регістра THR символ переданий у зсувовий регістр
передавача (TSR) і регістра THR готовий прийняти наступний байт. Якщо в регістрі
IER дозволене переривання по звільненню регістра THR, то при установці цього
біта в значенні 1 відбувається також переривання по звільненню регістра THR.");
}
else if(reg==10){
Memo2->Clear();
Memo2->Lines->Add("DSR є інвертованим сигналом Data Set Ready (DSR). У режимі
'шлейфа' (біт LB регістра MCR має значення 1) еквівалентний біту DTR регістра
MCR.");
}
}
//---------------------------------------------------------------------

void __fastcall TForm4::Button14Click(TObject *Sender)
{
```

Кафедра КБПЗ_2023рік

```
Button11->Font->Size=8;
Button12->Font->Size=8;
Button13->Font->Size=8;
Button14->Font->Size=14;
Button15->Font->Size=8;
Button16->Font->Size=8;
Button17->Font->Size=8;
Button18->Font->Size=8;
if(reg==1){
Memo2->Clear();
Memo2->Lines->Add("біт даних для передачі");
}
else if(reg==2){
Memo2->Clear();
Memo2->Lines->Add("прийнятий біт даних");
}
else if(reg==3){
Memo2->Clear();
Memo2->Lines->Add("один з бітів дільника");
}
else if(reg==4){
Memo2->Clear();
Memo2->Lines->Add("один з бітів дільника");
}
else if(reg==5){
Memo2->Clear();
Memo2->Lines->Add("Зарезервований, =0");
}
else if(reg==6){
Memo2->Clear();
Memo2->Lines->Add("Зарезервований, =0");
}
else if(reg==7){
Memo2->Clear();
Memo2->Lines->Add("EPS задає вибір режиму контролю парності. Якщо біт
встановлений в 0 і біт PA установлений в 1, генерується й перевіряється парна
кількість одиничних бітів символу посилки й біта контролю парності. Якщо біт
установлений в 1 і біт PA встановлений в 1, генерується й перевіряється непарна
кількість одиничних бітів символу посилки й біта контролю парності.");
}
else if(reg==8){
Memo2->Clear();
Memo2->Lines->Add("LB задає режим 'шлейфа' (Loopback) для діагностичних
цілей.");
}
else if(reg==9){
Memo2->Clear();
Memo2->Lines->Add("BI є індикатором стану 'пауза' (Break Interrupt). Стан
'пауза' фіксується в тому випадку, якщо  рівень прийнятого сигналу встановлений
в 0 на час прийому повної посилки, тобто загальний час стартового біта, бітів
даних, біта контролю парності й стоп-біта. Цей біт приймає значення 0 після
операції читання регістра LSR. Біти 4-1 є індикаторами помилки й установка
кожного із цих бітів у значення 1 проводить до породження переривання по стану
ліній приймача.");
}
else if(reg==10){
Memo2->Clear();
Memo2->Lines->Add("CTS - інвертований сигнал Clear to Send (CTS). При
встановленому режимі 'шлейфа' (біт LB регістра MCR має значення 1) цей біт
еквівалентний біту RTS регістра MCR. Біти  DDCD, TERI, DDSR і DCTS є
індикаторами зміни стану модему й установка кожного із цих бітів у значення 1
приводить до породження переривання по стану модему, якщо воно дозволено в
регістрі IER.");
}
}
//---------------------------------------------------------------------

void __fastcall TForm4::Button15Click(TObject *Sender)
{
```

Кафедра_КОПЗ_2023рік

```
Button11->Font->Size=8;
Button12->Font->Size=8;
Button13->Font->Size=8;
Button14->Font->Size=8;
Button15->Font->Size=14;
Button16->Font->Size=8;
Button17->Font->Size=8;
Button18->Font->Size=8;
if(reg==1){
Memo2->Clear();
Memo2->Lines->Add("біт даних для передачі");
}
else if(reg==2){
Memo2->Clear();
Memo2->Lines->Add("прийнятий біт даних");
}
else if(reg==3){
Memo2->Clear();
Memo2->Lines->Add("один з бітів дільника");
}
else if(reg==4){
Memo2->Clear();
Memo2->Lines->Add("один з бітів дільника");
}
else if(reg==5){
Memo2->Clear();
Memo2->Lines->Add("ICM  задає переривання при зміні стану модема:");
Memo2->Lines->Add("1 - переривання виробляється;");
Memo2->Lines->Add("0 - переривання заборонене.");
}
else if(reg==6){
Memo2->Clear();
Memo2->Lines->Add("Зарезервований, =0");
}
else if(reg==7){
Memo2->Clear();
Memo2->Lines->Add("PA є бітом дозволу контролю парності. Якщо біт установлений в
1, то генерується біт контролю парності між останнім бітом переданого символу й
стоп-бітом.");
}
else if(reg==8){
Memo2->Clear();
Memo2->Lines->Add("Out2 управляє сигналом Out2. При одиничному значенні біта
сигнал Out2 встановлюється рівним 1. Сигнал Out2  управляє генерацією переривань
контролера послідовного інтерфейсу. При одиничному знанні сигнал контролер
генерує переривання у відповідності  зі значенням регістра IER. При нульовому
значенні сигналу Out2 контролер не генерує переривань незалежно від значення
регістра IER.");
}
else if(reg==9){
Memo2->Clear();
Memo2->Lines->Add("FE є індикатором 'помилки стоп-бітів' (Framing Error).
Помилка стоп-бита фіксується в тому випадку, коли в прийнятому символі не
виявлено коректного стоп-біта, тобто біт, що слідує за останнім бітом даних або
за бітом контролю парності (у випадку контролю парності), має значення 0. Цей
біт приймає значення 0 після операції читання регістра LSR.");
}
else if(reg==10){
Memo2->Clear();
Memo2->Lines->Add("DDCD є індикатором зміни сигналу Data Carrier Detect (DCD).
Цей біт приймає значення 1 при зміні сигналу DCD після останньої операції
читання регістра MSR.");
}
}
//---------------------------------------------------------------------

void __fastcall TForm4::Button16Click(TObject *Sender)
{
Button11->Font->Size=8;
```

Кафедра_КБПЗ_2023рік

```
Button12->Font->Size=8;
Button13->Font->Size=8;
Button14->Font->Size=8;
Button15->Font->Size=8;
Button16->Font->Size=14;
Button17->Font->Size=8;
Button18->Font->Size=8;
if(reg==1){
Memo2->Clear();
Memo2->Lines->Add("біт даних для передачі");
}
else if(reg==2){
Memo2->Clear();
Memo2->Lines->Add("прийнятий біт даних");
}
else if(reg==3){
Memo2->Clear();
Memo2->Lines->Add("один з бітів дільника");
}
else if(reg==4){
Memo2->Clear();
Memo2->Lines->Add("один з бітів дільника");
}
else if(reg==5){
Memo2->Clear();
Memo2->Lines->Add("ICL визначає переривання при зміні стану лінії приймача:");
Memo2->Lines->Add("1 - переривання виробляється;");
Memo2->Lines->Add("0 - переривання заборонене.");

}
else if(reg==6){
Memo2->Clear();
Memo2->Lines->Add("1 біт I Type. Біти  I Type визначають тип очікуючого
переривання, якщо воно зберігається контролером (що визначається бітом II):");
Memo2->Lines->Add("11 - змінився стан лінії приймача;  ");
Memo2->Lines->Add("10 - прийняті дані доступні; ");
Memo2->Lines->Add("01 - звільнений регістр буфера;");
Memo2->Lines->Add("00 - змінився стан модему. ");

}
else if(reg==7){
Memo2->Clear();
Memo2->Lines->Add("NSB визначає кількість стоп-бітів у кожному символі,
переданому контролером послідовного інтерфейсу, і пов'язаний з довжиною слова
обміну (біти WLS). Якщо цей біт встановлений в 0, то генерується й перевіряється
один стоп-біт при будь-якій довжині слова обміну. Якщо цей біт встановлений в 1,
то при довжині слова обміну в 5 біт генерується й перевіряється 1.5 стоп-біта, а
при будь-якій іншій довжині слова обміну генерується й перевіряється 2 стоп-
біта. При асинхронній передачі поняття біта нерозривно зв'язане з тривалістю
сигналу, тому цілком можлива посилка нецілого числа стоп-бітів. Це може
знадобитися, якщо підключений до комп'ютера пристрій не програмується, а суворо
налагоджений на аналіз стоп-бітів заданої довжини.");
}
else if(reg==8){
Memo2->Clear();
Memo2->Lines->Add("Out1 управляє сигналом Out1. Якщо біт встановлений в 1,
сигнал Out1  встановлюється 1. При завданні значення 0 сигнал встановлюється в
нульовий рівень.");
}
else if(reg==9){
Memo2->Clear();
Memo2->Lines->Add("PE є індикатором 'помилки парності' (Parity Error). Помилка
парності фіксується, якщо в прийнятому символі виявлене некоректне значення біта
контролю парності. Цей біт приймає значення 0 після читання регістра LSR.");
}
else if(reg==10){
Memo2->Clear();
```

Кафедра_КОПЗ_2023рік

```
Memo2->Lines->Add("TERI є індикатором заднього фронту сигналу RI. Цей біт
приймає значення 1 при зміні сигналу RI з рівня логічної 1 на рівень логічного
нуля.");
}
}
//---------------------------------------------------------------------

void __fastcall TForm4::Button17Click(TObject *Sender)
{
Button11->Font->Size=8;
Button12->Font->Size=8;
Button13->Font->Size=8;
Button14->Font->Size=8;
Button15->Font->Size=8;
Button16->Font->Size=8;
Button17->Font->Size=14;
Button18->Font->Size=8;
if(reg==1){
Memo2->Clear();
Memo2->Lines->Add("біт даних для передачі");
}
else if(reg==2){
Memo2->Clear();
Memo2->Lines->Add("прийнятий біт даних");
}
else if(reg==3){
Memo2->Clear();
Memo2->Lines->Add("один з бітів дільника");
}
else if(reg==4){
Memo2->Clear();
Memo2->Lines->Add("один з бітів дільника");
}
else if(reg==5){
Memo2->Clear();
Memo2->Lines->Add("IFB  задає переривання при звільненні регістра буфера
прийнятих даних:");
Memo2->Lines->Add("1 - переривання виробляється;");
Memo2->Lines->Add("0 - переривання заборонене.");

}
else if(reg==6){
Memo2->Clear();
Memo2->Clear();
Memo2->Lines->Add("2 біт I Type. Біти  I Type визначають тип очікуючого
переривання, якщо воно зберігається контролером (що визначається бітом II):");
Memo2->Lines->Add("11 - змінився стан лінії приймача;  ");
Memo2->Lines->Add("10 - прийняті дані доступні; ");
Memo2->Lines->Add("01 - звільнений регістр буфера;");
Memo2->Lines->Add("00 - змінився стан модему. ");
}
else if(reg==7){
Memo2->Clear();
Memo2->Lines->Add("1 біт WLS. Біти WLS визначають довжину слова обміну:");
Memo2->Lines->Add("00 - 5 бітів;  ");
Memo2->Lines->Add("01 - 6 бітів; ");
Memo2->Lines->Add("10 - 7 бітів; ");
Memo2->Lines->Add("11 - 8 бітів.");

}
else if(reg==8){
Memo2->Clear();
Memo2->Lines->Add("RTS управляє сигналом 'запит на передачу' (Request to Send).
При значенні цього біта, рівному 1, сигнал 'запит на передачу' встановлюється
рівним 1. При завданні значення 0 сигнал встановлюється в нульовий рівень.");
}
else if(reg==9){
Memo2->Clear();
```

Кафедра_КбПЗ_2023рік

```
Memo2->Lines->Add("OR є індикатором 'помилки переповнення' (Overrun Error).
Помилка переповнення фіксується в тому випадку, якщо при записі чергового
символу в регістр RBR виявлено, що попередній вміст цього регістра не зчитане й,
таким чином, воно загублено. Цей біт приймає значення 0 після операції читання
регістра LSR.");
}
else if(reg==10){
Memo2->Clear();
Memo2->Lines->Add("DDSR  є індикатором зміни сигналу Data Set Ready (DSR). Цей
біт приймає значення 1 при зміні сигналу DSR після останньої операції читання
регістра MSR.");
}
}
//---------------------------------------------------------------------

void __fastcall TForm4::Button18Click(TObject *Sender)
{
Button11->Font->Size=8;
Button12->Font->Size=8;
Button13->Font->Size=8;
Button14->Font->Size=8;
Button15->Font->Size=8;
Button16->Font->Size=8;
Button17->Font->Size=8;
Button18->Font->Size=14;
if(reg==1){
Memo2->Clear();
Memo2->Lines->Add("біт даних для передачі");
}
else if(reg==2){
Memo2->Clear();
Memo2->Lines->Add("прийнятий біт даних");
}
else if(reg==3){
Memo2->Clear();
Memo2->Lines->Add("один з бітів дільника");
}
else if(reg==4){
Memo2->Clear();
Memo2->Lines->Add("один з бітів дільника");
}
else if(reg==5){
Memo2->Clear();
Memo2->Lines->Add("IDA  визначає переривання при доступності прийнятих даних:");
Memo2->Lines->Add("1 - переривання виробляється;");
Memo2->Lines->Add("0 - переривання заборонене.");
}
else if(reg==6){
Memo2->Clear();
Memo2->Lines->Add("Біт II є індикатором очікуючого переривання:");
Memo2->Lines->Add("0 - контролер послідовного інтерфейсу зберігає
переривання;");
Memo2->Lines->Add("1 - немає переривань, що очікують обробки.");

}
else if(reg==7){
Memo2->Clear();
Memo2->Lines->Add("2 біт WLS. Біти WLS визначають довжину слова обміну:");
Memo2->Lines->Add("00 - 5 бітів;  ");
Memo2->Lines->Add("01 - 6 бітів; ");
Memo2->Lines->Add("10 - 7 бітів; ");
Memo2->Lines->Add("11 - 8 бітів.");
}
else if(reg==8){
Memo2->Clear();
Memo2->Lines->Add("DTR задає рівень сигналу 'готовність термінала' (Data
Terminal Ready).  Якщо біт встановлений в 1, сигнал 'готовність термінала'
встановлюється рівним 1. При задані значення 0 сигнал встановлюватися в нульовий
рівень.");
```

Кафедра_КбПЗ_2023рік

```
}
else if(reg==9){
Memo2->Clear();
Memo2->Lines->Add("DR індикатор доступності прийнятих даних. Цей біт завжди
встановлюється в 1, коли приймачем повністю прийняті символ і поміщений у
регістр RBR. Біт приймає значення 0 після операції читання з регістра RBR.");
}
else if(reg==10){
Memo2->Clear();
Memo2->Lines->Add("DTCS є індикатором зміни сигналу Clear to Send (CTS). Цей біт
приймає значення 1 при зміні сигналу CTS після останньої операції читання
регістра MSR.");
}
}
//---------------------------------------------------------------------
```

Кафедра_КбПЗ_2023рік

Файл help.h – бібліотека для файлу help.cpp

```
//---------------------------------------------------------------------

#ifndef helpH
#define helpH
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------
class TForm4 : public TForm
{
__published: // IDE-managed Components
        TLabel *Label1;
        TMemo *Memo1;
        TButton *Button1;
        TButton *Button2;
        TButton *Button3;
        TButton *Button4;
        TButton *Button5;
        TButton *Button6;
        TButton *Button7;
        TButton *Button8;
        TButton *Button9;
        TButton *Button10;
        TLabel *Label2;
        TButton *Button11;
        TButton *Button12;
        TButton *Button13;
        TButton *Button14;
        TButton *Button15;
        TButton *Button16;
        TButton *Button17;
        TButton *Button18;
        TMemo *Memo2;
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall Button5Click(TObject *Sender);
        void __fastcall Button6Click(TObject *Sender);
        void __fastcall Button7Click(TObject *Sender);
        void __fastcall Button8Click(TObject *Sender);
        void __fastcall Button9Click(TObject *Sender);
        void __fastcall Button10Click(TObject *Sender);
        void __fastcall Button11Click(TObject *Sender);
        void __fastcall Button12Click(TObject *Sender);
        void __fastcall Button13Click(TObject *Sender);
        void __fastcall Button14Click(TObject *Sender);
        void __fastcall Button15Click(TObject *Sender);
        void __fastcall Button16Click(TObject *Sender);
        void __fastcall Button17Click(TObject *Sender);
        void __fastcall Button18Click(TObject *Sender);
private:    // User declarations
public:          // User declarations
        __fastcall TForm4(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm4 *Form4;
//---------------------------------------------------------------------
#endif
```

Кафедра КбПЗ_2023рік

Файл help2.cpp – довідка про контакти

послідовного порту

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "help2.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm5 *Form5;
//---------------------------------------------------------------------
__fastcall TForm5::TForm5(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm5::Button1Click(TObject *Sender)
{
Form5->Close();
}
//---------------------------------------------------------------------
```

Кафедра_КбПЗ_2023рік

Файл help2.h – бібліотека для файлу help2.cpp

```
//---------------------------------------------------------------------

#ifndef help2H
#define help2H
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------
class TForm5 : public TForm
{
__published:      // IDE-managed Components
        TMemo *Memo1;
        TButton *Button1;
        void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm5(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm5 *Form5;
//---------------------------------------------------------------------
#endif
```

Кафедра_КбПЗ_2023рік

Файл about.cpp – Вікно "Про програму..."

```
//---------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "about.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
//---------------------------------------------------------------------
__fastcall TForm3::TForm3(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
void __fastcall TForm3::Button1Click(TObject *Sender)
{
Form3->Close();
}
//---------------------------------------------------------------------
```

Кафедра_КбПЗ_2023рік

Файл about.h – бібліотека для файлу about.cpp

```
//---------------------------------------------------------------------

#ifndef aboutH
#define aboutH
//---------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <jpeg.hpp>
//---------------------------------------------------------------------
class TForm3 : public TForm
{
__published:      // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TButton *Button1;
        TLabel *Label8;
        TImage *Image1;
        void __fastcall Button1Click(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm3(TComponent* Owner);
};
//---------------------------------------------------------------------
extern PACKAGE TForm3 *Form3;
//---------------------------------------------------------------------
#endif
```

Кафедра_КбПЗ_2023рік