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 процесу передачі даних через USB”

Виконав здобувач вищої освіти
IV курсу, групи КІ-20-3СК
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Іванішив Д.С.
«____» _____ 2023 р.

Керівник проекту
доктор технічних наук, доцент
_____ Коваленко О.В.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

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

Іванішиву Данилу Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи візуалізації процесу передачі даних через USB*

2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 8-02 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 процесу передачі даних через USB*

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*

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 О.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Іванішив Д.С.
(прізвище та ініціали)

АНОТАЦІЯ

Іванішив Д.С. Програмне забезпечення системи візуалізації процесу передачі даних через USB. 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 процесу передачі даних через USB.

Метою розробки є програмне забезпечення системи візуалізації процесу передачі даних через USB.

Результат роботи – програмна реалізація системи візуалізації процесу передачі даних через USB.

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, передача даних, USB

ABSTRACT

Ivanishyv D.S. USB Data Transfer Process 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 the data transfer process via USB.

The purpose of the development is the software of the visualization system of the data transfer process via USB.

The result of the work is a software implementation of the visualization system of the data transfer process via USB.

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, data transfer, USB

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЕОМ	–	електронно-обчислювальна машина
КПК	–	кишеньковий персональний комп'ютера
ПЗ	–	программе забезпечення
ПК	–	персональниф комп'ютер
ОЗП	–	оперативний запам'товуючий пристрій
ФАПЧ	–	фазове автопідлаштування частоти
COM	–	послідовний порт
CRC	–	алгоритм обчислення контрольної суми
FIFO	–	буфер, що працює за принципом "перший зайшов – перший вийшов"
EOF	–	кінець пакета даних
LPT	–	парелельний порт
NRZI	–	метод кодування сигналів USB-інтерфейсу
PID	–	ідентифікатор процесу
SIE	–	автомат послідовного інтерфейсу
SOF	–	початок пакета даних
UFI	–	функціональний інтерфейсний модуль
USB	–	універсальна послідовна шина

ВСТУП

Актуальність теми. Розвиток засобів обчислювальної техніки відбувається в багатьох напрямках, що розширюють сферу застосування ЕОМ і підвищують ефективність їхнього використання. Однією з відносно нових технологій є технологія USB, яка вже випущена у трьох стандартах: USB 1.1, USB 2.0 та USB 3.0.

USB став дійсно універсальним стандартом і відповідні роз'єми вже мають практично всі цифрові пристрої. До комп'ютера по USB можна підключати мишки, клавіатури, принтери, сканери й безліч інших пристроїв.

Стандартні роз'єми USB є практично на кожному ноутбуці й ПК, а його зменшені варіації (mini-USB і micro-USB) використовуються в портативній техніці (мобільних телефонах, КПК, MP3-плеєрах, цифрових фотокамерах і т.д.).

Роз'єм USB може бути використаний як альтернативне джерело живлення, тому що по USB-кабелю подається електроживлення з напругою 5Вольт і силою струму 500мА. Тому периферійні пристрої (наприклад, мишки й флешки) можуть працювати, живлячись енергією прямо від комп'ютера.

Очевидні на сьогоднішній день переваги стандарту USB не обмежують його застосування в області мультимедійних додатків. Використання швидкісного, з можливістю підключення великої кількості пристроїв, USB-інтерфейсу в сфері комунікацій або спеціалізованого збору інформації й моніторингу підніме на більш високий якісний рівень роботу й обслуговування пристроїв, які спочатку проектувалися для портів COM і LPT.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи візуалізації процесу передачі даних через USB.

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

– Огляд існуючих систем візуалізації процесу передачі даних через USB.

– Дослідження системи візуалізації процесу передачі даних через USB.

– Програмна реалізація системи візуалізації процесу передачі даних через USB.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі візуалізації процесу передачі даних через USB.

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

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

USB (англ. Universal Serial Bus) – універсальна послідовна шина, призначена для підключення периферійних пристроїв. Шина USB являє собою послідовний інтерфейс передачі даних для середньошвидкісних та низькошвидкісних периферійних пристроїв.

Пристрої, що використовують USB-роз'єм**:** клавіатури, миші, джойстики, принтери, сканери, флеш-диски, дигитайзери, цифрові фотокамери, модеми для звичайних телефонних ліній, ланцюги керування монітором комп'ютера, колонки, ISDN модеми, зовнішні накопичувачі класу Iomega Zip, офісні АТС та ін.

Універсальна послідовна шина забезпечує більшу швидкість обміну даними між комп'ютером і периферійним пристроєм, у порівнянні зі стандартними портами введення/виведення (послідовний – COM і паралельний – LPT). Максимальна пропускна здатність USB 1.1 становить 12 Мб/с і 480 – для наступного покоління цього стандарту USB 2.0, що значно перевищує можливість послідовної передачі COM-порту. Для підтримки низькошвидкісних пристроїв передбачений режим передачі зі швидкістю 1,5 Мб/с.

Величезна розмаїтість периферійних пристроїв і їхнє поступове здешевлення дозволяють все більшому числу користувачів ПК скористатися перевагами USB-інтерфейсу.

Розробка специфікацій на шину USB здійснюється в рамках міжнародної некомерційної організації USB Implementers Forum (USB-IF), що об'єднує розроблювачів і виробників устаткування із шиною USB.

Для підключення периферійних пристроїв до шини USB використовується чотирьохдротовий кабель, при цьому два дроти (кручена пара) у диференціальному включенні використовуються для прийому й передачі даних, а

інші два дроти – для живлення периферійного пристрою.

Завдяки вбудованим лініям живлення, USB дозволяє підключати периферійні пристрої без власного джерела живлення (максимальна сила струму, споживаного пристроєм по лініях живлення шини USB, не повинна перевищувати 500мА).

До одного контролера шини USB можна приєднати до 127 пристроїв через ланцюжок концентраторів, що використовують топологію «зірка» (рисунок 1.1).

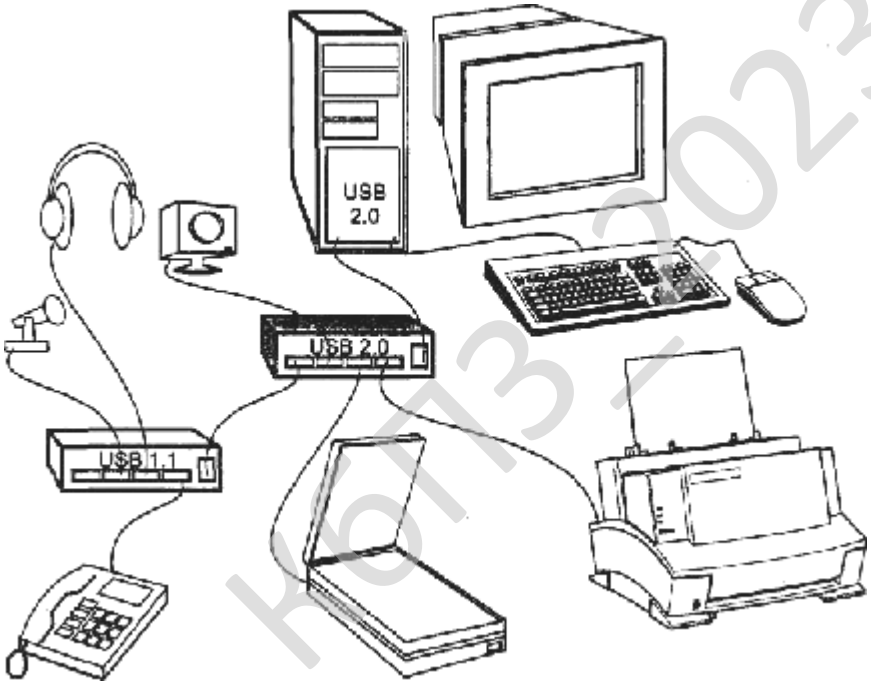

Рисунок 1.1 – Приклад конфігурації з'єднань USB-пристроїв

У цей час широко використовуються пристрої, виконані у відповідності зі специфікацією USB 2.0. Ведеться розробка специфікації USB 3.0.

Тому доцільно підключати до USB практично будь-які периферійні пристрої, крім цифрових відеокамер і високошвидкісних жорстких дисків.

Особливо зручний цей інтерфейс для підключення часто підключаємих/відключаємих пристроїв, таких як, наприклад, цифрові фотокамери. Конструкція роз'ємів для USB розрахована на багаторазове приєднання/від'єднання.

Існує п'ять видів USB-роз'ємів, всі вони показані на рисунку 1.2

Рисунок 1.2 – Зліва на право: micro USB, mini USB, B-type, A-type роз'єм, A-type конвектор

Micro USB – використовується в самих мініатюрних пристроях, наприклад, у плеєрах та мобільних телефонах.

Mini USB – також часто знаходиться на плеєрах, мобільних телефонах й на цифрових фотоапаратах, КПК тощо.

B-type – повнорозмірний роз'єм, що встановлюється в принтерах, сканерах та інших пристроях, де розмір не має дуже принципового значення;

A-type (приймач) – роз'єм, що використовується у комп'ютерах (або на подовжувачах USB), куди підключається коннектор типу A-type.

A-type (вилка) – коннектор, що підключається безпосередньо до ПК.

Основна мета даного стандарту, поставлена перед його розроблювачами – створити реальну можливість користувачам працювати в режимі Plug&Play з периферійними пристроями. Це означає, що повинно бути передбачене підключення пристрою до працюючого комп'ютера, автоматичне розпізнавання його негайно після підключення й наступної установки відповідних драйверів. Крім цього, бажано живлення малопотужних пристроїв подавати із самої шини. Швидкість шини повинна бути достатньою для переважної більшості периферійних пристроїв. Попутно вирішується історична проблема недостачі ресурсів на внутрішніх шинах IBM PC сумісного комп'ютера – контролер USB займає тільки одне переривання незалежно від кількості підключених до шини пристроїв.

1.2 Область застосування

Завдяки своїй універсальності й здатності ефективно передавати різнорідний трафік, шина USB застосовується для підключення до PC найрізноманітніших пристроїв. Вона покликана замінити традиційні порти PC – COM і LPT, а також порти ігрового адаптера й інтерфейсу MIDI. Специфікація USB 2.0 дозволяє говорити й про підключення традиційних «клієнтів» шин ATA й SCSI, а також захвату частини ніші застосування шини FireWire. Привабливість USB надає можливість підключення/відключення пристроїв на ходу й можливість їхнього використання практично відразу, без перезавантаження ОС. Зручна й можливість підключення великої кількості (до 127) пристроїв до однієї шини, щоправда, при наявності хабів. Хост-контролер інтегрований у більшість сучасних системних плат. Випускаються й карти розширення з контролерами USB (звичайно для шини PCI).

Однак повсемісне застосування USB стримується недостатньою активністю розроблювачів ПЗ (виробників устаткування): переглядаючи переліки пристроїв, ми бачимо, що для всіх вказується підтримка в Windows 98/SE/ME, а от у графах Linux, MacOS, Unix і навіть Windows 2000 часто стоять неприємні позначки N/A (Not Allowed – «не дозволена»).

Основні області застосування USB:

– *Пристрої введення* – клавіатури, миші, трекболи, планшетні покажчики і т.п. Тут USB надає для різних пристроїв єдиний інтерфейс. Доцільність використання USB для клавіатури неочевидна, хоча в парі з мишею USB (підключається до порту хаба, вбудованого в клавіатуру) скорочується кількість кабелів, що тягнуться від системного блоку на стіл користувача.

– *Принтери*. USB 1.1 забезпечує приблизно ту ж швидкість, що й LPT-порт у режимі ECP, але при використанні USB не виникає проблем з довжиною кабелю й підключенням декількох принтерів до одного комп'ютера (правда, потрібні хаби). USB 2.0 дозволить прискорити друк в режимі високого дозволу за

рахунок скорочення часу на передачу великих масивів даних. Однак є проблема зі старим ПЗ, що безпосередньо працює з LPT-портом на рівні регістрів, – на принтер USB воно друкувати не зможе.

– *Сканери*. Застосування USB дозволяє відмовитися від контролерів SCSI або від застосування LPT-порту. USB 2.0 при цьому дозволить ще й підвищити швидкість передачі даних.

– *Аудіопристрої* – колонки, мікрофони, головні телефони (навушники). USB дозволяє передавати потоки аудіоданих, достатні для забезпечення найвищої якості. Передача в цифровому вигляді від самого джерела сигналу (мікрофона з вбудованим перетворювачем і адаптером) до приймача й цифрова обробка в хост-комп'ютері дозволяють позбутися від наведень, властивих аналоговій передачі аудіосигналів. Використання цих аудіо-компонентів дозволяє в ряді випадків позбутися від звукової карти комп'ютера – аудіокодек (АЦП і ЦАП) виводиться за межі комп'ютера, а всі функції обробки сигналів (мікшер, еквалайзер) реалізуються центральним процесором чисто програмно. Аудіопристрої можуть і не мати власне колонок і мікрофона, а обмежитися перетворювачами й стандартними гніздами («Джеками») для підключення звичайних аналогових пристроїв.

– *Музичні синтезатори* й *MIDI-контролери* з інтерфейсом USB. Шина USB дозволяє комп'ютеру обробляти потоки безлічі каналів MIDI (пропускна здатність традиційного інтерфейсу MIDI уже набагато нижче можливостей комп'ютера).

– *Відео*– і *фотокамери*. USB 1.1 дозволяє передавати статичні зображення будь-якого дозволу за прийнятний час, а також передавати потік відеоданих (живе відео) з достатньою частотою кадрів (25-30 Кб/с) тільки з невисоким дозволом або стисненням даних, від якого, природно, страждає якість зображення. USB 2.0 дозволяє передавати потік відеоданих високого дозволу без стиснення (і втрати якості). З інтерфейсом USB випускають як камери, так і пристрої захвату зображення з телевізійного сигналу й TV-тюнери.

– *Комунікації*. З інтерфейсом USB випускають різноманітні модеми, включаючи кабельні й xDSL, адаптери високошвидкісного інфрачервоного зв'язку (IrDA FIR) – шина дозволяє перебороти межу швидкості COM-порту (115,2 Кб/с), не підвищуючи завантаження центрального процесора. Випускаються й мережні адаптери Ethernet, що підключаються до комп'ютера по USB. Для з'єднання декількох комп'ютерів у локальну мережу випускаються спеціальні пристрої, що виконують комутацію пакетів між комп'ютерами. Безпосередньо (без додаткових пристроїв) портами USB з'єднати між собою навіть два комп'ютери не можна – на одній шині може бути присутнім лише один хост-контролер. Спеціальний пристрій для зв'язку пари комп'ютерів виглядає як «таблетка», врізана в кабель USB із двома качанами типу «А» на кінцях. Об'єднання більше двох комп'ютерів ускладнюється й топологічними обмеженнями USB: довжина одного сегмента кабелю не повинна перевищувати 5 м, а використовувати хаби для збільшення дальності неефективно (кожний хаб дає всього 5 м додаткового видалення).

– *Перетворювачі інтерфейсів* дозволяють через порт USB, наявний тепер практично на всіх комп'ютерах, підключати пристрої з найрізноманітнішими інтерфейсами: Centronics і IEEE 1284 (LPT-порти), RS-232C (емуляція UART 16550A – основи COM-портів) та інші послідовні інтерфейси (RS-422, RS-485, V.35...), емулятори портів клавіатури й навіть Game-порту, перехідники на шину ATA, ISA, PC Card та будь-які інші, для яких досить продуктивності. Тут USB приходить на допомогу, коли встає проблема 2-го (3-го) LPT– або COM-порту в блокнотному ПК та в інших ситуаціях. При цьому ПЗ перетворювача може забезпечити емуляцію класичного варіанта «заліза» стандартних портів IBM PC, але тільки під керуванням ОС захищеного режиму. Додаток MS-DOS може звертатися до пристроїв по адресах введення-виведення, пам'яті, перериваннями, каналами DMA, але тільки із сеансу MS-DOS, відкритого в ОС із підтримкою USB (частіше це Windows). При завантаженні «голої» MS-DOS така допомога не працює. Перетворювачі інтерфейсів дозволяють продовжити життя пристроям із традиційними інтерфейсами, що зживаються з PC

новими специфікаціями. Швидкість передачі даних через конвертер USB – LPT може виявитися навіть вище, ніж у реального LPT-порту, що працює в режимі SPP.

– *Пристрої зберігання інформації* – вінчестери, пристрої читання й запису CD і DVD, стримери – при використанні USB 1.1 одержують швидкість передачі, порівнянну зі швидкістю їхнього підключення до LPT, але більш зручний інтерфейс (як апаратний, так і програмний). При переході на USB 2.0 швидкість передачі даних стає порівнянною з ATA й SCSI, а обмежень по кількості пристроїв досягти важко. Особливо цікаве використання USB для електронних пристроїв енергонезалежного зберігання (на флеш-пам'яті) – такий накопичувач досить компактний та ємний. Випускаються пристрої для мобільного підключення накопичувачів з інтерфейсом ATA-ATAPI – по суті, це лише перетворювачі інтерфейсів, поміщені в коробку формату 5" або 3,5", а іноді виконані прямо в корпусі 36-контактного роз'єму ATA. Є й пристрої читання-запису карт SmartMedia Card і CompactFlash Card.

– *Ігрові пристрої* – джойстики всіх видів (від «паличок» до автомобільного керма), пульти з різноманітними датчиками (безперервними й дискретними) і виконавчими механізмами – підключаються уніфікованим способом. При цьому виключається ресурсопожирающий інтерфейс старого ігрового адаптера (скасованого вже в специфікації PC'99).

– *Телефони* – аналогові й цифрові (ISDN). Підключення телефонного апарата дозволяє перетворити комп'ютер у секретаря з функціями автодозвона, автовідповідача, охорони й т.п.

– *Монітори* – тут шина USB використовується для керування параметрами монітора. Монітор повідомляє системі свій тип і можливості (параметри синхронізації) – це робилося й без USB по шині DDC. Однак USB-монітори дозволяють системі ще й управляти ними – регулювання яскравості, контрасту й т.п. можуть тепер виконуватися програмно, а не тільки від кнопок лицьової панелі монітора. У монітори, як правило, вбудовують хаби. Це зручно,

оскільки настільну периферію не завжди зручно вмикати в «підстольний» системний блок.

– *Електронні ключі* – пристрої з будь-яким рівнем інтелектуальності захисти – можуть бути виконані в корпусі вилок USB. Вони набагато компактніші й мобільніші аналогічних пристроїв для COM– і LPT-портів.

Звичайно ж, перерахованими класами пристроїв сфера застосування шини USB не обмежується. Популярність даного інтерфейсу призводить до постійного зростання USB-пристроїв.

Розроблене програмне забезпечення створене для навчання студентів роботі і системному програмуванню передачі даних через USB. Даний емулятор можна використовувати для виконання лабораторних робіт по програмуванню.

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

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

USB 1.0. Специфікація випущена в листопаді 1996 року.

Технічні характеристики:

– Висока швидкість обміну (full-speed signaling bit rate) – 12 Мб/с.

– Максимальна довжина кабелю для високої швидкості обміну – 5м.

– Низька швидкість обміну (low-speed signaling bit rate) – 1.5 Мб/с.

– Максимальна довжина кабелю для низької швидкості обміну – 3м.

– Можливе підключення пристроїв з різними швидкостями обміну.

– Відсутність необхідності в установці користувачем додаткових елементів, таких як термінатори для SCSI.

– Напруга живлення для периферійних пристроїв – 5В.

– Максимальна напруга живлення на один пристрій – 500мА

Символ стандарту USB 1.0 показаний на рисунку 2.1.

Рисунок 2.1 – Логотип USB

USB 1.1. Специфікація випущена у вересні 1998 року. Виправлено проблеми й помилки, виявлені у версії 1.0. Перша версія, що одержала масове поширення.

USB 2.0 Специфікація випущена у квітні 2000 року. USB 2.0 відрізняється від USB 1.1 введенням режиму Hi-speed. Символ стандарту USB 2.0 показаний на рисунку 2.2.

Рисунок 2.2 – Логотип USB 2.0 High Speed

Для пристроїв USB 2.0 регламентовано три режими роботи:

– Low-speed, 10-1500 Кб/с (використовується для інтерактивних пристроїв: клавіатури, миші, джойстики).

– Full-speed, 0,5-12 Мб/с (аудіо-, відеопристрою).

– Hi-speed, 25-480 Мб/с (відеопристрою, пристрою зберігання інформації).

Наступні модифікації

Наступні модифікації до специфікації USB публікуються в рамках Повідомлень про інженерні зміни (EngineeringChangeNotices – ECN). Найважливіші з модифікацій ECN представлені в наборі специфікацій USB 2.0 (USB2.0specificationpackage), доступному на сайті USB Implementers Forum.

USB OTG (від On-The-Go) – подальше розширення специфікації USB 2.0, призначене для легкого з'єднання периферійних USB-пристроїв один з одним без необхідності підключення до ПК.

Рисунок 2.3 – Логотип USB OTG

Символ стандарту USB OTG показаний на рисунку 2.3.

Наприклад, цифровий фотоапарат можна підключати до фотопринтера прямо, якщо вони обоє підтримують стандарт USB OTG. До моделей КПК і комунікаторів, що підтримують USB OTG, можна підключати деякі USB-пристрої. Звичайно це флеш-накопичувачі, цифрові фотоапарати, клавіатури, миші й інші пристрої, що не вимагають додаткових драйверів.

Цей стандарт виник через різко зростаючу останнім часом необхідності надійного з'єднання різних USB-пристроїв без використання ПК. У даній специфікації пристрою обходяться без персонального комп'ютера, тобто виступають як однорангові прийомопередатчики (насправді тільки створюється таке відчуття).

У дійсності ж пристрої визначають, який з них буде майстер-пристроєм, а який – підлеглим. Одноранговим інтерфейс USB бути не може.

USB wireless – новітня технологія USB (офіційна специфікація стала доступна тільки в травні 2005 року). Дозволяє організувати бездротовий зв'язок з високою швидкістю передачі інформації (до 480 Мб/с на відстані 3 метри й до 110 Мб/с на відстані 10 метрів). Символ стандарту USB wireless показаний на рисунку 2.4.

Рисунок 2.4 – Логотип USB wireless

23 липня 2007 року USB Implementers Forum (USB-IF) оголосила про сертифікацію шести перших споживчих продуктів з підтримкою Wireless USB.

USB 3.0 перебуває на фінальних стадіях розробки. Створенням USB 3.0 займаються компанії: Intel, Microsoft, Hewlett-Packard, Texas Instruments, NEC і NXP Semiconductors.

У специфікації USB 3.0 роз'єм й кабелі оновленого стандарту будуть фізично й функціонально сумісні з USB 2.0. Кабель USB 2.0 містить у собі чотири

лінії – пари для прийому/передачі даних, одну – для живлення й ще одну – для заземлення.

На додаток до них USB 3.0 додає п'ять нових ліній (у результаті чого кабель став набагато товщим), однак нові контакти розташовані паралельно стосовно старого на іншому контактному ряді. Тепер можна буде з легкістю визначити приналежність кабелю до тої або іншої версії стандарту, просто глянувши на його роз'єм.

Специфікація USB 3.0 підвищує максимальну швидкість передачі інформації до 4,8Гб/с – що на порядок більше 480 Мб/с, які може забезпечити USB 2.0. USB 3.0 може похвалитися не тільки більш високою швидкістю передачі інформації, але й збільшеною силою струму з 100 мА до 900 мА. Відтепер користувач зможе не тільки підживлювати від одного хаба набагато більшу кількість пристроїв, але й саме апаратне забезпечення, що раніше поставлялося з окремими блоками живлення, позбудеться від них.

Фінальна специфікація USB 3.0 з'явилася в 2008 році, а устаткування, що підтримує нову специфікацію, з'явиться в 2009-2010 роках.

Відповідно до специфікації, USB-шина може одночасно обслуговувати до 200 пристроїв, що цілком достатньо для будь-якого користувача. Підключати й відключати таке різноманіття периферії можна в гарячому режимі, не виключаючи й не перезавантажуючи комп'ютер.

Спосіб доступу до шини нагадує зіркову топологію (рисунок 2.5) з використанням концентратора (Hub). Саме до такого пристрою розгалуження підключається основна маса периферійних пристроїв. Досить підключити "хаб" до одного із двох (іноді чотирьох) USB-роз'ємів комп'ютера, так званому кореневому концентратору (Root Hub), щоб позбутися від мнимої недостачі USB-портів.

Тепер список використовуваних пристроїв не обмежується тільки клавіатурою або мишею, його можна поповнити, підключивши принтер, сканер, джойстик, фото– або відеокамеру й навіть колонку.

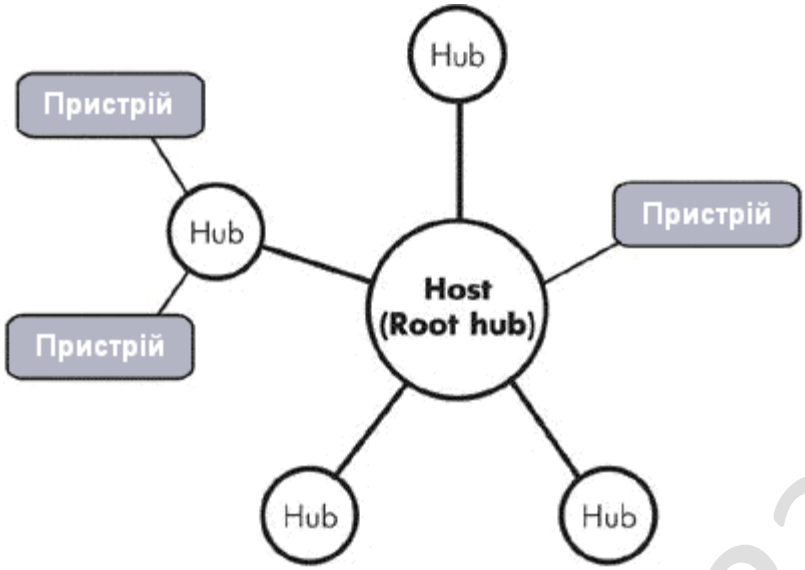

Рисунок 2.5 – Схема доступу до шини USB

Кожний пристрій, що підключається до шини, одержує свій унікальний ідентифікаційний номер, за допомогою якого здійснюється подальша конфігурування, керування й обмін даними. Сеанс зв'язку організується в пакетному режимі й може бути ініційований тільки самим комп'ютером (керуючим пристроєм).

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.

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 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 процесу передачі даних через USB.

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;

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.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Опис стандарту USB

Розглянемо будову та призначення контактів USB-роз'ємів. На рисунку 3.1 зображені USB-роз'єми стандарту А та В.

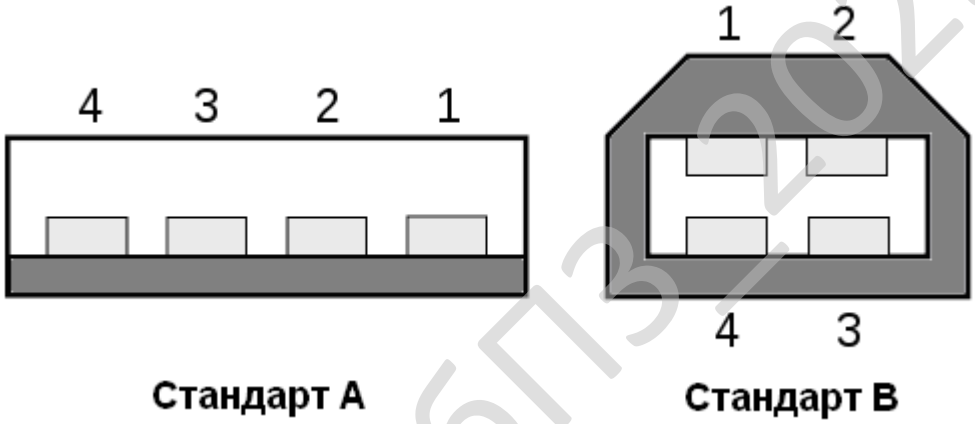

Рисунок 3.1 – Будова USB-роз'ємів стандарту А та В

Роз'єми стандарту А призначені тільки для підключення до джерела, тобто до комп'ютера або хабу. Роз'єми стандарту В призначені тільки для підключення до периферійного пристрою. Призначення контактів наведене у таблиці 3.1.

Таблиця 3.1 – Призначення контактів USB-роз'єму

Номер контакту	Призначення	Колір дроту
1	V BUS	Червоний
2	D–	Білий
3	D+	Зелений
4	GND	Чорний
Оплітка	Екран	Оплітка

GND – ланцюг "корпуса" для живлення периферійних пристроїв, VBus – +5V також для ланцюгів живлення.

Контакт D+ призначений для передачі даних по шині, а контакт D– для прийому даних.

Кабель для підтримки повної швидкості шини (full-speed) виконується як кручена пара, захищається екраном і може також використовуватися для роботи в режимі мінімальної швидкості (low-speed). Кабель для роботи тільки на мінімальній швидкості (наприклад, для підключення миші) може бути будь-яким і неекранованим.

Рівні сигналів передавачів у статичному режимі повинні бути нижче 0.3 В (низький рівень) або вище 2.8 В (високий рівень). Приймачі повинні витримувати вхідну напругу в межах -0.5...+3.8 В. Передавачі повинні мати можливість переходу у високоімпедансний стан для забезпечення двонаправленої напівдуплексної передачі даних по одній парі дротів.

Передача по двом дротам не обмежується лише диференціальними сигналами. Крім диференціального приймача, кожний пристрій має й лінійні приймачі сигналів D+ і D– , а передавачі цих ліній управляються індивідуально. Це дозволяє розрізняти множину станів лінії, використовуваних для організації апаратного інтерфейсу. Стани Diff0 і Diff1 визначаються по різниці потенціалів на лініях D+ і D– більше 200 мВ за умови, що на одній з них потенціал вище порога спрацьовування VSE. Стан, при якому на обох входах D+ і D– є присутнім низький рівень називається лінійним нулем (SE0 – single-ended zero). Інтерфейс визначає наступні стани:

– Data J State і Data K State – стан переданого біта (визначаються через стани Diff0 і Diff1).

– Idle State – пауза на шині.

– Resume State – сигнал "пробудження" для виводу пристрою зі сплячого режиму.

– Start of Packet (SOP) – початок пакета (перехід з "Idle" в "K").

– End of Packet (EOP) – кінець пакета.
– Disconnect – пристрій відключений від порту.
– Connect – пристрій підключений до порту.
– Reset – скидання пристрою.

Стани визначаються сполученнями диференціальних і лінійних сигналів, для повної й низкої швидкостей стану Diff0 і Diff1 мають протилежне призначення. У декодуванні стан Disconnect, Connect і Reset приймається в увагу й час знаходження ліній (більше 2.5 мс) у певних станах.

Шина має два режими передачі. Повна швидкість передачі сигналів USB становить 12 Мб/с, низька – 1.5 Мб/с. Для повної швидкості використовується екранована кручена пара з імпедансом 90 Ом і довжиною сегмента до 5 м, для низької – невитий і неекранований кабель при довжині сегмента до 3 м. Та сама система може використовувати обидва режими, перемикання для пристроїв здійснюється прозоро. Низька швидкість призначена для роботи з невеликою кількістю пристроїв, що не вимагають високої пропускної здатності каналу.

Швидкість, що використовується пристроєм, підключеним до конкретного порту визначається хабом по рівнях сигналів на лініях D+ і D-, що зміщаються навантажувальними резисторами R2 прийомопередатчиків. USB пристрій повинен вказувати свою швидкість шляхом підтяжки лінії D+ або D- за допомогою опору 1.5Ом +- 5% до напруги 3.3 В. Повношвидкісний пристрій використовує підтягуючий резистор підключений до лінії D+, щоб хост зміг визначити його як повношвидкісний пристрій. Цей опір хост або концентратор використовують для виявлення підключення пристрою до свого порту. Без нього ні хост, ні концентратор не зрозуміють, що до USB шини підключений новий пристрій.

Сигнали кодуються по методу NRZI (Non Retur n To Zero Invert) – при переході сигналу з 0 в 1 сигнал NRZI не змінюється, а при переході з 1 в 0 – змінюється на протилежний. Кожному пакету передує поле SYNC, що дозволяє приймачу настроїтися на частоту передавача.

Крім сигнальної пари кабель має лінії VBus і GND для передачі напруги живлення 5В до пристроїв.

Модель передачі даних

З погляду даних, кожний пристрій USB являє собою набір незалежних кінцевих точок, з якими хост-контролер може обмінюватися інформацією. Кінцеві точки описуються наступними параметрами:

– Необхідна частота доступу до шини й припустимі затримки обслуговування.

– Необхідна смуга пропускання каналу.

– Номер точки.

– Вимоги до обробки помилок.

– Максимальні розміри переданих і прийнятих пакетів.

– Тип обміну.

– Напрямок обміну (для суцільного й ізохронного обміну).

Кожний пристрій обов'язково має кінцеву точку з номером 0, що використовується для ініціалізації й загального керування логічним пристроєм та опитування його стану. Ця точка завжди сконфігурована при включенні живлення й підключенні пристрою до шини й підтримує передачі типу "керування".

Крім нульової точки, пристрою-функції можуть мати додаткові точки, що реалізують корисні обміни даними. Низькошвидкісні пристрої можуть мати до двох додаткових точок, повношвидкісні – до 16 точок введення й до 16 точок виведення. Всі ці точки не можуть бути використані до їхнього конфігурування.

Кожна кінцева точка може бути налагоджена на один з декількох типів USB-передач. Вона може бути пов'язана з одним банком або двома банками двопортового ОЗП (ДПП), використовуваного для зберігання поточних даних. Якщо використовуються два банка, то один банк ДПП зчитується або записується процесором, а інші зчитуються або записуються периферійним USB-пристроєм. Дана особливість є обов'язковою для ізохронних кінцевих точок. Опис кінцевих точок USB наведений у таблиці 3.2

Таблиця 3.2 – Опис кінцевих точок USB

Номер кінцевої точки	Позначення	Використання двох банків	Максимальний розмір кінцевої точки
0	EP0	немає	32
1	EP1	так	64
2	EP2	так	64
3	EP3	немає	8

Припинення й поновлення автоматично визначається USB-пристроєм, що повідомляє про це процесор шляхом генерації переривання. У деяких мікроконтролерів підтримується можливість відправлення сигналу активізації шини головному USB-контролеру (USB-хост).

Канал (pipe) – модель передачі даних між хост-контролером і кінцевою точкою пристрою (рисунок 3.2). Є два типи каналів – потоки й повідомлення. Канали організуються при конфігуруванні пристроїв USB. Один канал повідомлень (Control Pipe 0), по якому передається інформація конфігурування, керування й стану, обов'язково існує для кожного включеного пристрою.

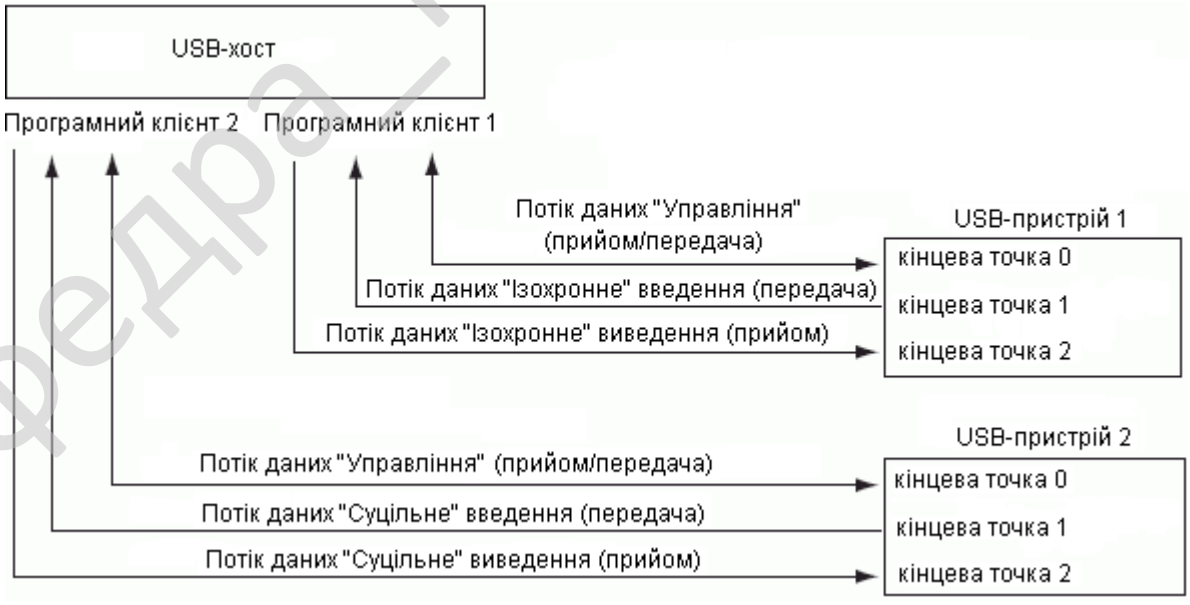

Рисунок 3.2 – Модель передачі даних між USB-хостом та кінцевими точками пристроїв

Потік доставляє дані від одного кінця каналу до іншого, він завжди односпрямований. Той самий номер кінцевої точки може використовуватися для двох потокових каналів – введення й виведення. Потік може використовувати наступні типи обміну – суцільний, переривання й ізохронний. Доставка завжди йде в порядку "перший увійшов – перший вийшов". Дані потоку не структуровані.

Повідомлення мають формат, визначений специфікацією USB. Обмін повідомленнями відрізняється від потоків: хост відсилає запит до кінцевої точки, після якого передається або приймається потік повідомлення, за яким слідує пакет з інформацією про стан кінцевої точки. При обробці помилок можливе скидання неопрацьованих повідомлень. Двосторонній обмін повідомленнями адресується до однієї й тої ж кінцевої точки. Для доставки повідомлень використовується тільки обмін типу "керування".

Типи передачі даних

USB підтримує кілька режимів зв'язку, як односпрямованих, так і двоспрямованих. Передача даних здійснюється між ПЗ хоста й конкретною кінцевою точкою пристрою. Пристрій може мати кілька кінцевих точок, зв'язок з кожною з них встановлюється незалежно від інших.

В архітектурі USB існують чотири типи переданих даних:

1. **Керуючі посилки** (Control transfers) – використовуються для конфігурування під час підключення й у процесі роботи для керування пристроями. Протокол забезпечує гарантовану доставку даних. Довжина поля даних керуючої посилки не перевищує 64 байти для повної швидкості й 8 байтів для низкої.

2. **Суцільні передачі** (Bulk Data Transfer) порівняно великих пакетів без жорстких вимог до часу доставки. Ці передачі займають всю вільну смугу пропускання шини не зайняту іншими класами передач. Пакети мають поле даних розміром 8, 16, 32 або 64 байт. Пріоритет цих передач найнижчий, вони можуть призупинитися при великому завантаженні шини. Допускаються тільки на повній швидкості передачі.

3. **Переривання** (Interrupts) – короткі (до 64 байт на повній швидкості й до 8 на низкій) передачі типу символів, що вводяться, або координат. Переривання мають спонтанний характер і повинні обслуговуватися не повільніше ніж того вимагає пристрій. Межа часу обслуговування встановлюється в діапазоні 1-255мс для повної швидкості й 10-255мс для низкої.

4. **Ізохронні передачі** – безперервні передачі в реальному часі, що займають попередньо погоджену частину пропускної здатності шини й мають задану затримку доставки. У випадку виявлення помилки ізохронні дані передаються без повтору – недійсні пакети просто ігноруються.

Узагальнемо інформацію про типи передачі даних USB-шини у таблиці 3.3.

Таблиця 3.3 – Типи передачі даних USB-шини

Передача	Напрямок	Пропускна здатність	Виявлення помилок	Повторна відправка
Керування	двоспрямований	не гарантована	є	автоматично
Ізохронна	односпрямований	гарантована	є	немає
Переривання	односпрямований	не гарантована	є	є
Суцільна	односпрямований	не гарантована	є	є

Смуга пропускання шини ділиться між всіма встановленими каналами. Виділена смуга закріплюється за каналом і якщо установка нового каналу вимагає такої смуги, що не вписується в уже існуючий розподіл, запит на виділення каналу відкидається.

Архітектура USB передбачає внутрішню буферизацію всіх пристроїв. USB повинна забезпечувати обмін з такою швидкістю, що б затримка даних у пристрої, викликаний буферизацією, не перевищувала одиниць мілісекунд.

Ізохронні передачі класифікуються по способу синхронізації кінцевих точок із системою: розрізняють асинхронний, синхронний і адаптивний класи пристроїв, кожному з яких відповідає свій тип каналу USB.

Протокол

Транзакції по USB складаються із двох-трьох пакетів. Кожна транзакція планується й починається з ініціативи контролера, що посилає пакет-маркер. Цей пакет описує тип і напрямок передачі, адресу пристрою та номер кінцевої точки.

У кожній транзакції можливий обмін між кінцевою точкою адресуємого пристрою і хоста. Адресуємий маркером пристрій USB розпізнає свою адресу й підготовляється до обміну. Джерело даних передає пакет даних (або повідомлення про відсутність даних, призначених для передачі). Після успішного прийому пакета приймач посилає пакет підтвердження (або "рукостискання"), що підтверджує прийом інформації. Послідовність пакетів у транзакціях ілюструє рисунок 3.2

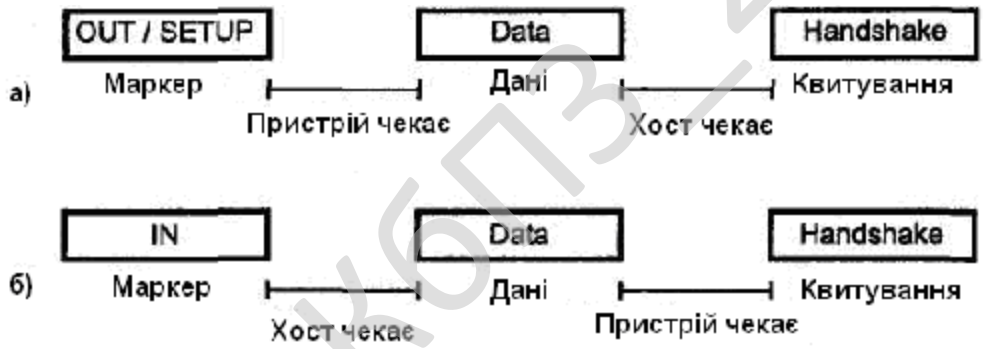

Рисунок 3.3 – Послідовність пакетів: виведення (а), введення (б)

Хост-контролер організує обміни із пристроями відповідно до свого плану розподілу ресурсів. Контролер циклічно (з періодом 1,0±0,0005 мс) формує кадри, у які укладаються всі заплановані транзакції (рисунок 3.3).

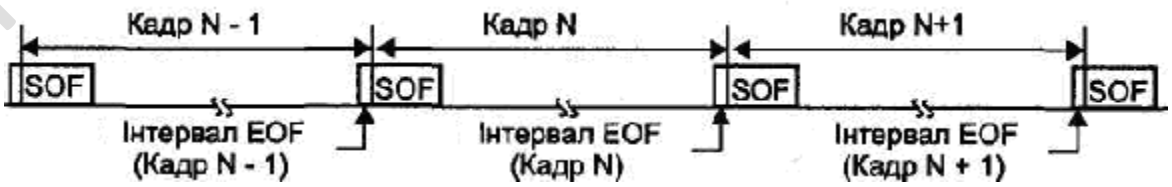

Рисунок 3.4 – Потік кадрів USB

Кожний кадр починається з посилки маркера SOF (Start Of Frame), який є синхронізуючим сигналом для всіх пристроїв, включаючи хаби. Наприкінці кожного кадру виділяється інтервал часу EOF (End Of Frame), на час якого хаби забороняють передачу в напрямку до контролера. У режимі HS пакети SOF передаються на початку кожного мікрокадру (період 125±0,0625 мкс). Хост планує завантаження кадрів так, щоб у них завжди перебувало місце для транзакцій керування й переривань. Вільний час кадрів може заповнюватися передачами масивів (bulk transfers). У кожному (мікро)кадрі може бути виконано кілька транзакцій, їхнє припустиме число залежить від довжини поля даних кожної з них.

Для виявлення помилок передачі кожний пакет має контрольні поля CRC-кодів, що дозволяють виявляти всі одиночні й подвійні бітові помилки. Апаратні засоби виявляють помилки передачі, а контролер автоматично робить трикратну спробу передачі. Якщо повтори безуспішні, повідомлення про помилку передається клієнтському ПЗ.

Всі подробиці організації транзакцій від клієнтського ПЗ ізолюються контролером USB і його системним програмним забезпеченням.

Планування транзакцій забезпечує керування потоковими каналами. На апаратному рівні використання відмови від транзакції (NACK) при неприпустимій інтенсивності транзакцій на шині охороняє буфери від переповнення або від переспустошення. Маркери відкинутих транзакцій передаються знову у вільний для шини час. Керування потоками дозволяє гнучко планувати обслуговування одночасних різнорідних потоків даних.

Стійкість до помилок забезпечують наступні властивості USB:

1. Висока якість сигналів, забезпечувана диференціальними приймачами, передавачами й екрануванням кабелів.
2. Захист полів керування й даних CRC-кодами.
3. Виявлення підключення й відключення пристроїв, конфігурування ресурсів на системному рівні.

4. Самовідновлення протоколу з використанням тайм-ауту при втраті пакетів.

5. Керування потоком для забезпечення ізохронності й керування апаратними буферами.

6. Незалежність одних функцій від невдалих обмінів з іншими функціями, забезпечувана конструкцією каналів.

Для виявлення помилок передачі кожний пакет використовує контрольні поля CRC-кодів, що дозволяють виявляти одиночні й подвійні бітові помилки. Апаратні засоби виявляють помилки передачі, а контролер робить трикратну спробу передачі. Якщо ці спроби безуспішні, то повідомлення про помилку передається клієнтському ПЗ для програмної обробки.

Системне конфігурування

USB підтримує підключення й відключення пристроїв під час роботи шини. Нумерація пристроїв шини є постійним процесом, що відслідковує динамічні зміни фізичної топології.

Всі пристрої USB підключаються через порти хабів. Хаби визначають підключення й відключення пристроїв до своїх портів і повідомляють стан портів у відповідь на запит від контролера. Хост дозволяє роботу порту й адресується до пристрою через канал керування використовуючи свою нульову адресу – USB default address. Всі пристрої адресуються цією адресою при початковому підключенні або після скидання.

Хост визначає, є підключений пристрій хабом або функцією й призначає йому унікальні адреса USB. Хост установлює із цим пристроєм канал керування використовуючи призначену адресу й нульовий номер точки призначення.

Якщо новий пристрій є хабом, хост визначає підключені до нього пристрою встановлює канали й призначає для них адреси. Якщо новий пристрій є "функцією" повідомлення про підключення віддається диспетчером USB відповідне ПЗ.

Коли пристрій відключається, хаб автоматично забороняє використання

відповідного порту й повідомляє про відключення контролеру, що видаляє відомості про даний пристрій із всіх структур даних. Якщо відключається хаб, то процес видалення повторюється для всіх підключених до нього пристроїв.

Нумерація пристроїв, підключених до шини, здійснюється динамічно в міру підключення або відключення їхнього живлення без будь-якого втручання користувача або клієнтського ПЗ. Процедура нумерації виконується в такий спосіб:

1. Хаб, до якого підключився пристрій, інформує хост про зміну стану свого порту відповіддю на опитування стану. Із цього моменту пристрій переходить у стан "Attached" ("приєднаний"), а порт, до якого воно приєднано, у стан "Disabled".

2. Хост уточнює стан порту.

3. Довідавшись порт, до якого підключився новий пристрій, хост дає команду скидання й дозволу порту.

4. Хаб формує сигнал RESET для даного порту (10 мс) і переводить його в стан "Enabled". Підключеному пристрою дозволяється споживати від шини струм живлення в межах 100 мА. Пристрій переходить у стан Powered, всі його регістри переводяться у вихідний стан, і воно озивається на обіг по нульовій адресі.

5. Доти поки пристрій не одержить унікальну адресу, вона доступна по черговому каналу, по якому хост-контролер може визначати максимально припустимий розмір поля даних пакета.

6. Хост повідомляє пристрій його унікальну адресу, і воно переходить у стан "Addressed".

7. Хост зчитує всі конфігурації пристрою, включаючи й заявлений струм споживання від шини.

8. Виходячи з отриманої інформації, хост конфігурує всі наявні кінцеві точки даного пристрою, що переводиться в стан Configured. Тепер хаб дозволяє пристрою споживати від шини повний струм, заявлений у конфігурації. З погляду

пристрою він стає готовим до використання.

Коли пристрій відділяється від шини, хаб повідомляє про це хост і робота порту забороняється, а хост обновляє свою поточну топологічну інформацію.

Хост-контролер

Хост-комп'ютер спілкується із пристроєм через контролер. Хост має наступне призначення:

– Виявлення підключення й від'єднання пристроїв USB.

– Маніпулювання потоком керування між пристроями й хостом.

– Керування потоками даних.

– Збір інформації про стан і статистики.

– Забезпечення енергозбереження підключеними пристроями.

Системне ПЗ контролера управляє взаємодією між пристроями і їхнім ПЗ, що функціонує на хост-комп'ютері. Області взаємодії наступні:

– Нумерація й конфігурація пристроїв.

– Ізохронні передачі даних.

– Асинхронні передачі даних.

– Керування енергоспоживанням.

– Інформація про керування пристроями й шиною.

По можливості, ПЗ USB у цих областях використовує існуюче системне ПЗ хост-комп'ютера.

Функції й хаби

Можливості шини USB дозволяють їй використовувати для підключення найрізноманітніших пристроїв. Всі пристрої повинні підтримувати нижчеперелічений набір загальних операцій.

Динамічне підключення й відключення має на увазі можливість цих дій у будь-який момент часу. Ці події відслідковуються хабом, що повідомляє про них хост-контролеру й виконує скидання підключеного пристрою. Пристрій після сигналу скидання повинен озиватися на нульову адресу, при цьому він не сконфігурований та не призупинений.

Після призначення адреси, за яке відповідає хост-контролер, пристрій повинен озиватися тільки на свою унікальну адресу.

Конфігурування пристроїв, виконуване хостом, є необхідним кроком для їхнього використання. Для конфігурації звичайно використовується інформація зчитана із самого пристрою. Пристрій може мати безліч інтерфейсів, кожному з них відповідає власна кінцева точка, що представляє хосту функцію пристрою. Крім того інтерфейс у конфігурації може мати альтернативні набори характеристик, зміна наборів підтримується протоколом. Для підтримки адаптивних драйверів дескриптори пристроїв і інтерфейсів мають поля класу, підкласу й протоколу.

Передача даних можлива за допомогою одного або чотирьох типів передач. Для кінцевих точок, що допускають різні типи передач, після конфігурування доступний тільки один з них.

Керування енергоспоживанням є досить розвиненою функцією USB. Для пристроїв, що живляться від шини потужність є обмеженим ресурсом. Будь-який пристрій при початковому підключенні не повинен споживати струм більше 100 мА. Робочий струм (не більше 500 мА) заявляється в конфігурації, і якщо хаб не зможе забезпечити пристрою заявлений струм, той не конфігурується.

Пристрій USB повинен підтримувати режим припинення, у якому його споживаний струм не повинен перевищувати 500 мкА. Пристрій повинен автоматично призупинятися при припиненні діяльності шини.

Можливість віддаленого пробудження дозволяє призупиненому пристрою подати сигнал хост-комп'ютеру, що теж може перебувати в призупиненому стані. Можливість віддаленого пробудження описується в конфігурації пристрою, і при конфігуруванні ця функція може бути й заборонена.

Хаб в USB виконує функції комутації сигналів і роздачі напруги живлення, а також відслідковує стан підключених до нього пристроїв, повідомляючи хост про зміни.

Хаб складається із двох складових – контролера й повторювача.

Повторювач являє собою керований ключ, що з'єднує вихідний порт із вхідним. Він також має засоби підтримки скидання й припинення передачі сигналів.

Спадаючі порти хабів можуть перебувати в наступних станах:

1. Power Off – стан, у якому на порт не подається живлення (можливий тільки для хабів, комутуючих живлення). Вихідні буфери переводяться у вісокоімпедансний стан, вхідні сигнали ігноруються.

2. Disconnected – порт не передає сигнали в жодному напрямку, але здатний виявити подію підключення пристрою. Виявивши підключення (по відсутності стану SE0 протягом 2.5 мкс), порт переходить у стан "Disabled", а по рівнях вхідних сигналів він визначає швидкість підключеного пристрою.

3. Disabled – порт може передавати тільки сигнал скидання (по команді від контролера), сигнали від порту, крім виявлення відключення, не сприймаються. При виявленні відключення порт переходить у стан "Disconnect", а якщо відключення виявлене "сплячим" хабом, контролеру буде посланий сигнал Resume.

4. Enabled – порт передає сигнали в обох напрямках. По команді контролера, або при виявленні помилки кадру порт переходить у стан "Disabled", а при виявленні відключення – у стан "Disconnect".

5. Suspended – порт передає сигнал переходу у стан призупинки. Якщо хаб перебуває в активному стані, сигнали через цей порт не пропускаються. Однак "сплячий" хаб сприймає сигнали зміни стану незаборонених портів, забезпечуючи можливість подачі "пробуджуючих" сигналів від пристрою, що активізувалося, навіть через ланцюжок "сплячих" хабів.

Стан кожного порту ідентифікується контролером хаба за допомогою окремих регістрів. Крім того, є загальний регістр, біти якого відбивають факт зміни стану кожного поту. Це дозволяє хост-контролеру швидко опитувати стан хаба, а у випадку виявлення змін спеціальними транзакціями уточнювати стан.

Регістри USB-інтерфейсу

USBCON (S:BCh) – основний керуючий регістр USB інтерфейсу. Формат цього регістра показаний на рисунку 3.5.

7	6	5	4	3	2	1	0
USBE	SUSPCLK	SDRMWUP	0	UPRSM	RMWUPE	CONFG	FADDEN

Рисунок 3.5 – Формат регістру USBCON

USBE – біт вмикання модуля USB. Установка біта вмикає USB-контролер. Скидання біта вимикає й скидає USB-контролер.

SUSPCLK – біт припинення синхронізації USB. Установка біта відключає вхід 48 МГц синхроімпульсів (Продовження детектування усе ще можливе). Скидання включає вхід 48 МГц синхроімпульсів.

SDRMWUP – біт передачі віддаленого пробудження. Встановлюється для виклику зовнішнього переривання USB контролера при віддаленому пробудженні. Резюме вихідного потоку передається тільки якщо біт RMWUPE встановлений, всі USB синхроімпульси активізовані й USB шина перебувала в стані припинення (SUSPEND) не менш 5 мс. Скидається програмно.

UPRSM – біт резюме вихідного потоку (тільки читання). Встановлюється апаратно після установки біта SDRMWUP якщо біт RMWUPE був також встановлений. Скидається апаратно після передачі резюме вихідного потоку.

RMWUPE – біт дозволу віддаленого пробудження. Встановлюється для дозволу запиту резюме вихідного потоку провідного пристрою. Скидається після відображення резюме вихідного потоку в RSMINPR. *Зауваження:* не встановлюйте цей біт якщо в провідного пристрою для приладу не встановлена функція DEVICE_REMOTE_WAKEUP.

CONFG – конфігураційний біт. Встановлюється після коректної обробки запиту SET_CONFIGURATION з ненульовим значенням. Скидається програмно

після одержання запиту SET_CONFIGURATION з нульовим значенням. Скидається апаратно при апаратному скиданні або після виявлення USB скидання на шині.

FADDEN – біт дозволу функції адресації. Встановлюється програмним забезпеченням приладу після успішного фазування статусу транзакції SET_ADDRESS. Надалі він не повинен скидатися програмно. Скидається апаратно при апаратному скиданні або після виявлення USB скидання на шині. Коли цей біт скинутий, використовується функція адресації за замовчуванням (0).

Після скидання регістр приймає значення 0000 0000b.

USBADDR (S:C6h) – регістр USB адреси. Формат цього регістра показаний на рисунку 3.6.

7	6	5	4	3	2	1	0
FEN	UADD6	UADD5	UADD4	UADD3	UADD2	UADD1	UADD0

Рисунок 3.6 – Формат регістру USBADDR

FEN – біт активізації функції. Встановлюється для активізації функції. Програмне забезпечення приладу встановить цей біт після прийому USB скидання й візьме участь у поточному конфігураційному процесі із установленою за замовчуванням адресою (FEN скинеться в 0).

UADD6:0 – біти USB адреси. Ці біти містять задану за замовчуванням адресу (0) після включення живлення або скидання USB шини. Запис їхнього стану відбудеться після прийняття програмним забезпеченням приладу запиту SET_ADDRESS.

Після скидання регістр приймає значення 0000 0000b.

USBINT (S:BDh) – регістр прапорів основних USB переривань. Формат

цього регістра показаний на рисунку 3.7.

7	6	5	4	3	2	1	0
0	0	WUPCPU	EORINT	SOFINT	0	0	SPIN T

Рисунок 3.7 – Формат регістру USBINT

WUPCPU – прапор переривання пробудження ЦП. Встановлюється апаратно коли контролер, що перебуває в режимі SUSPEND USB перезапускається сигналом non-idle USB шини (але не резюме вихідного потоку). Установка цього біта викликає USB переривання коли встановлений біт EWUPCPU у регістрі USBIEN. Скидається програмно після перемикання всіх USB синхроімпульсів.

EORINT – прапор переривання закінчення скидання. Встановлюється апаратно при виявленні USB контролером закінчення скидання. Установка цього біта викликає USB переривання коли встановлений біт EEORINT у регістрі USBIEN. Скидається програмно.

SOFINT – прапор переривання при виявленні початку кадру. Встановлюється апаратно після прийому USB пакета початку кадру (SOF). Установка цього біта викликає USB переривання коли встановлений біт ESOFINT у регістрі USBIEN. Скидається програмно.

SPINT – прапор переривання при припиненні. Встановлюється апаратно при виявленні USB припинення (шина не зайнята протягом трьох кадрових періодів: J стан протягом 3 мс). Установка цього біта викликає USB переривання коли встановлений біт ESPINT у регістрі USBIEN. Скидається програмно.

Після скидання регістр приймає значення 0000 0000b.

USBIEN (S:BEh) – регістр дозволів основних USB переривань.

7	6	5	4	3	2	1	0
0	0	EWUPCPU	EEORINT	ESOFINT	0	0	ESPINT

Рисунок 3.8 – Формат регістру USBIEN

Формат цього регістра показаний на рисунку 3.8.

EWUPCPU – біт дозволу переривання при пробудженні ЦП. Установка цього біта дозволяє переривання при пробудженні ЦП. Скидання біта забороняє переривання при пробудженні ЦП.

EEORINT – біт дозволу переривання по закінченню скидання. Установка цього біта дозволяє переривання по закінченню скидання. Скидання цього біта забороняє переривання по закінченню скидання.

ESOFINT – біт дозволу переривання при виявленні початку кадру. Установка цього біта дозволяє переривання при виявленні початку кадру. Скидання цього біта забороняє переривання при виявленні початку кадру.

ESPINT – біт дозволу переривання при виявленні припинення. Установка цього біта дозволяє переривання при виявленні припинення. Скидання цього біта забороняє переривання при виявленні припинення.

Після скидання регістр приймає значення 0000 0000b.

UEPNUM (S:C7h) – регістр номера USB кінцевої точки. Формат цього регістра показаний на рисунку 3.9.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	EPNUM1	EPNUM0

Рисунок 3.9 – Формат регістру UEPNUM

EPNUM1:0 – біти номера кінцевої точки. Задають номер кінцевої точки, до якої буде відбуватися звертання при зчитуванні й записі регістрів UEPSTAX,

UEPDATX, UBYCTLX або UEPCONX.

Після скидання регістр приймає значення 0000 0000b.

UEPCONX (S:D4h) – керуючий регістр кінцевої USB точки X (де X – номер, заданий у регістрі UEPNUM. Формат цього регістра показаний на рисунку 3.10.

7	6	5	4	3	2	1	0
EPEN	0	0	0	DTGL	EPDIR	EPTYPE1	EPTYPE0

Рисунок 3.10 – Формат регістру UEPCONX

EPEN – біт активізації кінцевої точки. Установка біта включає кінцеву точку відповідно до конфігурації приладу. Нульова кінцева точка завжди активізується після апаратного скидання або скидання USB шини й бере участь у конфігурації приладу. Скидання біта відключає кінцеву точку відповідно до конфігурації приладу.

DTGL – біт зміни статусу даних (тільки читання). Встановлюється апаратно при прийманні пакета DATA1. Скидається апаратно при прийманні пакета DATA0. *Зауваження:* Якщо новий пакет даних приймається до зміни стану біта з 0 на 1 або з 1 на 0, то можлива втрата пакета. Коли це відбувається в пакетної кінцевої точки, вбудоване програмне забезпечення приладу повинне припустити, що провідний пристрій повторив передачу правильно прийнятого пакета тому що провідний пристрій не прийняв правильного підтвердження (ACK), після цього програма повинна відмовитися від передачі нового пакета (Кінцева точка скидається до DATA0 тільки при конфігуруванні). В кінцевих точок, що є джерелом переривання, перемикання даних управляється також як і при використанні пакетних кінцевих точок. У керуючих кінцевих точок кожна транзакція SETUP починається з DATA0 і перемикання даних у цьому випадку використовується як і в пакетних кінцевих точок доти, поки не закінчиться стадія Data (при контролі запису посилки); стадія Status завершує передачу DATA1 (при

контролі зчитування посилки). Для ізохронних кінцевих точок вбудоване програмне забезпечення приладу буде відновлювати кожний новий пакет даних і може ігнорувати цей біт.

EPDIR – біт установки спрямованості кінцевих точок. Установка біта встановлює пакетні, ізохронні та кінцеві точки переривань в режим прийому. Скидання біта встановлює пакетні, ізохронні та кінцеві точки переривань в режим передачі. Біт не впливає на керуючі кінцеві точки.

EPTYPE1:0 – біти установки типу кінцевих точок. Ці біти дозволяють установити для кінцевої точки один з наступних типів (для нульової кінцевої точки завжди повинен бути встановлений "керуючий" тип):

0 0 – керуюча кінцева точка;

0 1 – ізохронна кінцева точка;

1 0 – пакетна кінцева точка;

1 1 – кінцева точка переривань.

Після скидання регістр приймає значення 0000 0000b.

UEPSTAX (S: CEh) – регістр керування й статусу кінцевої USB точки X. Формат цього регістра показаний на рисунку 3.11.

7	6	5	4	3	2	1	0
DIR	0	STALLRQ	TXRDY	STLCRC	RXSETUP	RXOUT	TXCMP

Рисунок 3.11 – Формат регістру UEPSTAX

DIR – біт керування напрямком кінцевої точки. Цей біт враховується тільки тоді, коли кінцевій точці привласнений тип контрольної. Повинен бути встановлений для стадії даних. Для інших випадків повинен бути скинутий. Зауваження: Цей біт повинен бути встановлений при RXSETUP перериванні до зміни стану будь-якого іншого біта. Також він визначає фазу статусу (IN для перевірки запису й OUT для контролю читання). Цей біт повинен скинутий для

стадії статусу контрольної вихідної транзакції.

STALLRQ – біт запиту зупинки встановлення зв'язку. Установка біта приведе до посилки відповіді STALL (зупинки) провідному пристрою для наступної установки зв'язку. У протилежному випадку цей біт повинен бути скинутий.

TXRDY – керуючий біт готовності передачі пакета. Біт повинен бути встановлений після запису пакета в FIFO буфер кінцевої точки для передачі IN даних. Дані повинні записуватися в FIFO буфер кінцевої точки тільки після скидання цього біта. Установка цього біта без запису даних в FIFO буфер приведе до посилки пакета нульової довжини, що рекомендується в загальному випадку й може знадобитися для позначення передачі в тих випадках, коли довжина останнього пакета даних дорівнює MaxPacketSize (наприклад, для контрольного зчитування передачі). Скидається апаратно відразу після посилки пакета ізохронної кінцевої точки або після одержання контрольної, пакетної чи кінцевої точки переривання підтвердження від провідного пристрою.

STLCRC – прапор переривання при припиненні посилки/Прапор переривання при виявленні CRC помилки.

Для контрольних, пакетних і кінцевих точок переривань: Встановлюється апаратно після посилки за допомогою STALLRQ запиту припинення встановлення зв'язку. Після цього відбудеться переривання кінцевої точки, якщо воно дозволено в регістрі UEPIEN. Скидається апаратно після прийому пакета SETUP (див. опис біта RXSETUP). Для ізохронних кінцевих точок: Встановлюється апаратно при виявленні помилки в прийнятих даних (CRC помилка в прийнятих даних). Після цього відбудеться переривання кінцевої точки якщо воно дозволено в регістрі UEPIEN. Скидається апаратно після прийому неушкоджених даних.

RXSETUP – прапор переривання при одержанні пакета SETUP. Встановлюється апаратно після одержання припустимого пакета SETUP від провідного пристрою. Після цього відбудеться переривання, якщо воно дозволено

в регістрі UEPIEN. Скидається програмно після зчитування SETUP даних з FIFO буфера кінцевої точки.

RXOUT – прапор переривання при прийнятті вихідних даних. Встановлюється апаратно після прийняття вихідного пакета. Після цього відбудеться переривання, якщо воно дозволено в регістрі UEPIEN і всі поточні вихідні пакети відхилені до скидання цього біта. Однак у керуючих кінцевих точках раніше прийнята SETUP транзакція може переписати вміст FIFO буфера кінцевої точки, навіть якщо був прийнятий пакет даних при встановленому цьому прапорі. Скидається програмно після зчитування вихідних даних з FIFO буфера кінцевої точки.

TXCMP – прапор переривання по закінченню передачі вхідних даних Встановлюється апаратно після передачі вхідного пакета ізохронною точкою або після одержання підтвердження прийому (ACK) від провідного пристрою контрольної, пакетної або кінцевої точки переривання. Після цього відбудеться переривання, якщо воно дозволено в регістрі UEPIEN. Скидається програмно перед наступною установкою біта TXRDY.

Після скидання регістр приймає значення 0000 0000b. **UEPRST** (S:D5h) – регістр скидання FIFO буферів кінцевих точок. Формат цього регістра показаний на рисунку 3.12.

7	6	5	4	3	2	1	0
0	0	0	0	EP3RST	EP2RST	EP1RST	EP0RST

Рисунок 3.12 – Формат регістру UEPRST

EP3RST – скидання FIFO буфера третьої кінцевої точки. Необхідно встановити й скинути для скидання FIFO буфера третьої кінцевої точки перед початком будь-якої операції до апаратного скидання або при одержанні скидання USB шини.

EP2RST – скидання FIFO буфера другої кінцевої точки. Необхідно встановити й скинути для скидання FIFO буфера другої кінцевої точки перед початком будь-якої операції до апаратного скидання або при одержанні скидання USB шини.

EP1RST – скидання FIFO буфера першої кінцевої точки. Необхідно встановити й скинути для скидання FIFO буфера першої кінцевої точки перед початком будь-якої операції до апаратного скидання або при одержанні скидання USB шини.

EP0RST – скидання FIFO буфера нульової кінцевої точки. Необхідно встановити й скинути для скидання FIFO буфера нульової кінцевої точки перед початком будь-якої операції до апаратного скидання або при одержанні скидання USB шини.

Після скидання регістр приймає значення 0000 0000b.

UEPINT (S:F8h) – (тільки читання) регістр переривань кінцевих USB точок. Формат цього регістра показаний на рисунку 3.13.

7	6	5	4	3	2	1	0
0	0	0	0	EP3INT	EP2INT	EP1INT	EP0INT

Рисунок 3.13 – Формат регістру UEPINT

EP3INT – прапор переривання від третьої кінцевої точки. Встановлюється апаратно після установки переривання в регістрі UEPSTAX і якщо переривання від третьої кінцевої точки дозволено в регістрі UEPIEN. Повинен бути скинутий програмно.

EP2INT – прапор переривання від другої кінцевої точки. Встановлюється апаратно після установки переривання в регістрі UEPSTAX і якщо переривання від другої кінцевої точки дозволено в регістрі UEPIEN. Повинен бути скинутий програмно.

EP1INT – прапор переривання від першої кінцевої точки. Встановлюється

апаратно після установки переривання в регістрі UEPSTAX і якщо переривання від першої кінцевої точки дозволено в регістрі UEPIEN. Повинен бути скинутий програмно.

EP0INT – прапор переривання від нульовий кінцевої точки. Встановлюється апаратно після установки переривання в регістрі UEPSTAX і якщо переривання від нульової кінцевої точки дозволено в регістрі UEPIEN. Повинен бути скинутий програмно.

Після скидання регістр приймає значення 0000 0000b.

UEPIEN (S:C2h) – регістр дозволів переривань кінцевих USB точок. Формат цього регістра показаний на рисунку 3.14.

7	6	5	4	3	2	1	0
0	0	0	0	EP3INTE	EP2INTE	EP1INTE	EP0INTE

Рисунок 3.14 – Формат регістру UEPIEN

EP3INTE – біт дозволу переривання третьої кінцевої точки. Установка дозволяє переривання від третьої кінцевої точки. Скидання забороняє переривання від третьої кінцевої точки.

EP2INTE – біт дозволу переривання другої кінцевої точки. Установка дозволяє переривання від другої кінцевої точки. Скидання забороняє переривання від другої кінцевої точки.

EP1INTE – біт дозволу переривання першої кінцевої точки. Установка дозволяє переривання від першої кінцевої точки. Скидання забороняє переривання від першої кінцевої точки.

EP0INTE – біт дозволу переривання нульової кінцевої точки. Установка дозволяє переривання від нульової кінцевої точки. Скидання забороняє переривання від нульової кінцевої точки.

Після скидання регістр приймає значення 0000 0000b.

UEPDATX (S:CFh) – регістр даних FIFO буфера кінцевої USB точки X (X– номер встановлений у регістрі UEPNUM). Формат цього регістра показаний на рисунку 3.15.

7	6	5	4	3	2	1	0
FADAT7	FADAT6	FADAT5	FADAT4	FADAT3	FADAT2	FADAT1	FADAT0

Рисунок 3.15 – Формат регістру UEPDATX

FADAT7:0 – дані FIFO буфера кінцевої точки X. Байт даних, записуваний в FIFO буфер або зчитувальний з FIFO буфера кінцевої точки X (див. регістр EPNUM).

Після скидання регістр приймає значення XXh.

UBYCTLX (S: E2h) – регістр лічильника байтів кінцевої USB точки X (X – номер встановлений у регістрі UEPNUM). Формат цього регістра показаний на рисунку 3.16.

7	6	5	4	3	2	1	0
0	BYCT6	BYCT5	BYCT4	BYCT3	BYCT2	BYCT1	BYCT0

Рисунок 3.16 – Формат регістру UBYCTLX

BYCT6:0 – лічильник байтів. Лічильник байтів прийнятих пакетів даних. Значення цього лічильника байтів дорівнює кількості байтів даних, прийнятих після одержання ідентифікатора (PID) даних.

Після скидання регістр приймає значення 0000 0000b.

UFNUML (S:BFh – тільки читання) – регістр молодших бітів номера USB кадру. Формат цього регістра показаний на рисунку 3.17.

7	6	5	4	3	2	1	0

FNUM7	FNUM6	FNUM5	FNUM4	FNUM3	FNUM2	FNUM1	FNUM0

Рисунок 3.17 – Формат регістру UFNUML

FNUM7:0 – номер кадру. Молодші 8 біт 11– бітного номера кадру.

Після скидання регістр приймає значення 00h.

UFNUMH (S:BBh – тільки читання) – регістр старших бітів номера USB кадру. Формат цього регістра показаний на рисунку 3.18.

7	6	5	4	3	2	1	0
0	0	CRCOK	CRCERR	0	FNUM10	FNUM9	FNUM8

Рисунок 3.18 – Формат регістру UFNUMH

CRCOK – Біт відсутності CRC помилки прийнятого номера кадру. Встановлюється апаратно після прийняття неушкодженого номера кадру в стартовому або кадровому пакеті. Обновляється після кожного прийняття стартового або кадрового пакета. Зауваження: Переривання при прийнятті початку кадру генерується відразу після одержання PID.

CRCERR – Біт наявності CRC помилки прийнятого номера кадру. Встановлюється апаратно після прийняття ушкодженого номера кадру в стартовому або кадровому пакеті. Обновляється після кожного прийняття стартового або кадрового пакета. Зауваження: Переривання при прийнятті початку кадру генерується відразу після одержання PID.

FNUM10:8 – Номер кадру. Старші 3 біти 11-бітного номера кадру. Вони доступні в останньому прийнятому SOF пакеті. Біти FNUM не змінюються, якщо прийнято ушкоджений SOF.

Після скидання регістр приймає значення 00h. **USBCLK** (S:EAh) – регістр дільника USB контролера синхронізації. Формат цього регістра показаний на рисунку 3.19.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	USBCD1	USBCD0

Рисунок 3.19 – Формат регістру USBCLK

USBCD1:0 – Дільник USB контролера синхронізації. 2-розрядний дільник для формування синхроімпульсів USB контролером синхронізації.

Після скидання регістр приймає значення 0000 0000b.

3.2 Розробка структурної схеми

На рисунку 3.20 зображена структурна схема апаратної частини USB-контролера.

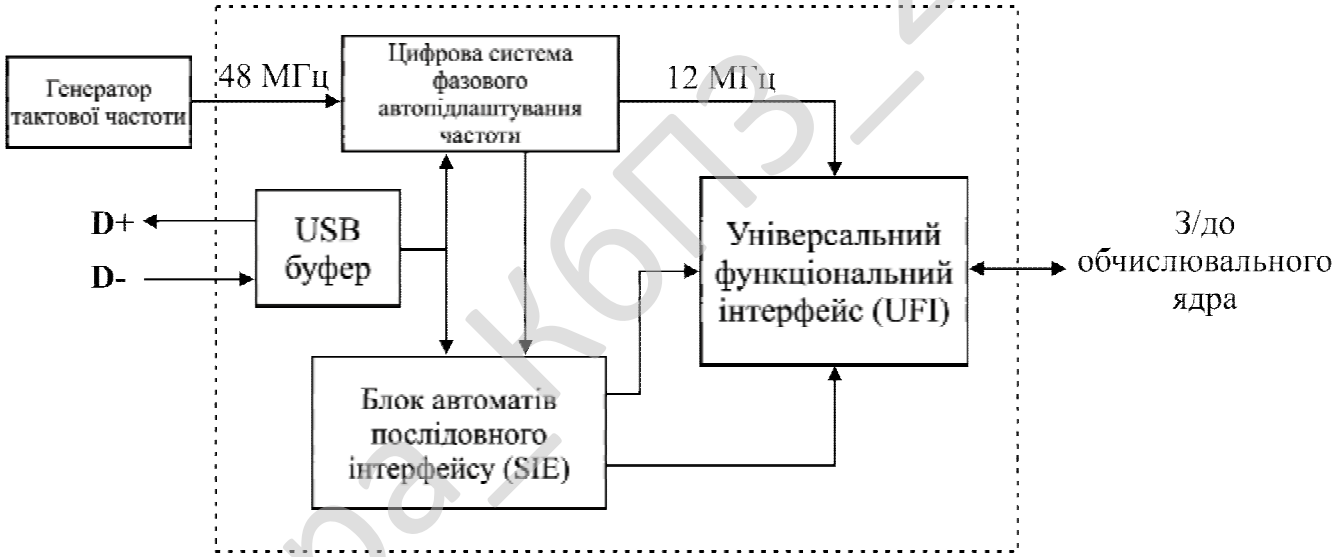

Рисунок 3.20 – Структурна схема апаратної частини USB-контролера

USB контролер містить апаратний модуль, що дозволяє забезпечувати обмін даними по USB інтерфейсу із двопортовою пам'яттю. Для цього необхідні опорні синхроімпульси із частотою 48 Мгц, які виробляються контролером синхронізації. Ці синхроімпульси використовуються для формування 12 Мгц тактових імпульсів із прийнятого диференціального USB потоку даних і передачі даних на високій швидкості, що відповідає вимогам до USB пристроїв. Відновлення синхроімпульсів виконується цифровою системою фазового автопідлаштування частоти (ФАПЧ).

Блок автоматів послідовного інтерфейсу (SIE) виконує NRZI кодування й декодування, вставку біта, формування бітів перевірки на парність (CRC кодування й декодування) і послідовно-паралельне перетворення даних.

Універсальний функціональний інтерфейс (UFI) контролює інтерфейс між потоком даних і двопортовою пам'яттю, а також інтерфейс безпосередньо з обчислювальним ядром.

Контролер синхронізації

USB контролер синхронізації тактується діленими синхроімпульсами системи ФАПЧ. Коефіцієнт розподілу задається бітами USBCD1:0 регістра в USBCLK. На рисунку 3.21 показаний USB контролер синхроімпульсів і формула для розрахунку коефіцієнта розподілу. Частота USB контролера синхронізації завжди повинна дорівнювати 48 Мгц.

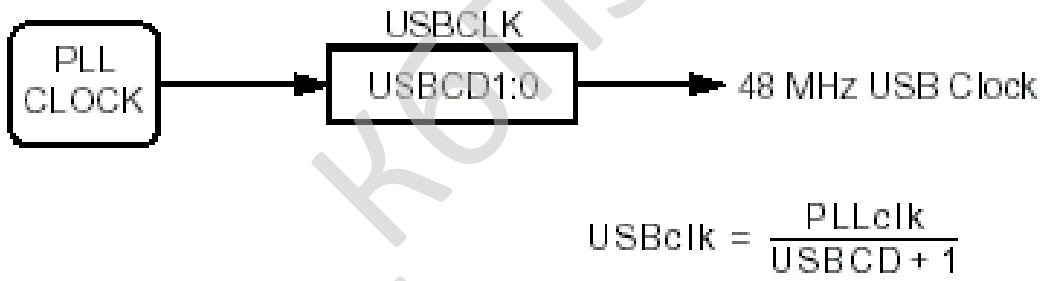

Рисунок 3.21 – Генератор тактової частоти для USB

Блок автоматів послідовного інтерфейсу (SIE)

SIE виконує наступні функції:

- NRZI кодування й декодування даних;
- Вставку й вилучення біта;
- CRC кодування й декодування;
- Автоматичне формування сигналів ACK і NACK;
- Ідентифікацію типу передавача;

– Контроль адрес;

– Відновлення синхроімпульсів (за допомогою DPLL).

–

Універсальний функціональний інтерфейс (UFI)

На рисунку 3.22 показана структурна схема універсального функціонального інтерфейсу.

Універсальний функціональний інтерфейс забезпечує інтерфейс між мікропроцесорами та та блоком автоматів послідовного інтерфейсу. Він управляє обміном на пакетному рівні з мінімальними програмними витратами, що виконують запис і зчитування FIFO буфера кінцевої точки.

Кінцеві точки є односпрямованими точками доступу для комунікації із пристроєм. Вони надають буфери для тимчасового зберігання одержаних та переданих від пристрою даних. Кожна кінцева точка в конфігурації має унікальну адресу, номер кінцевої точки та її напрямок. За замовчуванням завжди використовується кінцева точка 0, вона не є частиною ніякого інтерфейсу й доступна у всіх конфігураціях. Вона управляється рівнями послуг й безпосередньо не доступна драйверам пристроїв.

Нижченаведена конфігурація кінцевих точок:

– Кінцева точка 0 - 32 байта, прийом-передача команд керування.

– Кінцева точка 1 - 64 байта вихідні пакети.

– Кінцева точка 2 - 64 байта вхідні пакети.

– Кінцева точка 3 - 8 байтів вхідні переривання.

Регістри універсального функціонального інтерфейсу:

1. USBCON (S:BCh) – основний керуючий регістр USB інтерфейсу.
2. USBADDR (S:C6h) – регістр USB адреси.
3. USBINT (S:BDh) – регістр прапорів основних USB переривань.
4. USBIEN (S:BEh) – регістр дозволів основних USB переривань.
5. UEPNUM (S:C7h) – регістр номера USB кінцевої точки.
6. UEPCONX (S:D4h) – керуючий регістр кінцевої USB точки X (де X –

номер, заданий у регістрі UEPNUM.

7. UEPSTAX (S:CEh) – регістр керування й статусу кінцевої USB точки X.
8. UEPRST (S:D5h) – регістр скидання FIFO буферів кінцевих точок.
9. UEPINT (S:F8h) – регістр прапорів переривань кінцевих USB точок.
10. UEPIEN (S:C2h) – регістр дозволів переривань кінцевих USB точок.
11. UEPDATX (S:CFh) – регістр даних FIFO буфера кінцевої USB точки X.
12. UBYCTLX (S: E2h) – регістр лічильника байтів кінцевої USB точки X.
13. UFNUML (S:BFh – регістр молодших бітів номера USB кадру.
14. UFNUMH (S:BBh) – регістр старших бітів номера USB кадру.

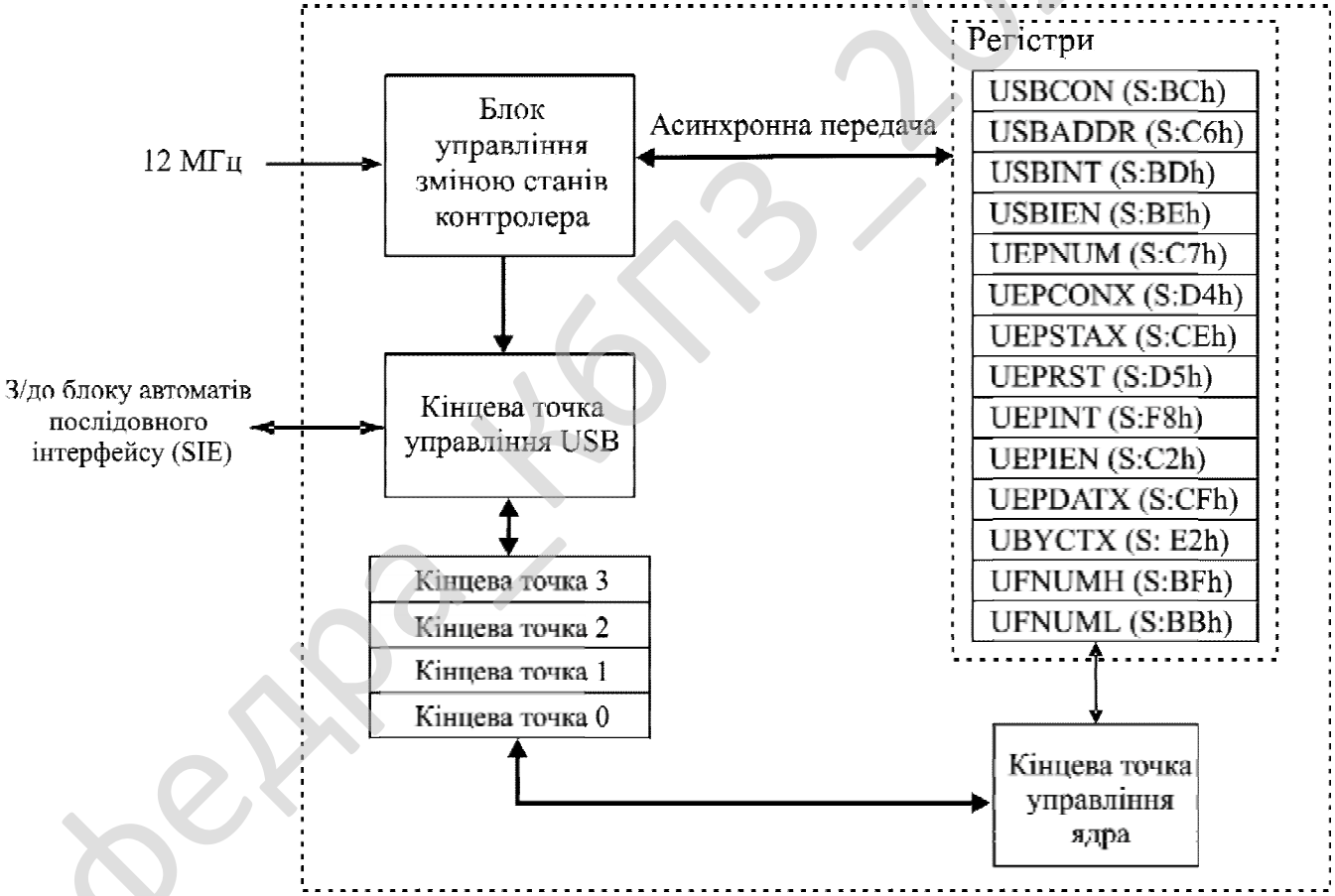

Рисунок 3.22 – Структурна схема універсального функціонального інтерфейсу (UFI)

USB система переривань

USB-контролер містить шістнадцять джерел переривання. Ці джерела переривання діляться на дві групи: переривання від кінцевих точок та переривання від контролера, об'єднані разом для того, щоб ядро могло оперувати одним перериванням.

Джерела переривання контролера

Є чотири джерела переривання контролера, які можна дозволити окремо установкою відповідних бітів у регістрі USBIEN:

– SPINT: прапор переривання при припиненні. Цей прапор викликає переривання при виявленні USB припинення (шина не зайнята протягом трьох кадрових періодів: J стан протягом 3 мс).

– SOFINT: прапор переривання при виявленні початку кадру. Цей прапор викликає переривання при виявленні початку кадру в прийнятому USB пакеті.

– EORINT: прапор переривання при виявленні закінчення скидання. Цей прапор викликає переривання при виявленні USB контролером закінчення скидання.

– WUPCPU: прапор переривання при виявленні пробудження ЦП. Цей прапор викликає переривання, коли USB контролер перебуває в режимі SUSPEND (припинення) і активізується сигналом non-idle від USB лінії.

3.3 Розробка функціональної схеми

USB забезпечує обмін даними між хост-комп'ютером і множиною одночасно доступних периферійних пристроїв. Розподіл пропускної здатності шини між підключеними пристроями планується хостом і реалізується їм за допомогою посилки маркерів. Шина дозволяє підключати, конфігурувати, використовувати й відключати пристрої під час роботи хоста й самих пристроїв – динамічне ("гаряче") підключення й відключення.

Пристрої (Device) USB можуть бути хабами, "функціями" або їхньою комбінацією.

Хаб (Hub) забезпечує додаткові точки підключення пристроїв до шини.

Функції (Function) USB надають системі додаткові можливості – наприклад підключення до ISDN, цифровий джойстик. акустичні колонки із цифровим інтерфейсом і т.д.

Пристрій USB повинен мати інтерфейс USB, що забезпечує підтримку протоколу USB, виконання стандартних операцій (конфігурування й скидання) і стандартне подання інформації, що описує пристрій. Багато пристроїв, що підключаються до USB, мають у своєму складі і функції, і хаби.

Роботою всієї системи USB управляє хост-контролер, що являється програмно-апаратною підсистемою хост-комп'ютера.

Фізичне з'єднання пристроїв здійснюється по топології багатоярусної зірки. Центром кожної зірки є хаб, кожний кабельний сегмент з'єднує дві точки – хаб з іншим хабом або хаб з функцією (рисунок 3.23).

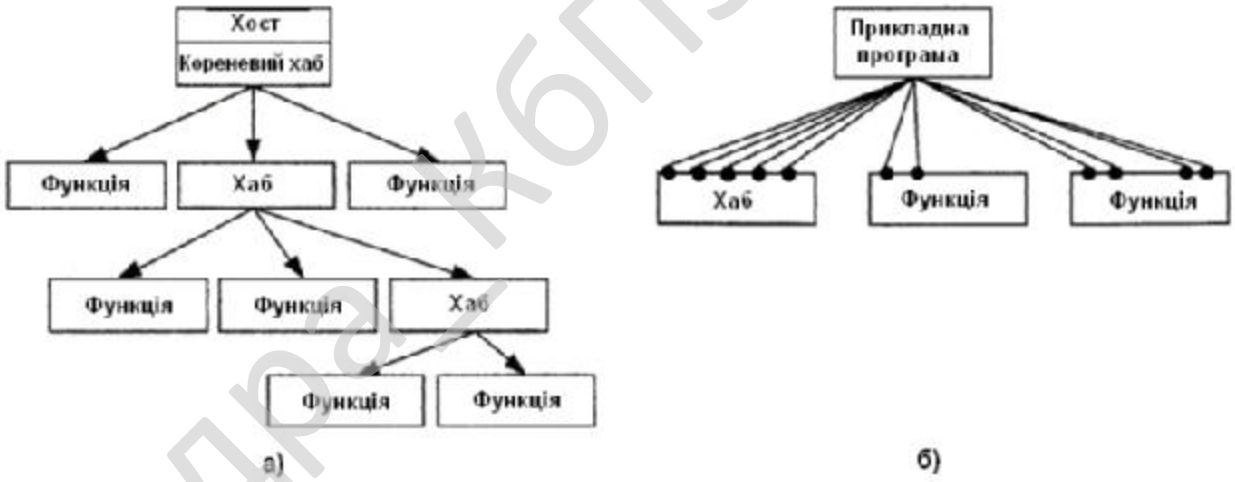

Рисунок 3.23 – Фізична (а) та логічна (б) архітектура USB

У системі USB є тільки один хост-контролер, розташований у вершині піраміди пристроїв і хабів USB. Хост-контролер інтегрується з кореневим хабом (root hub), що забезпечує одну або кілька точок підключення – портів. Контролер USB, що входить до складу чипсетів багатьох сучасних системних плат звичайно має двопортовий хаб.

Логічний пристрій підключений до будь-якого хабу й сконфігурований може розглядатися як підключений прямо до хост-контролеру.

Функції являють собою пристрої USB, здатні приймати або передавати дані або керуючу інформацію із шини. Фізично в одному корпусі може бути декілька функцій з вбудованим хабом, що забезпечує їхнє підключення до одного порту.

Кожна функція надає конфігураційну інформацію, що описує його можливості й вимоги до ресурсів. Перед використанням функція повинна бути сконфігурована хостом – їй повинна бути виділена смуга в каналі, обрані специфічні опції конфігурації.

Хаб – ключовий елемент системи Plug-and-Play в архітектурі USB. Хаб є кабельним концентратором, точки підключення називаються портами хаба. Кожний хаб перетворить одну точку підключення в їхню множину. Архітектура має на увазі можливість з'єднання декількох хабів.

У кожного хаба є один висхідний порт (upstream port), призначений для підключенню до хосту чи до хабу верхнього рівня. Інші порти є спадними (downstream) і призначені для підключення функцій і хабів нижнього рівня. Хаб може розпізнати підключення або відключення пристроїв до цих портів і управляти подачею живлення на їхні сегменти. Кожний із цих портів індивідуально може бути дозволений або заборонений і сконфігурований на повну або обмежену швидкість обміну. Хаб забезпечує ізоляцію сегментів з низькою швидкістю від високошвидкісних.

Хаби можуть мати можливість керування подачею живлення на спадні порти, передбачена керована установка обмеження на струм, споживаний кожним портом.

Система USB розділяється на три рівні з певними правилами взаємодії. Пристрій USB ділиться на інтерфейсну частину, частину пристрою й функціональну частину.

Хост теж ділиться на три частини – інтерфейсну, системну й ПЗ пристрою.

Кожна частина відповідає тільки за певне коло завдань, взаємодія між ними показана на рисунку 3.24.

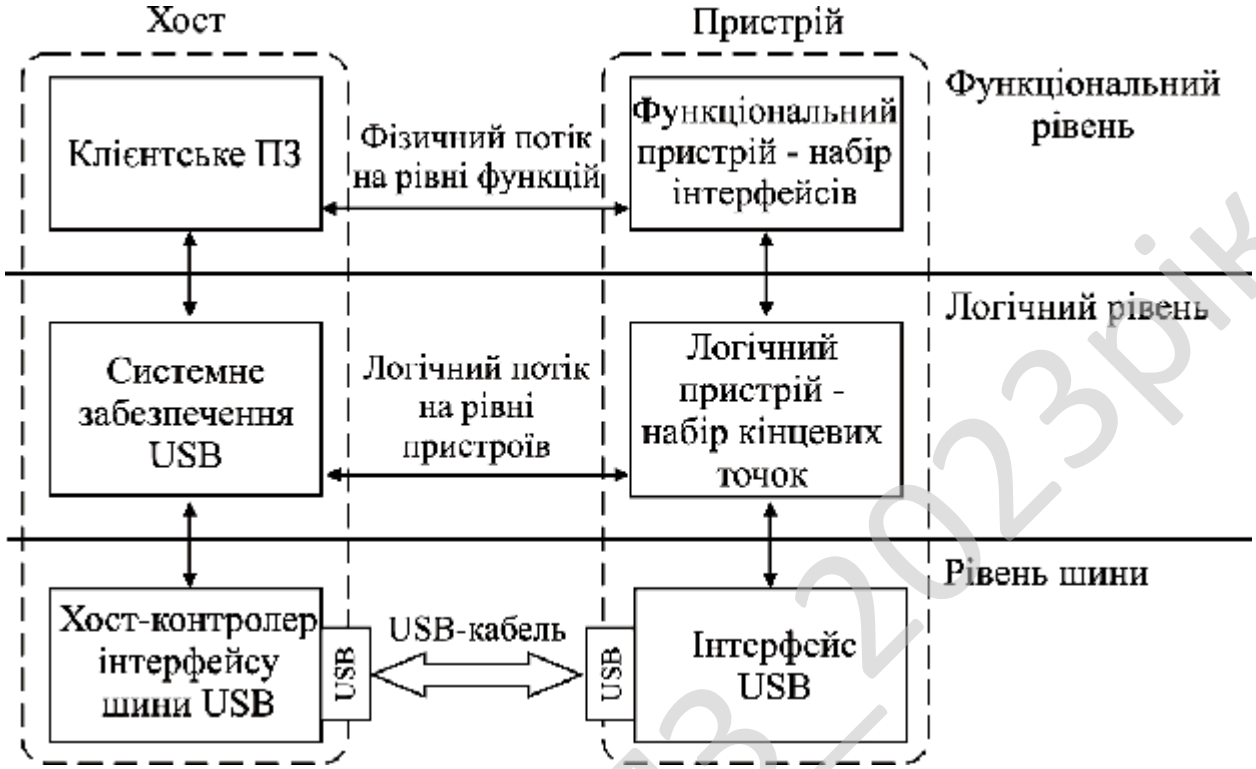

Рисунок 3.24 – Функціональна схема USB-інтерфейсу

Фізичний пристрій USB – пристрій на шині, що виконує функції, які цікавлять користувача.

Клієнтське ПЗ – програмне забезпечення, що відповідає конкретному пристрою і виконується на хост-комп'ютері. Може бути складовою частиною ОС або спеціальним продуктом.

Системне забезпечення USB пристрою – системна підтримка USB операційною системою, незалежна від конкретних пристроїв і клієнтського ПЗ.

Хост-контролер інтерфейсу шини USB – апаратні й програмні засоби, що забезпечують підключення пристроїв USB до хост-комп'ютера.

Розглянемо функціональну схему розробленої системи, зображену на рисунку 3.25.

Як видно з рисунку, розроблена система складається з трьох блоків:

– Емуляція кінцевих точок USB-пристроїв.

– Емуляція регістрів USB-інтерфейсу.

– Інтерфейс користувача.

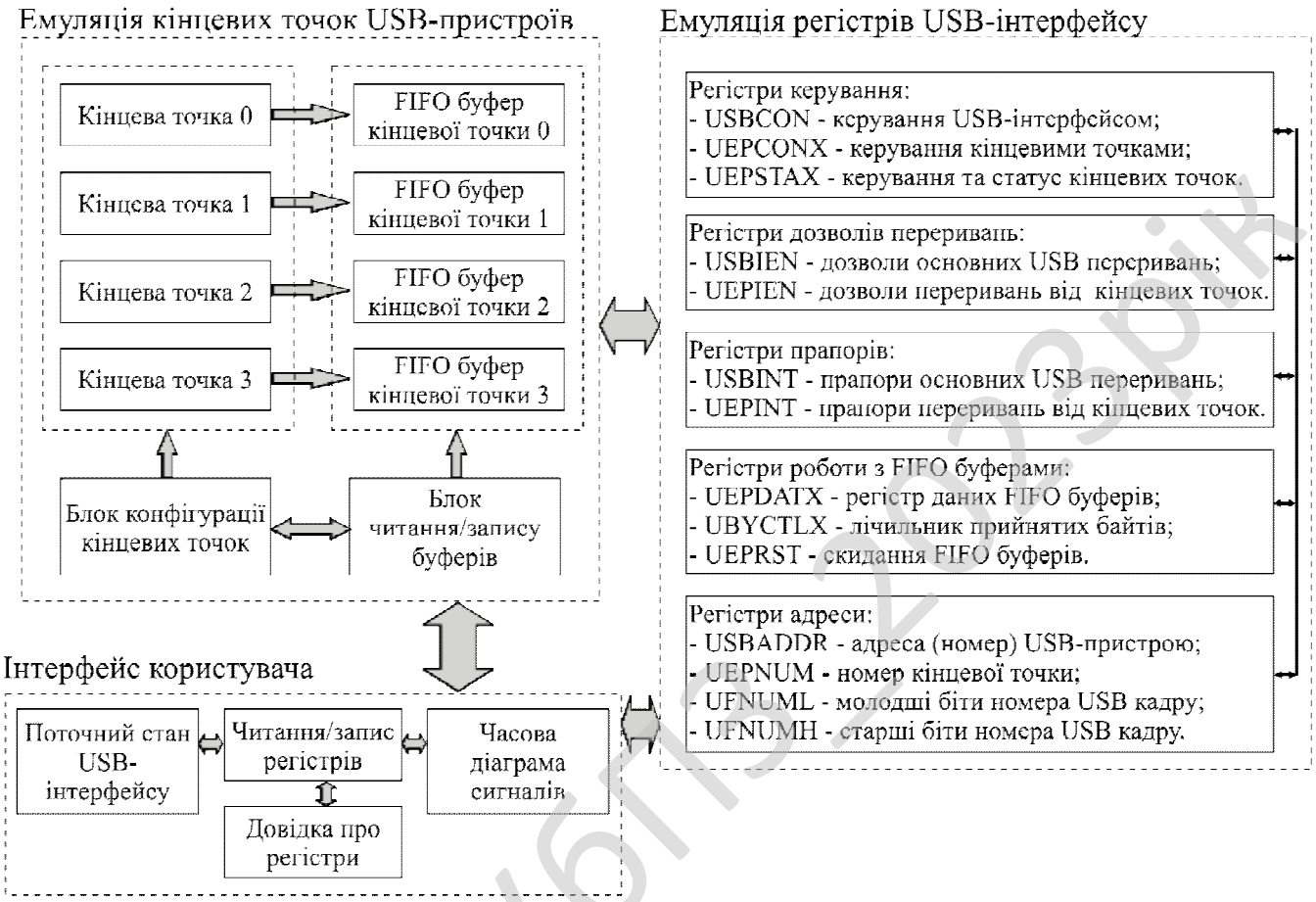

Рисунок 3.25 – Функціональна схема системи

Блок емуляції кінцевих точок USB-пристроїв емалює роботу чотирьох кінцевих точок 0-3 та їх FIFO буферів, дозволяє здійснювати активацію та конфігурування кінцевих точок, читати та записувати дані у FIFO буфери.

Блок емуляції регістрів USB-інтерфейсу дозволяє переглядати вміст та записувати нові значення у регістри інтерфейсу. Усі регістри розділено по їх призначенню на наступні групи: регістри керування, регістри дозволів переривань, регістри прапорів, регістри роботи з FIFO буферами та регістри адреси.

Блок інтерфейсу користувача візуалізує процес роботи і системного програмування передачі даних через USB. Він містить в собі наступні складові:

поточний стан USB-інтерфейсу, читання/запис регістрів, часова діаграма сигналів та довідка про регістри.

3.3 Розробка діаграми процесів

Розглянемо діаграму процесів розробленої системи, що зображена на рисунку 3.26.

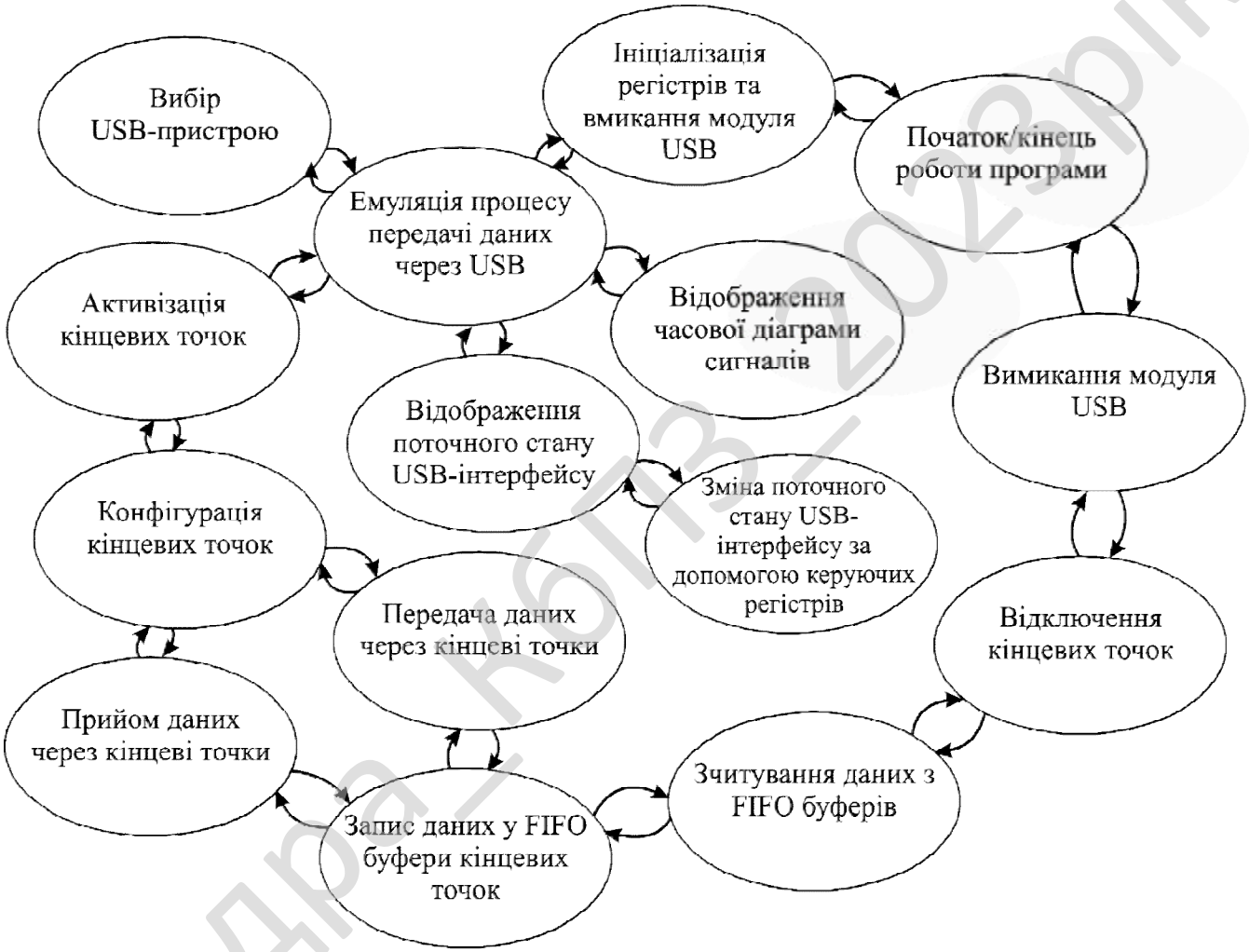

Рисунок 3.26 – Діаграма процесів системи

Після ввімкнення системи першим відбувається процес ініціалізації регістрів та вмикання USB модуля. Після цього здійснюється запуск процесу емуляції передачі даних через USB. Емуляція передачі даних через USB запускає наступні процеси:

– Вибір USB-пристрою.

– Відображення поточного стану USB-інтерфейсу.

– Відображення часової діаграми процесів.

– Активізація кінцевих точок.

Вибір USB-пристрою здійснюється шляхом вказання його адреси у регістрі USBADDR.

Активізація кінцевих точок здійснюється для створення каналів зв'язку між USB-пристроєм та хостом.

Після активізації кінцевої точки необхідно здійснити її конфігурування. Конфігурація кінцевих точок включає в себе вказання типу кінцевої точки (керуюча, пакетна, ізохронна чи переривання) та її спрямування (прийом чи передача).

Після вказання конфігурації активованих кінцевих точок, по ним (в залежності від їх типу) можна здійснювати прийом та передачу даних.

Прийняті та передані дані перед тим як поступити до пристрою, якому вони призначені, потрапляють у FIFO буфери кінцевих точок. Як тільки пристрій готовий прийняти дані, він починає зчитувати їх з буферів кінцевих точок.

При припиненні роботи з USB-інтерфейсом відбувається відключення кінцевих точок та вимикання модуля USB.

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.

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 зображена блок-схема роботи основної програми. З цієї блок-схеми ми бачимо, що розроблений у результаті виконання бакалаврської роботи програмний продукт працює наступним чином.

Спершу відбувається виведення основного вікна програми, на якому зображені усі регістри USB.

Після цього відбувається виклик функції ініціалізації USB-інтерфейсу, блок-схема роботи якої зображена на рисунку 4.2.

За ініціалізацією відбувається виведення поточного стану USB-інтерфейсу.

В залежності від завдання, яке стоїть перед користувачем програми. Відбувається зміна поточного стану USB-інтерфейсу, або ні.

Якщо відбувається зміна поточного стану USB-інтерфейсу то у регістри керування записуються нові значення.

Після цього стан USB-інтерфейсу змінюється згідно заданих регістрів.

Якщо ж поточний стан USB-інтерфейсу не треба змінювати, то переходимо до введення адреси пристрою у регістр адреси USBADDR.

Наведемо функцію запису адреси пристрою у регістр USBADDR.

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
int b0,b1,b2,b3,b4,b5,b6;
int r;
if(USBADDR0->Checked==true) b0=1; else b0=0;
if(USBADDR1->Checked==true) b1=1; else b1=0;
if(USBADDR2->Checked==true) b2=1; else b2=0;
if(USBADDR3->Checked==true) b3=1; else b3=0;
```

```
if(USBADDR4->Checked==true) b4=1; else b4=0;
if(USBADDR5->Checked==true) b5=1; else b5=0;
if(USBADDR6->Checked==true) b6=1; else b6=0;
r=b6*64 + b5*32 + b4*16 + b3*8 + b2*4 + b1*2 + b0*1;
NPr->Caption=IntToStr(r);
}
```

Після введення адреси пристрою у регістр адреси USBADDR користувач програми визначає потрібно передавати дані по USB-інтерфейсу, або ні.

Якщо передача потрібна то переходимо до функції активізації кінцевих точок пристрою.

Якщо ж передача даних не потрібна то переходимо до вибору приймати дані з USB-інтерфейсу або ні.

У випадку активізації кінцевих точок пристрою (блок-схема цієї функції зображена на рисунку 4.3), після вибору передачі даних по USB-інтерфейсу выдбувається введення даних для передачі у регістр UEPDATX.

Після цього відбувається відправка даних у FIFO буфер.

За відправкою даних у FIFO буфер користувач робить вибір про те працювати йому далі з програмою, або ні.

Повернемося до вибору користувачем процедури прийому даних до USB-інтерфейсу.

Якщо користувач вважає, що приймати дані не потрібно, то відбувається перехід до операції введення адреси пристрою у регістр адреси USBADDR.

Якщо ж користувач вважає, що приймо даних по USB-інтерфейсу потрібен то відбувається виклик функції активізації кінцевих точок пристрою, яка зображена на рисунку 4.3.

Після виконання функції активізації кінцевих точок відбувається виконання функції зчитування даних з FIFO буферу.

Коли відбувається операція зчитування данних з FIFO буфера то ці прийняті данні виводять на екран для того, що користувач, міг візуалізувати прцесс отримання даних по USB-інтерфейсу.

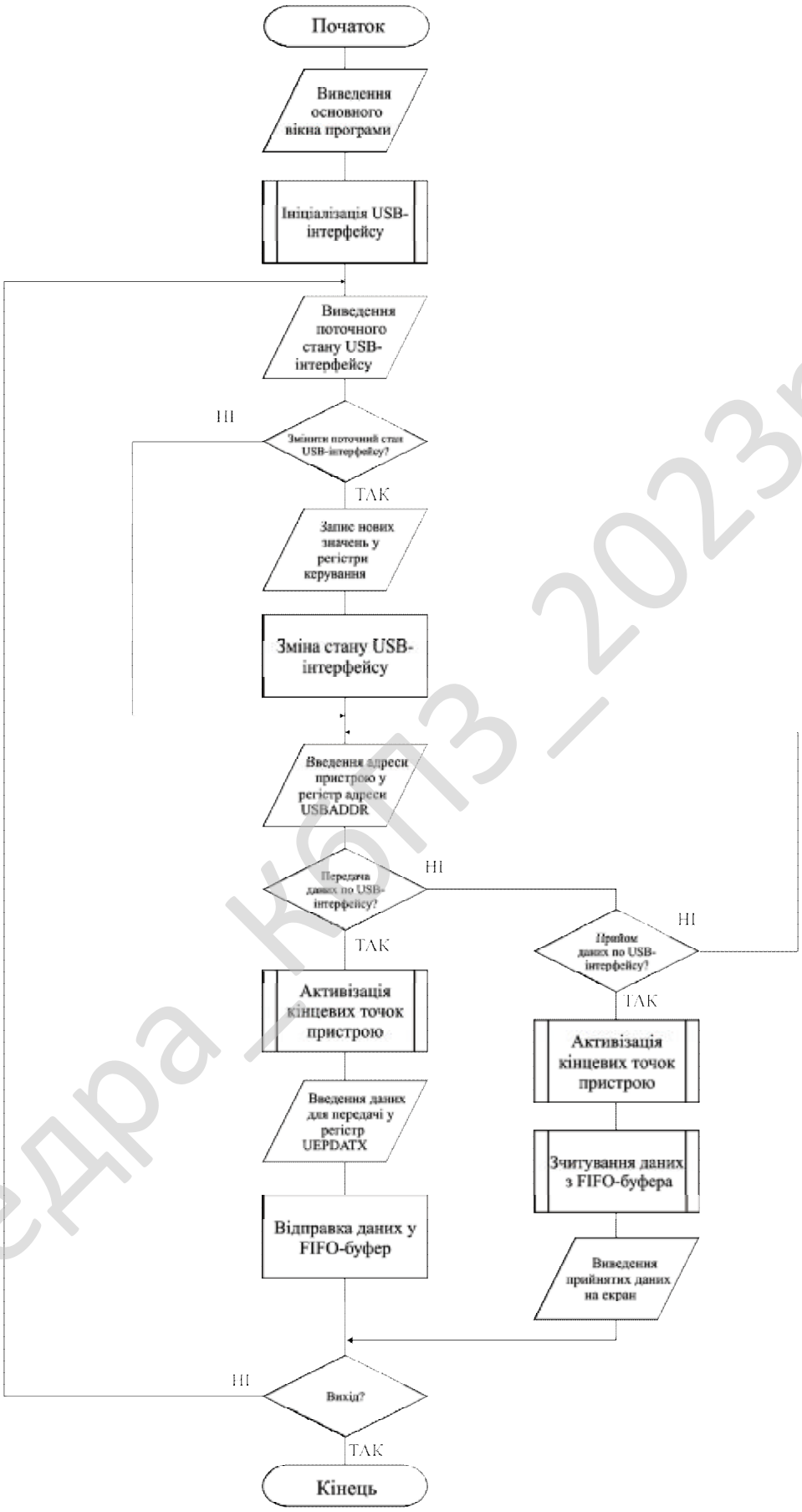

Рисунок 4.1 – Блок-схема роботи основної програми

Після того, як дані виведені на екран, то користувач також переходить до вибору, працювати йому далі з програмою, або ні.

Якщо він вирішує закінчувати роботу з програмою, то природно, що відбувається вихід з програми.

У противному ж випадку, користувач переходить на початок програми до операції виведення поточного стану USB-інтерфейсу.

На рисунку 4.2 зображена блок-схема роботи підпрограми ініціалізації USB-інтерфейсу.

З неї ми бачимо, що операції ініціалізації виконуються у наступній послідовності.

Спершу вмикається живлення мікроконтролера.

Після вмикання живлення мікроконтролера відбувається скидання сигналу POR, та відповідно запускається ядро.

Після запуску ядра заповнюється користувальницький дескриптор.

За цією операцією відбувається задання конфігурації кінцевою точки 0.

Коли конфігурація задана, то задається режим роботи блоку USB-та надається дозвіл на його роботу.

Після цього відбувається запит дескриптора хост-контролером.

Мікроконтролер видає дескриптор IN.

За цим передається нульовий пакет від хост-контролера OUT.

Після цього хост-контролер перезавантажується.

Після цього задається функціональна адреса хост-контролера.

За цим передається нульовий пакет від хост-контролера OUT.

Після цього відбувається завдання конфігурації та робота з USB-інтерфейсом.

Виконання усих вищеперерахованих дій приводить до ініціалізації USB-інтерфейсу.

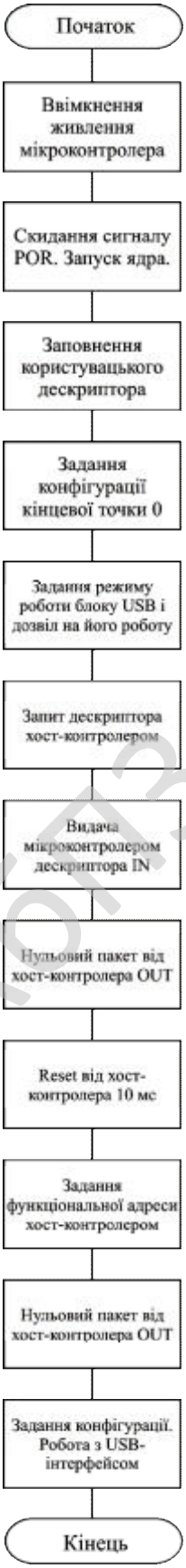

Рисунок 4.2 – Блок-схема роботи підпрограми ініціалізації USB-інтерфейсу

На рисунку 4.3 наведена блок-схема роботи підпрограми активації

кінцевих точок.

Послідовність активації кінцевих точок є наступною.

Спершу відбувається вибір кінцевої точки.

За її вибором відбувається активізація кінцевої точки.

Задається конфігурація кінцевої очки.

Відбувається перевірка правильності конфігурації.

Якщо конфігурація правильна то відбувається успішне створення кінцевої точки.

У противному випадку відбувається виведення повідомлення про помилку.

Після цього вважається, що кінцеві точки активовано, й функція повертається до основної програми.

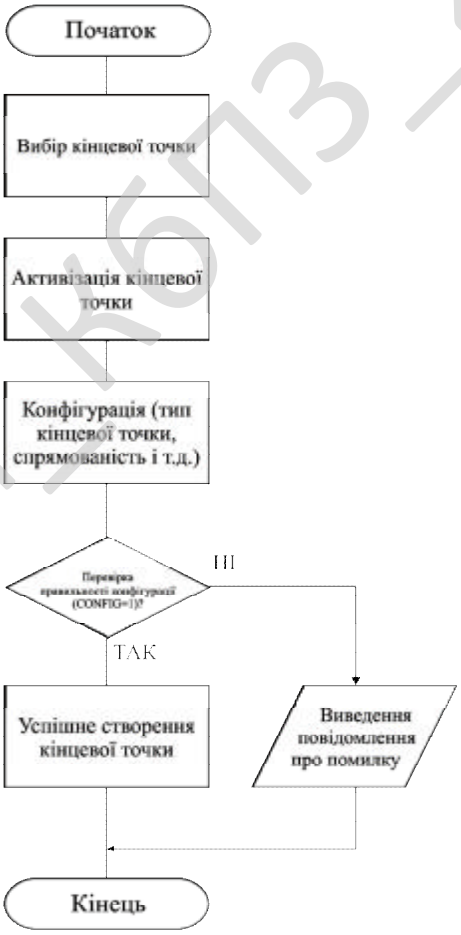

Рисунок 4.3 – Блок-схема роботи підпрограми активізації кінцевих точок

Функція роботи з кінцевою точкою наведена нижче.

```
//запис номеру кінцевої точки у регістр UEPNUM
void __fastcall TForm1::Button2Click(TObject *Sender)
{
int b0,b1;
if(NPr->Caption!='0') {
//визначення номеру кінцевої точки, записаного у регістр UEPNUM
if(UEPNUM0->Checked==true) b0=1; else b0=0;
if(UEPNUM1->Checked==true) b1=1; else b1=0;
x=b1*2 + b0*1;
nt->Caption=IntToStr(x);
//виведення стрілки біля вибраної кінцевої точки та
//блокування/розблокування кнопок в залежності від конфігурації
//вибраної кінцевої точки
if(x==0){
Image1->Visible=true;
Image2->Visible=false;
Image3->Visible=false;
Image4->Visible=false;
if(nn0->Caption=="Прийом/передача"){
BitBtn3->Enabled=true;
Button3->Enabled=true;
}
if(nn0->Caption=="Прийом"){
BitBtn3->Enabled=false;
Button3->Enabled=true;
}
if(nn0->Caption=="Передача"){
BitBtn3->Enabled=true;
Button3->Enabled=false;
}
if(nn0->Caption=="-"){
BitBtn3->Enabled=false;
Button3->Enabled=false;
}
}
if(x==1){
Image1->Visible=false;
Image2->Visible=true;
Image3->Visible=false;
Image4->Visible=false;
```

```
if(nn1->Caption=="Прийом/передача"){
BitBtn3->Enabled=true;
Button3->Enabled=true;
}
if(nn1->Caption=="Прийом"){
BitBtn3->Enabled=false;
Button3->Enabled=true;
}
if(nn1->Caption=="Передача"){
BitBtn3->Enabled=true;
Button3->Enabled=false;
}
if(nn1->Caption=="-"){
BitBtn3->Enabled=false;
Button3->Enabled=false;
}
}
if(x==2){
Image1->Visible=false;
Image2->Visible=false;
Image3->Visible=true;
Image4->Visible=false;
if(nn2->Caption=="Прийом/передача"){
BitBtn3->Enabled=true;
Button3->Enabled=true;
}
if(nn2->Caption=="Прийом"){
BitBtn3->Enabled=false;
Button3->Enabled=true;
}
if(nn2->Caption=="Передача"){
BitBtn3->Enabled=true;
Button3->Enabled=false;
}
if(nn2->Caption=="-"){
BitBtn3->Enabled=false;
Button3->Enabled=false;
}
}
if(x==3){
Image1->Visible=false;
Image2->Visible=false;
```

```
Image3->Visible=false;
Image4->Visible=true;
if(nn3->Caption=="Прийом/передача"){
BitBtn3->Enabled=true;
Button3->Enabled=true;
}
if(nn3->Caption=="Прийом"){
BitBtn3->Enabled=false;
Button3->Enabled=true;
}
if(nn3->Caption=="Передача"){
BitBtn3->Enabled=true;
Button3->Enabled=false;
}
if(nn3->Caption=="-"){
BitBtn3->Enabled=false;
Button3->Enabled=false;
}
}
}
else MessageDlg("Вкажіть адресу пристрою!", mtError, TMsgDlgButtons() <<
mbOK, 0); //виведення повідомлення про помилку, якщо користувач
намагається вказати кінцеву точку не вказавши пристрій
```

На рисунку 4.4 наведена блок-схема роботи підпрограми зчитування даних з FIFO-буферів кінцевих точок.

При її виконанні виконується наступна послідовність дій.

Спершу відбувається вибір кінцевої точки.

Після цього ініціалізується дескриптор USB.

Коли дескриптор ініціалізований, то перевіряється наявниість даних у черзі.

Якщо даних немає, то ми чекаємо, коли вони з’являться. У іншому випадку, тобто коли перевіка показала, що у черзі є дані, відбувається читання даних з буфера.

Ці дані записуються у регістр UEPDATX.

Після цього вони видаляються з буфера.

З цією операцією відбувається читання даних з регістру UEPDATX.

Читанням даних з регістру UEPDATX закінчується робота функції зчитування даних з FIFO-буферів кінцевих точок.

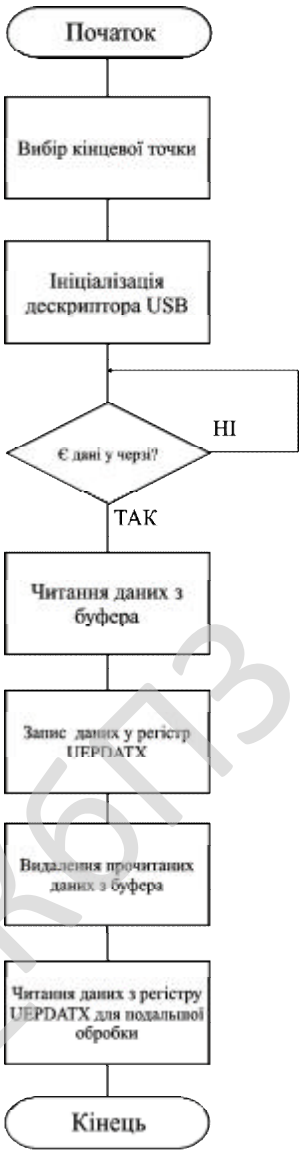

Рисунок 4.4 – Блок-схема роботи підпрограми зчитування даних з FIFO-буферів кінцевих точок

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою наступного алгоритму захисту інформації DSA. Відправник і одержувач електронного

документа використовують при обчисленні великі цілі числа: G і P – прості числа, L біт кожне ($512 < L < 1024$); q – простої число довжиною 160 біт (дільник числа (P-1). Числа G, P, q є відкритими й можуть бути загальними для всіх користувачів мережі.

Відправник вибирає випадкове ціле число X, $1 < X < q$. Число X є секретним ключем відправника для формування електронного цифрового підпису. Потім відправник обчислює значення $Y = G^X \bmod P$. Число Y є відкритим ключем для перевірки підпису відправника. Число Y передається всім одержувачам документів.

Цей алгоритм також передбачає використання однобічної функції гешування h(-). У стандарті DSS визначений алгоритм безпечного гешування SHA. Для того щоб підписати документ M, відправник гешує його в ціле геш-значення m: $m = h(M)$, $1 < m < q$, потім генерує випадкове ціле число K, $1 < K < q$, і обчислює число r: $r = (G^K \bmod P) \bmod q$. Потім відправник обчислює за допомогою секретного ключа X ціле число s:

$$s = \frac{m + r * X}{K} \bmod q \ .$$

Пара чисел r і s утворить цифровий підпис $S = (r, s)$ під документом M. Таким чином, підписане повідомлення являє собою трійку чисел $[M,r,s]$. Одержувач підписаного повідомлення $[M, r, s]$ перевіряє виконання умов $0 < r < q$, $0 < s < q$ і відкидає підпис, якщо хоча б одна із цих умов не виконана. Потім одержувач обчислює значення $w = 1/s \bmod q$, геш-значення $m = h(M)$ і числа $u_1 = (m*w) \bmod q$, $u_2 = (r*w) \bmod q$. Далі одержувач за допомогою відкритого ключа Y обчислює значення $v = ((G^{U1} * Y^{U2}) \bmod P) \bmod q$ і перевіряє виконання умови $v = r$. Якщо умова $v = r$ виконується, тоді підпис $S = (r,s)$ під документом M визнається одержувачем справжнім.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблена програма емулює процес роботи і системного програмування передачі даних через USB. Робота програми по обміну даними через USB-інтерфейс, зводиться до роботи з буферами FIFO кінцевих точок. Робота мікропроцесорного ядра із чергами може відбуватися одночасно із прийомом або передачею пакетів USB.

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

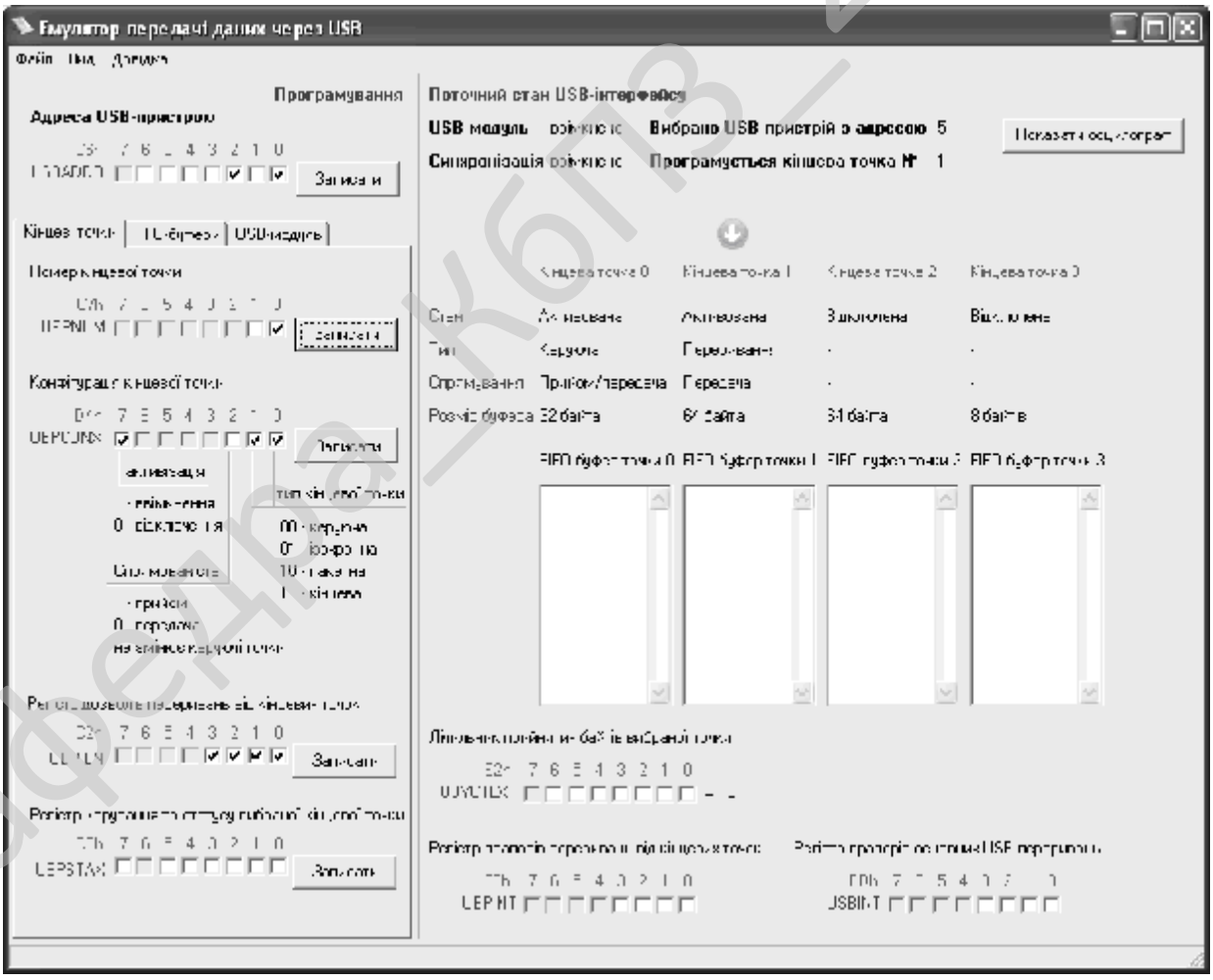

Рисунок 5.1 – Головне вікно програми (активізація кінцевих точок)

У лівій частині вікна розміщені регістри, які доступні для запису та дозволяють програмувати USB-інтерфейс.

У правій частині вікна відображається поточний стан USB-інтерфейсу, конфігурація кінцевих точок, вміст FIFO-буферів та вміст регістрів доступних тільки для читання.

Після запуску програми слід записати адресу пристою у регістр USBADDR та активізувати кінцеві точки для роботи з ним.

Активізація та конфігурування кінцевих точок відбувається на вкладці «Кінцеві точки» (рисунок 5.1).

Кінцева точка 0 активізується автоматично після вмикання USB-модуля. Її неможна вимикати програмно, так як ця точка необхідна для нормальної роботи USB-інтерфейсу. Інші кінцеві точки активізуються програмно.

Для програмування кінцевої точки спочатку необхідно вказати її номер у регістрі UEPNUM. Потім здійснити активацію та конфігурування за допомогою регістру UEPCONX.

Кінцеві точки можуть бути чотирьох типів:

– Керуючі.

– Ізохронні.

– Пакетні.

– Кінцеві.

Керуючі кінцеві точки завжди працюють у режимі «Прийом/передача».

Кінцева точка будь-якого іншого типу може працювати або у режимі «Прийом», або у режимі «Передача», в залежності від того, який режим був вказаний при її конфігуруванні.

Вкладка «Кінцеві точки» також дозволяє встановлювати дозволи переривань від кінцевих точок (регістр UEPIEN) та керувати роботою кінцевих точок (регістр UEPSTAX).

Після того, як активовані необхідні кінцеві точки, можна здійснити передачу та прийом даних.

Усі дані перед прийняттям чи передачею, потрапляють у FIFO-буфери кінцевих точок, з яких їх потім зчитує пристрій чи хост, в залежності від місця призначення.

Для роботи з FIFO-буферами слід перейти на вкладку «FIFO-буфери» (рисунок 5.2). Дана вкладка містить регістр даних вибраної кінцевої точки UEPDATX (під вибраною кінцевою точкою мається на увазі та точка, номер якої міститься у регістрі UEPNUM в даний час) та регістр скидання буферів кінцевих точок UEPRST. Регістр UEPDATX дозволяє здійснювати чинання та/або запис байту даних з FIFO-буфера вказаної кінцевої точки. Регістр UEPRST дозволяє скидати буфери заданих у ньому кінцевих точок (одразу всіх, або вибірково).

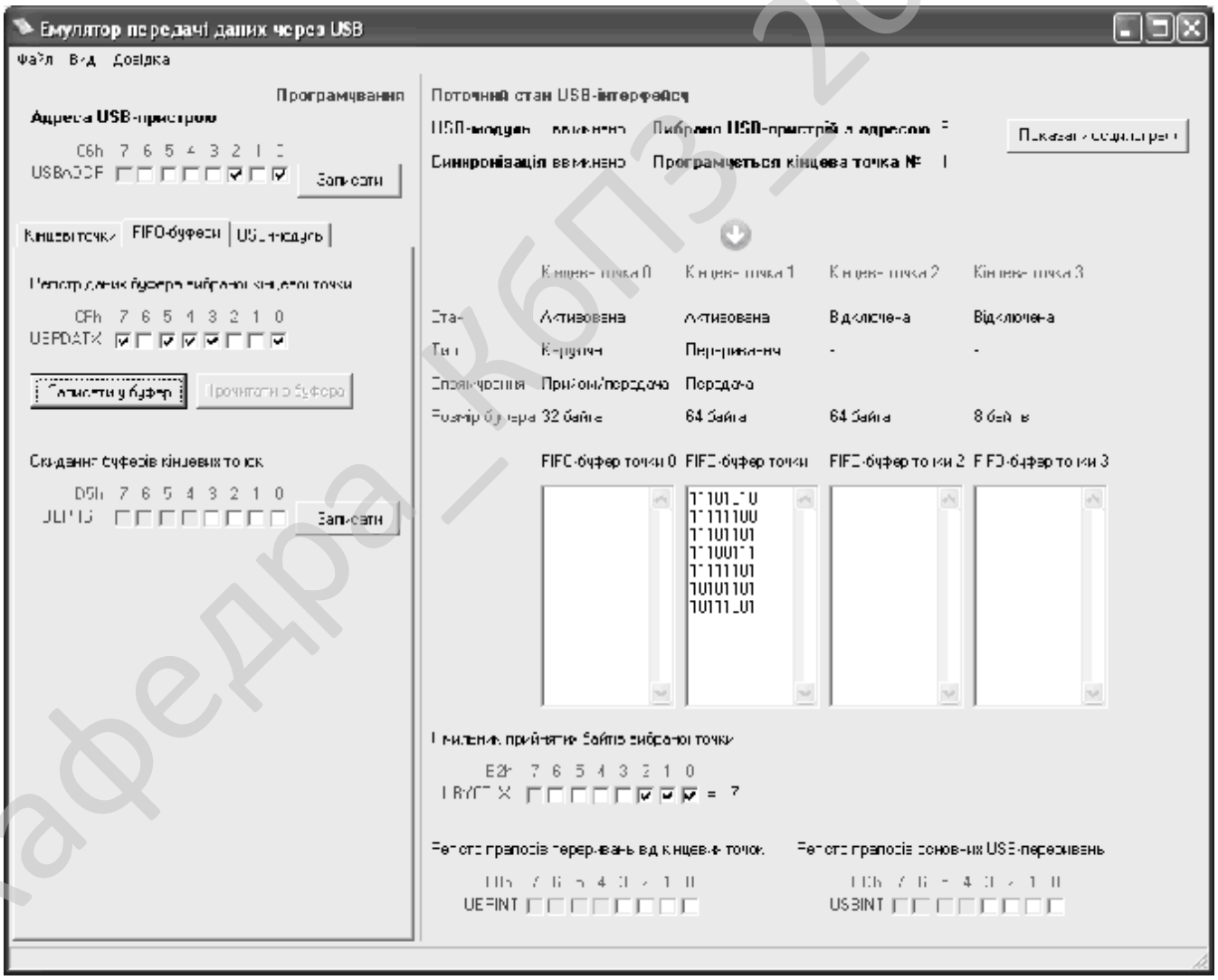

Рисунок 5.2 – Головне вікно програми (запис даних у FIFO-буфер)

Вкладка «USB-модуль» дозволяє керувати роботою USB-інтерфейсу. Вона містить регістр керування USB-інтерфейсом USBCON та регістр дозволів основних USB-переривань USBIEN.

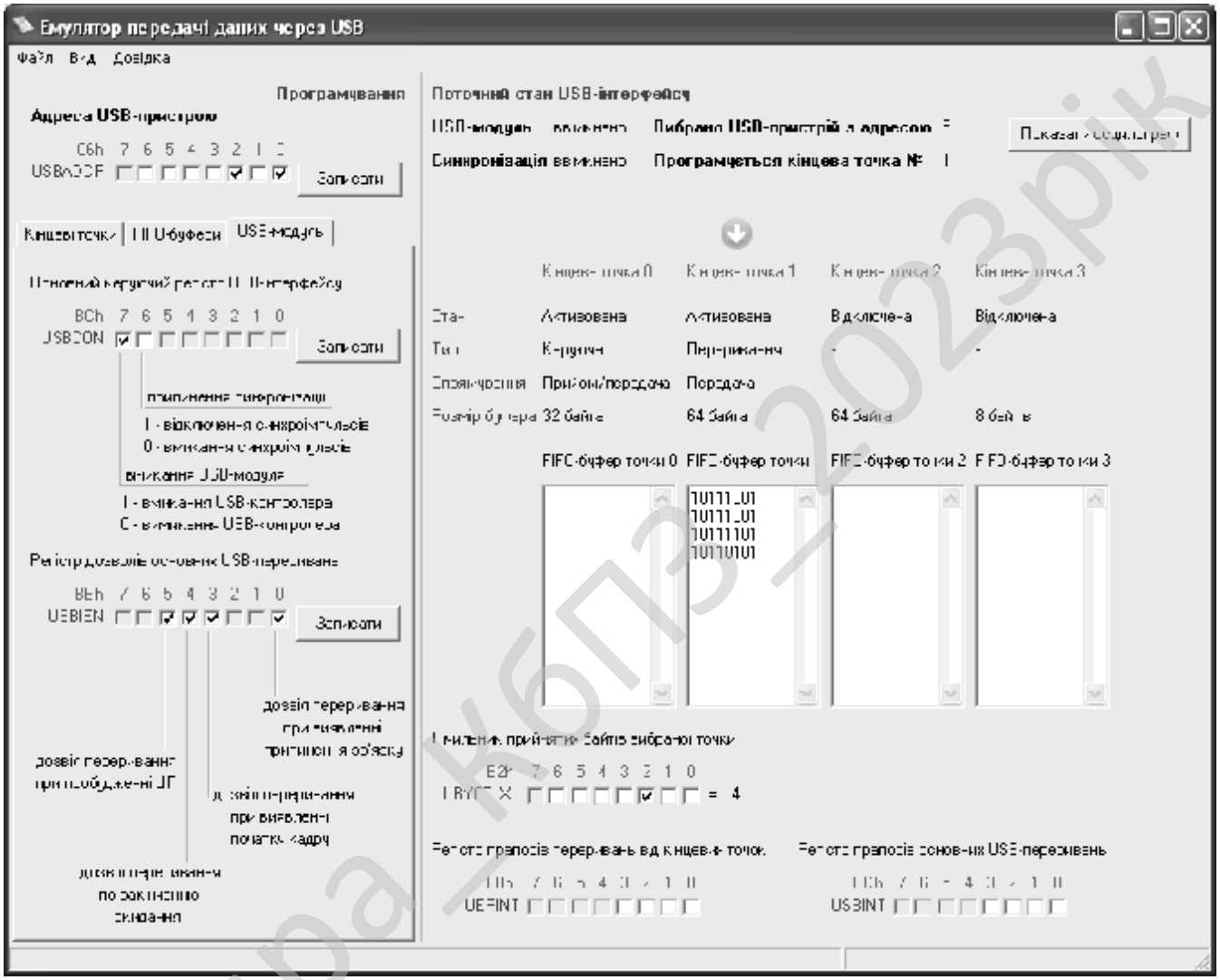

Рисунок 5.3 – Головне вікно програми (управління USB-модулем)

За допомогою регістру USBCON можна вмикати/вимикати USB-модуль та вмикати/вимикати сигнал синхронізації.

За допомогою регістру USBIEN можна дозволяти та забороняти наступні переривання:

– переривання при пробудженні ЦП;

– переривання по закінченню скидання;

– переривання при виявленні початку кадру;

– переривання при виявленні припинення.

Для перегляду часової діаграми сигналів USB-інтерфейсу слід натиснути кнопку «Показати осцилограф», або вибрати пункт меню Вид->Показати осцилограф.

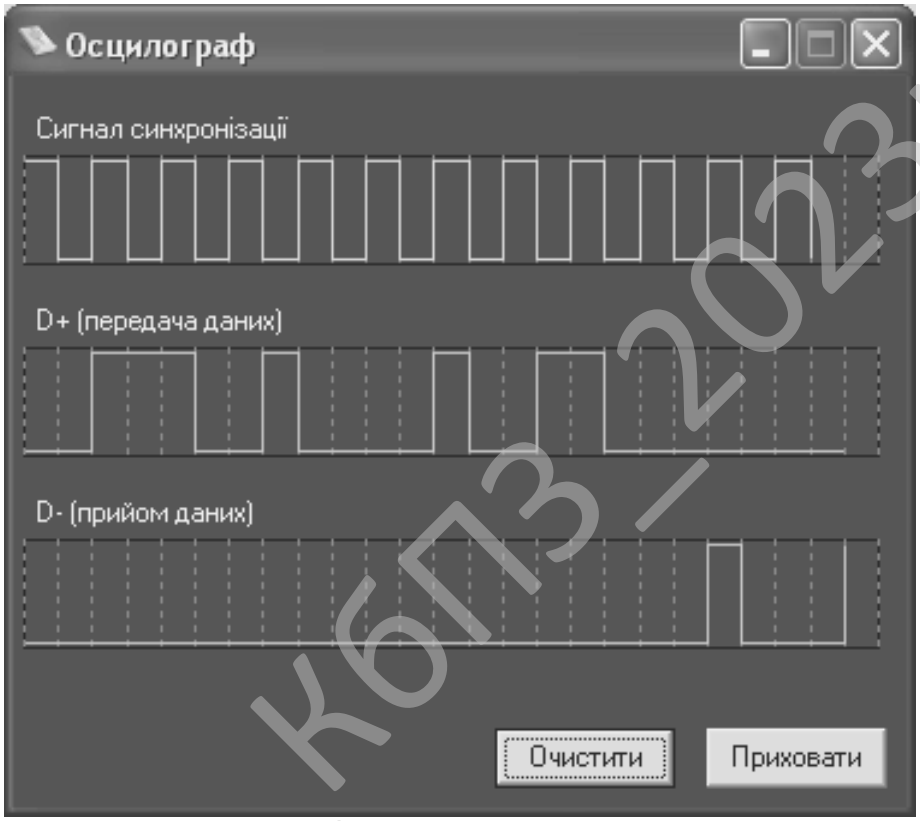

Рисунок 5.4 – Вікно «Осцилограф», що показує часову діаграму сигналів USB-інтерфейсу

Дане вікно дозволяє переглянути часові діаграми наступних сигналів:

– Сигнал синхронізації USB-інтерфейсу.

– Сигнал D+ (передача даних).

– Сигнал D- (прийом даних).

Кнопка «Очистити» дозволяє стерти попередні покази осцилографа.

Кнопка «Приховати» приховує вікно осцилографа.

В програмі є довідка про регістри USB-інтерфейсу, що дозволяє переглянути призначення та формат кожного з регістрів, а також окремо призначення кожного з їх бітів. Для відкриття довідки слід натиснути пункт меню Довідка->Інформація про регістри.

Для отримання інформації про потрібний регістр, або про один з його бітів, слід натиснути кнопку відповідно з назвою регістру чи номером біту навпроти регістру.

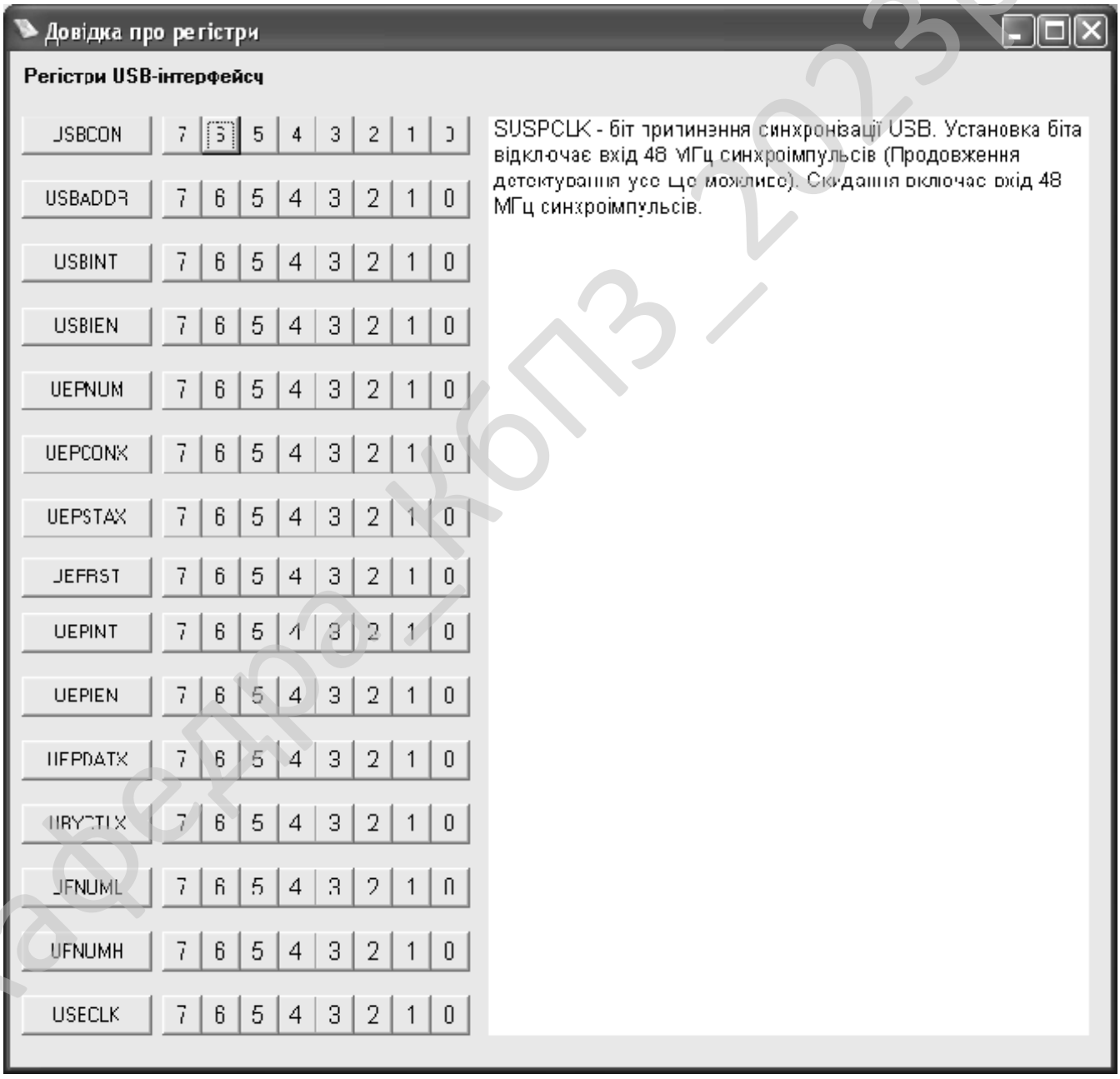

Рисунок 5.5 – Довідка про регістри USB-інтерфейсу

Переглянути короткі відомості про програму та її автора можна натиснувши пункт меню Довідка->Про програму…, після чого з'явиться діалогове вікно, зображене на рисунку 5.2.

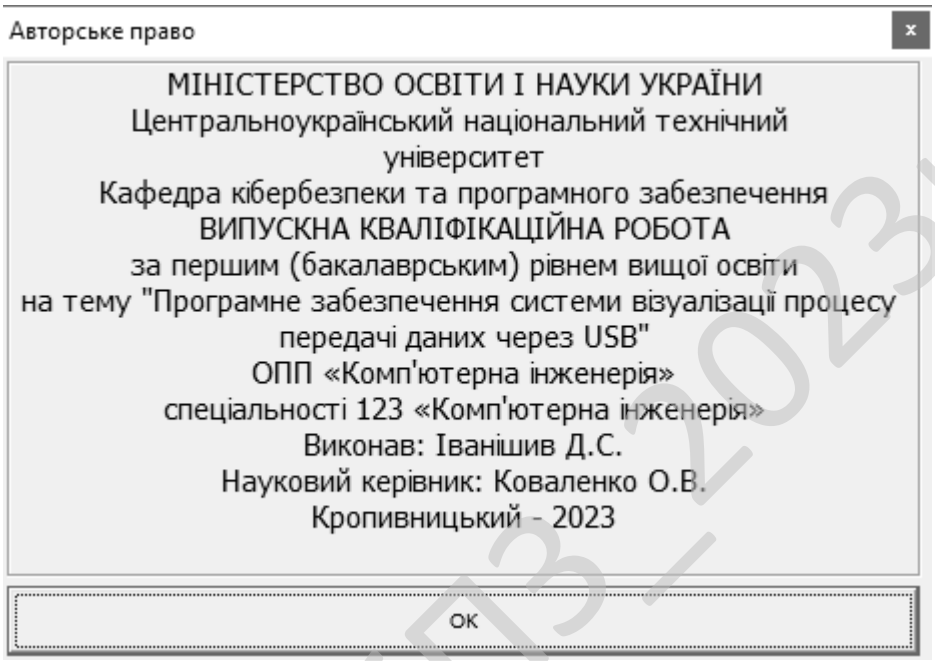

Рисунок 5.6 – Вікно «Про програму…»

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи візуалізації процесу передачі даних через USB.

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 процесу передачі даних через USB.

– Досліджена система візуалізації процесу передачі даних через USB.

– На основі отриманих результатів досліджень створена програмна реалізація системи візуалізації процесу передачі даних через USB.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання візуалізації процесу передачі даних через USB.

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 системи візуалізації процесу передачі даних через USB. 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 DSA.

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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Агуров П. В. Интерфейс USB. Практика использования и программирования. – СПб.: БХВ–Петербург,2004.–576 с.–ISBN 5–94157–202–6

2. Гук М. Аппаратные интерфейсы ПК. Энциклопедия. – СПб.: Питер, 2002. – 528 с.

3. Гук М. Аппаратные средства IBM PC. – СПб.: Питер, 2000. – 723 с.

4. Жедицький В.Ц. Охорона праці користувачів комп'ютерів: Навч. посібник.-2-ге вид. поп.– Львів: Афіша, 2000.-176 с.

5. Кулаков В. Программирование на аппаратном уровне: Спец. справ.. – СПб.: Питер, 2003.

6. Лапин А. Интерфейсы. Выбор и реализация. Издательство: Техносфера, 2005 г. - 168 стр.

7. Миценко І.М., Мезенцева О.М. Цивільна оборона: навчальний посібник – Кіровоград: Кіровоградська районна друкарня, 2003. – 404 с.

8. Таненбаум Э. Архитектура компьютера. – СПб.: Питер, 2002.

9. Тондо К, Гимпел С. Язык Си. – Кн. ответов: Пер. с англ. – М.: Финансы и статистика, 1994, – 160 с.

10. Страуструп Б. Язык программирования C++. – 3–е изд.: Пер. с англ. – СПб.– М.: Невский диалект – Издательство БИНОМ, 1999. – 991 с.

11. Харбисон СП., Стил Г.Л. Язык программирования С: Пер. с англ.– М.: ООО Бином–Пресс, 2004. – 528 с.

12. Хэзфилд Р., Кирби Л. Искусство программирования на С. Фундаментальные алгоритмы, структуры данных и примеры приложений. Энциклопедия программиста. – Киев: ДиаСофт, 2001. – 736 с.

13. Шилдт Г. Полный справочник по С. – 4–е изд.: Пер. с англ. – М.: Издат. дом "Вильяме", 2002. – 704 с.

14. Г. Шилдт Теория и практика C++ пер. с англ. – СПб.: BHV – Санкт –

Петербург, 1999.– 416 с. Ил.

15. Kovalenko O., Popereshnyak S., Grinenko S., Grinenko O., Radivilova T. «Methods for Assessing the Maturity Levels of Software Ecosystems». CEUR Workshop Proceedings Volume 2654, 2019, Pages 251-261. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083214331&origin=AuthorNamesList&txGid=5633fba897776a6e0f3d5633fbcd3fbe (Scopus).

16. Kovalenko O., Drieieva H., Simakhin V., Bondar S., Drieiev O., Zhumadilova M. «Multifractal Properties of Traffic Generator Based on Markov Chains ». CEUR Workshop Proceedings Volume 2588, 2019, Pages 567-579. Режим доступу: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083214331&origin=AuthorNamesList&txGid=176e2cada8976a6e0f3d5633fbcd3fbe (Scopus).

17. Kovalenko Oleksandr Qualitative risk analysis of software development / Oleksandr Kovalenko, Jamil Al-Azzeh, Oleksii Smirnov, Anna Kovalenko, Serhii Smirnov // Asian Journal of Information Technology. – Volume 17 Issue 3. – Medwell Journals. – 2018. – P. 218-230. ISSN: 1682-3915. URL: http://medwelljournals.com/abstract/?doi=ajit.2018.218.230 Doi: ajit.2018.218.230

18. Kovalenko Oleksandr, The mathematical model of the testing technology for DOM XSS vulnerabilities / O. Kovalenko, O. Smirnov, A.Kovalenko, S. Smirnov, V. Vialkova // Scientific & practical cyber security journal (SPCSJ) Volume 2 Issue 1, P. 22-28. Georgia. Tbilisi. Scientific Cyber Security Association (SCSA), 2018 ISSN: 2587-4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/04-3-o.kovalenko-a.kovalenko-o.smirnov-s.smirnov-v.vialkova.pdf

19. Коваленко А.В. Технология тестирования DOM XSS уязвимости / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Scientific & practical cyber security journal (SPCSJ) Volume 1. Issue 1. P. 38-45 Georgia. Tbilisi. Scientific Cyber Security Association (SCSA), 2017 ISSN: 2587-4667. URL: https://journal.scsa.ge/wp-content/uploads/2018/12/8-dom-xss-testing-technology-vulnerabilities.pdf

20. Коваленко О.В. Моделі та методи розроблення програмного

забезпечення комп'ютерних систем для підвищення безпеки даних: монографія / О.В. Коваленко // К.: Вид. «КОД» – 2019. – 305 с.

21. Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

22. Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

23. Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений / А.В. Коваленко, А.А. Смирнов // Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

24. Коваленко А.В. Задачи распознавания ситуаций в ERP системах / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник наукових праць "Системи обробки інформації". – Випуск 4(120). – Х.: ХУПС – 2014. – С. 161-164.

25. Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник наукових праць "Системи обробки інформації". – Випуск 5(142). – Х.: ХУПС – 2016. – С. 153-157.

26. Коваленко А.В. Проблемы анализа и оценки рисков информационной деятельности / А.В. Коваленко, А.А. Смирнов, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць "Системи обробки інформації". – Випуск 3(140). – Х.: ХУПС – 2016. – С. 40-42.

27. Коваленко А.В. Метод качественного анализа рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, Н.Н. Якименко, А.П. Доренский // Наука і техніка Повітряних Сил Збройних Сил України. – Випуск

2(23). – Харків: ХУПС. – 2016. – С. 150-158.

28. Коваленко А.В. Метод количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, Н.Н. Якименко, А.П. Доренский // Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. – 2016. – С. 128-133.

29. Коваленко А.В. Использование псевдобулевых методов бивалентного программирования для управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Системи управління, навігації та зв'язку. – Випуск 1 (37). – Полтава: ПолтНТУ. – 2016. – С. 98-103.

30. Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 2 (38). – Полтава: ПолтНТУ. – 2016. – С. 93-100.

31. Коваленко А.В. Технология тестирования уязвимости к SQL инъекциям / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 5 (45). – Полтава: ПолтНТУ. – 2017. – С. 66-71.

32. Коваленко А.В. Масштабирование имитационной модели технологии тестирования безопасности / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 6 (46). – Полтава: ПолтНТУ. – 2017. – С. 181-184.

33. Коваленко А.В. Имитационная модель технологии тестирования безопасности Web-приложений / А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 1 (47). – Полтава: ПолтНТУ. – 2018. – С. 114-123.

34. Коваленко О.В. Методи якісного аналізу та кількісної оцінки ризиків розроблення програмного забезпечення/ О.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 3 (49). – Полтава: ПолтНТУ. – 2018. – С. 116-125.

35. Коваленко О.В. Управління ризиками розроблення програмного забезпечення за умови обмеженості коштів виділених на усунення помилок безпеки/ О.В. Коваленко // Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Випуск 31. – Кропивницький:

ЦНТУ. – 2018. – С. 128-140.

36. Коваленко О.В. GERT-мережевий синтез технології тестування на вразливість WEB-додатків/ О.В. Коваленко // Захист інформації. – Випуск 20(2) – К.: НАУ. – 2018. – С. 89-94.

37. Коваленко О.В. Імітаційна модель технології тестування безпеки на основі положень теорії масштабування / О.В. Коваленко // Безпека інформації. – Випуск 24 (2). – К.: НАУ. – 2018. – С. 110-117.

38. Коваленко О.В. Оцінка ефективності технології тестування безпеки / О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 2, 2018. – С. 137-141

39. Коваленко О.В. Методи та засоби управління безпекою додатків / О.В. Коваленко // Інформаційно-керуючі системи на залізничному транспорті. №4, 2018. – С. 41-44.

40. Коваленко О.В. Розробка інформаційної технології передтестової компіляції та розподілу доступу / О.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 4 (50). – Полтава: ПолтНТУ. – 2018. – С. 115-119.

41. Коваленко О.В. Аналіз та дослідження інформаційних технологій розробки програмного забезпечення/ О.В. Коваленко // Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки. Том 29 (68) № 5, 2018. – С. 131-137.

42. Коваленко О.В. Удосконалений метод управління ризиками розроблення програмного забезпечення на основі напівмарковської моделі прийняття рішень/ О.В. Коваленко // Сучасні інформаційні системи. – Випуск 2(3). – Харків. – 2018. – С. 41-48.

43. Коваленко О.В. Математичні моделі технології тестування DOM XSS вразливості та вразливості до SQL ін'єкцій / О.В. Коваленко // Вісник Черкаського державного технологічного університету. Серія : Технічні науки №4, 2018. – С. 29-36.

44. Коваленко О.В. Математична модель технології тестування

вразливості до SQL ін’єкцій / О.В. Коваленко // Системи управління, навігації та зв’язку. – Випуск 6 (58). – Полтава: ПолтНТУ. – 2019. – С. 43-47.

45. Коваленко О.В. Математична модель технології тестування комплексу DOM XSS вразливостей для аналітичної оцінки часових витрат / О.В. Коваленко // Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 173-180, 2019.

46. Коваленко А.В. Проблемы анализа и оценки рисков информационной деятельности / А.В. Коваленко, А.А. Смирнов // Збірник наукових праць II міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 24-27 лютого 2016 р. – Київ: Європейський університет. – 2016. – С. 138-139.

47. Коваленко А.В. Анализ и оценка рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез «Securitea informationala 2015-2016». Conferenta internationala (editia a XII-a). Chisinau. Moldova. 3 martie 2016. – Chisinau: ADSEM. – 2016. – P. 96-102.

48. Коваленко А.В. Исследование источников и причин риска разработки программного обеспечения, этапов и работ, при выполнении которых возникает риск / А.В. Коваленко, А.А. Смирнов // Збірник тез VII всеукраїнської науково-практичної конференції "Інформатика та системні науки (ІСН-2016)". м. Полтава. 10-12 березня 2016 р. – Полтава.: ПУЕТ – 2016. – С. 264-266.

49. Коваленко А.В. Оценка показателя чистой приведенной стоимости для количественной оценки рисков проекта разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. м. Київ. 10-11 березня 2016 р. – Київ: КНУ ім. Тараса Шевченко – 2016. – С. 81-82.

50. Коваленко А.В. Методика структурной идентификации рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез Міжнародної науково-практичної конференції «Інформаційна безпека та

комп'ютерні технології» (IS&CT). м. Кіровоград. 24-25 березня 2016 р. – Кіровоград: КНТУ. – 2016. – С. 71-72.

51. Коваленко А.В. Методы качественного анализа рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016). м. Харків. 30 березня – 1 квітня 2016 р. – Харків: НТУ «ХПІ». – 2016. – С. 6-7.

52. Коваленко А.В. Структурная идентификация рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 15-16 квітня 2016 р. – Кіровоград: КНТУ. – 2016. – С. 175-182.

53. Коваленко А.В. Исследование разработанной методики структурной идентификации рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез VIII міжнародної науково-практичної конференції "Проблеми і перспективи розвитку IT-індустрії". м. Харків. 28-29 квітня 2016 р. – Харків: ХНЕУ. – 2016. – С. 49.

54. Коваленко А.В. Исследование дерева рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез III міжнародної науково-практичної конференції «Інформаційна та економічна безпека» (INFECO-2016)». м. Харків. 28-30 квітня 2016 р. – Харків: ХНМІ ДВНЗ «УБС». – 2016. – С. 174-178.

55. Коваленко А.В. Методы качественного анализа и количественной оценки рисков разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Сборник тезисов XII международной конференции "Стратегия качества в промышленности и образовании". г. Варна. Болгария. 30 мая – 02 июня 2016 г – Варна. ТУВ. – 2016. – С. 585-589.

56. Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко //

Матеріали Всеукраїнської науково-практичної конференції «Кібербезпека в Україні: правові та організаційні питання». м. Одеса, 21 жовтня 2016 р. – Одеса : ОДУВС, 2016. – С.146-148.

57. Коваленко А.В. Метод управления рисками разработки программного обеспечения с использованием псевдобулевых методов бивалентного программирования / А.В. Коваленко, А.А. Смирнов // Матеріали Всеукраїнської науково-практичної конференції «Актуальні задачі та досягнення у галузі кібербезпеки». м. Кропивницький, 23-25 листопада 2016 року – Кропивницький: ЦНТУ, 2016. – С. 162.

58. Коваленко А.В. Псевдобулевые методы бивалентного программирования для управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко, С.А. Смирнов // Збірник наукових праць III міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 22-25 лютого 2017 р. – Київ: Європейський університет. – 2017. – С. 158-162.

59. Коваленко А.В. Метод управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов // Збірник тез II науково-практичної конференції “Проблеми кібербезпеки інформаційно-телекомунікаційних систем”. м. Київ. 23-24 березня 2017 р. – Київ: КНУ ім. Тараса Шевченко – 2017. – С. 203-205.

60. Коваленко А.В. Алгоритмы анализа уязвимостей при управлении рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Conferenta internationala (editia a XIII-a). «Securitea informationala 2017». Chisinau. Republic of Moldova. 4-5 aprilie 2017. – Chisinau: ADSEM. – 2017. – P. 19-22.

61. Коваленко А.В. Алгоритм анализа DOM XSS уязвимости при управлении рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез дев’ятнадцятого міжнародного

науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кропивницький 7-8 квітня 2017 р. – Кропивницький: ГЛА НАУ. – 2017. – С. 125-127.

62. Коваленко А.В. Алгоритм анализа уязвимости SQL Injection для управления рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез другої міжнародної науково-технічної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2017). м. Харків.10-12 квітня 2017 р. – Харків: НТУ «ХПІ». – 2017. – С. 27.

63. Коваленко А.В. Метод управления рисками разработки программного обеспечения на основе алгоритмов анализа уязвимостей / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез Міжнародної науково-практичної конференції «Інформаційна безпека та комп'ютерні технології» (IS&CT). м. Кіровоград. 20-22 квітня 2017 р. Кіровоград: КНТУ. – 2017. – С. 92.

64. Коваленко А.В. Алгоритмы анализа DOM XSS уязвимости и уязвимости SQL Injection при управлении рисками разработки программного обеспечения / А.В. Коваленко, А.А. Смирнов, А.С. Коваленко // Збірник тез IX міжнародної науково-практичної конференції “Проблеми і перспективи розвитку IT-індустрії”. м. Харків. 20-21 квітня 2017 р. – Харків: ХНЕУ. – 2017. – С. 61.

65. Kovalenko O.V. Method of testing the dom xss vulnerability / Kovalenko Oleksandr, Kovalenko Anna, Smirnov Oleksii, Smirnov Serhii // International Conference «information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces “Alexandru cel Bun”. – 2017. – P. 7.

66. Коваленко О.В. Метод тестування DOM XSS уразливості / О.В. Коваленко, О.А. Смірнов, А.С. Коваленко, С.А. Смірнов // Збірник тез всеукраїнської науково-практичної інтернет-конференції «Автоматика та комп’ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті». м. Кропивницький. 16-17 листопада 2017 р. –

Кропивницький: ЦНТУ. – 2017. – С. 198-199.

67. Коваленко О.В. GERT-модель технології тестування DOM XSS уразливості / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник наукових праць IV міжнародної науково-практичної конференції «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації». м. Київ. 21-24 лютого 2018 р. – Київ: Європейський університет. – 2018. – С. 65-70.

68. Коваленко О.В. Технології тестування уразливостей Web-застосунків з використанням GERT-моделі / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник тез всеукраїнської науково-практичної конференції "Комп'ютерні інтелектуальні системи та мережі (КІСМ-2018)". м. Кривий Ріг. 21-23 березня 2018 р. – Кривий Ріг.: ДВНЗ КНУ – 2018. – С. 227-230.

69. Коваленко А.В. Тестирование уязвимости Web-приложений к атаке вида межсайтовый скриптинг / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Збірник тез «Securitea informationala 2018». Conferenta internationala (editia a XIV-a). Chisinau. Moldova. 20-21 martie 2018. – Chisinau: ADSEM. – 2018. – P. 54-56.

70. Коваленко А.В. Комплекс математических моделей технологии тестирования web-приложений / А.В. Коваленко, А.С. Коваленко, А.А. Смирнов, С.А. Смирнов // Збірник тез X міжнародної науково-практичної конференції “Проблеми і перспективи розвитку ІТ-індустрії”. м. Харків. 19-20 квітня 2018 р. – Харків: ХНЕУ. – 2018. – С. 38.

71. Коваленко О.В. Розробка методу передтестової компіляції й розподілу доступу / О.В. Коваленко, А.С. Коваленко, О.А. Смірнов, С.А. Смірнов // Збірник наукових праць III міжнародної науково-практичної конференції “Інформаційна безпека та комп’ютерні технології”, м. Кропивницькй. 19-20 квітня 2018 р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215.

72. Коваленко О.В. Оцінка ефективності технологій тестування безпеки уразливостей DOM XSS й SQL-ін'єкцій / О.В. Коваленко, А.С. Коваленко, О.А.

Смірнов, С.А. Смірнов // Сборник тезисов XIV международной конференции "Стратегия качества в промышленности и образовании", Варна, Болгария. 04-07 июня 2018 г – Варна. ТУВ. – 2018. – С. 271-274.

73. Коваленко О.В. Аналіз основних підходів математичного моделювання та методологій для забезпечення максимальних показників безпеки програмного забезпечення / О.В. Коваленко, А.С. Коваленко // Збірник наукових праць всеукр. наук.-практ. конф. здобувачів вищої освіти й молодих учених «Комп'ютерна інженерія і кібербезпека : досягнення та інновації, м. Кропивницькй. 27-29 листопада

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРБ-123.23.0017.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Іванішив Д.С.			Програмне забезпечення системи візуалізації процесу передачі даних через USB	Літ. Б	Аркуш 1 / Аркушів 6
Перевірив		Коваленко О.В.					
Н. Контр.		Гермак В.С.				ЦНТУ КІ-20-3СК	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи візуалізації процесу передачі даних через USB.

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. № 8-02 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 процесу передачі даних через USB.

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;

– 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 процесу передачі даних через USB;

– 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.

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++.

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 – 89 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).

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.

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 О.В.

Програмне забезпечення системи візуалізації процесу передачі даних через USB

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 50

Літера: РП

Кропивницький – 2023 року

Основна програма

Файл ProjectUSB.cpp основної програми

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

#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
USEFORM("Main.cpp", Form1);
USEFORM("pro_pr.cpp", Form2);
USEFORM("help.cpp", Form3);
USEFORM("signal.cpp", Form4);
//---------------------------------------------------------------------------
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->Run();
        }
        catch (Exception &exception)
        {
                 Application->ShowException(&exception);
        }
        catch (...)
        {
                 try
                 {
                         throw Exception("");
                 }
                 catch (Exception &exception)
                 {
                         Application->ShowException(&exception);
                 }
        }
        return 0;
}
//---------------------------------------------------------------------------
```

Файл Main.cpp основної програми

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

#include <vcl.h>
#pragma hdrstop

#include "Main.h"
#include "pro_pr.h"
#include "signal.h"
#include "help.h"

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
int x=0;                          //ініціалізація змінних
int i0=0,i1=0,i2=0,i3=0;
int rr0=0,rr1=0,rr2=0,rr3=0, rr4=0, rr5=0, rr7=0, intp=0;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------


//запис адреси пристрою у регістр USBADDR
void __fastcall TForm1::Button1Click(TObject *Sender)
{
int b0,b1,b2,b3,b4,b5,b6;
int r;

if(USBADDR0->Checked==true) b0=1; else b0=0;
if(USBADDR1->Checked==true) b1=1; else b1=0;
if(USBADDR2->Checked==true) b2=1; else b2=0;
if(USBADDR3->Checked==true) b3=1; else b3=0;
if(USBADDR4->Checked==true) b4=1; else b4=0;
if(USBADDR5->Checked==true) b5=1; else b5=0;
if(USBADDR6->Checked==true) b6=1; else b6=0;

r=b6*64 + b5*32 + b4*16 + b3*8 + b2*4 + b1*2 + b0*1;

NPr->Caption=IntToStr(r);

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

//запис номеру кінцевої точки у регістр UEPNUM
void __fastcall TForm1::Button2Click(TObject *Sender)
{
int b0,b1;

if(NPr->Caption!='0') {

//визначення номеру кінцевої точки, записаного у регістр UEPNUM
if(UEPNUM0->Checked==true) b0=1; else b0=0;
if(UEPNUM1->Checked==true) b1=1; else b1=0;

x=b1*2 + b0*1;

nt->Caption=IntToStr(x);

//виведення стрілки біля вибраної кінцевої точки та
//блокування/розблокування кнопок в залежності від конфігурації
//вибраної кінцевої точки

if(x==0){
```

```
Image1->Visible=true;
Image2->Visible=false;
Image3->Visible=false;
Image4->Visible=false;
if(nn0->Caption=="Прийом/передача"){
BitBtn3->Enabled=true;
Button3->Enabled=true;
}
if(nn0->Caption=="Прийом"){
BitBtn3->Enabled=false;
Button3->Enabled=true;
}
if(nn0->Caption=="Передача"){
BitBtn3->Enabled=true;
Button3->Enabled=false;
}
if(nn0->Caption=="-"){
BitBtn3->Enabled=false;
Button3->Enabled=false;
}
}
if(x==1){
Image1->Visible=false;
Image2->Visible=true;
Image3->Visible=false;
Image4->Visible=false;
if(nn1->Caption=="Прийом/передача"){
BitBtn3->Enabled=true;
Button3->Enabled=true;
}
if(nn1->Caption=="Прийом"){
BitBtn3->Enabled=false;
Button3->Enabled=true;
}
if(nn1->Caption=="Передача"){
BitBtn3->Enabled=true;
Button3->Enabled=false;
}
if(nn1->Caption=="-"){
BitBtn3->Enabled=false;
Button3->Enabled=false;
}
}
if(x==2){
Image1->Visible=false;
Image2->Visible=false;
Image3->Visible=true;
Image4->Visible=false;
if(nn2->Caption=="Прийом/передача"){
BitBtn3->Enabled=true;
Button3->Enabled=true;
}
if(nn2->Caption=="Прийом"){
BitBtn3->Enabled=false;
Button3->Enabled=true;
}
if(nn2->Caption=="Передача"){
BitBtn3->Enabled=true;
Button3->Enabled=false;
}
if(nn2->Caption=="-"){
BitBtn3->Enabled=false;
Button3->Enabled=false;
}
}
if(x==3){
Image1->Visible=false;
Image2->Visible=false;
Image3->Visible=false;
```

```
Image4->Visible=true;
if(nn3->Caption=="Прийом/передача"){
BitBtn3->Enabled=true;
Button3->Enabled=true;
}
if(nn3->Caption=="Прийом"){
BitBtn3->Enabled=false;
Button3->Enabled=true;
}
if(nn3->Caption=="Передача"){
BitBtn3->Enabled=true;
Button3->Enabled=false;
}
if(nn3->Caption=="-"){
BitBtn3->Enabled=false;
Button3->Enabled=false;
}
}


}
else MessageDlg("Вкажіть адресу пристрою!", mtError, TMsgDlgButtons() << mbOK,
0); //виведення повідомлення про помилку, якщо користувач намагається вказати
кінцеву точку не вказавши пристрій



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

void __fastcall TForm1::N5Click(TObject *Sender)
{
Form2->Show(); //відкриття вікна «Про програму...»
}
//---------------------------------------------------------------------------

void __fastcall TForm1::USB1Click(TObject *Sender)
{
Form3->Show();//відкриття вікна довідки
}
//---------------------------------------------------------------------------

//скидання буферів кінцевих точок
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
int b0,b1,b2,b3;
if(UEPRST0->Checked==true) b0=1; else b0=0;
if(UEPRST1->Checked==true) b1=1; else b1=0;
if(UEPRST2->Checked==true) b2=1; else b2=0;
if(UEPRST3->Checked==true) b3=1; else b3=0;

if(b0==1) buf0->Clear();
if(b1==1) buf1->Clear();
if(b2==1) buf2->Clear();
if(b3==1) buf3->Clear();

UEPRST0->Checked=false;
UEPRST1->Checked=false;
UEPRST2->Checked=false;
UEPRST3->Checked=false;



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

//активізація та конфігурування кінцевих точок
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
int a0,a1,a2,a3,a7;
```

```
int r1,r2;

if(NPr->Caption!='0') {

r1=x;

if(UEPCONX0->Checked==true) a0=1; else a0=0;
if(UEPCONX1->Checked==true) a1=1; else a1=0;
if(UEPCONX2->Checked==true) a2=1; else a2=0;
if(UEPCONX3->Checked==true) a3=1; else a3=0;
if(UEPCONX7->Checked==true) a7=1; else a7=0;


if((a7==1)&(r1==0))
{
s0->Caption="Активована";
if((a1==0)&(a0==0))
{
t0->Caption="Керуюча";
nn0->Caption="Прийом/передача";
BitBtn3->Enabled=true;
Button3->Enabled=true;
}
if((a1==0)&(a0==1)) t0->Caption="Ізохронна";
if((a1==1)&(a0==0)) t0->Caption="Пакетна";
if((a1==1)&(a0==1)) t0->Caption="Переривання";
if((a2==1)&(a1+a0!=0))
{
nn0->Caption="Прийом";
BitBtn3->Enabled=false;
Button3->Enabled=true;
}
if((a2==0)&(a1+a0!=0))
{
nn0->Caption="Передача";
BitBtn3->Enabled=true;
Button3->Enabled=false;
}
}

if((a7==0)&(r1==0))
{
MessageDlg("Не можна відключати кінцеву точку 0!", mtError, TMsgDlgButtons() <<
mbOK, 0);
}


if((a7==1)&(r1==1))
{
s1->Caption="Активована";
if((a1==0)&(a0==0))
{
t1->Caption="Керуюча";
nn1->Caption="Прийом/передача";
BitBtn3->Enabled=true;
Button3->Enabled=true;
}
if((a1==0)&(a0==1)) t1->Caption="Ізохронна";
if((a1==1)&(a0==0)) t1->Caption="Пакетна";
if((a1==1)&(a0==1)) t1->Caption="Переривання";
if((a2==1)&(a1+a0!=0))
{
nn1->Caption="Прийом";
BitBtn3->Enabled=false;
Button3->Enabled=true;
}
if((a2==0)&(a1+a0!=0))
{
nn1->Caption="Передача";
```

```
BitBtn3->Enabled=true;
Button3->Enabled=false;
}
}

if((a7==0)&(r1==1))
{
s1->Caption="Відключена";
t1->Caption="-";
nn1->Caption="-";
}

if((a7==1)&(r1==2))
{
s2->Caption="Активована";
if((a1==0)&(a0==0))
{
t2->Caption="Керуюча";
nn2->Caption="Прийом/передача";
BitBtn3->Enabled=true;
Button3->Enabled=true;
}
if((a1==0)&(a0==1)) t2->Caption="Ізохронна";
if((a1==1)&(a0==0)) t2->Caption="Пакетна";
if((a1==1)&(a0==1)) t2->Caption="Переривання";
if((a2==1)&(a1+a0!=0))
{
nn2->Caption="Прийом";
BitBtn3->Enabled=false;
Button3->Enabled=true;
}
if((a2==0)&(a1+a0!=0))
{
nn2->Caption="Передача";
BitBtn3->Enabled=true;
Button3->Enabled=false;
}
}


if((a7==0)&(r1==2))
{
s2->Caption="Відключена";
t2->Caption="-";
nn2->Caption="-";
}

if((a7==1)&(r1==3))
{
s3->Caption="Активована";
if((a1==0)&(a0==0))
{
t3->Caption="Керуюча";
nn3->Caption="Прийом/передача";
BitBtn3->Enabled=true;
Button3->Enabled=true;
}
if((a1==0)&(a0==1)) t3->Caption="Ізохронна";
if((a1==1)&(a0==0)) t3->Caption="Пакетна";
if((a1==1)&(a0==1)) t3->Caption="Переривання";
if((a2==1)&(a1+a0!=0))
{
nn3->Caption="Прийом";
BitBtn3->Enabled=false;
Button3->Enabled=true;
}
if((a2==0)&(a1+a0!=0))
{
nn3->Caption="Передача";
```

```
BitBtn3->Enabled=true;
Button3->Enabled=false;
}
}

if((a7==0)&(r1==3))
{
s3->Caption="Відключена";
t3->Caption="-";
nn3->Caption="-";
}
}
else MessageDlg("Вкажіть адресу пристрою!", mtError, TMsgDlgButtons() << mbOK,
0);



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

//Запис у FIFO-буфери за допомогою регістру UEPDATX
void __fastcall TForm1::BitBtn3Click(TObject *Sender)
{
int b0,b1,b2,b3,b4,b5,b6,b7;
ShortString  r;

if(NPr->Caption!='0') {
if(UEPDATX0->Checked==true) b0=1; else b0=0;
if(UEPDATX1->Checked==true) b1=1; else b1=0;
if(UEPDATX2->Checked==true) b2=1; else b2=0;
if(UEPDATX3->Checked==true) b3=1; else b3=0;
if(UEPDATX4->Checked==true) b4=1; else b4=0;
if(UEPDATX5->Checked==true) b5=1; else b5=0;
if(UEPDATX6->Checked==true) b6=1; else b6=0;
if(UEPDATX7->Checked==true) b7=1; else b7=0;

r=IntToStr(b7) + IntToStr(b6) + IntToStr(b5) + IntToStr(b4) + IntToStr(b3) +
IntToStr(b2) + IntToStr(b1) + IntToStr(b0);

if(x==0){
if(buf0->Lines->Count<32) buf0->Lines->Add(r);
else { buf0->Lines->Delete(0);  buf0->Lines->Add(r);MessageDlg("Переповнення
буфера!", mtWarning, TMsgDlgButtons() << mbOK, 0);}
}

if(x==1){
if(buf1->Lines->Count<64) buf1->Lines->Add(r);
else { buf1->Lines->Delete(0);  buf1->Lines->Add(r); MessageDlg("Переповнення
буфера!", mtWarning, TMsgDlgButtons() << mbOK, 0);}
}

if(x==2){
if(buf2->Lines->Count<64) buf2->Lines->Add(r);
else { buf2->Lines->Delete(0);  buf2->Lines->Add(r);  MessageDlg("Переповнення
буфера!", mtWarning, TMsgDlgButtons() << mbOK, 0);}
}

if(x==3)
{
if(buf3->Lines->Count<8) buf3->Lines->Add(r);
else { buf3->Lines->Delete(0);  buf3->Lines->Add(r); MessageDlg("Переповнення
буфера!", mtWarning, TMsgDlgButtons() << mbOK, 0); }
}


}
else MessageDlg("Вкажіть адресу пристрою!", mtError, TMsgDlgButtons() << mbOK,
0);
```

```
}

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

//Обчислення значення та виведення на екран регістру лічильника байтів у FIFO-
буфері
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
int b0,b1,b2,b3,b4,b5,b6,b7;
int r;
ShortString rr;
div_t d;


if(x==0) r=buf0->Lines->Count;
if(x==1) r=buf1->Lines->Count;
if(x==2) r=buf2->Lines->Count;
if(x==3) r=buf3->Lines->Count;
Label87->Caption=IntToStr(r);

if(r!=0) {
d = div(r,2); r=d.quot;  b0=d.rem;
d = div(r,2); r=d.quot;  b1=d.rem;
d = div(r,2); r=d.quot;  b2=d.rem;
d = div(r,2); r=d.quot;  b3=d.rem;
d = div(r,2); r=d.quot;  b4=d.rem;
d = div(r,2); r=d.quot;  b5=d.rem;
d = div(r,2); r=d.quot;  b6=d.rem;
d = div(r,2); r=d.quot;  b7=d.rem;

rr=IntToStr(b7) + IntToStr(b6) + IntToStr(b5) + IntToStr(b4) + IntToStr(b3) +
IntToStr(b2) + IntToStr(b1) + IntToStr(b0);
}
else  rr="00000000";

if(rr[8]=='1') UBYCTLX0->Checked=true; else UBYCTLX0->Checked=false;
if(rr[7]=='1') UBYCTLX1->Checked=true; else UBYCTLX1->Checked=false;
if(rr[6]=='1') UBYCTLX2->Checked=true; else UBYCTLX2->Checked=false;
if(rr[5]=='1') UBYCTLX3->Checked=true; else UBYCTLX3->Checked=false;
if(rr[4]=='1') UBYCTLX4->Checked=true; else UBYCTLX4->Checked=false;
if(rr[3]=='1') UBYCTLX5->Checked=true; else UBYCTLX5->Checked=false;
if(rr[2]=='1') UBYCTLX6->Checked=true; else UBYCTLX6->Checked=false;
if(rr[1]=='1') UBYCTLX7->Checked=true; else UBYCTLX7->Checked=false;

if((intp>0)&(i0==1)&(x==0)) UEPINT0->Checked=true;
if(((intp==0)|(i0==0))&(x==0)) UEPINT0->Checked=false;

if((intp>0)&(i1==1)&(x==1)) UEPINT1->Checked=true;
if(((intp==0)|(i1==0))&(x==1)) UEPINT1->Checked=false;

if((intp>0)&(i2==1)&(x==2)) UEPINT2->Checked=true;
if(((intp==0)|(i2==0))&(x==2)) UEPINT2->Checked=false;

if((intp>0)&(i3==1)&(x==3)) UEPINT3->Checked=true;
if(((intp==0)|(i3==0))&(x==3)) UEPINT3->Checked=false;
}
//---------------------------------------------------------------------------

//Читання FIFO-буферів за допомогою регістру UEPDATX
void __fastcall TForm1::Button3Click(TObject *Sender)
{
int b0,b1,b2,b3,b4,b5,b6,b7;
ShortString r;

if(NPr->Caption!='0') {

if(x==0)
if(buf0->Lines->Count>0)
{
```

```
if(buf0->Lines->Strings[0][8]=='1') UEPDATX0->Checked=true; else UEPDATX0-
>Checked=false;
if(buf0->Lines->Strings[0][7]=='1') UEPDATX1->Checked=true; else UEPDATX1-
>Checked=false;
if(buf0->Lines->Strings[0][6]=='1') UEPDATX2->Checked=true; else UEPDATX2-
>Checked=false;
if(buf0->Lines->Strings[0][5]=='1') UEPDATX3->Checked=true; else UEPDATX3-
>Checked=false;
if(buf0->Lines->Strings[0][4]=='1') UEPDATX4->Checked=true; else UEPDATX4-
>Checked=false;
if(buf0->Lines->Strings[0][3]=='1') UEPDATX5->Checked=true; else UEPDATX5-
>Checked=false;
if(buf0->Lines->Strings[0][2]=='1') UEPDATX6->Checked=true; else UEPDATX6-
>Checked=false;
if(buf0->Lines->Strings[0][1]=='1') UEPDATX7->Checked=true; else UEPDATX7-
>Checked=false;
buf0->Lines->Delete(0);
}

if(x==1)
if(buf1->Lines->Count>0)
{
if(buf1->Lines->Strings[0][8]=='1') UEPDATX0->Checked=true; else UEPDATX0-
>Checked=false;
if(buf1->Lines->Strings[0][7]=='1') UEPDATX1->Checked=true; else UEPDATX1-
>Checked=false;
if(buf1->Lines->Strings[0][6]=='1') UEPDATX2->Checked=true; else UEPDATX2-
>Checked=false;
if(buf1->Lines->Strings[0][5]=='1') UEPDATX3->Checked=true; else UEPDATX3-
>Checked=false;
if(buf1->Lines->Strings[0][4]=='1') UEPDATX4->Checked=true; else UEPDATX4-
>Checked=false;
if(buf1->Lines->Strings[0][3]=='1') UEPDATX5->Checked=true; else UEPDATX5-
>Checked=false;
if(buf1->Lines->Strings[0][2]=='1') UEPDATX6->Checked=true; else UEPDATX6-
>Checked=false;
if(buf1->Lines->Strings[0][1]=='1') UEPDATX7->Checked=true; else UEPDATX7-
>Checked=false;
buf1->Lines->Delete(0);
}

if(x==2)
if(buf2->Lines->Count>0)
{
if(buf2->Lines->Strings[0][8]=='1') UEPDATX0->Checked=true; else UEPDATX0-
>Checked=false;
if(buf2->Lines->Strings[0][7]=='1') UEPDATX1->Checked=true; else UEPDATX1-
>Checked=false;
if(buf2->Lines->Strings[0][6]=='1') UEPDATX2->Checked=true; else UEPDATX2-
>Checked=false;
if(buf2->Lines->Strings[0][5]=='1') UEPDATX3->Checked=true; else UEPDATX3-
>Checked=false;
if(buf2->Lines->Strings[0][4]=='1') UEPDATX4->Checked=true; else UEPDATX4-
>Checked=false;
if(buf2->Lines->Strings[0][3]=='1') UEPDATX5->Checked=true; else UEPDATX5-
>Checked=false;
if(buf2->Lines->Strings[0][2]=='1') UEPDATX6->Checked=true; else UEPDATX6-
>Checked=false;
if(buf2->Lines->Strings[0][1]=='1') UEPDATX7->Checked=true; else UEPDATX7-
>Checked=false;
buf2->Lines->Delete(0);
}

if(x==3)
if(buf3->Lines->Count>0)
{
if(buf3->Lines->Strings[0][8]=='1') UEPDATX0->Checked=true; else UEPDATX0-
>Checked=false;
```

```
if(buf3->Lines->Strings[0][7]=='1') UEPDATX1->Checked=true; else UEPDATX1-
>Checked=false;
if(buf3->Lines->Strings[0][6]=='1') UEPDATX2->Checked=true; else UEPDATX2-
>Checked=false;
if(buf3->Lines->Strings[0][5]=='1') UEPDATX3->Checked=true; else UEPDATX3-
>Checked=false;
if(buf3->Lines->Strings[0][4]=='1') UEPDATX4->Checked=true; else UEPDATX4-
>Checked=false;
if(buf3->Lines->Strings[0][3]=='1') UEPDATX5->Checked=true; else UEPDATX5-
>Checked=false;
if(buf3->Lines->Strings[0][2]=='1') UEPDATX6->Checked=true; else UEPDATX6-
>Checked=false;
if(buf3->Lines->Strings[0][1]=='1') UEPDATX7->Checked=true; else UEPDATX7-
>Checked=false;
buf3->Lines->Delete(0);
}

if(UEPDATX0->Checked==true) b0=1; else b0=0;
if(UEPDATX1->Checked==true) b1=1; else b1=0;
if(UEPDATX2->Checked==true) b2=1; else b2=0;
if(UEPDATX3->Checked==true) b3=1; else b3=0;
if(UEPDATX4->Checked==true) b4=1; else b4=0;
if(UEPDATX5->Checked==true) b5=1; else b5=0;
if(UEPDATX6->Checked==true) b6=1; else b6=0;
if(UEPDATX7->Checked==true) b7=1; else b7=0;

r=IntToStr(b7) + IntToStr(b6) + IntToStr(b5) + IntToStr(b4) + IntToStr(b3) +
IntToStr(b2) + IntToStr(b1) + IntToStr(b0);



}
else MessageDlg("Вкажіть адресу пристрою!", mtError, TMsgDlgButtons() << mbOK,
0);

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

//запис у регістр дозволів переривань від кінцевих точок UEPIEN
void __fastcall TForm1::Button4Click(TObject *Sender)
{
if(UEPIEN0->Checked==true) i0=1; else i0=0;
if(UEPIEN1->Checked==true) i1=1; else i1=0;
if(UEPIEN2->Checked==true) i2=1; else i2=0;
if(UEPIEN3->Checked==true) i3=1; else i3=0;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button5Click(TObject *Sender)
{
Form4->Show();  //відкриття вікна «Осцилограф» за допомогою кнопки
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N7Click(TObject *Sender)
{
Form4->Show();//відкриття вікна «Осцилограф» за допомогою меню користувача
}
//---------------------------------------------------------------------------

//запис у регістр керування та статусу вибраної кінцевої точки
void __fastcall TForm1::Button6Click(TObject *Sender)
{

if(UEPSTAX0->Checked==true) rr0=1; else rr0=0;
if(UEPSTAX1->Checked==true) rr1=1; else rr1=0;
if(UEPSTAX2->Checked==true) rr2=1; else rr2=0;
if(UEPSTAX3->Checked==true) rr3=1; else rr3=0;
if(UEPSTAX4->Checked==true) rr4=1; else rr4=0;
```

```
if(UEPSTAX5->Checked==true) rr5=1; else rr5=0;
if(UEPSTAX7->Checked==true) rr7=1; else rr7=0;

intp = rr0 + rr1 + rr2 + rr3 + rr4 + rr5 + rr7;
}
//---------------------------------------------------------------------------

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

//запис в основний керуючий регістр USB-інтерфейсу
void __fastcall TForm1::BitBtn4Click(TObject *Sender)
{
int b7, b6;

if(USBCON6->Checked==true) b6=1; else b6=0;
if(USBCON7->Checked==true) b7=1; else b7=0;

if((b7==1)&(NPr->Caption=="0"))
{
USB->Caption="ввімкнено";
Button1->Enabled=true;
Button2->Enabled=true;
BitBtn2->Enabled=true;
BitBtn3->Enabled=true;
Button3->Enabled=true;
BitBtn1->Enabled=true;
Button4->Enabled=true;
Button6->Enabled=true;
s0->Caption="Активована";
t0->Caption="Керуюча";
nn0->Caption="Прийом/передача";
s1->Caption="Відключена";
t1->Caption="-";
nn1->Caption="-";
s2->Caption="Відключена";
t2->Caption="-";
nn2->Caption="-";
s3->Caption="Відключена";
t3->Caption="-";
nn3->Caption="-";
}
if(b7==0)
{
USB->Caption="відключено";
Button1->Enabled=false;
Button2->Enabled=false;
BitBtn2->Enabled=false;
BitBtn3->Enabled=false;
Button3->Enabled=false;
BitBtn1->Enabled=false;
Button4->Enabled=false;
Button6->Enabled=false;
s0->Caption="Відключена";
t0->Caption="-";
nn0->Caption="-";
s1->Caption="Відключена";
t1->Caption="-";
nn1->Caption="-";
s2->Caption="Відключена";
t2->Caption="-";
nn2->Caption="-";
s3->Caption="Відключена";
t3->Caption="-";
nn3->Caption="-";
NPr->Caption="0";
```

```
nt->Caption="0";
buf0->Clear();
buf1->Clear();
buf2->Clear();
buf3->Clear();
Image1->Visible=true;
Image2->Visible=false;
Image3->Visible=false;
Image4->Visible=false;
}

if(b6==0) CLK->Caption="ввімкнено";   else CLK->Caption="відключено";

}
```

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 <Menus.hpp>
#include <Buttons.hpp>
#include <Graphics.hpp>
#include <ComCtrls.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:      // IDE-managed Components
        TLabel *Label2;
        TLabel *Label4;
        TCheckBox *USBADDR3;
        TCheckBox *USBADDR7;
        TCheckBox *USBADDR6;
        TCheckBox *USBADDR5;
        TCheckBox *USBADDR4;
        TCheckBox *USBADDR2;
        TCheckBox *USBADDR1;
        TCheckBox *USBADDR0;
        TButton *Button1;
        TLabel *NPr;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TLabel *Label9;
        TLabel *Label10;
        TLabel *Label11;
        TLabel *Label12;
        TBevel *Bevel1;
        TLabel *Label13;
        TLabel *Label14;
        TLabel *Label16;
        TLabel *Label17;
        TLabel *Label18;
        TLabel *Label19;
        TLabel *Label20;
        TLabel *Label21;
        TLabel *Label3;
        TLabel *s0;
        TLabel *s1;
        TLabel *s2;
        TLabel *s3;
        TMainMenu *MainMenu1;
        TMenuItem *N1;
        TMenuItem *N2;
        TMenuItem *N3;
        TMenuItem *N7;
        TMenuItem *N4;
        TMenuItem *USB1;
        TMenuItem *N5;
        TLabel *t0;
        TLabel *t1;
        TLabel *t2;
        TLabel *t3;
        TLabel *nn0;
```

```
TLabel *nn1;
TLabel *nn2;
TLabel *nn3;
TMemo *buf0;
TMemo *buf1;
TMemo *buf2;
TMemo *buf3;
TLabel *Label40;
TLabel *Label41;
TLabel *Label42;
TLabel *Label43;
TLabel *Label44;
TLabel *Label46;
TLabel *Label47;
TLabel *Label48;
TLabel *Label49;
TLabel *Label50;
TLabel *Label51;
TImage *Image1;
TImage *Image2;
TImage *Image3;
TImage *Image4;
TLabel *Label76;
TLabel *Label77;
TLabel *Label78;
TLabel *Label79;
TLabel *Label80;
TLabel *Label81;
TLabel *Label82;
TLabel *Label83;
TLabel *Label84;
TCheckBox *UBYCTLX3;
TCheckBox *UBYCTLX7;
TCheckBox *UBYCTLX6;
TCheckBox *UBYCTLX5;
TCheckBox *UBYCTLX4;
TCheckBox *UBYCTLX2;
TCheckBox *UBYCTLX1;
TCheckBox *UBYCTLX0;
TLabel *Label85;
TLabel *Label86;
TTimer *Timer1;
TLabel *Label87;
TLabel *Label88;
TLabel *Label89;
TLabel *nt;
TButton *Button5;
TStatusBar *StatusBar1;
TLabel *Label45;
TLabel *Label101;
TLabel *Label102;
TLabel *Label103;
TLabel *Label104;
TLabel *Label105;
TLabel *Label106;
TLabel *Label107;
TLabel *Label108;
TLabel *Label109;
TLabel *Label110;
TCheckBox *UEPINT3;
TCheckBox *UEPINT7;
TCheckBox *UEPINT6;
TCheckBox *UEPINT5;
TCheckBox *UEPINT4;
TCheckBox *UEPINT2;
TCheckBox *UEPINT1;
TCheckBox *UEPINT0;
TLabel *Label122;
TLabel *Label123;
```

```
TLabel *Label124;
TLabel *Label125;
TLabel *Label126;
TLabel *Label127;
TLabel *Label128;
TLabel *Label129;
TLabel *Label130;
TLabel *Label131;
TLabel *Label132;
TCheckBox *CheckBox7;
TCheckBox *CheckBox19;
TCheckBox *CheckBox20;
TCheckBox *CheckBox21;
TCheckBox *CheckBox22;
TCheckBox *CheckBox23;
TCheckBox *CheckBox24;
TCheckBox *CheckBox25;
TPageControl *PageControl1;
TTabSheet *TabSheet1;
TTabSheet *TabSheet2;
TLabel *UEPCONX;
TLabel *Label32;
TLabel *Label33;
TLabel *Label34;
TLabel *Label35;
TLabel *Label36;
TLabel *Label37;
TLabel *Label38;
TLabel *Label39;
TLabel *Label31;
TLabel *Label53;
TLabel *Label23;
TLabel *Label22;
TLabel *Label24;
TLabel *Label25;
TLabel *Label26;
TLabel *Label27;
TLabel *Label28;
TLabel *Label29;
TLabel *Label30;
TLabel *Label1;
TLabel *Label52;
TCheckBox *UEPCONX3;
TCheckBox *UEPCONX7;
TCheckBox *CheckBox9;
TCheckBox *CheckBox10;
TCheckBox *CheckBox11;
TCheckBox *UEPCONX2;
TCheckBox *UEPCONX1;
TCheckBox *UEPCONX0;
TBitBtn *BitBtn2;
TCheckBox *CheckBox1;
TCheckBox *CheckBox2;
TCheckBox *CheckBox3;
TCheckBox *CheckBox4;
TCheckBox *CheckBox5;
TCheckBox *CheckBox6;
TCheckBox *UEPNUM1;
TCheckBox *UEPNUM0;
TButton *Button2;
TTabSheet *TabSheet3;
TBevel *Bevel2;
TBevel *Bevel7;
TBevel *Bevel8;
TBevel *Bevel9;
TBevel *Bevel10;
TLabel *Label133;
TLabel *Label134;
TLabel *Label135;
```

```
TLabel *Label136;
TLabel *Label137;
TBevel *Bevel3;
TLabel *Label138;
TLabel *Label139;
TLabel *Label140;
TBevel *Bevel4;
TLabel *Label141;
TLabel *Label142;
TLabel *Label143;
TLabel *Label144;
TLabel *Label54;
TLabel *Label55;
TLabel *Label56;
TLabel *Label57;
TLabel *Label58;
TLabel *Label59;
TLabel *Label60;
TLabel *Label61;
TLabel *Label62;
TLabel *Label63;
TLabel *Label64;
TLabel *Label65;
TLabel *Label66;
TLabel *Label67;
TLabel *Label68;
TLabel *Label69;
TLabel *Label70;
TLabel *Label71;
TLabel *Label72;
TLabel *Label73;
TLabel *Label74;
TLabel *Label75;
TCheckBox *UEPRST3;
TCheckBox *CheckBox8;
TCheckBox *CheckBox12;
TCheckBox *CheckBox13;
TCheckBox *CheckBox14;
TCheckBox *UEPRST2;
TCheckBox *UEPRST1;
TCheckBox *UEPRST0;
TBitBtn *BitBtn1;
TCheckBox *UEPDATX3;
TCheckBox *UEPDATX7;
TCheckBox *UEPDATX6;
TCheckBox *UEPDATX5;
TCheckBox *UEPDATX4;
TCheckBox *UEPDATX2;
TCheckBox *UEPDATX1;
TCheckBox *UEPDATX0;
TBitBtn *BitBtn3;
TButton *Button3;
TLabel *Label15;
TLabel *Label145;
TLabel *Label146;
TLabel *Label147;
TLabel *Label148;
TLabel *Label149;
TLabel *Label150;
TLabel *Label151;
TLabel *Label152;
TLabel *Label153;
TLabel *Label154;
TCheckBox *CheckBox26;
TCheckBox *USBCON7;
TCheckBox *USBCON6;
TCheckBox *CheckBox29;
TCheckBox *CheckBox30;
TCheckBox *CheckBox31;
```

```
TCheckBox *CheckBox32;
TCheckBox *CheckBox33;
TBitBtn *BitBtn4;
TBevel *Bevel5;
TBevel *Bevel6;
TBevel *Bevel11;
TLabel *Label155;
TLabel *Label156;
TLabel *Label157;
TLabel *Label158;
TBevel *Bevel12;
TLabel *Label159;
TLabel *Label160;
TLabel *Label161;
TLabel *USB;
TLabel *Label163;
TLabel *CLK;
TLabel *Label90;
TLabel *Label91;
TLabel *Label92;
TLabel *Label93;
TLabel *Label94;
TLabel *Label95;
TLabel *Label96;
TLabel *Label97;
TLabel *Label98;
TLabel *Label99;
TLabel *Label100;
TLabel *Label111;
TLabel *Label112;
TLabel *Label113;
TLabel *Label114;
TLabel *Label115;
TLabel *Label116;
TLabel *Label117;
TLabel *Label118;
TLabel *Label119;
TLabel *Label120;
TLabel *Label121;
TCheckBox *UEPIEN3;
TCheckBox *CheckBox15;
TCheckBox *CheckBox16;
TCheckBox *CheckBox17;
TCheckBox *CheckBox18;
TCheckBox *UEPIEN2;
TCheckBox *UEPIEN1;
TCheckBox *UEPIEN0;
TButton *Button4;
TCheckBox *UEPSTAX3;
TCheckBox *UEPSTAX7;
TCheckBox *UEPSTAX6;
TCheckBox *UEPSTAX5;
TCheckBox *UEPSTAX4;
TCheckBox *UEPSTAX2;
TCheckBox *UEPSTAX1;
TCheckBox *UEPSTAX0;
TButton *Button6;
TLabel *Label162;
TLabel *Label164;
TLabel *Label165;
TLabel *Label166;
TLabel *Label167;
TLabel *Label168;
TLabel *Label169;
TLabel *Label170;
TLabel *Label171;
TLabel *Label172;
TLabel *Label173;
TCheckBox *CheckBox27;
```

```
        TCheckBox *CheckBox28;
        TCheckBox *CheckBox34;
        TCheckBox *CheckBox35;
        TCheckBox *CheckBox36;
        TCheckBox *CheckBox37;
        TCheckBox *CheckBox38;
        TCheckBox *CheckBox39;
        TButton *Button7;
        TBevel *Bevel13;
        TBevel *Bevel15;
        TBevel *Bevel17;
        TBevel *Bevel18;
        TBevel *Bevel19;
        TLabel *Label174;
        TLabel *Label175;
        TLabel *Label176;
        TLabel *Label177;
        TLabel *Label178;
        TLabel *Label179;
        TLabel *Label180;
        TLabel *Label181;
        TLabel *Label182;
        TLabel *Label183;
        TLabel *Label184;
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall N5Click(TObject *Sender);
        void __fastcall USB1Click(TObject *Sender);
        void __fastcall BitBtn1Click(TObject *Sender);
        void __fastcall BitBtn2Click(TObject *Sender);
        void __fastcall BitBtn3Click(TObject *Sender);
        void __fastcall Timer1Timer(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall Button5Click(TObject *Sender);
        void __fastcall N7Click(TObject *Sender);
        void __fastcall Button6Click(TObject *Sender);
        void __fastcall N2Click(TObject *Sender);
        void __fastcall BitBtn4Click(TObject *Sender);

private:    // User declarations
public:           // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Файл signal.cpp – побудова часової діаграми сигналів USB-інтерфейсу

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

#include <vcl.h>
#pragma hdrstop

#include "signal.h"
#include "Main.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm4 *Form4;
int xx=1, f1=0, h=0, p=0, i0=8, i1=8, i2=8, i3=8;
AnsiString f;
//---------------------------------------------------------------------------
__fastcall TForm4::TForm4(TComponent* Owner)
        : TForm(Owner)
```

```
{
}
//---------------------------------------------------------------------------
void __fastcall TForm4::Timer1Timer(TObject *Sender)
{
Series1->Add(xx,f1,0x00FEEBB1); // виведення на екран сигналу синхронізації

if(Form1->CLK->Caption=="ввімкнено") xx=(xx+1)%2; else xx=0;

f1++;
if(f1>25) //обмеження, щоб не вийти за межі вікна осцилографа
{
f1=0;
Series1->Clear();
LineSeries1->Clear();
LineSeries2->Clear();
}

//читання даних з FIFO-буфера кінцевої точки 0
if((Form1->nt->Caption=='0')&(Form1->buf0->Lines->Count>0))
{
h=StrToInt(Form1->buf0->Lines->Strings[0][i0]);
--i0;
if(i0<1) { i0=8;  Form1->buf0->Lines->Delete(0); }
f=Form1->nn0->Caption;
}


//читання даних з FIFO-буфера кінцевої точки 1
if((Form1->nt->Caption=='1')&(Form1->buf1->Lines->Count>0))
{
h=StrToInt(Form1->buf1->Lines->Strings[0][i1]);
--i1;
if(i1<1) { i1=8;  Form1->buf1->Lines->Delete(0); }
f=Form1->nn1->Caption;
}

//читання даних з FIFO-буфера кінцевої точки 2
if((Form1->nt->Caption=='2')&(Form1->buf2->Lines->Count>0))
{
h=StrToInt(Form1->buf2->Lines->Strings[0][i2]);
--i2;
if(i2<1) { i2=8;  Form1->buf2->Lines->Delete(0); }
f=Form1->nn2->Caption;
}

//читання даних з FIFO-буфера кінцевої точки 3
if((Form1->nt->Caption=='3')&(Form1->buf3->Lines->Count>0))
{
h=StrToInt(Form1->buf3->Lines->Strings[0][i3]);
--i3;
if(i3<1) { i3=8;  Form1->buf3->Lines->Delete(0); }
f=Form1->nn3->Caption;
}

if(f=="Прийом")
{
LineSeries1->Add(p,f1,0x00E9B9F7); //виведення на екран D-
LineSeries2->Add(h,f1,0x00E9B9F7);
}
else
{
LineSeries1->Add(h,f1,0x00E9B9F7); // виведення на екран D+
LineSeries2->Add(p,f1,0x00E9B9F7);
}

//якщо буфери порожні встановити на виходах логічний нуль
if((Form1->buf0->Lines->Count==0)&(Form1->nt->Caption=='0')) h=0;
if((Form1->buf1->Lines->Count==0)&(Form1->nt->Caption=='1')) h=0;
```

```
if((Form1->buf2->Lines->Count==0)&(Form1->nt->Caption=='2')) h=0;
if((Form1->buf3->Lines->Count==0)&(Form1->nt->Caption=='3')) h=0;
}
//---------------------------------------------------------------------------

//обробник натиснення кнопки «Очистити»
void __fastcall TForm4::Button1Click(TObject *Sender)
{
f1=0;
Series1->Clear();
LineSeries1->Clear();
LineSeries2->Clear();
}
//---------------------------------------------------------------------------

void __fastcall TForm4::FormCreate(TObject *Sender)
{
Timer1->Enabled=true;
}
//---------------------------------------------------------------------------
//обробник натиснення кнопки «Приховати»
void __fastcall TForm4::Button2Click(TObject *Sender)
{
Form4->Close();
}
//---------------------------------------------------------------------------
```

Файл signal.h – бібліотека для файлу signal.cpp

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

#ifndef signalH
#define signalH
//---------------------------------------------------------------------------
#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 TForm4 : public TForm
{
__published:      // IDE-managed Components
        TChart *Chart1;
        TLineSeries *Series1;
        TTimer *Timer1;
        TChart *Chart2;
        TLineSeries *LineSeries1;
        TChart *Chart3;
        TLineSeries *LineSeries2;
        TButton *Button1;
        TButton *Button2;
        void __fastcall Timer1Timer(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall FormCreate(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm4(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm4 *Form4;
//---------------------------------------------------------------------------
#endif
```

Файл help.cpp – довідка про регістри USB-інтерфейсу

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

#include <vcl.h>
#pragma hdrstop

#include "help.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
//---------------------------------------------------------------------------
__fastcall TForm3::TForm3(TComponent* Owner)
        : TForm(Owner)
{
}

void __fastcall TForm3::Button121Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("USBCON (S:BCh) - основний керуючий регістр USB інтерфейсу.
Формат регістра:");
Memo1->Lines->Add("Біт 0 - FADDEN");
Memo1->Lines->Add("Біт 1 - CONFG");
Memo1->Lines->Add("Біт 2 - RMWUPE");
Memo1->Lines->Add("Біт 3 - UPRSM");
Memo1->Lines->Add("Біт 4 - зарезервований, =0");
Memo1->Lines->Add("Біт 5 - SDRMWUP");
Memo1->Lines->Add("Біт 6 - SUSPCLK");
Memo1->Lines->Add("Біт 7 - USBE");
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button8Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("USBE - біт вмикання модуля USB. Установка біта вмикає USB-
контролер. Скидання біта вимикає й скидає USB-контролер.");

}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button122Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("USBADDR (S:C6h) - регістр USB адреси. Формат регістра: ");
Memo1->Lines->Add("Біт 0 - UADD0");
Memo1->Lines->Add("Біт 1 - UADD1");
Memo1->Lines->Add("Біт 2 - UADD2");
Memo1->Lines->Add("Біт 3 - UADD3");
Memo1->Lines->Add("Біт 4 - UADD4");
Memo1->Lines->Add("Біт 5 - UADD5");
Memo1->Lines->Add("Біт 6 - UADD6");
Memo1->Lines->Add("Біт 7 - FEN");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button123Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("USBINT (S:BDh) - регістр прапорів основних USB переривань.
Формат регістра:");
Memo1->Lines->Add("Біт 0 - SPINT");
Memo1->Lines->Add("Біт 1 - Зарезервований, =0");
Memo1->Lines->Add("Біт 2 - Зарезервований, =0");
Memo1->Lines->Add("Біт 3 - SOFINT");
Memo1->Lines->Add("Біт 4 - EORINT");
Memo1->Lines->Add("Біт 5 - WUPCPU");
```

```
Memo1->Lines->Add("Біт 6 - Зарезервований, =0");
Memo1->Lines->Add("Біт 7 - Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button124Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("USBIEN (S:BEh) - регістр дозволів основних USB переривань.
Формат регістра:");
Memo1->Lines->Add("Біт 0 - ESPINT");
Memo1->Lines->Add("Біт 1 - Зарезервований, =0");
Memo1->Lines->Add("Біт 2 - Зарезервований, =0");
Memo1->Lines->Add("Біт 3 - ESOFINT");
Memo1->Lines->Add("Біт 4 - EEORINT");
Memo1->Lines->Add("Біт 5 - EWUPCPU");
Memo1->Lines->Add("Біт 6 - Зарезервований, =0");
Memo1->Lines->Add("Біт 7 - Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button125Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UEPNUM (S:C7h) - регістр номера USB кінцевої точки. Формат
регістра:");
Memo1->Lines->Add("Біт 0 - EPNUM0");
Memo1->Lines->Add("Біт 1 - EPNUM1");
Memo1->Lines->Add("Біт 2 - Зарезервований, =0");
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");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button126Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UEPCONX (S:D4h) - керуючий регістр кінцевої USB точки X (де X
- номер, заданий у регістрі UEPNUM. Формат регістра:");
Memo1->Lines->Add("Біт 0 - EPTYPE0");
Memo1->Lines->Add("Біт 1 - EPTYPE1");
Memo1->Lines->Add("Біт 2 - EPDIR ");
Memo1->Lines->Add("Біт 3 - DTGL ");
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 - EPEN");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button127Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UEPSTAX (S: CEh) - регістр керування й статусу кінцевої USB
точки X. Формат регістра:");
Memo1->Lines->Add("Біт 0 - TXCMP ");
Memo1->Lines->Add("Біт 1 - RXOUT ");
Memo1->Lines->Add("Біт 2 - RXSETUP ");
Memo1->Lines->Add("Біт 3 - STLCRC ");
Memo1->Lines->Add("Біт 4 - TXRDY ");
Memo1->Lines->Add("Біт 5 - STALLRQ ");
Memo1->Lines->Add("Біт 6 - Зарезервований, =0");
Memo1->Lines->Add("Біт 7 - DIR ");
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm3::Button128Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UEPRST (S:D5h) - регістр дозволів основних USB переривань.
Формат регістра:");
Memo1->Lines->Add("Біт 0 - EP0RST ");
Memo1->Lines->Add("Біт 1 - EP1RST ");
Memo1->Lines->Add("Біт 2 - EP2RST ");
Memo1->Lines->Add("Біт 3 - EP3RST ");
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 ");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button129Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UEPINT (S:F8h) - (тільки читання) регістр переривань кінцевих
USB точок.  Формат регістра:");
Memo1->Lines->Add("Біт 0 - EP0INT ");
Memo1->Lines->Add("Біт 1 - EP1INT ");
Memo1->Lines->Add("Біт 2 - EP2INT ");
Memo1->Lines->Add("Біт 3 - EP3INT ");
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 ");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button130Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UEPIEN (S:C2h) - регістр дозволів переривань кінцевих USB
точок. Формат регістра: ");
Memo1->Lines->Add("Біт 0 - EP0INTE ");
Memo1->Lines->Add("Біт 1 - EP1INTE ");
Memo1->Lines->Add("Біт 2 - EP2INTE ");
Memo1->Lines->Add("Біт 3 - EP3INTE ");
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 ");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button131Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UEPDATX (S:CFh) - регістр даних FIFO буфера кінцевої USB
точки X (X- номер встановлений у регістрі UEPNUM). Формат регістра:");
Memo1->Lines->Add("Біт 0 - FADAT0");
Memo1->Lines->Add("Біт 1 - FADAT1");
Memo1->Lines->Add("Біт 2 - FADAT2 ");
Memo1->Lines->Add("Біт 3 - FADAT3");
Memo1->Lines->Add("Біт 4 - FADAT4");
Memo1->Lines->Add("Біт 5 - FADAT5");
Memo1->Lines->Add("Біт 6 - FADAT6");
Memo1->Lines->Add("Біт 7 - FADAT7");
Memo1->Lines->Add("Після скидання регістр приймає значення XXh.");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button132Click(TObject *Sender)
{
Memo1->Clear();
```

```
Memo1->Lines->Add("UBYCTLX (S: E2h) - регістр лічильника байтів кінцевої USB
точки X ( X - номер встановлений у регістрі UEPNUM). Формат регістра:");
Memo1->Lines->Add("Біт 0 - BYCT0");
Memo1->Lines->Add("Біт 1 - BYCT1");
Memo1->Lines->Add("Біт 2 - BYCT2");
Memo1->Lines->Add("Біт 3 - BYCT3");
Memo1->Lines->Add("Біт 4 - BYCT4");
Memo1->Lines->Add("Біт 5 - BYCT5");
Memo1->Lines->Add("Біт 6 - BYCT6");
Memo1->Lines->Add("Біт 7 - Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button133Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UFNUML (S:BFh - тільки читання) - регістр молодших бітів
номера USB кадру. Формат регістра:");
Memo1->Lines->Add("Біт 0 - FNUM0");
Memo1->Lines->Add("Біт 1 - FNUM1");
Memo1->Lines->Add("Біт 2 - FNUM2");
Memo1->Lines->Add("Біт 3 - FNUM3");
Memo1->Lines->Add("Біт 4 - FNUM4");
Memo1->Lines->Add("Біт 5 - FNUM5");
Memo1->Lines->Add("Біт 6 - FNUM6");
Memo1->Lines->Add("Біт 7 - FNUM7");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button134Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UFNUMH (S:BBh - тільки читання) - регістр старших битов
номера USB кадру. Формат регістра:");
Memo1->Lines->Add("Біт 0 - FNUM8");
Memo1->Lines->Add("Біт 1 - FNUM9");
Memo1->Lines->Add("Біт 2 - FNUM10");
Memo1->Lines->Add("Біт 3 - Зарезервований, =0 ");
Memo1->Lines->Add("Біт 4 - CRCERR");
Memo1->Lines->Add("Біт 5 - CRCOK ");
Memo1->Lines->Add("Біт 6 - Зарезервований, =0 ");
Memo1->Lines->Add("Біт 7 - Зарезервований, =0 ");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button135Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("SBCLK (S:EAh) - регістр дільника USB контролера
синхронізації. Формат регістра:");
Memo1->Lines->Add("Біт 0 - USBCD0");
Memo1->Lines->Add("Біт 1 - USBCD1");
Memo1->Lines->Add("Біт 2 - Зарезервований, =0");
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 ");
}
//---------------------------------------------------------------------------


void __fastcall TForm3::Button7Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("SUSPCLK - біт припинення синхронізації USB. Установка біта
відключає вхід 48 МГц синхроімпульсів (Продовження детектування усе ще можливе).
Скидання включає вхід 48 МГц синхроімпульсів.");
```

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

void __fastcall TForm3::Button6Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("SDRMWUP - біт передачі віддаленого пробудження.
Встановлюється для виклику зовнішнього переривання USB контролера при
віддаленому пробудженні. Резюме вихідного потоку передається тільки якщо біт
RMWUPE встановлений, всі USB синхроімпульси активізовані й USB шина перебувала в
стані припинення (SUSPEND) не менш 5 мс. Скидається програмно.");

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

void __fastcall TForm3::Button5Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button4Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UPRSM - біт резюме вихідного потоку (тільки читання).
Встановлюється апаратно після установки біта SDRMWUP якщо біт RMWUPE був також
встановлений. Скидається апаратно після передачі резюме вихідного потоку.");

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

void __fastcall TForm3::Button3Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("RMWUPE - біт дозволу віддаленого пробудження. Встановлюється
для дозволу запиту резюме вихідного потоку провідного пристрою. Скидається після
відображення резюме вихідного потоку в RSMINPR. Зауваження: не встановлюйте цей
біт якщо в провідного пристрою для приладу не встановлена функція
DEVICE_REMOTE_WAKEUP.");

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

void __fastcall TForm3::Button2Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("CONFG - конфігураційний біт. Встановлюється після коректної
обробки запиту SET_CONFIGURATION з ненульовим значенням. Скидається програмно
після одержання запиту SET_CONFIGURATION з нульовим значенням. Скидається
апаратно при апаратному скиданні або після виявлення USB скидання на шині.");

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

void __fastcall TForm3::Button1Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FADDEN - біт дозволу функції адресації. Встановлюється
програмним забезпеченням приладу після успішного фазування статусу транзакції
SET_ADDRESS. Надалі він не повинен скидатися програмно. Скидається апаратно при
апаратному скиданні або після виявлення USB скидання на шині. Коли цей біт
скинутий, використовується функція адресації за замовчуванням (0).");

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

void __fastcall TForm3::Button16Click(TObject *Sender)
{
```

```
Memo1->Clear();
Memo1->Lines->Add("FEN - біт активізації функції. Встановлюється для активізації
функції. Програмне забезпечення приладу встановить цей біт після прийому USB
скидання й візьме участь у поточному конфігураційному процесі із установленою за
замовчуванням адресою (FEN скинеться в 0).");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button15Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UADD6:0 - біти USB адреси. Ці біти містять задану за
замовчуванням адресу (0) після включення живлення або скидання USB шини. Запис
їхнього стану відбудеться після прийняття програмним забезпеченням приладу
запиту SET_ADDRESS.");

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

void __fastcall TForm3::Button14Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UADD6:0 - біти USB адреси. Ці біти містять задану за
замовчуванням адресу (0) після включення живлення або скидання USB шини. Запис
їхнього стану відбудеться після прийняття програмним забезпеченням приладу
запиту SET_ADDRESS.");

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

void __fastcall TForm3::Button13Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UADD6:0 - біти USB адреси. Ці біти містять задану за
замовчуванням адресу (0) після включення живлення або скидання USB шини. Запис
їхнього стану відбудеться після прийняття програмним забезпеченням приладу
запиту SET_ADDRESS.");

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

void __fastcall TForm3::Button12Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UADD6:0 - біти USB адреси. Ці біти містять задану за
замовчуванням адресу (0) після включення живлення або скидання USB шини. Запис
їхнього стану відбудеться після прийняття програмним забезпеченням приладу
запиту SET_ADDRESS.");

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

void __fastcall TForm3::Button11Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UADD6:0 - біти USB адреси. Ці біти містять задану за
замовчуванням адресу (0) після включення живлення або скидання USB шини. Запис
їхнього стану відбудеться після прийняття програмним забезпеченням приладу
запиту SET_ADDRESS.");

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

void __fastcall TForm3::Button10Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UADD6:0 - біти USB адреси. Ці біти містять задану за
замовчуванням адресу (0) після включення живлення або скидання USB шини. Запис
```

```
їхнього стану відбудеться після прийняття програмним забезпеченням приладу
запиту SET_ADDRESS.");

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

void __fastcall TForm3::Button9Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("UADD6:0 - біти USB адреси. Ці біти містять задану за
замовчуванням адресу (0) після включення живлення або скидання USB шини. Запис
їхнього стану відбудеться після прийняття програмним забезпеченням приладу
запиту SET_ADDRESS.");

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

void __fastcall TForm3::Button24Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button23Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button22Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("WUPCPU - прапор переривання пробудження ЦП. Встановлюється
апаратно коли контролер, що перебуває в режимі SUSPEND USB перезапускається
сигналом non-idle USB шини (але не резюме вихідного потоку). Установка цього
біта викликає USB переривання коли встановлений біт EWUPCPU у регістрі USBIEN.
Скидається програмно після перемикання всіх USB синхроімпульсів.");

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

void __fastcall TForm3::Button21Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EORINT - прапор переривання закінчення скидання.
Встановлюється апаратно при виявленні USB контролером закінчення скидання.
Установка цього біта викликає USB переривання коли встановлений біт EEORINT у
регістрі USBIEN. Скидається програмно.");

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

void __fastcall TForm3::Button20Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("SOFINT - прапор переривання при виявленні початку кадру.
Встановлюється апаратно після прийому USB пакета початку кадру (SOF). Установка
цього біта викликає USB переривання коли встановлений біт ESOFINT у регістрі
USBIEN. Скидається програмно.");

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

void __fastcall TForm3::Button19Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
```

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

void __fastcall TForm3::Button18Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button17Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("SPINT - прапор переривання при припиненні. Встановлюється
апаратно при виявленні USB припинення (шина не зайнята протягом трьох кадрових
періодів: J стан протягом 3 мс). Установка цього біта викликає USB переривання
коли встановлений біт ESPINT у регістрі USBIEN. Скидається програмно.");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button32Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button31Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button30Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EWUPCPU - біт дозволу переривання при пробудженні ЦП.
Установка цього біта дозволяє переривання при пробудженні ЦП. Скидання біта
забороняє переривання при пробудженні ЦП.");

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

void __fastcall TForm3::Button29Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EEORINT - біт дозволу переривання по закінченню скидання.
Установка цього біта дозволяє переривання по закінченню скидання. Скидання цього
біта забороняє переривання по закінченню скидання.");

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

void __fastcall TForm3::Button28Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("ESOFINT - біт дозволу переривання при виявленні початку
кадру. Установка цього біта дозволяє переривання при виявленні початку кадру.
Скидання цього біта забороняє переривання при виявленні початку кадру.");

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

void __fastcall TForm3::Button27Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
```

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

void __fastcall TForm3::Button26Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button25Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("ESPINT - біт дозволу переривання при виявленні припинення.
Установка цього біта дозволяє переривання при виявленні припинення. Скидання
цього біта забороняє переривання при виявленні припинення.");

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

void __fastcall TForm3::Button33Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EPNUM1:0 - біти номера кінцевої точки. Задають номер кінцевої
точки, до якої буде відбуватися звертання при зчитуванні й записі регістрів
UEPSTAX, UEPDATX, UBYCTLX або UEPCONX.");

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

void __fastcall TForm3::Button34Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EPNUM1:0 - біти номера кінцевої точки. Задають номер кінцевої
точки, до якої буде відбуватися звертання при зчитуванні й записі регістрів
UEPSTAX, UEPDATX, UBYCTLX або UEPCONX.");

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

void __fastcall TForm3::Button35Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button36Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button37Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button38Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button39Click(TObject *Sender)
{
```

```
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button40Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button48Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EPEN - біт активізації кінцевої точки. Установка біта включає
кінцеву точку відповідно до конфігурації приладу. Нульова кінцева точка завжди
активізується після апаратного скидання або скидання USB шини й бере участь у
конфігурації приладу. Скидання біта відключає кінцеву точку відповідно до
конфігурації приладу.");

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

void __fastcall TForm3::Button47Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button46Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button45Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button44Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("DTGL - біт зміни статусу даних (тільки читання).
Встановлюється апаратно при прийманні пакета DATA1. Скидається апаратно при
прийманні пакета DATA0.");

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

void __fastcall TForm3::Button43Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EPDIR - біт установки спрямованості кінцевих точок. Установка
біта встановлює пакетні, ізохронні та кінцеві точки переривань в режим прийому.
Скидання біта встановлює пакетні, ізохронні та кінцеві точки переривань в режим
передачі. Біт не впливає на керуючі кінцеві точки.");

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

void __fastcall TForm3::Button42Click(TObject *Sender)
{
Memo1->Clear();
```

```
Memo1->Lines->Add("EPTYPE1:0 - біти установки типу кінцевих точок. Ці біти
дозволяють установити для кінцевої точки один з наступних типів (для нульової
кінцевої точки завжди повинен бути встановлений ''керуючий'' тип):");
Memo1->Lines->Add("0    0  -  керуюча кінцева точка;");
Memo1->Lines->Add("0    1  -  ізохронна кінцева точка;");
Memo1->Lines->Add("1    0  -  пакетна кінцева точка;");
Memo1->Lines->Add("1    1  -  кінцева точка переривань.");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button41Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EPTYPE1:0 - біти установки типу кінцевих точок. Ці біти
дозволяють установити для кінцевої точки один з наступних типів (для нульової
кінцевої точки завжди повинен бути встановлений ''керуючий'' тип):");
Memo1->Lines->Add("0    0  -  керуюча кінцева точка;");
Memo1->Lines->Add("0    1  -  ізохронна кінцева точка;");
Memo1->Lines->Add("1    0  -  пакетна кінцева точка;");
Memo1->Lines->Add("1    1  -  кінцева точка переривань.");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button56Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("DIR - біт керування напрямком кінцевої точки. Цей біт
враховується тільки тоді, коли кінцевій точці привласнений тип контрольної.
Повинен бути встановлений для стадії даних. Для інших випадків повинен бути
скинутий. Зауваження: Цей біт повинен бути встановлений при RXSETUP перериванні
до зміни стану будь-якого іншого біта. Також він визначає фазу статусу (IN для
перевірки запису й OUT для контролю читання). Цей біт повинен скинутий для
стадії статусу контрольної вихідної транзакції.");

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

void __fastcall TForm3::Button55Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button54Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("STALLRQ - біт запиту зупинки встановлення зв'язку. Установка
біта приведе до посилки відповіді STALL (зупинки) провідному пристрою для
наступної установки зв'язку. У протилежному випадку цей біт повинен бути
скинутий.");

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

void __fastcall TForm3::Button53Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("TXRDY - керуючий біт готовності передачі пакета. Біт повинен
бути встановлений після запису пакета в FIFO буфер кінцевої точки для передачі
IN даних. Дані повинні записуватися в FIFO буфер кінцевої точки тільки після
скидання цього біта. Установка цього біта без запису даних в FIFO буфер приведе
до посилки пакета нульової довжини, що рекомендується в загальному випадку й
може знадобитися для позначення передачі в тих випадках, коли довжина останнього
пакета даних дорівнює MaxPacketSize (наприклад, для контрольного зчитування
передачі). Скидається апаратно відразу після посилки пакета ізохронної кінцевої
точки або після одержання контрольної, пакетної чи кінцевої точки переривання
підтвердження від провідного пристрою.");
```

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

void __fastcall TForm3::Button52Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("STLCRC -  прапор переривання при припиненні посилки/Прапор
переривання при виявленні CRC помилки. Для контрольних, пакетних і кінцевих
точок переривань:  Встановлюється апаратно після посилки за допомогою STALLRQ
запиту припинення встановлення зв'язку. Після цього відбудеться переривання
кінцевої точки, якщо воно дозволено в регістрі UEPIEN. Скидається апаратно після
прийому пакета SETUP (див. опис біта RXSETUP). Для ізохронних кінцевих точок:
Встановлюється апаратно при виявленні помилки в прийнятих даних (CRC помилка в
прийнятих даних). Після цього відбудеться переривання кінцевої точки якщо воно
дозволено в регістрі UEPIEN. Скидається апаратно після прийому неушкоджених
даних.");

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

void __fastcall TForm3::Button51Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("RXSETUP - прапор переривання при одержанні пакета SETUP.
Встановлюється апаратно після одержання припустимого пакета SETUP від провідного
пристрою. Після цього відбудеться переривання, якщо воно дозволено в регістрі
UEPIEN. Скидається програмно після зчитування SETUP даних з FIFO буфера кінцевої
точки.");

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

void __fastcall TForm3::Button50Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("RXOUT - прапор переривання при прийнятті вихідних даних.
Встановлюється апаратно після прийняття вихідного пакета. Після цього
відбудеться переривання, якщо воно дозволено в регістрі UEPIEN і всі поточні
вихідні пакети відхилені до скидання цього біта. Однак у керуючих кінцевих
точках раніше прийнята SETUP транзакція може переписати вміст FIFO буфера
кінцевої точки, навіть якщо був прийнятий пакет даних при встановленому цьому
прапорі. Скидається програмно після зчитування вихідних даних з FIFO буфера
кінцевої точки.");

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

void __fastcall TForm3::Button49Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("TXCMP - прапор переривання по закінченню передачі вхідних
даних  Встановлюється апаратно після передачі вхідного пакета ізохронною точкою
або після одержання підтвердження прийому (ACK) від провідного пристрою
контрольної, пакетної або кінцевої точки переривання. Після цього відбудеться
переривання, якщо воно дозволено в регістрі UEPIEN. Скидається програмно перед
наступною установкою біта TXRDY.");

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

void __fastcall TForm3::Button64Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button63Click(TObject *Sender)
{
```

```
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button62Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button61Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button60Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EP3RST - скидання FIFO буфера третьої кінцевої точки.
Необхідно встановити й скинути для скидання FIFO буфера третьої кінцевої точки
перед початком будь-якої операції до апаратного скидання або при одержанні
скидання USB шини.");

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

void __fastcall TForm3::Button59Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EP1RST - скидання FIFO буфера першої кінцевої точки.
Необхідно встановити й скинути для скидання FIFO буфера першої кінцевої точки
перед початком будь-якої операції до апаратного скидання або при одержанні
скидання USB шини.");

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

void __fastcall TForm3::Button58Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EP1RST - скидання FIFO буфера першої кінцевої точки.
Необхідно встановити й скинути для скидання FIFO буфера першої кінцевої точки
перед початком будь-якої операції до апаратного скидання або при одержанні
скидання USB шини.");

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

void __fastcall TForm3::Button57Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EP0RST - скидання FIFO буфера нульової кінцевої точки.
Необхідно встановити й скинути для скидання FIFO буфера нульової кінцевої точки
перед початком будь-якої операції до апаратного скидання або при одержанні
скидання USB шини.");

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

void __fastcall TForm3::Button72Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm3::Button71Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button70Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button69Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button68Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EP3INT - прапор переривання від третьої кінцевої точки.
Встановлюється апаратно після установки переривання в регістрі UEPSTAX і якщо
переривання від третьої кінцевої точки дозволено в регістрі UEPIEN. Повинен бути
скинутий програмно.");

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

void __fastcall TForm3::Button67Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EP2INT - прапор переривання від другої кінцевої точки.
Встановлюється апаратно після установки переривання в регістрі UEPSTAX і якщо
переривання від другої кінцевої точки дозволено в регістрі UEPIEN. Повинен бути
скинутий програмно.");

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

void __fastcall TForm3::Button66Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EP1INT - прапор переривання від першої кінцевої точки.
Встановлюється апаратно після установки переривання в регістрі UEPSTAX і якщо
переривання від першої кінцевої точки дозволено в регістрі UEPIEN. Повинен бути
скинутий програмно.");

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

void __fastcall TForm3::Button65Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EP0INT - прапор переривання від нульовий кінцевої точки.
Встановлюється апаратно після установки переривання в регістрі UEPSTAX і якщо
переривання від нульової кінцевої точки дозволено в регістрі UEPIEN. Повинен
бути скинутий програмно.");

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

void __fastcall TForm3::Button80Click(TObject *Sender)
{
Memo1->Lines->Add("Зарезервований, =0");
```

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

void __fastcall TForm3::Button79Click(TObject *Sender)
{
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button78Click(TObject *Sender)
{
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button77Click(TObject *Sender)
{
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button76Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EP3INTE - біт дозволу переривання третьої кінцевої точки.
Установка дозволяє переривання від третьої кінцевої точки. Скидання забороняє
переривання від третьої кінцевої точки.");

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

void __fastcall TForm3::Button75Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EP2INTE - біт дозволу переривання другої кінцевої точки.
Установка дозволяє переривання від другої кінцевої точки. Скидання забороняє
переривання від  другої кінцевої точки.");

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

void __fastcall TForm3::Button74Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EP1INTE - біт дозволу переривання першої кінцевої точки.
Установка дозволяє переривання від першої кінцевої точки. Скидання забороняє
переривання від першої кінцевої точки.");

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

void __fastcall TForm3::Button73Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("EP0INTE - біт дозволу переривання нульової кінцевої точки.
Установка дозволяє переривання від нульової кінцевої точки. Скидання забороняє
переривання від нульової кінцевої точки.");

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

void __fastcall TForm3::Button88Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FADAT7:0 - дані FIFO буфера кінцевої точки X. Байт даних,
записуваний в FIFO буфер або зчитувальний з FIFO буфера кінцевої точки X (див.
регістр EPNUM).");

}
```

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

void __fastcall TForm3::Button87Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FADAT7:0 - дані FIFO буфера кінцевої точки X. Байт даних,
записуваний в FIFO буфер або зчитувальний з FIFO буфера кінцевої точки X (див.
регістр EPNUM).");

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

void __fastcall TForm3::Button86Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FADAT7:0 - дані FIFO буфера кінцевої точки X. Байт даних,
записуваний в FIFO буфер або зчитувальний з FIFO буфера кінцевої точки X (див.
регістр EPNUM).");

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

void __fastcall TForm3::Button85Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FADAT7:0 - дані FIFO буфера кінцевої точки X. Байт даних,
записуваний в FIFO буфер або зчитувальний з FIFO буфера кінцевої точки X (див.
регістр EPNUM).");

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

void __fastcall TForm3::Button84Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FADAT7:0 - дані FIFO буфера кінцевої точки X. Байт даних,
записуваний в FIFO буфер або зчитувальний з FIFO буфера кінцевої точки X (див.
регістр EPNUM).");

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

void __fastcall TForm3::Button83Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FADAT7:0 - дані FIFO буфера кінцевої точки X. Байт даних,
записуваний в FIFO буфер або зчитувальний з FIFO буфера кінцевої точки X (див.
регістр EPNUM).");

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

void __fastcall TForm3::Button82Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FADAT7:0 - дані FIFO буфера кінцевої точки X. Байт даних,
записуваний в FIFO буфер або зчитувальний з FIFO буфера кінцевої точки X (див.
регістр EPNUM).");

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

void __fastcall TForm3::Button81Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FADAT7:0 - дані FIFO буфера кінцевої точки X. Байт даних,
записуваний в FIFO буфер або зчитувальний з FIFO буфера кінцевої точки X (див.
регістр EPNUM).");
```

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

void __fastcall TForm3::Button96Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button95Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("BYCT6:0 - лічильник байтів. Лічильник байтів прийнятих
пакетів даних. Значення цього лічильника байтів дорівнює кількості байтів даних,
прийнятих після одержання ідентифікатора (PID) даних.");

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

void __fastcall TForm3::Button94Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("BYCT6:0 - лічильник байтів. Лічильник байтів прийнятих
пакетів даних. Значення цього лічильника байтів дорівнює кількості байтів даних,
прийнятих після одержання ідентифікатора (PID) даних.");

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

void __fastcall TForm3::Button93Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("BYCT6:0 - лічильник байтів. Лічильник байтів прийнятих
пакетів даних. Значення цього лічильника байтів дорівнює кількості байтів даних,
прийнятих після одержання ідентифікатора (PID) даних.");

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

void __fastcall TForm3::Button92Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("BYCT6:0 - лічильник байтів. Лічильник байтів прийнятих
пакетів даних. Значення цього лічильника байтів дорівнює кількості байтів даних,
прийнятих після одержання ідентифікатора (PID) даних.");

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

void __fastcall TForm3::Button91Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("BYCT6:0 - лічильник байтів. Лічильник байтів прийнятих
пакетів даних. Значення цього лічильника байтів дорівнює кількості байтів даних,
прийнятих після одержання ідентифікатора (PID) даних.");

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

void __fastcall TForm3::Button90Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("BYCT6:0 - лічильник байтів. Лічильник байтів прийнятих
пакетів даних. Значення цього лічильника байтів дорівнює кількості байтів даних,
прийнятих після одержання ідентифікатора (PID) даних.");

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

```
void __fastcall TForm3::Button89Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("BYCT6:0 - лічильник байтів. Лічильник байтів прийнятих
пакетів даних. Значення цього лічильника байтів дорівнює кількості байтів даних,
прийнятих після одержання ідентифікатора (PID) даних.");

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

void __fastcall TForm3::Button104Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FNUM7:0 - номер кадру. Молодші 8 біт 11- бітного номера
кадру.");

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

void __fastcall TForm3::Button103Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FNUM7:0 - номер кадру. Молодші 8 біт 11- бітного номера
кадру.");

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

void __fastcall TForm3::Button102Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FNUM7:0 - номер кадру. Молодші 8 біт 11- бітного номера
кадру.");

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

void __fastcall TForm3::Button101Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FNUM7:0 - номер кадру. Молодші 8 біт 11- бітного номера
кадру.");

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

void __fastcall TForm3::Button100Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FNUM7:0 - номер кадру. Молодші 8 біт 11- бітного номера
кадру.");

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

void __fastcall TForm3::Button99Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FNUM7:0 - номер кадру. Молодші 8 біт 11- бітного номера
кадру.");

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

void __fastcall TForm3::Button98Click(TObject *Sender)
{
Memo1->Clear();
```

```
Memo1->Lines->Add("FNUM7:0 - номер кадру. Молодші 8 біт 11- бітного номера
кадру.");

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

void __fastcall TForm3::Button97Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FNUM7:0 - номер кадру. Молодші 8 біт 11- бітного номера
кадру.");

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

void __fastcall TForm3::Button120Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0 ");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button119Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0 ");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button118Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0 ");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button117Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0 ");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button116Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0 ");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button115Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0 ");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button113Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("USBCD1:0 - Дільник USB контролера синхронізації. 2-розрядний
дільник для формування синхроімпульсів USB контролером синхронізації.");

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

void __fastcall TForm3::Button114Click(TObject *Sender)
{
Memo1->Clear();
```

```
Memo1->Lines->Add("USBCD1:0 - Дільник USB контролера синхронізації. 2-розрядний
дільник для формування синхроімпульсів USB контролером синхронізації.");

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

void __fastcall TForm3::Button112Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0 ");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button111Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0 ");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button110Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("CRCOK - Біт відсутності CRC помилки прийнятого номера кадру.
Встановлюється апаратно після прийняття неушкодженого номера кадру в стартовому
або кадровому пакеті. Обновляється після кожного прийняття стартового або
кадрового пакета. Зауваження: Переривання при прийнятті початку кадру
генерується відразу після одержання PID.");

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

void __fastcall TForm3::Button109Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("CRCERR - Біт наявності CRC помилки прийнятого номера кадру.
Встановлюється апаратно після прийняття ушкодженого номера кадру в стартовому
або кадровому пакеті. Обновляється після кожного прийняття стартового або
кадрового пакета. Зауваження: Переривання при прийнятті початку кадру
генерується відразу після одержання PID.");

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

void __fastcall TForm3::Button108Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("Зарезервований, =0 ");
}
//---------------------------------------------------------------------------

void __fastcall TForm3::Button107Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FNUM10:8 - Номер кадру. Старші 3 біти 11-бітного номера
кадру. Вони доступні в останньому прийнятому SOF пакеті. Біти FNUM не
змінюються, якщо прийнято ушкоджений SOF. ");

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

void __fastcall TForm3::Button106Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FNUM10:8 - Номер кадру. Старші 3 біти 11-бітного номера
кадру. Вони доступні в останньому прийнятому SOF пакеті. Біти FNUM не
змінюються, якщо прийнято ушкоджений SOF. ");

}
```

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

void __fastcall TForm3::Button105Click(TObject *Sender)
{
Memo1->Clear();
Memo1->Lines->Add("FNUM10:8 - Номер кадру. Старші 3 біти 11-бітного номера
кадру. Вони доступні в останньому прийнятому SOF пакеті. Біти FNUM не
змінюються, якщо прийнято ушкоджений SOF. ");

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

Файл help.h – бібліотека для файлу help.cpp

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

#ifndef helpH
#define helpH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm3 : public TForm
{
__published:      // IDE-managed Components
        TLabel *Label16;
        TButton *Button1;
        TButton *Button2;
        TButton *Button3;
        TButton *Button4;
        TButton *Button5;
        TButton *Button6;
        TButton *Button7;
        TButton *Button8;
        TButton *Button9;
        TButton *Button10;
        TButton *Button11;
        TButton *Button12;
        TButton *Button13;
        TButton *Button14;
        TButton *Button15;
        TButton *Button16;
        TButton *Button17;
        TButton *Button18;
        TButton *Button19;
        TButton *Button20;
        TButton *Button21;
        TButton *Button22;
        TButton *Button23;
        TButton *Button24;
        TButton *Button25;
        TButton *Button26;
        TButton *Button27;
        TButton *Button28;
        TButton *Button29;
        TButton *Button30;
        TButton *Button31;
        TButton *Button32;
        TButton *Button33;
        TButton *Button34;
        TButton *Button35;
        TButton *Button36;
        TButton *Button37;
        TButton *Button38;
        TButton *Button39;
        TButton *Button40;
        TButton *Button41;
        TButton *Button42;
        TButton *Button43;
        TButton *Button44;
        TButton *Button45;
        TButton *Button46;
        TButton *Button47;
        TButton *Button48;
        TButton *Button49;
```

```
TButton *Button50;
TButton *Button51;
TButton *Button52;
TButton *Button53;
TButton *Button54;
TButton *Button55;
TButton *Button56;
TButton *Button57;
TButton *Button58;
TButton *Button59;
TButton *Button60;
TButton *Button61;
TButton *Button62;
TButton *Button63;
TButton *Button64;
TButton *Button65;
TButton *Button66;
TButton *Button67;
TButton *Button68;
TButton *Button69;
TButton *Button70;
TButton *Button71;
TButton *Button72;
TButton *Button73;
TButton *Button74;
TButton *Button75;
TButton *Button76;
TButton *Button77;
TButton *Button78;
TButton *Button79;
TButton *Button80;
TButton *Button81;
TButton *Button82;
TButton *Button83;
TButton *Button84;
TButton *Button85;
TButton *Button86;
TButton *Button87;
TButton *Button88;
TButton *Button89;
TButton *Button90;
TButton *Button91;
TButton *Button92;
TButton *Button93;
TButton *Button94;
TButton *Button95;
TButton *Button96;
TButton *Button97;
TButton *Button98;
TButton *Button99;
TButton *Button100;
TButton *Button101;
TButton *Button102;
TButton *Button103;
TButton *Button104;
TButton *Button105;
TButton *Button106;
TButton *Button107;
TButton *Button108;
TButton *Button109;
TButton *Button110;
TButton *Button111;
TButton *Button112;
TButton *Button113;
TButton *Button114;
TButton *Button115;
TButton *Button116;
TButton *Button117;
TButton *Button118;
```

```
TButton *Button119;
TButton *Button120;
TMemo *Memo1;
TButton *Button121;
TButton *Button122;
TButton *Button123;
TButton *Button124;
TButton *Button125;
TButton *Button126;
TButton *Button127;
TButton *Button128;
TButton *Button129;
TButton *Button130;
TButton *Button131;
TButton *Button132;
TButton *Button133;
TButton *Button134;
TButton *Button135;
void __fastcall Button121Click(TObject *Sender);
void __fastcall Button8Click(TObject *Sender);
void __fastcall Button122Click(TObject *Sender);
void __fastcall Button123Click(TObject *Sender);
void __fastcall Button124Click(TObject *Sender);
void __fastcall Button125Click(TObject *Sender);
void __fastcall Button126Click(TObject *Sender);
void __fastcall Button127Click(TObject *Sender);
void __fastcall Button128Click(TObject *Sender);
void __fastcall Button129Click(TObject *Sender);
void __fastcall Button130Click(TObject *Sender);
void __fastcall Button131Click(TObject *Sender);
void __fastcall Button132Click(TObject *Sender);
void __fastcall Button133Click(TObject *Sender);
void __fastcall Button134Click(TObject *Sender);
void __fastcall Button135Click(TObject *Sender);
void __fastcall Button7Click(TObject *Sender);
void __fastcall Button6Click(TObject *Sender);
void __fastcall Button5Click(TObject *Sender);
void __fastcall Button4Click(TObject *Sender);
void __fastcall Button3Click(TObject *Sender);
void __fastcall Button2Click(TObject *Sender);
void __fastcall Button1Click(TObject *Sender);
void __fastcall Button16Click(TObject *Sender);
void __fastcall Button15Click(TObject *Sender);
void __fastcall Button14Click(TObject *Sender);
void __fastcall Button13Click(TObject *Sender);
void __fastcall Button12Click(TObject *Sender);
void __fastcall Button11Click(TObject *Sender);
void __fastcall Button10Click(TObject *Sender);
void __fastcall Button9Click(TObject *Sender);
void __fastcall Button24Click(TObject *Sender);
void __fastcall Button23Click(TObject *Sender);
void __fastcall Button22Click(TObject *Sender);
void __fastcall Button21Click(TObject *Sender);
void __fastcall Button20Click(TObject *Sender);
void __fastcall Button19Click(TObject *Sender);
void __fastcall Button18Click(TObject *Sender);
void __fastcall Button17Click(TObject *Sender);
void __fastcall Button32Click(TObject *Sender);
void __fastcall Button31Click(TObject *Sender);
void __fastcall Button30Click(TObject *Sender);
void __fastcall Button29Click(TObject *Sender);
void __fastcall Button28Click(TObject *Sender);
void __fastcall Button27Click(TObject *Sender);
void __fastcall Button26Click(TObject *Sender);
void __fastcall Button25Click(TObject *Sender);
void __fastcall Button33Click(TObject *Sender);
void __fastcall Button34Click(TObject *Sender);
void __fastcall Button35Click(TObject *Sender);
void __fastcall Button36Click(TObject *Sender);
```

```
void __fastcall Button37Click(TObject *Sender);
void __fastcall Button38Click(TObject *Sender);
void __fastcall Button39Click(TObject *Sender);
void __fastcall Button40Click(TObject *Sender);
void __fastcall Button48Click(TObject *Sender);
void __fastcall Button47Click(TObject *Sender);
void __fastcall Button46Click(TObject *Sender);
void __fastcall Button45Click(TObject *Sender);
void __fastcall Button44Click(TObject *Sender);
void __fastcall Button43Click(TObject *Sender);
void __fastcall Button42Click(TObject *Sender);
void __fastcall Button41Click(TObject *Sender);
void __fastcall Button56Click(TObject *Sender);
void __fastcall Button55Click(TObject *Sender);
void __fastcall Button54Click(TObject *Sender);
void __fastcall Button53Click(TObject *Sender);
void __fastcall Button52Click(TObject *Sender);
void __fastcall Button51Click(TObject *Sender);
void __fastcall Button50Click(TObject *Sender);
void __fastcall Button49Click(TObject *Sender);
void __fastcall Button64Click(TObject *Sender);
void __fastcall Button63Click(TObject *Sender);
void __fastcall Button62Click(TObject *Sender);
void __fastcall Button61Click(TObject *Sender);
void __fastcall Button60Click(TObject *Sender);
void __fastcall Button59Click(TObject *Sender);
void __fastcall Button58Click(TObject *Sender);
void __fastcall Button57Click(TObject *Sender);
void __fastcall Button72Click(TObject *Sender);
void __fastcall Button71Click(TObject *Sender);
void __fastcall Button70Click(TObject *Sender);
void __fastcall Button69Click(TObject *Sender);
void __fastcall Button68Click(TObject *Sender);
void __fastcall Button67Click(TObject *Sender);
void __fastcall Button66Click(TObject *Sender);
void __fastcall Button65Click(TObject *Sender);
void __fastcall Button80Click(TObject *Sender);
void __fastcall Button79Click(TObject *Sender);
void __fastcall Button78Click(TObject *Sender);
void __fastcall Button77Click(TObject *Sender);
void __fastcall Button76Click(TObject *Sender);
void __fastcall Button75Click(TObject *Sender);
void __fastcall Button74Click(TObject *Sender);
void __fastcall Button73Click(TObject *Sender);
void __fastcall Button88Click(TObject *Sender);
void __fastcall Button87Click(TObject *Sender);
void __fastcall Button86Click(TObject *Sender);
void __fastcall Button85Click(TObject *Sender);
void __fastcall Button84Click(TObject *Sender);
void __fastcall Button83Click(TObject *Sender);
void __fastcall Button82Click(TObject *Sender);
void __fastcall Button81Click(TObject *Sender);
void __fastcall Button96Click(TObject *Sender);
void __fastcall Button95Click(TObject *Sender);
void __fastcall Button94Click(TObject *Sender);
void __fastcall Button93Click(TObject *Sender);
void __fastcall Button92Click(TObject *Sender);
void __fastcall Button91Click(TObject *Sender);
void __fastcall Button90Click(TObject *Sender);
void __fastcall Button89Click(TObject *Sender);
void __fastcall Button104Click(TObject *Sender);
void __fastcall Button103Click(TObject *Sender);
void __fastcall Button102Click(TObject *Sender);
void __fastcall Button101Click(TObject *Sender);
void __fastcall Button100Click(TObject *Sender);
void __fastcall Button99Click(TObject *Sender);
void __fastcall Button98Click(TObject *Sender);
void __fastcall Button97Click(TObject *Sender);
void __fastcall Button120Click(TObject *Sender);
```

```
        void __fastcall Button119Click(TObject *Sender);
        void __fastcall Button118Click(TObject *Sender);
        void __fastcall Button117Click(TObject *Sender);
        void __fastcall Button116Click(TObject *Sender);
        void __fastcall Button115Click(TObject *Sender);
        void __fastcall Button113Click(TObject *Sender);
        void __fastcall Button114Click(TObject *Sender);
        void __fastcall Button112Click(TObject *Sender);
        void __fastcall Button111Click(TObject *Sender);
        void __fastcall Button110Click(TObject *Sender);
        void __fastcall Button109Click(TObject *Sender);
        void __fastcall Button108Click(TObject *Sender);
        void __fastcall Button107Click(TObject *Sender);
        void __fastcall Button106Click(TObject *Sender);
        void __fastcall Button105Click(TObject *Sender);
private:    // User declarations
public:           // User declarations
        __fastcall TForm3(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm3 *Form3;
//---------------------------------------------------------------------------
#endif
```

Файл pro_pr.cpp – вікно «Про програму...»

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

#include <vcl.h>
#pragma hdrstop

#include "pro_pr.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm2::Button1Click(TObject *Sender)
{
Form2->Close();
}
//---------------------------------------------------------------------------
```

Файл pro_pr.h – бібліотека для файлу pro_pr.cpp

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

#ifndef pro_prH
#define pro_prH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm2 : public TForm
{
__published:    // IDE-managed Components
        TButton *Button1;
        TImage *Image1;
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        void __fastcall Button1Click(TObject *Sender);
private:        // User declarations
public:         // User declarations
        __fastcall TForm2(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm2 *Form2;
//---------------------------------------------------------------------------
#endif
```