Центральноукраїнський національний технічний університет
Центр заочної та дистанційної освіти
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти

на тему

“Програмне забезпечення системи інтерактивного керування структурованою кабельною системою мережі”

Виконав здобувач вищої освіти
IV курсу, групи КІ-20ПЗ
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Криворoженко Н.В.
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук
_____ Смірнова Т.В.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Центр *Заочної та дистанційної освіти*

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 2024 року

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

Криворожeнко Наталія Володимирівна
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи інтерактивного керування структурованою кабельною системою мережі*

2. Керівник роботи *Смірнова Тетяна Віталіївна, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 139-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи інтерактивного керування структурованою кабельною системою мережі*

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 2024 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	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Смірнова Т.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

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

АНОТАЦІЯ

Кривороженко Н.В. Програмне забезпечення системи інтерактивного керування структурованою кабельною системою мережі. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи інтерактивного керування структурованою кабельною системою мережі.

Метою розробки є програмне забезпечення системи інтерактивного керування структурованою кабельною системою мережі.

Результат роботи – програмна реалізація системи інтерактивного керування структурованою кабельною системою мережі.

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 з ОС Windows 10/11.

Програму розроблено в середовищі Delphi 10.4.

Ключові слова: комп'ютерна інженерія, структурована кабельна система мережі

ABSTRACT

Kryvorozhenko N.V. The software of the system of interactive management of the structured cabling system of the network. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for the system of interactive management of the structured cable system of the network.

The purpose of the development is the software of the system of interactive management of the structured cable system of the network.

The result of the work is the software implementation of the system of interactive management of the structured cable system of the network.

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 a PC with Windows 10/11 OS.

The program was developed in the Delphi 10.4 environment.

Keywords: computer engineering, structured cabling network system

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АСУ – автоматизована система управління

ДСТ – держстандарт

ЕМЗ – електромагнітний замок

ЕОМ – електроно-обчислювальна машина

ІСО – міжнародний стандарт

ОПС – охоронно-пожежна сигналізація

ПЗ – програмне забезпечення

ПЗП – постійний запам’ятовуючий пристрій

ПК – програмний комплекс

СКУД – система контролю та управління доступом

СУД – система управління доступом

PIN – personal identification cod – персональний ідентифікаційний код

RTE – Request To Exit – кнопка «Вихід»

ВСТУП

Актуальність теми. Інтерактивне управління охоплює не більше 15–20% всіх інсталюємих портів структурованих кабельних систем (СКС). У такій ситуації примусова реалізація комутаційного поля винятково за схемою крос-коннекта неможлива у принципі. Тому, щоб стимулювати замовників до застосування встаткування систем інтерактивного управління (СІУ) у проектах побудови СКС, постачальники повинні шукати ефективні шляхи його адаптації до схеми інтерконнекта.

Популярності СІУ сприяло те, що її застосування дозволяє підвищити ефективність адміністрування інформаційної проводки: зменшується ймовірність ненавмисної помилки в процесі зміни конфігурації кабельної системи, автоматизує ряд рутинних операцій поточного адміністрування, спрощується інвентаризація телекомунікаційних ресурсів і т.д.

Вимоги до кваліфікації обслуговуючого персоналу в результаті знижуються: у ряді фірмових описів СІУ підкреслюється, що перемикання може виконувати рядовий технік. Однак при складанні робочого завдання на ці процедури необхідно враховувати безліч зовнішніх факторів, для чого потрібний інший рівень кваліфікації й інженерні знання. Завдяки впровадженню СІУ збільшується продуктивність праці інженерно-технічного персоналу, що досягається за рахунок використання спеціалізованих засобів електронної обробки даних на фізичному рівні інформаційної системи, де їхнє застосування споконвічно не передбачалося моделлю відкритих систем. Ресурси, що звільнилися, можуть бути спрямовані на рішення інших завдань.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи інтерактивного керування структурованою кабельною системою мережі.

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

– Огляд існуючих систем інтерактивного керування структурованою кабельною системою мережі.

– Дослідження системи інтерактивного керування структурованою кабельною системою мережі.

– Програмна реалізація системи інтерактивного керування структурованою кабельною системою мережі.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі інтерактивного керування структурованою кабельною системою мережі.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи інтерактивного керування структурованою кабельною системою мережі, 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

Апаратна частина СІУ являє собою впорядкований набір датчиків для контролю підключення комутаційних шнурів до розеточних модулів роз’ємів комутаційних панелей. Вимога впорядкованості є обов'язковим і випливає із заборони на які-небудь паралельні підключення до ланцюгів передачі інформаційних сигналів у межах стаціонарних ліній. Тільки при його виконанні вдається однозначно прив'язати конкретний датчик до певного порту активного або пасивного мережного пристрою. Інакше кажучи, СІУ використовує непряму схему виявлення фактів підключення вилки до розетки порту і її відключення.

У найперших моделях цього датчика чутливий елемент працював за принципом замикання спеціальних контактів розеток портів, що з'єднуються, за допомогою додаткового провідника комутаційного шнура. Крім того, у випускаємих серійно СІУ застосовуються чутливі елементи на основі світлового затвора зі схемою роботи як «на прохід», так і «на відбиття», а також виконані у формі механічного мікроперемикача. Цей функціональний вузол датчика може бути реалізований з використанням статичної або перезаписуваної мітки RFID з різним обсягом збереженої інформації й т.п.

Електричний сигнал, генеруємий окремими датчиками апаратної частини СІУ, зчитується спеціалізованим керуючим ПЗ, що відповідно до вимог стандартів на адміністрування реалізується у вигляді бази даних. У перелік його основних функцій входять автоматизоване заповнення кабельного журналу, ведення реєстру подій, допомога системному адміністраторові при підготовці робочих завдань по наявних шаблонах, генерація різних звітів по запитах, організація діалогу з користувачами з наданням їм різних повноважень і т.д. Деякі виробники підтримують додатковий ряд сервісів – наприклад, часткову

автоматизацію формування робочого завдання за рахунок роботи в графічному режимі із залученням наочних і інтуїтивно зрозумілих процедур буксирування (drug-and-drop).

Сервер БД системи інтерактивного управління випробовує порівняно невелике навантаження від виконання штатних процедур адміністрування. надлишок, Що Утвориться, обчислювальної потужності цілком може бути спрямований на рішення найрізноманітніших допоміжних і сервісних завдань. Тому на програмну частину СІУ можуть бути покладені функції, досить далекі від адміністрування СКС, – наприклад по виявленню нових IP-пристроїв у мережі, у тому числі підключених туди несанкціоновано.

СКС являє собою сукупність стаціонарних ліній, що з'єднуються між собою й комутаційними шнурами, що підключаються до активного мережного встаткування. При цьому важливе значення мають наступні фактори:

– фокусною областю застосування СІУ є горизонтальна підсистема;

– у порівнянні з магістральною підсистемою горизонтальна відрізняється високою частотою зміни конфігурації;

– що обслуговується СКС активне мережне встаткування представлене головним чином комутаторами рівня робочої групи для локальної мережі.

1.2 Область застосування

Будь-яка розробка, який би вдалою вона не була, дозволяє лише в більшому або меншому ступені наблизитися до ідеалу. Так і чудова по своєму задумі СІУ володіє рядом обмежень із погляду забезпечуваних функціональних можливостей. Їхнє усунення стає однією із пріоритетних завдань, а кінцевою метою є поліпшення споживчих властивостей продукту. Проблема ускладнюється тим, що, залежно від ситуації, може досить істотно мінятися не тільки саме поняття ідеалу, але навіть критерії його якісної оцінки, не говорячи вже про кількісну.

У технічних приміщеннях, виділюваних під обслуговування інформаційної системи, установлюється групове комутаційне встаткування. Звичайно воно оформляється у вигляді зручних для інсталяції й експлуатації панелей, які оснащені модульними розніманнями – по 24-48 портів на 1U монтажної висоти. Із цих панелей формується комутаційне поле інформаційної кабельної системи. Останнє, відповідно до правил побудови СКС, розділяється на окремі функціональні секції зі строго певним призначенням.

Підключення активного мережного встаткування до кабельної системи виробляється різними способами. Відповідно до нормативів профільних стандартів вся їхня сукупність може бути зведена до двох основних схем: крос-коннекта й інтерконнекта.

У випадку крос-коннекта порт комутатора спочатку підключається до лінійної сторони проміжної панелі, що позначається як панель відображення. Вибір необхідного для цього шнурового виробу визначається конструктивними особливостями панелі: з її лінійної сторони можуть застосовуватися звичайні кінцевики з IDC-контактами або I-адаптери, тобто підключення може здійснюватися як монтажними, так і звичайними комутаційними шнурами. Подальше з'єднання з панеллю певної функціональної секції виробляється з користувальницької сторони за допомогою звичайних комутаційних шнурів. Іноді вони позначаються як джампери.

Таким чином, крос-коннект має дві характерні риси, досить важливі з погляду вибору конструктивних і проектних рішень в області СІУ: по-перше, у складі вже сформованого простого тракту присутні два шнурових вироби, а по-друге, при зміні конфігурації фізичного рівня інформаційної системи адміністраторові взагалі не потрібний прямий доступ до активного мережного встаткування.

При інтерконнекті відсутнє проміжна ланка у вигляді панелі відображення, тому з'єднання портів комутатора й панелей лінійних кабелів СКС здійснюється за допомогою єдиного апаратного шнура. У порівнянні із крос-

коннектом схема інтерконнекта застосовується значно частіше. Це обумовлено в першу чергу її простотою й меншою трудомісткістю при реалізації. Із проектної точки зору очевидне перевага складається в збільшенні ефективної щільності портів одиночного монтажного конструктива приблизно на 20%, що досягається за рахунок меншої кількості панелей. Немаловажне значення має природність інтерконнекта з користувальницької точки зору.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи інтерактивного керування структурованою кабельною системою мережі, 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

Кабельні системи адміністративних будинків

Структурована кабельна система являє собою основу інформаційної інфраструктури підприємства, що поєднує в єдину систему безліч інформаційних сервісів різного призначення, включаючи локальні обчислювальні й телефонні мережі, системи безпеки, відеоспостереження й т.д.

Структурована кабельна система (СКС) – кабельна система будинку або групи будинків, що має певну ієрархію й складається з елементів, які інтегруються в єдину систему й експлуатуються відповідно до певних правил.

До таких елементів ставляться: мідні й оптичні кабелі, крос-панелі, сполучні шнури, кабельні рознімання, модульні гнізда, інформаційні розетки й інше допоміжне устаткування.

Переваги СКС:

– Топологія ієрархічного кільця.

– Модульность побудови.

– Гнучкість і простота в експлуатації.

– Мінімізація експлуатаційних витрат.

– Архітектура кабельних систем

Кабельні системи будуються по ієрархічному принципі й складаються з наступних підсистем:

– Підсистеми робочої області, устаткування якої призначено для підключення користувачів локальної обчислювальної мережі (ЛОМ);

– Горизонтальної підсистеми, що забезпечує з'єднання робочих місць із кроссовим устаткуванням, розташованим у мережному центрі;

– Підсистеми адміністрування, що містить у собі кроссове встаткування центрального мережного центра, призначене для комутації сигналів, переданих по мідному кабелі.

Внутрицехові кабельні системи

Внутрицехові кабельні системи призначені для організації передачі даних і зв'язку усередині окремих цехів підприємства, включаючи мережі першого рівня й промислові мережі.

Підтримують наступні додатки: передача голосу, передача даних, передача цифрового й аналогового відеосигналу, системи автоматизації будинків, споруджень, промислового й енергетичного встаткування, системи управління енергетикою й енергообліку, системи протипожежної безпеки й пожежної сигналізації, системи забезпечення безпеки й контролю доступу, системи обігріву, вентиляції й кондиціювання.

Внутрицехові кабельні системи підрозділяються на:

– телекомунікаційні багатофункціональні кабельні системи, які виконуються на основі ВОЛЗ, МКС і СКС.

– промислові мережі.

Внутрицехові телекомунікаційні багатофункціональні кабельні системи на базі ВОЛЗ і МКС виконуються у відповідність із вимогами, пропонованими до магістральних кабельних систем.

Внутрицехові СКС виконуються у відповідність із вимогами, пропонованими до кабельних систем адміністративних будинків.

Промислові мережі призначені для організації фізичного й логічного зв'язку датчиків і виконавчих механізмів із системним інтелектом, роль якого виконують програмувальні логічні контролери (PLC) або промислові комп'ютери таким чином, щоб інформація із цього рівня була доступна загальнозаводській інформаційній системі.

Промислова мережа виступає основою для побудови розподілених систем збору даних і управління. Реалізація промислових мереж здійснюється на основі різних середовищ передачі даних, включаючи мідні, оптичні лінії зв'язку, а також застосування бездротових технологій.

Концепція побудови мережної інфраструктури на об'єкті містить у собі:

– Мережі нижнього (польового) рівня. Мережі даного рівня призначені для дистанційного збору даних від датчиків, виконавчих механізмів і їхніх вторинних приладів, щитів станцій управління, автоматичних вимикачів з вимірювальними пристроями, мікропроцесорних терміналів захисту встаткування електропостачання й дистанційного управління приводами.

– Мережі 1-ого (контролерного) рівня (PROFIBUS, MODBUS). Мережі даного рівня забезпечує передачу даних по протоколі PROFIBUS (MODBUS) між контролерами, пристроями збору й передачі даних, локальними панелями, датчиками положення, пристроями управління приводами й т.д., а також передачу даних на верхній рівень.

– Мережі 2-ого (АСУ ТП) рівні (Ethernet). Основним завданням 2-ого рівні АСУ ТП виступає реєстрація, зберігання й управління даними про технологічний процес. Основу цього рівня становить база даних реального часу. На підставі цих даних можна здійснювати візуалізацію, реєстрацію й зберігання історії тривог і статусів техпроцесу, управління на основі математичної моделі техпроцесом. Доступ до технологічної інформації на даному рівні має більше число автоматизованих робочих місць оперативного рівня.

– Мережі 3-ого рівні (АСУ П) (Ethernet). Третій рівень АСУ П призначений для інтеграції автоматизованої системи управління технологічним процесом (АСУ ТП) з автоматизованої системи управління підприємством (АСУ П). У деяких випадках, залежно від розмірів підприємства, можливе включення більше високого (четвертого) рівня й забезпечення інтеграції з вищим керівництвом, що може бути розташоване в різних країнах.

Системи управління інфраструктурою

Основне призначення систем управління інфраструктурою – централізоване управління всіма компонентами інформаційної інфраструктури підприємства: серверним устаткуванням, ПЕОМ, інформаційними мережами об'єкта, СУБД і додатками, використовуваними на об'єкті, службою підтримки користувачів.

Система управління інформаційною інфраструктурою є комплексним інтегральним набором підсистем, взаємозалежних між собою на основі технологій компаній – виробників програмного забезпечення, таких як Computer Associates, Symantec, Embarcadero.

Функціонально складається з 12 підсистем залежно від об'єктів управління:

– Підсистема моніторингу програмно-апаратного комплексу, подій прикладного, системного ПЗ й СУБД. Підсистема призначена для комплексного й всебічного управління й моніторингу різнорідної програмно-апаратної інфраструктури серверного комплексу об'єкта автоматизації.

– Підсистема моніторингу й управління завантаженістю інформаційних мереж. Підсистема призначена для оперативного моніторингу ключових параметрів мережних пристроїв, ідентифікації мережних проблем і пошуку методів їхнього рішення.

– Підсистема моніторингу й управління продуктивністю СУБД. Підсистема призначена для всебічного контролю прикладних систем на етапі їхньої експлуатації, оцінки їхньої продуктивності й ухвалення рішення по настроюванню раціонального управління ресурсами.

– Підсистема моніторингу web-серверів. Підсистема призначена для оперативного моніторингу ключових параметрів web-серверів.

– Підсистема управління конфігурацією комп'ютерних ресурсів. Підсистема призначена для оперативного одержання інформації про состав, а

також його зміні, програмних і апаратних конфігурацій комп'ютерів, що перебувають у зоні експлуатації й підключених у корпоративну мережу.

– Підсистема управління поширенням відновлень системного й прикладного ПЗ ПЕОМ і серверів. Підсистема призначена для централізованого й регламентованого відновлення системного й прикладного програмного забезпечення на ПЕОМ і серверах об'єкта автоматизації.

– Підсистема віддаленого адміністрування ПЕОМ і серверів. Підсистема призначена для віддаленого централізованого адміністрування клієнтських ПЕОМ і серверів у різних режимах з використанням вироблених політик управління.

– Підсистема управління сервісною службою підтримки користувачів АСУ. Підсистема призначена для забезпечення централізованого контролю взаємодії між клієнтами й сервісними службами, у контексті дозволу інцидентів і виконання заявок на обслуговування.

– Підсистема резервного копіювання, архівування й відновлення даних для серверів. Підсистема резервного копіювання призначена для централізації й автоматизації управління операціями резервного копіювання й відновлення інформаційно-технологічних даних, використовуваних у роботі інформаційних систем і ресурсів об'єкта автоматизації.

– Підсистема антивірусного захисту додатків і даних на ПЕОМ і серверах. Підсистема призначена для комплексного й керованого антивірусного захисту ПЕОМ і серверів об'єкта автоматизації.

– Підсистема контролю над діями персоналу при роботі з інформаційними системами (системні дії). Підсистема призначена для всебічного й централізованого аудита дій користувачів в інформаційних системах об'єкта автоматизації.

– Підсистема контролю над діями персоналу при роботі з інформаційними системами (дії із СУБД). Підсистема призначена для організації захисту критичних даних, створення умов забезпечення їхньої конфіденційності, а також

реалізації законодавчих вимог, за допомогою встановлення постійного спостереження за всіма звертаннями до баз даних і за всією пов'язаною із цим активністю.

Всі підсистеми здійснюють між собою взаємодію, заснована на убудованих інтеграційних механізмах. Дані механізми забезпечують єдину консолідовану шину подій, що відбуваються в різних підсистемах, синхронізацію різних об'єктних сховищ даних, систему побудови оброблювачів подій, оповіщень відповідальних осіб і адміністраторів, візуалізацію й звітність, а також управління інцидентами.

Система інтегрується в корпоративну службу каталогів MS Active Directory, а також використовує (у деяких підсистемах) транспортний механізм повідомлення про інциденти корпоративної електронної пошти й службу миттєвих повідомлень ICQ.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання

паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

- Додана підтримка Metal Driver GPU для macOS і iOS.
- Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.
- Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.
- Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.
- Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.
- Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

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 інтерактивного керування структурованою кабельною системою мережі.

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

Для більшої наочності ми будемо розглядати тільки прості тракти, у складі яких є присутнім усього одна стаціонарна лінія, тим більше що з погляду інтерактивного управління зміна конфігурації складених трактів виконується аналогічно. Дійсно, процедура багатоточкового адміністрування може бути представлена як послідовність операцій одноточкового адміністрування, виконуваних послідовно в кожному комутаційному вузлі.

Стосовно до апаратної частини завдання створення СІУ зводиться в основному до відпрацьовування конструкції датчика підключення шнура до панелі в широкому змісті цього терміна. Одна з таких панелей завжди являє собою функціонально закінчену конструктивну одиницю з одним або декількома рядами розеток модульних роз'ємів, які через свої кінцевики підключені до інсталяційних кабелів. Перехід на розетки інших типів вимагає лише незначних конструктивних змін у частині СІУ.

У випадку інтерконнекта під другою панеллю мається на увазі та пластина корпуса активного мережного встаткування (у більшості сучасних пристроїв – лицьова), на яку виведені розетки портів. При крос-коннекті функції другої панелі виконує панель відображення. Остання по конструктивному виконанню (принаймні, користувальницької сторони) ідентична панелям для підключення лінійних кабелів. Таким чином, важливою особливістю крос-коннекта є те, що всі інтерфейси змінюваної частини тракту, що перебуває під безпосереднім контролем СІУ, мають конструктивну симетрію.

Завдяки симетричності окремих функціональних секцій, панелі яких з'єднуються шнурами в процесі формування трактів, істотно спрощується створення апаратної частини датчика підключення. Отже, при впровадженні СІУ

більше вигідної виявляється реалізація комутаційного поля за схемою крос-коннекта, що явно суперечить практиці побудови кабельних систем, що зложилася в галузі.

Техніко-економічні переваги СІУ повною мірою проявляються лише у великих проектах. Як результат, інтерактивне управління охоплює не більше 15-20% всіх інсталюємих портів СКС. У такій ситуації примусова реалізація комутаційного поля винятково за схемою крос-коннекта, наприклад шляхом введення відповідного положення в стандарти й інші нормативні документи, неможлива в принципі. Тому, щоб стимулювати замовників до застосування встаткування СІУ в проектах побудови СКС, постачальники повинні шукати ефективні шляхи його адаптації до схеми інтерконнекта.

Уперше галузь серйозно зайнялася зазначеною проблемою приблизно в середині першого десятиліття нового століття. У підсумку було розроблено кілька рішень, що розрізняються між собою базовими принципами, покладеними в їхню основу.

Накладки й сенсорні смужки

Розглянуті далі рішення почали застосовуватися раніше інших. Їхнє конструктивне виконання було запозичено зі створеного раніше встаткування для відкладеного впровадження техніки СІУ в СКС. Наявні доповнення й переробки мінімальні й носять скоріше косметичний характер. Суть застосовуваного прийому полягає в тім, що для побудови функціональних секцій комутаційного поля, які охоплюються дією інтерактивного управління, спочатку використовуються більше дешеві панелі, а незабаром після початку експлуатації, з більшою або меншою відстрочкою, на них установлюються додаткові компоненти з датчиками підключення комутаційних шнурів, і в результаті звичайна СКС перетворюється в інтелектуальну.

На практиці зазначена ідея знайшла втілення у двох варіантах, що розрізняються використанням м'якої й твердої схеми виконання панельної частини датчиків підключення.

Першим варіантом є застосування гнучких смужок, на яких сформовані друковані струмоведучі провідники й контактні площадки (англ. sensor strip). Їхня установка на штатне робоче місце виробляється шляхом наклейки. Основна перевага полягає в простоті реалізації. Слабкі місця видні неозброєним поглядом: вузол кріплення до смужки розетки для підключення шлейфа сканера не відрізняється механічною міцністю, а елементи оптичної індикації відсутні. До того ж контактний вузол датчика не має високу надійність: струмоведучі елементи шнурової й панельної частин чутливих елементів датчика підключення мають точковий контакт.

У другому варіанті немає відзначених вище недоліків сенсорних смужок. Передбачена в ньому тверда накладка або приставка встановлюється на лицьову пластину корпуса панелі й кріпиться механічним способом за допомогою верхньої пари кріпильних гвинтів при звичайній для панелі чотирьохточкової схемі фіксації на 19-дюймових монтажних рейках. При такому підході конструкцію панелі можна не міняти. По цьому принципі був виготовлений перший в історії продукт даного різновиду, відомий під торговельною маркою ReView і серійно випускалася компанією Ri Technologies. Крім того, можна застосовувати панель, попередньо підготовлену для установки на неї лінійки датчиків (вироби серії AMPTRAC Ready компанії TE Connectivity).

Завдяки твердим накладкам значно послабляються обмеження на тип чутливого елемента датчика. Так, поряд з їхніми контактними різновидами серійно випускаються датчики, у яких вузол фіксації підключення вилки до розетки виконаний з використанням перезаписуваних RFID-міток. Ці датчики є одним із ключових апаратних компонентів системи FuturePatch німецької компанії ТКМ.

Вид чутливого елемента впливає на конструктивне виконання приставки. У датчиках контактного типу така приставка виконується у формі накладки, у безконтактних датчиках з RFID-мітками вона може мати вигляд невеликого козирка.

Впровадження рішень розглянутого різновиду утруднено в силу об'єктивних причин. Головна з них полягає в розмаїтості форм лицьових пластин корпусів активного мережного встаткування, причому які-небудь стандарти для них відсутні. У даній ситуації виробникові СІУ доводиться випускати смужки або накладки тільки для обмеженого числа найбільш популярних моделей комутаторів рівня робочої групи. Фінансування цієї роботи здійснюється за рахунок внутрішніх або притягнутих ресурсів виробника СКС.

При виборі дизайну сенсорних накладок розроблювач ураховує дві основних обставини: сформовану конфігурацію ринку активного встаткування Ethernet, орієнтованого на побудову офісних ІТС, і фокусну область застосування техніки СІУ в проектах топ-класу. Таким чином, як найбільше ймовірні пристрої розглядаються комутатори рівня робочої групи, випускаємі серійно компанією Cisco. Виробництво накладок для деяких моделей налагоджено, наприклад, згаданою вище компанією ТКМ.

Завдання створення датчиків підключення для інших менш розповсюджених різновидів мережного встаткування носить чисто технічний характер. При надходженні відповідного замовлення, виготовити більш-менш велику партію виробів можна досить швидко.

Сильною стороною методу накладок або сенсорних смужок є можливість використання таких же комутаційних шнурів, як і при крос-коннекті, що не вимагає розширення номенклатури виробів.

Датчик виносного типу

Головна незручність накладок і сенсорних смужок полягає в тому, що при виготовленні цього компонента доводиться строго дотримувати геометричних розмірів, а установку на лицьовій панелі корпуса комутатора виконувати дуже точно. Подібний недолік можна усунути за рахунок іншого конструктивного виконання панельної частини датчика.

Технічно цілком можливо реалізувати механічне сполучення сенсорної смужки або твердої планки з портами комутатора шляхом включення в

конструкцію елементів юстировки, але це підвищує вартість виробу. Для компенсації механічної непогодженості датчика й корпуса комутатора можна скористатися природною гнучкістю кабелю комутаційного шнура. У такому випадку лінійка датчиків виготовляється у вигляді переднього підтримуючого горизонтального організатора, що встановлюється по центрі лицьової пластини корпуса комутатора із зазором завбільшки кілька сантиметрів. Відповідно, шнурову частину датчика прийде трохи змінити – перенести частина контактних компонентів з вилки на кабель або розмістити на кабелі додатковий контакт. З урахуванням особливості підключення шнура до планки найкращим його виконанням є кільцева форма.

Зрозуміло, при виносному варіанті виконання сам датчик може реалізовувати тільки контактну схему чутливого елемента.

Індивідуальні датчики підключення

Механічна несумісність форм-факторів датчиків підключення шнура до комутатора й тої частини його корпуса, на яку виводяться розетки роз'ємів, досить ефективно усувається переходом на окрему для кожного порту схему реалізації чутливого елемента цього компонента СІУ. Оскільки датчики повинні бути механічно зафіксовані в робочому положенні, найбільш кращої є конструкція їхньої панельної частини у вигляді вставки, на якій монтуються пасивні й електронні компоненти чутливого елемента. Ідентифікація конкретного порту активного мережного пристрою здійснюється за рахунок прошивання унікального ідентифікаційного номера в електронній частині вставки.

Для мережних інтерфейсів, розрахованих на підключення до симетричних кабелів із кручених пар, краще виконання вставки у формі внутрішнього компонента, що у робочому положенні повністю схований у гнізді розетки. Для зменшення товщини стінок його корпус виготовляється з металу, має П-образну форму й утримується в робочому положенні звичайною засувкою важільного типу. Щоб уникнути випадкового натискання на важіль, його кінець не повинен виступати за габарити корпуса розетки.

Сама вилка шнура механічно фіксується на попередньо зібраному агрегаті розетка-вставка за допомогою підпружиненого Г-образного важеля-зачепа, закріпленого в технологічному виступі П-образної металевої основи. При цьому виступ виходить із розетки приблизно настільки ж, що й вільний кінець важеля традиційної засувки, а важіль технологічної засувки вставки коротшає таким чином, щоб він не перекривав вікно виступу.

Вимога мінімальної товщини корпуса виникла не випадково. Наявність вставки в гнізді модульного рознімання припускає обов'язкове зменшення габаритів вилки, але можливостей для цього мало.

Перераховані особливості схеми реалізації датчика обумовлюють несиметричне виконання комутаційного шнура. Інакше кажучи, на різні кінці його кабелю монтуються вилки з різним форм-фактором.

У результаті шнур можна підключати до панелі й комутатора тільки в строго певнім положенні.

Для оптичних портів цілком достатньо вставки-адаптера FM-типу, у робочому положенні частково виступаючої із гнізда розетки. Єдиною додатковою вимогою до цього компонента є мінімальні втрати, внесені в тракт передачі сигналу.

Вибір такого дизайну дає можливість не переробляти корпус вилки й розетки оптичного рознімання. Деяка експлуатаційна незручність через наявність виступаючої назовні частини корпуса не має великого значення через відносно невеликі обсяги застосування волоконно-оптичної техніки в проектах побудови СКС.

Перевагою індивідуальних датчиків у формі вставок є можливість прямого й гранично простого включення в область дії встаткування інтерактивного управління активних мережних пристроїв з невеликою кількістю портів – аж до одиночних. До даної групи виробів ставляться в першу чергу сервери, рідше зустрічаються різні перетворювачі середовища, маршрутизатори й аналогічне їм устаткування.

Ще одна перевага – відсутність необхідності в прямому механічному зв'язку панельної частини датчика з корпусом активного мережного пристрою. Це спрощує роботу з обслуговування змінних портів на основі модулів SFP і їм подібних.

3.2 Розробка структурної схеми

Процес управління конфігурацією СКС є самодостатнім у рамках першого фізичного рівня моделі відкритих систем. Однак до нього можна підключити й інші, більше високі рівні цієї моделі – ніяких перешкод для цього, крім, може бути, чисто технічних, не існує.

Основна перевага такого підходу полягає в тім, що розроблювачеві СІУ не потрібно забезпечувати винос датчиків підключення на порт комутатора й вирішувати всі супутні проблеми. Замість цього досить змінити або розширити алгоритм обробки сигналів, запитуваних у керованого активного мережного встаткування. У багатьох випадках це набагато простіше виконати із застосуванням протоколу SNMP.

Ідея чисто програмної схеми заснована на тому факті, що між портами активного встаткування, що використовують для зв'язку кабельні тракти СКС, існує єдиний шлях передачі сигналу. Для поширення цієї властивості на загальний випадок і усунення можливих невизначеностей висуваються наступні вимоги:

– попередня однозначна прив'язка портів активного мережного встаткування на одному з кінців стаціонарних ліній СКС із занесенням у БД системи адміністрування;

– розміщення змінюваної й контрольованої СІУ частини трактів передачі в загальній зоні комутації (англ. connectivity zone).

Рисунок 3.1 – Структурна схема системи

У цьому випадку для однозначного відновлення тракту серверу СІУ досить виконати просту двошагова дія за допомогою стандартних процедур систем управління мережею:

– На першому етапі за допомогою протоколу SNMP опитуються конфігурація комутатора й стан його портів і визначаються IP-адреси підключених до них мережних пристроїв.

– Потім зіставляються пари IP-адреса – порт панелі.

Опитування стану портів комутатора з метою одержання вихідної інформації, необхідної для встановлення зв'язку між активними мережними пристроями, виконується різними способами. Сукупність застосовуваних підходів можна розділити на дві основні групи.

Перший з можливих варіантів є класичним централізованим. У його основу покладене опитування комутаторів і іншого активного встаткування безпосередньо із сервера БД системи адміністрування. Застосування для рішення цього завдання популярної сьогодні хмарної моделі реалізації обчислювального процесу не міняє картини в цілому.

Друга схема, яку можна назвати розподіленою, або багаторівневою, заснована на звертанні до системи агентів, які запускаються на довільних серверах ІТС і збирають первинну інформацію про з'єднання. На центральний сервер відправляються тільки результати обробки цих даних. Головними перевагами розподіленого підходу є його гнучкість і більше висока продуктивність. Останнє пояснюється тим, що сервер не повинен безупинно взаємодіяти з агентами, а приступає до виконання відповідних процедур тільки після одержання запиту.

Програмно-апаратний комплекс

Звичайно контролер СІУ відіграє провідну роль у процесі опитування датчиків підключення шнурів. Він генерує сигнали, що активізують панельний елемент датчика, у тому числі той з них, що відслідковує підключення шнура до порту комутатора, і одержує інформацію про його стан. Дану схему можна модифікувати із залученням даних, що надходять із системи управління комутатором. Для цього повинне бути організоване взаємодія двох програмних продуктів, що працюють на різних рівнях ІТС. Як допоміжний засіб використовується протокол SNMP.

Перехід на іншу схему опитування стану портів, що комутируються, і видачі керуючих команд зажадало істотної переробки апаратної частини СІУ. Модернізація зводиться, в основному, до наступного:

– введення додаткового, 25-го, спеціалізованого порту, що розташовується на панелі в центральній частині її лицьової пластини;

– повний винос елементів датчика, що відслідковує підключення до порту комутатора, винятково на вилку відповідного кінця комутаційного шнура.

Окремо відзначимо, що й у цьому випадку шнурова частина датчика вилки, що підключається до комутатора, містить активні електронні компоненти.

Реалізується наступний алгоритм роботи. Після активізації робочого завдання системний адміністратор вставляє комутаційний шнур у порт комутатора. Факт підключення реєструється датчиком на підставі механічного переміщення стрижневого штовхача. Після того як качана другого кінця підключається до 25-му службового порту панелі, контролер СІУ починає взаємодіяти із системою управління комутатора за допомогою запущеного на сервері ПЗ адміністрування, зчитує інформацію про номер порту й зіставляє її з робочим завданням. За результатами порівняння визначається номер порту панелі, до якого повинен бути підключений відповідний порт комутатора, і цей порт панелі відзначається включенням світлодіодного індикатора. «Панельний» кінець шнура виймається з 25-го порту й негайно ж з'єднується з тим портом панелі, на який указує запалений СД. Процедура повторюється до повного виконання робочого завдання.

Особливість реалізації датчика підключення й обраний розроблювачем алгоритм обробки його сигналу впливають на конструктивне виконання апаратної частини СІУ. Крім додаткового спеціалізованого керуючого порту панелі, найбільш істотна зовнішня відмінність системи полягає в «інтерконнектному» комутаційному шнурі. Останній має несиметричну конструкцію зі спеціалізованими качанами роз’ємів, що досить типово для СІУ з підтримкою інтерконнекта. Такий шнур повинен підключатися до комутатора й панелі в строго певнім положенні.

Досить добірно вирішується проблема відкладеного впровадження СІУ у вже побудовану раніше структуровану проводку. Для цього 25-й спеціалізований

порт у вигляді модульної вставки встановлюється в спеціально призначене для неї гніздо. Відповідні елементи кріплення для установки інших апаратурних компонентів СІУ передбачені й на задній стороні панелі.

Завдання підтримки інтерконнекта стосовно до СІУ може бути вирішена різними способами, що помітно розрізняються конструктивними особливостями й – до певного ступеня – функціональними можливостями.

Всі провідні виробники СКС уводять у свої пропозиції серійну елементну базу й відповідним чином модифіковане ПЗ для включення в область дії СІУ комутаційного поля за схемою інтерконнекта.

Останні розробки в даній області припускають відмова від механічного переносу датчиків підключення, використовуваних у комутаційних панелях, на лицьову пластину корпуса комутаторів на користь оригінальних конструкцій інших різновидів.

Підтримка схеми інтерконнекта найчастіше припускає використання комутаційних шнурів з несиметричною конструкцією, які до панелі можна підключати тільки однієї із двох вилок.

Введення в СІУ підтримки схеми інтерконнекта стимулює розробки по інтеграції активних електронних компонентів у шнурову частину датчиків, що, у свою чергу, у ряді випадків вимагає використання 10-провідних комутаційних шнурів.

Завдання контролю інтерконнекта вирішуються більш ефективно, якщо задіються ресурси систем управління фізичним рівнем інформаційної інфраструктури й комутатора.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно. Програма надає великі сервісні можливості операторові, виводячи різноманітну інформацію на екран. Наприклад, на дисплеї

комп'ютера можна мати план одного або декількох приміщень із позначеними на ньому контрольованими точками, індикацію. На екран можуть виводитися численні повідомлення, наприклад, повні або короткі звіти про зареєстровані події з можливістю їхньої роздруківки на принтері.

Рисунок 3.2 – Функціональна схема системи

Програмне забезпечення повинне забезпечує:

– ініціалізацію ідентифікаторів (занесення кодів ідентифікаторів до пам’яті системи);

– завдання характеристик контрольованих точок;

– установку часових інтервалів доступу (вікон часу);

– установку рівнів доступу для користувачів;

– протоколювання поточних подій;

– ведення баз даних;

– збереження даних і установок при аваріях і збоях у системі.

Загальний принцип роботи будь-який RFID системи досить простий. У системі завжди є два основних компоненти: це зчитувач і ідентифікатор (карта, мітка, брелок). Зчитувач випромінює в навколишній простір електромагнітну енергію. Ідентифікатор приймає сигнал від зчитувача й формує відповідний сигнал, що приймається антеною зчитувача й обробляється його електронним блоком.

Рівень доступу – сукупність часових інтервалів доступу (вікон часу) і місць проходу (маршрутів переміщення), які призначаються певній особі або групі осіб, яким дозволений доступ у задані охоронювані зони в задані часові інтервали).

Програмне забезпечення повинне бути стійко до випадкових і навмисних впливів наступного виду:

– відключення керуючого комп'ютера;

– програмне скидання керуючого комп'ютера;

– апаратне скидання керуючого комп'ютера;

– натискання на клавіатурі випадковим образом клавіш;

– випадковий перебір пунктів меню програми.

Після зазначених впливів і після перезапуску програми повинна зберігатися працездатність системи й схоронність установлених даних. Зазначені впливи не повинні приводити до відкривання пристроїв загородження й зміні діючих кодів доступу.

Програмне забезпечення повинне бути захищене від навмисних впливів з метою зміни установок у системі.

Вид і ступінь захисту повинні бути встановлені в паспортах на конкретні види засобів або систем. Відомості наведені в технічній документації не повинні розкривати таємність захисту.

Програмне забезпечення при необхідності повинне бути захищене від несанкціонованого копіювання.

Програмне забезпечення повинне бути захищене від несанкціонованого доступу за допомогою паролів. Кількість рівнів доступу по паролях повинне бути не менш 3.

Рівні доступу, що рекомендуються, по типу користувачів:

– перший ("адміністрація") – доступ до всіх функцій контролю й доступу;

– другий ("оператор") – доступ тільки до функцій поточного контролю;

– третій ("системний програміст") – доступ до функцій конфігурації програмного забезпечення, без доступу до функцій, що забезпечують управління виконавчих пристроїв.

При уведенні пароля на екрані дисплея не повинні відображатися вводяться знаки, що.

Число символів пароля повинне бути не менш 5.

Вимоги до електроживлення

Основне електроживлення СІУ для СКС повинне здійснюватися від мережі змінного струму частотою 50 Гц із номінальною напругою 220 В.

СІУ для СКС повинні зберігати працездатність при відхиленнях напруги мережі від мінус 15 до +10 % і частоти до ±1 Гц від номінального значення.

Електроживлення окремих СІУ для СКС допускається здійснювати від інших джерел з іншими параметрами вихідних напруг вимоги до яких установлюються в нормативних документах на конкретні типи систем.

Електропостачання технічних засобів СІУ для СКС здійснюється від вільної групи щита чергового висвітлення. При відсутності на об'єкті щита чергового висвітлення або вільної групи на ньому, замовник установлює самостійний щит електроживлення на відповідну кількість груп. Щит електроживлення, установлюваний поза охоронюваним приміщенням, повинен розміщатися в металевій шафі, що замикається, і заблокований на відкривання.

СІУ для СКС повинні мати резервне електроживлення при проваллі основного електроживлення. Номінальна напруга резервного джерела живлення повинне бути 12 або 24 В.

Перехід на резервне живлення й назад повинен відбуватися автоматично без порушення встановлених режимів роботи й функціонального стану СІУ для СКС.

СІУ для СКС повинні зберігати працездатність при відхиленнях напруги резервного джерела живлення від мінус 15 до плюс 10 % від номінального значення.

Резервне джерело живлення повинен забезпечити функціонування системи при проваллі напруг у мережі на час не менш 8 ч.

При використанні як джерело резервного живлення акумулятора, повинен виконуватися автоматичну зарядку акумулятора.

Акумуляторні батареї (за винятком що не обслуговуються), як правило, розміщаються в спеціальних акумуляторних приміщеннях на стелажах або полках шафи, відповідно до вимог ТУ 4-ДО.610.236-87 у піддонах, стійких до впливу агресивних середовищ.

Свинцеві акумулятори ємністю не більше 72 А/год і лужні акумуляторні батареї ємністю не більше 100 А/год і напругою до 60 В можуть установлюватися в загальних виробничих невибухо- і непожежнонебезпечних приміщеннях у металевих шафах з відособленої приточно-витяжною вентиляцією.

Акумуляторні установки повинні бути обладнані відповідно до вимог ППЕ "Правила пристрою електроустановок".

При використанні як джерела резервного живлення, акумулятора або сухих батарей, повинна бути передбачена індикація розряду акумулятора або батареї нижче припустимої межі.

Для автономних систем індикація розряду повинна бути світлова або звукова, для мережних систем сигнал розряду акумулятора повинен передаватися на центральний пульт.

Хімічні джерела струму (батарейки), убудовані в активні ідентифікатори або забезпечуючи схоронність даних повинні забезпечувати працездатність засобів контролю й управління доступом протягом часу, не менш 5 років.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі.

Рисунок 3.3 – Діаграма взаємодії процесів

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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 ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю інтерактивного керування структурованою кабельною системою мережі.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

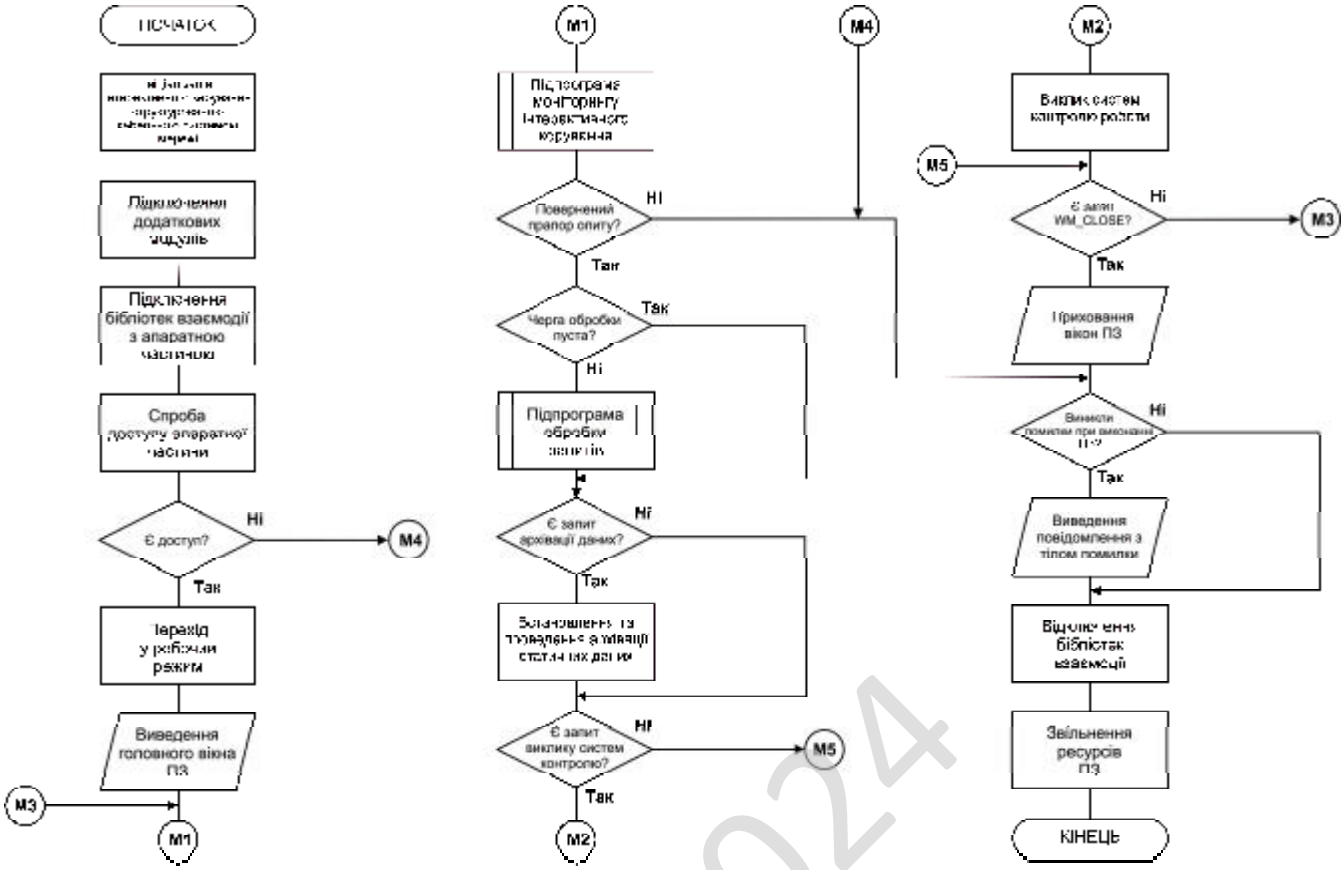

Рисунок 4.1 – Блок-схема основної програми

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML.

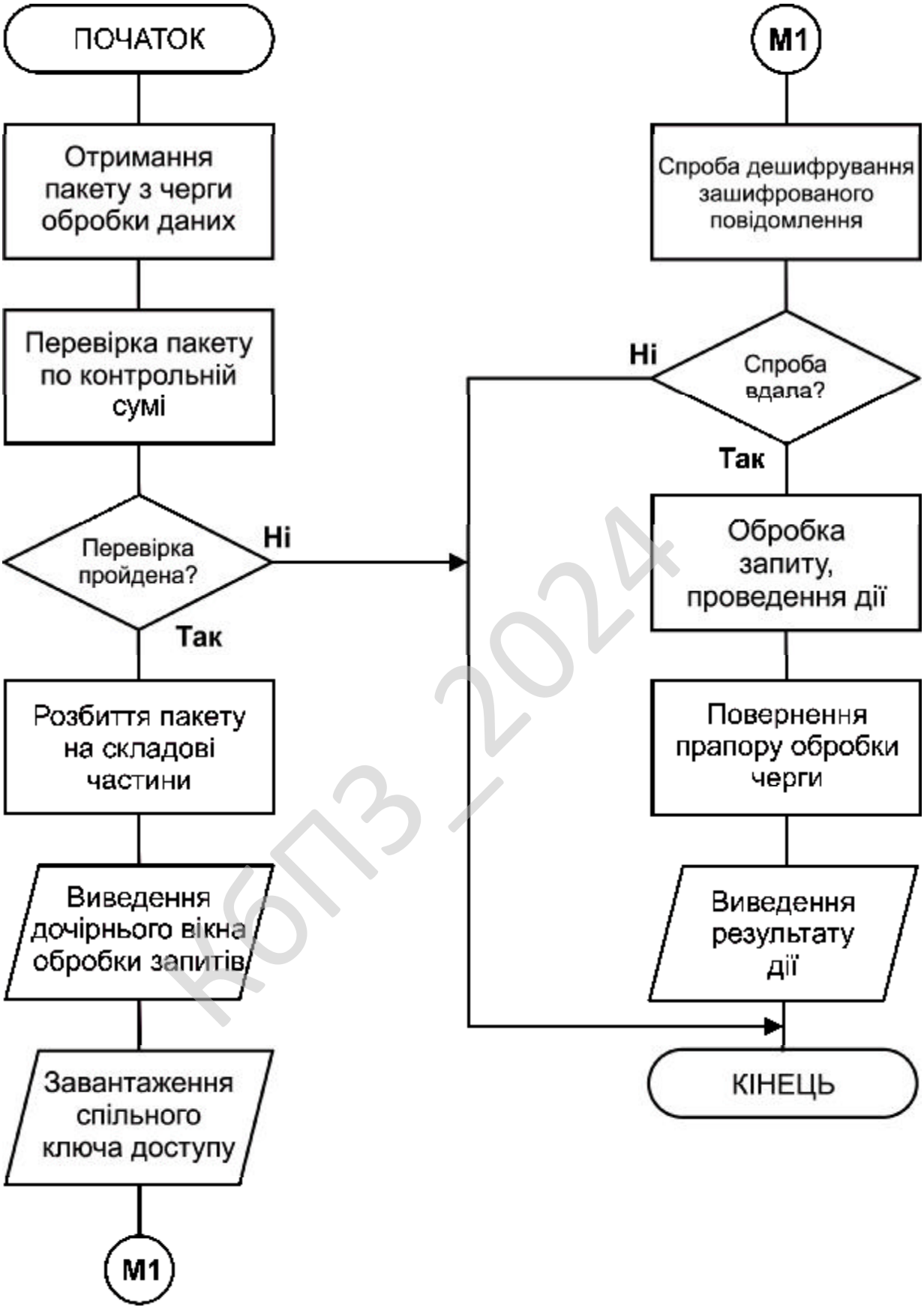

Рисунок 4.2 – Блок-схема роботи підпрограми

Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма компонент; Діаграма об'єктів; Діаграма розгортання.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням

тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Діаграма об'єктів в UML це діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.

Діаграма об'єктів відображає об'єкти та зв'язки в певний момент роботи програми. Об'єкти можуть містити інформацію про власні значення а не про описання. Для відображення загальних шаблонів об'єктів та зв'язків, що можуть багаторазово створюватись під час роботи програми, слід використовувати діаграму взаємодії, яка може відображати характеристики об'єктів та зв'язків. Екземпляр діаграми взаємодії створює діаграму об'єктів.

Діаграма об'єктів не відображає еволюцію системи під час роботи. Натомість, слід використовувати діаграми взаємодії з повідомленнями, або діаграми послідовності.

Діаграма розгортання (deployment diagram) це діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

Розглянемо код що дозволяє одержати покажчик на останній байт мітки.

```
Function CheckMemLabel(ptrStart,ptrEnd: pointer; strlabel: string;
                       var point: pointer): boolean;
 var
  ptrCurrent: pointer;
  currarray: array [1..8] of byte;
  lookarray: array [1..8] of byte;
  i: integer;
begin
 Result := false;
 point := ptrStart;
//Завантаження мітки
 for i := 1 to 8 do
 begin
  lookarray[i] := HexToByte(strlabel[pos('H',UpperCase(strlabel))-2]+
                  strlabel[pos('H',UpperCase(strlabel))-1]);
  currarray[i] := 0;
  Delete(strlabel,1,pos('H',UpperCase(strlabel)));
 end;
 //Пошук мітки
 ptrCurrent := ptrStart;
 while ptrCurrent <> ptrEnd do
 begin
  for i := 1 to 7 do
   currarray[i] := currarray[i + 1];
  currarray[8] := Byte(ptrCurrent^);
  Result := true;
  for i := 1 to 8 do
   Result := Result and (currarray[i] = lookarray[i]);
  if Result then
  begin
   //крапка, останнього байта в мітці
   point := ptrCurrent;
   break;
  end;
  Inc(Integer(ptrCurrent));
 end;
end;
```

У результаті виконання цієї функції point указує на останній байт мітки. Strlabel має той же зміст, що й при пошуку у файлі. Участки коду ptrStart і ptrEnd

це адреси з якого й по якого проводиться пошук влучні, для їхнього визначення використовувався наступний код:

```
CheckMemLabel(@StartPointProc,@EndPointProc,strlabel,
              point);
CheckMemLabel(@StartPointProc,Pointer(Integer(@StartPointProc)+1000),
              strlabel,point);
```

Розглянемо як проходе обчислення контрольних сум критичних ділянок коду. Для перевірки розробленого ПЗ і реалізації алгоритму від несанкціонованого використання в розробленій програмі використовувалися спеціальні процедури. Розглянемо їхній код, і дії проведені для їхнього створення.

Розглянемо дві процедури перевірки ПЗ на відповідність:

```
procedure Check_TrialMyDip;
begin
 ShowMessage('Незареєстроване використання ПЗ!.');
end;
procedure EndCheckTrial;
// Порожня процедура, що означає кінець CheckTrial – мітка
begin
end;
```

У тексті програми процедура EndCheckTrial обов'язково повинна випливати відразу після процедури CheckTrial. Вона потрібна, щоб знати адреса кінця коду процедури CheckTrial.

4.2 Захист розробленого програмного забезпечення

Tiny Encryption Algorithm (TEA) [1] – блочний алгоритм шифрування типу «Мережі Фейстеля». Алгоритм був розроблений на факультеті комп'ютерних наук Кембриджського університету Девідом Вілером (David Wheeler) і Роджером Нідгемом (Roger Needham) та вперше представлений в 1994 році [2] на симпозіумі зі швидкими алгоритмами шифрування в Льовені (Бельгія).

Шифр не патентований, широко використовується в ряді криптографічних додатків і широкому спектрі апаратного забезпечення, завдяки вкрай низькими вимогами до пам'яті й простоті реалізації. Алгоритм має як програмну реалізацію

на різних мовах програмування, так і апаратну реалізацію на інтегральних схемах типу FPGA.

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму TEA, який заснований на бітових операцій з 64-бітним блоком, має 128-бітний ключ шифрування. Стандартна кількість раундів мережі Фейстеля біля 64 (32 циклу), однак, для досягнення найкращої продуктивності або шифрування, число циклів можна варіювати від 8 (16 раундів) до 64 (128 раундів). Мережа Фейстеля несиметрична через використання в якості операції накладення додавання за модулем 2^{32}.

Перевагами шифру є його простота в реалізації, невеликий розмір коду й досить висока швидкість виконання, а також можливість оптимізації виконання на стандартних 32-бітних процесорах, так як в якості основних операцій використовуються операції виключна «АБО» (XOR), побітового зсуву й додавання за модулем 2^{32}. Оскільки алгоритм не використовує таблиць підстановки і раундова функція досить проста, алгоритму потрібно не менше 16 циклів (32 раундів) для досягнення ефективної дифузії, хоча повна дифузія досягається вже через 6 циклів (12 раундів).

Алгоритм має відмінну стійкість до лінійного криптоаналізу і досить гарну до диференціального криптоаналізу. Головним недоліком цього алгоритму шифрування є його вразливість до атак «на пов'язаних ключах» (англ. Related-key attack). Через простий розклад ключів кожен ключ має 3 еквівалентних ключа. Це означає, що ефективна довжина ключа складає всього 126 біт [3] [4], тому даний алгоритм не слід використовувати в якості геш-функції.

Опис алгоритму

Вихідний текст розбивається на блоки по 64 біта кожен. 128-бітний ключ K ділиться на чотири 32-бітних підключа K [0], K [1], K [2] і K [3]. На цьому підготовчий процес закінчується, після чого кожен 64-бітний блок шифрується протягом 32 циклів (64 раундів) за нижченаведеним алгоритмом. [5]

Припустимо, що на вхід n-го раунду ($1 \leq n \leq 64$) надходять права й ліва частини (L_n, R_n), тоді на виході n-го раунду будуть ліва й права частини (L_{n+1}, R_{n+1}), які обчислюються за такими правилами:

$$L_{n+1} = R_n.$$

Якщо $n = 2 * i - 1$ для $1 \leq i \leq 32$ (непарні раунди), то:

$$R_{n+1} = L_n (\{ [R_n 4] K [0] \} \{ R_n i * \delta \} \{ [R_n 5] K [1] \}).$$

Якщо $n = 2 * i$ для $1 \leq i \leq 32$ (парні раунди), то:

$$R_{n+1} = L_n (\{ [R_n 4] K [2] \} \{ R_n i * \delta \} \{ [R_n 5] K [3] \}).$$

Де

X Y – операція додавання чисел X і Y за модулем 232.

X Y – побітове виключне АБО» (XOR) чисел X і Y, яке в мові програмування Сі позначається як X ^ Y

X Y і X Y – операції побітового зсуву числа X на Y біт вліво й вправо відповідно.

Константа δ була виведена з Золотого перерізу:

$$\delta = (-1) * 2^{31} = 2654435769 = 9E3779B9_h.$$

У кожному раунді константа множиться на номер циклу i. Це було зроблено для запобігання простих атак, заснованих на симетрії раундів.

Також очевидно, що в алгоритмі шифрування TEA немає як такого алгоритму розкладу ключів. Замість цього в непарних раундах використовуються підключі K [0] та [1], у парних – K [2] і [3].

Так як це блочний шифроалгоритм, де довжина блоку 64-біт, а довжина даних може бути не кратна 64-біт, значення всіх байтів, які доповнюють блок до кратності в 64-біт, встановлюється в 0x01.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи.

Розроблене програмне забезпечення інтерактивного керування структурованою кабельною системою мережі складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Дані; Кабельна система; Налаштування; Довідка.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

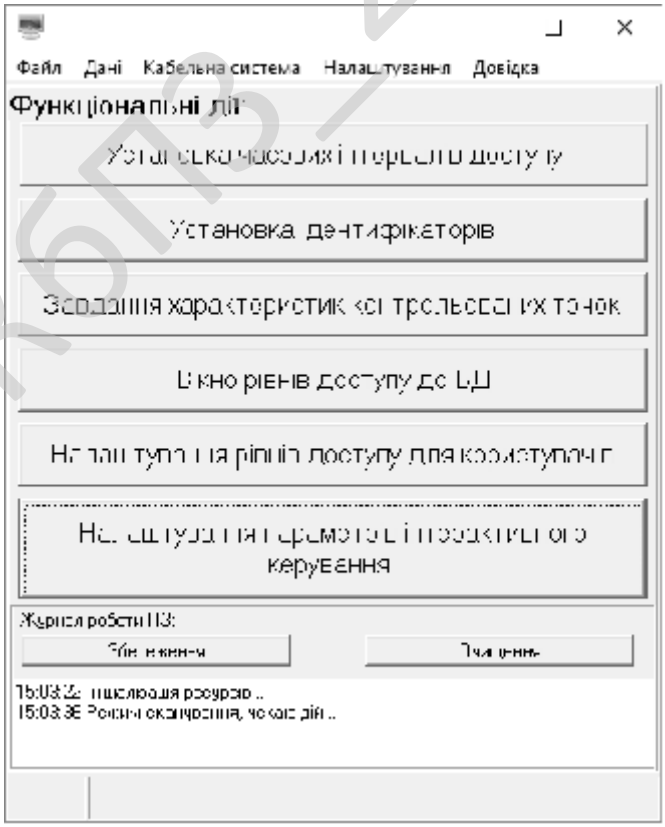

Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

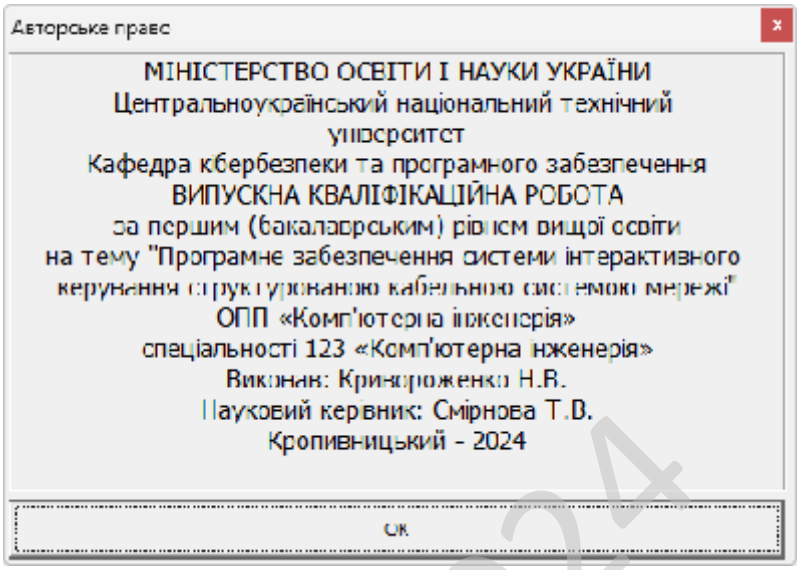

Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – proprietary software.

Програмне забезпечення, на яке зберігаються як немайнові, так і майнові авторські права. Отримавши або придбавши таке програмне забезпечення, користувач отримує обмежені права користування ним: може бути заборонено або закрито доступ до коду (вивчення), внесення змін, тиражування, розповсюдження та перепродаж. Програмне забезпечення вважається власницьким, якщо наявне хоча б одне з перелічених обмежень.

Найчастіше основним методом захисту майнових прав на власницьке ПЗ, поза ліцензійною угодою, власник обирає закриття сирцевого коду, захищаючи свій продукт від модифікації і вбудовуючи системи обмеження користування через авторизацію. Таке програмне забезпечення називається закритим. Проте, код власницького продукту може бути і відкритим, але власник може обмежити права користувача умовами користувацької ліцензії.

Власницьке програмне забезпечення та комерційне програмне забезпечення не є синонімами – власницьким може бути і безплатне (тобто, некомерційне) програмне забезпечення.

На противагу власницькому ПЗ існує вільне програмне забезпечення, автори і власники якого дозволяють вивчати, модифікувати і поширювати свій продукт. Саме визначення власницького програмного забезпечення виникло в результаті діяльності громадського руху вільного програмного забезпечення (представленого Фондом вільного програмного забезпечення та іншими організаціями) і осмислення умов свободи користування програмами.

Визначенням власницького програмного забезпечення є не невідповідність хоча б одній з базових умов вільного програмного забезпечення. Сама назва власницьке ПЗ підкреслює визначальне значення власника у способі використання і можливостях розвитку цього програмного забезпечення.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи інтерактивного керування структурованою кабельною системою мережі.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем інтерактивного керування структурованою кабельною системою мережі.

– Досліджена система інтерактивного керування структурованою кабельною системою мережі.

– На основі отриманих результатів досліджень створена програмна реалізація системи інтерактивного керування структурованою кабельною системою мережі.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання інтерактивного керування структурованою кабельною системою мережі.

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 Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи інтерактивного керування структурованою кабельною системою мережі.

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 TEA.

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. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.

2. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

3. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

4. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

5. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

6. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

7. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

8. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

9. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

10. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

11. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

12. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

13. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

14. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

15. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

16. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

17. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

18. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

19. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

20. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

21. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

22. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

23. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

24. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

25. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

26. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

27. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

28. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

29. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

30. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

31. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

32. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

33. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

34. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

35. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

36. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

37. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних

систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

38. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

39. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

40. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

41. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

42. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

43. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

44. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

45. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". - Випуск 5 (142). - Х.: ХУПС - 2016. - С. 148-152.

46. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 36-39.

47. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. - 2016. - С. 121-127.

48. Смірнов О.А., Смірнов С.А., Дідик А.К. Метод безпечної маршрутизації метаданих у хмарні антивірусні системи. Системи озброєння та військова техніка. - Випуск 2 (46) - Х.: ХУПС - 2016. - С. 146-149.

49. Смірнов О.А., Кавун С.В., Доренський О.П., Вялкова В.І. Інформаційна безпека в комп’ютерних мережах. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 151 с.

50. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

51. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп’ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи інтерактивного керування структурованою кабельною системою мережі.

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. № 139-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи інтерактивного керування структурованою кабельною системою мережі.

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 інтерактивного керування структурованою кабельною системою мережі;

– 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, 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 Delphi 10.4.

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 – 64 аркуші.

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.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 11.06.2024 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

_____ Смірнова Т.В.

Програмне забезпечення системи інтерактивного керування структурованою кабельною системою мережі

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 52

Літера: РП

Кропивницький – 2024 року

Авторське право

```
unit AB; // Модуль авторського права
// Кривороженко Наталія Володимирівна
// гр. КІ-20ПЗ
// Кропивницький 2024

Interface //інтерфейс модулю

uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
 Buttons, ExtCtrls; // Бібліотеки що використовуються

Type // Власті типи даних
  TAboutBox = class(TForm)
    Panel1: TPanel;
    ProgramIcon: TImage;
    ProductName: TLabel;
    Version: TLabel;
    Copyright: TLabel;
    Comments: TLabel;
    OKButton: TButton;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure OKButtonClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AboutBox: TAboutBox;

Implementation // Реалізація інтерфейсу модулю

uses Unit1; // Бібліотеки що використовуються

{$R *.dfm}

procedure TAboutBox.FormClose(Sender: TObject; var Action: TCloseAction);
begin
Action:=caNone;
end;

procedure TAboutBox.OKButtonClick(Sender: TObject);
begin
AboutBox.hide;
Form1.show;
end;

end.
```

Головний файл проекту

```
program D_PROJECT;

// Кривороженко Наталія Володимирівна
// гр. КІ-20ПЗ
// Кропивницький 2024

{%File 'PR.inc'}

Uses // Бібліотеки що використовуються
  Forms,
  Main in 'M1.pas' { Form1},
  AB in 'AB.pas' { Form2},
  Splash in 'Splash.pas' {wnd_Splash};

{$R *.res}

procedure StartPointProc;
begin
 asm
  DB 000h, 034h, 0F4h, 02Ah, 000h, 03Dh, 0FFh, 0CAh
 end;
end

procedure CheckTrial;
begin
 asm
  DB 09Dh, 03Dh, 04Ah, 061h, 025h, 0CDh, 0C7h, 0DFh
  DB 0DAh, 0DAh, 0E6h, 025h, 0DAh, 0DAh, 0DAh, 0DAh
  DB 03Ch, 025h, 025h, 025h, 071h, 057h, 04Ch, 044h
  DB 049h, 005h, 055h, 040h, 057h, 04Ch, 04Ah, 041h
  DB 005h, 04Dh, 044h, 056h, 005h, 040h, 05Dh, 055h
  DB 04Ch, 057h, 040h, 041h, 00Bh, 025h, 025h, 025h
 end;
end;
begin
  Application.Initialize; // Ініціалізація
  Application.Title := 'D_RFID';
      wnd_Splash:=Twnd_Splash.Create(Application);
      wnd_Splash.Show;//Показання
      wnd_Splash.Update;//Обновлення

  Application.CreateForm(Twnd_msi_Main, wnd_msi_Main);
  Application.CreateForm(Twnd_msi_Main, wnd_msi_Main);
  Application.CreateForm(Twnd_msi_Main, wnd_msi_Main);


  Application.Run; // початок роботи потоку
  wnd_Splash.Free;
end.
```

Головне вікно програми Unit1

```
unit Unit1; // Назва модулю

// Криворoженко Наталія Володимирівна
// гр. КІ-20ПЗ
// Кропивницький 2024

interface //інтерфейс модулю

uses // Бібліотеки що використовуються
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, ShellApi, Registry, ExtCtrls, unit3;

  Type // Власті типи даних
TForm1 = class(TForm)
Edit1: TEdit;
Label1: TLabel; Label2: TLabel; Label3: TLabel; Label4: TLabel;
Timer1: TTimer; Timer2: TTimer;
Panel1: TPanel;
gb1: TGroupBox; gb2: TGroupBox;
en_code: TSpeedButton; savet: TSpeedButton;
clear2: TSpeedButton; clear1: TSpeedButton;
loadt: TSpeedButton; GroupBox1: TGroupBox;
rb1: TRadioButton; rb2: TRadioButton;
GroupBox2: TGroupBox;
i1: TSpeedButton;
opend: TOpenDialog; saved: TSaveDialog;
ApplicationEvents1: TApplicationEvents;
RxTrayIcon1: TRxTrayIcon;
PopupMenu1: TPopupMenu;
N1: TMenuItem; N2: TMenuItem; N3: TMenuItem;
procedure rb1Click(Sender: TObject);
procedure rb2Click(Sender: TObject);
procedure en_codeClick(Sender: TObject);
procedure clear1Click(Sender: TObject);
procedure clear2Click(Sender: TObject);
procedure i1Click(Sender: TObject);
procedure loadtClick(Sender: TObject);
procedure savetClick(Sender: TObject);
procedure ApplicationEvents1ShowHint(var HintStr: String;
var CanShow: Boolean; var HintInfo: THintInfo);
procedure N3Click(Sender: TObject);
procedure RxTrayIcon1DblClick(Sender: TObject);
procedure N2Click(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure FormCreate(Sender: TObject);
procedure Label2Click(Sender: TObject);
procedure Label2MouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
procedure FormMouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
procedure Label3Click(Sender: TObject);
procedure Label3MouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
procedure Timer1Timer(Sender: TObject);
procedure Timer2Timer(Sender: TObject);
procedure FormKeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
procedure Panel1MouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
procedure FormResize(Sender: TObject);
private
      { Private declarations }
    public
      { Public declarations }
    end;
  var
    hMenuHandle:HMENU;
```

```
    hTaskBar : THandle;
    Form1: TForm1;
    Mask: Boolean;
    Guest: Boolean;
    ShowButton: Boolean;
    AccessEnabled: Boolean;
    Pass: String;
    GPass: String;
    F2:textfile;
    Reg: Tregistry;

Implementation // Реалізація інтерфейсу модулю

  {$R *.DFM} // файл ресурсу

procedure TForm1.FormClose(Sender:TObject; var Action:TCloseAction);
  begin
    chdir(extractfilepath(application.exename));
    AssignFile(F2,'users.log');
    append(F2);
    Writeln(F2,timetostr(time) + ':'+edit1.text );
    CloseFile(F2);
    if showbutton and AccessEnabled then
    begin
      AssignFile(F2,'users.log');
      append(F2);
      CloseFile(F2);
      Action:=caNone;
      timer2.Interval:= 0;
      frmshield.show;
    end;
    if pass = edit1.Text then
    begin
      AssignFile(F2,'users.log');
      append(F2);
      CloseFile(F2);
      if not (csDesigning in ComponentState) then
      RegisterServiceProcess(GetCurrentProcessID, 1);
      timer2.Interval := 0;
      SystemParametersInfo(SPI_SCREENSAVERRUNNING, 0, nil, 0);
      hTaskbar := FindWindow('Shell_TrayWnd', Nil);
      ShowWindow(hTaskBar, SW_SHOWNORMAL);
      frmshield.show;
    end;
    if (GPass = Edit1.Text) and guest then
    begin
      AssignFile(F2,'users.log');
      append(F2);
      CloseFile(F2);
      Action:=caNone;
      timer2.Interval := 0;
      frmshield.show;
    end;
    if (pass<>edit1.Text) and ((gpass<>edit1.Text)or not guest) then
    begin
      AssignFile(F2,'users.log');
      append(F2);
      CloseFile(F2);
      label4.Visible := true;
      timer1.Interval := 2000;
      Action:=caNone;
    end;
  end;

  procedure TForm1.FormCreate(Sender: TObject);
  begin
    form1.Top := 0;
    form1.Left := 0;
    form1.Width := screen.Width;
```

```
  form1.Height := screen.Height;
  hTaskbar := FindWindow('Shell_TrayWnd', Nil);
  ShowWindow(hTaskBar, SW_HIDE);
  if not (csDesigning in ComponentState) then
  RegisterServiceProcess(GetCurrentProcessID, 0);
  hMenuHandle := GetSystemMenu(Handle, FALSE);
  IF (hMenuHandle <> 0) THEN
  DeleteMenu(hMenuHandle, SC_CLOSE, MF_BYCOMMAND);
  SystemParametersInfo(SPI_SCREENSAVERRUNNING, 1, nil, 0);
  chdir(extractfilepath(application.exename));
  AssignFile(F2,'users.log');
  append(F2);
  Writeln(F2,'');
  CloseFile(F2);
  Reg := TRegistry.Create;
  Reg.RootKey := HKEY_CURRENT_USER;
  if not Reg.OpenKey('\Software\',True) then
  try
    Reg.RootKey := HKEY_CURRENT_USER;
    if Reg.OpenKey('\Software\',True)
    then
    begin
      Guest := reg.ReadBool ('Guest');
      Mask := reg.ReadBool ('Mask');
      ShowButton := reg.ReadBool ('ShowButton');
      Pass := Reg.ReadString('Pass');
      GPass := Reg.ReadString('GPass');
    end
  finally
    Reg.CloseKey;
    Reg.Free;
    inherited;
  end;
  edit1.PasswordChar := #0;
  if mask then edit1.PasswordChar := '*';
  if showbutton then label3.Visible := true;
end;

procedure TForm1.Label2Click(Sender: TObject);
begin
  close;
end;

procedure TForm1.Label2MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  label2.Font.Color := clYellow;
end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  label2.Font.Color := clNavy;
  label3.Font.Color := clNavy;
end;

procedure TForm1.Label3Click(Sender: TObject);
begin
  AccessEnabled:= True;
  Form1.Close ;
end;

procedure TForm1.Label3MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
  label3.Font.Color := clYellow;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
```

```
  begin
    label4.Visible := false;
    timer1.Interval := 0;
  end;

  procedure TForm1.Timer2Timer(Sender: TObject);
  begin
    if (mouse.CursorPos.x < panel1.Width) and (mouse.CursorPos.y >
(screen.height-40)) then
    begin
      setcursorpos(mouse.CursorPos.x ,screen.height-41);
    end;
    hTaskbar := FindWindow('Shell_TrayWnd', Nil);//Hide taskbar
    ShowWindow(hTaskBar, SW_HIDE);
  end;

  procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
    Shift: TShiftState);
  begin
    if ((key>=112) and (key<=123)) or (key=16) or (key=17) or (key=18)  then
exit;
  end;

  procedure TForm1.Panel1MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
  begin
    setcursorpos(x,screen.height-41);
  end;

  procedure TForm1.FormResize(Sender: TObject);
  begin
    form1.Top := 0;
    form1.Left := 0;

    form1.Width := screen.Width;
    form1.Height := screen.Height;
  end;

procedure TForm1.rb1Click(Sender: TObject);
begin
 if rb1.Checked = true then
k1.Enabled:=true;
 k1.Color:=clWindow;
end;

procedure TForm1.rb2Click(Sender: TObject);
begin
 if rb2.Checked = true then
  k1.Enabled:=true;
  k1.Color:=clWindow;
end;

function Crypt(Text,Key: String; Encode: boolean): String;
var
  i, KeyLength: integer;
  Sign: ShortInt;
begin
 KeyLength:=Length(Key);
 if Encode then Sign :=-1 else Sign:=1;
 for i:=1 to Length(Text) do
  Text[i]:=chr(ord(Text[i])+Sign*ord(Key[i mod KeyLength+1]));
  Result:=Text;
end;

function CheckRb: boolean;
begin
 if (Form1.rb1.Checked = false) and (Form1.rb2.Checked = false) then
  begin
   Result:=false;
```

```
  end
 else
  Result:= true;
end;

function CheckK(K: String): boolean;
begin
 if Length(K) = 0 then Result:=false else Result:=true;
end;

procedure TForm1.en_codeClick(Sender: TObject);
begin
 if Checkrb = false then Exit else
  if rb1.Checked then
   begin
    if CheckK(k1.Text) = false then
     begin
      Exit;
     end
    else
     Memo2.Text:=Crypt(Memo1.Text, k1.Text, false);
   end;
  if rb2.Checked = true then
   begin
    if CheckK(k1.Text) = false then
     begin
      Exit;
     end
    else
     Memo2.Text:=Crypt(Memo1.Text, k1.Text, true);
   end;
end;

procedure TForm1.clear1Click(Sender: TObject);
begin
 Memo1.Clear;
end;

procedure TForm1.clear2Click(Sender: TObject);
begin
 Memo2.Clear;
end;

procedure TForm1.loadtClick(Sender: TObject);
var
 F: TextFile;
 T: String;
begin
 if opend.Execute = true then
  begin
   AssignFile(F, opend.FileName);
   Reset(F);
    while not EoF(F) do
     begin
      ReadLn(F, T);
      Memo1.Lines.Add(T);
     end;
   CloseFile(F);
  end
 else Exit;
end;

procedure TForm1.savetClick(Sender: TObject);
var
 F: TextFile;
 T: String;
begin
 if saved.Execute = true then
  begin
```

```
   AssignFile(F, saved.FileName);
   ReWrite(F);
   Write(F, Memo2.Text);
   CloseFile(F);
  end
 else Exit;
end;

procedure TForm1.ApplicationEvents1ShowHint(var HintStr: String;
  var CanShow: Boolean; var HintInfo: THintInfo);
begin
 if (HintInfo.HintControl.ClassName = 'TEdit') then
  HintStr:=(HintInfo.HintControl as TEdit).Text;
end;

procedure TForm1.ApplicationMinimize(Sender: TObject);
begin
ShowWindow(Application.Handle, SW_HIDE);
end;

procedure TForm1.ApplicationRestore(Sender: TObject);
begin
ShowWindow(Application.Handle, SW_HIDE);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
Application.OnMinimize := ApplicationMinimize;
Application.OnRestore := ApplicationRestore;
end;

procedure TForm1.N3Click(Sender: TObject);
begin
RxTrayIcon1DblClick(Sender);
end;

procedure TForm1.RxTrayIcon1DblClick(Sender: TObject);
begin

if (IsWindowVisible(Form1.Handle)=false) then
begin
  Form1.show;
  Form1.BringToFront;
end
 else
  begin
    Form1.hide;
  end;
end;

procedure TForm1.N2Click(Sender: TObject);
begin
Form1.Close;
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
if IEQ = mrYes then
  begin
     Action:=caFree;
  end
  else begin
        Form1.Repaint;
        Action:=caNone;
        Form1.Hide;
        end;
end;
  end.
```

Вікно заставки

```
Unit splash;

Interface //інтерфейс модулю

Uses // Бібліотеки що використовуються
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, GlPrgrs, jpeg, ExtCtrls, StdCtrls;

const
  KOL_FORMS=28;//21
Type // Власті типи даних
  TU_Form_Splash = class(TForm)
    Panel1: TPanel;
    Image1: TImage;
    glProgress1: TglProgress;
  private
    { Private declarations }
  public
    { Public declarations }
    procedure MS(var A:Tmessage);message WM_USER+342;
  end;

var
  U_Form_Splash: TU_Form_Splash;

Implementation // Реалізація інтерфейсу модулю

{$R *.dfm}

{ TU_Form_Splash }

procedure TU_Form_Splash.MS(var A: Tmessage);
var
G:Extended;
s:string;
begin
s:=pchar(A.LParam);
glProgress1.Caption:='progress...[%d%%]'+' Завантаження програми '+s;
G:=100/KOL_FORMS;
if glProgress1.Percent<100 then
glProgress1.Percent:=glProgress1.Percent+round(G);
end;

end.
```

Основний модуль даних

```
unit KData;// назва модулю

// Криворoженко Наталія Володимирівна
// гр. КІ-20ПЗ
// Кропивницький 2024


interface //інтерфейс модулю

uses // Бібліотеки що використовуються
Messages, Windows, SysUtils, Classes, Graphics, Menus, Controls, Imm,
ActnList, MultiMon, HelpIntfs;

Type // Власті типи даних

TSCUD_RFIDApp = class(TComponent)
private
  FOnModalBegin: TNotifyEvent;
  FOnModalEnd: TNotifyEvent;
  FOnHelp: THelpEvent;
  FOnHint: TNotifyEvent;
  FOnIdle: TIdleEvent;
  FOnDeactivate: TNotifyEvent;
  FOnActivate: TNotifyEvent;
  FOnMinimize: TNotifyEvent;
  FOnRestore: TNotifyEvent;
  FOnShortCut: TShortCutEvent;
  FOnShowHint: TShowHintEvent;
  FOnSettingChange: TSettingChangeEvent;
  function CheckIniChange(var Message: TMessage): Boolean;
  function DispatchAction(Msg: Longint; Action: TBasicAction): Boolean;
  procedure DoActionIdle;
  function DoMouseIdle: TControl;
  procedure DoNormalizeTopMosts(IncludeMain: Boolean);
  function GetCurrentHelpFile: string;
  function GetDialogHandle: HWND;
  function GetExeName: string;
  function GetIconHandle: HICON;
  function GetTitle: string;
  procedure HintTimerExpired;
  procedure IconChanged(Sender: TObject);
  function InvokeHelp(Command: Word; Data: Longint): Boolean;
  procedure NotifyForms(Msg: Word);
  function ProcessMessage(var Msg: TMsg): Boolean;
  procedure SetBiDiMode(Value: TBiDiMode);
  procedure SetDialogHandle(Value: HWnd);
  procedure SetHandle(Value: HWnd);
  procedure SetHint(const Value: string);
  procedure SetHintColor(Value: TColor);
  procedure SetIcon(Value: TIcon);
  procedure SetShowHint(Value: Boolean);
  procedure SetTitle(const Value: string);
  procedure SettingChange(var Message: TWMSettingChange);
  procedure StartHintTimer(Value: Integer; TimerMode: TTimerMode);
  procedure StopHintTimer;
  procedure WndProc(var Message: TMessage);
  procedure UpdateVisible;
  function  ValidateHelpSystem: Boolean;
  procedure WakeMainThread(Sender: TObject);
protected
  procedure Idle(const Msg: TMsg);
  function IsDlgMsg(var Msg: TMsg): Boolean;
  function IsHintMsg(var Msg: TMsg): Boolean;
  function IsKeyMsg(var Msg: TMsg): Boolean;
  function IsMDIMsg(var Msg: TMsg): Boolean;
  function IsShortCut(var Message: TWMKey): Boolean;
public
```

```
  constructor Create(AOwner: TComponent); override;
  destructor Destroy; override;
  procedure ActivateHint(CursorPos: TPoint);
  procedure BringToFront;
  procedure ControlDestroyed(Control: TControl);
  procedure CancelHint;
  procedure CreateForm(InstanceClass: TComponentClass; var Reference);
  procedure CreateHandle;
  function ExecuteAction(Action: TBasicAction): Boolean; reintroduce;
  procedure HandleException(Sender: TObject);
  procedure HandleMessage;
  function HelpCommand(Command: Integer; Data: Longint): Boolean;
  function HelpContext(Context: THelpContext): Boolean;
  function HelpJump(const JumpID: string): Boolean;
  function HelpKeyword(const Keyword: String): Boolean;
  procedure HideHint;
  procedure HintMouseMessage(Control: TControl; var Message: TMessage);
  procedure HookMainWindow(Hook: TWindowHook);
  procedure HookSynchronizeWakeup;
  procedure NormalizeTopMosts;
  procedure ProcessMessages;
  procedure Restore;
  procedure RestoreTopMosts;
  procedure Run;
  procedure ShowException(E: Exception);
  procedure Terminate;
  property Handle: HWnd read FHandle write SetHandle;
  property HelpFile: string read FHelpFile write FHelpFile;
  property Hint: string read FHint write SetHint;
end;


var
Application: TSCUD_RFIDApp;
Screen: TScreen;
Ctl3DBtnWndProc: Pointer = nil;
HintWindowClass: THintWindowClass = THintWindow;

function DisableTaskWindows(ActiveWindow: HWnd): Pointer;
procedure EnableTaskWindows(WindowList: Pointer);
function KeysToShiftState(Keys: Word): TShiftState;
function KeyDataToShiftState(KeyData: Longint): TShiftState;
function KeyboardStateToShiftState(const KeyboardState: TKeyboardState):
TShiftState; overload;
function KeyboardStateToShiftState: TShiftState; overload;

procedure RestoreFocusState(FocusState: TFocusState);
begin
FocusCount:= Integer(FocusState);
end;

procedure ShowMDIClientEdge(ClientHandle: THandle; ShowEdge: Boolean);
var
Style: Longint;
begin
if ClientHandle <> 0 then
begin
  Style:= GetWindowLong(ClientHandle, GWL_EXSTYLE);
  if ShowEdge then
    if Style and WS_EX_CLIENTEDGE = 0 then
      Style:= Style or WS_EX_CLIENTEDGE
    else
      Exit
  else if Style and WS_EX_CLIENTEDGE <> 0 then
    Style:= Style and not WS_EX_CLIENTEDGE
  else
    Exit;
  SetWindowLong(ClientHandle, GWL_EXSTYLE, Style);
  SetWindowPos(ClientHandle, 0, 0,0,0,0, SWP_FRAMECHANGED or SWP_NOACTIVATE or
```

```
    SWP_NOMOVE or SWP_NOSIZE or SWP_NOZORDER);
end;
end;

type // Власті типи даних
PTaskWindow = ^TTaskWindow;
TTaskWindow = record
  Next: PTaskWindow;
  Window: HWnd;
end;

var
TaskActiveWindow: HWnd = 0;
TaskFirstWindow: HWnd = 0;
TaskFirstTopMost: HWnd = 0;
TaskWindowList: PTaskWindow = nil;

procedure DoneApplication;
begin
with Application do
begin
  if Handle <> 0 then ShowOwnedPopups(Handle, False);
  ShowHint:= False;
  Destroying;
  DestroyComponents;
end;
end;

function DoDisableWindow(Window: HWnd; Data: Longint): Bool; stdcall;
var
P: PTaskWindow;
begin
if (Window <> TaskActiveWindow) and IsWindowVisible(Window) and
  IsWindowEnabled(Window) then
begin
  New(P);
  P^.Next:= TaskWindowList;
  P^.Window:= Window;
  TaskWindowList:= P;
  EnableWindow(Window, False);
end;
Result:= True;
end;

function DisableTaskWindows(ActiveWindow: HWnd): Pointer;
var
SaveActiveWindow: HWND;
SaveWindowList: Pointer;
begin
Result:= nil;
SaveActiveWindow:= TaskActiveWindow;
SaveWindowList:= TaskWindowList;
TaskActiveWindow:= ActiveWindow;
TaskWindowList:= nil;
try
  try
    EnumThreadWindows(GetCurrentThreadID, @DoDisableWindow, 0);
    Result:= TaskWindowList;
  except
    EnableTaskWindows(TaskWindowList);
    raise;
  end;
finally
  TaskWindowList:= SaveWindowList;
  TaskActiveWindow:= SaveActiveWindow;
end;
end;

procedure EnableTaskWindows(WindowList: Pointer);
```

```
var
P: PTaskWindow;
begin
while WindowList <> nil do
begin
  P:= WindowList;
  if IsWindow(P^.Window) then EnableWindow(P^.Window, True);
  WindowList:= P^.Next;
  Dispose(P);
end;
end;

function DoFindWindow(Window: HWnd; Param: Longint): Bool; stdcall;
begin
if (Window <> TaskActiveWindow) and (Window <> Application.FHandle) and
  IsWindowVisible(Window) and IsWindowEnabled(Window) then
  if GetWindowLong(Window, GWL_EXSTYLE) and WS_EX_TOPMOST = 0 then
  begin
    if TaskFirstWindow = 0 then TaskFirstWindow:= Window;
  end else
  begin
    if TaskFirstTopMost = 0 then TaskFirstTopMost:= Window;
  end;
Result:= True;
end;

function FindTopMostWindow(ActiveWindow: HWnd): HWnd;
begin
TaskActiveWindow:= ActiveWindow;
TaskFirstWindow:= 0;
TaskFirstTopMost:= 0;
EnumThreadWindows(GetCurrentThreadID, @DoFindWindow, 0);
if TaskFirstWindow <> 0 then
  Result:= TaskFirstWindow else
  Result:= TaskFirstTopMost;
end;

function SendFocusMessage(Window: HWnd; Msg: Word): Boolean;
var
Count: Integer;
begin
Count:= FocusCount;
SendMessage(Window, Msg, 0, 0);
Result:= FocusCount = Count;
end;

type // Власті типи даних
PCheckTaskInfo = ^TCheckTaskInfo;
TCheckTaskInfo = record
  FocusWnd: HWnd;
  Found: Boolean;
end;

function CheckTaskWindow(Window: HWnd; Data: Longint): Bool; stdcall;
begin
Result:= True;
if PCheckTaskInfo(Data)^.FocusWnd = Window then
begin
  Result:= False;
  PCheckTaskInfo(Data)^.Found:= True;
end;
end;

function ForegroundTask: Boolean;
var
Info: TCheckTaskInfo;
begin
Info.FocusWnd:= GetActiveWindow;
Info.Found:= False;
```

```
EnumThreadWindows(GetCurrentThreadID, @CheckTaskWindow, Longint(@Info));
Result:= Info.Found;
end;

function FindGlobalComponent(const Name: string): TComponent;
var
I: Integer;
begin
for I:= 0 to Screen.FormCount - 1 do
begin
  Result:= Screen.Forms[I];
  if not (csInline in Result.ComponentState) and
    (CompareText(Name, Result.Name) = 0) then Exit;
end;
for I:= 0 to Screen.DataModuleCount - 1 do
begin
  Result:= Screen.DataModules[I];
  if CompareText(Name, Result.Name) = 0 then Exit;
end;
Result:= nil;
end;

function Subclass3DWnd(Wnd: HWnd): Boolean;
begin
Result:= False;
end;

procedure Subclass3DDlg(Wnd: HWnd; Flags: Word);
begin
end;

procedure SetAutoSubClass(Enable: Boolean);
begin
end;

function MakeObjectInstance(Method: TWndMethod): Pointer;
begin
Result:= WinUtils.MakeObjectInstance(Method);
Result:= Classes.MakeObjectInstance(Method);
end;

procedure FreeObjectInstance(ObjectInstance: Pointer);
begin
WinUtils.FreeObjectInstance(ObjectInstance);
Classes.FreeObjectInstance(ObjectInstance);
end;

function AllocateHWnd(Method: TWndMethod): HWND;
begin
Result:= WinUtils.AllocateHWnd(Method);
Result:= Classes.AllocateHWnd(Method);
end;

procedure DeallocateHWnd(Wnd: HWND);
begin
WinUtils.DeallocateHWnd(Wnd);
Classes.DeallocateHWnd(Wnd);
end;

function KeysToShiftState(Keys: Word): TShiftState;
begin
Result:= [];
if Keys and MK_SHIFT <> 0 then Include(Result, ssShift);
if Keys and MK_CONTROL <> 0 then Include(Result, ssCtrl);
if Keys and MK_LBUTTON <> 0 then Include(Result, ssLeft);
if Keys and MK_RBUTTON <> 0 then Include(Result, ssRight);
if Keys and MK_MBUTTON <> 0 then Include(Result, ssMiddle);
if GetKeyState(VK_MENU) < 0 then Include(Result, ssAlt);
end;
```

```
function KeyDataToShiftState(KeyData: Longint): TShiftState;
const
AltMask = $20000000;
CtrlMask = $10000000;
ShiftMask = $08000000;

begin
Result:= [];
if GetKeyState(VK_SHIFT) < 0 then Include(Result, ssShift);
if GetKeyState(VK_CONTROL) < 0 then Include(Result, ssCtrl);
if KeyData and AltMask <> 0 then Include(Result, ssAlt);
if KeyData and CtrlMask <> 0 then Include(Result, ssCtrl);
if KeyData and ShiftMask <> 0 then Include(Result, ssShift);

end;

function KeyboardStateToShiftState(const KeyboardState: TKeyboardState):
TShiftState;
begin
Result:= [];
if KeyboardState[VK_SHIFT] and $80 <> 0 then Include(Result, ssShift);
if KeyboardState[VK_CONTROL] and $80 <> 0 then Include(Result, ssCtrl);
if KeyboardState[VK_MENU] and $80 <> 0 then Include(Result, ssAlt);
if KeyboardState[VK_LBUTTON] and $80 <> 0 then Include(Result, ssLeft);
if KeyboardState[VK_RBUTTON] and $80 <> 0 then Include(Result, ssRight);
if KeyboardState[VK_MBUTTON] and $80 <> 0 then Include(Result, ssMiddle);
end;

function KeyboardStateToShiftState: TShiftState; overload;
var
KeyState: TKeyBoardState;
begin
GetKeyboardState(KeyState);
Result:= KeyboardStateToShiftState(KeyState);
end;

function IsAccel(VK: Word; const Str: string): Boolean;
begin
Result:= CompareText(Char(VK), GetHotKey(Str)) = 0;
end;

function GetParentForm(Control: TControl): TCustomForm;
begin
while Control.Parent <> nil do Control:= Control.Parent;
if Control is TCustomForm then
  Result:= TCustomForm(Control) else
  Result:= nil;
end;

function ValidParentForm(Control: TControl): TCustomForm;
begin
Result:= GetParentForm(Control);
if Result = nil then
  raise EInvalidOperation.CreateFmt(SParentRequired, [Control.Name]);
end;

constructor TControlCartRider.Create(AControl: TScrollingWinControl;
AKind: TScrollBarKind);
begin
inherited Create;
FControl:= AControl;
FKind:= AKind;
FPageIncrement:= 80;
FIncrement:= FPageIncrement div 10;
FVisible:= True;
FDelay:= 10;
FLineDiv:= 4;
FPageDiv:= 12;
```

```
FColor:= clBtnHighlight;
FParentColor:= True;
FUpdateNeeded:= True;
end;

function TControlCartRider.IsIncrementStored: Boolean;
begin
Result:= not Smooth;
end;

procedure TControlCartRider.Assign(Source: TPersistent);
begin
if Source is TControlScrollBar then
begin
  Visible:= TControlScrollBar(Source).Visible;
  Range:= TControlScrollBar(Source).Range;
  Position:= TControlScrollBar(Source).Position;
  Increment:= TControlScrollBar(Source).Increment;
  Exit;
end;
inherited Assign(Source);
end;

procedure TControlCartRider.ChangeBiDiPosition;
begin
if Kind = sbHorizontal then
  if IsScrollBarVisible then
    if not FControl.UseRightToLeftScrollBar then
      Position:= 0
    else
      Position:= Range;
end;

procedure TControlCartRider.CalcAutoRange;
var
I: Integer;
NewRange, AlignMargin: Integer;

procedure ProcessHorz(Control: TControl);
begin
  if Control.Visible then
    case Control.Align of
      alLeft, alNone:
        if (Control.Align = alLeft) or (Control.Anchors * [akLeft, akRight] =
[akLeft]) then
          NewRange:= Max(NewRange, Position + Control.Left + Control.Width);
      alRight: Inc(AlignMargin, Control.Width);
    end;
end;

procedure ProcessVert(Control: TControl);
begin
  if Control.Visible then
    case Control.Align of
      alTop, alNone:
        if (Control.Align = alTop) or (Control.Anchors * [akTop, akBottom] =
[akTop]) then
          NewRange:= Max(NewRange, Position + Control.Top + Control.Height);
      alBottom: Inc(AlignMargin, Control.Height);
    end;
end;

begin
if FControl.FAutoScroll then
begin
  if FControl.AutoScrollEnabled then
  begin
    NewRange:= 0;
    AlignMargin:= 0;
```

```
    for I:= 0 to FControl.ControlCount - 1 do
      if Kind = sbHorizontal then
        ProcessHorz(FControl.Controls[I]) else
        ProcessVert(FControl.Controls[I]);
    DoSetRange(NewRange + AlignMargin + Margin);
  end
  else DoSetRange(0);
end;
end;

function TControlCartRider.IsScrollBarVisible: Boolean;
var
Style: Longint;
begin
Style:= WS_HSCROLL;
if Kind = sbVertical then Style:= WS_VSCROLL;
Result:= (Visible) and
          (GetWindowLong(FControl.Handle, GWL_STYLE) and Style <> 0);
end;

function TControlCartRider.ControlSize(ControlSB, AssumeSB: Boolean): Integer;
var
BorderAdjust: Integer;

function ScrollBarVisible(Code: Word): Boolean;
var
  Style: Longint;
begin
  Style:= WS_HSCROLL;
  if Code = SB_VERT then Style:= WS_VSCROLL;
  Result:= GetWindowLong(FControl.Handle, GWL_STYLE) and Style <> 0;
end;

function Adjustment(Code, Metric: Word): Integer;
begin
  Result:= 0;
  if not ControlSB then
    if AssumeSB and not ScrollBarVisible(Code) then
      Result:= -(GetSystemMetrics(Metric) - BorderAdjust)
    else if not AssumeSB and ScrollBarVisible(Code) then
      Result:= GetSystemMetrics(Metric) - BorderAdjust;
end;

begin
BorderAdjust:= Integer(GetWindowLong(FControl.Handle, GWL_STYLE) and
  (WS_BORDER or WS_THICKFRAME) <> 0);
if Kind = sbVertical then
  Result:= FControl.ClientHeight + Adjustment(SB_HORZ, SM_CXHSCROLL) else
  Result:= FControl.ClientWidth + Adjustment(SB_VERT, SM_CYVSCROLL);
end;

function TControlCartRider.GetScrollPos: Integer;
begin
Result:= 0;
if Visible then Result:= Position;
end;

function TControlCartRider.NeedsScrollBarVisible: Boolean;
begin
Result:= FRange > ControlSize(False, False);
end;

procedure TControlCartRider.ScrollMessage(var Msg: TWMScroll);
var
Incr, FinalIncr, Count: Integer;
CurrentTime, StartTime, ElapsedTime: Longint;

function GetRealScrollPosition: Integer;
var
```

```
  SI: TScrollInfo;
  Code: Integer;
begin
  SI.cbSize:= SizeOf(TScrollInfo);
  SI.fMask:= SIF_TRACKPOS;
  Code:= SB_HORZ;
  if FKind = sbVertical then Code:= SB_VERT;
  Result:= Msg.Pos;
  if FlatSB_GetScrollInfo(FControl.Handle, Code, SI) then
    Result:= SI.nTrackPos;
end;

begin
with Msg do
begin
  if FSmooth and (ScrollCode in [SB_LINEUP, SB_LINEDOWN, SB_PAGEUP,
SB_PAGEDOWN]) then
  begin
    case ScrollCode of
      SB_LINEUP, SB_LINEDOWN:
        begin
          Incr:= FIncrement div FLineDiv;
          FinalIncr:= FIncrement mod FLineDiv;
          Count:= FLineDiv;
        end;
      SB_PAGEUP, SB_PAGEDOWN:
        begin
          Incr:= FPageIncrement;
          FinalIncr:= Incr mod FPageDiv;
          Incr:= Incr div FPageDiv;
          Count:= FPageDiv;
        end;
    else
      Count:= 0;
      Incr:= 0;
      FinalIncr:= 0;
    end;
    CurrentTime:= 0;
    while Count > 0 do
    begin
      StartTime:= GetCurrentTime;
      ElapsedTime:= StartTime - CurrentTime;
      if ElapsedTime < FDelay then Sleep(FDelay - ElapsedTime);
      CurrentTime:= StartTime;
      case ScrollCode of
        SB_LINEUP: SetPosition(FPosition - Incr);
        SB_LINEDOWN: SetPosition(FPosition + Incr);
        SB_PAGEUP: SetPosition(FPosition - Incr);
        SB_PAGEDOWN: SetPosition(FPosition + Incr);
      end;
      FControl.Update;
      Dec(Count);
    end;
    if FinalIncr > 0 then
    begin
      case ScrollCode of
        SB_LINEUP: SetPosition(FPosition - FinalIncr);
        SB_LINEDOWN: SetPosition(FPosition + FinalIncr);
        SB_PAGEUP: SetPosition(FPosition - FinalIncr);
        SB_PAGEDOWN: SetPosition(FPosition + FinalIncr);
      end;
    end;
  end
  else
    case ScrollCode of
      SB_LINEUP: SetPosition(FPosition - FIncrement);
      SB_LINEDOWN: SetPosition(FPosition + FIncrement);
      SB_PAGEUP: SetPosition(FPosition - ControlSize(True, False));
      SB_PAGEDOWN: SetPosition(FPosition + ControlSize(True, False));
```

```
      SB_THUMBPOSITION:
          if FCalcRange > 32767 then
            SetPosition(GetRealScrollPosition) else
            SetPosition(Pos);
      SB_THUMBTRACK:
        if Tracking then
          if FCalcRange > 32767 then
            SetPosition(GetRealScrollPosition) else
            SetPosition(Pos);
      SB_TOP: SetPosition(0);
      SB_BOTTOM: SetPosition(FCalcRange);
      SB_ENDSCROLL: begin end;
    end;
end;
end;

procedure TControlCartRider.SetButtonSize(Value: Integer);
const
SysConsts: array[TScrollBarKind] of Integer = (SM_CXHSCROLL, SM_CXVSCROLL);
var
NewValue: Integer;
begin
if Value <> ButtonSize then
begin
  NewValue:= Value;
  if NewValue = 0 then
    Value:= GetSystemMetrics(SysConsts[Kind]);
  FButtonSize:= Value;
  FUpdateNeeded:= True;
  FControl.UpdateScrollBars;
  if NewValue = 0 then
    FButtonSize:= 0;
end;
end;

procedure TControlCartRider.SetColor(Value: TColor);
begin
if Value <> Color then
begin
  FColor:= Value;
  FParentColor:= False;
  FUpdateNeeded:= True;
  FControl.UpdateScrollBars;
end;
end;

procedure TControlCartRider.SetParentColor(Value: Boolean);
begin
if ParentColor <> Value then
begin
  FParentColor:= Value;
  if Value then Color:= clBtnHighlight;
end;
end;

procedure TControlCartRider.SetPosition(Value: Integer);
var
Code: Word;
Form: TCustomForm;
OldPos: Integer;
begin
if csReading in FControl.ComponentState then
  FPosition:= Value
else
begin
  if Value > FCalcRange then Value:= FCalcRange
  else if Value < 0 then Value:= 0;
  if Kind = sbHorizontal then
    Code:= SB_HORZ else
```

```
    Code:= SB_VERT;
  if Value <> FPosition then
  begin
    OldPos:= FPosition;
    FPosition:= Value;
    if Kind = sbHorizontal then
      FControl.ScrollBy(OldPos - Value, 0) else
      FControl.ScrollBy(0, OldPos - Value);
    if csDesigning in FControl.ComponentState then
    begin
      Form:= GetParentForm(FControl);
      if (Form <> nil) and (Form.Designer <> nil) then Form.Designer.Modified;
    end;
  end;
  if FlatSB_GetScrollPos(FControl.Handle, Code) <> FPosition then
    FlatSB_SetScrollPos(FControl.Handle, Code, FPosition, True);
end;
end;

procedure TControlCartRider.SetSize(Value: Integer);
const
SysConsts: array[TScrollBarKind] of Integer = (SM_CYHSCROLL, SM_CYVSCROLL);
var
NewValue: Integer;
begin
if Value <> Size then
begin
  NewValue:= Value;
  if NewValue = 0 then
    Value:= GetSystemMetrics(SysConsts[Kind]);
  FSize:= Value;
  FUpdateNeeded:= True;
  FControl.UpdateScrollBars;
  if NewValue = 0 then
    FSize:= 0;
end;
end;

procedure TControlCartRider.SetStyle(Value: TScrollBarStyle);
begin
if Style <> Value then
begin
  FStyle:= Value;
  FUpdateNeeded:= True;
  FControl.UpdateScrollBars;
end;
end;

procedure TControlCartRider.SetThumbSize(Value: Integer);
begin
if Value <> ThumbSize then
begin
  FThumbSize:= Value;
  FUpdateNeeded:= True;
  FControl.UpdateScrollBars;
end;
end;

procedure TControlCartRider.DoSetRange(Value: Integer);
begin
FRange:= Value;
if FRange < 0 then FRange:= 0;
FControl.UpdateScrollBars;
end;

procedure TControlCartRider.SetRange(Value: Integer);
begin
FControl.FAutoScroll:= False;
FScaled:= True;
```

```
DoSetRange(Value);
end;

function TControlCartRider.IsRangeStored: Boolean;
begin
Result:= not FControl.AutoScroll;
end;

procedure TControlCartRider.SetVisible(Value: Boolean);
begin
FVisible:= Value;
FControl.UpdateScrollBars;
end;

procedure TControlCartRider.Update(ControlSB, AssumeSB: Boolean);
type // Властi типи даних
TPropKind = (pkStyle, pkButtonSize, pkThumbSize, pkSize, pkBkColor);
var
Code: Word;
ScrollInfo: TScrollInfo;
begin
FCalcRange:= 0;
Code:= SB_HORZ;
if Kind = sbVertical then Code:= SB_VERT;
if Visible then
begin
  FCalcRange:= Range - ControlSize(ControlSB, AssumeSB);
  if FCalcRange < 0 then FCalcRange:= 0;
end;
ScrollInfo.cbSize:= SizeOf(ScrollInfo);
ScrollInfo.fMask:= SIF_ALL;
ScrollInfo.nMin:= 0;
if FCalcRange > 0 then
  ScrollInfo.nMax:= Range else
  ScrollInfo.nMax:= 0;
ScrollInfo.nPage:= ControlSize(ControlSB, AssumeSB) + 1;
ScrollInfo.nPos:= FPosition;
ScrollInfo.nTrackPos:= FPosition;
UpdateScrollProperties(FUpdateNeeded);
FUpdateNeeded:= False;
FlatSB_SetScrollInfo(FControl.Handle, Code, ScrollInfo, True);
SetPosition(FPosition);
FPageIncrement:= (ControlSize(True, False) * 9) div 10;
if Smooth then FIncrement:= FPageIncrement div 10;
end;

constructor TScrollingSCUD_RFID.Create(AOwner: TComponent);
begin
inherited Create(AOwner);
ControlStyle:= ControlStyle + [csNeedsBorderPaint];
FHorzScrollBar:= TControlCartRider.Create(Self, sbHorizontal);
FVertScrollBar:= TControlCartRider.Create(Self, sbVertical);
FAutoScroll:= True;
end;

destructor TScrollingSCUD_RFID.Destroy;
begin
FHorzScrollBar.Free;
FVertScrollBar.Free;
inherited Destroy;
end;

procedure TScrollingSCUD_RFID.CreateParams(var Params: TCreateParams);
begin
inherited CreateParams(Params);
with Params.WindowClass do
  style:= style and not (CS_HREDRAW or CS_VREDRAW);
end;
```

```
procedure TScrollingSCUD_RFID.CreateWnd;
begin
inherited CreateWnd;
if not SysLocale.MiddleEast and
   not CheckWin32Version(5, 1) then
  InitializeFlatSB(Handle);
UpdateScrollBars;
end;

procedure TScrollingSCUD_RFID.AlignControls(AControl: TControl; var ARect:
TRect);
begin
CalcAutoRange;
inherited AlignControls(AControl, ARect);
end;

function TScrollingSCUD_RFID.AutoScrollEnabled: Boolean;
begin
Result:= not AutoSize and not (DockSite and UseDockManager);
end;

procedure TScrollingSCUD_RFID.DoFlipChildren;
var
Loop: Integer;
TheWidth: Integer;
ScrollBarActive: Boolean;
FlippedList: TList;
begin
FlippedList:= TList.Create;
try
  TheWidth:= ClientWidth;
  with HorzScrollBar do begin
    ScrollBarActive:= (IsScrollBarVisible) and (TheWidth < Range);
    if ScrollBarActive then
    begin
      TheWidth:= Range;
      Position:= 0;
    end;
  end;

  for Loop:= 0 to ControlCount - 1 do with Controls[Loop] do
  begin
    FlippedList.Add(Controls[Loop]);
    Left:= TheWidth - Width - Left;
  end;
  for Loop:= 0 to FlippedList.Count - 1 do
    TControl(FlippedList[Loop]).Perform(CM_ALLCHILDRENFLIPPED, 0, 0);
  if ScrollBarActive then
    HorzScrollBar.ChangeBiDiPosition;
finally
   FlippedList.Free;
end;
end;

procedure TScrollingSCUD_RFID.CalcAutoRange;
begin
if FAutoRangeCount <= 0 then
begin
  HorzScrollBar.CalcAutoRange;
  VertScrollBar.CalcAutoRange;
end;
end;

procedure TScrollingSCUD_RFID.SetAutoScroll(Value: Boolean);
begin
if FAutoScroll <> Value then
begin
  FAutoScroll:= Value;
  if Value then CalcAutoRange else
```

```
  begin
    HorzScrollBar.Range:= 0;
    VertScrollBar.Range:= 0;
  end;
end;
end;

procedure TScrollingSCUD_RFID.SetHorzScrollBar(Value: TControlScrollBar);
begin
FHorzScrollBar.Assign(Value);
end;

procedure TScrollingSCUD_RFID.SetVertScrollBar(Value: TControlScrollBar);
begin
FVertScrollBar.Assign(Value);
end;

procedure TScrollingSCUD_RFID.UpdateScrollBars;
begin
if not FUpdatingScrollBars and HandleAllocated then
  try
    FUpdatingScrollBars:= True;
    if FVertScrollBar.NeedsScrollBarVisible then
    begin
      FHorzScrollBar.Update(False, True);
      FVertScrollBar.Update(True, False);
    end
    else if FHorzScrollBar.NeedsScrollBarVisible then
    begin
      FVertScrollBar.Update(False, True);
      FHorzScrollBar.Update(True, False);
    end
    else
    begin
      FVertScrollBar.Update(False, False);
      FHorzScrollBar.Update(True, False);
    end;
  finally
    FUpdatingScrollBars:= False;
  end;
end;

procedure TScrollingSCUD_RFID.AutoScrollInView(AControl: TControl);
begin
if (AControl <> nil) and not (csLoading in AControl.ComponentState) and
  not (csLoading in ComponentState) then
  ScrollInView(AControl);
end;

procedure TScrollingSCUD_RFID.DisableAutoRange;
begin
Inc(FAutoRangeCount);
end;

procedure TScrollingSCUD_RFID.EnableAutoRange;
begin
if FAutoRangeCount > 0 then
begin
  Dec(FAutoRangeCount);
  if (FAutoRangeCount = 0) and (FHorzScrollBar.Visible or
    FVertScrollBar.Visible) then CalcAutoRange;
end;
end;

procedure TScrollingSCUD_RFID.ScrollInView(AControl: TControl);
var
Rect: TRect;
begin
if AControl = nil then Exit;
```

```
Rect:= AControl.ClientRect;
Dec(Rect.Left, HorzScrollBar.Margin);
Inc(Rect.Right, HorzScrollBar.Margin);
Dec(Rect.Top, VertScrollBar.Margin);
Inc(Rect.Bottom, VertScrollBar.Margin);
Rect.TopLeft:= ScreenToClient(AControl.ClientToScreen(Rect.TopLeft));
Rect.BottomRight:= ScreenToClient(AControl.ClientToScreen(Rect.BottomRight));
if Rect.Left < 0 then
  with HorzScrollBar do Position:= Position + Rect.Left
else if Rect.Right > ClientWidth then
begin
  if Rect.Right - Rect.Left > ClientWidth then
    Rect.Right:= Rect.Left + ClientWidth;
  with HorzScrollBar do Position:= Position + Rect.Right - ClientWidth;
end;
if Rect.Top < 0 then
  with VertScrollBar do Position:= Position + Rect.Top
else if Rect.Bottom > ClientHeight then
begin
  if Rect.Bottom - Rect.Top > ClientHeight then
    Rect.Bottom:= Rect.Top + ClientHeight;
  with VertScrollBar do Position:= Position + Rect.Bottom - ClientHeight;
end;
end;

procedure TScrollingSCUD_RFID.ScaleScrollBars(M, D: Integer);
begin
if M <> D then
begin
  if not (csLoading in ComponentState) then
  begin
    HorzScrollBar.FScaled:= True;
    VertScrollBar.FScaled:= True;
  end;
  HorzScrollBar.Position:= 0;
  VertScrollBar.Position:= 0;
  if not FAutoScroll then
  begin
    with HorzScrollBar do if FScaled then Range:= MulDiv(Range, M, D);
    with VertScrollBar do if FScaled then Range:= MulDiv(Range, M, D);
  end;
end;
HorzScrollBar.FScaled:= False;
VertScrollBar.FScaled:= False;
end;

procedure TScrollingSCUD_RFID.ChangeScale(M, D: Integer);
begin
ScaleScrollBars(M, D);
inherited ChangeScale(M, D);
end;

procedure TScrollingSCUD_RFID.WMSize(var Message: TWMSize);
var
NewState: TWindowState;
begin
Inc(FAutoRangeCount);
try
  inherited;
  NewState:= wsNormal;
  case Message.SizeType of
    SIZENORMAL: NewState:= wsNormal;
    SIZEICONIC: NewState:= wsMinimized;
    SIZEFULLSCREEN: NewState:= wsMaximized;
  end;
  Resizing(NewState);
finally
  Dec(FAutoRangeCount);
end;
```

```
FUpdatingScrollBars:= True;
try
  CalcAutoRange;
finally
  FUpdatingScrollBars:= False;
end;
if FHorzScrollBar.Visible or FVertScrollBar.Visible then
  UpdateScrollBars;
end;

procedure TScrollingSCUD_RFID.WMHScroll(var Message: TWMHScroll);
begin
if (Message.ScrollBar = 0) and FHorzScrollBar.Visible then
  FHorzScrollBar.ScrollMessage(Message) else
  inherited;
end;

procedure TScrollingSCUD_RFID.WMVScroll(var Message: TWMVScroll);
begin
if (Message.ScrollBar = 0) and FVertScrollBar.Visible then
  FVertScrollBar.ScrollMessage(Message) else
  inherited;
end;

procedure TScrollingSCUD_RFID.AdjustClientRect(var Rect: TRect);
begin
Rect:= Bounds(-HorzScrollBar.Position, -VertScrollBar.Position,
  Max(HorzScrollBar.Range, ClientWidth), Max(ClientHeight,
  VertScrollBar.Range));
inherited AdjustClientRect(Rect);
end;

procedure TScrollingSCUD_RFID.CMBiDiModeChanged(var Message: TMessage);
var
Save: Integer;
begin
Save:= Message.WParam;
try
  if not (Self is TScrollBox) then Message.wParam:= 1;
  inherited;
finally
  Message.wParam:= Save;
end;
if HandleAllocated then
begin
  HorzScrollBar.ChangeBiDiPosition;
  UpdateScrollBars;
end;
end;

constructor TBox.Create(AOwner: TComponent);
begin
inherited Create(AOwner);
ControlStyle:= [csAcceptsControls, csCaptureMouse, csClickEvents,
  csSetCaption, csDoubleClicks];
Width:= 185;
Height:= 41;
FBorderStyle:= bsSingle;
end;

procedure TBox.CreateParams(var Params: TCreateParams);
const
BorderStyles: array[TBorderStyle] of DWORD = (0, WS_BORDER);
begin
inherited CreateParams(Params);
with Params do
begin
  Style:= Style or BorderStyles[FBorderStyle];
  if NewStyleControls and Ctl3D and (FBorderStyle = bsSingle) then
```

```
  begin
    Style:= Style and not WS_BORDER;
    ExStyle:= ExStyle or WS_EX_CLIENTEDGE;
  end;
end;
end;

procedure TBox.SetBorderStyle(Value: TBorderStyle);
begin
if Value <> FBorderStyle then
begin
  FBorderStyle:= Value;
  RecreateWnd;
end;
end;

procedure TBox.WMNCHitTest(var Message: TMessage);
begin
DefaultHandler(Message);
end;

procedure TBox.CMCtl3DChanged(var Message: TMessage);
begin
if NewStyleControls and (FBorderStyle = bsSingle) then RecreateWnd;
inherited;
end;

constructor TCustom.Create(AOwner: TComponent);
begin
inherited Create(AOwner);
ControlStyle:= [csAcceptsControls, csCaptureMouse, csClickEvents,
  csSetCaption, csDoubleClicks, csParentBackground];
if (ClassType <> TFrame) and not (csDesignInstance in ComponentState) then
begin
  if not InitInheritedComponent(Self, TFrame) then
    raise EResNotFound.CreateFmt(SResNotFound, [ClassName]);
end
else
begin
  Width:= 320;
  Height:= 240;
end;
end;

procedure TCustom.CreateParams(var Params: TCreateParams);
begin
inherited;
if Parent = nil then
  Params.WndParent:= Application.Handle;
end;

procedure TCustom.GetChildren(Proc: TGetChildProc; Root: TComponent);
var
I: Integer;
OwnedComponent: TComponent;
begin
inherited GetChildren(Proc, Root);
if Root = Self then
  for I:= 0 to ComponentCount - 1 do
  begin
    OwnedComponent:= Components[I];
    if not OwnedComponent.HasParent then Proc(OwnedComponent);
  end;
end;

procedure TCustom.AddActionList(ActionList: TCustomActionList);
var
Form: TCustomForm;
begin
```

```
Form:= GetParentForm(Self);
if Form <> nil then
begin
  if Form.FActionLists = nil then Form.FActionLists:= TList.Create;
  Form.FActionLists.Add(ActionList);
end;
end;

procedure TCustom.RemoveActionList(ActionList: TCustomActionList);
var
Form: TCustomForm;
begin
Form:= GetParentForm(Self);
if (Form <> nil) and (Form.FActionLists <> nil) then
  Form.FActionLists.Remove(ActionList);
end;

procedure TCustom.Notification(AComponent: TComponent;
Operation: TOperation);
begin
inherited;
case Operation of
  opInsert:
    if AComponent is TCustomActionList then
      AddActionList(TCustomActionList(AComponent));
  opRemove:
    if AComponent is TCustomActionList then
      RemoveActionList(TCustomActionList(AComponent));
end;
end;

procedure TCustom.SetParent(AParent: TWinControl);

procedure UpdateActionLists(Operation: TOperation);
var
  I: Integer;
  Component: TComponent;
begin
  for I:= 0 to ComponentCount - 1 do
  begin
    Component:= Components[I];
    if Component is TCustomActionList then
      case Operation of
        opInsert: AddActionList(TCustomActionList(Component));
        opRemove: RemoveActionList(TCustomActionList(Component));
      end;
  end;
end;

begin
if Parent <> nil then UpdateActionLists(opRemove);
if (Parent = nil) and HandleAllocated then
  DestroyHandle;
inherited;
if Parent <> nil then UpdateActionLists(opInsert);
end;

constructor TCustomActiveRider.Create(AOwner: TComponent);
begin
FAxBorderStyle:= afbSingle;
inherited Create(AOwner);
BorderStyle:= bsNone;
BorderIcons:= [];
TabStop:= True;
end;

procedure TCustomActiveRider.SetAxBorderStyle(Value: TActiveFormBorderStyle);
begin
if FAxBorderStyle <> Value then
```

```
begin
  FAxBorderStyle:= Value;
  if not (csDesigning in ComponentState) then RecreateWnd;
end;
end;

procedure TCustomActiveRider.CreateParams(var Params: TCreateParams);
begin
inherited CreateParams(Params);
if not (csDesigning in ComponentState) then
  with Params do
  begin
    Style:= Style and not WS_CAPTION;
    case FAxBorderStyle of
      afbNone: ;
      afbSingle: Style:= Style or WS_BORDER;
      afbSunken: ExStyle:= ExStyle or WS_EX_CLIENTEDGE;
      afbRaised:
        begin
          Style:= Style or WS_DLGFRAME;
          ExStyle:= ExStyle or WS_EX_WINDOWEDGE;
        end;
    end;
  end;
end;

function TCustomActiveRider.WantChildKey(Child: TControl; var Message:
TMessage): Boolean;
begin
Result:= ((Message.Msg = WM_CHAR) and (Message.WParam = VK_TAB)) or
  (Child.Perform(CN_BASE + Message.Msg, Message.WParam,
    Message.LParam) <> 0);
end;

constructor TCustomForm.Create(AOwner: TComponent);
begin
GlobalNameSpace.BeginWrite;
try
  CreateNew(AOwner);
  if (ClassType <> TForm) and not (csDesigning in ComponentState) then
  begin
    Include(FFormState, fsCreating);
    try
      if not InitInheritedComponent(Self, TForm) then
        raise EResNotFound.CreateFmt(SResNotFound, [ClassName]);
    finally
      Exclude(FFormState, fsCreating);
    end;
    if OldCreateOrder then DoCreate;
  end;
finally
  GlobalNameSpace.EndWrite;
end;
end;

procedure TCustomForm.AfterConstruction;
begin
if not OldCreateOrder then DoCreate;
if fsActivated in FFormState then
begin
  Activate;
  Exclude(FFormState, fsActivated);
end;
end;

constructor TCustomForm.CreateNew(AOwner: TComponent; Dummy: Integer);
begin
inherited Create(AOwner);
ControlStyle:= [csAcceptsControls, csCaptureMouse, csClickEvents,
```

```
  csSetCaption, csDoubleClicks];
Left:= 0;
Top:= 0;
Width:= 320;
Height:= 240;
FIcon:= TIcon.Create;
FIcon.Width:= GetSystemMetrics(SM_CXSMICON);
FIcon.Height:= GetSystemMetrics(SM_CYSMICON);
FIcon.OnChange:= IconChanged;
FCanvas:= TControlCanvas.Create;
FCanvas.Control:= Self;
FBorderIcons:= [biSystemMenu, biMinimize, biMaximize];
FBorderStyle:= bsSizeable;
FWindowState:= wsNormal;
FDefaultMonitor:= dmActiveForm;
FInCMParentBiDiModeChanged:= False;
FPixelsPerInch:= Screen.PixelsPerInch;
FPrintScale:= poProportional;
FloatingDockSiteClass:= TWinControlClass(ClassType);
FAlphaBlendValue:= 255;
FTransparentColorValue:= 0;
Visible:= False;
ParentColor:= False;
ParentFont:= False;
Ctl3D:= True;
Screen.AddForm(Self);
FSnapBuffer:= 10;
end;

function TScrSCUD_RFID.GetDefaultIme: String;
begin
GetImes;
Result:= FDefaultIme;
end;

procedure TScrSCUD_RFID.IconFontChanged(Sender: TObject);
begin
Application.NotifyForms(CM_SYSFONTCHANGED);
if (Sender = FHintFont) and Assigned(Application) and Application.ShowHint then
begin
  Application.ShowHint:= False;
  Application.ShowHint:= True;
end;
end;

function TScrSCUD_RFID.GetDataModule(Index: Integer): TDataModule;
begin
Result:= FDataModules[Index];
end;

function TScrSCUD_RFID.GetDataModuleCount: Integer;
begin
Result:= FDataModules.Count;
end;

function TScrSCUD_RFID.GetCursors(Index: Integer): HCURSOR;
var
P: PCursorRec;
begin
Result:= 0;
if Index <> crNone then
begin
  P:= FCursorList;
  while (P <> nil) and (P^.Index <> Index) do P:= P^.Next;
  if P = nil then Result:= FDefaultCursor else Result:= P^.Handle;
end;
end;

procedure TScrSCUD_RFID.SetCursor(Value: TCursor);
```

```
var
P: TPoint;
Handle: HWND;
Code: Longint;
begin
if Value <> Cursor then
begin
  FCursor:= Value;
  if Value = crDefault then
  begin
    GetCursorPos(P);
    Handle:= WindowFromPoint(P);
    if (Handle <> 0) and
      (GetWindowThreadProcessId(Handle, nil) = GetCurrentThreadId) then
    begin
      Code:= SendMessage(Handle, WM_NCHITTEST, 0,
LongInt(PointToSmallPoint(P)));
      SendMessage(Handle, WM_SETCURSOR, Handle, MakeLong(Code, WM_MOUSEMOVE));
      Exit;
    end;
  end;
  Windows.SetCursor(Cursors[Value]);
end;
Inc(FCursorCount);
end;

procedure TScrSCUD_RFID.SetCursors(Index: Integer; Handle: HCURSOR);
begin
if Index = crDefault then
  if Handle = 0 then
    FDefaultCursor:= LoadCursor(0, IDC_ARROW)
  else
    FDefaultCursor:= Handle
else if Index <> crNone then
begin
  DeleteCursor(Index);
  if Handle <> 0 then InsertCursor(Index, Handle);
end;
end;

procedure TScrSCUD_RFID.SetHintFont(Value: TFont);
begin
FHintFont.Assign(Value);
end;

procedure TScrSCUD_RFID.SetIconFont(Value: TFont);
begin
FIconFont.Assign(Value);
end;

procedure TScrSCUD_RFID.SetMenuFont(Value: TFont);
begin
FMenuFont.Assign(Value);
end;

procedure TScrSCUD_RFID.GetMetricSettings;
var
LogFont: TLogFont;
NonClientMetrics: TNonClientMetrics;
SaveShowHint: Boolean;
begin
SaveShowHint:= False;
if Assigned(Application) then SaveShowHint:= Application.ShowHint;
try
  if Assigned(Application) then Application.ShowHint:= False;
  if SystemParametersInfo(SPI_GETICONTITLELOGFONT, SizeOf(LogFont), @LogFont, 0)
then
    FIconFont.Handle:= CreateFontIndirect(LogFont)
  else
```

```
    FIconFont.Handle:= GetStockObject(SYSTEM_FONT);
  NonClientMetrics.cbSize:= SizeOf(NonClientMetrics);
  if SystemParametersInfo(SPI_GETNONCLIENTMETRICS, 0, @NonClientMetrics, 0) then
  begin
    FHintFont.Handle:= CreateFontIndirect(NonClientMetrics.lfStatusFont);
    FMenuFont.Handle:= CreateFontIndirect(NonClientMetrics.lfMenuFont);
  end else
  begin
    FHintFont.Size:= 8;
    FMenuFont.Handle:= GetStockObject(SYSTEM_FONT);
  end;
  FHintFont.Color:= clInfoText;
  FMenuFont.Color:= clMenuText;
finally
  if Assigned(Application) then Application.ShowHint:= SaveShowHint;
end;
end;

procedure TScrSCUD_RFID.DisableAlign;
begin
Inc(FAlignLevel);
end;

procedure TScrSCUD_RFID.EnableAlign;
begin
Dec(FAlignLevel);
if (FAlignLevel = 0) and (csAlignmentNeeded in FControlState) then Realign;
end;

procedure TScrSCUD_RFID.Realign;
begin
AlignForm(nil);
end;

procedure TScrSCUD_RFID.AlignForms(AForm: TCustomForm; var Rect: TRect);
var
AlignList: TList;

function InsertBefore(C1, C2: TCustomForm; AAlign: TAlign): Boolean;
begin
  Result:= False;
  case AAlign of
    alTop: Result:= C1.Top < C2.Top;
    alBottom: Result:= (C1.Top + C1.Height) > (C2.Top + C2.Height);
    alLeft: Result:= C1.Left < C2.Left;
    alRight: Result:= (C1.Left + C1.Width) > (C2.Left + C2.Width);
  end;
end;

procedure DoPosition(Form: TCustomForm; AAlign: TAlign);
var
  NewLeft, NewTop, NewWidth, NewHeight: Integer;
begin
  with Rect do
  begin
    NewWidth:= Right - Left;
    if (NewWidth < 0) or (AAlign in [alLeft, alRight]) then
      NewWidth:= Form.Width;
    NewHeight:= Bottom - Top;
    if (NewHeight < 0) or (AAlign in [alTop, alBottom]) then
      NewHeight:= Form.Height;
    if (AAlign = alTop) and (Form.WindowState = wsMaximized) then
    begin
      NewLeft:= Form.Left;
      NewTop:= Form.Top;
      NewWidth:= GetSystemMetrics(SM_CXMAXIMIZED);
    end
    else
    begin
```

```
      NewLeft:= Left;
      NewTop:= Top;
    end;
    case AAlign of
      alTop: Inc(Top, NewHeight);
      alBottom:
        begin
          Dec(Bottom, NewHeight);
          NewTop:= Bottom;
        end;
      alLeft: Inc(Left, NewWidth);
      alRight:
        begin
          Dec(Right, NewWidth);
          NewLeft:= Right;
        end;
    end;
  end;
  Form.SetBounds(NewLeft, NewTop, NewWidth, NewHeight);
  if Form.WindowState = wsMaximized then
  begin
    Dec(NewWidth, NewLeft);
    Dec(NewHeight, NewTop);
  end;
  if (Form.Width <> NewWidth) or (Form.Height <> NewHeight) then
    with Rect do
      case AAlign of
        alTop: Dec(Top, NewHeight - Form.Height);
        alBottom: Inc(Bottom, NewHeight - Form.Height);
        alLeft: Dec(Left, NewWidth - Form.Width);
        alRight: Inc(Right, NewWidth - Form.Width);
        alClient:
          begin
            Inc(Right, NewWidth - Form.Width);
            Inc(Bottom, NewHeight - Form.Height);
          end;
      end;
end;

procedure DoAlign(AAlign: TAlign);
var
  I, J: Integer;
  Form: TCustomForm;
begin
  AlignList.Clear;
  if (AForm <> nil) and (AForm.Parent = nil) and
    not (csDesigning in AForm.ComponentState) and
    AForm.Visible and (AForm.Align = AAlign) and
    (AForm.WindowState <> wsMinimized) then
    AlignList.Add(AForm);
  for I:= 0 to CustomFormCount - 1 do
  begin
    Form:= TCustomForm(CustomForms[I]);
    if (Form.Parent = nil) and (Form.Align = AAlign) and
      not (csDesigning in Form.ComponentState) and
      Form.Visible and (Form.WindowState <> wsMinimized) then
    begin
      if Form = AForm then Continue;
      J:= 0;
      while (J < AlignList.Count) and not InsertBefore(Form,
        TCustomForm(AlignList[J]), AAlign) do Inc(J);
      AlignList.Insert(J, Form);
    end;
  end;
  for I:= 0 to AlignList.Count - 1 do
    DoPosition(TCustomForm(AlignList[I]), AAlign);
end;

function AlignWork: Boolean;
```

```
var
  I: Integer;
begin
  Result:= True;
  for I:= CustomFormCount - 1 downto 0 do
    with TCustomForm(CustomForms[I]) do
      if (Parent = nil) and not (csDesigning in ComponentState) and
        (Align <> alNone) and Visible and (WindowState <> wsMinimized) then
Exit;
  Result:= False;
end;
begin
if AlignWork then
begin
  AlignList:= TList.Create;
  try
    DoAlign(alTop);
    DoAlign(alBottom);
    DoAlign(alLeft);
    DoAlign(alRight);
    DoAlign(alClient);
  finally
    AlignList.Free;
  end;
end;
end;

procedure TScrSCUD_RFID.AlignForm(AForm: TCustomForm);
var
Rect: TRect;
begin
if FAlignLevel <> 0 then
  Include(FControlState, csAlignmentNeeded)
else
begin
  DisableAlign;
  try
    SystemParametersInfo(SPI_GETWORKAREA, 0, @Rect, 0);
    AlignForms(AForm, Rect);
  finally
    Exclude(FControlState, csAlignmentNeeded);
    EnableAlign;
  end;
end;
end;

function TScrSCUD_RFID.GetFonts: TStrings;
var
DC: HDC;
LFont: TLogFont;
begin
if FFonts = nil then
begin
  FFonts:= TStringList.Create;
  DC:= GetDC(0);
  try
    FFonts.Add('Default');
    FillChar(LFont, sizeof(LFont), 0);
    LFont.lfCharset:= DEFAULT_CHARSET;
    EnumFontFamiliesEx(DC, LFont, @EnumFontsProc, LongInt(FFonts), 0);
    TStringList(FFonts).Sorted:= TRUE;
  finally
    ReleaseDC(0, DC);
  end;
end;
Result:= FFonts;
end;

procedure TScrSCUD_RFID.ResetFonts;
```

```
begin
FreeAndNil(FFonts);
end;

function GetHint(Control: TControl): string;
begin
while Control <> nil do
  if Control.Hint = '' then
    Control:= Control.Parent
  else
  begin
    Result:= Control.Hint;
    Exit;
  end;
Result:= '';
end;

function GetHintControl(Control: TControl): TControl;
begin
Result:= Control;
while (Result <> nil) and not Result.ShowHint do Result:= Result.Parent;
if (Result <> nil) and (csDesigning in Result.ComponentState) then Result:= nil;
end;

procedure HintTimerProc(Wnd: HWnd; Msg, TimerID, SysTime: Longint); stdcall;
begin
if Application <> nil then
try
  Application.HintTimerExpired;
except
  Application.HandleException(Application);
end;
end;

var
HintThreadID: DWORD;
HintDoneEvent: THandle;

procedure HintMouseThread(Param: Integer); stdcall;
var
P: TPoint;
begin
HintThreadID:= GetCurrentThreadID;
while WaitForSingleObject(HintDoneEvent, 100) = WAIT_TIMEOUT do
begin
  if (Application <> nil) and (Application.FHintControl <> nil) then
  begin
    GetCursorPos(P);
    if FindVCLWindow(P) = nil then
      Application.CancelHint;
  end;
end;
end;

var
HintHook: HHOOK;
HintThread: THandle;

function HintGetMsgHook(nCode: Integer; wParam: Longint; var Msg: TMsg):
Longint; stdcall;
begin
Result:= CallNextHookEx(HintHook, nCode, wParam, Longint(@Msg));
if (nCode >= 0) and (Application <> nil) then Application.IsHintMsg(Msg);
end;

procedure HookHintHooks;
var
ThreadID: DWORD;
begin
```

```
if not Application.FRunning then
begin
  if HintHook = 0 then
    HintHook:= SetWindowsHookEx(WH_GETMESSAGE, @HintGetMsgHook, 0,
GetCurrentThreadID);
  if HintDoneEvent = 0 then
    HintDoneEvent:= CreateEvent(nil, False, False, nil);
  if HintThread = 0 then
    HintThread:= CreateThread(nil, 1000, @HintMouseThread, nil, 0, ThreadID);
end;
end;

procedure UnhookHintHooks;
begin
if HintHook <> 0 then UnhookWindowsHookEx(HintHook);
HintHook:= 0;
if HintThread <> 0 then
begin
  SetEvent(HintDoneEvent);
  if GetCurrentThreadId <> HintThreadID then
    WaitForSingleObject(HintThread, INFINITE);
  CloseHandle(HintThread);
  HintThread:= 0;
end;
end;

function GetAnimation: Boolean;
var
Info: TAnimationInfo;
begin
Info.cbSize:= SizeOf(TAnimationInfo);
if SystemParametersInfo(SPI_GETANIMATION, SizeOf(Info), @Info, 0) then
  Result:= Info.iMinAnimate <> 0 else
  Result:= False;
end;

procedure SetAnimation(Value: Boolean);
var
Info: TAnimationInfo;
begin
Info.cbSize:= SizeOf(TAnimationInfo);
BOOL(Info.iMinAnimate):= Value;
SystemParametersInfo(SPI_SETANIMATION, SizeOf(Info), @Info, 0);
end;

procedure ShowWinNoAnimate(Handle: HWnd; CmdShow: Integer);
var
Animation: Boolean;
begin
Animation:= GetAnimation;
if Animation then SetAnimation(False);
ShowWindow(Handle, CmdShow);
if Animation then SetAnimation(True);
end;

function TScrSCUD_RFID.GetDesktopRect: TRect;
begin
Result:= Bounds(DesktopLeft, DesktopTop, DesktopWidth, DesktopHeight);
end;

function TScrSCUD_RFID.GetWorkAreaHeight: Integer;
begin
with WorkAreaRect do
  Result:= Bottom - Top;
end;

function TScrSCUD_RFID.GetWorkAreaLeft: Integer;
begin
Result:= WorkAreaRect.Left;
```

```
end;

function TScrSCUD_RFID.GetWorkAreaRect: TRect;
begin
SystemParametersInfo(SPI_GETWORKAREA, 0, @Result, 0);
end;

function TScrSCUD_RFID.GetWorkAreaTop: Integer;
begin
Result:= WorkAreaRect.Top;
end;

function TScrSCUD_RFID.GetWorkAreaWidth: Integer;
begin
with WorkAreaRect do
  Result:= Right - Left;
end;

const
MonitorDefaultFlags: array[TMonitorDefaultTo] of DWORD =
(MONITOR_DEFAULTTONEAREST,
  MONITOR_DEFAULTTONULL, MONITOR_DEFAULTTOPRIMARY);

function TScrSCUD_RFID.MonitorFromPoint(const Point: TPoint;
MonitorDefault: TMonitorDefaultTo): TMonitor;
begin
Result:= FindMonitor(MultiMon.MonitorFromPoint(Point,
  MonitorDefaultFlags[MonitorDefault]));
end;

function TScrSCUD_RFID.MonitorFromRect(const Rect: TRect;
MonitorDefault: TMonitorDefaultTo): TMonitor;
begin
Result:= FindMonitor(MultiMon.MonitorFromRect(@Rect,
  MonitorDefaultFlags[MonitorDefault]));
end;

function TScrSCUD_RFID.MonitorFromWindow(const Handle: THandle;
MonitorDefault: TMonitorDefaultTo): TMonitor;
begin
Result:= FindMonitor(MultiMon.MonitorFromWindow(Handle,
  MonitorDefaultFlags[MonitorDefault]));
end;

var
WindowClass: TWndClass = (
  style: 0;
  lpfnWndProc: @DefWindowProc;
  cbClsExtra: 0;
  cbWndExtra: 0;
  hInstance: 0;
  hIcon: 0;
  hCursor: 0;
  hbrBackground: 0;
  lpszMenuName: nil;
  lpszClassName: 'TSCUD_RFIDApp');

constructor TSCUD_RFIDApp.Create(AOwner: TComponent);
var
P: PChar;
ModuleName: array[0..255] of Char;
begin
FTopMostList:= TList.Create;
FWindowHooks:= TList.Create;
FHintControl:= nil;
FHintWindow:= nil;
FHintColor:= DefHintColor;
FHintPause:= DefHintPause;
FHintShortCuts:= True;
```

```
FHintShortPause:= DefHintShortPause;
FHintHidePause:= DefHintHidePause;
FShowHint:= False;
FActive:= True;
FAutoDragDocking:= True;
FIcon:= TIcon.Create;
FIcon.Handle:= LoadIcon(MainInstance, 'MAINICON');
FIcon.OnChange:= IconChanged;
GetModuleFileName(MainInstance, ModuleName, SizeOf(ModuleName));
OemToAnsi(ModuleName, ModuleName);
P:= AnsiStrRScan(ModuleName, '\');
if P <> nil then StrCopy(ModuleName, P + 1);
P:= AnsiStrScan(ModuleName, '.');
if P <> nil then P^:= #0;
AnsiLower(ModuleName + 1);
FTitle:= ModuleName;
if not IsLibrary then CreateHandle;
UpdateFormatSettings:= True;
UpdateMetricSettings:= True;
FShowMainForm:= True;
FAllowTesting:= True;
FTestLib:= 0;
ValidateHelpSystem;
HookSynchronizeWakeup;
end;

destructor TSCUD_RFIDApp.Destroy;
type // Власті типи даних
TExceptionEvent = procedure (E: Exception) of object;
var
P: TNotifyEvent;
E: TExceptionEvent;
begin
UnhookSynchronizeWakeup;
P:= HandleException;
if @P = @Classes.ApplicationHandleException then
  Classes.ApplicationHandleException:= nil;
E:= ShowException;
if @E = @Classes.ApplicationShowException then
  Classes.ApplicationShowException:= nil;
if FTestLib <> 0 then
  FreeLibrary(FTestLib);
FActive:= False;
CancelHint;
ShowHint:= False;
inherited Destroy;
UnhookMainWindow(CheckIniChange);
if (FHandle <> 0) and FHandleCreated then
begin
  if NewStyleControls then SendMessage(FHandle, WM_SETICON, 1, 0);
  DestroyWindow(FHandle);
end;
if FHelpSystem <> nil then FHelpSystem:= nil;
if FObjectInstance <> nil then WinUtils.FreeObjectInstance(FObjectInstance);
if FObjectInstance <> nil then Classes.FreeObjectInstance(FObjectInstance);
FWindowHooks.Free;
FTopMostList.Free;
FIcon.Free;
end;

procedure TSCUD_RFIDApp.CreateHandle;
var
TempClass: TWndClass;
SysMenu: HMenu;
begin
if not FHandleCreated
  and not IsConsole then
  then
begin
```

```
  FObjectInstance:= WinUtils.MakeObjectInstance(WndProc);
  FObjectInstance:= Classes.MakeObjectInstance(WndProc);
  WindowClass.lpfnWndProc:= @DefWindowProc;
  if not GetClassInfo(HInstance, WindowClass.lpszClassName, TempClass) then
  begin
    WindowClass.hInstance:= HInstance;
    if Windows.RegisterClass(WindowClass) = 0 then
      raise EOutOfResources.Create(SWindowClass);
  end;
  FHandle:= CreateWindow(WindowClass.lpszClassName, PChar(FTitle),
    WS_POPUP or WS_CAPTION or WS_CLIPSIBLINGS or WS_SYSMENU
    or WS_MINIMIZEBOX,
    GetSystemMetrics(SM_CXSCREEN) div 2,
    GetSystemMetrics(SM_CYSCREEN) div 2,
    0, 0, 0, 0, HInstance, nil);
  FTitle:= '';
  FHandleCreated:= True;
  SetWindowLong(FHandle, GWL_WNDPROC, Longint(FObjectInstance));
  if NewStyleControls then
  begin
    SendMessage(FHandle, WM_SETICON, 1, GetIconHandle);
    SetClassLong(FHandle, GCL_HICON, GetIconHandle);
  end;
  SysMenu:= GetSystemMenu(FHandle, False);
  DeleteMenu(SysMenu, SC_MAXIMIZE, MF_BYCOMMAND);
  DeleteMenu(SysMenu, SC_SIZE, MF_BYCOMMAND);
  if NewStyleControls then DeleteMenu(SysMenu, SC_MOVE, MF_BYCOMMAND);
end;
end;

procedure TSCUD_RFIDApp.ControlDestroyed(Control: TControl);
begin
if FMainForm = Control then FMainForm:= nil;
if FMouseControl = Control then FMouseControl:= nil;
if Screen.FActiveControl = Control then Screen.FActiveControl:= nil;
if Screen.FActiveCustomForm = Control then
begin
  Screen.FActiveCustomForm:= nil;
  Screen.FActiveForm:= nil;
end;
if Screen.FFocusedForm = Control then Screen.FFocusedForm:= nil;
if FHintControl = Control then FHintControl:= nil;
Screen.UpdateLastActive;
end;

type
PTopMostEnumInfo = ^TTopMostEnumInfo;
TTopMostEnumInfo = record
  TopWindow: HWND;
  IncludeMain: Boolean;
end;

function GetTopMostWindows(Handle: HWND; Info: Pointer): BOOL; stdcall;
begin
Result:= True;
if GetWindow(Handle, GW_OWNER) = Application.Handle then
  if (GetWindowLong(Handle, GWL_EXSTYLE) and WS_EX_TOPMOST <> 0) and
    ((Application.MainForm = nil) or PTopMostEnumInfo(Info)^.IncludeMain or
    (Handle <> Application.MainForm.Handle)) then
    Application.FTopMostList.Add(Pointer(Handle))
  else
  begin
    PTopMostEnumInfo(Info)^.TopWindow:= Handle;
    Result:= False;
  end;
end;

procedure TSCUD_RFIDApp.DoNormalizeTopMosts(IncludeMain: Boolean);
var
```

```
I: Integer;
Info: TTopMostEnumInfo;
begin
if Application.Handle <> 0 then
begin
  if FTopMostLevel = 0 then
  begin
    Info.TopWindow:= Handle;
    Info.IncludeMain:= IncludeMain;
    EnumWindows(@GetTopMostWindows, Longint(@Info));
    if FTopMostList.Count <> 0 then
    begin
      Info.TopWindow:= GetWindow(Info.TopWindow, GW_HWNDPREV);
      if GetWindowLong(Info.TopWindow, GWL_EXSTYLE) and WS_EX_TOPMOST <> 0 then
        Info.TopWindow:= HWND_NOTOPMOST;
      for I:= FTopMostList.Count - 1 downto 0 do
        SetWindowPos(HWND(FTopMostList[I]), Info.TopWindow, 0, 0, 0, 0,
          SWP_NOMOVE or SWP_NOSIZE or SWP_NOACTIVATE or SWP_NOOWNERZORDER);
    end;
  end;
  Inc(FTopMostLevel);
end;
end;

procedure TSCUD_RFIDApp.ModalStarted;
begin
Inc(FModalLevel);
if (FModalLevel = 1) and Assigned(FOnModalBegin) then
  FOnModalBegin(Self);
end;

procedure TSCUD_RFIDApp.ModalFinished;
begin
Dec(FModalLevel);
if (FModalLevel = 0) and Assigned(FOnModalEnd) then
  FOnModalEnd(Self);
end;

procedure TSCUD_RFIDApp.NormalizeTopMosts;
begin
DoNormalizeTopMosts(False);
end;

procedure TSCUD_RFIDApp.NormalizeAllTopMosts;
begin
DoNormalizeTopMosts(True);
end;

procedure TSCUD_RFIDApp.RestoreTopMosts;
var
I: Integer;
begin
if (Application.Handle <> 0) and (FTopMostLevel > 0) then
begin
  Dec(FTopMostLevel);
  if FTopMostLevel = 0 then
  begin
    for I:= FTopMostList.Count - 1 downto 0 do
      SetWindowPos(HWND(FTopMostList[I]), HWND_TOPMOST, 0, 0, 0, 0,
        SWP_NOMOVE or SWP_NOSIZE or SWP_NOACTIVATE or SWP_NOOWNERZORDER);
    FTopMostList.Clear;
  end;
end;
end;

function TSCUD_RFIDApp.IsRightToLeft: Boolean;
begin
Result:= SysLocale.MiddleEast and (FBiDiMode <> bdLeftToRight);
end;
```

```
function TSCUD_RFIDApp.UseRightToLeftReading: Boolean;
begin
Result:= SysLocale.MiddleEast and (FBiDiMode <> bdLeftToRight);
end;

function TSCUD_RFIDApp.UseRightToLeftAlignment: Boolean;
begin
Result:= SysLocale.MiddleEast and (FBiDiMode = bdRightToLeft);
end;

function TSCUD_RFIDApp.UseRightToLeftScrollBar: Boolean;
begin
Result:= SysLocale.MiddleEast and
          (FBiDiMode in [bdRightToLeft, bdRightToLeftNoAlign]);
end;

function TSCUD_RFIDApp.CheckIniChange(var Message: TMessage): Boolean;
begin
Result:= False;
if (Message.Msg = RM_TaskbarCreated) or
   (Message.Msg = WM_WININICHANGE) then
begin
  if UpdateFormatSettings then
  begin
    SetThreadLocale(LOCALE_USER_DEFAULT);
    GetFormatSettings;
  end;
  if UpdateMetricSettings then
    Screen.GetMetricSettings;

  if Message.Msg = RM_TaskbarCreated then
  begin
    Screen.ResetFonts;
  end;
end;
end;

procedure TSCUD_RFIDApp.SettingChange(var Message: TWMSettingChange);
begin
if Assigned(FOnSettingChange) then
  with Message do
    FOnSettingChange(Self, Flag, Section, Result);
end;


procedure TSCUD_RFIDApp.WndProc(var Message: TMessage);
type // Власті типи даних
TInitTestLibrary = function(Size: DWord; PAutoClassInfo: Pointer): Boolean;
stdcall;

var
I: Integer;
SaveFocus, TopWindow: HWnd;
InitTestLibrary: TInitTestLibrary;

procedure Default;
begin
  with Message do
    Result:= DefWindowProc(FHandle, Msg, WParam, LParam);
end;

procedure DrawAppIcon;
var
  DC: HDC;
  PS: TPaintStruct;
begin
  with Message do
  begin
```

```
    DC:= BeginPaint(FHandle, PS);
    DrawIcon(DC, 0, 0, GetIconHandle);
    EndPaint(FHandle, PS);
  end;
end;

begin
try
  Message.Result:= 0;
  for I:= 0 to FWindowHooks.Count - 1 do
    if TWindowHook(FWindowHooks[I]^)(Message) then Exit;
  CheckIniChange(Message);
  with Message do
    case Msg of
      WM_SYSCOMMAND:
        case WParam and $FFF0 of
          SC_MINIMIZE: Minimize;
          SC_RESTORE: Restore;
        else
          Default;
        end;
      WM_CLOSE:
        if MainForm <> nil then MainForm.Close;
      WM_PAINT:
        if IsIconic(FHandle) then DrawAppIcon else Default;
      WM_ERASEBKGND:
        begin
          Message.Msg:= WM_ICONERASEBKGND;
          Default;
        end;
      WM_QUERYDRAGICON:
        Result:= GetIconHandle;
      WM_SETFOCUS:
        begin
          PostMessage(FHandle, CM_ENTER, 0, 0);
          Default;
        end;
      WM_ACTIVATEAPP:
        begin
          Default;
          FActive:= TWMActivateApp(Message).Active;
          if TWMActivateApp(Message).Active then
          begin
            RestoreTopMosts;
            PostMessage(FHandle, CM_ACTIVATE, 0, 0)
          end
          else
          begin
            NormalizeTopMosts;
            PostMessage(FHandle, CM_DEACTIVATE, 0, 0);
          end;
        end;
      WM_ENABLE:
        if TWMEnable(Message).Enabled then
        begin
          RestoreTopMosts;
          if FWindowList <> nil then
          begin
            EnableTaskWindows(FWindowList);
            FWindowList:= nil;
          end;
          Default;
        end else
        begin
          Default;
          if FWindowList = nil then
            FWindowList:= DisableTaskWindows(Handle);
          NormalizeAllTopMosts;
        end;
```

```
      WM_CTLCOLORMSGBOX..WM_CTLCOLORSTATIC:
        Result:= SendMessage(LParam, CN_BASE + Msg, WParam, LParam);
      WM_ENDSESSION: if TWMEndSession(Message).EndSession then FTerminate:=
True;
      WM_COPYDATA:
        if (PCopyDataStruct(Message.lParam)^.dwData = DWORD($DE534454)) and
          (FAllowTesting) then
          if FTestLib = 0 then
          begin
            if FTestLib <> 0 then
            begin
              Result:= 0;
              @InitTestLibrary:= GetProcAddress(FTestLib, 'RegisterAutomation');
              if @InitTestLibrary <> nil then
                InitTestLibrary(PCopyDataStruct(Message.lParam)^.cbData,
                  PCopyDataStruct(Message.lParam)^.lpData);
            end
            else
            begin
              Result:= GetLastError;
              FTestLib:= 0;
            end;
          end
          else
            Result:= 0;
      CM_ACTIONEXECUTE, CM_ACTIONUPDATE:
        Message.Result:= Ord(DispatchAction(Message.Msg,
TBasicAction(Message.LParam)));
      CM_APPKEYDOWN:
        if IsShortCut(TWMKey(Message)) then Result:= 1;
      CM_APPSYSCOMMAND:
        if MainForm <> nil then
          with MainForm do
            if (Handle <> 0) and IsWindowEnabled(Handle) and
              IsWindowVisible(Handle) then
            begin
              FocusMessages:= False;
              SaveFocus:= GetFocus;
              Windows.SetFocus(Handle);
              Perform(WM_SYSCOMMAND, WParam, LParam);
              Windows.SetFocus(SaveFocus);
              FocusMessages:= True;
              Result:= 1;
            end;
      CM_ACTIVATE:
        if Assigned(FOnActivate) then FOnActivate(Self);
      CM_DEACTIVATE:
        if Assigned(FOnDeactivate) then FOnDeactivate(Self);
      CM_ENTER:
        if not IsIconic(FHandle) and (GetFocus = FHandle) then
        begin
          TopWindow:= FindTopMostWindow(0);
          if TopWindow <> 0 then Windows.SetFocus(TopWindow);
        end;
      WM_HELP,
     CM_INVOKEHELP: InvokeHelp(WParam, LParam);
      CM_WINDOWHOOK:
        if wParam = 0 then
          HookMainWindow(TWindowHook(Pointer(LParam)^)) else
          UnhookMainWindow(TWindowHook(Pointer(LParam)^));
      CM_DIALOGHANDLE:
        if wParam = 1 then
          Result:= FDialogHandle
        else
          FDialogHandle:= lParam;
      WM_SETTINGCHANGE:
        begin
          Mouse.SettingChanged(wParam);
          SettingChange(TWMSettingChange(Message));
```

```
          Default;
        end;
      WM_FONTCHANGE:
        begin
          Screen.ResetFonts;
          Default;
        end;
      WM_THEMECHANGED:
        if ThemeServices.ThemesEnabled then
          ThemeServices.ApplyThemeChange;
      WM_NULL:
        CheckSynchronize;
    else
      Default;
    end;
except
  HandleException(Self);
end;
end;

function TSCUD_RFIDApp.GetIconHandle: HICON;
begin
Result:= FIcon.Handle;
if Result = 0 then Result:= LoadIcon(0, IDI_APPLICATION);
end;

procedure TSCUD_RFIDApp.Minimize;
begin
if not IsIconic(FHandle) then
begin
  NormalizeTopMosts;
  SetActiveWindow(FHandle);
  if (MainForm <> nil) and (ShowMainForm or MainForm.Visible)
    and IsWindowEnabled(MainForm.Handle) then
  begin
    SetWindowPos(FHandle, MainForm.Handle, MainForm.Left, MainForm.Top,
      MainForm.Width, 0, SWP_SHOWWINDOW);
    DefWindowProc(FHandle, WM_SYSCOMMAND, SC_MINIMIZE, 0);
  end else
    ShowWinNoAnimate(FHandle, SW_MINIMIZE);
  if Assigned(FOnMinimize) then FOnMinimize(Self);
end;
end;

procedure TSCUD_RFIDApp.Restore;
begin
if IsIconic(FHandle) then
begin
  SetActiveWindow(FHandle);
  if (MainForm <> nil) and (ShowMainForm or MainForm.Visible)
    and IsWindowEnabled(MainForm.Handle) then
    DefWindowProc(FHandle, WM_SYSCOMMAND, SC_RESTORE, 0)
  else ShowWinNoAnimate(FHandle, SW_RESTORE);
  SetWindowPos(FHandle, 0, GetSystemMetrics(SM_CXSCREEN) div 2,
    GetSystemMetrics(SM_CYSCREEN) div 2, 0, 0, SWP_SHOWWINDOW);
  if (FMainForm <> nil) and (FMainForm.FWindowState = wsMinimized) and
    not FMainForm.Visible then
  begin
    FMainForm.WindowState:= wsNormal;
    FMainForm.Show;
  end;
  RestoreTopMosts;
  if Screen.ActiveControl <> nil then
    Windows.SetFocus(Screen.ActiveControl.Handle);
  if Assigned(FOnRestore) then FOnRestore(Self);
end;
end;

procedure TSCUD_RFIDApp.BringToFront;
```

```
var
TopWindow: HWnd;
begin
if Handle <> 0 then
begin
  TopWindow:= GetLastActivePopup(Handle);
  if (TopWindow <> 0) and (TopWindow <> Handle) and
    IsWindowVisible(TopWindow) and IsWindowEnabled(TopWindow) then
    SetForegroundWindow(TopWindow);
end;
end;

function TSCUD_RFIDApp.GetTitle: string;
var
Buffer: array[0..255] of Char;
begin
if FHandleCreated then
  SetString(Result, Buffer, GetWindowText(FHandle, Buffer,
    SizeOf(Buffer))) else
  Result:= FTitle;
end;

procedure TSCUD_RFIDApp.SetIcon(Value: TIcon);
begin
FIcon.Assign(Value);
end;

procedure TSCUD_RFIDApp.SetBiDiMode(Value: TBiDiMode);
var
Loop: Integer;
begin
if FBiDiMode <> Value then
begin
  FBiDiMode:= Value;
  with Screen do
    for Loop:= 0 to FormCount-1 do
      Forms[Loop].Perform(CM_PARENTBIDIMODECHANGED, 0, 0);
end;
end;

procedure TSCUD_RFIDApp.SetTitle(const Value: string);
begin
if FHandleCreated then
begin
  if (GetTitle <> Value) or (FTitle <> '') then
  begin
    SetWindowText(FHandle, PChar(Value));
    FTitle:= '';
  end;
end
else
  FTitle:= Value;
end;

procedure TSCUD_RFIDApp.SetHandle(Value: HWnd);
begin
if not FHandleCreated and (Value <> FHandle) then
begin
  if FHandle <> 0 then UnhookMainWindow(CheckIniChange);
  FHandle:= Value;
  if FHandle <> 0 then HookMainWindow(CheckIniChange);
end;
end;

function TSCUD_RFIDApp.IsDlgMsg(var Msg: TMsg): Boolean;
begin
Result:= False;
if FDialogHandle <> 0 then
  Result:= IsDialogMessage(FDialogHandle, Msg);
```

```
end;

function TSCUD_RFIDApp.IsMDIMsg(var Msg: TMsg): Boolean;
begin
Result:= False;
if (MainForm <> nil) and (MainForm.FormStyle = fsMDIForm) and
   (Screen.ActiveForm <> nil) and (Screen.ActiveForm.FormStyle = fsMDIChild)
then
  Result:= TranslateMDISysAccel(MainForm.ClientHandle, Msg);
end;

function TSCUD_RFIDApp.IsKeyMsg(var Msg: TMsg): Boolean;
var
Wnd: HWND;
begin
Result:= False;
with Msg do
  if (Message >= WM_KEYFIRST) and (Message <= WM_KEYLAST) then
  begin
    Wnd:= GetCapture;
    if Wnd = 0 then
    begin
      Wnd:= HWnd;
      if (MainForm <> nil) and (Wnd = MainForm.ClientHandle) then
        Wnd:= MainForm.Handle
      else
      begin
        while (FindControl(Wnd) = nil) and (Wnd <> 0) do
          Wnd:= GetParent(Wnd);
        if Wnd = 0 then Wnd:= HWnd;
      end;
      if SendMessage(Wnd, CN_BASE + Message, WParam, LParam) <> 0 then
        Result:= True;
    end
    else if (LongWord(GetWindowLong(Wnd, GWL_HINSTANCE)) = HInstance) then
    begin
      if SendMessage(Wnd, CN_BASE + Message, WParam, LParam) <> 0 then
        Result:= True;
    end;
  end;
end;

function TSCUD_RFIDApp.IsHintMsg(var Msg: TMsg): Boolean;
begin
Result:= False;
if (FHintWindow <> nil) and FHintWindow.IsHintMsg(Msg) then
  CancelHint;
end;

function TSCUD_RFIDApp.IsShortCut(var Message: TWMKey): Boolean;
begin
Result:= False;
if Assigned(FOnShortCut) then FOnShortCut(Message, Result);
Result:= Result or (MainForm <> nil) and IsWindowEnabled(MainForm.Handle) and
  MainForm.IsShortCut(TWMKey(Message))
end;

function TSCUD_RFIDApp.ProcessMessage(var Msg: TMsg): Boolean;
var
Handled: Boolean;
begin
Result:= False;
if PeekMessage(Msg, 0, 0, 0, PM_REMOVE) then
begin
  Result:= True;
  if Msg.Message <> WM_QUIT then
  begin
    Handled:= False;
    if Assigned(FOnMessage) then FOnMessage(Msg, Handled);
```

```
    if not IsHintMsg(Msg) and not Handled and not IsMDIMsg(Msg) and
      not IsKeyMsg(Msg) and not IsDlgMsg(Msg) then
    begin
      TranslateMessage(Msg);
      DispatchMessage(Msg);
    end;
  end
  else
    FTerminate:= True;
end;
end;

procedure TSCUD_RFIDApp.HandleMessage;
var
Msg: TMsg;
begin
if not ProcessMessage(Msg) then Idle(Msg);
end;

procedure TSCUD_RFIDApp.HookMainWindow(Hook: TWindowHook);
var
WindowHook: ^TWindowHook;
begin
if not FHandleCreated then
begin
  if FHandle <> 0 then
    SendMessage(FHandle, CM_WINDOWHOOK, 0, Longint(@@Hook));
end else
begin
  FWindowHooks.Expand;
  New(WindowHook);
  WindowHook^:= Hook;
  FWindowHooks.Add(WindowHook);
end;
end;

procedure TSCUD_RFIDApp.UnhookMainWindow(Hook: TWindowHook);
var
I: Integer;
WindowHook: ^TWindowHook;
begin
if not FHandleCreated then
begin
  if FHandle <> 0 then
    SendMessage(FHandle, CM_WINDOWHOOK, 1, Longint(@@Hook));
end else
  for I:= 0 to FWindowHooks.Count - 1 do
  begin
    WindowHook:= FWindowHooks[I];
    if (TMethod(WindowHook^).Code = TMethod(Hook).Code) and
      (TMethod(WindowHook^).Data = TMethod(Hook).Data) then
    begin
      Dispose(WindowHook);
      FWindowHooks.Delete(I);
      Break;
    end;
  end;
end;

procedure TSCUD_RFIDApp.Initialize;
begin
if InitProc <> nil then TProcedure(InitProc);
end;

procedure TSCUD_RFIDApp.CreateForm(InstanceClass: TComponentClass; var
Reference);
var
Instance: TComponent;
begin
```

```
Instance:= TComponent(InstanceClass.NewInstance);
TComponent(Reference):= Instance;
try
  Instance.Create(Self);
except
  TComponent(Reference):= nil;
  raise;
end;
if (FMainForm = nil) and (Instance is TForm) then
begin
  TForm(Instance).HandleNeeded;
  FMainForm:= TForm(Instance);
end;
end;

procedure TSCUD_RFIDApp.Run;
begin
FRunning:= True;
try
  AddExitProc(DoneApplication);
  if FMainForm <> nil then
  begin
    case CmdShow of
      SW_SHOWMINNOACTIVE: FMainForm.FWindowState:= wsMinimized;
      SW_SHOWMAXIMIZED: MainForm.WindowState:= wsMaximized;
    end;
    if FShowMainForm then
      if FMainForm.FWindowState = wsMinimized then
        Minimize else
        FMainForm.Visible:= True;
    repeat
      try
        HandleMessage;
      except
        HandleException(Self);
      end;
    until Terminated;
  end;
finally
  FRunning:= False;
end;
end;

procedure TSCUD_RFIDApp.Terminate;
begin
if CallTerminateProcs then PostQuitMessage(0);
end;

procedure TSCUD_RFIDApp.HandleException(Sender: TObject);
begin
if GetCapture <> 0 then SendMessage(GetCapture, WM_CANCELMODE, 0, 0);
if ExceptObject is Exception then
begin
  if not (ExceptObject is EAbort) then
    if Assigned(FOnException) then
      FOnException(Sender, Exception(ExceptObject))
    else
      ShowException(Exception(ExceptObject));
end else
  SysUtils.ShowException(ExceptObject, ExceptAddr);
end;

function TSCUD_RFIDApp.MessageBox(const Text, Caption: PChar; Flags: Longint):
Integer;
var
ActiveWindow: HWnd;
WindowList: Pointer;
MBMonitor, AppMonitor: HMonitor;
MonInfo: TMonitorInfo;
```

```
Rect: TRect;
FocusState: TFocusState;
begin
ActiveWindow:= GetActiveWindow;
MBMonitor:= MonitorFromWindow(ActiveWindow, MONITOR_DEFAULTTONEAREST);
AppMonitor:= MonitorFromWindow(Handle, MONITOR_DEFAULTTONEAREST);
if MBMonitor <> AppMonitor then
begin
  MonInfo.cbSize:= Sizeof(TMonitorInfo);
  GetMonitorInfo(MBMonitor, @MonInfo);
  GetWindowRect(Handle, Rect);
  SetWindowPos(Handle, 0,
MonInfo.rcMonitor.Left+((MonInfo.rcMonitor.Right - MonInfo.rcMonitor.Left) div
2),
MonInfo.rcMonitor.Top+((MonInfo.rcMonitor.Bottom - MonInfo.rcMonitor.Top) div
2),
    0, 0, SWP_NOACTIVATE or SWP_NOREDRAW or SWP_NOSIZE or SWP_NOZORDER);
end;
WindowList:= DisableTaskWindows(0);
FocusState:= SaveFocusState;
if UseRightToLeftReading then Flags:= Flags or MB_RTLREADING;
try
  Result:= Windows.MessageBox(Handle, Text, Caption, Flags);
finally
  if MBMonitor <> AppMonitor then
    SetWindowPos(Handle, 0,
      Rect.Left + ((Rect.Right - Rect.Left) div 2),
      Rect.Top + ((Rect.Bottom - Rect.Top) div 2),
      0, 0, SWP_NOACTIVATE or SWP_NOREDRAW or SWP_NOSIZE or SWP_NOZORDER);
  EnableTaskWindows(WindowList);
  SetActiveWindow(ActiveWindow);
  RestoreFocusState(FocusState);
end;
end;

procedure TSCUD_RFIDApp.ShowException(E: Exception);
var
Msg: string;
begin
Msg:= E.Message;
if (Msg <> '') and (AnsiLastChar(Msg) > '.') then Msg:= Msg + '.';
MessageBox(PChar(Msg), PChar(GetTitle), MB_OK + MB_ICONSTOP);
end;

function TSCUD_RFIDApp.InvokeHelp(Command: Word; Data: Longint): Boolean;
var
CallHelp: Boolean;
HelpHandle: HWND;
ActiveForm: TCustomForm;
begin
Result:= False;
CallHelp:= True;
ActiveForm:= Screen.ActiveCustomForm;

if Assigned(ActiveForm) and Assigned(ActiveForm.FOnHelp) then
  Result:= ActiveForm.FOnHelp(Command, Data, CallHelp)
else if Assigned(FOnHelp) then
  Result:= FOnHelp(Command, Data, CallHelp);

if Assigned(ActiveForm) then
begin
  if csDesigning in ActiveForm.ComponentState then CallHelp:= False;
  if (ActiveForm.TabOrder = -1) and (ActiveForm.Visible = false) and
     (not Assigned(ActiveForm.ActiveControl)) then CallHelp:= False;
end;

if CallHelp and (not Result) then
  if Assigned(ActiveForm) and ActiveForm.HandleAllocated and
(ActiveForm.FHelpFile <> '') then
```

```
  begin
    HelpHandle:= ActiveForm.Handle;
    if ValidateHelpSystem then
      Result:= HelpSystem.Hook(Longint(HelpHandle), ActiveForm.FHelpFile,
Command, Data);
  end
  else
  if FHelpFile <> '' then
  begin
    HelpHandle:= Handle;
    if FMainForm <> nil then HelpHandle:= FMainForm.Handle;
    if ValidateHelpSystem then Result:= HelpSystem.Hook(Longint(HelpHandle),
FHelpFile, Command, Data);
  end else
    if not FHandleCreated then
      PostMessage(FHandle, CM_INVOKEHELP, Command, Data);
end;

function TSCUD_RFIDApp.HelpKeyword(const Keyword: String): Boolean;
begin
Result:= true;
if ValidateHelpSystem then
  HelpSystem.ShowHelp(Keyword, GetCurrentHelpFile)
else Result:= false;
end;

function TSCUD_RFIDApp.HelpContext(Context: THelpContext): Boolean;
begin
Result:= true;
if ValidateHelpSystem then
  HelpSystem.ShowContextHelp(Context, GetCurrentHelpFile)
else Result:= false;
end;

function TSCUD_RFIDApp.HelpCommand(Command: Integer; Data: Longint): Boolean;
begin
Result:= InvokeHelp(Command, Data);
end;

function TSCUD_RFIDApp.HelpJump(const JumpID: string): Boolean;
begin
Result:= true;
if ValidateHelpSystem then
  HelpSystem.ShowTopicHelp(JumpID, GetCurrentHelpFile)
else Result:= false;
end;

function TSCUD_RFIDApp.GetExeName: string;
begin
Result:= ParamStr(0);
end;

procedure TSCUD_RFIDApp.SetShowHint(Value: Boolean);
begin
if FShowHint <> Value then
begin
  FShowHint:= Value;
  if FShowHint then
  begin
    FHintWindow:= HintWindowClass.Create(Self);
    FHintWindow.Color:= FHintColor;
  end else
  begin
    FHintWindow.Free;
    FHintWindow:= nil;
  end;
end;
end;
```

```
procedure TSCUD_RFIDApp.SetHintColor(Value: TColor);
begin
if FHintColor <> Value then
begin
  FHintColor:= Value;
  if FHintWindow <> nil then
    FHintWindow.Color:= FHintColor;
end;
end;

procedure TSCUD_RFIDApp.DoActionIdle;
var
I: Integer;
begin
for I:= 0 to Screen.CustomFormCount - 1 do
  with Screen.CustomForms[I] do
    if HandleAllocated and IsWindowVisible(Handle) and
      IsWindowEnabled(Handle) then
      UpdateActions;
end;

function TSCUD_RFIDApp.DoMouseIdle: TControl;
var
CaptureControl: TControl;
P: TPoint;
begin
GetCursorPos(P);
Result:= FindDragTarget(P, True);
CaptureControl:= GetCaptureControl;
if FMouseControl <> Result then
begin
  if ((FMouseControl <> nil) and (CaptureControl = nil)) or
    ((CaptureControl <> nil) and (FMouseControl = CaptureControl)) then
    FMouseControl.Perform(CM_MOUSELEAVE, 0, 0);
  FMouseControl:= Result;
  if ((FMouseControl <> nil) and (CaptureControl = nil)) or
    ((CaptureControl <> nil) and (FMouseControl = CaptureControl)) then
    FMouseControl.Perform(CM_MOUSEENTER, 0, 0);
end;
end;

procedure TSCUD_RFIDApp.Idle(const Msg: TMsg);
var
Control: TControl;
Done: Boolean;
begin
Control:= DoMouseIdle;
if FShowHint and (FMouseControl = nil) then
  CancelHint;
Application.Hint:= GetLongHint(GetHint(Control));
Done:= True;
try
  if Assigned(FOnIdle) then FOnIdle(Self, Done);
  if Done then DoActionIdle;
except
  HandleException(Self);
end;
if (GetCurrentThreadID = MainThreadID) and CheckSynchronize then
if (Libc.GetCurrentThreadID = MainThreadID) and CheckSynchronize then
  Done:= False;
if Done then WaitMessage;
end;

function TSCUD_RFIDApp.ExecuteAction(Action: TBasicAction): Boolean;
begin
Result:= False;
if Assigned(FOnActionExecute) then FOnActionExecute(Action, Result);
end;
```

```
function TSCUD_RFIDApp.UpdateAction(Action: TBasicAction): Boolean;
begin
Result:= False;
if Assigned(FOnActionUpdate) then FOnActionUpdate(Action, Result);
end;

procedure TSCUD_RFIDApp.WakeMainThread(Sender: TObject);
begin
PostMessage(Handle, WM_NULL, 0, 0);
end;

procedure TSCUD_RFIDApp.HookSynchronizeWakeup;
begin
Classes.WakeMainThread:= WakeMainThread;
end;

procedure TSCUD_RFIDApp.UnhookSynchronizeWakeup;
begin
Classes.WakeMainThread:= nil;
end;

end.
```