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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID”

Виконав здобувач вищої освіти
IV курсу, групи КБ-20
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Колесник Д.С.
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *125 “*Кібербезпека*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

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

Колесник Дарині Сергіївні
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID*

2. Керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 135-02 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 кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID*

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 кібербезпеки в промислову експлуатацію.

6. Висновки

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

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*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
_____ Смірнов С.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

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

АНОТАЦІЯ

Колесник Д.С. Програмне забезпечення системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID.

Метою розробки є програмне забезпечення системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID.

Результат роботи – програмна реалізація системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID.

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 RAD Studio Delphi 10.

Ключові слова: кібербезпека, контроль та керування доступом, смарт-карти, RFID

ABSTRACT

Kolesnyk D.S. Cybersecurity system software for access control and management using RFID smart cards. 125 Cyber security. 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 cyber security system of control and access control using smart cards using RFID technology.

The purpose of the development is the software of the cyber security control and access control system using smart cards using RFID technology.

The result of the work is the software implementation of the cyber security control and access control system using smart cards using RFID technology.

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 RAD Studio Delphi 10 environment.

Keywords: cybersecurity, access control and management, smart cards, RFID

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. Усі підприємства, організації та громадські будівлі мають певну форму контролю над тим, хто може отримати доступ до їхніх приміщень, території чи певних частин їхніх будівель.

Контроль керування доступом йде рука об руку з базовою безпекою, за допомогою якої організація повинна захистити свій персонал, своє майно та будь-які обмежені чи безпечні частини приміщень від несанкціонованого доступу. Таким чином, цілями управління доступом є:

– запобігання несанкціонованому доступу та полегшення авторизованого доступу;

– увімкнути моніторинг доступу та виходу та надати обліковий запис того, хто перебуває на сайті;

– не допускати пронесення заборонених предметів та вилучення майна.

У більшості комерційних приміщень контроль доступу є важливою частиною систем безпеки. Персонал повинен мати доступ до тих частин будівель, які їм дозволено входити. З іншого боку, відвідувачам і підрядникам потрібно буде запобігти необмеженому проникненню за допомогою певної форми контролю доступу, як правило, замкнених дверей або шлагбаума, через який вони не можуть пройти до отримання дозволу, наприклад, турнікет. У зонах рецепції з персоналом функція «сторожового» зазвичай виконується охороною або персоналом рецепції, який запитає відвідувача про його справи та дотримується встановленого протоколу, щоб дозволити їм доступ. У зонах входу без персоналу цю функцію можна виконувати через домофон або електронний контроль доступу.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID.

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

– Огляд існуючих систем контролю та керування доступом з використанням смарт-карт за технологією RFID.

– Дослідження системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID.

– Програмна реалізація системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі контролю та керування доступом з використанням смарт-карт за технологією RFID.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID, 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

Технологія RFID

Технологія RFID, що означає радіочастотну ідентифікацію, використовує радіохвилі для передачі та отримання інформації між RFID-мітками та зчитувачами. Мітки складаються з мікрочіпа та антени, тоді як зчитувачі є пристроями, які фіксують інформацію з тегів. Ця технологія зробила революцію в різних галузях промисловості та пропонує ряд переваг перед традиційними системами штрих-кодів.

Технологія RFID працює шляхом випромінювання радіохвиль від міток, які потім сприймаються зчитувачами в певному діапазоні. Потім зчитувачі передають зібрані дані в центральну систему, де вони обробляються та стають доступними для відстеження в реальному часі. На відміну від штрих-кодів, RFID-мітки не вимагають прямої видимості або ручного сканування. Це забезпечує швидкий і точний збір даних, що робить технологію RFID ідеальною для відстеження запасів.

Як працює технологія RFID?

RFID-мітки складаються з мікрочіпа та антени. Мікрочіп зберігає унікальний ідентифікаційний номер та інші відповідні дані, а антена дозволяє мітці передавати та приймати радіохвилі. Коли зчитувач наближається до RFID-мітки, він випромінює радіохвилі, які живлять мікрочіп на мітці. Потім тег надсилає збережену інформацію назад до зчитувача, який збирає та обробляє дані.

Зібрані дані потім передаються в центральну систему, де їх можна аналізувати та використовувати для різних цілей, таких як управління запасами, оптимізація ланцюга постачання та відстеження активів. Центральна система

може забезпечувати бачення в реальному часі розташування та статусу позначених елементів, дозволяючи компаніям приймати обґрунтовані рішення та підвищувати ефективність роботи.

Перезаписувані ідентифікатори

Для ухвалення рішення про допуск людини в приміщення або для підрахунку кількості коробок на піддоні досить, щоб кожний ідентифікатор мав свій унікальний номер. Однак є великий клас завдань, коли в мітку необхідно поміщати додаткову інформацію, що відбиває хід технологічного процесу.

У цьому випадку використовують перезаписувані ідентифікатори з додатковою енергонезалежною пам'яттю, у якій інформація зберігається й після провалля живлення. Обсяг такої пам'яті може коливатися від декількох десятків біт до десятків кілобайт, залежно від прикладного завдання.

Частотні діапазони й стандарти

У технології RFID є два ключових визначення:

– proximity карти й брелоки – ідентифікатори малої дальності, як правило близько 10 см. Використовуються в системах доступу, транспортних додатках;

– vicinity – ідентифікатори середньої дальності (біля півтора метрів). Використовуються для ідентифікації товарів і продукції в основному в логістичних додатках. З погляду робочих частот основними є низькочастотний діапазон (125 або 134 кГц), середньочастотний (13,56 Мгц) і високочастотний (800 Мгц... 2,45 Ггц).

Низькочастотний діапазон найбільш популярний у системах доступу, а також використовується для ідентифікації тваринних і металевих предметів (наприклад, пивних кегів).

У цей час найбільш популярний середньочастотний діапазон. Він використовується в транспортному й іншому аналогічному додатках, де потрібна робота з перезаписуваними картами. Базовим стандартом є ISO 14443, і практично всі смарт карти виробляються відповідно до цього стандарту.

Для міток у среднечастотному діапазоні актуальні два стандарти: ISO 15693 і EPC.

По ISO 15693 в основному виробляються перезаписувані мітки з досить широкою функціональністю. EPC (electronic product code) має більше просту структуру і є електронним аналогом штрихових кодів.

Високочастотний діапазон (800 Мгц...2,45 Ггц) почав освоюватися порівняно недавно, але становить великий інтерес через те, що при існуючих нормах на рівень потужності випромінювання в даному діапазоні на пасивних ідентифікаторах досягаються дальності до 4...8 метрів, що дуже важливо, наприклад, для складських додатків. У цьому діапазоні домінують два стандарти: ISO 18000 і EPC.

На сьогоднішній день можна затверджувати, що стандарт EPC для середньочастотного й високочастотного діапазонів є дуже перспективним, особливо для логістичних додатків.

1.2 Область застосування

Застосування RFID

Чому RFID технологія завойовує усе більше й більше широкі ринки, впроваджуючись у всілякі області діяльності, де потрібно швидка й надійна ідентифікація предметів? Тому, що має цілий ряд переваг, зокрема:

– RFID міткам не потрібний контакт або пряма видимість;

– RFID мітки читаються швидко й точно (наближаючись до 100%-ний ідентифікації);

– RFID може використовуватися навіть в агресивних середовищах, а RFID-мітки можуть читатися через бруд, фарбу, пару, воду, пластмасу, деревину;

– пасивні RFID-мітки мають фактично необмежений строк експлуатації;

– RFID-мітки несуть велику кількість інформації й можуть бути інтелектуальні;

– RFID-мітки практично неможливо підробити;

– RFID-мітки можуть бути не тільки для читання, але й із записом досить великого обсягу інформації.

Програми лояльності

Застосування RFID-технологій у роздрібній торгівлі дозволяє перейти від використання програмних комплексів управління взаєминами із клієнтами заснованих на централізованій базі даних до систем з розподіленими базами даних, причому локальні клієнтські бази даних зберігаються в енергонезалежній пам'яті RFID-карт. При використанні RFID карт для створення систем лояльності в мережі торговельних площадок, розташованих у різних частинах міста або в різних містах, немає необхідності зв'язувати всі ці площадки потужними корпоративними обчислювальними мережами. Це стає можливим оскільки всі рішення по управлінню взаєминами із клієнтами на кожній площадці приймаються не по даним єдиної централізованої базии клієнтів, а по локальних базах, що зберігається в RFID-картах. Таке рішення не тільки принципово здешевляє створення систем лояльності для таких об'єктів, але й уможливлює їхнє застосування там, де з технічних причин високопродуктивну корпоративну мережу не можна встановити. А сьогодні в Україні це майже всі торговельні мережі із площадками в різних населених пунктах і більшість мереж, торговельні площадки яких розкидані по території одного населеного пункту.

Транспортні додатки

У транспортних додатках основне місце (близько 80%) займають карти Mifare® виробництва Philips Semiconductors. Зокрема, вони використовуються в Київському метрополітені, у приміських поїздах і в ряді інших додатків. Карти відповідають третьому рівню ISO 14443 A і доповнені власним механізмом криптозахисту, що виключає підробку транспортних карт аматорами покататися за чужий рахунок. Ці ж карти використовуються в мережах автозаправних станцій, у клубних системах і в безлічі інших додатків, де незамінні безконтактна технологія й потрібна захист від несанкціонованого використання.

Логістика й склад

У даних додатках працюють ідентифікатори двох стандартів середньочастотного діапазону (ISO 15693 і EPC), а також ідентифікатори високочастотного діапазону по стандарті ISO 18000.

Необхідність появи стандарту EPC (electronic product code) викликана тими обставинами, що, по-перше, перезаписувані мітки по ISO 15693 нерентабельні в тих додатках, де потрібно тільки позначити товар, а, по-друге, при їхньому використанні порушується принцип приватності, що було причиною декількох скандальних розглядів. EPC аналогічний штриховому коду (по форматі даних), а функція деактивації мітки дозволяє руйнувати її в момент, коли потреба в ній відпадає.

Мітки високочастотного діапазону (800 Мгц... 2,45 Ггц) забезпечують максимальну дальність запису й читання (до 8...10 метрів), що незамінно при впровадженні технології RFID у процеси управління складськими запасами.

Електронні документи

Це зовсім новий, але дуже перспективний напрямок використання технології RFID. Швидкість зчитування й надійність, висока захищеність від несанкціонованого доступу дозволила почати впровадження електронних міток у паспорти, водійські посвідчення, авіаційні квитки й інші документи. У цей час у багатьох країнах у роботі перебувають проекти по перекладу внутрішніх паспортів на електронну основу. При цьому в пам'яті імплантованої в паспорт мітки будуть заноситися не тільки звичайні дані власника (ПІБ, рік народження й так далі), але й біометричні ознаки, а також кольорова цифрова фотографія.

Системи кібербезпеки контролю й управління доступом (СКУД)

Це історично саме старе застосування технології RFID. Зараз доступ в офіс або на підприємство по безконтактній пластиковій карті (proximity карта) став повсякденною справою. Перші рішення на основі технології proximity були відносно дорогими (якщо порівнювати з найбільш популярними тоді магнітними картами), однак зручність і надійність, забезпечувані RFID, дозволили за кілька

років практично витиснути з ринку професійних систем доступу всі конкуруючі технології. Основна маса карт і зчитувачів для систем доступу працюють у пасивному режимі в частотному діапазоні 125 кГц. Реально устояних стандартів немає, але найбільш популярні й поширені формати компаній EM Marin, HID і Motorola (Indala).

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID, 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

Система контролю й керування доступом TSS2000 Profi («Сім печаток»)

Програмна частина СКУД TSS2000 Profi – сукупність модулів, взаємодіючих по TCP/IP протоколу за принципом клієнт-сервер. Ядро системи реалізоване як служби Windows. СУБД – Firebird.

Кількість кодів ключів необмежено, (автономний режим – 65 025). Обсяг журналу подій необмежений (автономний режим – 250 000). Число маршрутів доступу необмежено. Число розкладів необмежено (автономний режим – 16).

СКУД TSS2000 Profi може бути побудована як локальна система із трьома способами підключення обладнання: безпосередньо до сервера СКУД, до серверів обладнання, через ЛОМ (число контрольованих пунктів проходу від 1 до 2032). Другий варіант побудови – розподілена система із синхронізацією баз даних необмеженого числа локальних СКУД (у тому числі по повільних лініях зв'язку), що дозволяє вести єдину систему реєстрації персоналу й формування звітів про робочий час (число контрольованих пунктів необмежено).

До складу системи (крім ядра й програм адміністрування) входять модулі «Бюро пропусків», «Прохідна», «Моніторинг роботи» (у текстовому й графічному виді), «Звіти». Додаткове ПЗ: «Створення й друк пропусків», «Облік робочого часу», «Реєстрація відвідувачів», сигнальна система, система відеоспостережень, інтеграція із зовнішніми системами безпеки, система імпорту-експорту даних.

IP-СКУД Tempo Reale (ВАТ НПП «Альфа-Прилад»)

Tempo Reale – це універсальна система, що базується на контролерах серії АПДА й призначена для керування доступом абонентів відповідно до призначених прав, автоматизованого обліку робочого часу й контролю за трудовою дисципліною.

Завдяки архітектурним особливостям СКУД Tempo Reale як IP-рішення забезпечується її надійність, висока пропускна здатність, швидке розгортання на об'єкті й легке масштабування.

Контролери серії АПДА можуть використовуватися як в автономному режимі, так і в складі мережний СКУД під керуванням ПЗ Tempo Reale. АПДА.21 і АПДА.41 розраховані на підключення відповідно до 2 і 4 зчитувачів з інтерфейсом Weigand і забезпечують керування виконавчими пристроями різних бар'єрів (дверей, воріт, турнікетів, шлюзових кабін і т.п.).

Завдяки наявності додаткових входів і виходів контролери АПДА дозволяють реалізувати найрізноманітніші режими доступу й алгоритми відповідних реакцій системи на виникаючі позаштатні ситуації (зв'язки подій).

Безсумнівною перевагою контролерів АПДА є наявність мережного порту Ethernet, що дозволяє будувати територіально розподілені системи з можливістю віддаленого моніторингу й керування пристроями, що входять до складу системи.

Особливого згадування заслуговує новий IP-контролер АПДА.21. Уперше для систем російського виробництва організація контролю повторного входу (anti-passback) і зв'язків подій здійснюється на рівні «контролер-контролер», що збільшує надійність і відказостійкість системи при втраті зв'язку із центральним сервером. Крім того, розроблювачі наділили АПДА.21 внутрішньою пам'яттю на 20 000 карт і 30 000 подій і можливістю живлення по Ethernet (PoS).

Програмна складова системи – ПЗ Tempo Reale – це клієнт-серверне рішення, що надає користувачеві можливості контролювати події в системі в режимі реального часу, прямо управляти обладнанням СКУД, автоматизувати облік робочого часу, включаючи побудову більше 20 видів звітів і уніфікованого табельного аркуша за формою Т-13.

На програмному рівні IP-СКУД Tempo Reale підтримує інтеграцію з IP-камерами AXIS, що дозволяє виконувати відеоверифікацію доступу абонентів у контрольовані зони й здійснювати запис відеофрагментів, що відповідають системним подіям.

Система контролю й керування доступом GATE («Равелін»)

СКУД GATE являє собою гнучкий, легко масштабований програмно-апаратний комплекс на базі контролерів GATE-4000, одного або декількох комп'ютерів зі спеціалізованим програмним забезпеченням, зчитувачів, виконавчих і зовнішніх пристроїв.

СКУД GATE була створена як універсальна система, що має різні області застосування: офіси в бізнес-центрах, промислові підприємства, навчальні заклади, паркування, їдальні великих підприємств, спортивні комплекси. Система GATE – конструктор, що має величезний потенціал і здатний вирішити завдання різного ступеня складності.

СКУД GATE проста й зручна (система будується на базі єдиного типового контролера), зберігає працездатність при відмові комп'ютера або обриві зв'язку. Гнучкість і масштабованість системи дає можливість вільно розширювати її при необхідності. Система може бути інтегрований («Інтелект», Essel, Laguna, 1С ін.), має європейський сертифікат якості CE.

Система складається з контролерів GATE (до 255 у лінії), комп'ютера зі спеціалізованим програмним забезпеченням, зчитувачів (HID, EM-Marine, Mifare, радіобрелоків), виконавчих пристроїв (замки, турнікети, шлагбауми й ін.)

Можливості GATE – це до 8144 користувачів і до 7 розкладів на кожну точку проходу, локальний буфер на 4095 подій у кожному контролері (при проваллі зв'язку події не губляться), збереження всіх подій у комп'ютері.

Система генерує звіти, має функцію безпосереднього керування з комп'ютера, веде моніторинг поточних подій системи в реальному часі. Є також функції видачі часових пропусків, підтримки фотоверифікації, організації віддалених робочих місць.

Програмне забезпечення GATE призначене для:

– Налаштування й конфігурування системи.

– Визначення контролерів, зчитувачів і розкладів.

– Роботи із пропусками, зміни прав доступу.

– Моніторингу в реальному часі.

– Контролю за переміщеннями.

– Читання подій з пам'яті контролерів і збереження їх у комп'ютері.

– Одержання звітів.

– Обліку робочого часу персоналу.

– Фотоверифікації.

– Контролю співробітників, що залишилися.

СКУД «Кронверк» (СКД)

СКУД «Кронверк» побудована з таким розрахунком, щоб задовольнити всім вимогам, висунутим до сучасної системи керування доступом.

СКУД «Кронверк» дозволяє:

– Підключення системних контролерів «Кронверк» у мережу Ethernet об'єкта.

– Визначення місця розташування співробітників і гостей у межах усього об'єкта.

– Автономну зміну режиму доступу для окремих приміщень.

– Організацію обліку робочого часу з формуванням стандартного табеля.

– Взаємодія із програмними додатками відділу кадрів підприємства – дані про співробітника, уведені у відділі кадрів, автоматично попадають у базу даних СКУД.

– Програмування автоматично реакції на події, що виникають у системі (автоматична зміна режиму роботи окремих точок доступу по події й за часом, виконання дій по командах від інших систем, розгорнутих на даному підприємстві).

– Інтеграцію із системами охоронно-пожежної сигналізації й системами охоронного телебачення.

– Синхронізацію баз даних з бізнесами-додатками.

Апаратну частину СКУД «Кронверк» представляють промислово системні контролери, що виготовляються. Особливостями системного контролера «Кронверк СМ-01 ісп. Е» є підключення прямо до комп'ютерної мережі Ethernet. На устаткування надається гарантія – 12місяців.

Програмне забезпечення «Кронверк Професіонал» перебуває у вільному доступі (в інтернеті або на дисках із супровідною документацією для системних контролерів СКУД «Кронверк»), кількість робочих місць тиражується відповідно до вимог замовника без додаткової оплати.

Інтегрована система ParsecNET 2.5 («РЕЛВЕСТ»)

Крім керування доступом система забезпечує підтримку функції охоронної сигналізації й інтеграцію із цифровим відеоспостереженням, що дозволяє забезпечити комплексний захист об'єкта без використання додаткових засобів. Система здатна забезпечити безпеку на об'єктах різного масштабу

Відмінною рисою нової версії ПЗ є практично повністю перероблене ядро системи. Всі зростаючі масштаби систем і їх розподіленість вимагають більше високих швидкостей і надійності в передачі й обробці інформації. Зміни торкнулися як механізмів передачі даних по локальній мережі, так і внутрішніх алгоритмів по обробці й реакції на події системи.

Перехід на клієнт-серверну СУБД MS SQL дозволяє забезпечити необхідну швидкість по обробці й поданню інформації (у першу чергу це стосується звітів і віддаленого доступу до бази дані системи). Для об'єднання територіально віддалених сегментів системи може використовуватися устаткування з Ethernet-інтерфейсом. Дані пристрої дозволяють без прокладки додаткових комунікацій поєднувати віддалені об'єкти, якщо між ними існує локальна мережа.

Номенклатура устаткування системи розширена новою лінійкою контролерів із кріпленням на DIN-рейку, що полегшує монтаж контролерів в офісних приміщеннях, де вже існують електричні шафи.

ПЗ ParsecNET 2.5 має потужну систему обліку робочого часу із правилами, що набудовуються гнучко, розрахунку відпрацьованого часу. При необхідності адаптації вихідних форм звітів під вимоги замовника в новій версії системи користувачеві наданий інструментарій, що дозволяє досить швидко й просто створювати будь-які власні форми для результуючих звітів. Разом із системою поставляється кілька типових форм, але, якщо вони замовника не влаштовують, він може створювати будь-яку кількість власних форм.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки 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 кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID.

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 кібербезпеки 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 кібербезпеки в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Опис технології RFID

Далека ідентифікація

Система далекої ідентифікації призначена для використання в додатках, де потрібно виявляти й відслідковувати постачені активними мітками об'єкти на більших відстанях від 5 – 7 до 50 метрів, залежно від типу використовуваних антен.

Прикладами таких додатків можуть бути:

– Автомобільні в'їзди, обладнані автоматичними воротами або шлагбаумами.

– Охоронювані автомобільні стоянки.

– В'їзди на платні автомобільні дороги.

– Системи моніторингу контейнерів на площадках для зберігання.

– А також інші аналогічні додатки.

Система ідентифікації використовує активні мітки з автономним батарейним живленням, які можуть кріпитися на об'єкти моніторингу або використовуватися як брелоки, що носяться на зв'язуванні ключів.

Читання міток здійснює зчитувач, що передає необхідну інформацію на хост-систему, у якості якої може бути контролер системи управління доступом, персональний комп'ютер або спеціалізований мікропроцесорний контролер.

Система працює в неліцензуємому діапазоні 2,45 Ггц, що, поряд з мінімальною випромінюваною потужністю, дозволяє використовувати неї без дозволу на виділення частотного діапазону.

Зчитувач виконаний двоканальним, що знижує питому вартість одного каналу зчитування в порівнянні з існуючими аналогами, а у випадку застосування

в системах управління доступом забезпечує рішення протиріч і колізій, що в аналогах принципово неможливо без використання додаткових апаратно-програмних засобів.

Відмінними рисами системи є:

– Унікальний механізм антиколізії, що дозволяє контролювати в полі зчитувача одночасно до сотні й більше міток.

– Наявність у зчитувачі одночасно двох каналів із двома виносними антенами, що дозволяє істотно знизити кінцеву вартість системи.

– Програмно регульована дальність зчитування роздільно по кожному з каналів.

– Повністю герметичне виконання корпуса зчитувача, що дозволяє використовувати його у вуличних умовах.

– Наявність убудованої програмувальної логіки відпрацьовування логіки проїзду автомобіля при використанні в системах управління доступом.

– Додаткові входи зчитувача для підключення датчиків автоматики воріт або шлагбаумів.

– Комбіновані мітки (активний плюс пасивний), що забезпечують одночасне використання мітки як для далекої ідентифікації, так і для звичайного доступу в приміщення.

– Три варіанти кріплення мітки для різних варіантів інсталяції.

– Можливість швидкої заміни батарейки мітки в процесі експлуатації, що вигідно відрізняє її від аналогічних пристроїв.

– Наявність енергонезалежного буфера транзакцій для моніторингу об'єктів у режимі

– Off-line. Для ідентифікації часу транзакцій зчитувач постачений убудованими годинниками реального часу.

Режим моніторингу

У режимі моніторингу зчитувач PR-G07 у комплекті з далекими мітками дозволяє вирішувати завдання по обліку наявності й переміщення

контрольованих об'єктів у реальному часі (з можливістю запису й зберігання одержуваної інформації), наприклад :

1. Контроль місцезнаходження шахтарів у шахті. По всій довжині шахти з періодичністю до 100 м установлюються зчитувачів PR-G07. Мітки монтуються в касках шахтарів і живляться від акумуляторів ліхтарів. У будь-який момент часу відоме місце розташування шахтарів у стовбурі шахти з точністю до 100 м.

2. Контроль особового складу в районі заданої точки (КПП, пост охорони й т.п.). Реєструється вхід/вихід носія мітки в реєструєму зону. У випадку покидання вартовим поста – генерується сигнал тривоги.

3. Контроль спортивного інвентарю (мотоцикли, картинги, велосипеди й т.п.). При недозволеному покиданні заданої області виникає сигнал тривоги.

4. Контроль виносу дорогого устаткування. При несанкціонованому виносі устаткування, на якому встановлена мітка, через прохідну – виникає сигнал тривоги.

5. Контроль і облік робочого часу службовців. Реєструються всі часові моменти, пов'язані з: приходом, відходом, знаходженням на території офісу службовцями.

6. Контроль переміщення вантажних візків по території аеропорту. Вирішується проблема оптимізації роботи служби перевезень вантажів і багажу в аеропорті.

Використання RFID у СКУД

Стандартні карти

Системи доступу – одне з перших дійсно масових застосувань технології RFID. Пояснюється це, видимо, двома факторами:

– простота реалізації самої технології стосовно до СКУД (досить використовувати ідентифікатори R/0 тільки для читання з невеликою – у три або чотири байти – довжиною коду);

– зручність в порівнянні з будь-якими іншими типами ідентифікаторів: контактних, з магнітною смугою, Wiegand (не плутати з форматом передачі коду зчитувачами).

Картка Proximity може бути прочитана зчитувачем навіть через гаманець і одночасно здатна відігравати роль пропуску з фотографією або бейджа. Крім того, у порівнянні з магнітними картами, що домінували до появи технології RFID, сьогоднішні карти Proximity мають більше високий рівень захищеності від копіювання й підробки. Можна сказати, що магнітні карти "вижили" тільки там, де вони реально забезпечують перевагу – наприклад, доступ у банкомати в нічний час. До речі, у цьому плані самою консервативною країною виявилася Америка – саме там дотепер збереглася найбільша кількість старих систем на магнітних картах.

Активні карти

Поява в СКУД технології Proximity (у дослівному перекладі "близький", "ближній") викликало природне бажання збільшити дальність зчитування коду карти. У результаті "народилися" системи Hands free ("вільні руки"). Але оскільки природу обдурити неможливо, для збільшення дальності довелося оснащувати ідентифікатори (карти) малогабаритною літієвою батарейкою. Зате тепер для живлення мікросхеми карти не була потрібна велика потужність випромінювання зчитувача, і дальність подібних систем перевищила один метр. Історично лідером тут стала англійська компанія Cotag. Її активні карти мають строк життя не менш п'яти років.

Технологія активних ідентифікаторів одержала подальший розвиток у системах упізнавання автомобілів. Такий ідентифікатор (уже не у вигляді карти, а виді невеликого блоку, що кріпиться до кузова автомобіля й одержує живлення від його бортової мережі) працював, як правило, зі зчитувачем, антена якого являла собою дротову петлю, що закривається в дорожнє полотно.

Нові діапазони

Ідея ідентифікації продовжила рух уперед у системах діапазону 2,5 ГГц. У цьому діапазоні лінійні розміри антен виходять досить маленькими, і зчитувач навіть розміром із взуттєву коробку здатний легко перебороти метровий бар'єр на пасивних ідентифікаторах, а при активному – типове значення максимальної дальності зчитування стає рівним 10 м. Прикладами таких систем, відомих у

Україні, можуть служити Tag-Master (Швеція) і Nedap (Голландія). При цьому варто пам'ятати, що перебувати тривалий час у поле роботи зчитувача великої дальності ЗВЧ-діапазону небезпечно (це стосується навіть потужності порядку 100 мВт).

Нові можливості

Поява перезаписуваних (R/W) карт відкриває стосовно до СКУД ряд нових можливостей. Це, наприклад, організація глобального антипассбек'а навіть при відсутності зв'язку між контролерами, що обслуговують різні точки проходу однієї його області.

Інші цікаві перспективи пов'язані з тим, що в карту нескладно записати біометричні характеристики людини (скажемо, відбитки пальців) і використовувати їх на об'єктах підвищеної таємності. Таке рішення просуває на ринку компанія BioScript.

У принципі на карту реально записати кольорову фотографію її власника й використовувати на точках проходу для відеоверифікації. На жаль, подальший розвиток потенціалу перезапису інформації, що зберігається в карті, обмежене тим, що на сьогоднішній день практично не випускається контролерів СКУД, що дозволяють безпосередньо працювати з картами R/W. Це стосується й використовуваного повсюдно протоколу обміну зі зчитувачами Wiegand, і структур баз даних контролера, і його логіки роботи. Все це обумовлено великий інерційністю в частині розробки технічних засобів безпеки – світові лідери індустрії СКУД не обновляють лінійки своїх продуктів по 10-15 років.

3.2 Розробка структурної схеми

Основні компоненти СКУД

Системи кібербезпеки контролю й управління доступом дозволяють здійснювати:

– обмеження доступу співробітників і відвідувачів об'єкта в охоронювані приміщення;

– часовий контроль переміщень співробітників і відвідувачів по об'єкту;

– контроль за діями охорони під час чергування;

– табельний облік робочого часу кожного співробітника;

– фіксацію часу приходу й відходу відвідувачів;

– часовий і персональний контроль відкриття внутрішніх приміщень (коли й ким відкриті);

– спільну роботу із системами охоронно-пожежної сигналізації й телевізійного відеоконтролю (при спрацьовуванні оповіщувачів блокуються або навпаки, наприклад, при пожежі розблокуються двері охоронюваного приміщення або включається відеокамера);

– реєстрацію й видачу інформації про спроби несанкціонованого проникнення в охоронюване приміщення.

СКУД звичайно складається з наступних основних компонентів:

– обладнання ідентифікації (ідентифікатори й зчитувачі);

– обладнання контролю й управління доступом (контролери);

– обладнання центрального управління (комп'ютери).

– обладнання виконавчого (замки, приводи дверей, шлагбаумів, турнікетів і т.).

Залежно від застосовуваної СКУД на об'єкті, окремі її пристрої можуть бути об'єднані в один блок (контролер зі зчитувачем) або взагалі бути відсутнім (персональний комп'ютер).

Обладнання ідентифікації доступу

Обладнання ідентифікації доступу (ідентифікатори й зчитувачі) зчитує й розшифровує інформацію, записану на ідентифікаторах різного типу й встановлює права людей, майна, транспорту на переміщення в охоронюваній зоні (об'єкті).

Контрольовані місця, де безпосередньо здійснюється контроль доступу, наприклад, двері, турнікет, кабіна проходу, обладнаються зчитувачем, виконавчим пристроєм і іншими необхідними засобами.

Ідентифікатор – предмет, у який (на який) за допомогою спеціальної технології занесена кодова інформація, що підтверджує права його власника й службовець для управління доступом в охоронювану зону. Ідентифікатори можуть бути виготовлені у вигляді карток, ключів і т.п.

Зчитувач – електронний пристрій, призначений для зчитування кодової інформації з ідентифікатора й перетворення її в стандартний формат, переданий для аналізу й ухвалення рішення у контролер.

У СКУД існує порядку десяти видів ідентифікаторів і зчитувачів, що використовують різні способи запису, зберігання й зчитування кодової інформації, що забезпечують різний рівень таємності й які мають ціни, що істотно відрізняються.

Найбільш широке поширення одержали наступні види ідентифікаторів і зчитувачів.

Картка перфорована – картка із двошарової недеформуємої пластмаси. Інформація записується на ній за допомогою пробивання спеціальних отворів один раз при виготовленні. Зчитування інформації здійснюється оптичним або механічним зчитувачами. Дана картка найпростіший і дешевий тип ідентифікатора, але який практично не забезпечує таємність коду й легко підробляється. Термін служби картки 1-2 роки. Вартість карток і механічного зчитувача досить низка: картка коштує ~0,5 у.о., зчитувач ~ 100 у.о. Механічний зчитувач дуже примхливий в експлуатації.

Картка зі штриховим кодом – картка з нанесеними на поверхню смугами іншого кольору, ніж інша поверхня, ширина й відстань між якими являють собою кодову послідовність. Кодова послідовність наноситься на картку при її виготовленні (звичайно вона визначається генератором випадкових чисел), і надалі не може бути змінена. Код зчитується оптичним зчитувачем

(інфрачервоним або лазерним). Найпоширеніші системи штрихового кодування, код 39 (3 з 9) і код 25 (2 з 5).

Оптичний зчитувач не містить частин, що рухаються, і при скануванні картки вона не контактує зі зчитувачем фізично. Тому зчитувач надійний у роботі й з успіхом може застосовуватися поза приміщеннями. Картка зі штриховим кодом може бути пропущена через зчитувач у будь-якому напрямку. Імовірність підробки картки така ж як і магнітних картках. Вартість картки й зчитувача досить низка: картка коштує ~ 0,5 у.о., зчитувач – ~ 100 у.о.

Картка магнітна – картка з магнітною смугою, на якій записаний код. Даний тип носія є самоочисним і не залишає окислів на зчитувачі. При бажанні код, записаний на доріжках магнітної смуги може бути легко перепрограмований, а при втраті картки можна швидко, дешево й без проблем закодувати нову картку. Код з картки зчитується магнітним зчитувачем, принцип роботи якого аналогічний зчитувачу звичайного магнітофона: інформація зчитується при переміщенні картки між магнітними головками зчитувача. Картки з магнітною смугою є дешевими, але не дуже надійними, тому що існує ймовірність їхньої підробки. До їхніх недоліків можна також віднести наявність механічного контакту при зчитуванні з головками зчитувача, що скорочує термін служби (середній термін служби 1 рік) і необхідність дуже акуратного обігу, пов'язаного з можливістю перекручування або знищення записаної інформації у відносно слабких магнітних полях і температур навколишнього повітря понад 80°С.

Розмір картки збігається із кредитними й банківськими картками, що дозволяє використовувати вже наявну в користувача картку (наприклад, кредитну) дня СКУД. При цьому із трьох магнітних доріжок одна використовується для банківської інформації, інша для СКУД і третя для будь-якої іншої інформації. Вартість карток і зчитувача досить низка: картка коштує 1 – 8 у.о., зчитувач залежно від типу 100 – 300 у.о.

Віганд-картка – картка з обрізками, що втримувалися усередині, тонких металевих дротиків, розташованих у певному порядку, що представляє собою

кодову комбінацію. Розташування дротиків на картці фіксується спеціальним клеєм, після цього переорієнтація дротиків не можлива. При переміщенні даної картки в магнітному полі зчитувача дротика створюють магнітний імпульс, що несе інформацію записану на картці. Такий тип карток не підданий впливу електромагнітних полів і високих температур навколишнього повітря. Підробка практично виключена. Зчитувачі можуть працювати поза приміщеннями, тому що всі їхні електронні компоненти залиті спеціальним захисним компаундом. Недоліком є те, що картки тендітні й можуть бути ушкоджені при вигині. Крім того, код кожної картки записується в неї при виготовленні й не може бути змінений. Вартість картки й зчитувача досить низка: картка коштує 3 – 8 у.о., зчитувач залежно від типу – 250 – 460 у.о. У цей час один із самих перспективних типів ідентифікаторів.

Картка безконтактна (Proximity) – картка, усередині якої розташована мікросхема (чип) із записаною в ній інформацією. Інформація з таких карток зчитується радіочастотним способом на відстані від 5 до 90 см. (для автомобільних ідентифікаторів даного типу відстань зчитування досягає 2 м). Картки діляться на активні й пасивні. У пасивних картках інформація записується один раз на увесь час дії картки, а в активні існує можливість зміни інформації в мікросхемі. Пасивні картки харчуються енергією, одержуваної від зчитувача, термін служби їх необмежений і вони не можуть бути підроблені. Активні – мають убудовані, незамінні батарейки, строк роботи якої звичайно досить великий – до 10 років. У надійності ці картки уступають Віганд-карткам, але вони більше зручні в застосуванні. Зчитувач може бути потай розміщений за не металевою стінкою. Ця технологія ідеально сполучить ефективний контроль із волею переміщення. Інформація з картки може бути зчитана, навіть якщо вона перебуває в гаманці або кишені. Недоліком є неможливість роботи при впливі сильних електромагнітних полів. Вартість пасивних карток становить 2 – 10 у.о., зчитувача залежно від типу 800 – 3400 у.о. Вартість активних карток приблизно в 5-10 разів дорожче пасивних. Ця картка незамінна для випадків, коли необхідно

забезпечити високу пропускну здатність, скритність місця установки зчитувача або дистанційний контроль доступу.

Електронні ключі "Touch Memory" виконані у вигляді брелоків. Всі необхідні дані записуються на укладену в них мікросхему. Запис, додавання або стирання ключа здійснюється майстер-ключем з контролера. Зчитується інформація при торканні ключем зчитувача. Мікросхема, як правило, харчується від вмонтованої в ключ батарейки. Строк її роботи досить великий – кілька років, але рано або пізно ключ підлягає заміні. Ключ дуже надійний у роботі, стійкий до механічних, електромагнітних впливів. Вартість ключа й зчитувача досить низка, ключ коштує 5 – 25 у.о., зчитувач залежно від типу – 400 – 1500 у.о. Широко застосовуються в невеликих СКУД, коли необхідно контролювати велику кількість дверей при малій кількості користувачів.

Крім перерахованих вище можуть використовуватися ідентифікатори наступних типів:

– з використанням цифрової клавіатури (PIN-код). Носієм інформації є користувач, що набирає на клавіатурі замка особистий код (умовне число) і, якщо він вірний, то одержуєте право доступу. Це найбільш простий і дешевий засіб контролю доступу, але яке легко обходиться. Хоча, останнім часом, з'явилися клавіатури, у яких після кожного натискання, змінюється порядок цифр на клавіатурі за випадковим законом, що виключає можливість "підглянути" порядок натискання кнопок або визначити найбільше часто використовувані кнопки;

– біометричні – зчитування індивідуальних фізичних ознак особистості (відбитки пальців, рисунок долоні, голос і т.д.). Основна перевага біометричного контролю – це повне рішення завдання контролю доступу – ідентифікується особистість людини, а не який-небудь предмет (картка). Через дуже високу вартість, малої оперативності й великого обсягу машинної пам'яті, займаної одним таким "зліпком ключа" вони застосовуються надзвичайно рідко, в основному в установах з підвищеною таємністю. Для підвищення швидкодії

біометричного контролю, як мінімум на порядок, разом з ним використовується будь-який інший спосіб ідентифікації.

Вартість зчитувача залежно від типу – 2000 – 7000 у.о.

Пристрої контролю й управління доступам

Контролери – електронні пристрої, що контролюють роботу зчитувачів і управляють пристроями виконавчими.

Контролери бувають однофункціональними й багатофункціональними.

Зв'язок контролерів між собою в єдину мережу здійснюється через стандартний інтерфейс RS-485. Для зв'язку провідного контролера з комп'ютером використовується стандартний інтерфейс RS 232. Багатофункціональні контролери працюють в основному в мережному режимі (централізований контроль і управління доступом).

Вартість контролерів залежно від фірми виготовлювача, номенклатури й комплекту поставки може коливатися в широких межах 800 – 3000 у.о.

Пристрій центрального управління

Персональний комп'ютер призначений для програмування СКУД, одержання інформації про користувачів системи, даті й часу проходу користувачів через контрольні пристрої, спрацьовуванні засобів охоронно-пожежної сигналізації (ОПС), відеоконтролю, спроб, несанкціонованого проходу, аварійних ситуаціям і т.п.

Для роботи в СКУД може використовуватися будь-який персональний IBM-сумісний комп'ютер. Поряд з роботою в складі СКУД він може виконувати й інші функція, тому що комп'ютер потрібний в основному лише для програмування системи й одержання звітів про роботу системи. Персональний комп'ютер, використовуючи спеціально розроблене для охоронюваного об'єкта програмне забезпечення (бажано українізоване), здійснює загальне управління й програмування СКУД, збирає інформацію з контролерів, створює загальний банк даних, формує різні звіти й зведення. Українізоване програмне забезпечення під Windows дозволяє здійснювати автоматичний запис даних по всіх операціях

входу/виходу. У будь-який момент можна запросити різноманітні відомості, наприклад, про місцезнаходження співробітників і відвідувачів. Поточний стан СКУД відображається в зручній графічній формі. У комп'ютер уводиться план охоронюваного об'єкта, на якому стандартними значками вказуються зчитувачі, замки, технічні засоби охоронно-пожежної сигналізації, відеоконтролю й т.п. На плані система автоматично в реальному масштабі часу показує стан всіх нанесених об'єктів контролю – відкрита або закрита двері, який саме оповіщувач спрацював у випадку тривоги. Таким чином, у будь-який момент часу можна швидко оцінити ситуацію й у випадку позаштатної ситуації оперативно й ефективно вжити заходів обережності.

Виконавчі пристрої

Виконавчі пристрої приймають команди управління з контролерів і забезпечують блокування можливих шляхів несанкціонованого проникнення через пристрої загородження (двері, ворота, турнікети, кабіни проходу й т.п.) людей, майна, транспорту в приміщення, будинки й на територію.

У виконавчих пристроях застосовуються виконавчі механізми електромеханічного й електромагнітного принципу дії.

Електромеханічний принцип дії виконавчого механізму заснований на переміщенні закриваючих елементів (засувки, ригелів замків і т.п.) за допомогою включення на час їхнього пересування електромотора або електромагніта.

У виконавчих механізмах з електромагнітним принципом дії відсутні закриваючі елементи, що рухаються механічні, тобто блокування пристроїв загородження, наприклад дверей, здійснюється за допомогою сил магнітного притягання, створюваних потужним магнітом.

Часто в виконавчих пристроях застосовується електромагнітне блокування (магнітні засувки, засувки й т.п.) закриваючих елементів з можливістю переміщення їх вручну при відкриванні або закриванні в екстремальних умовах.

Для повернення пристроїв загородження в закритий стан, вони доустатковуються спеціальними пристроями доведення, без яких СКУД

втрачають свою основну функцію – обмеження доступу, тому що без них пристрій загородження може перебувати в будь-якому стані. По виду виконавчого механізму пристрої доведення підрозділяються на пружинні, пневматичні, гідравлічні й електромеханічні.

Функція доведення – не тільки гарантувати закриття пристрою загородження (наприклад, двері), але й оберігати замок від механічних ударів, а при пожежі автоматично розкривати двері й допомагати евакуації. У деяких типах доведення використовується, так звана "система гальмування з підтягом" – з початку пристрій доведення дає розігнатися, потім гальмує рух і вже наприкінці, у самій дверній коробці, різко підтягує двері, забезпечуючи гарантоване її закриття. Крім того деякі пристрої доведення можуть мати убудований режим безпеки.

Вибір СКУД для устаткування об'єкта

Обстеження об'єкта

Вибір варіанта устаткування об'єкта засобами СКУД варто починати з його обстеження. При обстеженні визначаються характеристики значимості приміщень об'єкта, його будівельні й архітектурно-планувальні рішення, умови експлуатації, режими роботи, обмеження або, навпаки, розширення права доступу окремих співробітників, параметри встановлених (або передбачуваних до установки на даному об'єкті) пристроїв, що входять у СКУД. За результатами обстеження визначаються тактичні характеристики й структура СКУД, технічні характеристики її компонентів, а також складається технічне завдання на устаткування об'єкта СКУД.

У технічному завданні вказується:

– призначення системи, технічне обґрунтування й опис системи;

– розміщення складових частин системи;

– умови експлуатації складових частин системи;

– основні технічні характеристики такі як:

– пропускна здатність в охоронювані зони особливо в годину-пік;

– максимально можливе число користувачів на один зчитувач;

– максимальне число й види карток-пропусків;

– вимоги до маскування й захисту складових частин СКУД від вандалізму;

– оповіщення про тривожні й аварійні ситуації й вживання відповідних заходів по їхньому припиненню або попередженню;

– можливість роботи й збереження даних без комп'ютера або при його відмові;

– програмне забезпечення системи;

– вимоги до безпеки;

– вимоги до електроживлення;

– обслуговування й ремонт системи;

– вимоги до можливості включення системи СКУД в інтегровану систему безпеки.

Архітектурно-планувальні й будівельні рішення

Шляхом вивчення креслень, обходу й огляду об'єкта, а також проведення необхідних вимірів визначаються:

– кількість входів/виходів і їхні геометричні розміри (площа, лінійні розміри, пропускна здатність і т.п.);

– матеріал будівельних конструкцій;

– кількість окремо вартих будинків, їхня поверховість;

– кількість відкритих площадок;

– кількість опалювальних і неопалюваних приміщень і їхнє розташування.

Умови експлуатації

Ураховувати шкідливий вплив навколишнього середовища треба лише для виконавчих пристроїв, зчитувачів і контролерів (сполучених зі зчитувачами в одному конструктивному блоці), призначених для роботи поза опалювальними закритими приміщеннями або в особливих умовах (запиленість, підвищена вологість, негативна температура й т.п.). Для надійної роботи СКУД на об'єкті необхідно враховувати вплив електромагнітних перешкод, перепади напруги

живлення, далекість зчитувачів і контролерів від керуючого центра, заземлення складових частин системи й т.п.

Інтегровані системи охорони (ІСО)

У цей час будь-який великий і особливий важливий об'єкт має весь набір технічних засобів безпеки, що включає в себе системи ОПС, ТСВ, СКУД і т.і. Різноманіття й розрізненість цих систем на одному об'єкті приводить до неефективності їхньої роботи, труднощах у управлінні й обслуговуванні. Об'єднання всіх систем у єдиний програмно-апаратний комплекс (або іншими словами створення ІСО із загальним інформаційним середовищем і єдиною базою даних) дозволяє:

– мінімізувати капітальні витрати на оснащення об'єкта. Апаратна частина значно скорочується як за рахунок виключення дублюючої апаратури в різних системах, так і через збільшення ефективності роботи кожної системи;

– на основі повної й об'єктивної інформації, що надходить операторові значно скорочується час, необхідне на прийняття відповідних рішень по припиненню несанкціонованого проникнення, проходу й іншим надзвичайним ситуаціям на об'єкті;

– оптимізувати необхідне число постів охорони й істотно знизити витрати на їхній зміст, а також зменшити вплив суб'єктивного людського фактора;

– чітко розмежувати права доступу як своїх співробітників, так і сторонніх в охоронювані приміщення й до одержання інформації;

– автоматизувати процеси узяття, зняття охоронюваних приміщень, включення телевізійних камер, контролю шлейфів охоронно-пожежної сигналізації й т.п.

При створенні ІСО варто враховувати:

– можливість спільної синхронізації всіх складових ІСО пристроїв;

– можливість інтеграції на програмному, апаратному й релейному рівнях;

– можливість організації ліній зв'язку стандартних інтерфейсів RS-485 і RS 232 (при значній далекості панелей систем сигналізації й управління доступом);

– стан виходів тривоги засобів сигналізації й управління доступом у різних режимах, тому що вітчизняні й більшість закордонних засобів охоронної сигналізації мають у черговому режимі на виході замкнуті контакти, які розмикаються при тривозі.

Типи інтерфейсів, що рекомендуються:

– між контролерами – RS-485;

– між контролерами й керуючим комп'ютером – RS 232.

СКУД для автономного режиму роботи

СКУД 1-го й 2-го класів, що працюють в автономному режимі, звичайно обладнаються: квартири, котеджі, невеликі офіси, магазини, аптеки, готелі й т.п. і мало значимі зони на важливих об'єктах. Це дозволяє раціонально зменшити число каналів, що обслуговуються дорогими СКУД 3-го й 4-го класів. Дані СКУД це невеликі й недорогі системи, що обслуговують, як правило, до 8-мі пристроїв загородження (дверей, воріт, турнікетів і т.п.). СКУД 1-го й 2-го класів можна застосовувати й на важливих об'єктах або приміщеннях, якщо необхідний рівень безпеки забезпечується системами охоронної сигналізації й відеоконтролю.

У системі можна встановлювати, так званий, офісний режим Його зміст полягає в тому, що користувач відкриває закритий замок за допомогою ідентифікатора й проходить у приміщення. Далі зовні відкривати замок можна вільно, простим натисканням ручки. Цей режим установлюється за бажанням користувача, наприклад, для того, щоб щораз не підходити до дверям (не натискати кнопку автоматичного відкривання двері) і відкривати її зсередини, коли стукаються відвідувачі.

При реалізації даного варіанта на об'єкті рекомендується:

– використовувати системи, що мають міцний металевий корпус, кодонаборну клавіатуру з металевими кнопками, убудовану індикацію режимів роботи, антисаботажну захист для запобігання навмисного злому корпуса контролера й зчитувача;

– використовувати системи які мають енергонезалежну пам'ять і що дозволяють зберігати дані тривалий час;

– використовувати системи що дозволяють змінювати час розблокування дверей;

– програмування системи здійснювати за допомогою майстер-картки й клавіатури.

Даний состав СКУД може варіюватися в широких межах і в мінімумі складатися з одного конструктивно закінченого блоку (у вигляді замка), у якому розміщені зчитувач, контролер, виконавчий пристрій (запор, ригель, засувка й т.п.), індикатори режимів роботи. При цьому СКУД працює в режимі звичайного замка, тобто при збігу кодів ідентифікатора й зчитувача запірний механізм спрацьовує й розблокує двері, дозволяючи через неї прохід.

У процесі розширення системи додатково може встановлюватися ще один зчитувач для контролю проходу у зворотну сторону (або організації багаторівневого контролю доступу), виносні світлові/звукові оповіщувачі, пристрою автоматичного відкривання/закривання двері й т.д.

У систему можуть бути уведені додаткові функції:

– контроль проходу у двох напрямках;

– автоматичне відкриття й закриття дверей при аварійних і тривожних ситуаціях;

– передача тривожних повідомлень на пост охорони;

– реєстрація подій, що відбуваються, за допомогою принтера, що підключається до контролера.

Програмування системи здійснюється як за допомогою майстер-картки й клавіатури, так і за допомогою переносного комп'ютера.

У своєму закінченому виді дану систему можна легко включити в СКУД, що працює в мережному режимі. Для цього необхідно використовувати контролер що дозволяє працювати в мережному режимі з іншими контролерами або

використовувати додатковий модуль зв'язку, що забезпечує об'єднання контролерів через інтерфейс RS-485.

СКУД для мережного режиму роботи

СКУД 3-го й 4-го класів призначені для устаткування великих об'єктів таких як банки, великі установи й фірми. Безсумнівним достоїнством цих систем є можливість практично не обмеженого розширення. Такі системи дозволяють обслуговувати десятки тисяч користувачів.

У відносно невеликих і недорогих системах 3-го класи використовується побудова системи СКУД, при якому в одну контрольовану лінію інтерфейсу RS-485 включаються всі контролери, а база даних завантажується в один керуючий контролер (майстер-контролер).

Така побудова забезпечує гнучкість вбудовування СКУД в інтер'єр приміщень, мінімізацію комунікаційних з'єднань і більші відстані між об'єктами управління.

Ефективність роботи СКУД 4-го класу, обумовлена можливістю створювати розгалужені, досить численні з'єднання контролерів і керуючих комп'ютерів у єдину систему. Модульність побудови даних систем забезпечує:

– гнучкість конфігурації;

– простоту монтажу, технічного обслуговування й ремонту;

– можливість розширення системи;

– цінову ефективність;

– легкість сполучення із пристроями сервісної автоматики (управління ліфтом, висвітленням, системами кондиціювання й т.д.).

З'єднання контролерів між собою й підключення контролера до різних периферійних пристроїв, що входять до складу системи забезпечується за допомогою різних модулів.

До одного контролера може бути підключене до 8 зчитувачів різного типу, наприклад, зчитувач магнітних карток, зчитувач безконтактних карток, клавіатура (кодонаборник) і т.і. Підключення зчитувачів здійснюється через відповідний

модуль, що зчитує, працюючий із двома пристроями, що зчитують. Крім зчитувачів, він також контролює датчики стану дверей і кнопки їхнього відкривання, інші допоміжні пристрої.

Інформація про стан інших зовнішніх пристроїв надходить у контролер через модуль входу/виходу. За допомогою цього ж модуля контролер управляє роботою виконавчих пристроїв, пристроєм видачі тривожних повідомлень.

Модуль зв'язку забезпечує об'єднання контролерів у єдину систему, довжиною до 1 км за допомогою інтерфейсу RS-485, а також при необхідності об'єднання контролерів і керуючого комп'ютера в комп'ютеризовану систему за допомогою інтерфейсу RS 232. Модуль приймально-передачі управляє роботою зчитувачів безконтактних карток (Proximity).

Один контролер може обслуговувати до 10000 користувачів. Для збільшення числа користувачів може застосовуватися модуль розширення пам'яті.

Системи 4-го класу звичайно будуються на базі таких же багатофункціональних контролерів, які використовуються для побудови СКУД 3-го класу, об'єднаних у єдину комп'ютерну мережу. При створенні комп'ютерної мережі контролери в кількості до 32 одиниць можуть бути об'єднані в одну гілку. У цьому випадку модуль зв'язку включається в перший один за одним контролер гілки. Через нього здійснюється зв'язок цього контролера з комп'ютером по інтерфейсі RS 232. Обмін інформацією між контролерами робить по інтерфейсу RS-485. Крім того, модуль зв'язку здійснює перетворення формату й швидкості передачі даних RS 232/ RS-485. Кожний контролер у гілки має свою адресу.

Подальше нарощування системи можливо шляхом організації декількох (до 10) гілок контролерів. Модуль зв'язку першого контролера перетворює з однієї сторони потік даних, що посилаються з керуючого комп'ютера на контролер, а з іншого боку – потік вихідних даних, паралельно подаваних на адресні модулі зв'язку в галузях. Кожний адресний модуль зв'язку обмінюється даними з контролерами в галузях і модулями зв'язку. Така розширена мережа дозволяє обслуговувати до 320 контролерів і 2048 контрольованих точок.

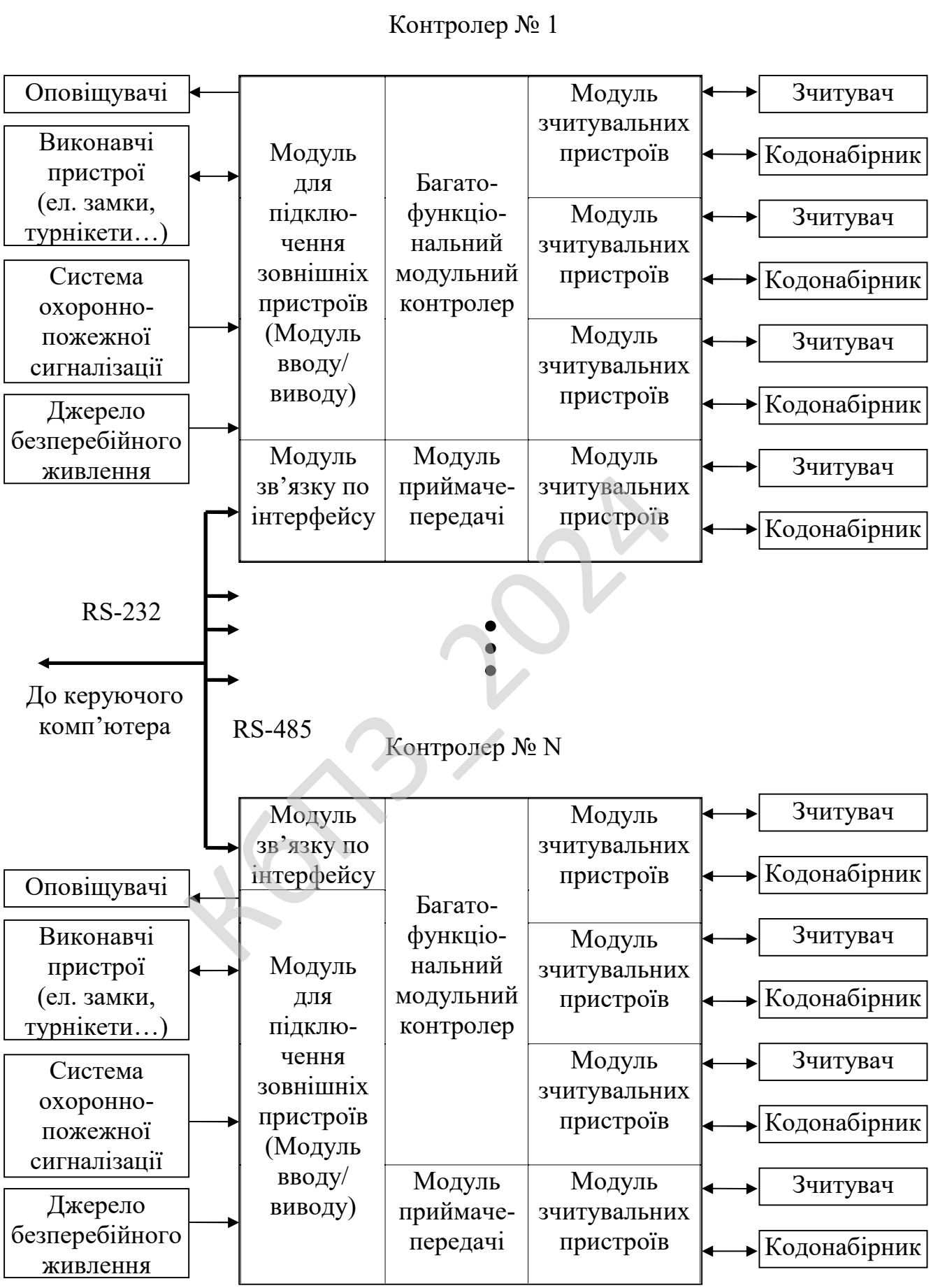

Рисунок 3.1 – Структурна схема системи

При необхідності гілку контролерів може бути збільшена ще на 1 км. Для цього подовжується гілка, що підключається до першого контролера нової гілки через модуль зв'язку. Для зв'язку між контролерами по колишньому використовується інтерфейс RS-485.

При організації комп'ютеризованих СКУД рекомендується застосовувати IBM-сумісні комп'ютери типу PENTIUM з ОЗУ 16 МБ і більше, двома послідовними портами й обсягом вінчестера не менш 528 МБ. Комп'ютери цього типу задовольняють потреби будь-якої системи й дозволяють модернізувати її в майбутньому.

Наявність описаних модулів багатофункціонального контролера створює більші можливості по управлінню різноманітною периферією системи. Як контрольовані точки можуть виступати головки, що зчитують, PIN-Клавіатури, замкнуті/розімкнуті контакти кнопок, реле, вихідні контакти різних об'ємних або поверхневих оповіщувачів. Як виконавчі пристрої можуть використовуватися електрозамки дверей, виконавчі пристрої шлагбаумів, турнікетів, пристрою тривожного оповіщення й висвітлення, телевізійні камери й т.д.

Логічний пристрій (процесор) контролера дозволяє робити необхідну установку параметрів доступу в кожній контрольній точці за допомогою програмного забезпечення, тобто конфігурувати систему. Системний програміст може задавати параметри (замкнуте/розімкнутий стан контактів реле або кнопок, стан і режим роботи лічильників, стан флатових регістрів, часові інтервали реєстраторів подій і т.д.) прямо із клавіатури комп'ютера. Це дає можливість реалізовувати різні варіанти організації контролю й управління доступом, гнучко міняючи їх відповідно до поточних вимог.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно. Програма надає більші 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 несанкціонованих проникненні (якщо потрібно – зі звуковим супроводом). На екран можуть виводитися численні повідомлення, наприклад, повні або короткі звіти про зареєстровані події з можливістю їхньої роздруківки на принтері.

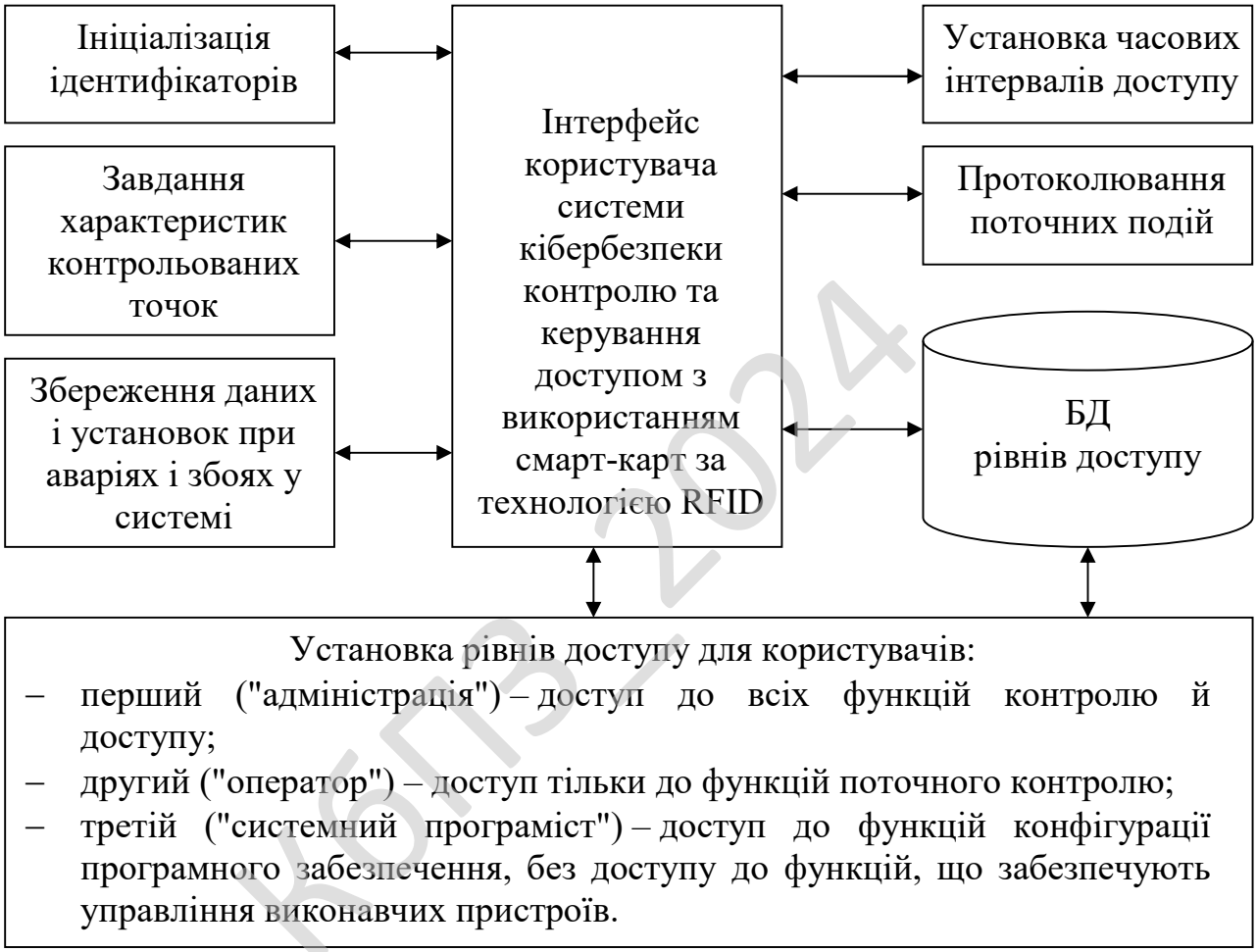

Рисунок 3.2 – Функціональна схема системи

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 СКУД повинне здійснюватися від мережі змінного струму частотою 50 Гц із номінальною напругою 220 В.

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

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

Електропостачання технічних засобів СКУД 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.

СКУД 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 СКУД.

СКУД 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 У 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 взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.3.

Рисунок 3.3 – Діаграма взаємодії процесів

Після початку роботи ми потрапляємо до основного блоку ПЗ далі після налаштування ПЗ ми потрапляємо до модулю захисту ПЗ через обробник помилок. Далі через інтерфейс адміністратора проводимо перевірка ступені захищеності, сканування ресурсів системи та можемо потрапити до модуль аналізу даних з якого проходе взаємодія зі службами та архівація даних.

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

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків.

Спершу відбувається ініціалізація ресурсів ПЗ, підключення додаткових модулів та підключення бібліотек взаємодії з апаратною частиною. Далі проходе спроба доступу до апаратної частини і якщо доступ є проводиться перехід у робочий режим та виведення головного вікна програми.

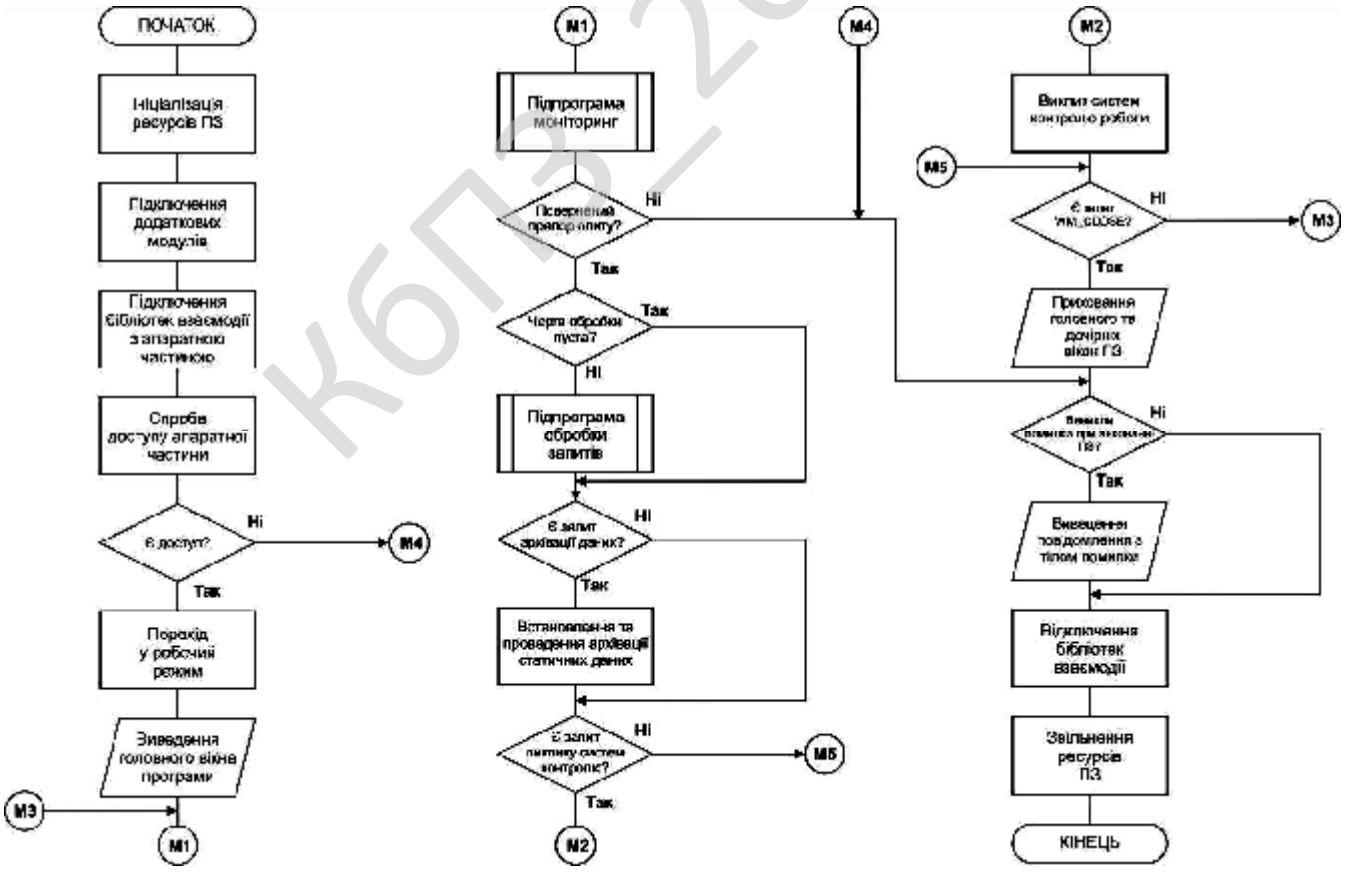

Рисунок 4.1 – Блок-схема основної програми

Далі викликається підпрограма моніторингу яка зображена на рисунку 4.2. в якій проводиться опит апаратної частини та повернення кодів опиту. Далі проходе перевірка поверненого прапору опиту з якого встановлюється чи пуста чи ні черга обробки. Якщо є черга викликається друга підпрограма – обробки запитів (рис. 4.2).

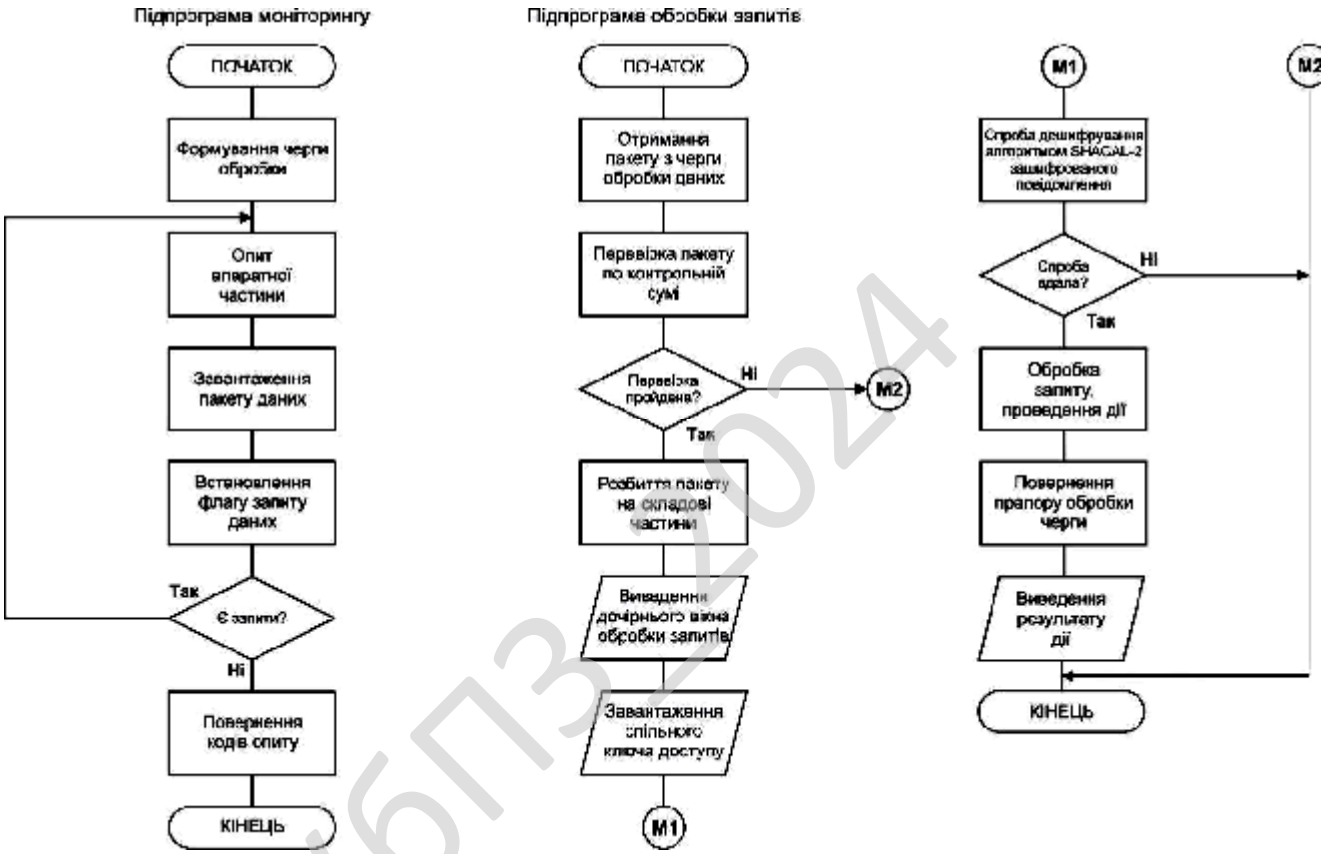

Рисунок 4.2 – Блок-схема роботи підпрограм

У підпрограмі обробки запитів зосереджено основний функціонал тобто проводиться обробка пакету з черги повідомлень, розбиття пакету на складові частини та проведення дії.

Далі при наявності запитів архівації даних та виклику систем контролю проводиться проведення архівації статичних даних та виклик систем контролю роботи.

Після проведення ітерації ПЗ і наявності запиту WM_CLOSE проходе завершення роботи ПЗ: приховання головного та дочірніх вікон ПЗ; виведення повідомлення з тілом помилки (якщо вони виникли); відключення бібліотек взаємодії; звільнення ресурсів ПЗ.

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. Для шифрування процедури CheckTrial необхідно виконати наступний код (результат шифрування виведемо в тестове вікно TMemo):

```
var
 ptrAddr: Pointer; {Адреса процедури CheckTrial}
 aByte: Byte; {Байт процедури}
 dwOldProtect: DWORD;
begin
 ptrAddr := @CheckTrial;
```

```
 VirtualProtect(@CheckTrial, 4096, PAGE_READWRITE, @dwOldProtect);
 while ptrAddr <> @EndCheckTrial do
 begin
  Byte(ptrAddr^) := Byte(ptrAddr^) xor $25;
  aByte := Byte(ptrAddr^);
  Memo1.Lines.Add('0' + IntToHex(aByte, 2) + 'h');
  inc(Integer(ptrAddr));
 end;
end;
```

Після виконання коду в TMemo з'явився текст процедури CheckTrial. Цей варіант процедури вставляється у вихідний код програми. Вийде наступний асемблерний код:

```
procedure Check_TrialMyDip;
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;
procedure EndCheckTrial; {Порожня процедура, що означає кінець CheckTrial}
begin
end;
```

Розглянемо процедуру обробки коду CheckTrialMyDip:

```
var
 ptrAddr: Pointer; {Адреса процедури CheckTrial}
 dwOldProtect: DWORD;
begin
 ptrAddr := @CheckTrial;
 VirtualProtect(@CheckTrial, 4096, PAGE_READWRITE, @dwOldProtect);
 while ptrAddr <> @EndCheckTrial do
 begin
  Byte(ptrAddr^) := Byte(ptrAddr^) xor $25;
  inc(Integer(ptrAddr));
 end;
CheckTrialMyDip;
end;
```

Описаний тут підхід зовсім прозорий для програміста. Застосування даного підходу в бакалаврському проекті набагато збільшив універсальність коду.

Але для повної працездатності розробленого бакалаврського коду, необхідно також подбати про зсув посилань у вихідному коді. Рядок `'Незареєстроване використання ПЗ!.'`, використовувана при виклику ShowMessage зберігається в пам'яті по певній адресі.

Фактично у функції ShowMessage використовується тільки ця адреса. Тепер при використанні зазначеної процедури й підставлені отриманого байтового коду між командами asm і end – відбувається зсув коду, і після компіляції адреси напевно зміняться.

У результаті по тій адресі буде лежати вже не рядок `'Незареєстроване використання ПЗ!'`, а що-небудь інше. Те ж саме може відбутися й з будь-якими іншими адресами в коді. У результаті процедура не працює – для подолання даних труднощів необхідно написати зовнішню міні програму, тобто спочатку створюється бакалаврське програмне забезпечення, у ній передбачається код Data (який до остаточного складання проекту закоментовано). Коли розробка завершена, код розкоментується й компілюється остаточний розроблений бакалаврський проект. Після цього запускається розроблена зовнішня міні програма, що у вже скомпільованому файлі (Exe – файлі) шифрує потрібні ділянки коду.

Після чого розроблена бакалаврська програма стає повністю функціональною й захищеною від зовнішніх атак. Для забезпечення правильної роботи коду у вихідний код програми я помістив мітки, які складаються з випадкової, неповторюваної комбінації:

```
procedure StartPointProc;
begin
 asm
  // неповторна в коді комбінація
  DB 000h, 034h, 0F4h, 02Ah, 000h, 03Dh, 0FFh, 0CAh
 end;
```

```
end;
procedure ProtectedProc;
begin
 ShowMessage('Незареєстроване використання ПЗ!.');
end;
procedure EndPointProc;
begin
 asm
  // неповторна в коді комбінація
  DB 001h, 035h, 0F5h, 02Bh, 001h, 03Eh, 000h, 0CBh
 end;
end;
```

Тут DB 000h, 034h, 0F4h, 02Ah, 000h, 03Dh, 0FFh, 0CAh і DB 001h, 035h, 0F5h, 02Bh, 001h, 03Eh, 000h, 0CBh являють собою мітки. Розроблена зовнішня міні програма шукає в exe – файлі ці послідовності байт. Самі мітки не шифруються, оскільки вони використовуються програмою для пошуку тої ділянки пам'яті, що повинен бути розшифрований.

Розроблена зовнішня міні програма дозволяється застосовувати різні методи. Далі наведено вихідний код функції, що шукає мітку у файлі розробленої програми:

```
function CheckFileLabel(FileName:string; strlabel:string;
                        var point:LongInt):boolean;
var
 F: file;
 currarray: array [1..8] of byte;
 lookarray: array [1..8] of byte;
 i: integer;
begin
 Result:= false;
 point:= -1;
 //Завантаження міток
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 міток
```

```
 AssignFile(F,FileName);
 FileMode := fmOpenRead;
 Reset(F,1);
 Seek(F,0);
 while not EOF(F) do
 begin
  for i := 1 to 7 do
   currarray[i] := currarray[i + 1];
  BlockRead(F,currarray[8],1);
  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 := FilePos(F) - 1;
   break;
  end;
 end;
 CloseFile(F);
end;
```

Функція повертає true, якщо мітка знайдена. Функція HexToByte також нестандартна розроблена під час бакалаврського проектування, реалізація її проста, вона переводить 16 число в байтове подання. У якості мітки можна передавати відразу строкову константу 'DB 000h, 034h, 0F4h, 02Ah, 000h, 03Dh, 0FFh, 0CAh'.

Для реалізації нешифруємої процедури необхідно написати наступний код:

```
exit;
 StartPointProc;
 EndPointProc;
end;
```

Розглянемо реалізацію CRC функції виклику процедури. Розглянемо реалізований код:

```
goto lab1;
asm
 DB 000h, 034h, 0F4h, 02Ah, 000h, 03Dh, 0FFh, 0CAh
end;
```

```
lab1:
< код >
goto lab2;
asm
 DB 001h, 035h, 0F5h, 02Bh, 001h, 03Eh, 000h, 0CBh
end;
lab2:
```

Як видно розроблений код, для якого обчислюється CRC перебуває між двома мітками. Зовнішня міні програма шукає їх і обчислює CRC коду.

Зовнішня міні програма шукає мітку DB 0CBh, 001h, 035h, 0F5h, 02Bh, 001h, 03Eh, 000h і в наступні 4 байти записує CRC.

Розроблена програма шукає в пам'яті мітки DB 000h, 034h, 0F4h, 02Ah, 000h, 03Dh, 0FFh, 0CAh і DB 001h, 035h, 0F5h, 02Bh, 001h, 03Eh, 000h, 0CBh, обчислює CRC коду між ними, шукає мітку DB 0CBh, 001h, 035h, 0F5h, 02Bh, 001h, 03Eh, 000h, зчитує правильний CRC з наступних 4 байт і порівнює отримані CRC.

У такий спосіб здійснюється перевірка того, модифікований код чи програми ні. У такий же спосіб можна зберігати в тілі програми й специфічну інформацію, що властиво й використовується для визначення легальності копії в розробленій бакалаврській програмі системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використати фіналіста конкурсу AES – шифр Rijndael. Він є нетрадиційним блоковим шифром, оскільки не використовує мережу Фейштеля для криптоперетворень. Алгоритм представляє кожний блок кодуємих даних у вигляді двовимірного масиву байт розміром 4x4, 4x6 або 4x8 залежно від установленої довжини блоку. Далі на відповідних етапах перетворення відбуваються або над незалежними стовпцями, або над незалежними рядками, або взагалі над окремими байтами в таблиці.

Всі перетворення в шифрі мають строге математичне обґрунтування. Сама структура й послідовність операцій дозволяють виконувати даний алгоритм ефективно як на 16-бітних так і на 64-бітних процесорах. У структурі алгоритму закладена можливість паралельного виконання деяких операцій, що на багатопроцесорних робочих станціях може ще підняти швидкість шифрування в 4 рази.

Алгоритм складається з деякої кількості раундів (від 10 до 14 – це залежить від розміру блоку й довжини ключа), у яких послідовно виконуються наступні операції :

ByteSub – Таблична підстановка 8x8 біт (рисунок 4.3).

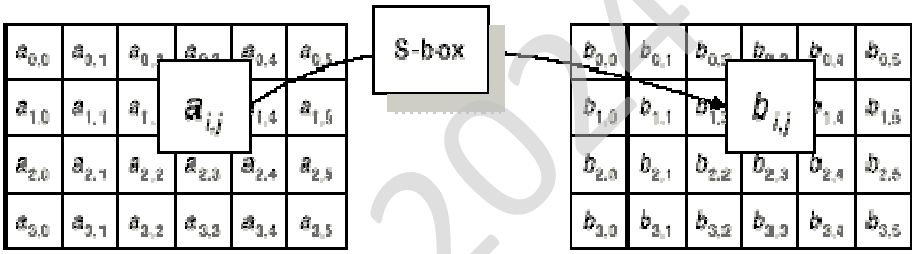

Рисунок 4.3 – Таблична підстановка 8x8 біт

ShiftRow – зрушення рядків у двовимірному масиві на різні зсуви (рисунок 4.4).

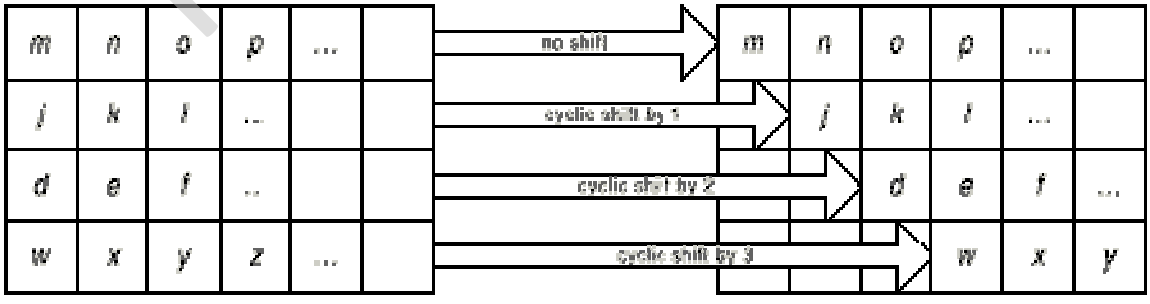

Рисунок 4.4 – Зрушення рядків у двовимірному масиві на різні зсуви

MixColumn – математичне перетворення, що перемішує дані усередині стовпця (рисунок 4.5).

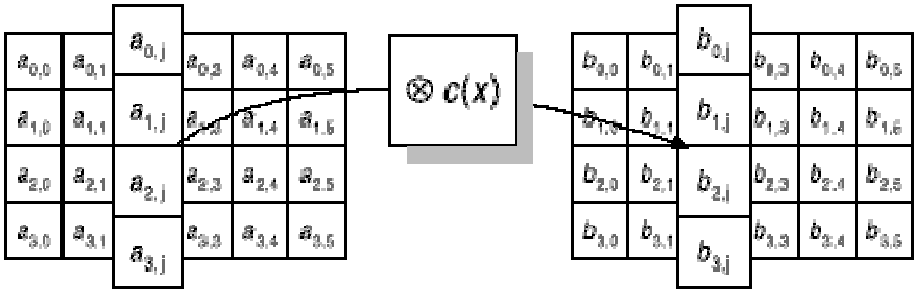

Рисунок 4.5 – Математичне перетворення, що перемішує дані усередині стовпця

AddRoundKey – додавання матеріалу ключа операцією XOR (рисунок 4.6).

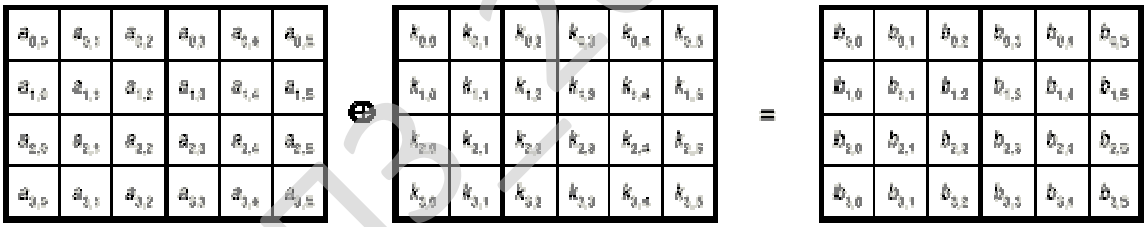

Рисунок 4.6 – Додавання матеріалу ключа операцією XOR

В останньому раунді операція перемішування стовпців відсутня, що робить всю послідовність операцій симетричною.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо роботу бакалаврського ПЗ. Головне вікно ПЗ зображено на рисунку 5.1.

З лівого боку вікна знаходиться блок який відображає всі наявні мітки RFID у вигляді дерево образної структури. Як приклад це можуть бути зали у магазині. Це дозволяє швидко проводити сортування міток.

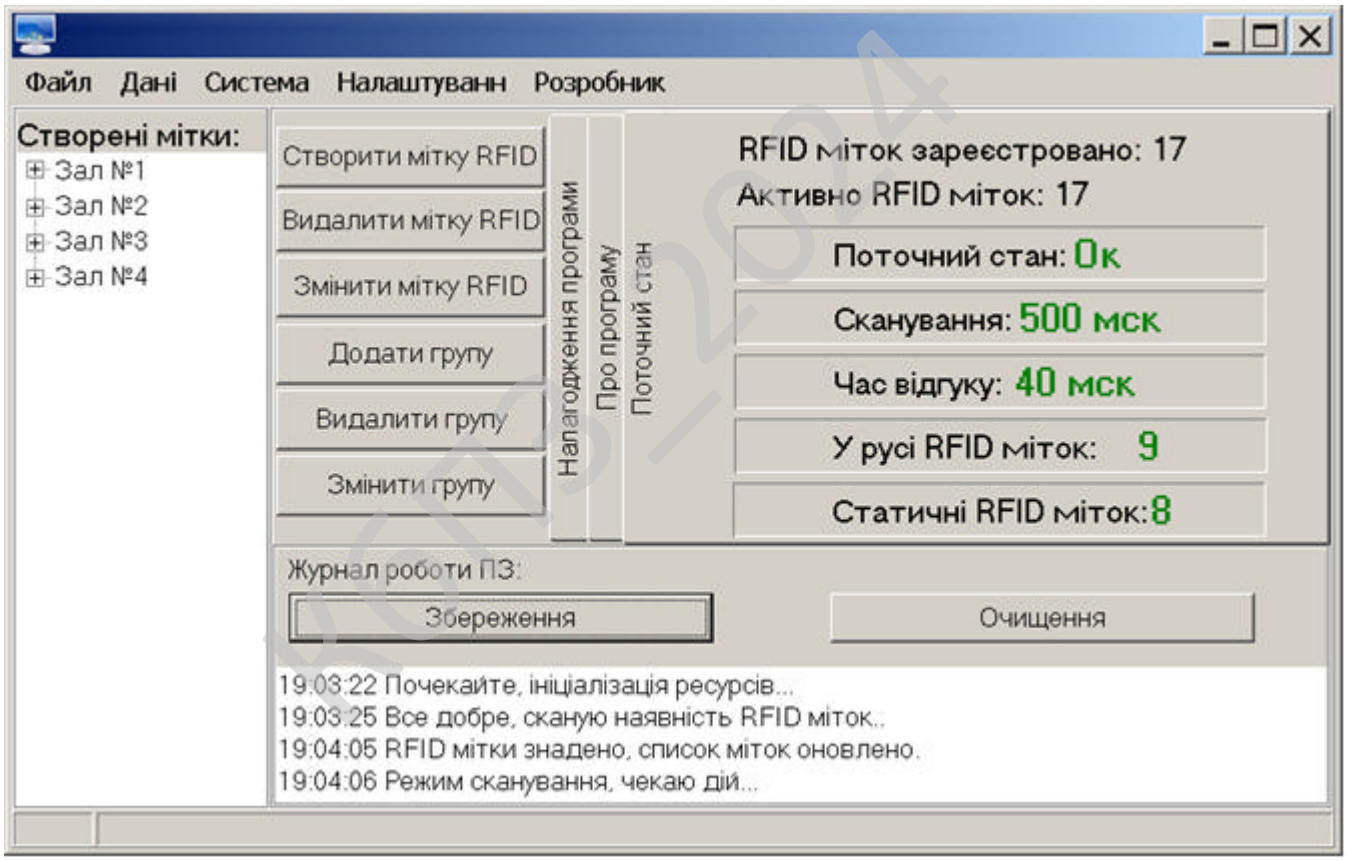

Рисунок 5.1 – Головне вікно ПЗ

По центру вікна знаходяться кнопки які відповідають за основний дії програми:

– Створити мітку RFID.

– Видалити мітку RFID.

– Змінити мітку RFID.

– Додати групу.

– Видалити групу.

– Змінити групу.

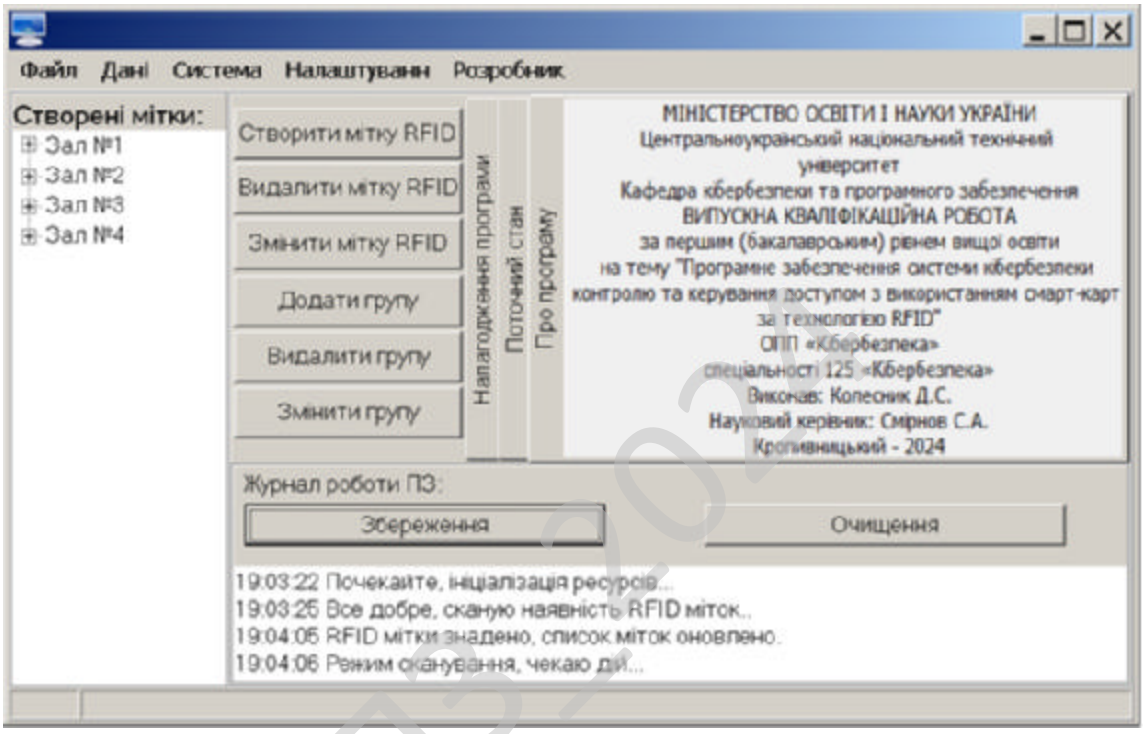

Рисунок 5.2 – Вікно розробника ПЗ

З правого боку зображені розділи:

– Поточний стан. Відповідає за відображення всіх активних міток RFID. З відображенням скільки міток зареєстровано, їх поточний стан, час за який проводиться сканування, час відгуку міток та скільки міток у русі і у статичному виді.

– Налагодження програми. Забезпечує доступ до налаштування як ПЗ так і інформації яка заноситься у RFID мітки.

– Про програму. Відображає дані автора (рисунок 5.2).

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем контролю та керування доступом з використанням смарт-карт за технологією RFID.

– Досліджена система контролю та керування доступом з використанням смарт-карт за технологією RFID.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання контролю та керування доступом з використанням смарт-карт за технологією RFID.

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 RAD Studio Delphi 10. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID. 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 кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм AES.

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. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

2. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

3. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

4. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

5. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

6. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

7. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

8. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

9. 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.

10. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

11. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

12. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile*

Networks. Lecture Notes on Data Engineering and Communications Technologies, vol 131. 2023. Springer, Singapore. pp. 21-34.

13. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

14. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418

15. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

16. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

17. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

18. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

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

20. 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.

21. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

22. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

23. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.

24. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

25. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

26. 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.

27. 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.

28. 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.

29. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

30. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

31. 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.

32. 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.

33. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

34. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and*

Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P.399-405.

35. 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.

36. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

37. 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.

38. 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.

39. 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.

40. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884.

41. Smirnov A.A., Kuznetsov 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.

42. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *II Міжнародна науково-практична Інтернет-конференція «Інновації та перспективні шляхи hозвитку інформаційних технологій (ІПШРІТ-2023)»* м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

43. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

44. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

45. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

46. 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.

47. 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.

48. 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.

49. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». *CEUR Workshop Proceedings* Volume 2732, 2020, Pages 214-227.

50. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Книшук А.В. «Вступ до кібербезпеки»: навчальний посібник – Кропивницький: ЦНТУ – 2022. – 968 с.

51. Теорія та практика сучасного інформаційно-психологічного протиборства: навчальний посібник / [В.М. Петрик, С.О. Гнатюк, М.М. Присяжнюк та ін.]; за заг. ред. С.О. Гнатюка, В.М. Петрика та О.А Смірнова. – Полтава, 2022. – 334 с.

52. 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.

53. 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.

54. 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.

55. 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.

56. Smirnov, O., Kuznetsov, A., Shekhanin, K., Chepurko, I. Detecting Hidden Information in FAT. 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. 412-429. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook)

57. 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).

58. 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.

59. Смірнов О.А., Стасєв Ю.В., Бараннік В.В. Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Інформаційна безпека держави. Підручник – Кіровоград: РВЛ КНТУ, 2016. – 263 с

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

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID.

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. № 135-02 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 кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID.

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 кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID;

– 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 RAD Studio Delphi 10.

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 – 73 аркушів.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

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 4.06.2024 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

_____ Смірнов С.А.

Програмне забезпечення системи кібербезпеки контролю та керування доступом з використанням смарт-карт за технологією RFID

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

// Колесник Дарина Сергіївна
// КБ-20
// 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;

// Колесник Дарина Сергіївна
// КБ-20
// 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

// Колесник Дарина Сергіївна
// КБ-20
// 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

// Колесник Дарина Сергіївна
// КБ-20
// 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.
```