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 програмування контролера USB у навчальних цілях”

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

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Петренюк В.І.
«____» _____ 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

Будунову Владиславу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи візуалізації програмування контролера USB у навчальних цілях*

2. Керівник роботи *Петренюк Володимир Ілліч, канд. фіз.-мат. 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 програмування контролера USB у навчальних цілях*

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
_____ Петренюк В.І.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Будунов В.О.
(прізвище та ініціали)

АНОТАЦІЯ

Будунов В.О. Програмне забезпечення системи візуалізації програмування контролера USB у навчальних цілях. 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 програмування контролера USB у навчальних цілях.

Метою розробки є програмне забезпечення системи візуалізації програмування контролера USB у навчальних цілях.

Результат роботи – програмна реалізація системи візуалізації програмування контролера USB у навчальних цілях.

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, візуалізації програмування контролера USB

ABSTRACT

Budunov V.O. USB controller programming visualization system software for educational purposes. 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 USB controller programming visualization system for educational purposes.

The purpose of the development is the software of the USB controller programming visualization system for educational purposes.

The result of the work is the software implementation of the USB controller programming visualization system for educational purposes.

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, USB controller programming visualizations

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ІТ	–	інформаційні технології
ОТ	–	обчислювальна техніка
ПЗ	–	програмне забезпечення
ПІТ	–	перспективні інформаційні технології

ВСТУП

Актуальність теми. Підготовка висококваліфікованого фахівця з системного програмування неможливо без наявності з однієї сторони сучасної обчислювальної техніки, а з іншої сторони навчаючих програмних продуктів та емуляторів й візуалізаторів роботи елементів ЕОМ та операційних систем.

Ряд авторів [2,6] передають думку про те, що тверді методичні установки в сфері комп'ютерного навчання з таких питань, як індивідуалізація й контроль за засвоєнням знань, не годяться для розробки універсальних дидактичних правил. Вибір типу навчального середовища, найбільш відповідний завданню досягнення студентом, який вивчає системне програмування, навчальних цілей, у значній мірі залежить від індивідуальних здатностей того, кого навчають, і характеру самих цілей навчання. Існуючі різні по характеру програми для комп'ютерів прямо або побічно відбивають деякі теоретичні посилки про сутність процесу навчання. Немає необхідності при відборі навчальних матеріалів виходити з оцінки валідности або яких-небудь інших аспектів цих теорій. Набагато вірніше вирішити питання із практичної точки зору, яка парадигма найбільше підходить для даних конкретних умов. Варто також визнати, що індивідуальні розходження в способах засвоєння матеріалу можуть стати вирішальним фактором, деякі студенти більше сприйнятливі до певного методу навчання. Таким чином, стає очевидним – ефективне застосування комп'ютерів у навчальному процесі при підготовці системного програміста, цілком залежить від якості й концептуальної основи програм, які закладаються в ЕОМ.

Одним з основних елементів архітектури сучасних ЕОМ є універсальна шина передачі даних (USB). Саме через цю шину сполучається основна частина пристроїв, які підключаються до сучасних ЕОМ.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи візуалізації програмування контролера USB у навчальних цілях.

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

– Огляд існуючих систем візуалізації програмування контролера USB у навчальних цілях.

– Дослідження системи візуалізації програмування контролера USB у навчальних цілях.

– Програмна реалізація системи візуалізації програмування контролера USB у навчальних цілях.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі візуалізації програмування контролера USB у навчальних цілях.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи візуалізації програмування контролера USB у навчальних цілях, 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

Система призначена для реалізації програмного забезпечення візуалізації програмування контролера USB у навчальних цілях.

USB (Universal Serial Bus – «універсальна послідовна шина») – послідовний інтерфейс передачі даних для середньошвидкісних і низькошвидкісних периферійних пристроїв в обчислювальній техніці. Символом USB є чотири геометричні фігури: велике коло, мале коло, трикутник, квадрат.

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

Для підключення периферійних пристроїв до шини USB використовується чотирьохдротовий кабель, при цьому два дроти (кручена пари) у диференціальному включенні використовуються для прийому й передачі даних, а два дроти – для живлення периферійного пристрою. Завдяки убудованим лініям живлення USB дозволяє підключати периферійні пристрої без власного джерела живлення (максимальна сила струму, споживаного пристроєм по лініях живлення шини USB, не повинна перевищувати 500 мА).

До одного контролера шини USB можна приєднати до 127 пристроїв по топології «зірка», у тому числі й концентратори. На одній шині USB може бути до 127 пристроїв і до 5 рівнів каскадування хабів, не вважаючи кореневого.

На сучасний час широко використовуються пристрої, виконані у відповідності зі специфікацією USB 2.0. Ведеться впровадження у виробництво пристроїв специфікації USB 3.0.

Специфікації для USB 1.0 були представлені в листопаді 1995 року. Розробка USB підтримувалася Intel, Microsoft, Philips і US Robotics. USB став

«загальним знаменником» під трьома не зв'язаними один з одним прагненнями різних компаній:

– Розширення функціональності комп'ютера. На той момент для підключення зовнішніх периферійних пристроїв до персонального комп'ютера використовувалося декілька «традиційних» інтерфейсів (PS/2, послідовний порт, паралельний порт, порт для підключення джойстика, SCSI), і з появою нових зовнішніх пристроїв розробляли й новий роз'єм. Передбачалося, що USB замінить їх всі й заодно підхльосне розробку нетрадиційних пристроїв.

– Підключити до комп'ютера мобільний телефон. У той час піднімалися на ноги комп'ютерні мережі, телефони переходили на цифрову передачу голосу, і жоден з наявних інтерфейсів не годився для передачі з телефону на комп'ютер як мови, так і даних.

– Простота для користувача. Старі інтерфейси (наприклад, COM– і LPT-порти) були вкрай прості для розроблювача, але не давали реального «plug and play». Були потрібні нові механізми взаємодії комп'ютера з низько– і середньошвидкісними зовнішніми пристроями – можливо, більш складні для конструкторів, але надійні, дружньому й придатні до «гарячого» підключенню.

Підтримка USB вийшла у вигляді патча до Windows 95b, надалі вона ввійшла в стандартну поставку Windows 98. Пристроїв було мало, і шину називали «useless serial bus» – «марна послідовна шина». Втім, виробники швидко усвідомили користь USB, і вже до 2000 року більшість принтерів і сканерів працювали з новим інтерфейсом.

Hewlett-Packard, Intel, Lucent (нині Alcatel-Lucent), Microsoft, NEC, і Philips спільно виступили з ініціативою по розробці більше швидкісної версії USB. Специфікація USB 2.0 була опублікована у квітні 2000 року, і наприкінці 2001 року ця версія була стандартизована USB Implementers Forum. USB 2.0 є назад сумісною з усіма попередніми версіями USB.

У середині 2000-х років BIOS'и комп'ютерів почали масово підтримувати USB. Це дозволило завантажуватися із флеш-дисків; пропала потреба в PS/2-

клавіатурі, наприклад, для переустановки ОС. Все частіше з'являються материнські плати, що взагалі не мають LPT– і COM-портів.

Основні відомості

Кабель USB складається з 4 мідних дротів – 2 дроти живлення й 2 дроти даних у кручений парі, і заземленою опліткою/екрана.

Кабелі USB орієнтовані, тобто мають фізично різні наконечники «до пристрою» і «до хосту». Можлива реалізація USB пристрою без кабелю, з убудованим у корпус наконечником «до хосту». Можливо й нероз'ємне вбудовування кабелю в пристрій, як у мишу (стандарт забороняє це для пристроїв full і high speed, але виробники його порушують). Існують (хоча й заборонені стандартом) і пасивні USB подовжувачі, що мають роз'єм «від хосту» і «до хосту».

Шина строго орієнтована, має поняття «головний пристрій» (хост, він же USB контролер, звичайно убудований у мікросхему південного мосту на материнській платі) і «периферійні пристрої». Шина має деревоподібну топологію, оскільки периферійним пристроєм може бути розвітвитель (hub), у свою чергу маючий кілька спадних роз'ємів «від хосту». Розвітвитель є складний електронний пристрій, пасивних розвітвителей не буває.

З'єднання 2 комп'ютерів – або 2 периферійних пристроїв – пасивним USB кабелем неможливо. Існують активні USB кабелі для з'єднання 2 комп'ютерів, але вони містять у собі складну електроніку, емулюючу Ethernet адаптер, і вимагають установки драйверів по обидва боки.

Пристрої можуть бути заживленні від шини, але можуть і вимагати зовнішнє джерело живлення. Підтримується й черговий режим для пристроїв і розвітвителей по команді із шини зі зняттям основного живлення при збереженні чергового живлення й включенням по команді із шини.

USB підтримує «гаряче» підключення й відключення пристроїв.

На логічному рівні пристрій USB підтримує транзакції прийому й передачі даних. Кожний пакет кожної транзакції містить у собі номер кінцевої точки

(endpoint) на пристрої. При підключенні пристрою драйвери в ядрі ОС читають із пристрою список кінцевих точок і створюють керуючі структури даних для спілкування з кожною кінцевою точкою пристрою. Сукупність кінцевої точки й структур даних у ядрі ОС називається каналом (pipe).

Кінцеві точки, а виходить, і канали, відносяться до одного з 4 класів – потоковий (bulk), керуючий (control), ізохронний (isoch) і переривання (interrupt). Низькошвидкісні пристрої, такі, як миша, не можуть мати ізохронні й потокові канали.

Керуючий канал призначений для обміну із пристроєм короткими пакетами «питання-відповідь». Будь-який пристрій має керуючий канал 0, що дозволяє програмному забезпеченню ОС прочитати коротку інформацію про пристрій, у тому числі коди виробника й моделі, використовувані для вибору драйвера, і список інших кінцевих точок.

Канал переривання дозволяє доставляти короткі пакети й у тім, і в іншому напрямку, без одержання на них відповіді/підтвердження, але з гарантією часу доставки – пакет буде доставлений не пізніше, ніж через N мілісекунд. Наприклад, використовується в пристроях уведення людиною (клавіатури/миші/джойстики).

Ізохронний канал дозволяє доставляти пакети без гарантії доставки й без відповідей/підтверджень, але з гарантованою швидкістю доставки в N пакетів на один період шини (1 Кгц в low і full speed, 8 Кгц в high speed). Використовується для передачі аудіо– і відеоінформації.

Потоковий канал дає гарантію доставки кожного пакета, підтримує автоматичне припинення передачі даних по небажанню пристрою (переповнення або спустошення буфера), але не дає гарантій швидкості й затримки доставки. Використовується, наприклад, у принтерах і сканерах.

Час шини ділиться на періоди, на початку періоду контролер передає всій шині пакет «початок періоду». Далі протягом періоду передаються пакети

переривань, потім ізохронні в необхідній кількості, у час, що залишився, у періоді передаються керуючі пакети й в останню чергу потокові.

Активною стороною шини завжди є контролер, передача пакета даних від пристрою до контролера реалізоване як коротке питання контролера й довгий, утримуючий дані, відповідь пристрою. Розклад руху пакетів для кожного періоду шини створюється спільним зусиллям апаратури контролера й ПО драйвера, для цього багато контролерів використовують украй складний DMA зі складної DMA-програмою, формованої драйвером.

Розмір пакета для кінцевої точки є вшита в таблицю кінцевих точок пристрою константа, зміні не підлягає. Він вибирається розроблювачем пристрою із числа тих, що підтримуються стандартом USB

Версії специфікації

– **USB 0.7**: специфікація випущена в листопаді 1994 року.

– **USB 0.8**: специфікація випущена в грудні 1994 року.

– **USB 0.9**: специфікація випущена у квітні 1995 року.

– **USB 0.99**: специфікація випущена в серпні 1995 року.

– **USB 1.0 Release Candidate**: специфікація випущена в листопаді 1995 року.

USB 1.0

Специфікація випущена в листопаді 1995 року.

Технічні характеристики:

– два режими передачі даних:

– режим з високою пропускною здатністю (Full-Speed) – 12 Мбіт/с;

– режим з низькою пропускною здатністю (Low-Speed) – 1,5 Мбіт/с;

– максимальна довжина кабелю для режиму з високою пропускною здатністю – 5 м;

– максимальна довжина кабелю для режиму з низькою пропускною здатністю – 3 м;

– максимальна кількість підключених пристроїв (включаючи розмножувачі) – 127;

– можливе підключення пристроїв, що працюють у режимах з різною пропускною здатністю до одного контролера USB;

– напруга живлення для периферійних пристроїв – 5В;

– максимальний струм, споживаний периферійним пристроєм – 500 мА;

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

Рисунок 1.1 – Логотип USB 2.0 High Speed

Специфікація випущена у квітні 2000 року.

USB 2.0 відрізняється від USB 1.1 введенням режиму Hi-speed.

Для пристроїв USB 2.0 регламентовано три режими роботи:

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

– Full-speed, 0,5-12 Мбіт/с (аудіо-, відеопристрої).

– Hi-speed, 25-480 Мбіт/с (відеопристрої, пристрої зберігання інформації).

Наступні модифікації

Наступні модифікації до специфікації USB публікуються в рамках Повідомлень про інженерні зміни (Engineering Change Notices – ECN). Найважливіші з модифікацій ECN представлені в наборі специфікацій USB 2.0 USB 2.0 specification package, доступному на сайті USB Implementers Forum.

– **Mini-B Connector ECN**: повідомлення випущене в жовтні 2000 року.

– **Errata, починаючи із грудня 2000**: повідомлення випущене в грудні 2000 року.

– **Pull-up/Pull-down Resistors ECN**: повідомлення випущене в травні 2002 року.

– **Errata, починаючи із травня 2002**: повідомлення випущене в травні 2002 року.

– **Interface Associations ECN**: повідомлення випущене в травні 2003 року. Були додані нові стандарти, що дозволяють асоціювати безліч інтерфейсів з однією функцією пристрою.

– **Rounded Chamfer ECN**: повідомлення випущене в жовтні 2003 року.

– **Unicode ECN**: повідомлення випущене в лютому 2005 року. Дане ECN специфікує, що рядки закодовані з використанням UTF-16LE.

– **Inter-Chip USB Supplement**: повідомлення випущене в березні 2006 року.

– **On-The-Go Supplement 1.3**: повідомлення випущене в грудні 2006 року. USB On-The-Go уможливлює зв'язок двох USB-пристроїв один з одним без окремого USB-Хосту. На практиці один із пристроїв відіграє роль хосту для іншого.

USB OTG

Рисунок 1.2 – Логотип USB OTG

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. Наприклад, цифровий фотоапарат можна

підключати до фотопринтера прямо, якщо вони обоє підтримують стандарт USB OTG. До моделей КПК і коммуникаторов, 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 зрослої останнім часом необхідності надійного з'єднання різних пристроїв без використання ПК.

Хоча з'єднання USB OTG виглядає як однорангове, насправді тільки створюється таке відчуття – у дійсності пристрої самі визначають, який з них буде майстер-пристроєм, а який – підлеглим. Одноранговий інтерфейс USB існувати не може.

USB Wireless

Рисунок 1.3 – Логотип USB wireless

USB wireless – технологія USB (офіційна специфікація доступна із травня 2005 року). Дозволяє організувати бездротовий зв'язок з високою швидкістю передачі інформації (до 480 Мбіт/с на відстані 3 метри й до 110 Мбіт/с на відстані 10 метрів).

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

USB 3.0

Стандарт USB версія 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, плюс і нуль живлення. 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 4,8 Гбіт/с – що на порядок більше 480 Мбіт/с, які може забезпечити USB 2.0.

Версія 3.0 може похвастатися не тільки більше високою швидкістю передачі інформації, але й збільшеною силою струму з 500 мА до 900 мА. Відтепер користувач зможе не тільки підживляти від одного хаба набагато більшу кількість пристроїв, але й саме апаратне забезпечення позбудеться від окремих блоків, що раніше поставлялися, живлення.

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

Компанія Asus анонсувала материнську плату P6X58 Premium у якої є два USB 3.0 порти.

У блоці розроблювача Linux USB subsystem Sarah Sharp оголошено про підтримку USB 3.0 ядром Linux.

Фірмою Intel анонсована попередня версія програмної моделі контролера USB 3.0.

Але в жовтні 2009 року з'явилася інформація (від EE Times з посиланням на співробітника однієї з найбільших компаній по виробництву персональних комп'ютерів), що корпорація Intel вирішила почекати із впровадженням підтримки USB 3.0 у свої чипсети до 2011 р. Це рішення приведе до того, що даний стандарт не стане масовим ще як мінімум півтора року.

Кабелі й роз’єм USB 1.0 і 2.0

Рисунок 1.4 – USB Тип B

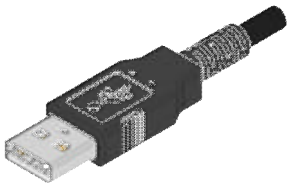

Рисунок 1.5 – USB Тип A

Специфікація 1.0 регламентувала два типи роз’ємів: A – на стороні контролера або концентратора USB і B – на стороні периферійного пристрою. Згодом були розроблені мініатюрні роз’єми для застосування USB у переносних і мобільних пристроях, що одержали назву Mini-USB. Нова версія мініатюрних роз’ємів, називаних Micro-USB, була представлена USB Implementers Forum 4 січня 2007 року.

Розміри роз’ємів: USB Тип A – 4x12 мм, USB Тип B – 7x8 мм, USB mini A і USB mini B – 2x7 мм.

Рисунок 1.6 – Micro USB Тип B

Рисунок 1.7 – Mini USB Тип A (ліворуч) і Mini USB Тип B (праворуч)

Існують також роз’єм типу Mini-AB і Micro-AB, з якими з'єднуються відповідні коннектори як типу A, так і типу B.

На відміну від інших стандартних типів роз’ємів, USB-A вдало сполучить довговічність і механічну міцність, незважаючи на відсутність гвинтового затягування. Однак зменшені варіанти роз’ємів, що мають тонкі пластмасові виступи, що високо виступають із підложки гнізда, погано переносять часте змикання-розмикання й вимагають більше дбайливого поводження.

Сигнали USB передаються по двох проводам екранованого чотирьохдротового кабелю.

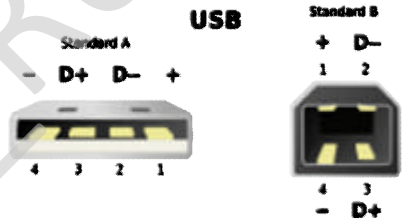

Рисунок 1.8 – Розпаювання USB

Таблиця 1.1 – Розміщення дротів

Номер контакту	Позначення	Колір проведення
1	V BUS	червоний
2	D−	білий
3	D+	зелений
4	GND	чорний

Тут GND – ланцюг «корпуса» для живлення периферійних пристроїв, VBus – +5В, так само для ланцюгів живлення. Дані передаються по проводам D+ і D− диференціально (стану 0 і 1 (у термінології офіційної документації diff0 і diff1 відповідно) визначаються по різниці потенціалів межу лініями більше 0,2 В и за умови, що на одній з ліній (D− у випадку diff0 і D+ при diff1) потенціал відносно GND вище 2,8 В.

Диференціальний спосіб передачі є основним, але не єдиним (наприклад, при ініціалізації пристрій повідомляє хосту про режим, підтримуваному пристроєм (Full-Speed або Low-Speed), підтягуванням однієї з ліній даних до V_BUS через резистор 1,5 кому (D− для режиму Low-Speed і D+ для режимів Full-Speed і High-Speed.).

Рисунок 1.9 – Коннектор USB 3.0 тип B

Рисунок 1.10 – Коннектор USB 3.0 тип A

Недоліки USB

Хоча пікова пропускна здатність USB 2.0 становить 480 Мбіт/с (60 Мбайт/с), на практиці забезпечити пропускну здатність, близьку до пікової, не

вдається (~33,5 Мбайт/сек на практиці). Це пояснюється досить великими затримками шини USB між запитом на передачу даних і властиво початком передачі. Наприклад, шина FireWire хоча й має меншу пікову пропускну здатність 400 Мбіт/с, що на 80 Мбіт/с менше, ніж в USB 2.0, у реальності дозволяє забезпечити більшу пропускну здатність для обміну даними з жорсткими дисками й іншими пристроями зберігання інформації. У зв'язку із цим різноманітні мобільні накопичувачі вже давно «упираються» у недостатню практичну пропускну здатність USB 2.0

1.2 Область застосування

Областю застосування розроблювального, у результаті виконання бакалаврського проектування, системного програмного забезпечення є системи навчання та підготовка фахівців у галузі інформаційних технологій та програмування, з використання комп’ютерної техніки.

Прогрес у науці й техніку змушує переглядати викладачів методику підготовки фахівців, здатних до активної професійної діяльності в різних областях суспільного й державного життя, у сфері науки й культури.

Впровадження електронної обчислювальної техніки в усі сфери діяльності людини дозволило дійти висновку про те, що подальший розвиток системи утворення тільки традиційними шляхами неможливо, необхідні нові підходи використання електронної обчислювальної техніки, що дозволить розвити й підсилити потенційні можливості особистості.

Використання в навчальному процесі електронної обчислювальної машини (ЕОМ) викликано "...внутрішніми потребами самої системи утворення, визначається логікою розвитку педагогічної науки, необхідністю істотного підвищення якості навчально-виховного процесу, оптимізацією керування в системі утворення, удосконалюванням науково-педагогічних досліджень, посиленням їхнього впливу на педагогічну практику" [4].

Вивчення всіх дисциплін у школах, СПТУ, технікумах і вузах із застосуванням персональних комп'ютерів (ПК) у цей час є гострою проблемою, тому що прямо пов'язане з рівнем підготовки викладача, умінням його особисто застосовувати ЕОМ на заняттях.

Коло проблем, пов'язаних з пошуком шляхів і методів інформатизації утворення, відповідно до питань ефективного використання комп'ютерної техніки в сфері освіти й педагогічної науки розглянутий у дослідженнях, присвячених теоретико-методологічним положенням комп'ютеризації навчання.

Не вивчені також і можливості використання структурованого навчального матеріалу, наприклад, у навчальних програмах для комп'ютера, націлених на самостійність і індивідуалізацію дидактично-методичної підготовки студентів. Під терміном "дидактичні умови" у педагогічній науці багато які дидакти розуміють як фактори, які забезпечують успішне навчання [7]. Дидактичні умови навчання – обстановка, при якій компоненти навчального процесу (навчальний предмет, викладання й навчання) представлені в найкращому взаємовідношені і яка дає можливість викладачеві плідно викладати, керувати навчальним процесом, а студентам – успішно вчитися.

Впровадження електронно-обчислювальної техніки в навчання дозволить більш успішно вирішити завдання розвитку особистості, а також оцінювати розвиваючий ефект і ефект, що виховує, відповідній методиці навчання. Комп'ютер відрізняється точністю, об'єктивністю, неупередженістю, терпінням, готовністю до спілкування. Тільки використовуючи ЕОМ у навчанні, викладач може зайнятися своєю прямою справою – прийняттям конкретних рішень по інтенсифікації процесу навчання індивідуальних для кожного студента.

Впровадження ЕОМ у навчання дозволити також автоматизувати рутинну, непродуктивну, стомлюючу працю викладача. Реалізація програми інформатизації утворення припускає якісне перетворення методів навчальної роботи, а також зміна змісту навчання відповідно до нових цілей. У цей час, практично на всіх заняттях, у процесі вивчення того або іншого навчального предмета застосовуються комп'ютерні програми. У той же час поки немає досліджень по виявленню дидактичних основ застосування персональних

комп'ютерів у процесі підготовки майбутнього системного програміста. Отже, залишається нереалізованим цілий ряд потенційних можливостей використання ЕОМ для підвищення ефективності в підготовці майбутніх системних програмістів.

Реально виниклі протиріччя різко зрослим в останні роки рівнем розвитку апаратних засобів обчислювальної техніки нового покоління й настійною потребою педагогічної практики у відповідних розробках програмно-педагогічних засобів, які дозволили б моделювати роботу елементів ЕОМ, і сприяли б тим самим підвищенню ефективності підготовки майбутніх системних програмістів; вимогам суспільства до підвищення якості підготовки майбутніх системних програмістів і скороченням часу на цю підготовку; між переважним застосуванням репродуктивних форм у навчанні системного програміста й творчим характером його діяльності вимагають додаткових педагогічних рішень.

Крім того, проблема комп'ютерного навчання студентів завжди перебувала в центрі уваги педагогів. Однак інтерес до неї зростає, оскільки зміна соціальних умов життя привело до перегляду поглядів різних вчених на цілі й зміст комп'ютера як засіб розвитку особистості. Для успішного впровадження комп'ютерного навчання в підготовку майбутніх системних програмістів важливо не тільки (і не стільки) забезпечення навчальних закладів сучасною комп'ютерною технікою, але й вирішити ряд більше важливих проблем, таких як розробка нових навчальних програм і планів, що припускають використання нових інформаційних технологій; створення високоякісних, що відповідають сучасним вимогам педагогічних програмних засобів; вироблення науково-обґрунтованих технологій, їхнє використання в навчальному процесі.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи візуалізації програмування контролера USB у навчальних цілях, 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

У цьому розділі розглянемо існуючі емулятори що використовують USB.

JTAG емулятор SAU 510-USB ISO PLUS

JTAG емулятор SAU 510-USB ISO PLUS призначений для використання разом із цифровими сигнальними процесорами (DSP) і мікропроцесорами, які підключаються через інтерфейс JTAG з рівнями від +1.65V до +5V. Емулятор підключається до ПК за допомогою USB інтерфейсу. Це означає, що він не вимагає живлення від відладникової плати.

Емулятор SAU 510-USB ISO PLUS сполучимо з існуючими відладниковими засобами, що поставляються Texas Instruments.

JTAG емулятор SAU 510-USB ISO PLUS має наступні характеристики:

– Підтримує цифрові сигнальні процесори (C2000, C5000, C6000, DaVinciTM, OMAPTM) і 16/ 32-бітні RISC мікроконтролери TMS470R1x компанії Texas Instrument з інтерфейсом JTAG (IEEE 1149.1).

– Сполучимий з емулятором XDS510 компанії Texas Instrument.

– Забезпечує гальванічну ізоляцію 2500VRMS між ПК і відладнуємим пристроєм.

– Має поліпшений JTAG-контролер, що забезпечує високу ефективність емуляції.

– Сполучимий з USB 1.1 і USB 2.0 (high speed/full speed).

– Підключається до ПК через USB-Інтерфейс і не вимагає додаткового джерела живлення.

– Підтримує JTAG інтерфейси з рівнями сигналу від +1.65V до +5V.

– Має три індикатори (LED) для відображення стану й режиму роботи.

– Електроживлення забезпечується від USB порту або USB концентратора.

– Підтримує програмування й конфігурування FPGA і CPLD за допомогою SVF-плеєра (SVF Specification Rev.E + розширення Lattice Semiconductor).

– Сполучимий з Code Composer Studio IDE компанії Texas Instruments.

– Сполучимий з операційними системами Windows 2000, Windows XP, Windows Vista (32-bit).

Основні елементи:

– Індикатори режимів роботи.

– JTAG роз’єм.

– Роз’єм mini-USB для підключення до ПК.

У наведених нижче списках перераховане необхідне для роботи JTAG емулятора SAU 510-USB ISO PLUS устаткування й програмне забезпечення.

Необхідне встаткування.

– ПК: будь-який ПК або ноутбук, що має жорсткий диск, USB-порт і CD-ROM привод.

– Пам'ять: мінімум 32MB.

– Монітор: кольоровий VGA або LCD.

– Емулятор: JTAG емулятор SAU 510-USB ISO PLUS.

– Відладна плата: будь-яка відладна плата із джерелом живлення, побудоване на базі DSP або мікроконтролера компанії TI.

– Роз’єм для підключення: 14-контактне роз’єм (два ряди по семи штирьков), 20-контактний ARM роз’єм або 20-контактний CTI роз’єм.

Необхідне програмне забезпечення.

– Операційна система: Windows 2000, Windows XP илиWindows Vista (32-bit).

– Програмні засоби налагодження: Code Composer Studio.

– Драйвери: драйвери від компанії Sauris Gmb для відладникового середовища Code Composer Studio компанії TI (поставляються в комплекті з емулятором SAU 510-USB ISO PLUS, а також доступні на сайті компанії Sauris Gmb).

Засіб розробки для мікроконтролерів AVR фірми Atmel

AVR-JTAG-USB – оптоізольований USB-JTAG емулятор для мікроконтролерів фірми Atmel сімейства AVR.

Таблиця 2.1 – Технічні параметри позиції AVR-JTAG-USB програматор-емулятор

Тип компонента	емулятор
Ядро базового компонента	AVR
Розрядність	8
Найменування базового компонента	AVR
Найменування допоміжного компонента	-
RS-232	-
LPT	-
USB	1.00
Ethernet	-
PCI	-
CAN	-
JTAG\BDM	1.00
ISP\ICP\BSL	-
LCD	-
PA	-
Other	-

Особливості емулятора:

– Програмування всіх AVR мікроконтролерів з підтримкою JTAG інтерфейсу.

– Цільова напруга 3,0 – 5,0В.

– Живлення від USB інтерфейсу.

– JTAG коннектор сполучимо з Atmel 2x5 пін коннектором JTAG (відрізняється від ICSP коннектора).

– Сполучимо з Atmel AVR STUDIO для програмування, емуляції в режимі реального часу, налагодження, виконання програми в покроковому режимі, установки точок останова, дампа пам'яті й т.д.

– Повна емуляція всіх аналогових і цифрових функцій.

– Повна підтримка програмування через JTAG порт.

– Відновлення через AVR STUDIO.

– Роз’єм USB інтерфейсу – типу «А».

Комплектація: програматор/емулятор AVR-JTAG-USB.

Для роботи може знадобитися USB кабель « A-A» – SCUAA-1.

QEMU

QEMU – емулятор процесора, що використовує динамічну трансляцію для досягнення гарної швидкості емуляції.

QEMU має два режими роботи:

– Емуляція всієї системи. У цьому режимі QEMU емулює повноцінну систему (наприклад ПК), включаючи процесор і різне периферійне встаткування. Його можна використовувати для запуску різних операційних систем без необхідності перезавантаження ПК або для налагодження коду системи.

– Емуляція користувальницького режиму (тільки Linux-машини). У цьому режимі QEMU може запускати процеси Linux, скомпільовані для одного процесора, на іншому процесорі. Його можна використовувати для запуску Wine – емулятора API Windows, або для спрощеної крос-компіляції й кросу-налагодження.

QEMU може працювати на системі-хазяїні без драйвера, забезпечуючи при цьому задовільну продуктивність.

При емуляції цілої системи підтримується наступне устаткування:

– PC (процесор x86 або x86_64).

– PREP (процесор PowerPC).

– G3 BW PowerMac (процесор PowerPC).

– Mac99 PowerMac (процесор PowerPC, у процесі розробки.

– Sun4m (32-бітний процесор Sparc).

– Sun4u (64-бітний процесор Sparc processor, у процесі розробки).

– Malta board (32-бітний процесор MIPS, у процесі розробки).

При емуляції користувача підтримуються процесори x86, PowerPC, ARM і Sparc32/64.

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 програмування контролера USB у навчальних цілях.

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

Програмування контролера USB

Для успішної розробки програмної частини USB-пристрою необхідно добре знати фізичну реалізацію шини. У цьому розділі представлені відомості про загальну фізичну архітектуру шини, логічну архітектуру USB-пристроїв, типи і формати передач даних по шині USB. Частину роботи з формування службового трафіку й форматуванню корисних даних беруть на себе контролери USB. Однак, знання й розуміння цих подробиць дозволить найбільше ясно представляти схему взаємодії пристроїв на шині й найбільш ефективно реалізувати свій пристрій.

Загальна організація шини USB

Шина USB має деревоподібну ієрархічну топологію. У вершині дерева перебуває хост-контролер, що здійснює централізоване монопольне керування всією шиною. Хост визначає підключення пристроїв, здійснює їхнє конфігурування й призначення їм адрес, розподіляє живлення від шини й смугу пропущення між пристроями, веде збір статистики. У виді центрального місця хосту на шині напрямок передачі даних прийнято визначати щодо його положення: якщо передача йде від хосту до пристрою, то потік даних має напрямок OUT і називається спадним; при передачі від пристрою до хосту напрямок потоку IN, а називається він висхідним.

Всі інші пристрої на шині крім хосту є веденими. Вони діляться на два типи: комунікаційні пристрої – хаби й кінцеві пристрої, що виконують деяку прикладну функцію. Хаби використовуються для подовження й розгалуження шини. Хост періодично опитує хаб і по зміні його стану визначає підключення нових пристроїв до шини або відключення раніше підключених.

Ще раз відмітимо, активним на шині є тільки хост. Пристрої не вправі передавати дані по шині по своєму баченню можуть почати передачу тільки на вимогу хосту. Єдиним виключенням із цього правила є можливість припиненого пристрою (переведеного в режим зниженого енергоспоживання, «засипання») повідомити хосту про своє пробудження від зовнішнього впливу. У такому випадку пристрій має право повідомити хост про зміну свого стану не чекаючи запиту. Можливість пробудження вказується в конфігурації пристрою. Однак при конфігуруванні ця можливість може бути заборонена хостом.

Шина USB підтримує три швидкісних режими передачі даних:

– низькошвидкістний (Low Speed) зі швидкістю передачі до 1,5 Мбіт/с;

– повношвидкістний (Full Speed) зі швидкістю передачі до 12 Мбіт/с;

– високошвидкісний (High Speed) зі скоорстью передачі до 480 Мбіт/с. Останній режим доступний тільки на шині USB 2.0.

Швидкісний режим, установлений для пристрою визначається не тільки можливостями пристрою, але й можливостями конкретної шини. Високошвидкісний пристрій, підключений до повільного хосту/хабу, не зможе повністю використовувати свій швидкісний потенціал, а буде обмежено можливостями провідного пристрою.

Живлення пристрою можуть одержувати від власного джерела або від внутрішнього джерела шини. Знову підключений до шини пристрій не повинне споживати від її струм більше 100 мА. Під час конфігурування пристрій указує свої потреби струму до 500 мА. Якщо хаб не зможе забезпечити пристрою необхідний струм, то він не буде використовуватися. У режимі припинення пристрій не повинне споживати від шини струм більше 500 мкА.

Відповідно до специфікації, до одного роз'єму можна підключити до 127 пристроїв одночасно. Однак, на практиці це представляється якщо не сумнівним, те, принаймні, досить напруженим. Не варто забувати про два моменти:

– По-перше, про обмежену потужність шини. Ця проблема стосується пристроїв з живленням від шини й вирішується підключенням їх через хаби з

автономним живленням.

– По-друге, про те, що її пропускна здатність кінцева. У шини стандарту USB 2.0 вона досить велика, але все-таки кінцева й навряд чи її вистачить для нормальної роботи 127 в окремому випадку високошвидкісних пристроїв.

Логічна структура USB-пристроїв

Стандарт USB передбачає логічну розбивку внутрішньої структури пристроїв по декількох рівнях ієрархії. Так само як і топологію шини USB логічну організацію USB-пристроїв схематично зручно представити у вигляді дерева. Вузли гілок позначають режими роботи, листи дерева певні функції. Настроювання пристрою на деякий конкретний режим роботи виконуються вказівкою вузлів відповідної гілки. Установку й зміну режимів здійснює хост. Режим залишається активним до наступної команди зміни режиму. Функції, доступні в активному режимі, називаються кінцевими точками. Виконання функції здійснюється при обігу хосту до відповідної кінцевої точки. У логічній структурі пристрою можливі режими, у яких відсутні які-небудь функції (гілка cfg 1-if0-alt0). Такі режими звичайно використовуються для перекладу пристрою в якийсь бездіяльний стан типу простою.

Розглянемо докладніше елементи логічної структури USB-пристроїв.

Конфігурації (cfg) є верхнім рівнем узагальнення й визначають найбільш загальні настроювання. Пристрій може мати до 255 конфігурацій, але хоча б 1 воно мати повинне. Нумерація конфігурацій починається з 1. Номер 0 використовується для позначення того, що пристрій не зконфігуровано.

На наступному рівні ієрархії перебувають інтерфейси (if). Вони дозволяють розділити загальний режим роботи, обумовлений конфігурацією, по різних специфічних особливостях. Хоча б один інтерфейс повинен бути присутнім у будь-якій конфігурації, а максимально їх може бути 256. Нумерація інтерфейсів починається з 1.

Після інтерфейсів ще більш конкретно уточнюють режим роботи альтернативні установки (alt), вони описують окремі випадки настроювання

режиму роботи пристрою. Альтернативні установки визначають кількість підтримуваних у даному режимі кінцевих точок-функцій і способи роботи з ними. Нумерація альтернативних установок починається з нуля, а їхня кількість у кожному інтерфейсі може бути від 1 до 256.

Функції, які пристрій здатний виконувати в поточному режимі, визначаються доступними кінцевими точками (ep). Кінцева точка це логічний канал передачі даних. При записі даних у канал пристрій виконує деяку функцію; при читанні з каналу пристрій повертає результати виконання операцій, що течуть параметри й т.д. Кожна кінцева точка має свою унікальну 8-ми розрядну адресу. Кінцеві точки, що входять до складу альтернативних установок, є односпрямованими, напрямок передачі для даної кінцевої точки визначає старший біт її адреси: 0 у старшому біті адреси мають точки напрямку OUT, 1 – точки, напрямку IN. Альтернативна установка може містити до 15 точок напрямку OUT з адресами 1..0Fh і до 15 точок напрямку IN з адресами 81h..8Fh.

Особливе значення має кінцева точка 0 (ep0). Вона називається контрольною й належить не якій-небудь альтернативній установці, а всьому пристрою в цілому. Контрольна точка є двонаправленою і доступна в будь-якому режимі роботи пристрою, через неї здійснюється ідентифікація й конфігурування пристрою.

Для приклада організації внутрішньої структури USB-пристрою розглянемо деякий уявлюваний мультиметр. Допустимо, цей мультиметр може вимірювати електричну напругу, струм, можливо інші електричні параметри; а також має функцію генератора електричних імпульсів. Безлика структура пристрою, представлена на рисунку 2, адаптована для подання мультиметра й із вказівкою конкретних величин і позначень.

Перша гілка-режим cfg1-if 0-alt0 використовується для перекладу приладу в режим простою (idle cfg-idle if-idle alt).

Конфігурація cfg2 призначена для установки режиму виміру електричної напруги (вузол U). Інтерфейс if0 позначає вимір постійної напруги (вузол "=");

інтерфейс if1 – вимір змінної напруги (вузол "~"). Мультиметр здатний вимірювати змінну напругу в декількох діапазонах частот: альтернативні установки alt0 і alt1 інтерфейсу виміру змінної напруги визначають відповідно діапазони від 25Гц до 100Гц і від 100Гц до 1Кгц. На низьких частотах мультиметр здатний вимірювати змінну напругу по 2-м вхідним каналам (листи ch1 і ch2), що визначає наявність у даній альтернативній установці 2-х кінцевих точок (ep81 і ep82) напрямку IN. На частотах від 100Гц до 1Кгц прилад здатний сприймати сигнал з одного каналу, що визначає тільки одну кінцеву точку для даної альтернативної установки.

Конфігурація cfg3 () використовується для виміру електричного струму (вузол I). Розподіл її підрежимів може бути аналогічно такому для виміру напруги.

Конфігурація cfgn призначена для установки режиму генератора. Представимо що прилад здатний генерувати пилкоподібні й прямокутні імпульси. Для вибору одного із цих варіантів використовуються інтерфейси if0 і if1 конфігурації cfgn (позначення). Допустимо, пристрій може генерувати імпульси зі значенням амплітуди, що належить деякому дискретному ряду, наприклад, 3В и 5В. Альтернативні установки дозволяють вибрати конкретну амплітуду генеруємого сигналу. Кінцеві точки ep1 напрямку OUT для кожної з альтернативної установок призначені для завдання частоти імпульсів.

Відзначимо ще раз, альтернативні установки можуть мати одночасно як точки напрямку IN, так і точки напрямку OUT довільних адрес із припустимого діапазону.

Підключення й робота пристрою

Підключення пристрою до шини виробляється через порт хаба. Хост, періодично опитуючи стан хаба, розпізнає підключення нового пристрою й дозволяє відповідний порт. У цей час пристрій вважається не адресованим і не зконфигурованим. Хост звертається до пристрою за адресою 0 через контрольну точку ep0, доступну в будь-якому режимі роботи пристрою. Першою командою

хост привласнює пристрою унікальна адреса, з яким воно працює до моменту відключення від шини. Потім, хост зчитує опис пристрою й опис всіх його конфігурацій. Пристрій зобов'язаний мати хоча б одну конфігурацію. Хост установлює першу доступну конфігурацію, не аналізую її призначення. Після цього пристрій вважається зконфигурованим і готовим до роботи. Отримана інформація дозволяє операційній системі ідентифікувати пристрій і завантажити підходящий драйвер. Подальше керування пристроєм передається драйверу.

Для вибору необхідного режиму драйвер через контрольну точку ep0 передає запити установки відповідних конфігурацій і інтерфейсів. Для зміни режимів ці запити видаються повторно.

Для активізації деякої функції пристрою в даному режимі драйвер звертається до якої-небудь кінцевої точки.

Передача даних по шині USB

На шині USB організована пакетна передача даних. Пакетна передача даних, що веде роль хосту на шині й контроль цілісності даних, закладений на рівні протоколу, у сукупності визначають загальний цикл обміну пакетом, що складається з 3-х тактів:

– запиту;

– передачі корисних даних;

– підтвердження.

Запит визначає тип передачі й адреси одержувачів: адресу пристрою на шині й адресу кінцевої точки в поточному режимі роботи. Підтвердження показує цілісність даних і готовність пристрою до продовження обміну.

Формат пакета запиту представлений на рисунку 3.1. Він складається з маркера запиту, що визначає його тип, адреси пристрою, адреси кінцевої точки, контрольної суми CRC5.

Маркер запиту {SETUP | IN | OUT | PING}	Адреса пристрою	Адреса кінцевої точки	CRC5

Рисунок 3.1 – Формат пакета запиту

Маркер запиту може мати одне з 4-х значень:

– SETUP – означає що хост починає контрольну передачу для зазначеної точки пристрою.

– IN – хост очікує дані від пристрою.

– OUT – хост починає передачу даних кінцевій точці пристрою.

– PING – хост на високошвидкісній шині перевіряє стан кінцевої точки напрямку OUT.

За пакетом запиту треба пакет корисних даних. Його формат показаний на рисунку 3.2. Пакет містить у собі маркер даних, безпосередньо дані й контрольну суму CRC16.

Маркер даних {DATA0 | DATA1 | DATA2 | MDATA}	Дані	CRC16

Рисунок 3.2 – Формат пакета даних

Маркери даних покликані забезпечувати контроль цілісності даних на рівні потоку. Застосовуються наступні значення:

– DATA0 позначає парний пакет даних.

– DATA1 – непарний пакет даних.

– Значення DATA2 і MDATA використовуються при ізохронному обміні на високошвидкісній шині.

Пакети підтвердження складаються тільки з відповідного маркера. Вони призначені для повідомлення про результати передачі даних і поточного статусу кінцевої точки. У протоколі передбачені наступні маркери підтвердження:

– ACK – дані отримані без помилок і будуть оброблені.

– NAK: для точки напрямку OUT означає що дані отримані без помилок, але в цей момент не можуть бути оброблені й потрібно їхня повторна передача. для точки напрямку IN говорить про те, що дані ще не готові, хост може повторити спробу пізніше.

– STALL – запит не підтримується.

– NYET – дані поточного пакета отримані коректно й будуть оброблені, але наступний пакет точка прийняти відразу не зможе.

Крім перерахованих у протоколі застережені ще кілька типів пакетів: SOF, PRE, ERR, SPLIT. Вони виконують службові функції. Наприклад, пакет SOF використовується для синхронізації.

На шині USB передбачено 4 режими передачі даних:

– CONTROL.

– BULK.

– INTERRUPT.

– ISOCHRONOUS.

Передача типу CONTROL використовується при звертанні до контрольної точки пристрою. Для такого способу передачі хост гарантовано виділяє 10% смуги пропущення шини. Повна транзакція для даного виду передачі складається із трьох фаз (рисунок 3.3).

Перша фаза називається фаза SETUP. Під час її хост у пакеті фіксованої довжини 8 байт передає вимогу, який необхідно виконати пристрою. Помітимо, що в цій фазі маркером підтвердження SETUP-пакета є строго маркер ACK. Якщо пристрій одержав непідтримуєму або некоректну вимогу, вона повинне підтвердити одержання маркером ACK, а в наступній фазі повернути маркер STALL.

Друга фаза – фаза даних, що є необов'язковою. Вона бере участь у транзакції в тому випадку, якщо для виконання зазначеної вимоги потрібні додаткові дані. Розглянемо докладніше структуру потоку даних на цій фазі. На рисунку 3.4 наведений приклад фази дані напрямки IN. У зазначеному прикладі в першому циклі цієї фази хост спочатку передає запит на одержання даних від пристрою, пристрій відповідає пакетом даних, далі хост підтверджує його одержання маркером ACK. У другому циклі пристрій не готовий передати другий пакет даних і повертає маркер NAK. При повторному запиті пристрій встиг сформувати дані для відправлення й успішно передає пакет хосту.

Рисунок 3.3 – Контрольна передача

Фаза даних може бути як напрямку IN, так і напрямку OUT. Приклад фази дані напрямки OUT окремо показаний на рисунку 3.4. Тут хост у плині 2-х циклів успішно передає дані пристрою. В 3-му циклі пристрій, зайнятий обробкою 2-х попередніх пакетів, маркером NAK повідомляє про те, що не може обробити в сучасний момент 3-й пакет. Хост згодом повторює передачу того ж пакета даних.

Останньою фазою контрольної транзакції є фаза статусу. У цій фазі хост очікує підтвердження виконання вимоги пристроєм. Він посилає запити протилежного напрямку, щодо запитів, які використовувалися у фазі даних, а якщо фаза даних була відсутня – запити мають напрямок IN. При передачі запиту OUT хост відправляє пакет даних нульової довжини. Під час виконання запиту пристрій відповідає маркером NAK, а по завершенні роботи повертає ACK.

OUT-запит	Пакет даних 1	ACK

OUT-запит	Пакет даних 2	ACK

OUT-запит	Пакет даних 3	NAK

OUT-запит	Пакет даних 3	ACK

Рисунок 3.4 – Фаза дані напрямки OUT

Ще раз звернемо увагу на те, що якщо пристрій одержав вимогу, що не здатний виконати, то воно повинне повернути маркер STALL у фазі даних або статусу, а одержання SETUP-пакета зобов'язано підтвердити маркером ACK.

Розмір пакета для контрольних передач на високошвидкісній шині становить 64 байта, на повношвидкістній – 64, 32, 16 або 8 байт.

Передача типу BULK використовується в тому випадку, якщо потрібна гарантована доставка пакетів, але час доставки не критичний. Для пересилання більших обсягів даних звичайно використовується саме BULK-передача. Наприклад, такий спосіб передачі звичайно використовується USB-сканерами й принтерами. Протоколом гарантується цілісність доставки даних, що забезпечується за допомогою контрольної суми CRC16 і різним маркуванням парних і непарних пакетів. При виявленні помилки прийомна сторона не підтверджує коректний прийом пакета й передавальна сторона повторює посилку.

BULK-трафік займає всю вільну смугу пропущення шини, але має найнижчий пріоритет і може припинятися на відносно більші проміжки часу залежно від завантаження шини.

Розмір пакета даних при цьому способі обміну може бути кожним, у тому числі й нульовим, але не перевищувати максимального значення. Для високошвидкісної шини ця границя становить 512 байт, для повношвидкістної – 8, 16, 32 або 64 байт.

Структура потоку даних на шині повторює таку для контрольної передачі у фазі даних. Для приклада можна ще раз звернутися до контрольної передачі: передача BULK-IN представлена в середній частині рисунка 3.3, передача BULK-OUT – на рисунку 3.4. Зупинимося на останньому ледве докладніше. Він демонструє типовий обмін на повношвидкістной шині і його головний недолік. Недолік полягає в тім, що загублений 3-й пакет необхідно повторювати. Це знижує корисну пропускну здатність шини. Очевидно, чим більше розмір непродуктивного пакета, тим відчутніше зниження. На високошвидкісній шині із цим недоліком борються за допомогою запиту PING і підтвердження NYET. У подібній ситуації на високошвидкісній шині після прийому 2-го пакети пристрій відповість маркером NYET. При одержанні такої відповіді, хост відкладе передачу 3-го пакету й буде відслідковувати готовність пристрою за допомогою короткого запиту PING. При одержанні підтвердження ACK хост зможе відправити 3-й пакет.

Слід зазначити, що й при використанні механізму опитування PING/NYET втрати й повторні передачі пакетів можливі, але в кардинально іншому випадку: через некоректний прийом даних. Причиною тому може служити, наприклад, дія зовнішньої електричної перешкоди. Зв'язування ж PING/NYET бореться саме із проблемою даремних витрат пропускної здатності шини через активність хосту, не скоректованої на можливості пристрою.

Тип передачі INTERRUPT використовується тоді, коли необхідно здійснювати обмін через заданий часовий інтервал. Хост забезпечує опитування

із заданим тимчасовим інтервалом і враховує це при плануванні завантаження шини. Інтервал вказується пристроєм при конфігуруванні й може лежати в переділах 0.125-4мс для високошвидкісної шини й 1-255мс для повношвидкістної шини.

Розмір пакета для цього виду передачі коливається в межах від 1 до 1024 байт для високошвидкісної шини й від 1 до 64 байт для повношвидкістної.

Структура потоку даних на шині подібна розглянутої вище. Помітимо, що відсутність даних для кінцевої точки INTERRUPT-IN є штатною ситуацією. Наступний запит хост пошле після закінчення ще одного заданого інтервалу.

Передачі типу ISOCHRONOUS на противагу BULK-передачам використовуються для трафіку, цілісністю доставки якого жертвують на користь швидкості. Контроль цілісності в цьому режимі передачі виробляється тільки за допомогою контрольної суми CRC16. Ушкоджений пакет відкидається на приймаючій стороні, що передає сторона про це не повідомляється. Такий вид передач підходить для пересилання некомпресованих аудіо– і відеопотоків. У таких потоках втрата одного-двох пакетів буде означати лише легеню, можливо навіть непомітне “заїкуватість” для аудіотрафіку або невелика перекручену (найімовірніше просто темну) область для відеотрафіку. Звертаємо увагу на те, що компресовані аудіо/відеодані вкрай чутливі до втрат і не можуть передаватися за допомогою ISOCHRONOUS-передач.

Розмір пакетів даних для цього виду передачі на високошвидкісній шині може бути до 1024 байт, на повношвидкістній – до 1023 байт.

Для того щоб скористатися будь-яким типом передачі крім контрольної пристрій повинне вказати хосту свої можливі режими роботи й доступні в них функції – кінцеві точки, підтримувані ними типи передач. Пристрій ідентифікує себе й конфігурується хостом за допомогою стандартних вимог протоколу USB.

Ідентифікація й конфігурування USB-пристроїв

Одна з найбільш зручних якостей USB-пристроїв це можливість їх «гарячого» підключення, а також підтримка уніфікованих механізмів

самоідентифікація й конфігурування, закріплених у стандарті. Самоідентифікація й загальне настроювання USB-пристроїв реалізується за допомогою спеціальних вимог – розширюваної системи команд протоколу. Існують стандартні – загальні вимоги; вимоги, що ставляться до пристроїв деяких класів, таких, наприклад, як HID-клас. Крім цього пристрої можуть підтримувати додаткові вимоги. Забезпечити підтримку необхідних вимог – першочергове завдання при розробці USB-пристрою.

Кодування вимог USB

Доставка вимог здійснюється за допомогою спеціально призначеної для цього контрольної передачі. Нагадаємо що повна транзакція такого виду передачі складається з 3-х фаз: фази SETUP, фази даних і фази статусу. Дані, що ідентифікують вимогу і його параметри, передаються у фазі SETUP в однойменному пакеті фіксованої довжини 8 байт. У форматі пакета SETUP розрізняють 5 полів (таблиця 3.1).

Таблиця 3.1 – Формат пакета SETUP

№	Поле	Розмір	Опис
1	bmRequestType	1	Бітова маска, що визначає тип вимоги
2	bRequest	1	Номер вимоги
3	wValue	2	Значення поля варіюється залежно від вимоги. Звичайно містить якесь значення
4	wIndex	2	Значення варіюється. Звичайно передає деякий індекс або зсув
5	wLength	2	Кількість байт, для передачі у фазі даних

Поле bmRequestType несе в собі інформацію про тип вимоги. Воно містить 3 субполя. Його формат представлений у таблиці 3.2.

Номера (значення поля bRequest) стандартних вимог і деяких вимог, специфічних для USB-пристроїв класу HID наведені в таблиці 3.3.

Таблиця 3.2 – Формат поля bmRequestType

Біти	7	6	5	4	3	2	1	0
Суб-поле	Напрямок передачі у фазі даних: 0 – OUT 1 – IN	Тип вимоги: 0 – стандартне 1 – класу 2 – виробника 3 – резерв		Одержувач вимоги: 0 – пристрій 1 – інтерфейс 2 – точка 3 – інші одержувачі 4..31 – резерв				

Таблиця 3.3 – Коди стандартних вимог і деяких вимог класу HID

Вимога	Номер	Опис
Стандартні вимоги		
GET_STATUS	0	Одержати стан зазначеного одержувача
CLEAR_FEATURE	1	Одержати деяку властивість одержувача
SET_FEATURE	3	Установити деяку властивість одержувача
SET_ADDRESS	5	Установити адресу
GET_DESCRIPTOR	6	Одержати опис одержувача/властивості
SET_DESCRIPTOR	7	Установити опис одержувача/властивості
GET_CONFIGURATION	8	Одержати номер установленої конфігурації
SET_CONFIGURATION	9	Установити конфігурацію
GET_INTERFACE	10	Одержати номер поточної альтернативної установки для заданого інтерфейсу
SET_INTERFACE	11	Установити номер поточної альтернативної установки для заданого інтерфейсу
SYNCH_FRAME	12	Одержати номер фрейму
Деякі вимоги класу HID		
GET_REPORT	1	Одержати репорт пристрою
SET_REPORT	9	Установити репорт

Призначення полів wValue і wIndex варіюється залежно від вимоги.

Поле wLength показує яка кількість інформації повинне бути передане у фазі даних. Напрямок передачі визначає старший біт поля bmRequestType (таблиця 3.2).

При розробці USB-пристроїв стандарт визначає можливість вмонтувати підтримку нових вимог. При цьому обов'язковим є строге дотримання структури поля bmRequestType і правильне використання поля wLength. Інші поля розроблювач може використовувати за своїм розсудом, перед відправленням хост їх не аналізує. При додаванні вимоги, що буде підтримувати тільки один тип пристроїв , треба біти 6 і 5 поля bmRequestType установлювати в значення 10b, що є ознакою вимоги виробника (продавця). При додаванні вимоги, що буде підтримувати цілий клас пристроїв, буде логічно описати ця вимога як приналежному класу, тобто біти 6 і 5 виставити в значення 01b.

Стандартні вимоги

Вимога GET_STATUS (одержати стан)

Вимога GET_STATUS використовується хостом для визначення стану пристрою, інтерфейсу або кінцевої точки. Ще раз відзначимо, що одержувач вимоги вказується в молодших 5-ти бітах поля bmRequestType (таблиця 3.2). У відповідь на цю вимогу відповідний одержувач повертає 16-ти бітне слово стану.

При обігу вимоги до пристрою:

– поле bmRequestType має значення 10000000b, що кодує IN-передачу у фазі даних (біт 7 дорівнює 1) стандартної вимоги (біти 6, 5 містять 0), зверненого до пристрою (біти 4-0 містять 0);

– поле wValue містить 0 (як і для всіх вимог GET_STATUS – спрямованих будь-якому одержувачеві);

– поле wIndex дорівнює 0;

– поле wLength дорівнює 2 – закріплений розмір відповіді на цю вимогу в байтах для всіх вимог GET_STATUS, адресованого будь-якому одержувачеві.

Пристрій відповідає на цю вимогу 16-ти бітною посилкою, у якій старші 14 біт зарезервовані для майбутнього використання й повинні бути скинуті в нуль. Біт 1 називається RemoteWakeup і показує можливість пристрою самостійно повідомити хосту про вихід із припиненого стану. Як ми вже вказували під час обговорення загальної організації шини USB, пробудження від «сну» від зовнішнього впливу це єдиний випадок, коли пристрій потенційно може почати передачу даних без запрошення хосту. Наявністю такої можливості управляє прапор RemoteWakeup. Хост може скидати й установлювати цей прапор командами CLEAR_FEATURE і SET_FEATURE відповідно. Біт 0 слова стану пристрою має ім'я SelfPowered і показує джерело живлення пристрою: якщо біт установлений – пристрій має незалежне джерело живлення, скинутий – пристрій харчується від шини USB. Хост на спосіб живлення пристрою впливу не має.

При обігу хосту з вимогою GET_STATUS до інтерфейсу:

– поле bmRequestType має значення 10000001b (субполе одержувача містить код інтерфейсу);

– поле wValue містить 0;

– поле wIndex містить номер інтерфейсу. Номер інтерфейса повинен бути припустимим у рамках поточної встановленої конфігурації.

– поле wLength має значення 2.

Інтерфейс передає 2-х байтну посилку, всі біти якої скинуті в нуль і зарезервовані для майбутнього використання.

Поля пакета вимоги GET_STATUS, адресованого точці мають значення:

– bmRequestType – 10000010b (одержувач – точка);

– поля wValue і wLength містять типові для цієї вимоги значення – 0 і 2 відповідно;

– wIndex містить номер припустимої в поточному режимі роботи пристрою точки. Ще раз відмітимо: контрольна точка 0 доступна в будь-якому режимі роботи пристрою.

У слові стану точки зарезервовані й мають нульове значення 15 старших біт, а молодший називається Halt і показує можливість обміну bulk– і interrupt-точок. Якщо цей біт має встановлений, значить точка перебуває в непрацездатному стані й на будь-які спроби хосту зав'язати з нею діалог відмовляє, висилаючи маркер STALL. Хост може управляти цією рисою за допомогою команд CLEAR_FEATURE/SET_FEATURE.

Вимога CLEAR_FEATURE (очистити властивість)

За допомогою цієї вимоги хост має можливість скинути деякі властивості пристрою, інтерфейсу або кінцевої точки.

Фаза даних у транзакції обробки цієї вимоги відсутній, поле wLength для нього завжди нульове. Поле bmRequestType аналогічно такому для попередньої вимоги з відповідними одержувачами за винятком ознаки напрямку передачі у фазі даних. Для вимоги CLEAR_FEATURE і будь-якої іншої вимоги з відсутньою фазою даних ознака напрямки передачі в поле bmRequestType скинутий в 0 і кодує неіснуючу OUT-передачу у фазі даних. Нагадаємо, що, по-перше, напрямок запитів у фазі статусу протилежно напрямку запитів у фазі даних, а по-друге, у відсутності фази дані запити у фазі статусу мають напрямок IN.

Поле wIndex уточнює вимоги отримувача. Для вимоги, зверненої до пристрою це поле містить 0. Для вимоги, спрямованої до інтерфейсу, поле wIndex містить номер інтерфейсу. Інтерфейс із таким номером повинен існувати в поточній активній конфігурації. Для вимоги CLEAR_FEATURE, адресованого кінцевій точці, wIndex задає адреса точки. Точка з такою адресою повинна підтримуватися в поточній активній конфігурації.

Поле wValue визначає властивість, який необхідно очистити. Для пристрою значення 1 у поле wValue означає, що потрібно скинути прапор RemoteWakeup. Це означає, що пристрій втрачає можливість повідомляти хост про вихід із припиненого режиму, тобто втрачає єдину можливість самостійно почати передачу не чекаючи запиту хосту. Для кінцевої точки значення 0 говорить про те, що для неї необхідно скинути прапор стану Halt. Це означає

переклад точки в робочий стан. При відпрацьовуванні цієї вимоги для bulk-IN і interrupt-IN точок необхідно встановити маркер DATA0 для наступного пакета даних.

Вимога SET_FEATURE (установити властивість)

Вимога SET_FEATURE протилежно вимозі CLEAR_FEATURE по призначенню й аналогічно йому за структурою. Вимогою SET_FEATURE хост змушує одержувача встановити властивість, що може бути скинуто аналогічною вимогою CLEAR_FEATURE.

Фаза даних при обробці цієї вимоги також відсутній, поле wLength для нього завжди нульове. Поле bmRequestType для вимог SET_FEATURE, звернених до пристрою, інтерфейсу й точці мають відповідно значення 00000000b, 00000001b і 00000010b.

Для вимоги, адресованого пристрою, значення 1 у поле wValue указує, що пристрій повинне встановити прапор RemoteWakeup, поле wIndex при цьому містить 0. Установка прапора RemoteWakeup означає, що пристрій одержує можливість самостійно сповіщати хост про вихід із припиненого режиму вследствии зовнішнього впливу. У всіх інших випадках пристрій має право почати передачу тільки із запиту хосту.

Для пристроїв, що працюють у високошвидкісному режимі, у полі wValue припустиме значення 2, що зобов'язує пристрій перейти в тестовий режим і провести тест, номер якого зазначений у старшому байті поля wIndex.

Для вимоги, адресованого інтерфейсу, його номер вказується в поле wIndex. Інтерфейс із таким номером повинен існувати в активній конфігурації.

Для вимоги, адресованого точці, її адреса вказується в поле wIndex. Точка з такою адресою повинна існувати в поточному режимі роботи пристрою. Значення 0 у поле wValue визначає, що для точки повинен бути встановлений прапор Halt. Установка цього прапора означає для bulk– або interrupt-точки припинення нормальної роботи: на всі запити хосту вона повинна відповідати маркером STALL.

Вимога SET_ADDRESS (установити адресу)

Вимога SET_ADDRESS застосовується для установки адреси пристрою на шині. Коректними адресами на шині USB є числа від 1 до 127, а також значення 0, що вказує що пристрій є неадресованим.

Нова адреса пристрою воно одержує в поле wValue даної вимоги. Фаза даних при обробці вимоги SET_ADDRESS відсутня – значення поля wLength дорівнює нулю; поле wIndex не використовується й містить нуль. Одержувачем даної вимоги розуміє є тільки пристрій, поле bmRequestType також містить 0.

Вимога GET_DESCRIPTOR (одержати опис)

Вимога GET_DESCRIPTOR використовується для одержання описів визначеного типу.

Насамперед варто звернути увагу на наступне. Всі описи, використовувані в стандарті USB, можна умовно розділити на два види:

– опису загального призначення, застосовувані для всіх USB-пристроїв;

– опису, специфічні для USB-пристроїв деякого класу, наприклад, класу HID.

Що стосується вимог GET_DESCRIPTOR на одержання загальних описів, то для них одержувачем може бути тільки пристрій, що закріплено в специфікації USB. При запиті специфічних описів одержувачем вимог може бути не тільки пристрій. Наприклад, при запиті HID-специфічних описів, одержувачем вимог є інтерфейс, що закріплено в специфікації HID.

Розглянемо докладніше формат вимоги.

Поле bmRequestType для вимог GET_DESCRIPTOR містить значення 100xxxxxb, де xxxxx – двійковий код одержувача, якому адресована вимога. Одержувачі пристрій і інтерфейс кодуються значеннями 00000 і 00001. Список всіх кодів одержувачів для стандартних вимог USB у таблиці 3.2. Значення зазначеного виду в поле bmRequestType означають IN-передачу у фазі дані обробки стандартної вимоги, зверненого відповідному одержувачеві.

Старший байт поля wValue визначає тип опису, що потрібно одержати. Застосовувані значення, їхній зміст, а також зміст молодшого байта, що уточнює тип опису, наведені в таблиці 3.4.

Таблиця 3.4 – Коди типів описів USB-пристроїв

Значення	Тип опису	Призначення молодшого байта
0x01	Пристрій	Не використовується, завжди 0
0x02	Конфігурація	Індекс конфігурації
0x03	Строковий опис	Індекс строкового опису
0x06	Пристрій при роботі в іншому швидкісному режимі	Не використовується, завжди 0
0x07	Конфігурація при роботі в іншому швидкісному режимі	Не використовується, завжди 0
Описи, специфічні для HID-пристроїв		
0x21	Опис HID	Не використовується, завжди 0
0x22	Опис HID-репорту	Не використовується, завжди 0

Значення 0x06 і 0x07 застосовні тільки для пристроїв, що працюють у високошвидкісному режимі. Значення 0x21 і 0x22 застосовні тільки для пристроїв HID-класу.

У пристрої може бути кілька описів типу конфігурація й рядок, тому в молодшому байті поле wValue задається індекс опису цього типу. Для інших типів описів молодший байт не використовується й повинен бути нульовим.

Поле wIndex нульове для всіх типів загальних описів крім рядка. Для строкових описів у поле wIndex задається номер кодової сторінки. Якщо як номер сторінки буде заданий 0, то пристрій повинне повернути перший доступний опис рядка із заданим індексом. Для HID-Описів поле wIndex містить номер інтерфейсу, якому звернена вимога.

Довжина опису заздалегідь не відома, тому хост звичайно зчитує його за два етапи. На першому етапі він задає в поле wLength деяке орієнтовне значення.

Пристрій у відповідь на вимогу з таким прогнозованим значенням розміру передає початкові wLength байт опису. Першим байтом опису передається його розмір. Далі хост, повторює вимога з уточненим значенням поля wLength.

Вимога SET_DESCRIPTOR (установити опис)

Застосовується для установки описів заданого типу. Це вимога аналогічно вимозі GET_DESCRIPTOR за структурою його SETUP-пакета й протилежно йому по дії.

Вимога GET_CONFIGURATION (одержати конфігурацію)

Вимога дає можливість одержати номер поточної конфігурації пристрою.

Поле bmRequestType для цієї вимоги дорівнює 10000000b, а поле wLength містить значення 1. Разом ці поля передають, що хост розраховує одержати від пристрою номер його активної конфігурації у фазі даних у посилці розміром 1 байт. Значення, що повертається, 0 є ознакою того, що пристрій не зконфігуровано.

Поля wValue і wIndex для цієї вимоги не несуть корисної інформації й містять нульові значення.

Вимога SET_CONFIGURATION (установити конфігурацію)

Ця вимога призначена для установки конфігурації пристрою.

Поле bmRequestType містить значення 00000000b, фаза даних при обробці цієї вимоги відсутній і поле wLength містить нульове значення. Поле wIndex не використовується й також містить нульове значення. Номер установлюваної конфигурації передається в поле wValue.

Установка конфігурації номер 0 означає деконфігурування пристрою. Описудоступних конфігурацій хост одержує у відповідь на вимогу GET_DESCRIPTOR. При виборі доступної конфігурації відмінної від 0, необхідно крім того виконати установку інтерфейсу 0 у даній конфігурації й альтернативній установці 0. Нагадаємо, що будь-яка конфігурація містить хоча б один інтерфейс, а будь-який інтерфейс містить хоча б одну альтернативну установку. Установка необхідних інтерфейсу й альтернативної установки

здійснюється за допомогою вимоги SET_INTERFACE. Для всіх bulk– і interrupt-точок напрямку IN в альтернативній потрібно встановити маркер DATA0 для наступного пакета даних. Прапор Halt скидається для всіх кінцевих точок.

Вимога GET_INTERFACE (одержати інтерфейс)

За допомогою цієї вимоги хост може одержати номер поточної альтернативної установки зазначеного інтерфейсу активної конфігурації пристрою.

Це вимога хост адресує інтерфейсу, від якого у фазі даних очікує одержати номер альтернативної установки в пакеті даних розміром 1 байт: поле bmRequestType має значення 10000001b, а поле wLength значення 1. Поле wValue не використовується, а поле wIndex містить номер інтерфейсу, до якого звернено вимогу.

Вимога SET_INTERFACE (установити інтерфейс)

Вимога SET_INTERFACE використовується хостом для вибору альтернативної установки в заданому інтерфейсі активної конфігурації пристрою.

Хост цією вимогою звертається до інтерфейсу, номер якого заданий у поле wIndex, указуючи йому встановити альтернативну установку з номером, визначаємим полем wValue. Фаза даних при обробці цієї вимоги відсутній. Поле bmRequestType містить значення 00000001b, а поле wLength – 0.

Номера доступних інтерфейсів і їхніх альтернативних установок хост довідається за допомогою вимоги GET_DESCRIPTOR. При зміні альтернативної установки для всіх bulk– і interrupt-точок Направленн-IN-напрямку потрібно встановити маркер DATA0 для наступного пакета даних. Прапор Halt для всіх точок скидається.

Вимога SYNCH_FRAME (синхронізувати кадр)

За допомогою цієї вимоги хост контролює номер кадру синхронізації для кінцевої точки в ізохронному режимі передачі з неявною синхронізацією.

Вимоги класу HID

Вимога GET_REPORT (одержати репорт)

Вимога GET_REPORT дозволяє хосту одержати від пристрою репорт – дані в спеціальному форматі – через контрольну точку.

Як це ясно із призначення вимоги, у транзакції його обробки використовується фаза дані напрямки IN. Поле wLength містить довжину репорту. Поле bmRequestType має значення 10100001b, що кодує вимогу класу, звернена до інтерфейсу, при обробці якого є присутнім IN-Фаза даних. Поле wIndex містить номер інтерфейсу, до якого звернене вимога. Поле wValue кодує тип репорту – молодший байт – і його ідентифікатор – старший байт. Ідентифікатор репорту може не використовуватися: у цьому випадку старший байт репорту повинен містити 0. Для визначення типу репорту використовуються наступні значення:

– 1 – INPUT-репорт;

– 2 – OUTPUT-репорт;

– 3 – FEATURE-репорт;

– значення 4-255 зарезервовані для майбутнього використання.

Вимога SET_REPORT (установити репорт)

Ця вимога аналогічно попередній по призначенню й кодуванню полів і протилежно йому по призначенню: за допомогою вимоги SET_REPORT хост може передати дані пристрою через контрольну точку.

Поле bmRequestType містить значення 00100001b. У фазі дані напрямки OUT хост передає пристрою репорт розміром, зазначеному в поле wLength. У молодшому байті поля wValue вказується тип репорту, у старшому – його ідентифікатор. Поле wIndex містить номер інтерфейсу, до якого звернена вимога.

3.2 Розробка структурної схеми

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).

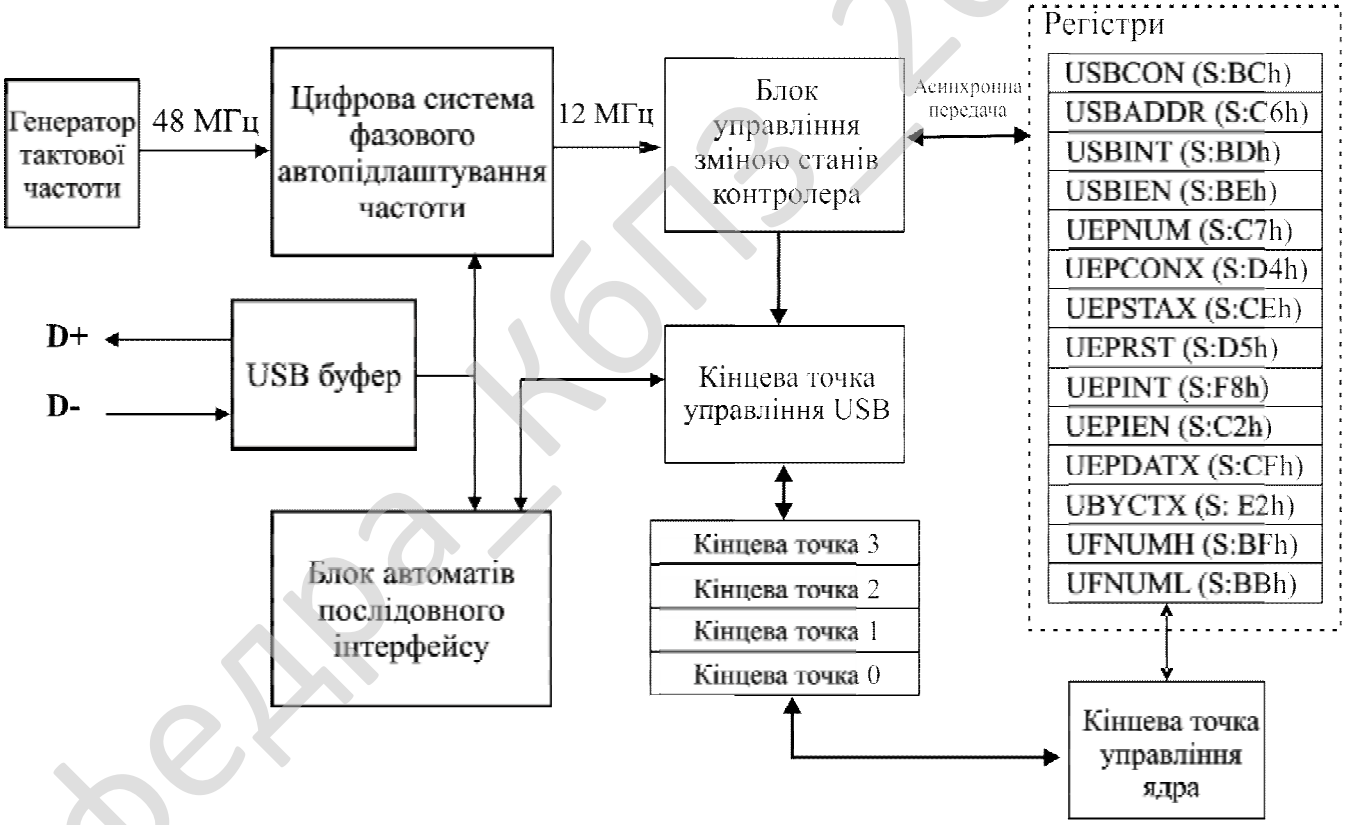

Рисунок 3.5 – Структурна схема системи

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. Частота USB контролера синхронізації завжди повинна дорівнювати 48 Мгц.

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)

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.

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

На рисунку 3.6 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

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

– Емуляція контролера USB.

– Емуляція регістрів USB-інтерфейсу.

– Емуляція кінцевих точок USB-пристроїв.

Блок емуляції контролера USB візуалізує роботу контролера та процес передачі даних через USB. Він містить в собі наступні складові: поточний стан USB-інтерфейсу, читання/запис регістрів, часова діаграма сигналів, блок конфігурації кінцевих точок та блок читання/запису буферів.

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

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

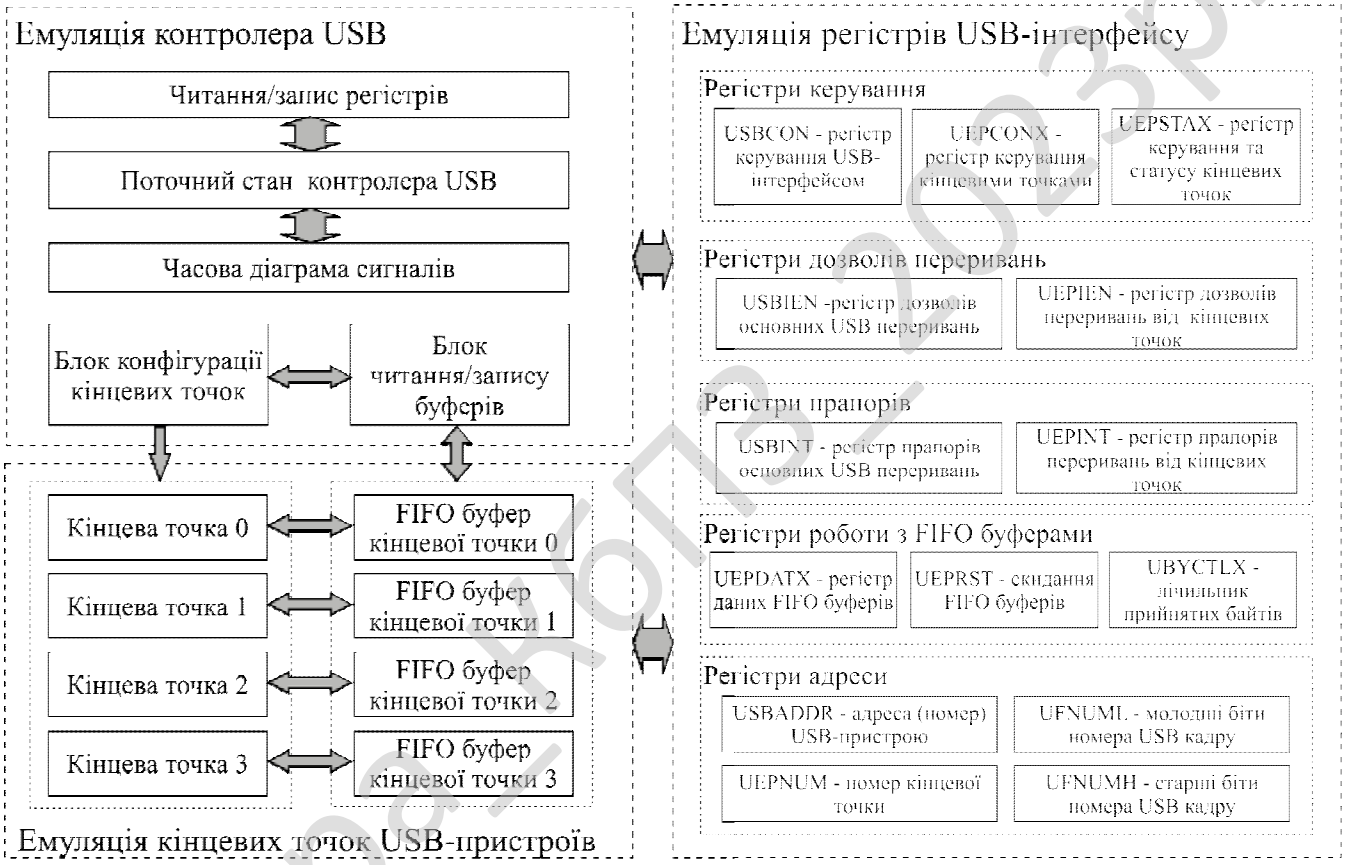

Рисунок 3.6 – Функціональна схема системи

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.7.

Як ми бачимо з рисунка у системі взаємодіють наступні процеси.

Спершу завантажується процес початку/кінця роботи програми.

Він взаємодіє з наступними процесами:

– Процесом вимикання модуля USB.

– Процесом ініціалізації регістрів та вмикання модуля USB.

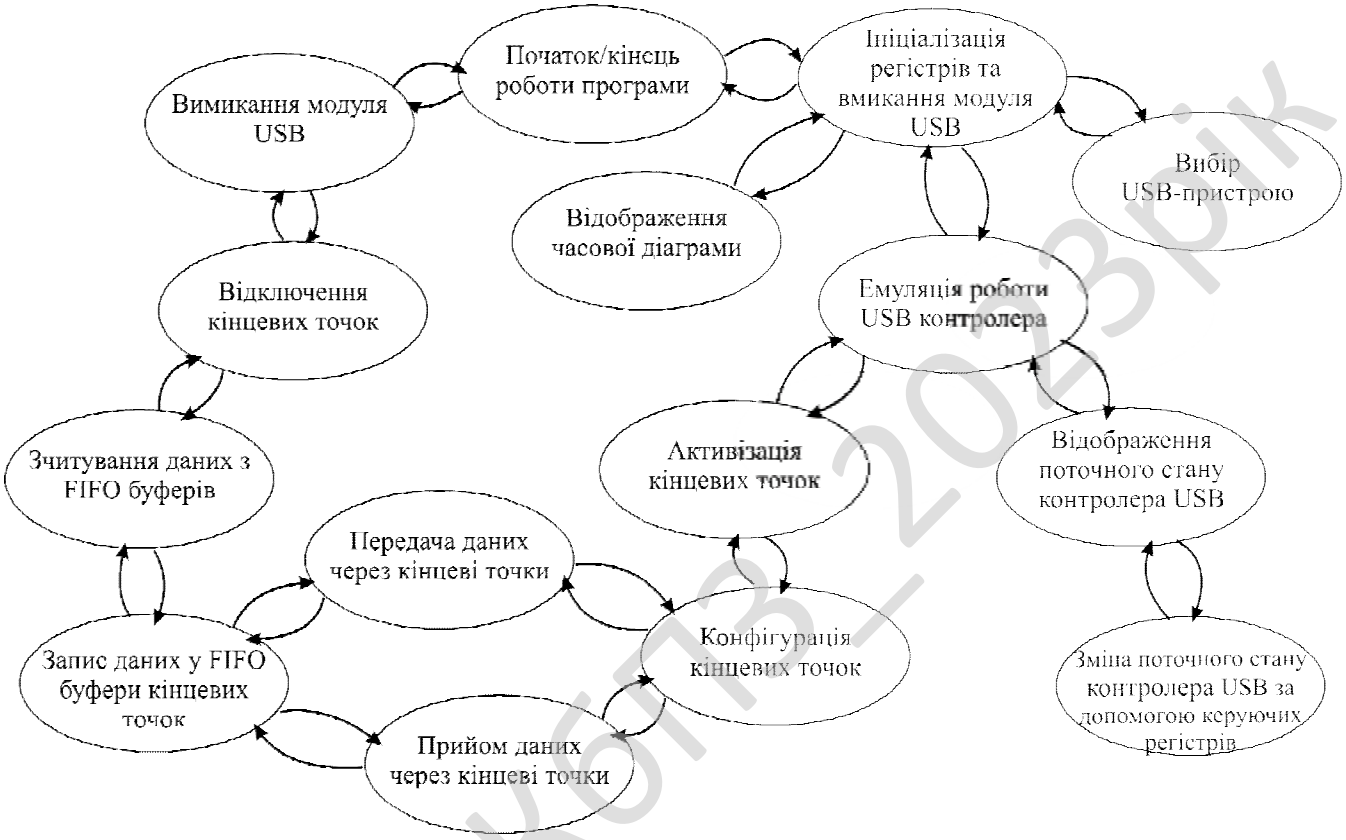

Рисунок 3.7 – Діаграма процесів

Процес ініціалізації регістрів та вмикання модуля USB взаємодіє з наступними процесами:

– Процесом відображення часової діаграми.

– Процесом емуляції роботи USB контролера.

– Процесом вибору USB-пристрою.

Процес емуляції роботи USB контролера взаємодіє з наступними процесами:

– Процесом активізації кінцевих точок.

– Процесом відображення поточного стану контролера USB.

Процес відображення поточного стану контролера USB взаємодіє з процесом зміни поточного стану контролера USB за допомогою керуючих регістрів.

Процес активізації кінцевих точок взаємодіє з процесом конфігурації кінцевих точок.

Процес конфігурації кінцевих точок, у свою чергу, взаємодіє з наступними процесами:

– Процесом прийому даних через кінцеві точки.

– Процесом передачі даних через кінцеві точки

Обидва останні процеси, з іншої сторони, взаємодіють з процесом запису даних у FIFO буфери кінцевих точок.

Процес запису даних у FIFO буфери кінцевих точок взаємодіє з процесом зчитування даних з FIFO буферів кінцевих точок.

Процес зчитування даних з FIFO буферів кінцевих точок взаємодіє з процесом відключення кінцевих точок.

Процесом відключення кінцевих точок є передостаннім процесом у системі й взаємодіє з процесом вимикання модуля USB, який є завершуючим процесом у системі.

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 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків.

Спершу відбувається виведення основного вікна програми. Після цього ініціалізується контролер USB.

Після ініціалізації контролера USB відбувається виведення поточного стану контролера USB.

Якщо необхідно змінити поточний стан контролера USB, то відбувається виконання послідовності наступних дій:

– Запис нових значень у регістри керування.

– Зміна стану контролера USB.

У іншому випадку відбувається введення адреси пристрою у регістр адреси.

Після цього користувач обирає передавати або приймати дані по USB-інтерфейсу.

Якщо він обирає передавати дані по USB-інтерфейсу, то відбувається послідовність наступних дій:

– Викликається функція активізації кінцевих точок пристрою.

– Вводяться дані для передачі у відповідний регістр.

– Відправляються дані у FIFO буфер.

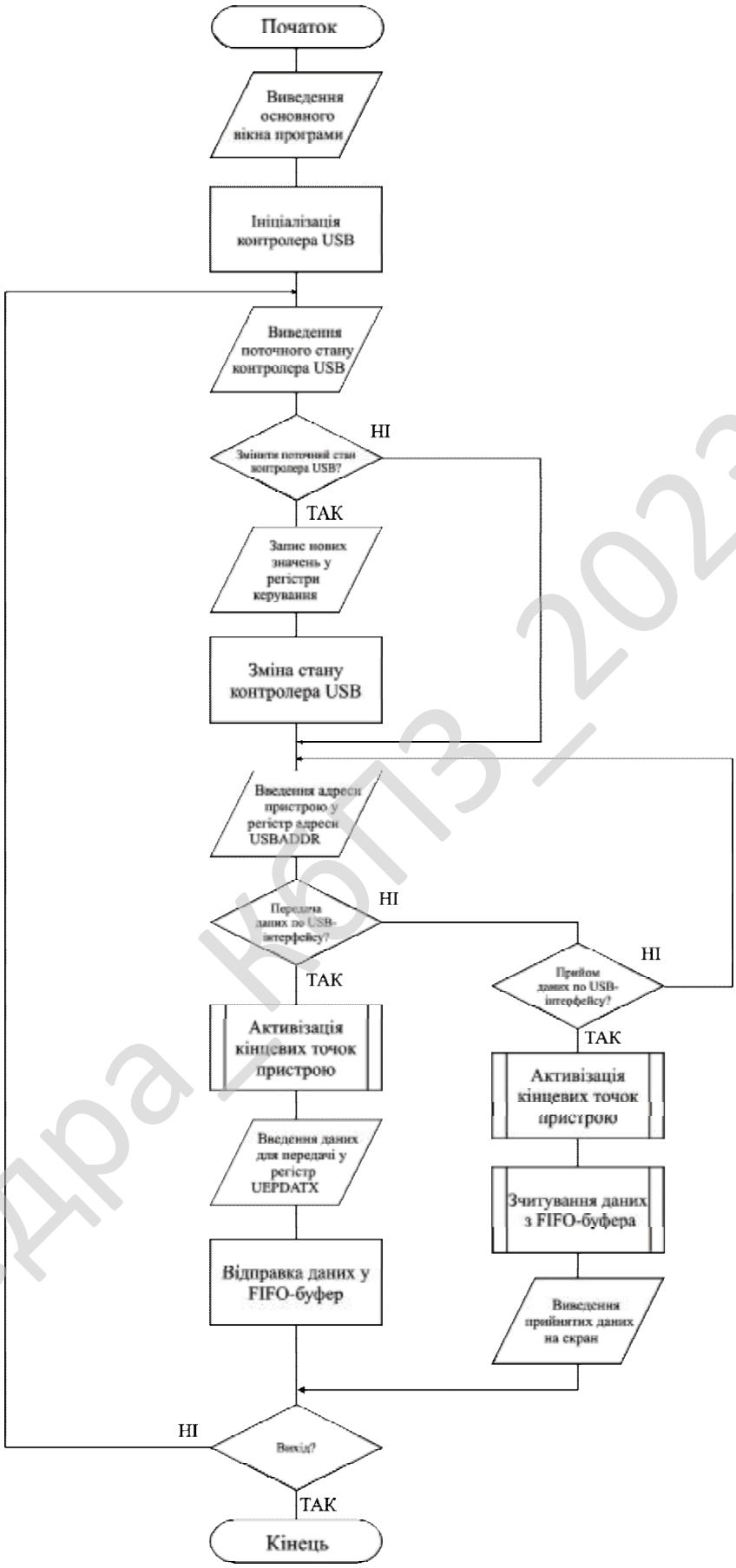

Рисунок 4.1 – Блок-схема роботи основної програми

Якщо ж користувач обирає приймати дані по USB-інтерфейсу, то відбувається послідовність наступних дій:

– Викликається функція активізації кінцевих точок пристрою.

– Зчитуються дані з FIFO буфера.

– Виводяться отримані дані на екран.

Після цього користувач обирає завершувати роботу з програмою, або ні.

Нижче наведемо частину коду, яка реалізує описаний алгоритм.

```
// Створення форм вікон
USEFORM("Main.cpp", Form1);
USEFORM("about.cpp", Form2);
USEFORM("help.cpp", Form3);
USEFORM("signal.cpp", Form4);
// Створення головного вікна
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;
}
//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 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);
}
```

На рисунку 4.2 зображена блок-схема роботи підпрограми активізації кінцевих точок.

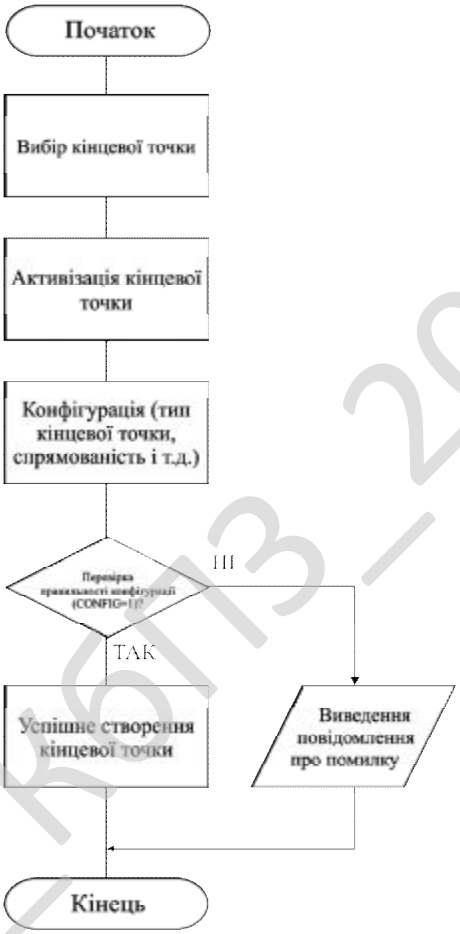

Рисунок 4.2 – Блок-схема роботи підпрограми активізації кінцевих точок

З неї ми бачимо, що підпрограма працює наступним чином.

Спершу відбувається вибір кінцевої точки.

Наступним кроком є активізація кінцевої точки.

Після цього визначається конфігурація кінцевої точки, тобто тип кінцевої точки, спрямованість і т.д.

Потім відбувається перевірка правильності конфігурації.

Якщо конфігурація правильна, то відбувається успішне створення кінцевої

точки.

У іншому випадку відбувається виведення інформації про помилку.

Наведемо частину коду, у якому реалізовано вищеперераховане.

```
//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 4.3 зображена блок-схема роботи підпрограми зчитування даних з FIFO-буферів кінцевих точок. Ця функція працює наступним чином.

Спершу відбувається вибір кінцевої точки.

Після цього відбувається ініціалізація дескриптора USB.

Якщо є дані у черзі, то відбуваються наступні дії:

– Зчитуються дані з буфера.

– Записуються дані у регістр UEPDATX.

– Прочитані дані видаляються з буфера.

– Читаються дані з регістру UEPDATX для подальшої обробки.

Рисунок 4.3 – Блок-схема роботи підпрограми зчитування даних з FIFO-буферів кінцевих точок

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм ДСТУ 8845:2019 – алгоритм симетричного потокового перетворення. В основі ДСТУ 8845:2019 лежить класична схема підсумовуючого генератора подібна генератору SNOW 2.0, SNOW 3.0 та SNOW V. В основі ДСТУ 8845:2019 збережені всі базові операції шифрів сімейства SNOW, а раунд шифрування AES замінений на функцію нелінійної підстановки T, що реалізовує

перестановку елементів скінченного поля GF(2^{64}) за допомогою компонентів національного стандарту симетричного криптоперетворення ДСТУ 7624:2014.

ДСТУ 8845:2019 використовує 256-бітний вектор ініціалізації IV та 256-бітний або 512-бітний секретний ключ K і забезпечує високий та надвисокий рівень стійкості із врахуванням можливого застосування квантового криптографічного аналізу. При розробці алгоритму ДСТУ 8845:2019 орієнтувалися на сучасні 64-бітні обчислювальні системи, тому розмір слова обрано рівним 8 байт. запису байтів застосовують подання від старшого до молодшого. Генератор ключових потоків ДСТУ 8845:2019 у режимі генерації гами шифру схематично приведено у стандарті.

Як випливає з генератора ключових потоків ДСТУ 8845:2019 у режимі генерації гами шифру основними компонентами генератору є регістр зсуву з лінійним зворотнім зв'язком та скінчений автомат на базі якого виконується нелінійне перетворення T. Вхідні дані (ключ шифрування K та вектор ініціалізації IV) використовуються для ініціалізації змінної стану $S_i (i \geq 0)$, яка складається із двох компонент до складу яких входять[12]:

– 16 змінних $s^{(i)}$ – комірок регістра зсуву з лінійним зворотнім зв'язком: $s^{(i)} = (s_{15}^{(i)}, s_{14}^{(i)}, s_{13}^{(i)}, s_{12}^{(i)}, s_{11}^{(i)}, s_{10}^{(i)}, s_9^{(i)}, s_8^{(i)}, s_7^{(i)}, s_6^{(i)}, s_5^{(i)}, s_4^{(i)}, s_3^{(i)}, s_2^{(i)}, s_1^{(i)}, s_0^{(i)})$;

– Два регістри скінченного автомату $r^{(i)}$: $r^{(i)} = (r_2^{(i)}, r_1^{(i)})$. На виході отримуємо ключовий потік (гамма), який формується з 8-байтних слів Z_i.

З рисунку слідує, що відводи регістра зсуву з лінійним оберненим зв'язком побудовані за примітивним над полем GF(2^{64}) поліномом: $f(x) = x^{16} + x^{13} + \alpha^{-1} x^{11} + \alpha$, де α є коренем примітивного над полем GF(2^8) поліному $gz(z) = z^8 + \beta^{170} z^7 + \beta^{166} z^6 + \beta^2 z^5 + \beta^{224} z^4 + \beta^{70} z^3 + \beta^2$. Поле GF(2^8) як і в ДСТУ 7624:2014 побудовано за примітивним на полем GF(2) поліномом $p(y) = x^8 + x^4 + x^3 + x^2 + 1$, а коефіцієнти $g(z)$ подаються через ступінь примітивного елементу β поля GF(2^8), тобто β корінь поліному $p(y)$. Тобто, у нас є вежа полів: $GF(2) \subset GF(2^8) \subset GF(2^{64}) \subset GF(2^{1024})$, де:

– поле $GF(2^{1024})$ задається відводами зворотного зв'язку як фактор кільце $GF(2^{64})[x]/(f(x))$;

– поле $GF(2^{16424})$ задається як фактор кільце $GF(2^8)[z]/(g(z))$;

– поле $GF(2^{1024})$ задається як фактор кільце $GF(2)[y]/(p(y))$.

З вищезазначеного, слідує що період вихідної послідовності становить 2^{1024}.

Структурно в алгоритмі симетричного потокового перетворення ДСТУ 8845:2019 виділяють три основні функції:

– функція ініціалізації, яка приймає в якості вхідних даних 256-бітний вектор ініціалізації IV та 256-бітний або 512-бітний секретний ключ K, і виробляє початкове значення змінної стану $S_0 = (s^{(0)}, r^{(0)})$;

– функція наступного стану Next, яка приймає на вхід зміну стану $S_i = (s^{(i)}, r^{(i)})$ та виробляє наступне значення змінної стану $S_{i+1} = (s^{(i+1)}, r^{(i+1)})$;

– функція ключового потоку Strm, що приймає на вході змінну стану $S_i = (s^{(i)}, r^{(i)})$ та виробляє на виході 64-бітний ключовий потік Z^i.

Також функція Next може виконуватися в двох режимах, в залежності від способу виконання ітерації, як частина реалізації ініціалізації алгоритму ДСТУ 8845:2019 або як частина функції ключового потоку Strm.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблена програма здійснює візуалізацію програмування контролера USB у навчальних цілях.

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

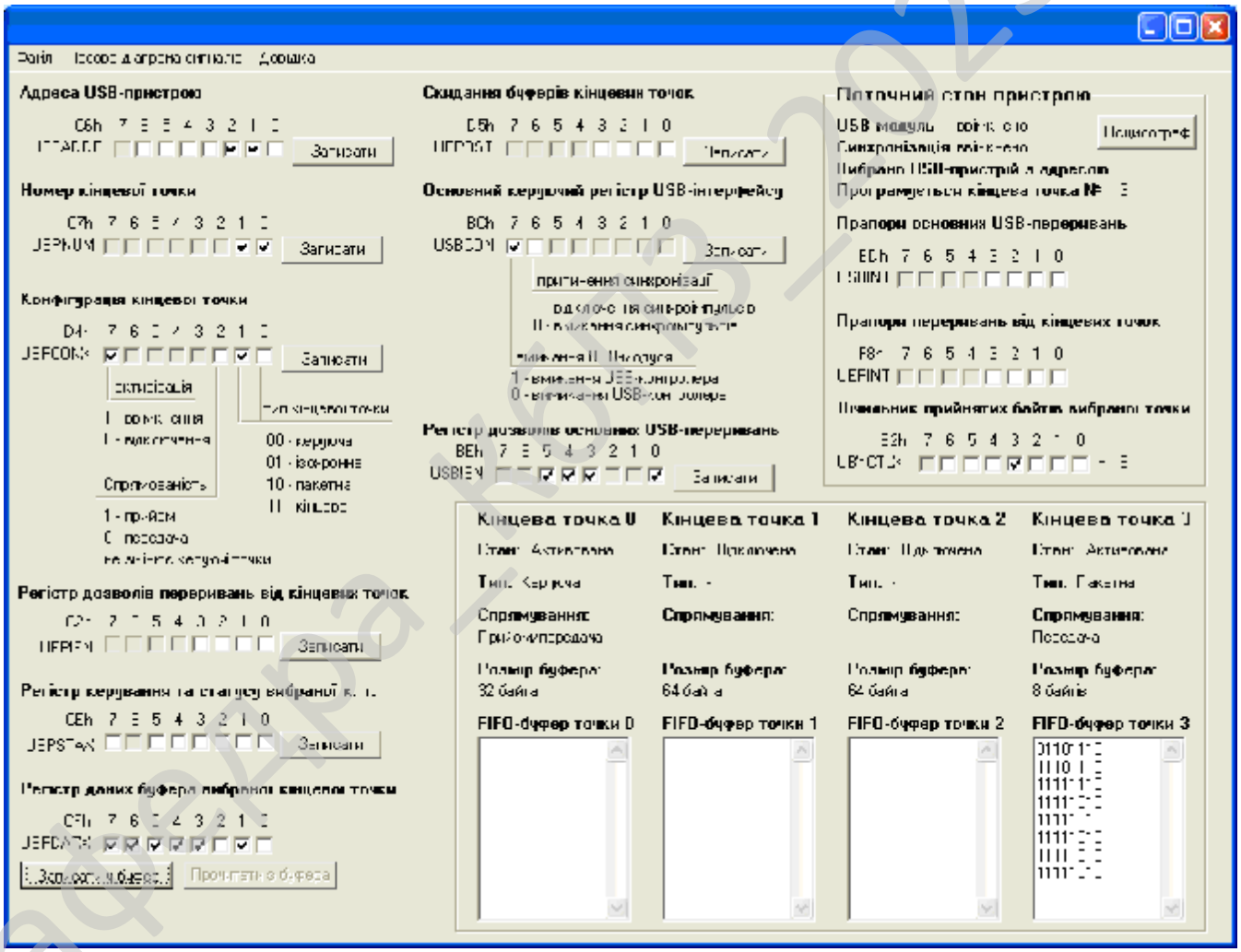

Рисунок 5.1 – Головне вікно програми

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 поточний стан пристрою і значення регістрів прапорів

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

Активізація та конфігурування кінцевих точок відбувається за допомогою регістру конфігурування.

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 використати 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).

Керувати роботою USB-інтерфейсу можна за допомогою регістру керування USB-інтерфейсом USBCON та регістру дозволів основних USB-переривань USBIEN.

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 «Осцилограф», або вибрати пункт меню Часова діаграма сигналів->Показати осцилограф. Після чого з’явиться вікно, зображене на рисунку 5.2.

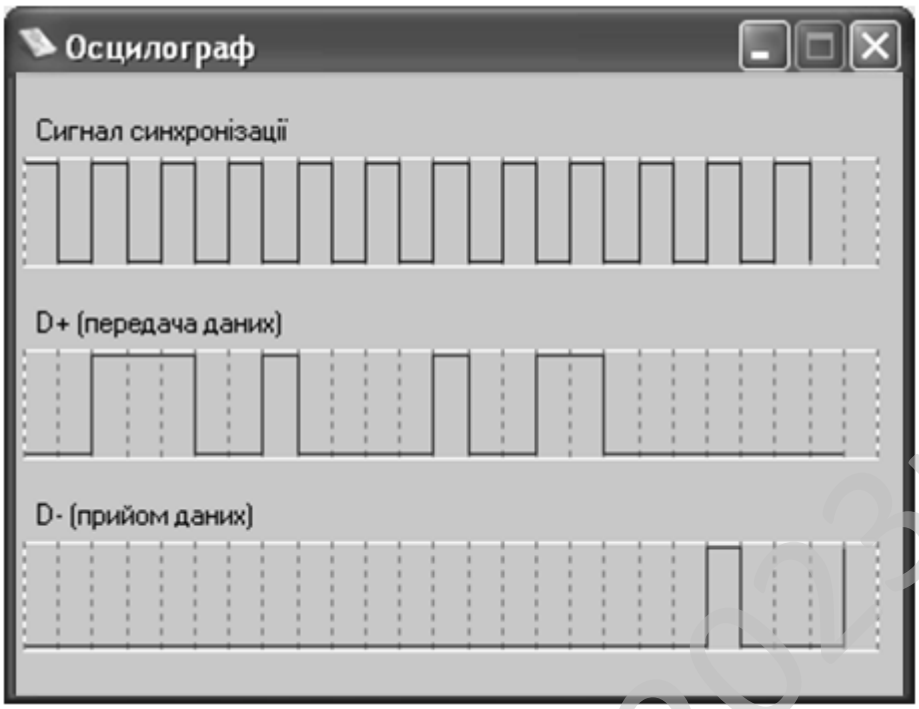

Рисунок 5.2 – Вікно «Осцилограф», що показує часову діаграму сигналів USB-інтерфейсу

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 5.3).

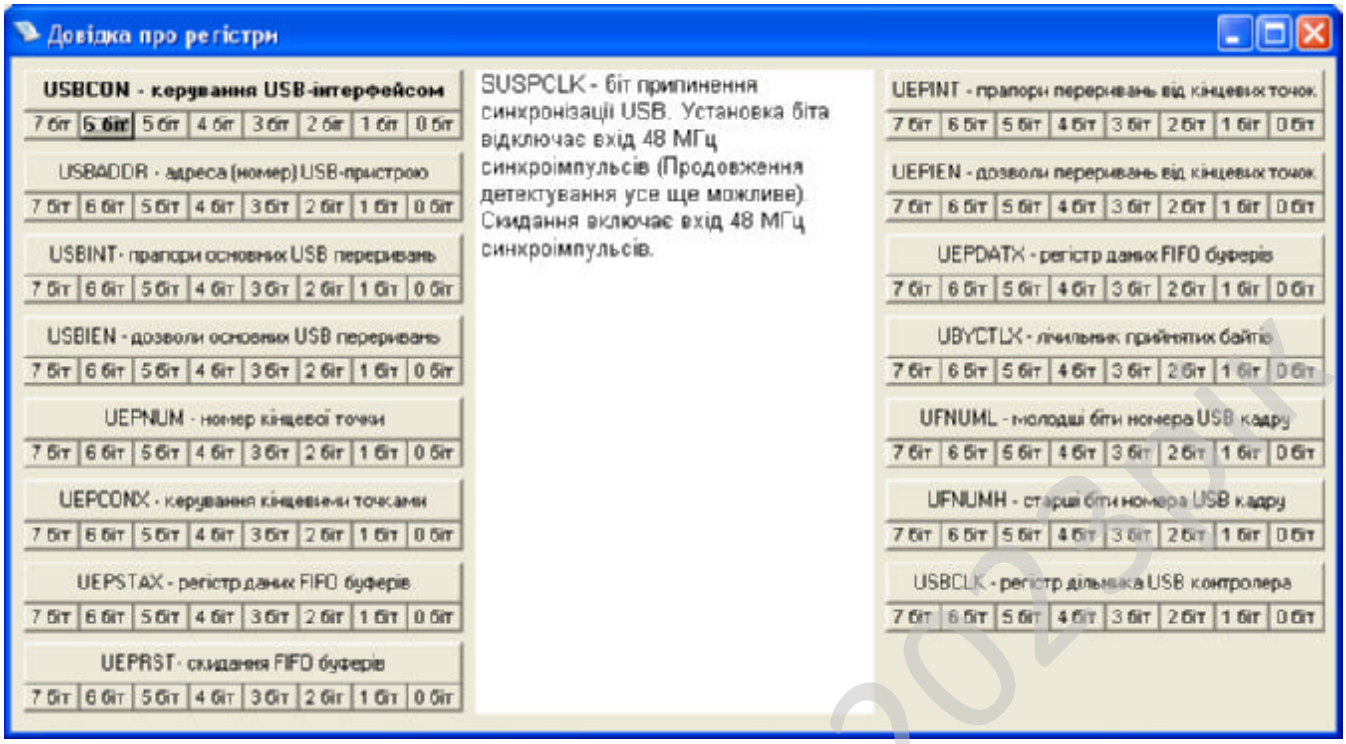

Рисунок 5.3 – Довідка про регістри USB-інтерфейсу

Переглянути короткі відомості про програму та її автора можна натиснувши пункт меню Довідка->Про програму…, після чого з'явиться діалогове вікно, зображене на рисунку 5.4.

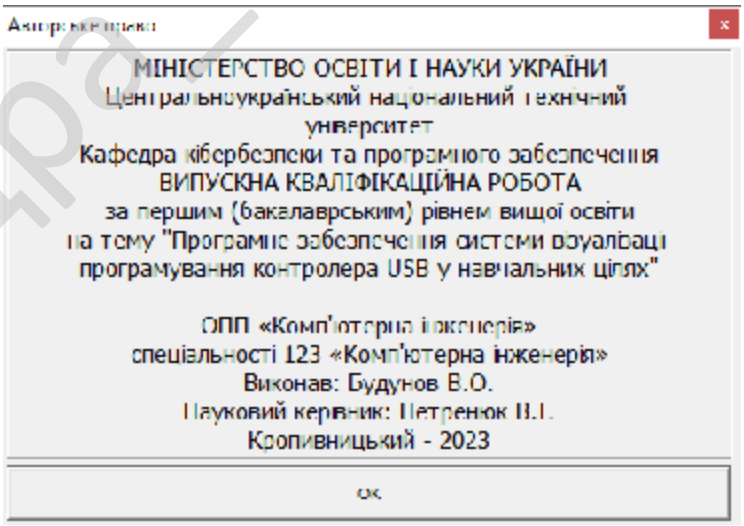

Рисунок 5.4 – Вікно «Про програму…»

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи візуалізації програмування контролера USB у навчальних цілях.

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 програмування контролера USB у навчальних цілях.

– Досліджена система візуалізації програмування контролера USB у навчальних цілях.

– На основі отриманих результатів досліджень створена програмна реалізація системи візуалізації програмування контролера USB у навчальних цілях.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання візуалізації програмування контролера USB у навчальних цілях.

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 програмування контролера USB у навчальних цілях. 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 ДСТУ 8845:2019.

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. 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)**.

2. 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)**.

3. 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

4. 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

5. 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

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. 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.

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

24. 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

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

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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: ХННІ ДВНЗ «УБС». – 2016. – С. 174-178.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

51. 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.

52. 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.

53. 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.

54. 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.

55. 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.

56. 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.

57. 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.

58. 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.

59. 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

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи візуалізації програмування контролера USB у навчальних цілях.

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 програмування контролера USB у навчальних цілях.

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 програмування контролера USB у навчальних цілях;

– 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 – 86 аркушів.

8 Етапи розробки

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

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 5.06.2023 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

_____ Петренюк В.І.

Програмне забезпечення системи візуалізації програмування контролера USB у навчальних цілях

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 50

Літера: РП

Кропивницький – 2023 року

Файл ProjectUSB.cpp основної програми

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

#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
// Створення форм вікон
USEFORM("Main.cpp", Form1);
USEFORM("about.cpp", Form2);
USEFORM("help.cpp", Form3);
USEFORM("signal.cpp", Form4);
//---------------------------------------------------------------------------
// Створення головного вікна
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 "about.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:      // Компоненти IDE-управління
        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:    // Визначається користувачем
public:           // Визначається користувачем
        __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:      // Компоненти IDE-управління
        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:    // Визначається користувачем
public:           // Визначається користувачем
        __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:      // Компоненти IDE-управління
        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:    // Визначається користувачем
public:           // Визначається користувачем
        __fastcall TForm3(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm3 *Form3;
//---------------------------------------------------------------------------
#endif
```

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

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

#include <vcl.h>
#pragma hdrstop

#include "about.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 about.h – бібліотека для файлу about.cpp

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

#ifndef aboutH
#define aboutH
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Graphics.hpp>
//---------------------------------------------------------------------------
class TForm2 : public TForm
{
__published:    // Компоненти IDE-управління
        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:         // Визначається користувачем
public:           // Визначається користувачем
        __fastcall TForm2(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm2 *Form2;
//---------------------------------------------------------------------------
#endif
```